計算機組織 Lab3

9617145 許晏峻

程式碼：(灰色部分代表之前實作部份)

✧ **Test pattern：**

```
`timescale 1ns / 1ps
module Decoder_test();

    // Inputs
    reg Clk;
    reg Reset;
    reg [31:0] Instruction;

    // Instantiate the Unit Under Test (UUT)
    Decoder uut (
        .Clk(Clk),
        .Reset(Reset),
        .Instruction(Instruction)
    );

    initial begin
        // Initialize Inputs
        Clk = 0;
        Reset = 0;
        Instruction = 0;

        #10 Reset=1;
        #10 Reset=0;
        #20 Instruction=32'b10000000_00000000_00000000_00000000; // (LOAD M0 R0)
        #20 Instruction=32'b01000000_00000000_00000001_00000000; // (STORE M0 R1)
        #20 Instruction=32'b00100000_00000010_00000011_00000001; // (ADD R2 R3 R1)
        #20 Instruction=32'b00001000_00000001_00000010_00000000; // (SL R1 2 R0)

        // Add stimulus here
    end

    always
        #10 Clk = ~Clk;
endmodule
```

✧ Decoder：

```
module Decoder(Clk, Reset, Instruction, Instruction );
    input wire Clk;
    input wire Reset;
    input wire [31:0] Instruction;

    wire RegDst;
    wire MemRead;
    wire MemtoReg;
    wire [1:0]ALUOp;
    wire MemWrite;
    wire ALUSrc;
    wire RegWrite;

    Control    uut1                              // Link to Controler
(.Clk(Clk), .Reset(Reset), .Instruction(Instruction), .RegDst(RegDst), .MemRead(MemRead), .MemtoR
eg(MemtoReg), .ALUOp(ALUOp), .MemWrite(MemWrite), .ALUSrc(ALUSrc), .RegWrite(RegWrite));

    wire [31:0]ReadData1;
    wire [31:0]ReadData2;
    reg [7:0]ReadReg1;
    reg [7:0]ReadReg2;
    reg [7:0]WriteReg;
    reg [31:0]WriteData;

    RegisterFile uut2                            // Link to Register
(.Clk(Clk), .Reset(Reset), .Instruction(Instruction), .RegWrite(RegWrite), .ReadReg1(ReadReg1), .Read
Reg2(ReadReg2), .WriteReg(WriteReg), .WriteData(WriteData), .ReadData1(ReadData1), .ReadData2
(ReadData2));

    wire [31:0]ReadData;
    reg [7:0]Addr;

    DataMemory uut3                              // Link to Memory
(.Clk(Clk), .Reset(Reset), .Instruction(Instruction), .MemWrite(MemWrite), .MemRead(MemRead), .
Addr(Addr), .WriteData(WriteData), .ReadData(ReadData));

    wire [31:0]ALUResult;
    reg [31:0]A,B;

    ALU uut4                                     // Link to ALU
(.Clk(Clk), .Reset(Reset), .ALUOp(ALUOp), .A(A), .B(B), .ALUResult(ALUResult));
```

```
    always@(*)
    begin
        ReadReg1<=Instruction[23:16];
        ReadReg2<=Instruction[15:8];
        Addr<=Instruction[23:16];
        A<=ReadData1;
        if(RegDst==1)                       // rd or rt?
            WriteReg<=Instruction[7:0];
        else
            WriteReg<=Instruction[15:8];
        if(MemWrite ==1)                    // Register to Memory?
            WriteData<=ReadData2;
        else if(MemtoReg ==1)               // Memory Value or ALU result?
            WriteData<=ReadData;
        else
            WriteData<=ALUResult;
        if(ALUSrc==1)                       // Immediate or rt?
            B<={8'b0000_0000,Instruction[15:8]};
        else
            B<=ReadData2;
    end

endmodule
```

