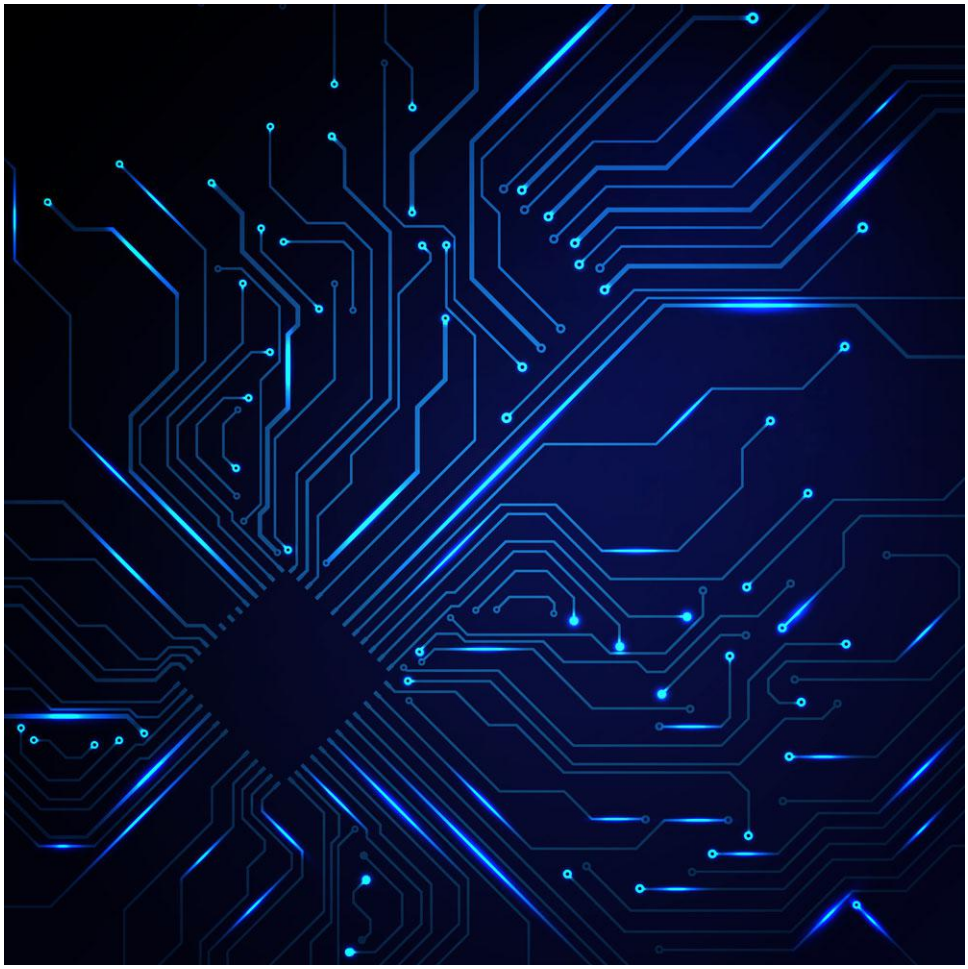


2022/11/7

實驗七

立即數定址

姓名：王嘉羽　　學號：00957116

班級：資工 3B

E-mail：vayne20011125@gmail.com

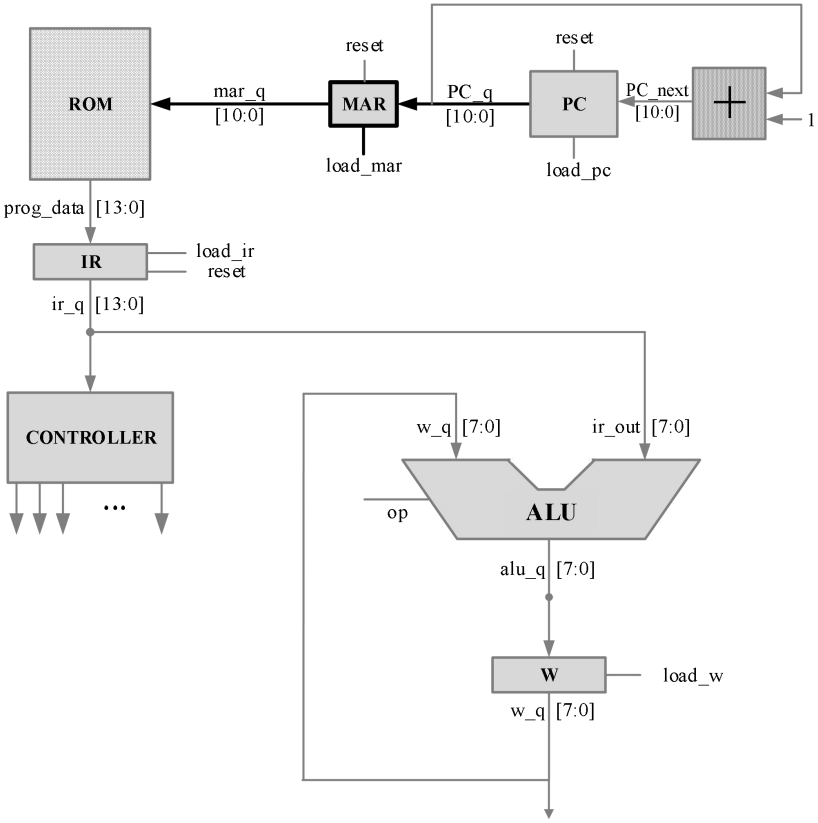
注意

1. 繳交時一律轉 PDF 檔
2. 繳交期限為
   上完課後
   當週五晚上 12 點前
3. 一人繳交一份
4. 檔名：學號_HW?.pdf
   檔名請按照作業檔名格式進行填寫
   未依照格式不予批改

● 實驗說明：

1. 如圖所示，設計一個架構實現立即定址的指令
2. 輸入：clk, reset
3. 輸出：w_q[7:0]
   下方有附 Rom 的截圖，請務必按照規定的 input 及 output 來做

● 系統硬體架構方塊圖（接線圖）：



架構圖

```
module Program_Rom(
    output logic [13:0] Rom_data_out,
    input [10:0] Rom_addr_in
);
//---------

    logic [13:0] data;
    always_comb
    begin
        case (Rom_addr_in)
            11'h0:  data = 14'h3044;        //MOVLW
            11'h1:  data = 14'h3E01;        //ADDLW
            11'h2:  data = 14'h3802;        //IORLW
            11'h3:  data = 14'h39FE;        //ANDLW
            11'h4:  data = 14'h3C47;        //SUBLW
            11'h5:  data = 14'h3A55;        //XORLW
            11'h6:  data = 14'h3AAA;        //XORLW
            default:data = 14'h0;
        endcase
    end
    assign Rom_data_out = data;

endmodule
```

Program_Rom

- **系統架構程式碼、測試資料程式碼與程式碼說明(.sv 檔及.do 檔都要截圖)**

  截圖請善用 win+shift+S

  - hw_1107.sv

```
1  `timescale 1ns/10ps
2  module hw_1107(
3      input clk,
4      input reset,
5      output logic [7:0] w_q
6  );
7      logic [10:0] pc_next, pc_q,mar_q;
