



國立台灣科技大學

計算機組織

指導教授：陳雅淑 教授

計算機組織作業一報告

班 級 ： 四電機三甲
學 生 ： 呂佳祐
學 號 ： B11107055

A. 32-bits Read Only Register File

1. 主程式

```
1 module RF(  
2     //Inputs  
3     input [4:0] Rs_addr,  
4     input [4:0] Rt_addr,  
5     //Outputs  
6     output [31:0] Src_1,  
7     output [31:0] Src_2  
8  
9 );  
10     reg [31:0] R[0:31];  
11     assign Src_1 = R[Rs_addr];  
12     assign Src_2 = R[Rt_addr];  
13 endmodule
```

RF.v 程式碼

RF 唯讀暫存器寬度大小為 32bit、深度為 32bit。將輸入的 Rs_addr、Rt_addr 作為索引，透過陣列表示法，讀取對應暫存器內的值。因為 Register File 內有 $32(2^5)$ 個暫存器，所以 Address 大小宣告為 5 個 bit。而各暫存器內大小為 32bit，所以 Src_1、Src_2 大小宣告為 32 個 bit。

2. Testbench and Simulation

1	723D_1A01	17	A4F8_0A70
2	4E19_B210	18	E649_F2DB
3	D476_4746	19	8D4F_C5FF
4	4472_EE1F	20	EC46_E907
5	5002_4638	21	45B3_C03E
6	8371_E367	22	F900_9998
7	6DEC_8FB2	23	EA24_FDD1
8	AACC_9602	24	AB65_B051
9	7A4F_D3F4	25	E95F_ECD8
10	49A3_C944	26	B980_5B12
11	4E4B_5007	27	3732_D400
12	EF8E_3656	28	E0E2_2572
13	6171_70A5	29	4079_917D
14	E8CB_FA87	30	BE28_A15F
15	162F_31FF	31	FF64_D472
16	B487_3F9A	32	E856_F102

RF.dat 儲存 Register File 內各暫存器內資料

```

1 // Setting timescale
2 `timescale 10 ns / 1 ns
3 // Configuration
4 `define DELAY          1 // # * timescale
5 `define REGISTER_SIZE  32 // bit width
6 `define MAX_REGISTER   32 // index
7 `define DATA_FILE      "testbench/RF.dat"
8 `define OUTPUT_FILE     "testbench/tb_RF.out"
9 // Declaration
10 `define LOW             1'b0
11 `define HIGH            1'b1
12 module tb_RF;
13     // Inputs
14     reg [4:0] Rs_addr;
15     reg [4:0] Rt_addr;
16     // Outputs
17     wire [31:0] Rs_data;
18     wire [31:0] Rt_data;
19     // Clock
20     reg clk = `LOW;
21     // Testbench variables
22     reg [`REGISTER_SIZE-1:0] register [0:`MAX_REGISTER-1];
23     integer output_file;
24     integer i;

```

tb_RF.v 初始化設定

設定時間單位 10ns、時間精度 1ns。

```

25 // Instantiate the Unit Under Test (UUT)
26 RF UUT(
27     // Inputs
28     .Rs_addr(Rs_addr),
29     .Rt_addr(Rt_addr),
30     // Outputs
31     .Src_1(Rs_data),
32     .Src_2(Rt_data)
33 );
34 initial
35 begin : Preprocess
36     // Initialize inputs
37     Rs_addr = 32'b0;
38     Rt_addr = 32'b0;
39
40     // Initialize testbench files
41     $readmemh(`DATA_FILE, register);
42     output_file = $fopen(`OUTPUT_FILE);
43
44     // Initialize internal register
45     for (i = 0; i < `MAX_REGISTER; i = i + 1)
46     begin
47         UUT.R[i] = register[i];
48     end
49
50     #`DELAY; // Wait for global reset to finish
51 end