✧ Control：

```
module Control(Clk, Reset, Instruction, RegDst, MemRead, MemtoReg, ALUOp, MemWrite, ALUSrc,
RegWrite);

    input wire Clk;
    input wire Reset;
    input wire [31:0]Instruction;

    output reg RegDst;
    output reg MemRead;
    output reg MemtoReg;
    output reg [1:0]ALUOp;
    output reg MemWrite;
    output reg ALUSrc;
    output reg RegWrite;

    always@(*)
    begin
```

```
if(Reset==1)
begin
	MemWrite=0;
	RegWrite=0;
end
else if(Instruction[31:24]==8'b1000_0000)		//LOAD
begin
	RegDst=0;
	MemRead=1;
	MemtoReg=1;
	ALUOp=2'b00;
	MemWrite=0;
	ALUSrc=1;
	RegWrite=1;
end
else if(Instruction[31:24]==8'b0100_0000)		//STORE
begin
	RegDst=0;
	MemRead=0;
	MemtoReg=0;
	ALUOp=2'b00;
	MemWrite=1;
	ALUSrc=1;
	RegWrite=0;
end
else if(Instruction[31:24]==8'b0010_0000)		//ADD
begin
	RegDst=1;
	MemRead=1;
	MemtoReg=0;
	ALUOp=2'b00;
	MemWrite=0;
	ALUSrc=0;
	RegWrite=1;
end
else if(Instruction[31:24]==8'b0001_0000)		//SUB
begin
	RegDst=1;
	MemRead=1;
	MemtoReg=0;
	ALUOp=2'b01;
	MemWrite=0;
```

```
                ALUSrc=0;
                RegWrite=1;
            end
            else if(Instruction[31:24]==8'b0000_1000)                    //ShiftLeft
            begin
                RegDst=1;
                MemRead=1;
                MemtoReg=0;
                ALUOp=2'b10;
                MemWrite=0;
                ALUSrc=1;
                RegWrite=1;
            end
            else if(Instruction[31:24]==8'b0000_0100)                    //ShiftRight
            begin
                RegDst=1;
                MemRead=1;
                MemtoReg=0;
                ALUOp=2'b11;
                MemWrite=0;
                ALUSrc=1;
                RegWrite=1;
            end
        end

endmodule
```

