**Computer Organization**

**Homework 1**

(32-bits Complete ALU)

**Student :**

B11107157 電機三乙 林明宏

**Teacher :**

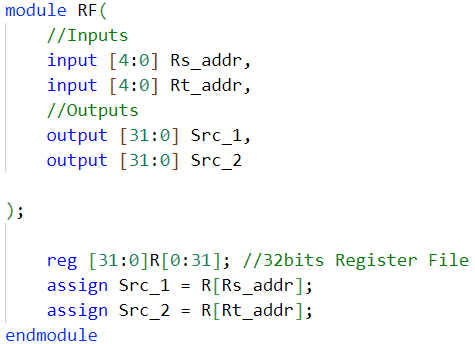
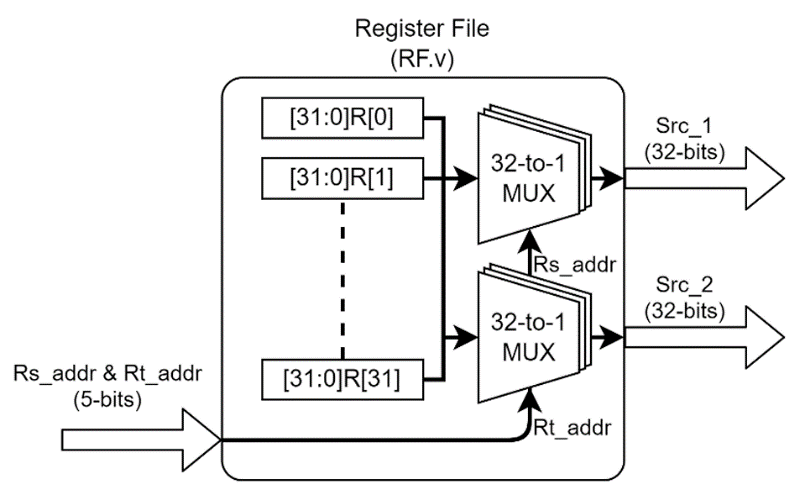
陳雅淑

**RTL Simulator :**

**ModelSim-Intel FPGA Standard Edition, Version 20.1.1, windows**



1. **Screenshot and descriptions of each module :**
2. **32-bits Read Only Register File :**

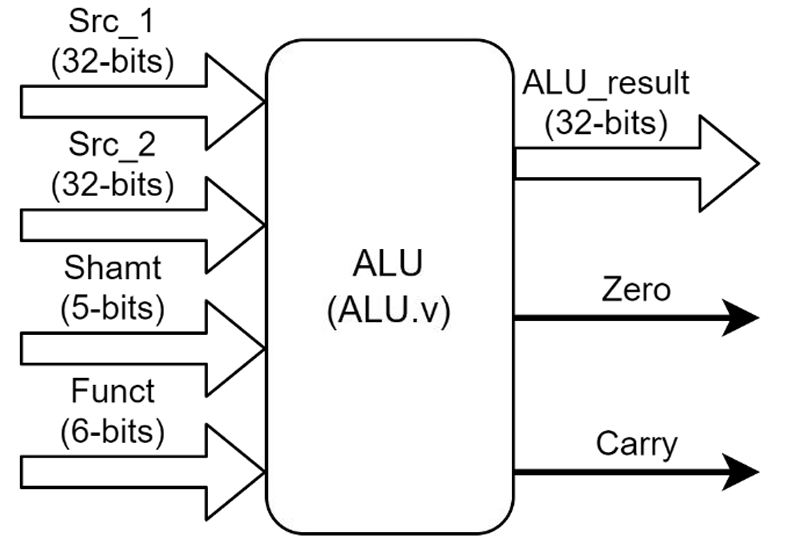
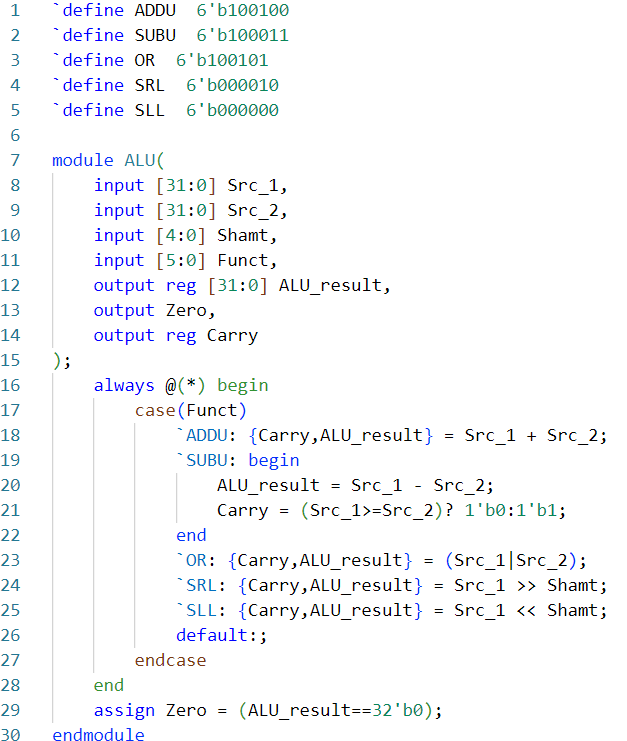