```

tb_RF.v 連接 RF.v module 與預處理

與主程式 module 連接。將 RF.dat 內資料讀取並存入 register 變數內，再將資料轉移至 Register File 裡，之後從 Register File 讀到的就是 RF.dat 內的資料。

```

52     always
53     begin : ClockGenerator
54         #`DELAY;
55         clk <= ~clk;
56     end
57     always
58     begin : StimuliProcess
59         // Start testing
60         for (i = 0; i < `MAX_REGISTER; i = i + 1)
61         begin
62             Rs_addr = i[`REGISTER_SIZE-1:0];
63             Rt_addr = `REGISTER_SIZE'd`MAX_REGISTER-1 - i[`REGISTER_SIZE-1:0];
64             @(clk); // Wait clock
65             $display("Rs_addr:%h, Rt_addr:%h", Rs_addr, Rt_addr);
66             $display("Src_1:%h, Src_2:%h", Rs_data, Rt_data);
67             $fdisplay(output_file, "%t,%h,%h", $time, Rs_data, Rt_data);
68         end
69
70         // Close output file for safety
71         $fclose(output_file);
72
73         // Stop the simulation
74         $stop();
75     end
76 endmodule

```

tb_RF.v 讀取暫存器資料

設定 clock 時序，週期為兩個`DELAY 的時間(20ns)。Source Register 是從頭開始讀取，而 Target Register 從最後一個暫存器開始讀。變數 i 資料型態為 integer，大小 32 個 bit；而 Address 大小為 5 個 bit，所以要用陣列表示法限定 i 的前 5 個 bit 賦值到 Rs_addr/Rt_addr。最後將讀取結果輸出到終端並記錄到 tb_RF.out。

		Msgs			
+ /tb_RF/Rs_addr	00	00	01	02	
+ /tb_RF/Rt_addr	1f	1f	1e	1d	
+ /tb_RF/Rs_data	723d1a01	723d1a01	4e19b210	d4764746	
+ /tb_RF/Rt_data	e856f102	e856f102	ff64d472	be28a15f	
+ /tb_RF/i	0	0	1	2	

03	04	05	06	07	08
1c	1b	1a	19	18	17
4472ee1f	50024638	8371e367	6dec8fb2	aacc9602	7a4fd3f4
4079917d	e0e22572	3732d400	b9805b12	e95fec8	ab65b051
3	4	5	6	7	8

09	0a	0b	0c	0d	0e	0f
16	15	14	13	12	11	10
49a3c944	4e4b5007	ef8e3656	617170a5	e8cbfa87	162f31ff	b4873f9a
ea24fdd1	f9009998	45b3c03e	ec46e907	8d4fc5ff	e649f2db	a4f80a70
9	10	11	12	13	14	15

tb_RF.v 模擬

模擬後總共會有 32 次讀取的資料，因為有兩個暫存器，一個從頭開始讀，一個從尾，所以這邊截圖到第 16 次讀取的資料就可以把所以暫存器的值都讀完。對比一下各位址的值，可以發現是與 RF.dat 吻合的。

B. 32-bits Arithmetic Logic Unit

1. 主程式

```
1  `define ADDU    6'b100100
2  `define SUBU    6'b100011
3  `define OR      6'b100101
4  `define SRL     6'b000010
5  `define SLL     6'b000000
6
7  module ALU(
8      // Inputs
9      input  [31:0] Src_1,
10     input  [31:0] Src_2,
11     input  [4:0]  Shamt,
12     input  [5:0]  Funct,
13     // Outputs
14     output reg [31:0] ALU_result,
15     output reg      Zero,
16     output reg      Carry
17 );
```

