附錄 A　MIPS CPU 指令集格式

MIPS CPU 指令集格式分為 R-Type、I-Type 及 J-Type 三種類型，各類型指令格式如表 A-1、表 A-2、表 A-3 所示。

表 A-1 R-Type 指令格式

31 26	25 21	20 16	15 11	10 6	5 0
OP-Code	Rs	Rt	Rd	Shamt	Func-Code

OP-Code(Operand Code)欄位為主要指令碼，在 R-Type 指令中 OP-Code 欄位皆為 000000，另需配合 Func-Code(Function Code)欄位做完整指令判斷。

R-Type 指令除移位指令外，其他均為暫存器對暫存器運算指令，因此具有 2 個來源暫存器(Rs、Rt)及 1 個目的暫存器(Rd)。移位指令運算是由 Rt 暫存器內容與移位量(Shamt)運算後，儲存於 Rd 暫存器。

表 A-2 I-Type 指令格式

31 26	25 21	20 16	15 0
OP-Code	Rs	Rt	Immed

I-Type 指令為暫存器與立即值(Immediate Value)的運算，其中 Rs 為來源暫存器，Rt 則為目的暫存器，Immed 欄位為立即值輸入。

表 A-3 J-Type 指令格式

31 26	25 0
OP-Code	Addr

J-Type 指令為分支指令中的一種，屬無條件分支指令。Addr(Address)欄位為直接跳躍位址。

在本研究中將指令集再細分為 R-Type、I-Type、J-Type、M-Type 及 B-Type 五種類型，以便設計過程中做為區分。M-Type 是記憶體存取相關指令，B-Type 是分支指令中的條件分支指令。因此，本研究所實作的記憶體相關(memory-reference)指令、算術邏輯運算(arithmetic-logic)指令、分支指令，已包涵 MIPS 指令集中的主要指令。

在實作指令集編碼(Encoding)方面，遵循 MIPS CPU 指令集編碼並未

做任何修改，因此能與其他 MIPS CPU 相容。本研究所實作的 MIPS CPU 指令集編碼如表 A-4 所示，可與第三章第二節之表 3-3 對照參考。

表 A-4 實作指令集編碼一覽表

Encoding / Mnemonic	Encoding						Inst. Type
	31 26	25 21	20 16	15 11	10 6	5 0	
LW	100011	sssss	ttttt	iiiii	iiiii	iiiiii	M
SW	101011	sssss	ttttt	iiiii	iiiii	iiiiii	M
ADD	000000	sssss	ttttt	ddddd	00000	100000	R
ADDU	000000	sssss	ttttt	ddddd	00000	100001	R
SUB	000000	sssss	ttttt	ddddd	00000	100010	R
SUBU	000000	sssss	ttttt	ddddd	00000	100011	R
AND	000000	sssss	ttttt	ddddd	00000	100100	R
OR	000000	sssss	ttttt	ddddd	00000	100101	R
XOR	000000	sssss	ttttt	ddddd	00000	100110	R
NOR	000000	sssss	ttttt	ddddd	00000	100111	R
SLL	000000	sssss	ttttt	ddddd	hhhhh	000000	R
SRL	000000	-----	ttttt	ddddd	hhhhh	000010	R
SRA	000000	-----	ttttt	ddddd	hhhhh	000011	R
ADDI	001000	sssss	ttttt	iiiii	iiiii	iiiiii	I
ADDIU	001001	sssss	ttttt	iiiii	iiiii	iiiiii	I
ANDI	001100	sssss	ttttt	iiiii	iiiii	iiiiii	I
ORI	001101	sssss	ttttt	iiiii	iiiii	iiiiii	I
XORI	001110	sssss	ttttt	iiiii	iiiii	iiiiii	I
BEQ	000100	sssss	ttttt	iiiii	iiiii	iiiiii	B
BGEZ	000001	sssss	00001	iiiii	iiiii	iiiiii	B
BGTZ	000111	sssss	00000	iiiii	iiiii	iiiiii	B
BLEZ	000110	sssss	00000	iiiii	iiiii	iiiiii	B
BLTZ	000001	sssss	00000	iiiii	iiiii	iiiiii	B
BNE	000101	sssss	ttttt	iiiii	iiiii	iiiiii	B
J	000010	iiiii	iiiii	iiiii	iiiii	iiiiii	J

附錄B　MIPS CPU設計及I/O控制之教學與實作 範例教材

本研究之CPU設計教學及I/O介面控制之教學與實作範例教材分為五章，如下所示。

第一章　組合邏輯電路設計
- 1-1 編碼器與解碼器
- 1-2 多工器與解多工器
- 1-3 比較器
- 1-4 加減法器與BCD加法器
- 1-5 三態閘

第二章　順序邏輯電路設計
- 2-1 D型、T型、JK正反器
- 2-2 計數器
- 2-3 除頻電路
- 2-4 移位電路
- 2-5 狀態機介紹及設計範例

第三章　Multiple Clock Cycles MIPS CPU設計
- 3-1 指令暫存器與暫存器堆
- 3-2 記憶體與程式計數器
- 3-3 Barrel Shifter
- 3-4 算術邏輯運算單元與ALU控制單元
- 3-5 資料路徑整合
- 3-6 CU控制單元
- 3-7 MIPS CPU整合
- 3-8 MIPS CPU測試與模擬

第四章　I/O介面控制電路設計
- 4-1 防彈跳電路
- 4-2 鍵盤控制電路
- 4-3 七段顯示器控制電路
- 4-4 文字型LCD控制電路

第五章　系統測試
- 5-1 測試流程
- 5-2 測試程式

本範例教材僅針對第三章至第四章的設計加以編輯。

第三章 Multiple Clock Cycles MIPS CPU 設計

Multiple Clock Cycles MIPS CPU 其中的 Multiple Clock Cycles 是指將每個指令的執行週期，切割成數個 Clocks 完成。把執行指令所需的運算總量平均在 n 個 Clock 中完成，每一個 Clock 僅僅執行一個主要的功能，在最後一個週期，儲存未來將使用的運算結果。此種設計的優點為電路中各功能模組單元能分享，能節省 FPGA 晶片的 Gate Count 使用量。

Multiple Clock Cycles MIPS CPU 設計架構如圖 4-10 所示。實作指令語法如表 B-1 所示。

表 B-1 實作指令語法一覽表

NO.	Mnemonic	OP Code	Syntax		Operation
1	LW	100011	lw	$t, offset($s)	$t=MEM[$s+offset]
2	SW	101011	sw	$t, offset($s)	MEM[$s+offset] =$t
3	ADD	000000	add	$d, $s, $t	$d =$s + $t
4	ADDU	000000	addu	$d, $s, $t	$d =$s + $t
5	SUB	000000	sub	$d, $s, $t	$d =$s - $t
6	SUBU	000000	subu	$d, $s, $t	$d =$s - $t
7	AND	000000	and	$d, $s, $t	$d =$s & $t
8	OR	000000	or	$d, $s, $t	$d =$s | $t
9	XOR	000000	xor	$d, $s, $t	$d =$s ^ $t
10	NOR	000000	nor	$d, $s, $t	$d = ~($s | $t)
11	SLL	000000	sll	$d, $t, shamt	$d =$s << shamt
12	SRL	000000	srl	$d, $t, shamt	$d =$s >> shamt
13	SRA	000000	sra	$d, $t, shamt	$d =$s >> shamt
14	ADDI	001000	addi	$t, $s, immed	$t =$s + immed
15	ADDIU	001001	addiu	$t, $s, immed	$t =$s + immed
16	ANDI	001100	andi	$t, $s, immed	$t =$s & immed
17	ORI	001101	ori	$t, $s, immed	$t =$s | immed
18	XORI	001110	xori	$t, $s, immed	$t =$s ^ immed
19	BEQ	000100	beq	$s, $t, offset	if $s==$t advance_pc
20	BGEZ	000001	bgez	$s, $1, offset	if $s>=$t advance_pc
21	BGTZ	000111	bgtz	$s, $0, offset	if $s>$t advance_pc
22	BLEZ	000110	blez	$s, $0, offset	if $s<=$t advance_pc
23	BLTZ	000001	bltz	$s, $0, offset	if $s<$t advance_pc
24	BNE	000101	bne	$s, $t, offset	if $s!=$t advance_pc
25	J	000010	j	target	pc=pc&(000000 | target)

在熟悉 Multiple Clock Cycles MIPS CPU 基本架構與運算流程後，進行實作設計項目。整體設計模組將在 3-1 至 3-7 節中說明，最後於 3-8 節進行功能模擬。

3-1 指令暫存器與暫存器堆

3-1-1 指令暫存器(Instruction Register)

一、相關知識：指令暫存器必須暫存由記憶體取出的指令，並依指令格式將指令的每個位元所代表的意義，發派至下一單元。構成的基本元件為 D 型正反器。

二、輸出入接腳規劃及說明：輸出入接腳規劃如圖 B-1 所示。說明如下。

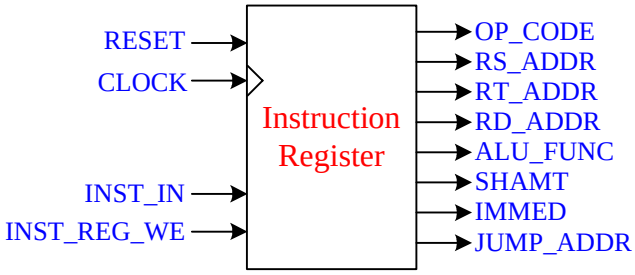


圖 B-1 指令暫存器模組輸出入接腳規劃

(一)輸入：

1. RESET：重置。重設指令暫存器，當 RESET=1 時，將指令暫存器內容設定為 0x00000000。
2. CLOCK：時脈輸入。採用 CLOCK 正緣觸發動作。
3. INST_IN[31:0]：指令輸入，來源為記憶體模組的輸出端。
4. INST_REG_WE：指令暫存器寫入致能。

(二)輸出：

1. OP_CODE[5:0]：操作碼輸出。
2. RS_ADDR[4:0]：暫存器 Rs 位址指定輸出。
3. RT_ADDR[4:0]：暫存器 Rt 位址指定輸出。
4. RD_ADDR[4:0]：暫存器 Rd 位址指定輸出。
5. ALU_FUNC[5:0]：R-Type 指令之 Function Code 輸出。
6. SHAMT[4:0]：移位指令之移位量指定。
7. IMMED[15:0]：I-Type 指令之立即值指定。
8. JUMP_ADDR[25:0]：J-Type 指令之跳躍位址指定。

三、Instruction Register 的輸出-輸入控制信號，如表 B-2 所示。整合模組輸出入接腳及輸出入控制信號表，設計 Instruction Register 之 VHDL

程式。

四、軟體模擬結果如圖 B-37 所示。

表 B-2 Instruction Register Input-Output Table

Input			Output
RESET	CLOCK	INST_REG_WE	Function
1	X	X	All Outputs = 0x00000000
0	L	X	All Outputs = 0x00000000
	H	X	All Outputs = 0x00000000
	↓	X	All Outputs = 0x00000000
	↑	0	All Outputs = 0x00000000
		1	OP_CODE = INST_IN[31:26]
			RS_ADDR = INST_IN[25:21]
			RT_ADDR = INST_IN[20:16]
			RD_ADDR = INST_IN[15:11]
			SHAMT = INST_IN[10:6]
			ALU_FUNC = INST_IN[5:0]
			IMMED = INST_IN[15:0]
			JUMP_ADDR = INST_IN[25:0]

3-1-2 暫存器堆(Register Files)

一、相關知識：暫存運算執行過程中的運算值。暫存器是數位系統中的基本元件，資料在暫存器中傳遞及處理。由正反器及相關電路形成的循序邏輯電路都可稱為暫存器。對儲存於暫存器中的資料進行運算稱為微運算(micro-operations)，例如把資料從一個暫存器移到另一個暫存器中、將兩個暫存器的值相加及將暫存器的值加 1…等。執行微運算時，所需的來源暫存器(Source Register, Rs)內容被送到 ALU 的輸入，ALU 執行某些運算後再把結果回存入目的暫存器(Destination Register, Rd)中。因為 ALU 為組合邏輯電路，所以整個暫存器資料轉移過程，從來源暫存器到 ALU 再到目的暫存器可在一個時脈中完成[13]。

二、基本暫存器之輸出入接腳規劃如圖 B-2 所示。組成基本元件為 D 型正反器，每個 D 型正反器僅能儲存 1 Bits 的資料，所以一個 32 Bits 的資料需要 32 個 D 型正反器構成。

圖 B-2 基本暫存器輸出入接腳規劃

三、暫存器堆之輸出入接腳規劃及說明：如圖 B-3 所示，並說明如下。

暫存器堆由 32 個 32 Bits 暫存器所組成，暫存器位址為 00-31。可同時指定 2 組暫存器做資料輸出，並成為算術邏輯單元之兩個運算元輸入端，亦可設定寫入資料至指定的暫存器。通常暫存器 0($0)、暫存器 1($1)不使用，永遠儲存 0x00000000 的內容。往後下載至實驗平台時，為了能方便由外部輸出裝置觀察 MIPS CPU 內部 Register File 每個暫存器的暫存資料，所以再設定一組暫存器輸出埠觀察之。

(一)輸入：

1. RESET：重置。重設暫存器堆，當 RESET=1 時，將所有 32 個暫存器內容皆設為 0x00000000。
2. CLOCK：時脈輸入。採用 CLOCK 正緣觸發動作。
3. REGA_ADDR[4:0]：指定暫存器 A 的位址。

4. REGB_ADDR[4:0]：指定暫存器 B 的位址。
5. WREG_WE：暫存器寫入致能。當 WREG_WE=1 時，允許寫入資料至指定暫存器。
6. WREG_ADDR[4:0]：寫入暫存器選擇。選擇欲寫入的暫存器位址，搭配 WREG_WE、WREG_DIN[31:0]執行指定暫存器資料寫入。
7. WREG_DIN[31:0]：寫入暫存器之資料輸入端。
8. REG_SEL[4:0]：暫存器選擇。

(二)輸出：

1. REGA_DOUT[31:0]：暫存器 A 資料輸出端。輸出由 REGA_ADDR 所指定之暫存器內容。
2. REGB_DOUT[31:0]：暫存器 B 資料輸出端。輸出由 REGB_ADDR 所指定之暫存器內容。
3. REG_DOUT[31:0]：暫存器資料輸出端。輸出由 REG_SEL 所指定之暫存器內容。在與外部 I/O 模組相接時，能由外部 I/O 模組隨時觀看指定暫存器之資料內容。

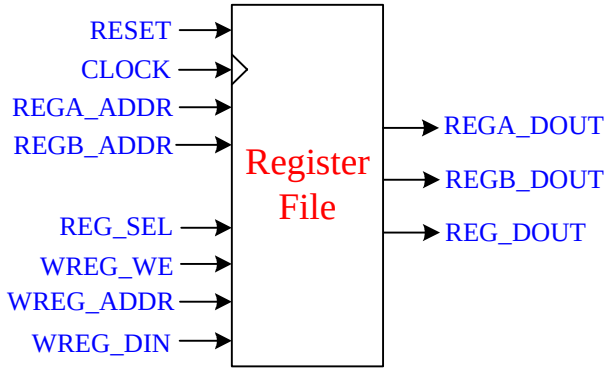


圖 B-3 暫存器堆輸出入接腳規劃

四、暫存器堆內部電路方塊圖如圖 B-4 所示。暫存器堆資料讀取輸出在實際電路中，32 個暫存器是同時將所存資料送出，因此在內部規劃三個多工器，由暫存器位址選擇(REGA_ADDR、REGB_ADDR、REG_SEL)，將所指定的資料送至外部連接埠。REGA_DOUT、REGB_DOUT 主要是 MIPS CPU 內的 ALU 模組的兩個運算資料來源。REG_DOUT 則是接至外部 I/O 裝置顯示用，便於由硬體顯示電路觀察 MIPS CPU 內部 Register File 的資料內容。