透過Register File圖

可以知道該架構是透過Rs\_addr和Rt\_addr兩個Address來選擇要輸出的Register資料

因此我們先宣告出32個32bits的Register，再利用assign來輸出想要的位址即可完成。

1. **32-bits Arithmetic Logic Unit :**



利用case語法去判斷Funct的值

然後分流各個不同的運算式

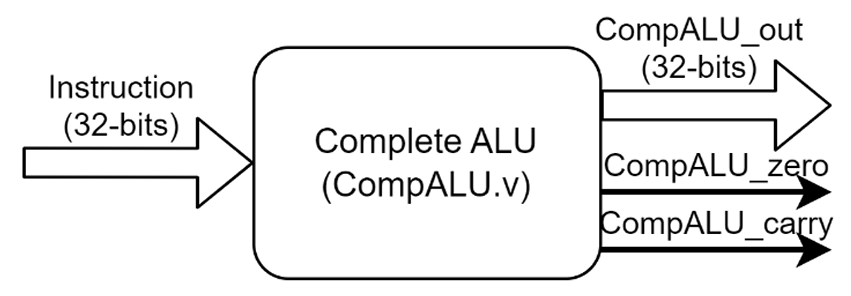
接著利用assign讓Zero在結果為0時為High，反之則Low。

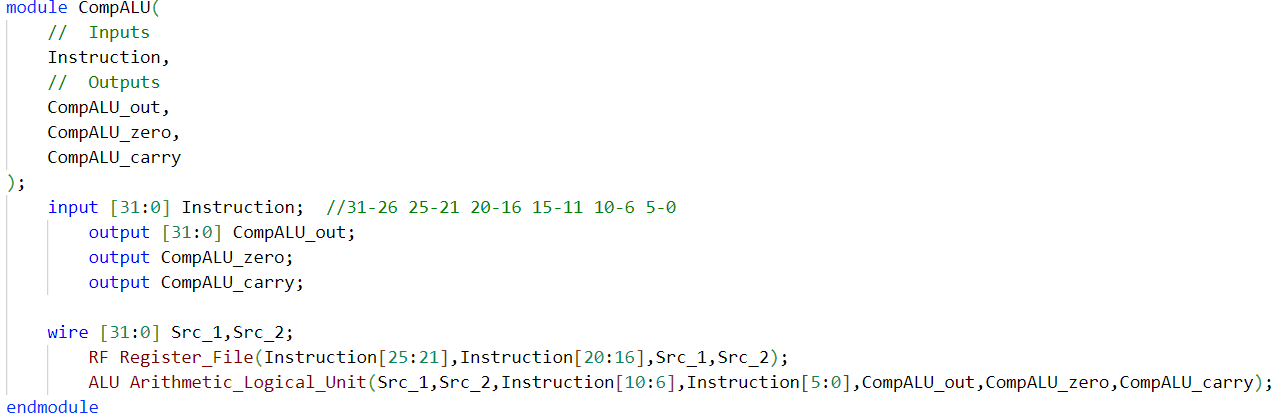
Funct的運算式如其定義名稱所示

例如ADDU代表無號數的加法…等。

Carry則當有進位或借位的發生設為1。

1. **32-bits Complete ALU :**