ALU.v 初始化設定

定義各 Function Code 對應的功能常數(ADDU、SUBU、...)，宣告此 module 要用到的 input 及 output。

```
18     always @(*) begin
19         {Carry, Zero, ALU_result} = 0;
20         case (Funct)
21             `ADDU: {Carry, ALU_result} = Src_1 + Src_2;
22             `SUBU: {Carry, ALU_result} = Src_1 - Src_2;
23             `OR:   ALU_result = {1'b0, Src_1 | Src_2};
24             `SRL:  ALU_result = Src_1 >> Shamt;
25             `SLL:  ALU_result = Src_1 << Shamt;
26             default: {Carry, ALU_result, Zero} = 0;
27         endcase
28         Zero = (ALU_result == 0) ? 1 : 0;
29     end
30
31 endmodule
```

ALU.v 主要功能

定義此 ALU 為由其中一個輸入變化而改變輸出的組合邏輯電路。

利用 Switch-Case 判斷 Function Code 對應的功能做運算，最後再用三元運算子判斷 ALU_result 是否全為 0；若是，則 Zero 旗標為 1。

2. Testbench and Simulation

```
1 00011000000110000000000000000011_0001000000010000001000110100001_00001_100100
2 00000000000000000000000000000000_000000000000000000000000000000_00100_100100
3 00000000000011100000000000100000_111111111111111111111111111111_00101_100100
4 00010000001000000000000000000011_000000000000000000000000000001_00001_100011
5 000000100000000100000000000000100_0000001000000001000000000000100_00100_100011
6 0000000001111100000000000100000_111111111111111111111111111111_00101_100011
7 00000010000000000000000000000011_00000001100000000000000001100001_00001_100101
8 00000110000010000000011000010000_0001100000000000001110000000100_00100_100101
9 00010000000000011000000100000_111000000001110000000000001000_00101_100101
10 11111111111111111111111111111111_000000000000001110000000000000_00001_000010
11 11111111111111111111111111111111_0000000000000000000000001100000000_11111_000010
12 11111111111111111111111111111111_00000000000000000000000000000000_00101_000010
13 11111111111111111111111111111111_0000000000000000000000000110000_10000_000000
14 11111111111111111111111111111111_00000000110000000000000000000000_11111_000000
15 11111111111111111111111111111111_00000000000000001100000000000000_00101_000000
```

tb_ALU.in 所有測試向量

測試向量將 5 個功能都測試到，每個功能都有三組資料。

```
1 // Setting timescale
2 `timescale 10 ns / 1 ns
3 // Configuration
4 `define DELAY      1 // # * timescale
5 `define INPUT_FILE "testbench/tb_ALU.in"
6 `define OUTPUT_FILE "testbench/tb_ALU.out"
7 // Declaration
8 `define LOW      1'b0
9 `define HIGH     1'b1
10 module tb_ALU;
11     // Inputs
12     reg [31:0] Src_1;
13     reg [31:0] Src_2;
14     reg [4:0] Shamt;
15     reg [5:0] Funct;
16     // Outputs
17     wire [31:0] ALU_result;
18     wire Zero;
19     wire Carry;
20     // Clock
21     reg clk = `LOW;
22     // Testbench variables
23     reg [74:0] read_data;
24     integer input_file;
25     integer output_file;
26     integer i;
27
28     // Instantiate the Unit Under Test (UUT)
29     ALU UUT(
30         .Src_1(Src_1),
31         .Src_2(Src_2),
32         .Shamt(Shamt),
33         .Funct(Funct),
34         // Outputs
35         .ALU_result(ALU_result),
36         .Zero(Zero),
37         .Carry(Carry)
38     );
```

tb_ALU.v 初始化設定

設定時間單位 10ns、時間精度 1ns。宣告各暫存器、變數、連接線，並與 ALU Module 連接。

```
40     begin : Preprocess
41         // Initialize inputs
42         Src_1    = 32'b0;
43         Src_2    = 32'b0;
44         Shamt    = 5'b0;
45         Funct    = 6'b0;
46         // Initialize testbench files
47         input_file = $fopen(`INPUT_FILE, "r");
48         output_file = $fopen(`OUTPUT_FILE);
49         #`DELAY;    // Wait for global reset to finish
50     end
51     always
52     begin : ClockGenerator
53         #`DELAY;
54         clk <= ~clk;
55     end
```