寫入 Register File，是利用一個 5 對 32 的解碼電路配合暫存器資料寫入致能(WREG_EN)及寫入暫存器選擇(WREG_ADDR)產生 32 個暫存器寫入致能信號(WEN0~WEN31)，完成指定暫存器資料寫入功能。

讀取 Register File 的輸出-輸入控制信號，如表 B-3 所示，表 B-4 則為寫入 Register File 的輸出-輸入控制信號。整合模組輸出入接腳及輸出入控制信號表，設計 Register File 之 VHDL 程式。

五、軟體模擬結果：如圖 B-38 所示。

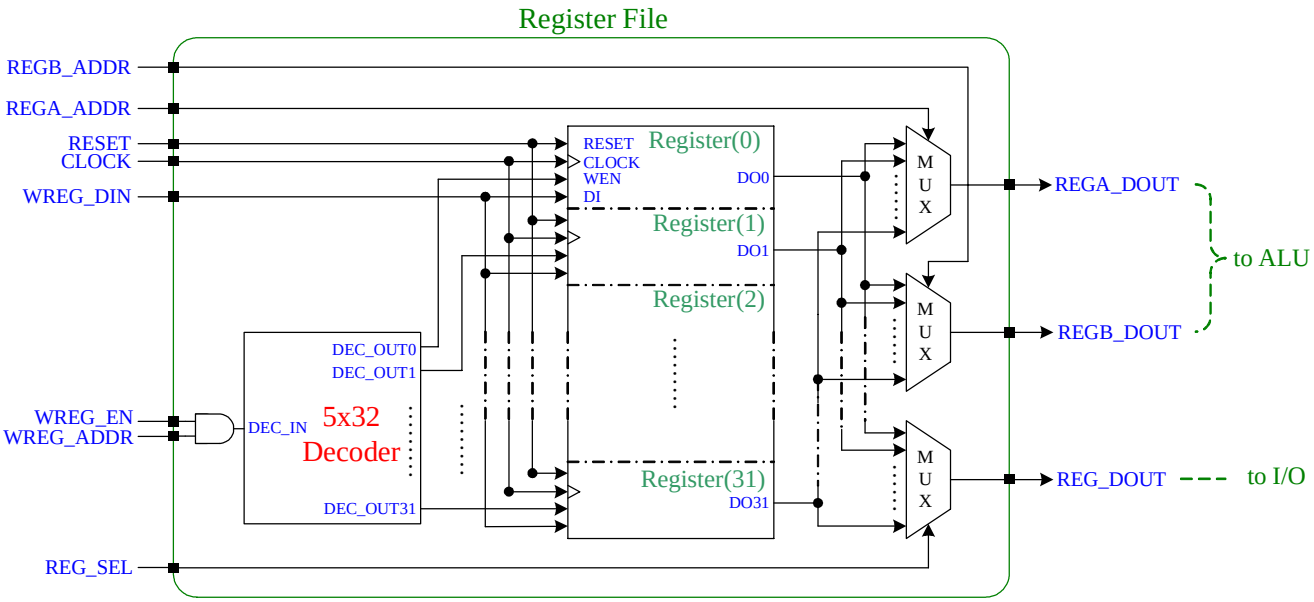


圖 B-4 暫存器堆模組內部電路方塊圖

表 B-3 Register File “Read” Input-Output Table

Input						Output			
RESET	CLOCK	WREG_WE	REGA_ADDR[4:0]	REGB_ADDR[4:0]	REG_SEL[4:0]	REGA_DOUT[31:0]	REGB_DOUT[31:0]	REG_DOUT[31:0]	備註
1	X	X	X	X	X	0x00000000	0x00000000	0x00000000	REG_FILE[31:0] =0x00000000
0	L	X	X	X	X	0x00000000	0x00000000	0x00000000	
	H	X	X	X	X	0x00000000	0x00000000	0x00000000	
	↓	X	X	X	X	0x00000000	0x00000000	0x00000000	
	↑	0	REGA_ADDR	REGB_ADDR	REG_SEL	REG_FILE (REGA_ADDR)	REG_FILE (REGB_ADDR)	REG_FILE (REG_SEL)	
		1	X	X					

表 B-4 Register File “Write” Input-Output Table

Input				Output
RESET	CLOCK	WREG_WE	WREG_ADDR[4:0]	REG_FILE
1	X	X	X	REG_FILE[31:0]= 0x00000000
0	L	X	X	X
	H	X	X	X
	↓	X	X	X
	↑	0	X	X
		1	WREG_ADDR	REG_FILE(WREG_ADDR) = WREG_DIN

3-2 記憶體與程式計數器

3-2-1 記憶體(Memory)

一、相關知識：記憶體是儲存程式碼及資料的所在，它提供使用者大量的二進位資訊，暫時或永久儲存之用。這些二進位資訊必須能由記憶體被取出，而送到各暫存器或其他邏輯電路所組合而成的硬體，待處理完畢後再將結果送回原來的記憶體或其他的記憶裝置儲存。此外，輸入與輸出裝置亦可以與記憶體交互聯繫。來自輸入裝置的資訊可以放在記憶體內，以便它能用來加以處理運作。而處理硬體的輸出資訊也可以放在記憶體內，再從記憶體送至輸出裝置。

記憶體規劃可分為兩大類，一是將儲存程式的程式記憶體與儲存資料的資料記憶體分開設計；另一種則是將程式記憶體與資料記憶體一起規劃於相同的記憶體中。在本設計當中規劃一塊 256x32 Bits 大小的記憶區塊，記憶位址 000 至 127 為程式儲存區段；記憶位址 128-255 為資料儲存區段。

記憶體設計本身將會使用到非常多的 FPGA 內部 Gate Count 數，若要設計非常大容量的記憶體，也會因為 FPGA 本身的 Gate Count 數而受限，因此解決的方法有 2 種：一為利用 FPGA 內部提供的 Block Memory，其優點為不消耗 FPGA 本身提供的 Gate Count 數，但所提供的記憶容量仍舊有限；另一為使用外部記憶體 IC 來輔助。以下的設計實例，乃是採用 Block Memory 的設計方式。首先必須先使用 Xilinx ISE v6.2 中的 Core Generator 方式產生記憶區塊元件，產生的檔案為「檔名.xco」，再利用 VHDL 程式將記憶元件引入。

二、Core Generator 相關知識：Xilinx ISE Core Generator 啟動方式為 Project →New Source→IP(CoreGen)。

Core Generator 提供 2 種 Block RAM，分別為 Single-Port 及 Dual-Port Block RAM，如圖 B-5 所示。在本研究實際設計上，記憶體致能腳(EN)及重設腳(RESET)非為必需，因此在使用 Core Generator 產生 Block RAM 過程中，將 EN、RESET 這 2 個輸入腳 disable 不選。

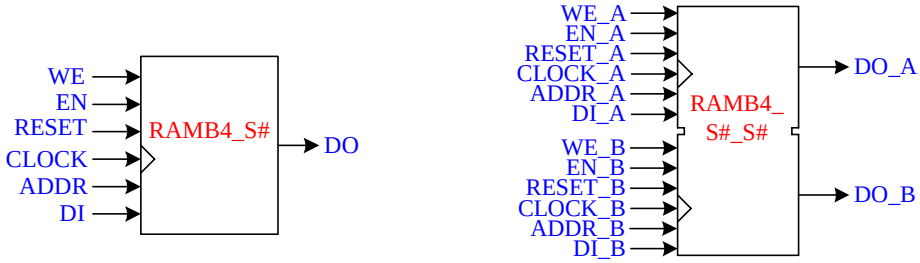


圖 B-5 Single-Port Block RAM 與 Dual-Port Block RAM 原型

三、輸出入接腳規劃及說明：輸出入接腳規劃如圖 B-6 所示。說明如下。

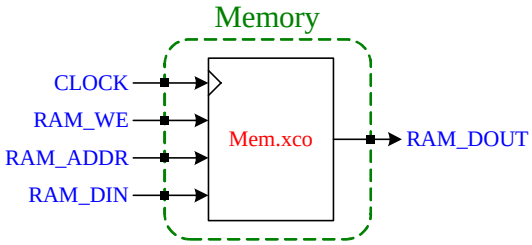


圖 B-6 記憶體輸出入接腳規劃

(一)輸入：

1. CLOCK：時脈輸入。採用 CLOCK 正緣觸發動作。
2. RAM_WE：記憶體寫入致能。當 RAM_WE=1 時，允許寫入資料至記憶體。
3. RAM_ADDR[7:0]：記憶體位址指定。
4. RAM_DIN[31:0]：記憶體寫入資料輸入端。

(二)輸出：

1. RAM_DOUT[31:0]：記憶體資料輸出端。

四、Memory 的輸出-輸入控制信號，如表 B-5 所示。整合模組輸出入接腳及輸出入控制信號表，設計 Memory 之 VHDL 程式。

表 B-5 Memory Input-Output Table

Input				Output
RESET	CLOCK	RAM_WE	RAM_ADDR [7:0]	Function
1	X	X	X	RAM_DOUT[31:0] = 0x00000000
0	L	X	X	RAM_DOUT[31:0] = 0x00000000
	H	X	X	RAM_DOUT[31:0] = 0x00000000
	↓	X	X	RAM_DOUT[31:0] = 0x00000000
	↑	0	RAM_ADDR	RAM_DOUT[31:0] = RAM(RAM_ADDR)
		1	RAM_ADDR	RAM(RAM_ADDR) = RAM_DIN[31:0]

3-2-2 程式計數器(Program Counter)

一、相關知識：為了能夠依序執行指令，必須知道被執行指令的位址所在，而位址則是由程式計數器(Program Counter, PC)提供。為了某些決策狀態，PC 通常需要具備由外界載入位址的能力，才能根據某些狀態而隨意改變運算執行順序。程式中的指令儲存於記憶體的連續位址。當程式執行時，指令一個接一個從記憶體被讀取與執行。每次有一個指令從記憶體被讀取時 PC 即遞增，故其內含值即為下一個指令的位址所在。另一方面，當執行一個程式流程控制指令時。也可以改變 PC 內容，使程式流程被更改[13]。

在 Multiple Clock Cycles MIPS CPU 中，程式計數器並非獨立的計數元件，而是利用 ALU 單元來進行記憶體位址的運算。由 MIPS CPU 架構圖中可看到(如圖 4-10 所示)，新 PC 值的載入係由一個 PC 控制模組(PC Controller)產生的控制訊號 PC_LOAD 決定。在本設計中並未將程式計數器獨立設計為一個模組，而是在將各別模組整合為資料路徑(Data Path)模組後，以幾行 VHDL 並行(concurrent)的行為化描述成一組合邏輯電路，產生 PC Controller 的輸出控制訊號控制 PC Buffer，即可決定新的 PC 值。

二、輸出入接腳規劃及說明：輸出入接腳規劃如圖 B-7 所示。說明如下。

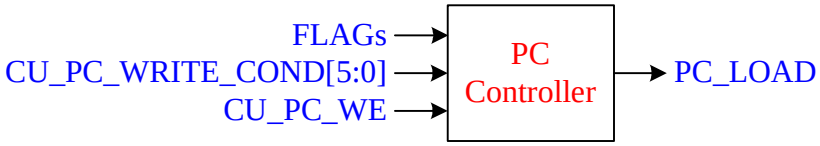


圖 B-7 PC Controller 輸出入接腳規劃示意圖

(一)輸入：

1. FLAGs：運算旗標輸入端。執行條件分支 B-Type 指令(BEQ、BGEZ、BGTZ、BLEZ、BLTZ、BNE)時，決定 Branch 與否，判斷條件即是 ALU 運算後產生的各種旗標，判斷結束再選擇 Branch 的目標位址。這些旗標包含零值旗標(ZERO_FLAG)、大於旗標(GREAT_FLAG)、等於旗標(EQUAL_FLAG)、小於旗標(SMALL_FLAG)等。

2. CU_PC_WRITE_COND[5:0]：PC 寫入條件輸入端。在本設計中具有 6 個 B-Type 指令，每個 B-Type 指令需有一個位元的判斷條件，以決定是否載入新的 PC 值，因此 PC 寫入條件輸入端具有 6 個位元。CU_PC_WRITE_COND[5:0]與 B-Type 指令關係表如表 B-6 所示。
3. CU_PC_WE：PC 寫入致能。當 CU_PC_WE=1 時，不管其他旗標條件是否成立，PC_LOAD 被設定為 1 輸出，致能 PC Buffer。

(二)輸出：

1. PC_LOAD：PC Controller 輸出。為一組合邏輯電路輸出，用來控制 PC Buffer，決定是否將新的 PC 值載入。

三、由 PC Controller 的輸出入對應表，設計的 PC Controller 組合邏輯電路設計如圖 B-8 所示。

表 B-6 PC Controller Input Table

Condition / Mnemonic		CU_PC_WE_COND[5:0]	FLAGs
BEQ	Branch Equal to Zero	CU_PC_WE_COND[0]=1	ZERO_FLAG=1
BNE	Branch Not Equal to Zero	CU_PC_WE_COND[1]=1	ZERO_FLAG=0
BGEZ	Branch Greater than or Equal to Zero	CU_PC_WE_COND[2]=1	LARGE_FLAG=1 ZERO_FLAG=1
BGTZ	Branch Greater than Zero	CU_PC_WE_COND[3]=1	LARGE_FLAG=1
BLEZ	Branch Less than or Equal to Zero	CU_PC_WE_COND[4]=1	SMALL_FLAG=1 ZERO_FLAG=1
BLTZ	Branch Less than Zero	CU_PC_WE_COND[5]=1	SMALL_FLAG=1

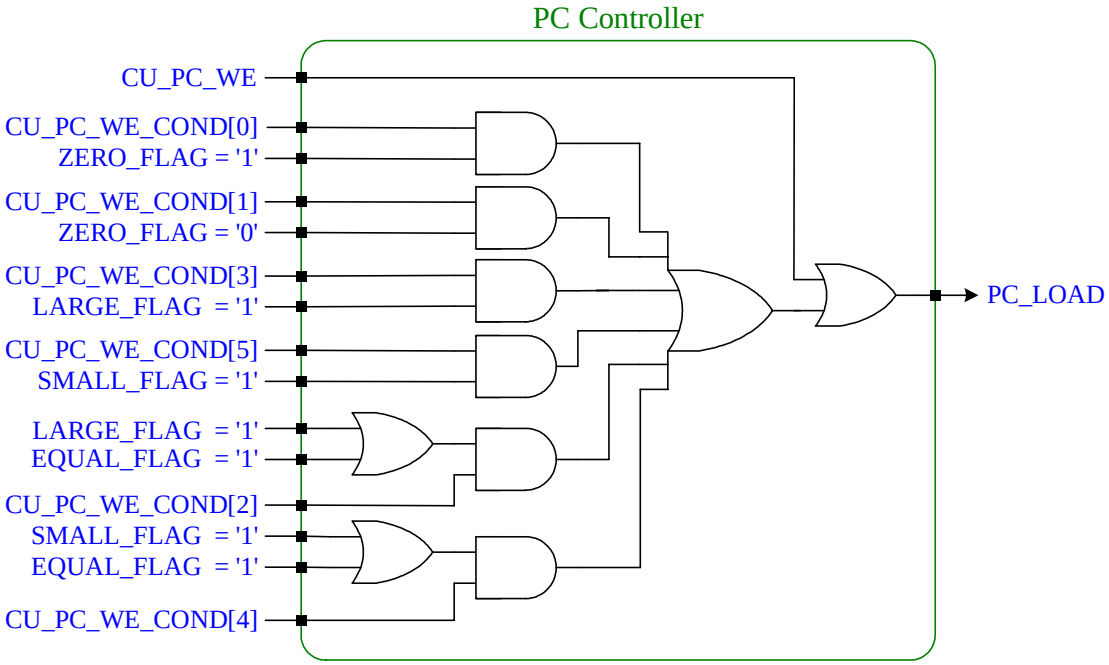


圖 B-8 PC Controller 內部組合邏輯電路設計

3-3 Barrel Shifter

一、相關知識：Barrel Shifter 是一個執行快速移位的模組。移位單元對輸入資料做移位動作，而移位微運算的方式由指令集決定。基本的移位轉換為左移(Shift Left)與右移(Shift Right)，資料載入暫存器後，下一步進行移位轉換，再將結果傳回到目的暫存器中。