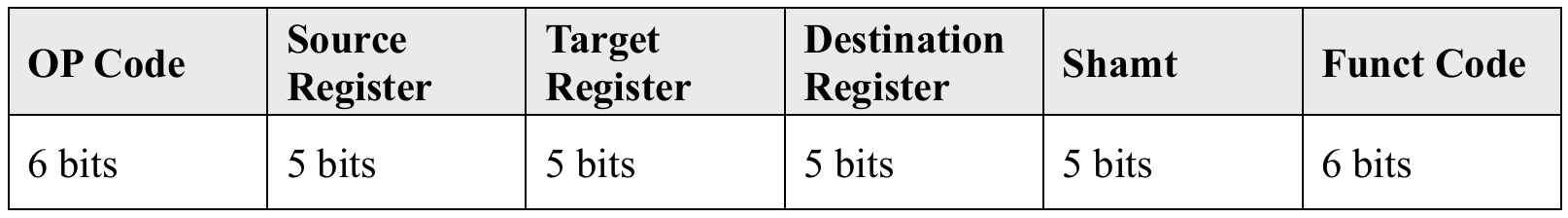




利用一個Complete ALU Module 把上述的(a)、(b) Module連接起來。

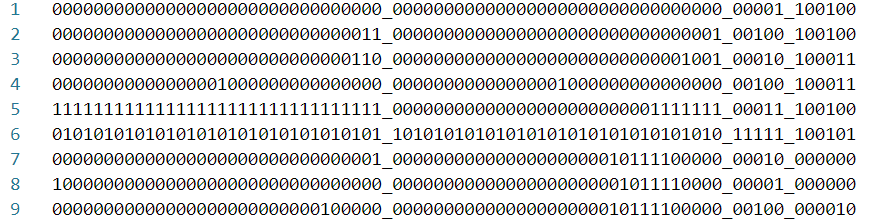
而32bits的Instruction則是按照R-Format 去設定，如下圖所示

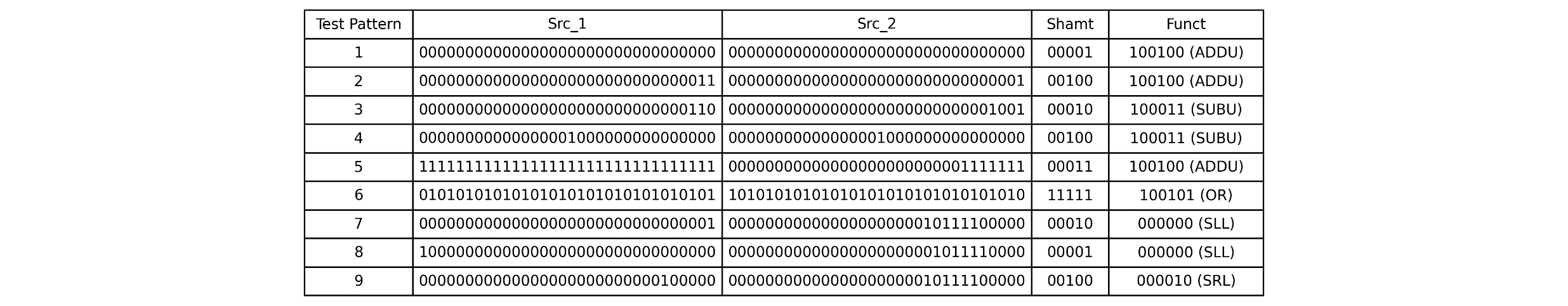
依照Instruction各個部份的功能分別接入RF和ALU裡面。



(R-Format Instruction)

1. **Screenshots and descriptions of test commands for each module :**
2. **tb\_ALU.in:**



以下為每個Test Pattern的每個input表格 :

測試目的 :