tb_ALU.v 預處理

初始化各變數、讀取 tb_ALU.in 內測試向量、設定 clock。

```
56     always
57     begin : StimuliProcess
58         // Start testing
59         while (!$feof(input_file))
60             begin
61                 $fscanf(input_file, "%b\n", read_data);
62                 @(posedge clk); // Wait clock
63                 {Src_1, Src_2, Shamt, Funct} = read_data;
64                 @(negedge clk); // Wait clock
65                 $display("Src_1:%b, Src_2:%b, Shamt:%b, Funct:%b", Src_1, Src_2, Shamt, Funct);
66                 $display("ALU_result:%d, Z:%b, C:%b", ALU_result, Zero, Carry);
67                 $fdisplay(output_file, "%t,%b,%b,%b", $time, ALU_result, Zero, Carry);
68             end
69         // Close output file for safety
70         $fclose(output_file);
71         // Stop the simulation
72         $stop();
73     end
74 endmodule
```

tb_ALU.v 測試

將各行測試向量賦值進 read_data 變數，並利用連結運算子將 read_data 拆分進 Src_1、Src_2、Shamt、Funct。最後輸出結果至終端與 tb_ALU.out。

```

56     always
57     begin : StimuliProcess
58         // Start testing
59         while (!$feof(input_file))
60             begin
61                 $fscanf(input_file, "%b\n", read_data);
62                 @(posedge clk); // Wait clock
63                 {Src_1, Src_2, Shamt, Funct} = read_data;
64                 @(negedge clk); // Wait clock
65                 $display("Src_1:%b, Src_2:%b, Shamt:%b, Funct:%b", Src_1, Src_2, Shamt, Funct);
66                 $display("ALU_result:%d, Z:%b, C:%b", ALU_result, Zero, Carry);
67                 $fdisplay(output_file, "%t,%b,%b,%b", $time, ALU_result, Zero, Carry);
68             end
69         // Close output file for safety
70         $fclose(output_file);
71         // Stop the simulation
72         $stop();
73     end
74 endmodule

```

tb_ALU.v 測試

I. 加法測試

	Msgs	
/tb_ALU/Src_1	10200003	00000000 18180003
/tb_ALU/Src_2	00000001	00000000 100811a1
/tb_ALU/Shamt	01	00 01
/tb_ALU/Funct	23	00 24
/tb_ALU/ALU_result	10200002	00000000 282011a4
/tb_ALU/Zero	St0	
/tb_ALU/Carry	St0	
/tb_ALU/dk	0	
/tb_ALU/read_data	0100800201008002123	0c0c0001880408d0864 0000000000000000124

	Msgs	
/tb_ALU/Src_1	00000000	00000000 000e0020
/tb_ALU/Src_2	00000000	00000000 ffffffff
/tb_ALU/Shamt	04	04 05
/tb_ALU/Funct	24	24
/tb_ALU/ALU_result	00000000	00000000 000e001f
/tb_ALU/Zero	St1	
/tb_ALU/Carry	St0	
/tb_ALU/dk	1	
/tb_ALU/read_data	0000000000000000124	0000000000000000... 000700107ffffff964 08100001800000000863

tb_ALU.v 模擬加法部分

三組資料測試後可發現加法後的結果正確、Zero 與 Carry 旗標正常運作、Shamt 不影響加法功能。由此可判斷加法功能正常執行。

II. 減法測試

		Mags																	
tb_ALU_Src_1	10200003	10200003						07010004								007c0020			
tb_ALU_Src_2	00000001	00000001						02010004								fffffff			
tb_ALU_Shamt	01	01						04								05			
tb_ALU_Funct	23	23																	
tb_ALU_ALU_result	10200002	10200002						00000000								007c0021			
tb_ALU_Zero	St0																		
tb_ALU_Carry	St0																		
tb_ALU_dk	1																		
tb_ALU_read_data	081000018000000000863	081000018000000000863					0100800201008002123									003e00107fffff563			0100000180000000865