因 MIPS CPU 具有邏輯移位及算術移位指令，在本設計中移位指令包含 SLL、SRL、SRA，Barrel Shifter 能在一個 CLOCK 之內即完成 n 位元的移位要求。

二、輸出入接腳規劃及說明：輸出入接腳規劃如圖 B-9 所示。說明如下。

圖 B-9 Barrel Shifter 模組輸出入接腳規劃

(一)輸入：

1. SHIFT_IN[31:0]：移位資料輸入。將進行移位運算的原始資料輸入端。
2. SHIFT_TIMES[4:0]：移位次數指定。依據 MIPS CPU 指令集結構中，SHAMT 移位量指定具有 5 個位元，因此可指定的移位次數為 0-31 次。
3. SHIFT_FUNC[1:0]：移位功能指定。本設計中支援 SLL、SRL、SRA 三種快速移位運算指令，由 SHIFT_FUNC[1:0]指定執行何種移位運算指令。

(二)輸出：

1. SHIFT_OUT[31:0]：移位資料輸出。執行移位運算後之結果輸出端。

三、SHIFT_FUNC[1:0]編碼功能表如表 B-7 所示。由表 B-7 可規劃 Barrel Shifter 內部電路方塊圖，如圖 B-10 所示。在每一次移位資料(SHIFT_DIN[31:0])及移位量(SHIFT_TIMES[4:0])輸入時，內部的三個功能方塊：邏輯左移(Shift Left Logical)、邏輯右移(Shift Right

Logical)、算術右移(Shift Right Arithmetic)，由 SHFIT_FUNC[1:0] 產生的功能方塊致能信號(EN)，致能功能方塊運算輸出，並經由多工器選擇運算結果輸出。

四、軟體模擬結果：如圖 B-39 所示。

表 B-7 SHIFT_FUNC[1:0] Encoding

Shift Mode \ Encoding		SHIFT_FUNC[1:0]	Function
SRL	Shift Right Logical	00	SHIFT_OUT[31:0] = SHIFT_IN srl (SHIFT_TIMES)
SLL	Shift Left Logical	01	SHIFT_OUT[31:0] = SHIFT_IN sll (SHIFT_TIMES)
SRA	Shift Right Arithmetic	10、11	SHIFT_OUT[31:0] = SHIFT_IN sra (SHIFT_TIMES)

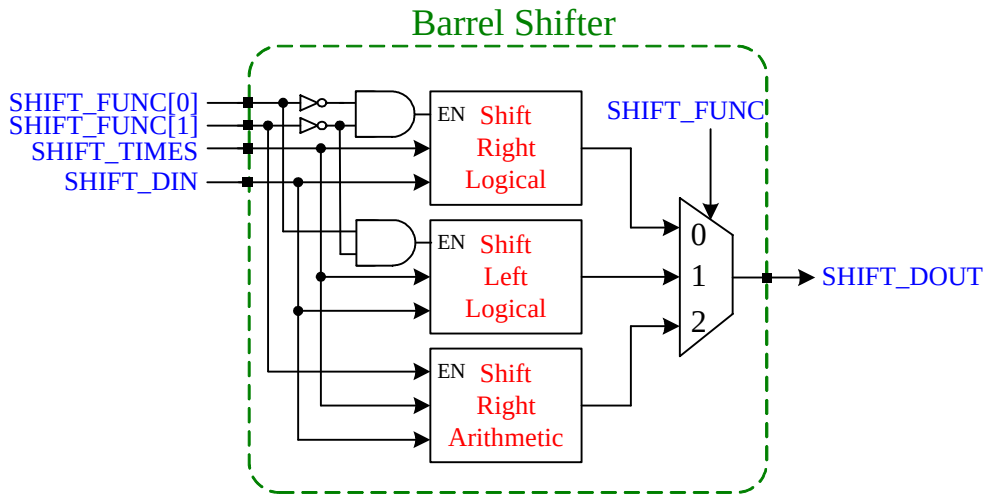


圖 B-10 Barrel Shifter 內部電路方塊圖

3-4 算術邏輯運算單元(ALU)與 ALU 控制單元

3-4-1 算術邏輯運算單元

一、相關知識：算術邏輯運算單元是整個 MIPS CPU 運算執行中心，功能除了一般算術邏輯運算、產生各種運算旗標外，還必須計算新的 PC 值。ALU 模組執行的運算類別有：為算術運算、邏輯運算、移位運算、比較運算。

算術運算的基本元件為並列加法器，由全加器串接而成，可藉由控制加法器的輸入而得到不同的運算，如減法運算在計算機系統中是以轉換為 2's 補數再執行加法運算而得。

邏輯運算是將每一筆資料的每一個位元分別執行邏輯微運算。

移位運算是依所指定的移位量，將輸入資料進行移位微運算。

本設計中運算指令包含 ADD、SUB、AND、OR、XOR、NOR、SLL、SRL、SRA 等 R-Type 及 I-Type 指令，因此需規劃一個控制輸入端，以選擇 ALU 執行的運算子。

二、輸出入接腳規劃及說明：輸出入接腳規劃如圖 B-11 所示，說明如下。

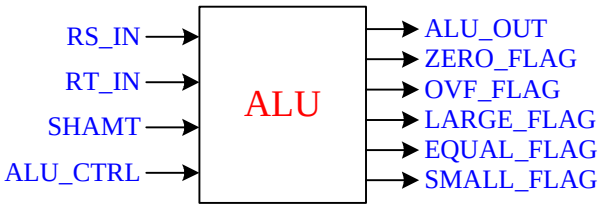


圖 B-11 ALU 模組輸出入接腳規劃示意圖

(一)輸入：

1. RS_IN[31:0]：運算元輸入。來源運算元資料輸入端，由 Register File 輸入。
2. RT_IN[31:0]：運算元輸入。來源運算元資料輸入端，執行 R-Type 運算時，由 Register File 輸入；執行 I-Type 運算時，則為立即值輸入。
3. SHAMT[4:0]：移位次數設定。執行移位運算指令時(SRL、SLL、SRA)之移位次數設定輸入端。

4. ALU_CTRL[3:0]：運算子指定。選擇 ALU 執行運算控制輸入端。ALU_CTRL 設定如表 B-8 所示。

(二)輸出：

1. ALU_OUT[31:0]：運算結果輸出。RS_IN function RT_IN 的運算結果輸出端。
2. ZERO_FLAG：零值旗標。
3. OVF_FLAG：溢位旗標。
4. LARGE_FLAG：大於旗標。
5. EQUAL_FLAG：等於旗標。
6. SMALL_FLAG：小於旗標。

表 B-8 ALU_CTRL[3:0] Encoding

Encoding / ALU Operation	ALU_CTRL[3:0]	Function
ADD	1011	RS_IN + RT_IN
SUB	1100	RS_IN - RT_IN
AND	0010	RS_IN & RT_IN
OR	0011	RS_IN | RT_IN
XOR	0110	RS_IN ^ RT_IN
NOR	0111	~(RS_IN | RT_IN)
SLL	0100	RT_IN sll SHAMT
SRL	0101	RT_IN srl SHAMT
SRA	1000	RT_IN sra SHAMT

四、ALU 內部方塊規劃圖如圖 B-12 所示。基本算術運算(ADD、SUB…)與比較單元(Compare Unit)被歸為基本運算模組(Basic Function)，Barrel Shifter 則使用 3-3 節中所設計的 Barrel Shifter 模組，必須注意的是 Barrel Shifter 模組的資料輸入來源是 RT_IN，與算術運算模組的資料輸入來源不同。另外需再設定一個零值偵測(Zero Detect)及溢位偵測(Overflow Detect)電路，產生零值旗標(ZERO_FLAG)及溢位旗標(OVF_FLAG)，並與比較單元產生的大於(LARGE_FLAG)、小於(SMALL_FLAG)及等於(EQUAL_FLAG)旗標，做為分支指令的條件判斷之用。

五、軟體模擬結果：如圖 B-40 所示。

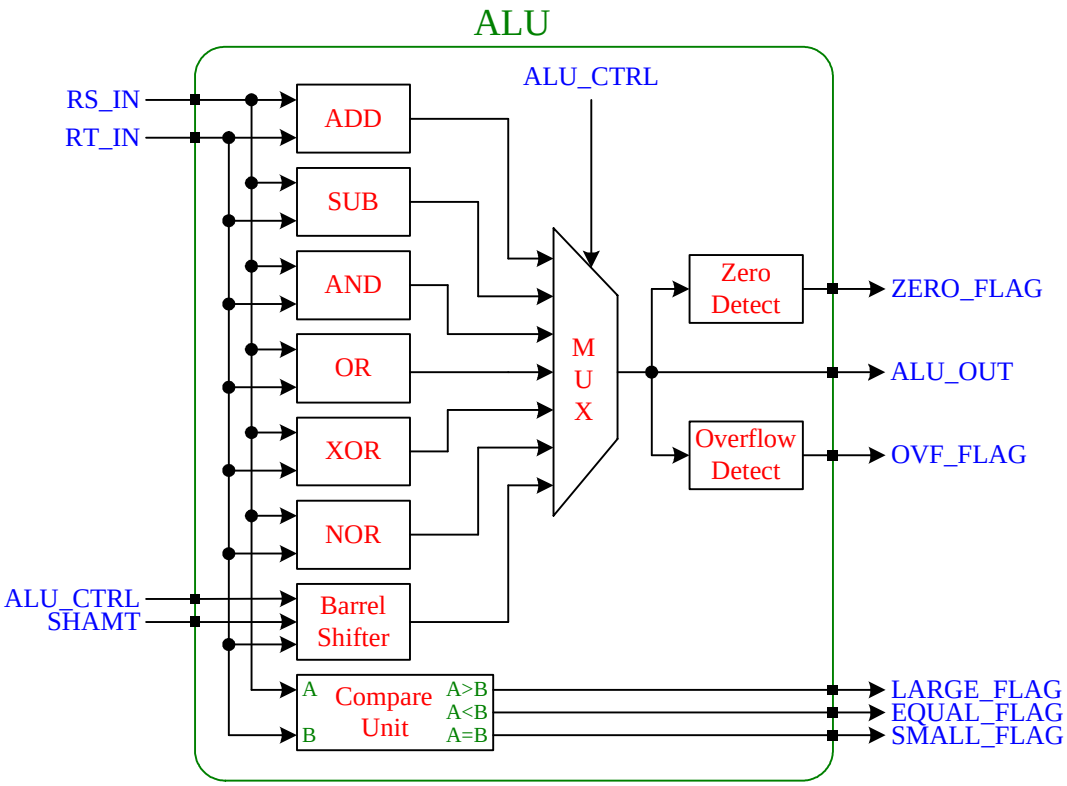


圖 B-12 ALU 內部電路方塊圖

3-4-2 ALU 控制單元

一、相關知識：ALU 控制單元的主要功能在產生選擇 ALU 模組的運算子。由於 ALU 必須執行 R-Type、I-Type 運算外，還須執行 PC 位址計算，如遞增、B-Type 指令，因此需要一個 ALU 控制單元，綜合判斷 R-Type 指令中的 Function Code 和 OP Code，而輸出 ALU 運算子選擇。此模組須與 ALU 模組整合，組成完整 ALU 運算單元。

二、輸出入接腳規劃及說明：輸出入接腳規劃如圖 B-13 所示。說明如下。

圖 B-13 ALU_CTRL 模組輸出入接腳規劃

(一)輸入：

1. ALU_OP[3:0]：ALU 操作碼(OP Code)輸入。ALU_OP 設定如表 B-9 所示。
2. ALU_FUNC[5:0]：ALU 運算功能輸入。MIPS 指令集(如附錄 A)中 R-Type 指令中 Func-Code 欄位輸入。

(二)輸出：

1. ALU_CTRL_OUT[3:0]：ALU 運算子選擇輸出。

三、ALU_OP 及 ALU_FUNC 編碼如表 B-9 及表 B-10 所示。ALU 控制單元依據表 B-9 及表 B-10 進行編碼設計，產生對應的編碼輸出(ALU_CTRL_OUT[3:0])，正確啟動 ALU 模組進行算術邏輯運算。

表 B-9 ALU_OP Encoding

Encoding / Mnemonic or Function		ALU_OP[3:0]
R-TYPE		0010
I-TYPE	ADDI	0011
	ADDIU	0100
	ANDI	0101
	ORI	0110
	XORI	0111
B-TYPE		0001
Program Counter ADD		0000
Program Counter SUB		0001

表 B-10 ALU_FUNC[5:0] Encoding

Encoding / Mnemonic	ALU_FUNC[5:0]
ADD、ADDI、ADDIU	100000
ADDU	100001
SUB	100010
SUBU	100011
AND、ANDI	100100
OR、ORI	100101
XOR、XORI	100110
NOR	100111
SLL	000000
SRL	000010
SRA	000011

四、ALU 與 ALU_CTRL 模組整合方塊圖，如圖 B-14 所示。

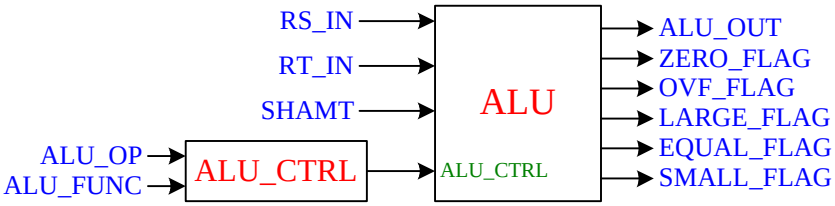


圖 B-14 ALU 與 ALU_CTRL 模組整合方塊圖

3-5 資料路徑整合

一、相關知識：本設計的模組架構圖如圖 4-10 所示，整個 MIPS CPU 由兩大單元資料路徑及控制單元所組成。將先前所設計的各個功能模組，如指令暫存器、暫存器堆、記憶體、ALU 等於資料路徑中整合，即 MIPS CPU 架構圖(如圖 4-10)所示。除了各模組的控制訊號由控制單元產生外，圖 4-10 中的各個資料路徑、多工器選擇、Buffer 設計、PC 控制器(Program Counter Controller)符號擴充單元(Sign Extend)等，都在此處完成。多工器選擇及 PC 控制器與符號擴充單元，可使用 VHDL 中的共時性行為描述來指定；Buffer 則利用 Process 結構做順序性的行為描述指定。而整個設計中需預留各控制訊號輸入端，以接收由控制單元產生的各模組控制訊號或多工器選擇訊號。

二、輸出入接腳規劃及說明：輸出入接腳規劃如圖 B-15 所示。說明如下。

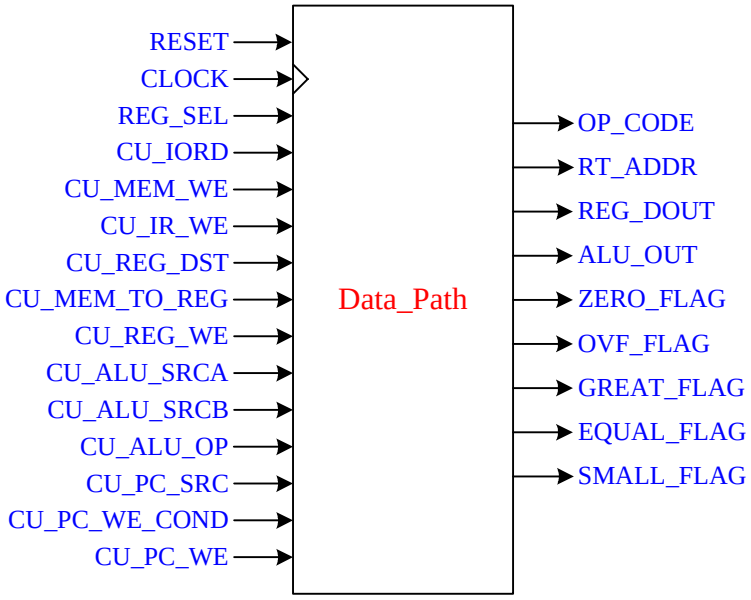


圖 B-15 Data Path 模組輸出入接腳規劃

(一)輸入：

1. RESET：重置。
2. CLOCK：時脈輸入。採用 CLOCK 正緣觸發動作。
3. REG_SEL[4:0]：Register File 選擇輸入。
4. CU_IORD：多工器選擇訊號，指令(Instruction)/資料(Data)指定。
5. CU_MEM_WE：記憶體寫入致能。
6. CU_IR_WE：指令暫存器寫入致能。