Test Pattern 1 : 測試加法和Zero Flag

Test Pattern 2 : 測試加法

Test Pattern 3 : 測試減法與借位Carry Flag

Test Pattern 4 : 測試減法和Zero Flag

Test Pattern 5 : 測試加法和進位Carry Flag

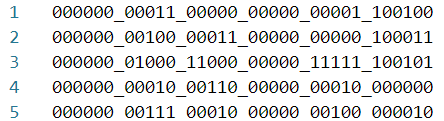
Test Pattern 6 : 測試OR

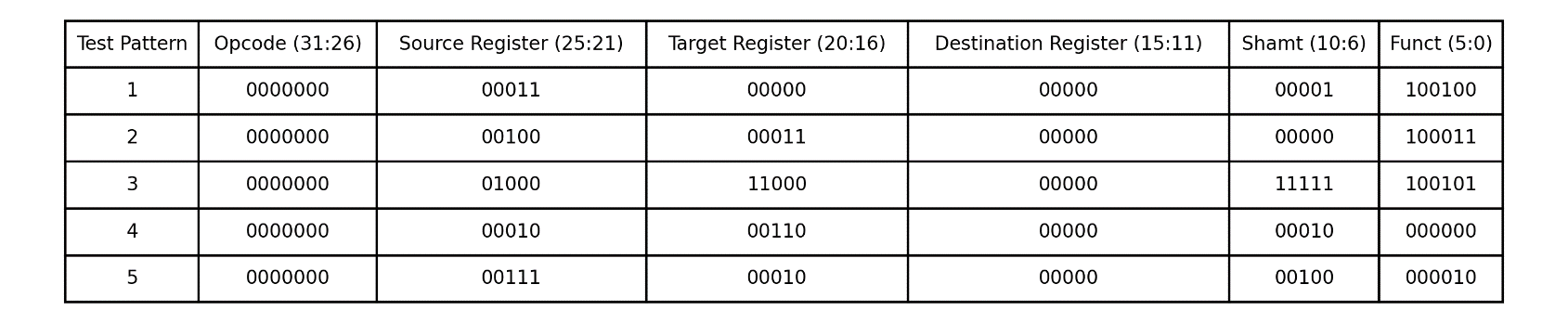
Test Pattern 7 : 測試向左位移

Test Pattern 8 : 測試向右位移和Carry Flag和Zero Flag

Test Pattern 9 : 測試向右位移

1. **tb\_CompALU.in:**



以下為每個Test Pattern Instruction各部分的表格 :

測試目的 :

Test Pattern 1 : 測試加法

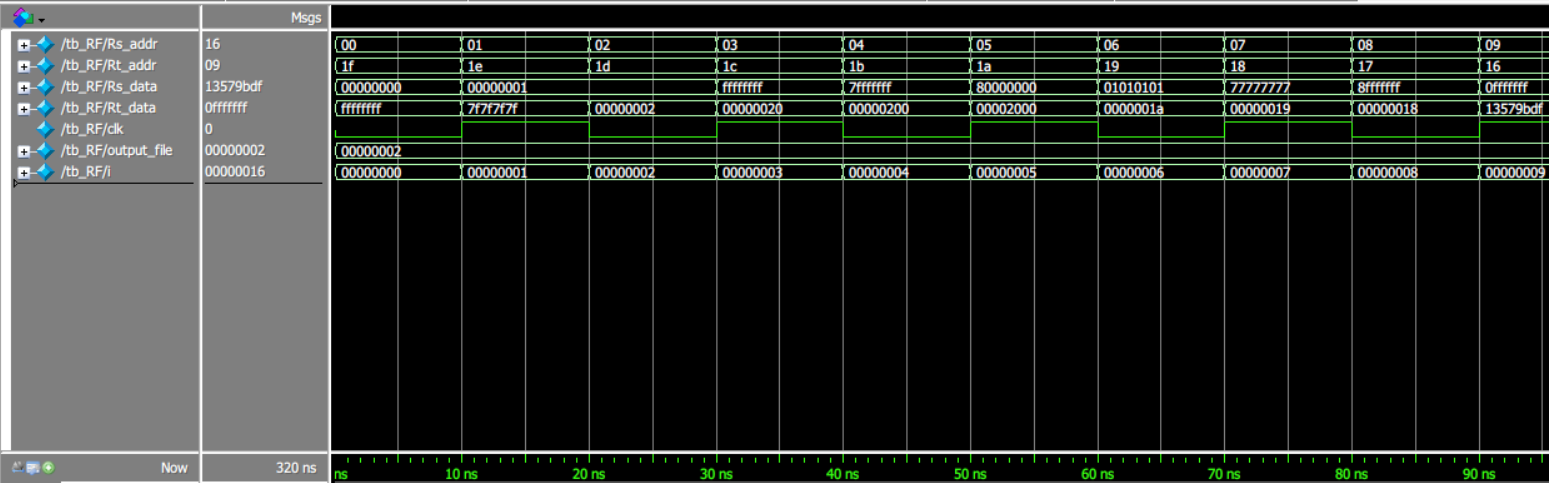
Test Pattern 2 : 測試減法

Test Pattern 3 : 測試OR

Test Pattern 4 : 測試向左位移

Test Pattern 5 : 測試向右位移

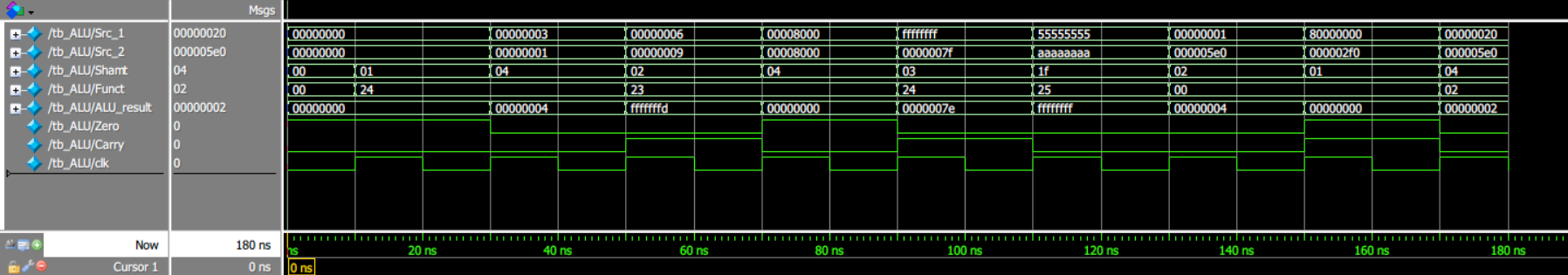
1. **Screenshots and explanations of the test result for each module :**
2. **tb\_RF**