tb_ALU.v 模擬減法部分

三組資料測試發現減法結果正確、Zero 與 Carry(借位)旗標正常運作、Shamt 不影響減法功能。由此可判斷減法功能正常執行。

III. OR 測試

	Maps								
/tb_ALU Src_1	02000003	02000003				06080e10		10006020	
/tb_ALU Src_2	01800061	01800061				18001c04		e03ee008	
/tb_ALU Zshamt	01	01				04		05	
/tb_ALU Funct	25	25							
/tb_ALU ALU_result	03800063	03800063				1e081e14		f00e6028	
/tb_ALU Zero	S10								
/tb_ALU Carry	S10								
/tb_ALU dk	1								
/tb_ALU read_data	0100000180cd00030865	0100000180cd00030865	03f40308cd00e02125			08003d107007000+165		7fffffff8000e000042	

tb_ALU.v 模擬 OR 部分

三組資料測試發現或閘結果正確、Zero 與 Carry 正常運作、Shamt 不影響或閘功能。由此可判斷或閘功能正常執行。

IV. 右移測試

[illegible]

三組資料測試發現右移正常、Zero 與 Carry 正常運作、右移位元數確實依照 Shamt 決定。由此可判斷右移功能正常執行。

V. 左移測試

tb_ALU/Src_1	ffffff	ffffff							
tb_ALU/Src_2	00000030	00000030			00c00000			0000c000	
tb_ALU/Shamt	10	10			1f			05	
tb_ALU/Func	00	00							
tb_ALU/ALU_result	ffff0000	ffff0000			80000000			ffffffe0	
tb_ALU/Zero	St0								
tb_ALU/Carry	St0								
tb_ALU/dk	1								
tb_ALU/read_data	7ffffff80000018400	7ffffff80000018400			7ffffff806000007c0			7ffffff80005000140	

tb_ALU.v 模擬左移部分

三組資料測試發現左移正常、Zero 與 Carry 正常運作、左移位元數確實依照 Shamt 決定。由此可判斷左移功能正常執行。

C. 32-bits Complete ALU

1. 主程式

```

1  ~ module CompALU(
2      // Inputs
3      input  [31:0] Instruction,
4      // Outputs
5      output [31:0] CompALU_out,
6      output          CompALU_zero,
7      output          CompALU_carry
8  );
9  wire [31:0] Inner_Src_1;
10 wire [31:0] Inner_Src_2;
11 ~ RF Register_File(
12     // Inputs
13     .Rs_addr(Instruction[25:21]),
14     .Rt_addr(Instruction[20:16]),
15     // Outputs
16     .Src_1(Inner_Src_1),
17     .Src_2(Inner_Src_2)
18 );
19 ~ ALU Arithmetic_Logical_Unit(
20     // Inputs
21     .Src_1(Inner_Src_1),
22     .Src_2(Inner_Src_2),
23     .Shamt(Instruction[10:6]),
24     .Funct(Instruction[5:0]),
25     // Outputs
26     .ALU_result(CompALU_out),
27     .Zero(CompALU_zero),
28     .Carry(CompALU_carry)
29 );
30 endmodule

```

CompALU.v 程式碼

這程式把 Register File 與 ALU 結合起來，輸入 32-bits 的 Instruction，利用陣列表示法將 Instruction 分離出 Source Register、Target Register、Shamt、Function Code，並各自傳入 RF 與 ALU module。就像是在 CompALU 這黑盒子內將 RF 與 ALU 連接起來。