7. CU_REG_DST：多工器選擇訊號，目標暫存器指定。執行 R-Type 指令時，Rt 被當作來源暫存器之一、Rd 則為目標暫存器，故 CU_REG_DST 應被設為 1；執行 I-Type 指令時，Rt 則被當作目標暫存器，故 CU_REG_DST 應被設為 0。
8. CU_MEM_TO_REG：多工器選擇訊號，記憶體資料/ALU 運算輸出選擇設定。在執行暫存堆寫入時，寫入資料的來源有二，一為記憶體；二為 ALU 運算的結果。當欲寫入暫存器堆的資料來源為記憶體時，CU_MEM_TO_REG 應被設為 1；欲寫入暫存器堆的資料來源為 ALU 的運算結果時，CU_MEM_TO_REG 應被設為 0。
9. CU_REG_WE：Register File 寫入致能。
10. CU_ALU_SRCA：多工器選擇訊號，PC 值或暫存器內容選擇。先前於 3-4-1 節提到過，ALU 除了一般算術邏輯運算外，亦需負責計算 PC 值，因此由 CU_ALU_SRCA 來選擇是計算 PC 值或計算一般算術邏輯運算。CU_ALU_SRCA=0，ALU 資料來源之一為 PC 值，即為執行新 PC 值的計算；CU_ALU_SRCA=1，ALU 資料來源為暫存器，即為執行一般算術邏輯運算。
11. CU_ALU_SRCB[1:0]：多工器選擇訊號，暫存器內容、遞增及 Sign-Extend 選擇。CU_ALU_SRCB[1:0]=00，為執行 R-Type 指令運算(除移位運算外)，ALU 的另一個資料來源輸入端為暫存器；CU_ALU_SRCB[1:0]=01，執行 PC 值計算，因為 PC 永遠指著下一道指令的位址，所以永遠做遞增+1 的運算；CU_ALU_SRCB[1:0]=10，執行 Unsigned 立即值運算操作，即 I-Type 指令中 ADDIU 指令運算。CU_ALU_SRCB[1:0]=11，執行 Signed 立即值運算操作，即 I-Type 指令中 ADDI、ANDI、ORI、XORI 指令。
12. CU_ALU_OP[3:0]：操作碼設定。
13. CU_PC_SRC[1:0]：多工器選擇訊號，PC 來源選擇。
14. CU_PC_WE_COND[5:0]：PC Buffer 寫入致能條件輸入。相關說明參考 3-2-2 節程式計數器說明。
15. CU_PC_WE：PC 寫入致能。

(二)輸出：

1. OP_CODE[5:0]：操作碼輸出。控制單元依此操作碼輸出，判斷欲執行的指令操作碼並產生正確的控制訊號至各單元模組。
2. RT_ADDR[4:0]：Rt 暫存器位址輸出。
3. ALU_OUT[31:0]：運算結果輸出。
4. REG_DOUT[31:0]：指定暫存器內容資料輸出。
5. ZERO_FLAG：零值旗標。
6. OVF_FLAG：溢位旗標。
7. LARGE_FLAG：大於旗標。
8. EQUAL_FLAG：等於旗標。
9. SMALL_FLAG：小於旗標。

三、緩衝器設計(Buffer)：在資料路徑中，Multiple Clock Cycles MIPS CPU 考慮指令執行步驟，必須將部分資料暫存起來，以等待控制單元的控制排程進行運算，因此在資料路徑中有各種用途的 Buffer，例如 ALU 模組運算輸出 Buffer(ALUOut Buffer)、暫存器 Rs Buffer(A Buffer)、Rt Buffer(B Buffer)、程式計數器 Buffer(PC Buffer)等。基本 Buffer 輸出入接腳規劃如圖 B-16 所示，實際上即為暫存器設計。Buffer 設計的輸出入訊號表如表 B-11 所示。

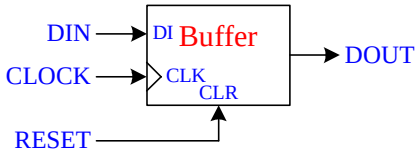


圖 B-16 Buffer 輸出入接腳規劃

表 B-11 Buffer Input-Output Table

Input			Output
RESET	CLOCK	DIN[(n-1):0]	DOUT[(n-1):0]
1	X	X	x"00000000"
0	L	X	x"00000000"
	H	X	x"00000000"
	↓	X	x"00000000"
	↑	X	x"00000000"
		DIN	DIN

四、符號擴充(Sign Extend)單元設計：立即值運算有分帶符號及不帶符號，因此 ALU 的運算來源除了暫存器之外，亦可能有立即值擴充資料的相關運算，因此資料路徑中須再額外設計一個符號擴充單元。Sign Extend 的外部接腳規劃如圖 B-17 所示。內部規劃方塊如圖 B-18 所示。

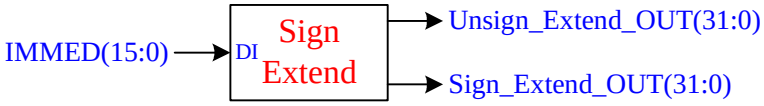


圖 B-17 Sign Extend 輸出入接腳規劃

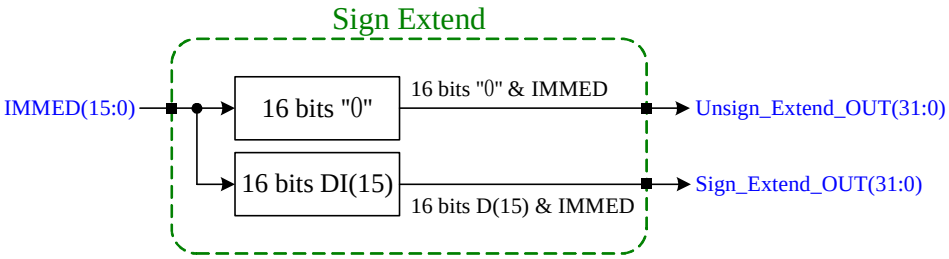


圖 B-18 Sign Extend 內部電路方塊圖

3-6 控制單元(Control Unit)

一、相關知識：控制單元掌控整個計算機系統的運算排程，負責產生所有模組的控制時序、多工器選擇訊號及 Buffer 控制訊號。為了產生一連串的運算，控制單元會送出一連串適當的控制訊號至各個模組，然後再由各個模組取回狀態訊號(Status signals)，這些狀態訊號反應各模組目前的狀態，使控制單元依這些狀態訊號來決定下一個應產生的運算次序。

實作之 MIPS 指令分為 M-Type、R-Type、I-Type、B-Type 及 J-Type 五大類，控制單元接到這些指令的 OP Code 後，即依指令執行順序依序產生控制時序，以確保每一模組在執行週期間的運作正常，產生正確的運算結果。控制單元採用狀態機方式設計，因此設計者必須熟悉每一道指令的運算流程，依此設計出每個狀態及正確的控制訊號。

二、輸出入接腳規劃及說明：輸出入接腳規劃如圖 B-19 所示。說明如下。

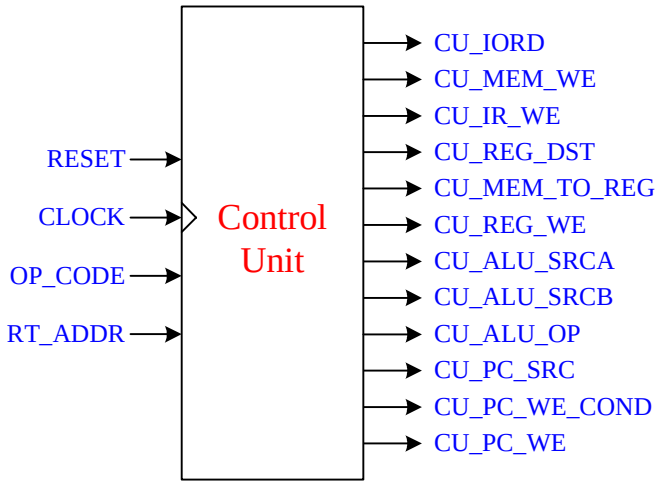


圖 B-19 Control Unit 模組輸出入接腳規劃

(一)輸入：

1. RESET：重置。RESET=1 時，將所有輸出訊號重設為 0。
2. CLOCK：時脈輸入。採用 CLOCK 正緣觸發動作。
3. OP_CODE[5:0]：操作碼輸入。
4. RT_ADDR[4:0]：Rt 暫存器位址輸入。由於 B-Type 部分指令集編碼的關係，如附錄 A 表 A-4 所示。B-Type 部分指令必須將指

令結構中的第 16 至 20 個位元，即 R-Type 指令格式中的 Rt 納入判斷，所以在控制單元預留此部分為輸入端。

(二)輸出：

1. CU_IORD：多工器選擇訊號，指令(Instruction)/資料(Data)選擇設定。
2. CU_MEM_WE：記憶體寫入致能。
3. CU_IR_WE：指令暫存器寫入致能。
4. CU_REG_DST：多工器選擇訊號，目標暫存器指定。
5. CU_MEM_TO_REG：多工器選擇訊號，記憶體資料/ALU 運算輸出選擇設定。
6. CU_REG_WE：暫存器堆寫入致能。
7. CU_ALU_SRCA：多工器選擇訊號，PC 值或暫存器內容選擇。
8. CU_ALU_SRCB[1:0]：多工器選擇訊號，暫存器內容、遞增及 Sign-Extend 選擇。
9. CU_ALU_OP[3:0]：操作碼設定。
10. CU_PC_SRC[1:0]：多工器選擇訊號，PC 來源選擇。
11. CU_PC_WE_COND[5:0]：PC Buffer 寫入致能條件輸入。
12. CU_PC_WE：PC 寫入致能。

三、狀態機設計：

(一)M-Type 指令控制時序

所有 CPU 指令中，通常花費最多時脈週期的是記憶體存取指令中的「Load」指令。由指令提取、指令解碼開始，經記憶體位址計算、指定記憶體位址資料提取、資料回寫至暫存器等歷程。在 MIPS CPU 中的 M-Type 指令，記憶體資料載入(如：LW、LB…)必須花費 5 個時脈週期完成。圖 B-20 為 M-Type 指令的「LW」、「SW」指令實作控制時序圖，說明如下。

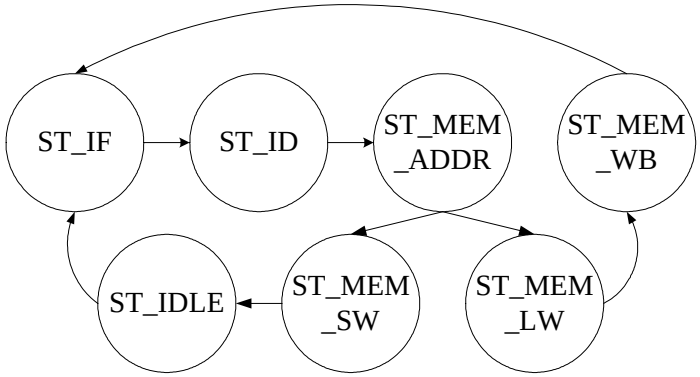


圖 B-20 M-Type 指令控制時序

指令流程敘述：

1. ST_IF：指令提取。由記憶體載入欲執行的指令，並執行記程式計數器遞增計算(PC=PC+1)。

2. ST_ID：指令解碼/暫存器提取。進行指令解碼，並提取指令中所指定的暫存器內容。

3. ST_MEM_ADDR：記憶體位址計算。執行記憶體目標位址計算。M-Type 實作指令有二個，為 SW、LW，在此狀態須再進行 M-Type 進階指令解碼。

4. ST_MEM_LW：LW 指令執行狀態。設定記憶體位址來源為 ALU 的運算結果，由指定的記憶體位址將資料提取出來。

5. ST_MEM_SW：SW 指令執行狀態。設定記憶體位址來源為 ALU 的運算結果，將資料存回指定的記憶體位址。

6. ST_WB：回寫狀態。將 ST_MEM_LW 狀態提取出的資料寫入 Register File 中指定的暫存器。

7. ST_IDLE：由於 FPGA 內部 Block RAM 的特性所致，所設定的特殊狀態。Block RAM 在接收到記憶體位址時，必須在下一個時脈週期進入後，才能將該指定位址之資料送出。

(二)R-Type 指令控制時序

R-Type 指令執行中，除了移位運算指令之外，兩個運算元來源皆為暫存器內暫存之資料，因此在指令格式中會先指定兩個暫存器為運算元

來源(Rs、Rt)。共需耗費 4 個時脈週期，圖 B-21 為 R-Type 指令控制時序圖。

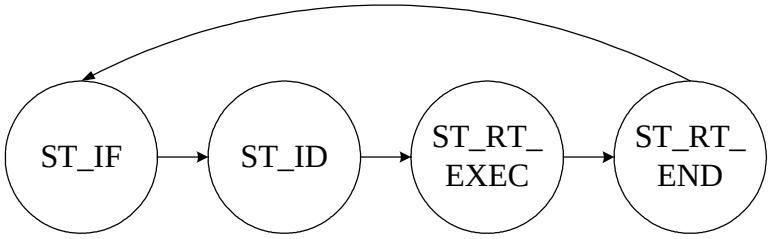


圖 B-21 R-Type 指令控制時序

指令流程敘述：

1. ST_IF：指令提取。由記憶體載入欲執行的指令，並執行程式計數器遞增計算(PC=PC+1)。
2. ST_ID：指令解碼/暫存器提取。提取指令中所指定的暫存器內容。
3. ST_RT_EXEC：R-Type 指令執行狀態。執行 R-Type 指令中的算術邏輯運算指令。
4. ST_RT_END：R-Type 指令執行結束狀態。R-Type 指令運算完成狀態，並將運算結果存回到 R-Type 指令中指定的 Rd 暫存器。

(三)I-Type 指令控制時序

I-Type 指令執行與 R-Type 指令類似，不同點在於 I-Type 指令的第二個運算元來源為指令格式中的立即值輸出，指令執行週期亦與 R-Type 指令相同為 4 個時脈週期，圖 B-22 為 I-Type 指令控制時序圖。

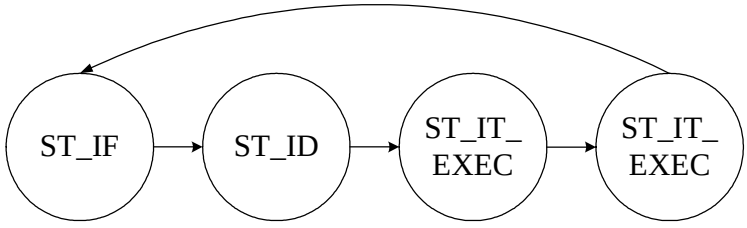


圖 B-22 I-Type 指令控制時序

指令流程敘述：

1. ST_IF：指令提取。由記憶體載入欲執行的指令，並執行程式計

數器遞增計算(PC=PC+1)。

2. ST_ID：指令解碼/暫存器提取。提取指令中所指定的暫存器內容。

3. ST_IT_EXEC：I-Type 指令執行狀態。I-Type 實作指令計有 5 個，在此狀態進行 I-Type 指令進階判斷，決定 ALU 所應執行算術邏輯運算子，並指定 ALU 的第二個運算元輸入的來源為 Unsigned Extend 或 Sign Extend。