實作說明：

decoder_test.v 檔為 test pattern 負責做 clk,reset 和下指令到 instruction 中。

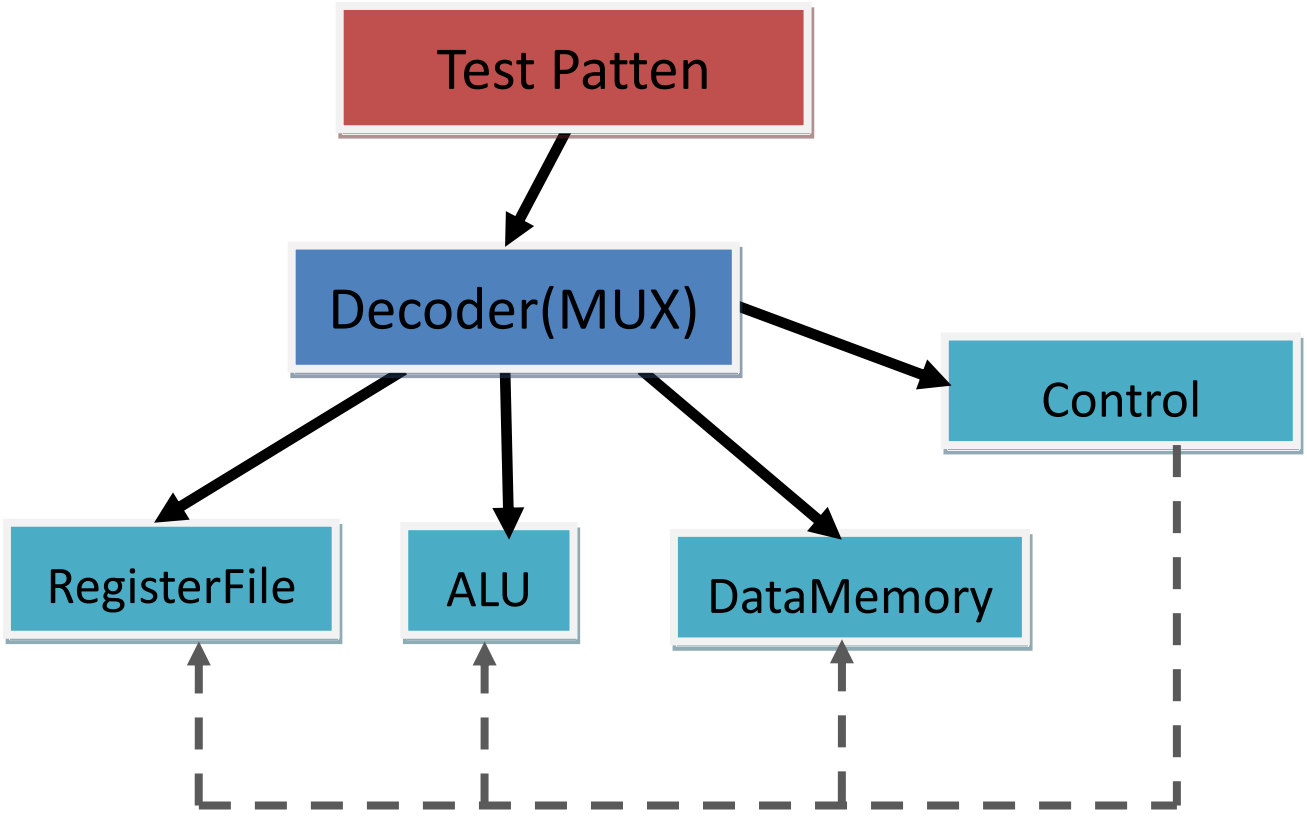
decoder.v 檔則負責在接到指令後做所有 MUX 要做的事，先傳 opcode 給 control，再由各種 enable flag 值決定哪些 data 要放哪些值去做運算和存值，設定好再傳給 ALU、Register、Memory 完成指令。

control.v 負責做後面用到所有 MUX 要判斷的 0,1 值，設成哪些值是根據 decoder 傳進來的 opcode 判斷

RegisterFile.v 接收從 Decoder 來的值做 input，看 enable flag 的值做 read or write。

DataMemory.v 接收從 Decoder 來的值做 input，看 enable flag 的值做 read or write。
(DataMemory 在這次 lab2 沒用到)