該Test 送入了很多組不同的Address，由波型圖可觀察到其Rs\_data和Rt\_data

可以參考RF.dat中的檔案來確定是否正確。

1. **tb\_ALU**

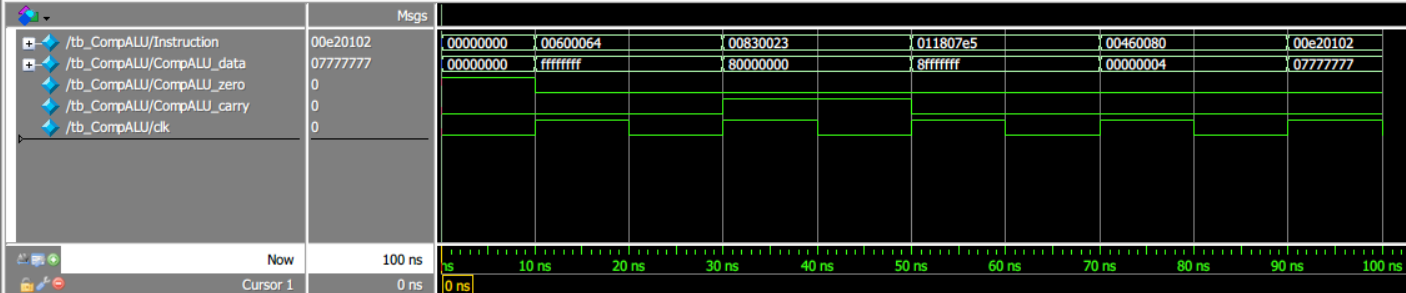
****

該Test如第二部分(a)所示，送入了9組不同的Test Pattern，分別測試不同的運算式。

如上圖波型所示，皆為正確結果。

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Test Pattern | 運算式 | 正確Carry | 正確Zero | 正確ALU\_Result |
| 1 | 00000000 + 00000000 | 0 | 1 | 00000000 |
| 2 | 00000003 + 00000001 | 0 | 0 | 00000004 |
| 3 | 00000006 - 00000009 | 1(借位) | 0 | fffffffd |
| 4 | 00000080 - 00000800 | 0 | 1 | 00000000 |
| 5 | ffffffff + 0000007f | 1(進位) | 0 | 0000007e |
| 6 | 55555555 | aaaaaaaa | 0 | 0 | ffffffff |
| 7 | 00000001 << 2 | 0 | 0 | 00000004 |
| 8 | 80000000 << 1 | 1(進位) | 0 | 00000000 |
| 9 | 00000020 >> 4 | 0 | 0 | 00000002 |

1. **tb\_CompALU**



該Test 把(a)和(b)統整成了CompALU

送入RF Address得到Data後，在透過ALU來計算不同的運算式得到的結果

該Test也如第二部分(b)所示，送入了5組不同的Test Pattern，分別測試不同的運算式。

1. **Conclusion and insight on this homework :**

對於本來就是數位IC組的我來說，在處理這份作業的過程中幾乎沒有遇到什麼問題，畢竟是我常常撰寫的Verilog和使用的Modelsim，反而是在書面上的報告遇到了一點癥結，思考著要怎麼呈現我的Test Pattern於報告才是最好的，後來就決定自己用Python製作了一個表格，方便閱讀。

雖然本次作業對我難度不高，不過在第一次模擬的助教課上，我發現身邊有些同學對於Modelsim怎麼操作，TestBench怎麼是什麼，甚至連Verilog的撰寫都快沒甚麼印象，看著助教都快應付不來大家的問題，有餘力的我當然也馬上幫助同學處理了許多問題，後來發現我還滿喜歡這個過程的，不過很可惜許多相關的課程都需要碩士才能當助教，沒有辦法一圓我的助教夢。

總之，在這次的作業中，我更加熟悉了關於ALU的運算、Register File的架構，也和很多位同學互相交流許多相關知識，很期待未來的三個Project，希望可以成功設計出一顆屬於自己的CPU架構 ~