4. ST_IT_END：I-Type 指令執行結束狀態。I-Type 指令運算完成狀態，將運算結果回存到 I-Type 指令中指定的 Rt 暫存器。

(四)B-Type 指令控制時序

B-Type 是條件分支指令，設定兩個來源暫存器內容的比較條件，當條件成立或不成立時進行記憶體目標位址計算。圖 B-23 為 B-Type 指令控制時序圖。

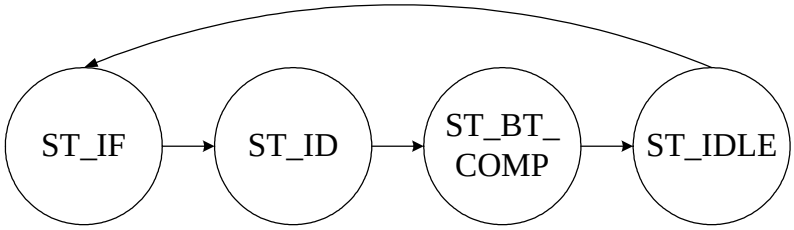


圖 B-23 B-Type 指令控制時序

指令流程敘述：

1. ST_IF：指令提取。由記憶體載入欲執行的指令，並執行程式計數器遞增計算(PC=PC+1)。

2. ST_ID：指令解碼/暫存器提取。提取指令中所指定的暫存器內容，並預先計算 Branch 目標位址，將目標位址計算結果存於 ALUOut Buffer。

3. ST_BT_COMP：B-Type 指令執行狀態。B-Type 實作指令計有 6 個，在此狀態進行 B-Type 指令進階判斷，依指令判斷結果決定 CU_PC_WRITE_COND[5:0]，同時決定下一個 PC 值的來源。

4. ST_IDLE：閒置狀態。等待一個 CLOCK 的時間，取得正確記憶體目標位址內的資料。

(五)J-Type 指令控制時序

J-Type 指令與 B-Type 指令運作流程類似，不同點在於 J-Type 指令為無條件分支指令。圖 B-24 為 J-Type 指令控制時序圖。

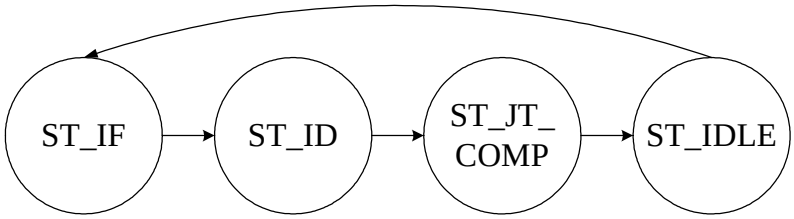


圖 B-24 J-Type 指令控制時序

指令流程敘述：

1. ST_IF：指令提取。由記憶體載入欲執行的指令，並執行程式計數器遞增運算(PC=PC+1)。
2. ST_ID：指令解碼/暫存器提取。提取指令中所指定的暫存器內容；三為預先計算 Branch 目標位址，並將目標位址計算結果存於 ALUOut Buffer 中，
3. ST_JT_COMP：J-Type 指令執行狀態。跳躍目的位址為 J-Type 指令中所指定的跳躍位址(PC=PC[31:28] || IR[25:0])，在這個狀態下須設定 PC 的來源(CU_PC_SRC[1:0])。
4. ST_IDLE：閒置狀態。等待一個 CLOCK 的時間，取得正確記憶體目標位址內的資料。

以上為控制單元各狀態設定的說明，經整合後之控制單元狀態機如圖 4-9 所示。各狀態對應產生的控制訊號列表如表 4-1、表 4-2 所示。

3-7 MIPS CPU 整合

一、相關知識：整個 MIPS CPU 由 2 大單元模組資料路徑及控制單元所組成，在第 3-5 及 3-6 節完成上述 2 個單元模組後，最後完成完整 MIPS CPU 整合電路。

二、輸出入接腳規劃及內部連接方塊圖：如圖 B-25 所示。說明如下。

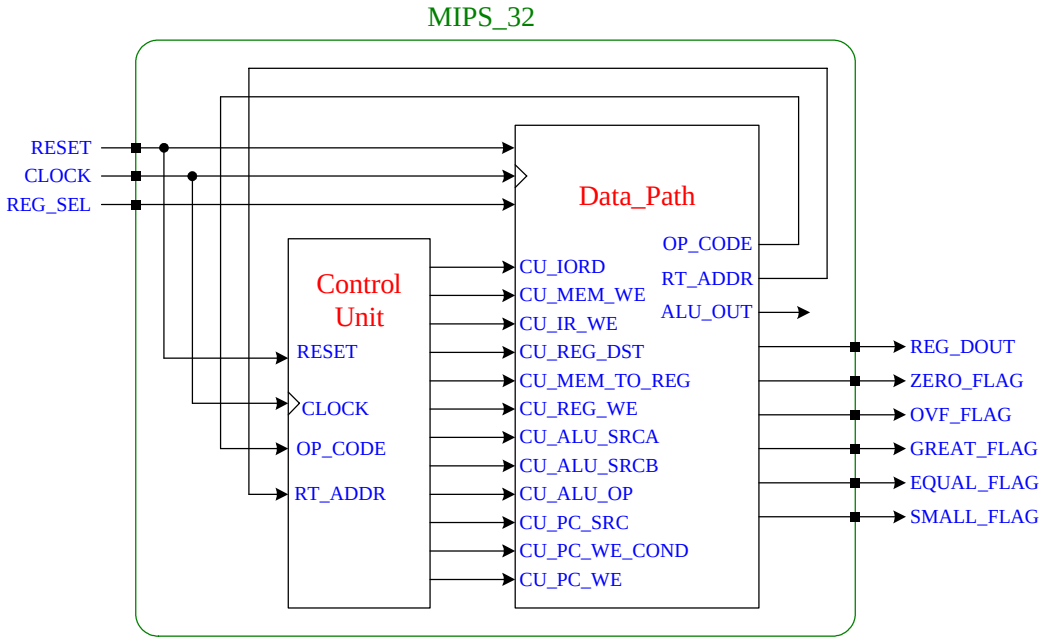


圖 B-25 MIPS CPU 模組內部連接方塊圖

(一)輸入：

1. RESET：重置。RESET=1 時，重設 MIPS CPU 內部所有控制訊號，並將暫存器堆暫存的內容全部清為 0x00000000。
2. CLOCK：時脈輸入。採用 CLOCK 正緣觸發動作。
3. REG_SEL[4:0]：暫存器選擇輸入。依此輸入端選擇 Register File 中任一個暫存器的資料。

(二)輸出：

1. REG_DOUT[31:0]：暫存器資料輸出。
2. ZERO_FLAG：零值旗標。
3. OVF_FLAG：溢位旗標。
4. LARGE_FLAG：大於旗標。
5. EQUAL_FLAG：等於旗標。
6. SMALL_FLAG：小於旗標。

3-8 MIPS CPU 模擬測試

進行指令模擬測試前，須先將測試程式載入記憶體模組中，載入測試程式可使用二種方式：一為利用 Core Generator 介面直接將程式碼輸入；二為預先設計符合 Core Generator 資料格式的測試程式直接將其載入至記憶體，本研究採用第二種方式。測試程式檔案為「檔名.coe」。依指令類別而設計測試程式的優點在於：

一、能在設計初期，依指令類別不同，循序進行 MIPS CPU 設計。例如：先設計僅具 M-Type、R-Type 指令集的 CPU，完成後進行軟體模擬，確認功能無誤後，再慢慢加入 I-Type、B-Type、J-Type 指令，逐漸將 CPU 設計架構擴大。

二、能逐一測試各個指令。若整個設計僅具一個測試程式，常會遺漏某些指令未進行測試，便無法完整保證 CPU 的功能性。

三、能詳細觀測指令動作流程。

以下將對各類別的指令測試程式與模擬結果做說明。

3-8-1 R-Type 指令模擬測試

一、R-Type 指令測試程式如表 B-12 所示。

(一) Memory Address 000-003：LW 指令功能測試。將整個記憶體區分為程式區段及資料區段，LW 指令應能由程式區段跳至某指定的資料區段內擷取資料，並將其儲存至指定暫存器。

(二) Memory Address 004-007：算術運算指令測試。

(三) Memory Address 008：SW 指令功能測試，將指定的暫存器內容存至特定記憶體位址，配合 013 行的 LW 指令驗證 SW 指令功能正確。將$5 的資料存至記憶體 132H 的位址，再存到$11 暫存器。

(四) Memory Address 009-012：移位運算指令測試。

二、軟體模擬結果：如圖 B-41 所示。

表 B-12 R-Type 指令測試程式

INSTRUCTION SEGMENT				
Program		Memory Address	Machine Code	Function
LW	$2,128($0)	000	8C020080	$2 = 37
LW	$3,129($0)	001	8C030081	$3 = 36
LW	$4,130($0)	002	8C040082	$4 = 1D
LW	$9,131($0)	003	8C090083	$9 = F156D640
SUB	$2,$2,$3	004	00431022	$2 = 1
ADD	$4,$2,$4	005	00822020	$4 = 1E
AND	$3,$4,$0	006	00801824	$3 = 0
OR	$5,$2,$4	007	00442825	$5 = 1F
SW	$5,132($0)	008	AC050083	MEM[132] = 1F
SLL	$6,$2,5	009	00023140	$6 = 20
SLL	$8,$2,31	011	000538C2	$8 = 0
SRL	$7,$5,3	010	000247C0	$7 = 3
SRA	$10,$9,3	012	00095103	$10 = FF156D64
LW	$11,132($0)	013	8C0B0083	$11 = 1F
DATA SEGMENT				
		Memory Address	Machine Code	
		128	00000037	
		129	00000036	
		130	0000001D	
		131	F156D640	

3-8-2 I-Type 指令模擬測試

一、I-Type 指令測試，如表 B-14 所示。

(一) Memory Address 000-003：指定記憶體位址並將資料載入記憶體。

(二) Memory Address 004-013：立即值運算指令測試，分別以 Unsigned 立即值運算與 Signed 立即值運算做測試。

二、軟體模擬結果：如圖 B-42 所示。

表 B-13 I-Type 指令測試程式

INSTRUCTION SEGMENT				
Program		Memory Address	Machine Code	Function
LW	$2,128($0)	000	8C020080	$2 = 37
LW	$3,129($0)	001	8C030081	$3 = 36
LW	$4,130($0)	002	8C040082	$4 = 34
LW	$5,131($0)	003	8C050083	$5 = 3
ADDI	$6,$2,FF13	004	2046FF13	$6 = FFFFFF4A
ADDI	$7,$2,0013	005	20470013	$7 = 0000004A
ADDIU	$8,$3,0015	006	24680015	$8 = 0000004B
ADDIU	$9,$3,FF15	007	2469FF15	$9 = 0000FF4B
ANDI	$10,$4,F0F0	008	308AF0F0	$10 = 00000030
ANDI	$6,$4,0F0F	009	30860F0F	$6 = 00000004
ORI	$7,$5,0F0F	010	34A70F0F	$7 = 00000F0F
ORI	$8,$5,F0F0	011	34A8F0F0	$8 = FFFFF0F3
XORI	$9,$2,FFFF	012	3849FFFF	$9 = FFFFFFC8
XORI	$10,$2,0000	013	384A0000	$10 = 00000037
DATA SEGMENT				
		Memory Address	Machine Code	
		128	00000037	
		129	00000036	
		130	00000034	
		131	00000003	

3-8-3 B-Type、J-Type 指令模擬測試

一、B-Type、J-Type 指令測試，如表 B-14 所示。

(一) Memory Address 000-003：指定記憶體位址並將資料載入記憶體。

(二) Memory Address 004-010：B-Type 指令測試，執行$4 暫存器內存值遞增、$5 暫存器內存值遞減。

(三) Memory Address 011：J-Type 指令測試。

二、軟體模擬結果：如圖 B-43 所示。

表 B-14 B-Type、I-Type 指令測試程式

INSTRUCTION SEGMENT				
Program		Memory Address	Machine Code	Function
LW	$2,128($0)	000	8C020080	$2 = 37
LW	$3,129($0)	001	8C030081	$3 = 36
LW	$4,130($0)	002	8C040082	$4 = 34
LW	$5,131($0)	003	8C050083	$5 = 3
SUB	$2,$2,$3	004	00431022	$2 = 1
LOOP1:				
ADD	$4,$2,$4	005	00822020	$4 = $4+1
BNE	$3,$4,LOOP1	006	1464FFFF	
LOOP2:				
BGEZ	$5,LOOP3	007	04A10002	
BGTZ	$5,LOOP4	008	1C400003	
LOOP3:				
SUB	$5, $5,$2	009	00A22822	$5 = $5-1
BNE	$5,LOOP2	010	14A0FFFD	
LOOP4:				
JMP	LOOP5	011	0800000D	
BL TZ	$2,LOOP6	012	04400002	
LOOP5:				
SUB	$2,$2,$2	013	00421022	$2= $2 - $2
LOOP6:				
BLEZ	$2,LOOP5	014	1840FFFE	
DATA SEGMENT				
		Memory Address	Machine Code	
		128	00000037	
		129	00000036	
		130	00000034	
		131	00000003	

由以上 3-8-1、3-8-2、3-8-3 節的測試程式及模擬結果，可確定所實作的 MIPS CPU 各類指令是否能產出預期的結果，確信實作指令運作的正確性。

第四章 I/O 介面控制電路設計

4-1 防彈跳電路

一、相關知識：防彈跳電路是所有按鍵式輸入模組所需嵌入的基本電路，可防止按鍵式輸入模組因按鍵的機械開關不規則彈跳，而取得錯誤的鍵值。

二、輸出入接腳規劃及說明：防彈跳電路輸出入接腳規劃如圖 B-26 所示，並說明如下。

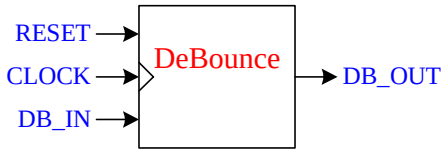


圖 B-26 防彈跳電路模組輸出入接腳規劃

(一)輸入：

1. RESET：重置。
2. CLOCK：時脈輸入。
3. DB_IN：按鍵輸入。由按鍵直接取得的鍵值。

(二)輸出：

1. DB_OUT：防彈跳輸出。

三、內部電路連接圖如圖 5-1 所示。表 B-15 為防彈跳電路之真值表。

表 B-15 Debounce Truth Table

Input				Output
Q3	Q2	Q1	DB_BUF	DB_OUT
0	0	0	0	0
0	0	0	1	0
0	0	1	0	0
0	0	1	1	0
0	1	0	0	0
0	1	0	1	0
0	1	1	0	0
0	1	1	1	1
1	0	0	0	0
1	0	0	1	0
1	0	1	0	0
1	0	1	1	1
1	1	0	0	0
1	1	0	1	1
1	1	1	0	1
1	1	1	1	1

4-2 鍵盤控制電路

一、相關知識：鍵盤可視為多個按鍵組合，其鍵值判定比單個按鍵稍為複雜。以一個 4x4 的鍵盤為例，我們必須先設計一個鍵盤掃描訊號，以 4 個時脈週期的時間，分別送出 4「列」的掃描訊號，再由「行」取得的資料，與掃描訊號做解碼，才能正確取得按鍵值。由以上的敘述可得知，一個鍵盤掃描介面控制電路，基本上須包括：防彈跳電路、掃描訊號產生電路、鍵盤解碼電路三大部分。