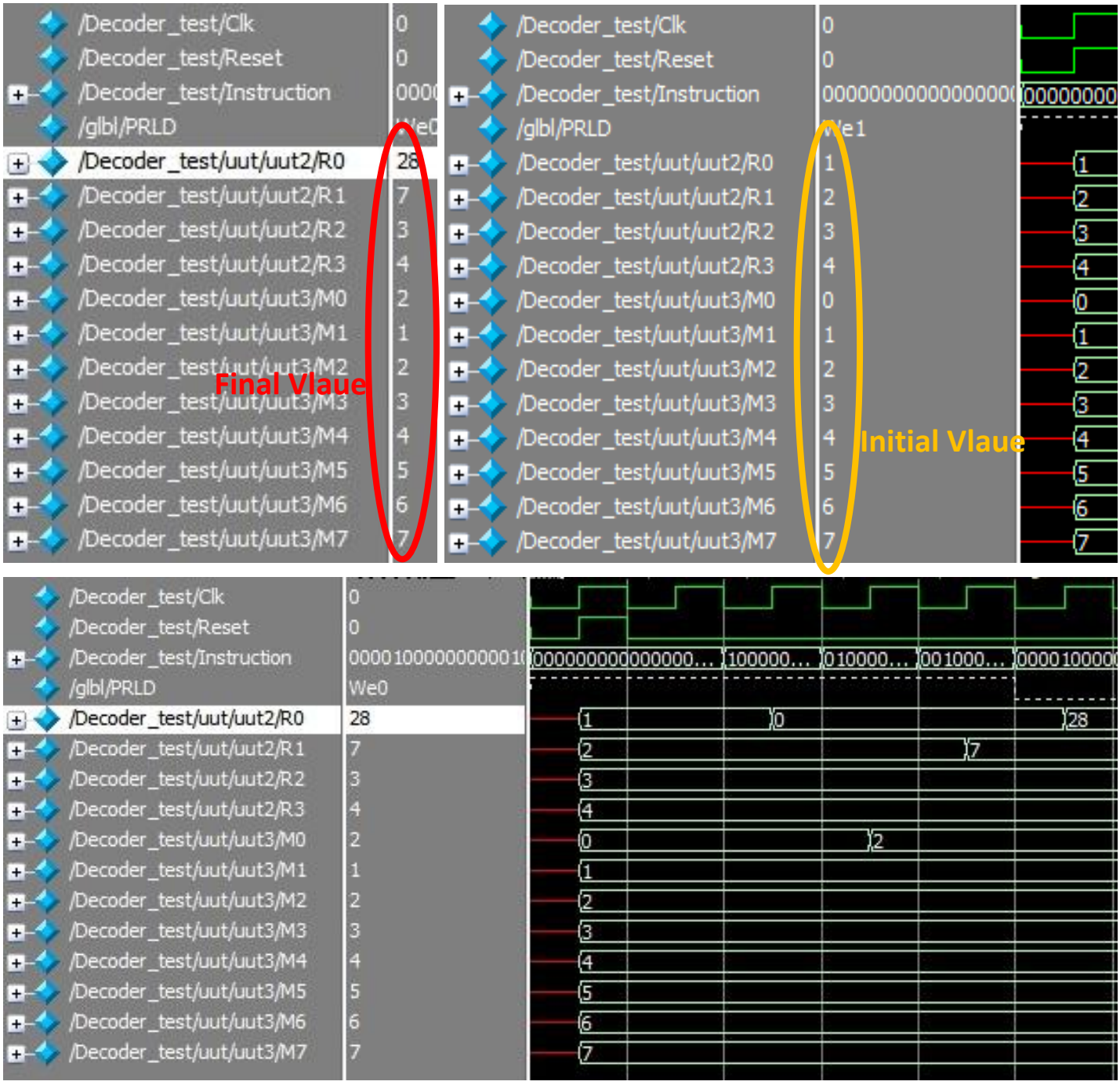
ALU.v 接收從 Decoder 來的 ALUOp 決定要將 A、B 做 Add、Sub、Shift Left 或 Shift Right 並將結果值存入 ALUResult 傳回給 Decoder 再作存值得動作(傳至 RegisterFile 中)。



Table：

I/O	Signals	LOAD	STORE	ADD	SL
Input	**Op7**	1	0	0	0
	Op6	0	1	0	0
	Op5	0	0	1	0
	Op4	0	0	0	0
	Op3	0	0	0	1
	Op2	0	0	0	0
	Op1	0	0	0	0
	Op0	0	0	0	0
Output	**RegDst**	0	0	1	1
	ALUSrc	X	X	0	1
	MemtoReg	1	X	0	0
	RegWrite	1	0	1	1
	MemRead	1	0	0	0
	MemWrite	0	1	0	0
	ALUOp1	X	X	0	1
	ALUOp0	X	X	0	0

波型圖：


/Decoder_test/Clk
/Decoder_test/Reset
/Decoder_test/Instruction
/glbl/PRLD
/Decoder_test/uut/uut2/R0
/Decoder_test/uut/uut2/R1
/Decoder_test/uut/uut2/R2
/Decoder_test/uut/uut2/R3
/Decoder_test/uut/uut3/M0
/Decoder_test/uut/uut3/M1
/Decoder_test/uut/uut3/M2
/Decoder_test/uut/uut3/M3
/Decoder_test/uut/uut3/M4
/Decoder_test/uut/uut3/M5
/Decoder_test/uut/uut3/M6
/Decoder_test/uut/uut3/M7
Final Vlaue
Initial Vlaue