2. Testbench and Simulation

```

1 000000_10001_00010_00000_00000_100100
2 000000_01011_00100_00000_00000_100100
3 000000_00101_00110_00000_00000_100100
4 000000_00111_00010_00000_00000_100011
5 000000_10011_00100_00000_00000_100011
6 000000_00001_01000_00000_00000_100011
7 000000_01001_00100_00000_00000_100101
8 000000_10101_00110_00000_00000_100101
9 000000_00111_01000_00000_00000_100101
10 000000_11001_00000_00000_10101_000010
11 000000_00000_00000_00000_11111_000010
12 000000_10111_00000_00000_00100_000010
13 000000_11101_00000_00000_00000_000000
14 000000_11111_00000_00000_11011_000000
15 000000_00100_00000_00000_01100_000000

```

tb_CompALU.in 測試向量

每個功能都有三組測試資料，總共十五組測試向量。

```

1 // Setting timescale
2 `timescale 10 ns / 1 ns
3 // Declarations
4 `define DELAY      1 // # * timescale
5 `define REGISTER_SIZE 32 // bit width
6 `define MAX_REGISTER 32 // index
7 `define DATA_FILE "testbench/RF.dat"
8 `define INPUT_FILE "testbench/tb_CompALU.in"
9 `define OUTPUT_FILE "testbench/tb_CompALU.out"
10 // Declaration
11 `define LOW 1'b0
12 `define HIGH 1'b1
13 module tb_CompALU;
14     // Inputs
15     reg [31:0] Instruction;
16     // Outputs
17     wire [31:0] CompALU_data;
18     wire CompALU_zero;
19     wire CompALU_carry;
20     // Clock
21     reg clk = `LOW;
22     // Testbench variables
23     reg [`REGISTER_SIZE-1:0] register [0:`MAX_REGISTER-1];
24     reg [31:0] read_data;
25     integer input_file;
26     integer output_file;
27     integer i;
28
29     // Instantiate the Unit Under Test (UUT)
30     CompALU UUT(
31         // Inputs
32         .Instruction(Instruction),
33         // Outputs
34         .CompALU_data(CompALU_out),
35         .CompALU_zero(CompALU_zero),
36         .CompALU_carry(CompALU_carry)
37     );

```

tb_CompALU.v 初始化設定

設定時間單位 10ns、時間精度 1ns。宣告各變數並連接至 CompALU module。

```

37     initial
38     begin : Preprocess
39         // Initialize inputs
40         // Format: OpCode_Src1addr_Src2addr_RESERVED_shamt_funcnt
41         Instruction = 32'b0000000_000000_000000_000000_000000;
42         // Initialize testbench files
43         $readmemh(`DATA_FILE, register);
44         input_file = $fopen(`INPUT_FILE, "r");
45         output_file = $fopen(`OUTPUT_FILE);
46         // Initialize internal register
47         for (i = 0; i < `MAX_REGISTER; i = i + 1)
48             begin
49                 UUT.Register_File.R[i] = register[i];
50             end
51         #`DELAY;    // Wait for global reset to finish
52     end
53     always
54     begin : ClockGenerator
55         #`DELAY;
56         clk <= ~clk;
57     end

```

tb_CompALU.v 預處理

將 RF.dat 記憶體資料一一匯入 Redister File 個暫存器內，並打開 tb_CompALU.in，將檔案位址傳進 input_file 中，方便之後使用。ClockGenerator 產生週期為 20ns 的 clock。

```

58     always
59     begin : StimuliProcess
60         // Start testing
61         while (!$feof(input_file))
62             begin
63                 $fscanf(input_file, "%b\n", read_data);
64                 @(posedge clk); // Wait clock
65                 Instruction = read_data;
66                 @(negedge clk); // Wait clock
67                 $display("Instruction:%b", read_data);
68                 $display("CompALU_data:%d, Z:%b, C:%b", CompALU_data, CompALU_zero, CompALU_carry);
69                 $fdisplay(output_file, "%t,%b,%b,%b", $time, CompALU_data, CompALU_zero, CompALU_carry);
70             end
71         // Close output file for safety
72         $fclose(output_file);
73         // Stop the simulation
74         $stop();
75     end
76 endmodule