二、輸出入接腳規劃及說明：鍵盤控制電路之輸出入接腳規劃及鍵盤對應資料訊號如圖 B-27 所示，並說明如下。

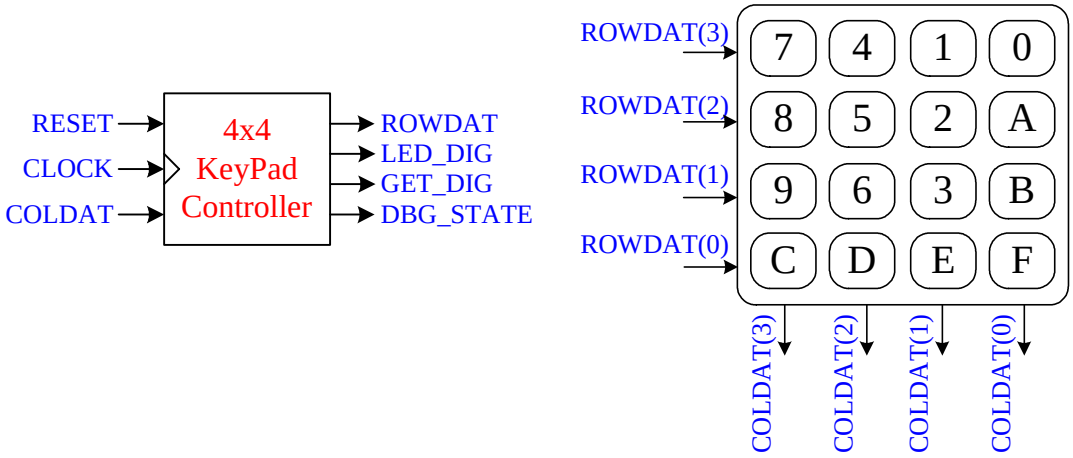


圖 B-27 鍵盤控制電路輸出入接腳及鍵盤對應資料訊號

(一)輸入：

1. RESET：重置。
2. CLOCK：時脈輸入。
3. COLDAT[3:0]：按鍵掃描訊號輸入。由「行」取得的訊號輸入，將被用來與鍵盤掃描訊號(ROWDAT[3:0])搭配鍵盤解碼電路，做按鍵值判斷。

(二)輸出：

1. ROWDAT[3:0]：按鍵掃描訊號輸出。產生鍵盤所需的「列」掃描訊號。
2. GET_DIG：按鍵旗標。當 GET_DIG=1 時，表示鍵盤有某個鍵被

按下。

3. LED_DIG[3:0]:按鍵解碼輸出。由 COLDAT[3:0]與 ROWDAT[3:0]經由組合邏輯電路解碼，送出所按下的鍵值。
4. DBG_STATE[3:0]：指示目前鍵盤掃描電路正在執行的狀態機狀態。

三、內部電路連接方塊圖：參考圖 5-2 的 4x4 鍵盤控制模組內部電路方塊。圖 B-28 為鍵盤控制模組狀態機，各狀態動作說明如下。表-16 為鍵值解碼電路之解碼顯示表。

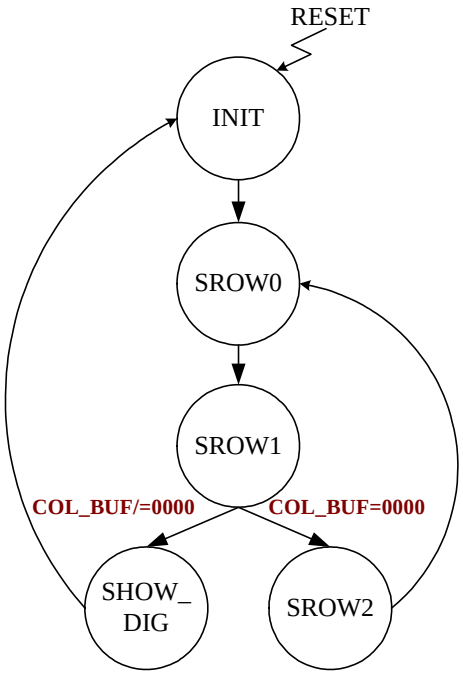


圖 B-28 4x4 鍵盤控制模組狀態機

各狀態執行工作說明如下：

(一)INIT：初始狀態。送出第 1 列鍵盤掃描訊號(ROWDAT[3:0]=1000)。

(二)SROW0：時脈延遲狀態。

(三)SROW1：判斷是否有按下按鍵，即判斷圖 5-2 中的 COLBUF[3:0]。若 COLBUF[3:0]=0000，表示未按下按鍵，若 COLBUF[3:0 若不為 0000，則進入 SHOWDIG 狀態。

(四)SROW2：掃描訊號移位狀態。當上一列的掃描訊號送出，但未取得鍵值時，表示鍵盤上該列未有按鍵被按下，因此再送下一列掃描訊

號。

(五)SHOWDIG：顯示鍵值狀態。表示鍵盤有某個按鍵被按下，送出按鍵旗標(GET_DIG)，表示按下按鍵，並利用組合邏輯電路解碼被按下的按鍵值。

表 B-16 KeyPad Decode Table

COLDAT[3:0]	ROWDAT[3:0]	LED_DIG[3:0]	Number
1000	1000	0111	7
	0100	0100	4
	0010	0001	1
	0001	0000	0
0100	1000	1000	8
	0100	0101	5
	0010	0010	2
	0001	1010	A
0010	1000	1001	9
	0100	0110	6
	0010	0011	3
	0001	1011	B
0001	1000	1100	C
	0100	1101	D
	0010	1110	E
	0001	1111	F

4-2 七段顯示器控制電路

一、相關知識：單顆七段顯示器只需將七段顯示器之 COM 腳(共陽極接電源，共陰極接地)，再送入七段顯示器各節相對信號，即可對單顆七段顯示器作顯示控制。而一顆具有 4 個顯示位元的七段顯示器，各顯示字節是共通的，各個顯示位元的 COM 腳被獨立出來，利用人類視覺暫留的特性，調整時脈週期，對每個 COM 腳輪流送出掃描訊號，在人類眼中看來似乎是 4 個顯示位元都是點亮的狀態。

二、輸出入接腳規劃及說明：由於 FPGA 晶片外部 I/O 接腳有限的狀態(Xilinx XC2S200 具有 140 個 I/O 埠可供規劃)，所以部分外部 I/O 接腳需共接。在 FPGA 實驗平台硬體電路規劃中，七段顯示器配合 74LS47 或 74LS78 作顯示解碼，COM 端掃描訊號產生再利用 74LS138 的 3 對 8 解碼 IC，可順利解出 FPGA 實驗平台中 8 個七段顯示器 COM 端掃描訊號，所以在七段顯示器模組方面，即可省下 8 個 I/O 埠可供其他模組應用，硬體連接圖如圖 5-3 所示。

七段顯示器控制電路模組只需規劃 3 個位元的掃描訊號送至 74LS138，以及 4 個位元的顯示資料送至 74LS47(或 74LS48)即可控制 4 個顯示位元的七段顯示器。其輸出入接腳規劃如圖 B-29 所示並說明如下。

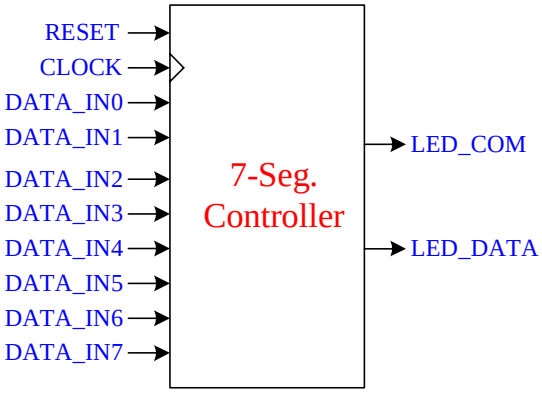


圖 B-29 七段顯示器控制模組輸出入接腳規劃

(一)輸入：

1. RESET：重置。
2. CLOCK：時脈輸入。

3. DATA_IN~DATA_IN7[3:0]：資料輸入埠 0~7。

(二)輸出：

1. LED_COM[2:0]：LED 掃描訊號輸出。
2. LED_DATA[3:0]：LED 顯示資料輸出。對七段顯示器各顯示字節送出顯示訊號。

表 B-17 為七段顯示器控制模組輸出入控制訊號表。

表 B-17 7-Segment Input-Output Table

Input		Output	
RESET	CLOCK	LED_COM[2:0]	LED_DATA[3:0]
1	X	000	"0000"
0	L	000	"0000"
	H	000	"0000"
	↓	000	"0000"
	↑	000	DATA_IN0
		001	DATA_IN1
		010	DATA_IN2
		011	DATA_IN3
		100	DATA_IN4
		101	DATA_IN5
		110	DATA_IN6
		111	DATA_IN7

三、實際下載模組：如圖 B-30 所示。其中 Clock Delay 為硬體時脈延遲電路，與七段顯示器之掃描訊號產生頻率相關。

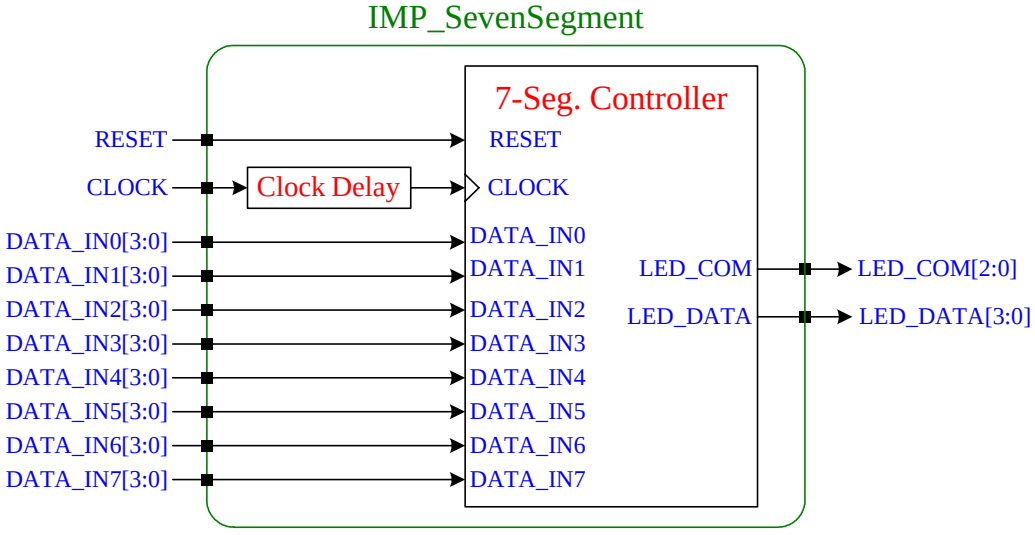


圖 B-30 七段顯示器控制模組實際下載方塊圖

4-4 文字型 LCD 控制電路

一、相關知識：LCD 可概分為文字型與繪圖型兩種，文字型 LCD 僅應用於顯示文字及符號，繪圖型則更可被應用於圖形顯示，相對的繪圖型 LCD 控制方式更為複雜。在本章是以文字型 LCD 當作顯示控制的範例。

常見的 LCD 模組大致有三種規格：16x1、16x2 與 20x2。所使用的控制器編號多為 HD44780，所以控制方式大都相同。FPGA 實驗平台使用 20x2 的 LCD 模組。在 HD44780 的 Data Sheet 上可查到相關的控制時序，依控制時序器對 LCD 作顯示控制，LCD 所接受的顯示字元需為 ASCII Code，在命令 LCD 顯示時須先將 LCD 初始化，即設定 LCD 顯示模式。

HD44780 內有 2 個 8 Bits 暫存器，分別為指令暫存器(Instruction Register, IR)與資料暫存器(Data Register, DR)，由暫存器選擇信號 RS 可選擇命令模式(指令暫存器輸入)或顯示模式(資料暫存器輸入)。相關控制資料可參考 HD44780 的 Data Sheet。

二、文字型 LCD 控制模組包含 2 個子模組，以下就 LCD_CTRL 模組進行說明。LCD_CTRL 模組的功能在於產生 HD44780 控制器的控制時序，包括 LCD 讀寫控制時序、LCD 內部暫存器選擇訊號輸出、LCD 致能訊號、LCD 資料輸出顯示及資料輸入判斷等…。圖 B-31 為 LCD_CTRL 模組輸出入接腳規劃並說明如下。

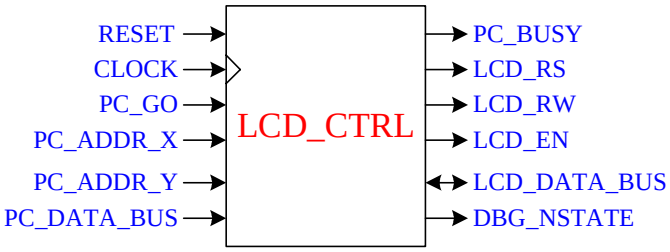


圖 B-31 LCD_CTRL 模組輸出入接腳規劃

(一)輸入：

1. RESET：重置。
2. CLOCK：時脈輸入。
3. PC_GO：啟動，當 PC_GO=1，LCD 模組被致能，開始接受命令

或顯示資料輸入。

4. PC_ADDR_X[4:0]：LCD 模組「行」顯示位置控制。LCD 模組有共 20 行的顯示位置設定，因而需要 5 個位元作為控制輸入。
5. PC_ADDR_Y：LCD 模組「列」顯示位置控制。LCD 模組有 2 列的顯示位置設定。
6. PC_DATA_BUS[7:0]：控制或顯示資料輸入。

(二)輸出：

1. LCD_RS：LCD 模組內部暫存器選擇。LCD_RS=0 選擇 IR，LCD_RS=1 選擇 DR。
2. LCD_RW:LCD 模組讀/寫控制輸出。LCD_RS 與 LCD_RW 搭配，設定 LCD 內部 IR 及 DR 動作狀態。
3. LCD_EN：LCD 模組致能輸出。開始設定前須將 LCD_EN 設定為高準位，設定完畢再將 LCD_EN 設為低準位。
4. PC_BUSY：顯示 LCD_CTRL 模組目前為忙碌狀態，正在執行 LCD 模組控制。
5. LCD_DATA_BUS[7:0]：LCD 資料輸出入匯流排。作 LCD 顯示控制時 LCD_DATA 為輸出埠，輸出設定指令或顯示資料；讀取 LCD 模組狀態時，則為輸入埠。
6. DBG_NSTATE：顯示將被執行的下一個狀態機，使用於軟體功能模擬。

三、LCD_CTRL 狀態機設計：LCD_CTRL 狀態機設計依功能區分，可分為 2 個狀態機，分別為初始及設定狀態機、忙碌檢查及致能狀態機，如圖 B-32 及圖 B-33 所示並說明如下，表 B-18 為各狀態對應控制訊號表。

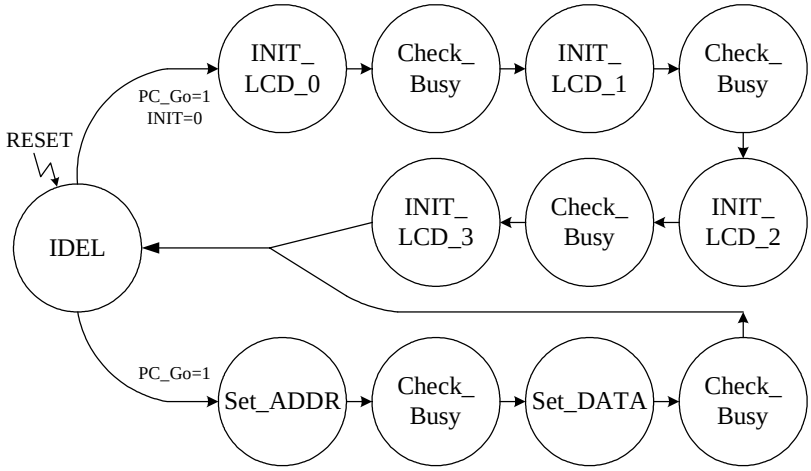


圖 B-32 LCD_CTRL 模組初始及設定狀態機

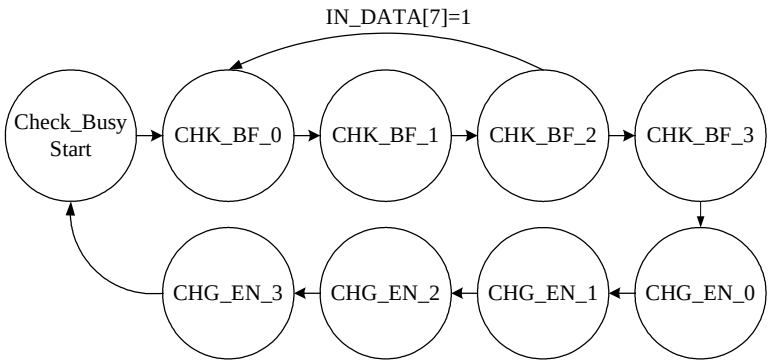


圖 B-33 LCD_CTRL 模組忙碌檢查及致能狀態機

(一)初始及設定狀態機：LCD_CTRL 模組主要控制訊號產生狀態機。

1. IDLE：閒置狀態。LCD 控制模組重置後進入 IDLE 狀態，所有 LCD_CTRL 模組的輸出訊號皆被設為 0。
2. INIT_LCD_0：當 PC_GO=1 且 INITED=0 進入此狀態，送出 LCD 控制指令 38H，設定 LCD 模組為 8 位元且雙行顯示 5x7 點字型。
3. Check_Busy：進入「忙碌檢查及致能狀態機」。
4. INIT_LCD_1：送出 LCD 控制指令 0CH，設定 LCD 顯示器打開、不顯示游標且游標不閃爍。
5. INIT_LCD_2：送出 LCD 控制指令 06H，設定讀/寫資料到 LCD 內部的 Display Data RAM(DD RAM)時計數器會加 1，且資料到 DD RAM 時顯示資料不會左移一格。

6. INIT_LCD_3：送出 LCD 控制指令 01H，設定 DD RAM 所有位址填入空白碼 20H，且將 DD RAM 的 Address Counter 設為 00H、游標向右移動。

7. Set_ADDR：送出 LCD 顯示資料起始位址，並設定 LCD_CTRL 之 PC_BUSY=1。

8. Set_DATA：送出 LCD 顯示資料。

(二)忙碌檢查及致能狀態機：產生 LCD_RS、LCD_RW、LCD_EN 控制訊號。對 LCD 模組做任何控制前須此狀態機。LCD 模組目前處理狀態，LCD 模組忙碌時，必須等待 LCD 模組處理完目前的工作才能再對 LCD 作其他控制。

1. CHK_BF_0：輸出 LCD_RS=0、LCD_RW=1、LCD_EN=0。

2. CHK_BF_1：輸出 LCD_EN=1。

3. CHK_BF_2：若 LCD_DATA_BUS[7]=1 表示 LCD 模組處於忙碌狀態，須等 LCD 處理完目前工作再作 LCD 控制。

4. CHK_BF_3：輸出 LCD_EN=0。

5. CHG_EN_0：當 LCD 模組容許被控制時進入此狀態。當傳送位址或指令時，設定 LCD_RS=0；傳送資料時，設定 LCD_RS=1。

6. CHG_EN_1：設定 LCD_EN=1。

7. CHG_EN_2：設定 LCD_EN=1。

8. CHG_EN_3：設定 LCD_EN=0。

表 B-18 LCD_CTRL 模組各狀態對應控制訊號表

Signal / State	PC_BUSY	LCD_RS	LCD_RW	LCD_EN	CMD_DAT	OUT_DATA[7:0]	INITED
IDLE	0	0	0	0	0	00000000	0
INIT_LCD_0	1	--	--	--	--	00111000	1
INIT_LCD_1	--	--	--	--	--	00001100	--
INIT_LCD_2	--	--	--	--	--	00000110	--
INIT_LCD_3	--	--	--	--	--	00000001	--
SET_ADDR	1	--	--	--	--	'1' & PC_ADDR_Y & '0' & PC_ADDR_X	--
SET_DATA	--	--	--	--	1	PC_DATABUS	--
CHK_BF_0	--	0	1	0	--	OUT_DATA[7:0]	--
CHK_BF_1	--	--	--	1	--	OUT_DATA[7:0]	--
CHK_BF_2	--	--	--	1	--	OUT_DATA[7:0]	--
CHK_BF_3	--	--	--	0	--	OUT_DATA[7:0]	--
CHG_EN_0	--	CMD_DAT	0	0	--	OUT_DATA[7:0]	--
CHG_EN_1	--	--	--	1	--	OUT_DATA[7:0]	--
CHG_EN_2	--	--	--	1	--	OUT_DATA[7:0]	--
CHG_EN_3	--	--	--	0	--	OUT_DATA[7:0]	--

備註：符號「--」代表保留上一個狀態的設定值。

五、LCD_TOP 是上層控制模組，引入 LCD_CTRL，設定 LCD_TOP 模組根據第三章 MIPS CPU 的 REG_SEL[4:0]選擇 Register File 任一個暫存器的資料由 LCD 模組顯示。LCD_CTRL 輸出入接腳規劃如圖 B-34 所示並說明如下。

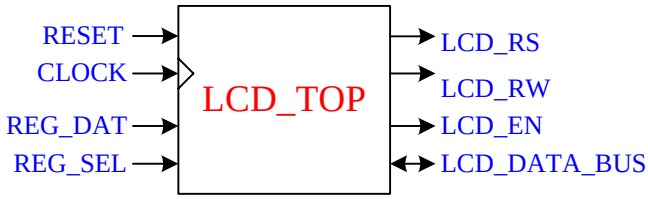


圖 B-34 LCD_TOP 模組輸出入接腳規劃

(一)輸入：

1. RESET：重置。RESET=1，設定圖 5-5 的 LCD_Initial ROM 的內容，如表 B-19 所示。
2. CLOCK：時脈輸入。
3. REG_DAT[31:0]：暫存器資料輸出。
4. REG_SEL[4:0]：MIPS CPU Register File 暫存器選擇。

(二)輸出：

1. LCD_RS：LCD 模組內部暫存器選擇。
2. LCD_RW：LCD 模組讀/寫控制輸出。
3. LCD_EN：LCD 模組致能輸出。
4. LCD_DATA[7:0]：LCD 資料輸出入匯流排。

六、內部電路方塊圖：如圖 5-5 所示。

表 B-19 LCD_Initial ROM 預設資料表

Addr	Data[7:0] (Hex)	ASCII Code 對應字元
0	52	R
1	45	E
2	47	G
3	3A	:
4	44	D
5	41	A
6	54	T
7	3A	:

七、LCD_TOP 狀態機設計：LCD_TOP 狀態機可分為 2 大類，分別為設定指令及資料狀態機、等待及檢查狀態機，如圖 B-35、圖 B-36 所示。表 B-20 為各狀態對應控制訊號表。

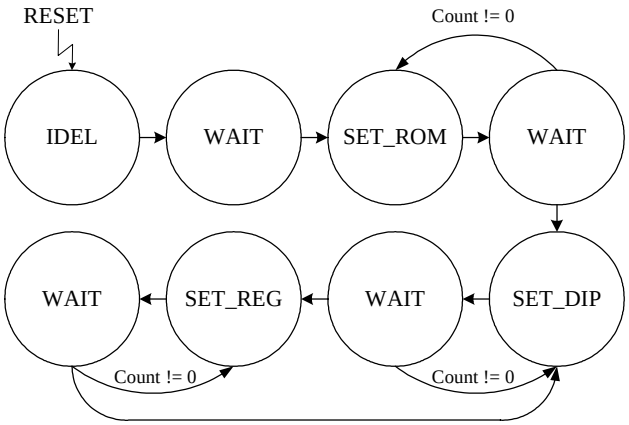


圖 B-35 LCD_TOP 模組設定指令及資料狀態機

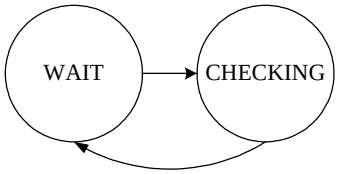


圖 B-36 LCD_TOP 模組等待及檢查狀態機

(一)設定指令及資料狀態機

1. IDLE：閒置狀態。設定 LCD_CTRL 的 PC_GO=1，開始執行 LCD 控制。等待 LCD_CTRL 的 IDLE 狀態。
2. WAIT：進入「等待及檢查狀態機」。
3. SET_ROM：指定 PC_ADDR_X 及 PC_ADDR_Y 給 LCD_CTRL 模組設定資料顯示位址，送出 LCD_Initial ROM 的資料至 LCD。
4. SET_DIP：讀入由外部輸入裝置所設定的暫存器編號，並經由 ASCII Code 轉碼電路，顯示在 LCD 上。
5. SET_REG：經由 ASCII Code 轉碼電路顯示暫存器內存資料。

(二)等待及檢查狀態機

1. WAIT：等待狀態。設定 PC_GO=0。
2. CHECKING：狀態判斷。當 LCD_CTRL 模組為 IDLE 狀態，LCD_Initial ROM、SET_DIP、SET_REG 才能繼續被執行。

表 B-20 LCD_TOP 模組各狀態對應控制訊號表

Signal / State	PC_GO_BUF	COUNT		V_PC_ADDR_X	V_PC_DATABUS
IDLE	0	COUNT-1			
SET_ROM	1	COUNT-1		IF COUNT=3 V_PC_ADDR_X +7 ELSE V_PC_ADDR_X +1	INIT_ROM(7 - COUNT)
SET_DIP	1	COUNT-1		V_PC_ADDR_X +1	V_PC_DATABUS =(0~9)
SET_REG	1	COUNT-1		V_PC_ADDR_X +1	V_PC_DATABUS =(0~9,A~F)
WAITT	0	--		--	V_PC_DATABUS
CHECKING	--	DBG_NSTATE_BUF ="0000" COUNT =0	T0_STATE = SET_ROM THEN 8	V_PC_ADDR_X +1	V_PC_DATABUS
			T0_STATE = SET_DIP THEN 2	V_PC_ADDR_X +1	V_PC_DATABUS
			T0_STATE = SET_REG THEN 8	V_PC_ADDR_X +1	V_PC_DATABUS