```

tb_CompALU.v 開始傳入資料

將 tb_CompALU.in 內的測試向量，每隔一個 clock 週期，在上緣觸發時傳入 CompALU 中。最後將結果輸出至終端與 tb_CompALU.out 檔案裡。

I. 加法測試

/tb_CompALU/Instruction	02640023	00000000	02220024	01640024	00a60024
/tb_CompALU/CompALU_data	9c44a2cf	723d1a01	bac03a21	3f907c8e	f15e7319
/tb_CompALU/CompALU_zero	S10				
/tb_CompALU/CompALU_carry	S10				
/tb_CompALU/dk	0				
/tb_CompALU/UUT/Register_File/Rs_addr	19	0	17	11	5
/tb_CompALU/UUT/Register_File/Rt_addr	4	0	2	4	6
/tb_CompALU/UUT/Register_File/Src_1	ec46e907	723d1a01	e649f2db	ef8e3656	8371e367
/tb_CompALU/UUT/Register_File/Src_2	50024638	723d1a01	d4764746	50024638	6dec8fb2
/tb_CompALU/UUT/Arithmetic_Logical_Unit/Shamt	00000	00000			
/tb_CompALU/UUT/Arithmetic_Logical_Unit/Funct	10011	000000	100100		
/tb_CompALU/UUT/Arithmetic_Logical_Unit/ALU_result	9c44a2cf	723d1a01	bac03a21	3f907c8e	f15e7319

tb_CompALU.v 模擬加法測試

根據 address 從 RF 取的值與 RF.dat 對比後相符、加法結果正確、Zero 與 Carry 旗標正確。以上確認加法功能正常。

II. 減法測試

/tb_CompALU/Instruction	00e20023	00e20023	02640023	00280023
/tb_CompALU/CompALU_data	d6564ebc	d6564ebc	9c44a2cf	d3c9de1c
/tb_CompALU/CompALU_zero	S10			
/tb_CompALU/CompALU_carry	S10			
/tb_CompALU/dk	1			
/tb_CompALU/UUT/Register_File/Rs_addr	7	7	19	1
/tb_CompALU/UUT/Register_File/Rt_addr	2	2	4	8
/tb_CompALU/UUT/Register_File/Src_1	aacc9602	aacc9602	ec46e907	4e19b210
/tb_CompALU/UUT/Register_File/Src_2	d4764746	d4764746	50024638	7a4fd3f4
/tb_CompALU/UUT/Arithmetic_Logical_Unit/Shamt	00000	00000		
/tb_CompALU/UUT/Arithmetic_Logical_Unit/Funct	10011	10011		
/tb_CompALU/UUT/Arithmetic_Logical_Unit/ALU_result	d6564ebc	d6564ebc	9c44a2cf	d3c9de1c

tb_CompALU.v 模擬減法測試

根據 address 從 RF 取的值與 RF.dat 對比後相符、減法結果正確、Zero 與 Carry 旗標正確。以上確認減法功能正常。

III. OR 測試

/tb_CompALU/Instruction	01240025	01240025	02a60025	00a80025
/tb_CompALU/CompALU_data	59a3cf7c	59a3cf7c	fdcc9fba	facfd7f6
/tb_CompALU/CompALU_zero	S10			
/tb_CompALU/CompALU_carry	S10			
/tb_CompALU/dk	1			
/tb_CompALU/UUT/Register_File/Rs_addr	9	9	21	7
/tb_CompALU/UUT/Register_File/Rt_addr	4	4	6	8
/tb_CompALU/UUT/Register_File/Src_1	49a3c944	49a3c944	f9009998	aacc9602
/tb_CompALU/UUT/Register_File/Src_2	50024638	50024638	6dec8fb2	7a4fd3f4
/tb_CompALU/UUT/Arithmetic_Logical_Unit/Shamt	00000	00000		
/tb_CompALU/UUT/Arithmetic_Logical_Unit/Funct	100101	100101		
/tb_CompALU/UUT/Arithmetic_Logical_Unit/ALU_result	59a3cf7c	59a3cf7c	fdcc9fba	facfd7f6

tb_CompALU.v 模擬 OR 測試

或閘運算結果正確、Zero 與 Carry 旗標正常反應。以上確認減法功能正常。

IV. 右移測試

/tb_CompALU/Instruction	03200542	03200542	000007c2	02e00102
/tb_CompALU/CompALU_data	000005cc	000005cc	00000000	0ab65b05
/tb_CompALU/CompALU_zero	St0			
/tb_CompALU/CompALU_carry	St0			
/tb_CompALU/dk	1			
/tb_CompALU/IUT/Register_File/Rs_addr	25	25	0	23
/tb_CompALU/IUT/Register_File/Rt_addr	0	0		
/tb_CompALU/IUT/Register_File/Src_1	b9805b12	b9805b12	723d1a01	ab65b05
/tb_CompALU/IUT/Register_File/Src_2	723d1a01	723d1a01		
/tb_CompALU/IUT/Arithmetic_Logical_Unit/Shamt	10101	10101	11111	00100
/tb_CompALU/IUT/Arithmetic_Logical_Unit/Funct	000010	000010		
/tb_CompALU/IUT/Arithmetic_Logical_Unit/ALU_result	000005cc	000005cc	00000000	0ab65b05

tb_CompALU.v 模擬右移測試

右移數量有確實依照 shamt 決定，結果正確、旗標正常運作，以上確定右移功能正常。

V. 左移測試

/tb_CompALU/Instruction	03a00000	03a00000	03e006c0	00800300
/tb_CompALU/CompALU_data	be28a15f	be28a15f	10000000	24638000
/tb_CompALU/CompALU_zero	St0			
/tb_CompALU/CompALU_carry	St0			
/tb_CompALU/dk	1			
/tb_CompALU/IUT/Register_File/Rs_addr	29	29	31	4
/tb_CompALU/IUT/Register_File/Rt_addr	0	0		
/tb_CompALU/IUT/Register_File/Src_1	be28a15f	be28a15f	e856f102	50024638
/tb_CompALU/IUT/Register_File/Src_2	723d1a01	723d1a01		
/tb_CompALU/IUT/Arithmetic_Logical_Unit/Shamt	000000	000000	11011	01100
/tb_CompALU/IUT/Arithmetic_Logical_Unit/Funct	000000	000000		
/tb_CompALU/IUT/Arithmetic_Logical_Unit/ALU_result	be28a15f	be28a15f	10000000	24638000

tb_CompALU.v 模擬左移測試

左移數量與 shamt 相符，輸出結果正確、旗標正常運作，以上確定左移功能正常。

D. Conclusion and Insight

一個 CPU 裡包含許多功能，我們可以把這些功能先分開實作，而每個獨立的實作就像是個黑盒子。確定個別實作沒問題後，我們只要把各個黑盒子相對應的接腳連接上就行，最後再組成一個更大的黑盒子。如此這樣模組化、階層化的去看待電路，會使得開發及維護更輕鬆。

在設計測試向量時，除了要有正常的數據外，還需添加邊緣條件，確保電路的每個輸出端都正常運作、電路可以正常處理近乎全部的輸入。像是在此專案，為了要測試 Zero 旗標，可能就要故意將兩個指定暫存器的值設相同，或兩個 Address 都指向同個暫存器。或是測試在不需要用到 Shamt 時的場景，例如

加法時，給 Shamt 一個值，看會不會對加法功能有影響，若有，表示電路不如預期。