備註：符號「--」代表保留上一個狀態的設定值。

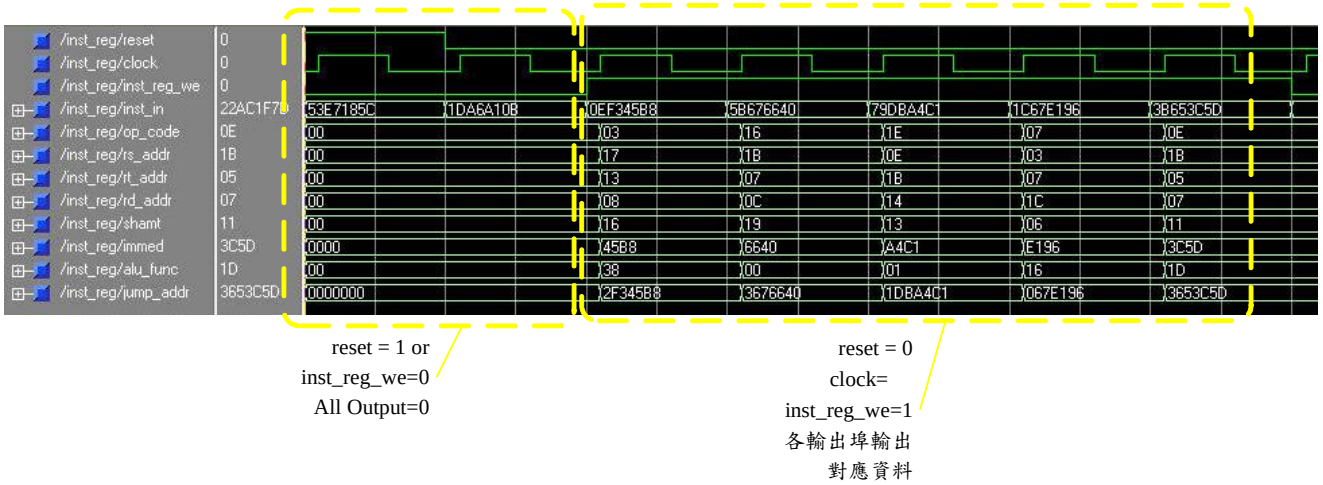


圖 B-37 指令暫存器軟體模擬結果

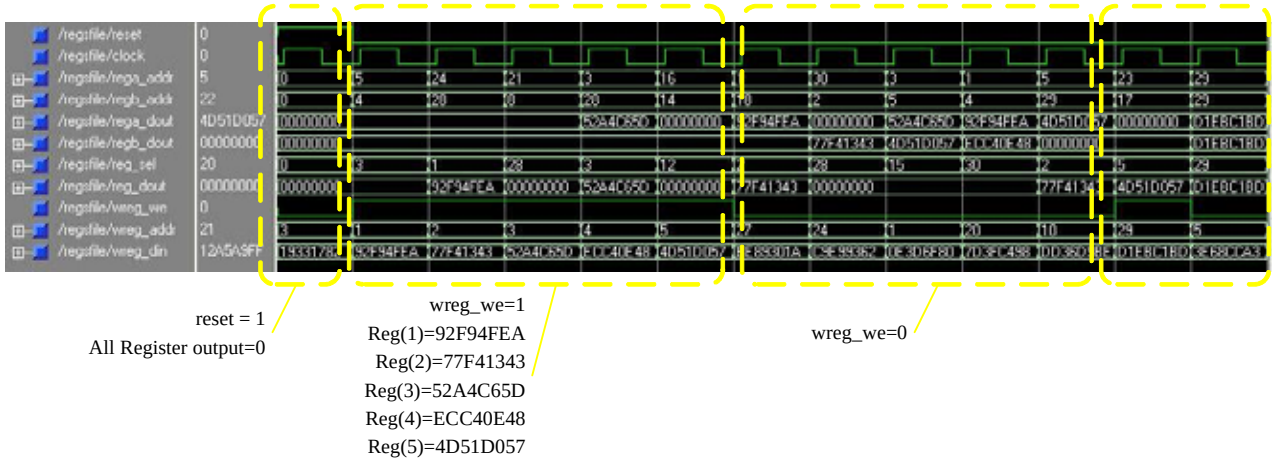


圖 B-38 暫存器堆模擬結果

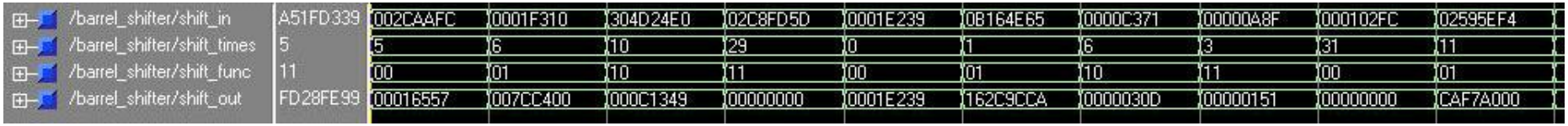


圖 B-39 Barrel Shifter 模擬結果

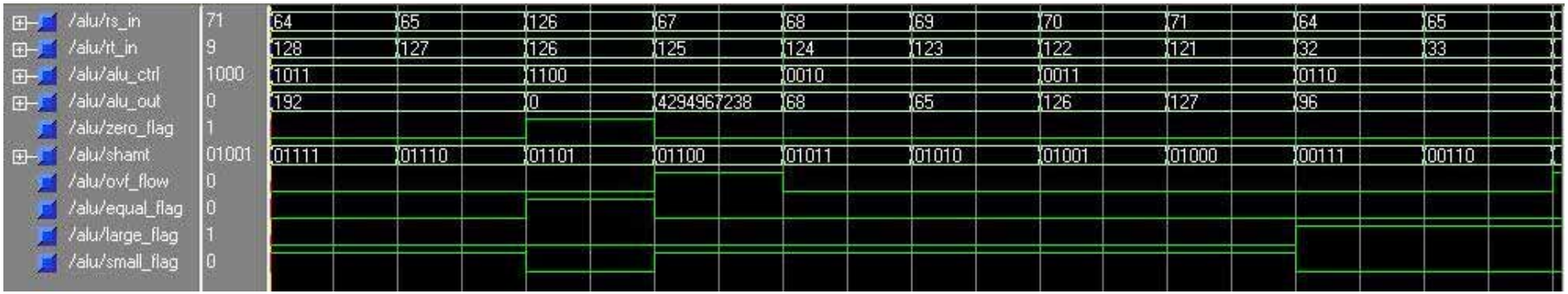
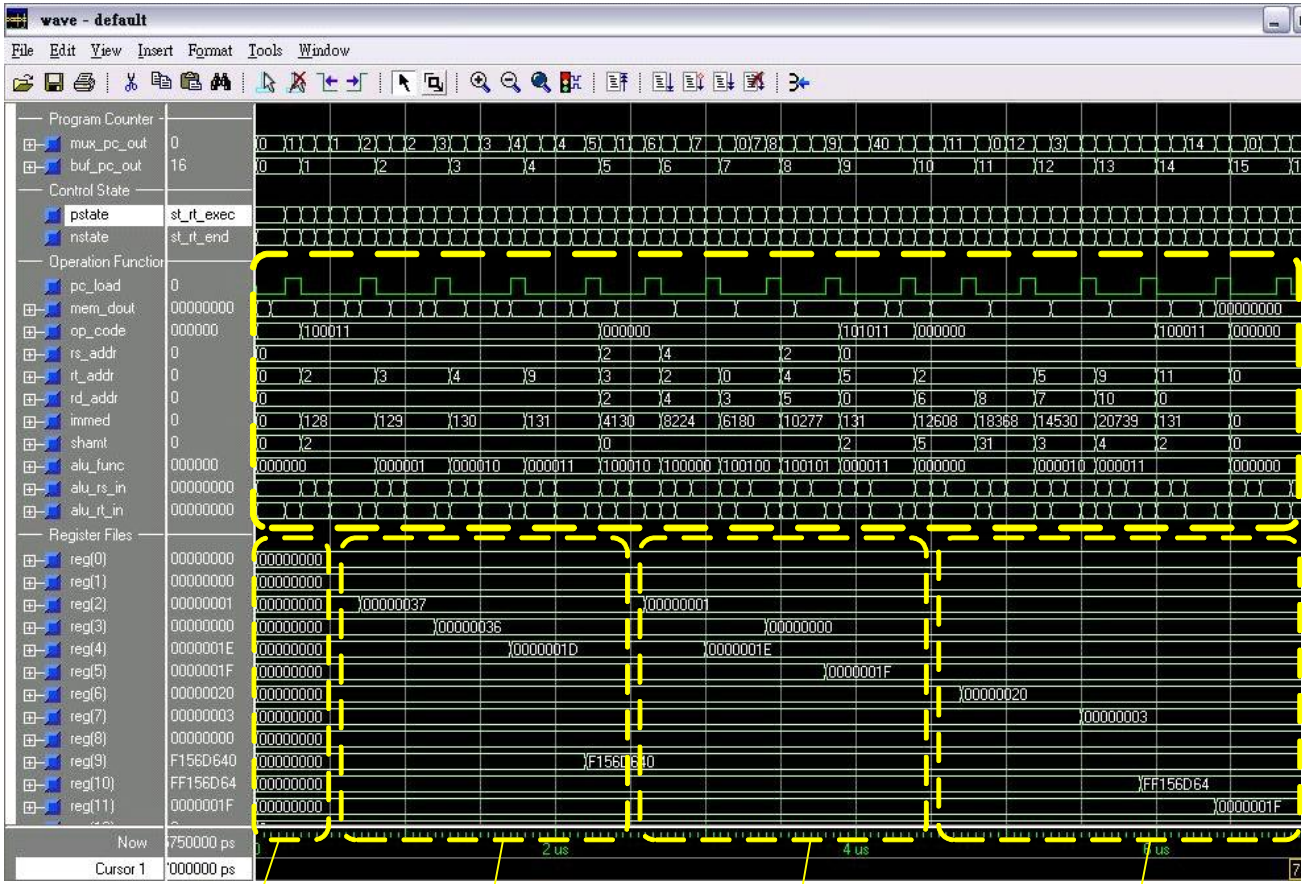


圖 B-40 ALU 模擬結果



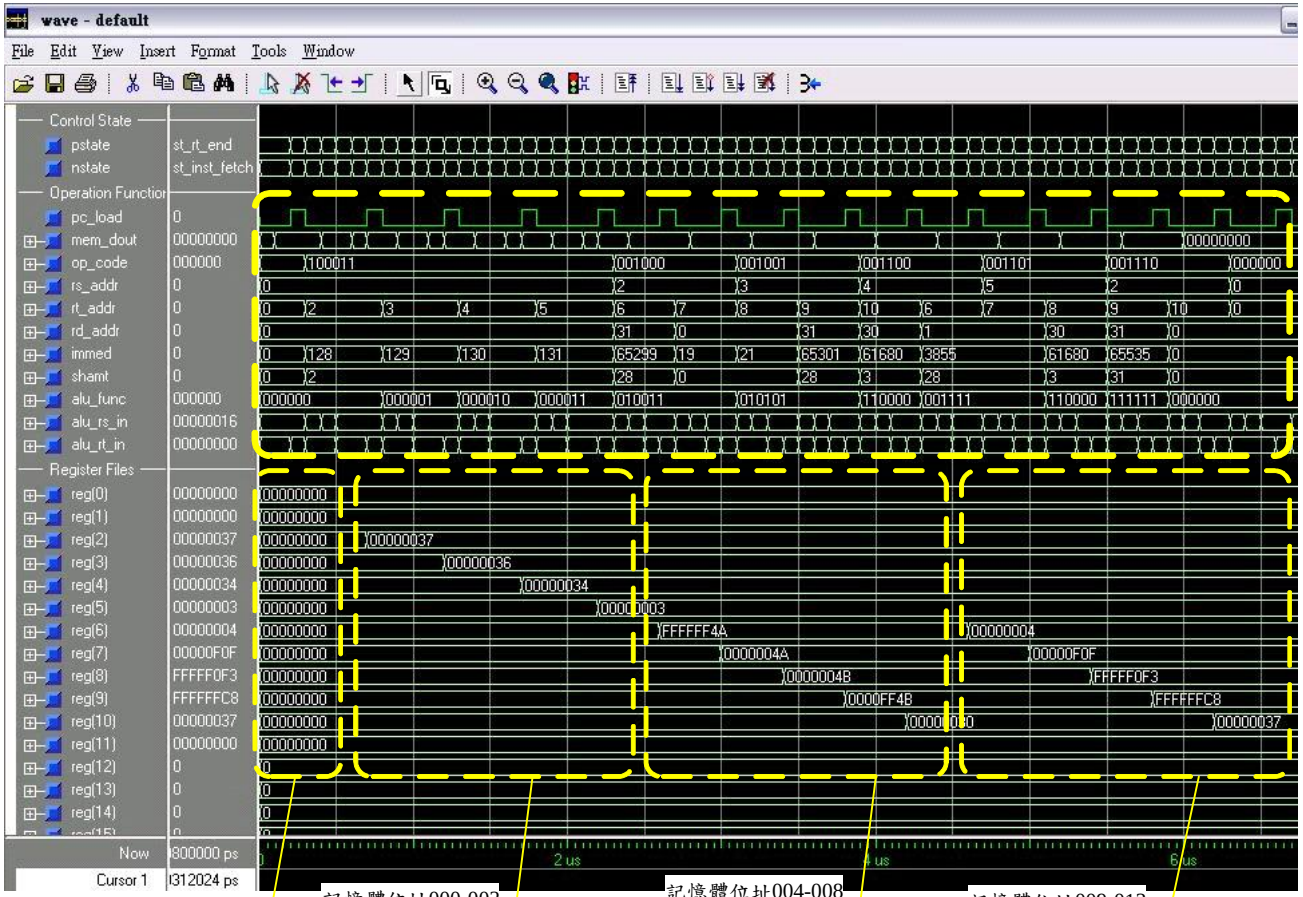
程式啟始狀態
UHVHW@4
所有暫存器內容重設
為33333333K

記憶體位址3330336
指令執行結果
'5@3333336:K
'6@33333369K
'7@3333334GK
'<@I489G973K

記憶體位址337033;
指令執行結果
'5@33333334K
'7@3333334HK
'6@33333333K
'8@3333334IK
PHP^465`@4I

記憶體位址33<0346
指令執行結果
'9@33333353K
';@33333333K
':@33333336K
'43@II489G97K
'44@3333334IK

圖 B-41 R-Type 指令模擬結果



程式啟始狀態
RESET =1
所有暫存器內容重設
為00000000H

記憶體位址000-003
指令執行結果
$2=00000037H
$3=00000036H
$4=00000034H
$9=00000003H

記憶體位址004-008
指令執行結果
$6=FFFFFF4AH
$7=0000004AH
$8=0000004BH
$9=0000FF4BH
$10=00000030H

記憶體位址009-013
指令執行結果
$6=00000004H
$7=00000F0FH
$8=FFFFF0F3H
$9=FFFFFFC8H
$10=00000037H

圖 B-42 I-Type 指令模擬結果

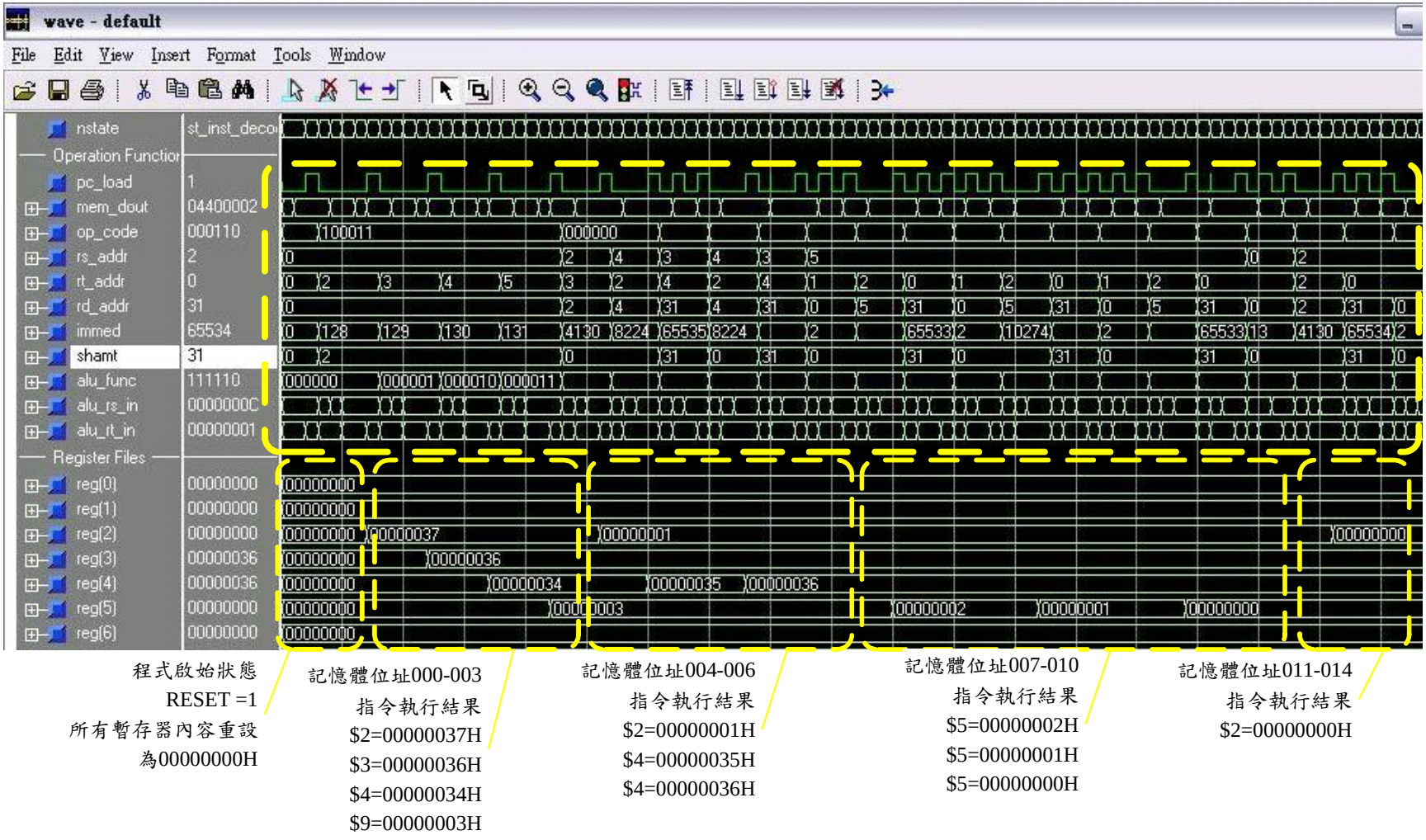


圖 B-43 B-Type、I-Type 指令模擬結果

附錄 C　FPGA 實驗平台實體圖

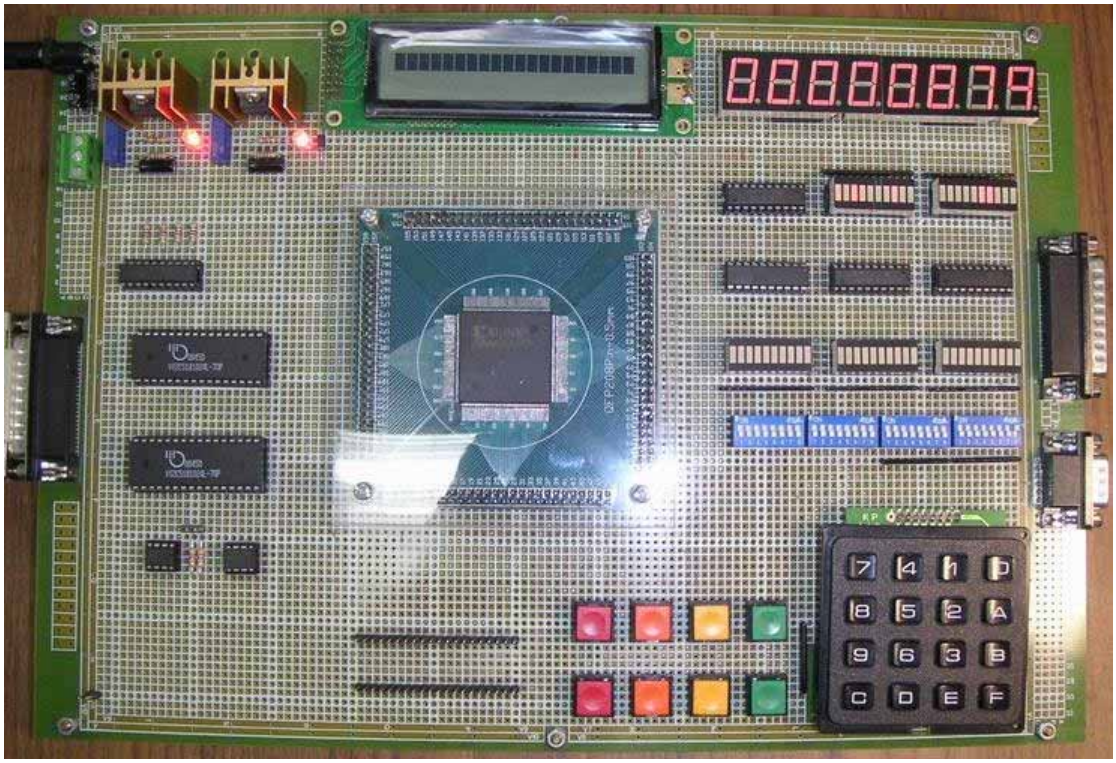


圖 C-1 FPGA 實驗平台實體圖