8      logic load_pc, load_mar, load_ir_q,load_w;
9      logic [13:0] Rom_out,ir_q;
10     logic reset_ir_q;
11     logic [2:0] ps,ns;
12     logic [2:0] op;
13     logic [7:0] alu_q;
14     logic MOVLW;
15     logic ADDLW;
16     logic SUBLW;
17     logic ANDLW;
18     logic IORLW;
19     logic XORLW;
20     //-------------pc------------
21     assign pc_next = pc_q + 1;            //找到下一個指令
23     always_ff @(posedge clk)               //有load信號，再讀取
24     begin
25        if(reset)
26           pc_q <= #1 0;
27        else if(load_pc)
28           pc_q <= #1 pc_next;
29     end
30
31     //------------mar-----------
32     always_ff @(posedge clk)
33     begin
34        if(load_mar)
35           mar_q <= #1 pc_q;
36     end
37
38     //------------ROM-----------
39     Program_Rom rom1(Rom_out,mar_q);
40
41     //------------IR------------
42     always_ff @(posedge clk)
43     begin
44        if(reset_ir_q)
45           ir_q <= #1 0;
46        else if(load_ir_q)
47           ir_q <= #1 Rom_out;
48     end
50     //--------load_w------------
51     always_ff @(posedge clk)
52     begin
53        if(reset)
54           w_q <= #1 0;
55        else if(load_w)
56           w_q <= #1 alu_q;
57     end
58
59     //--------controller--------
60     //解碼指令，並給op值
61     assign MOVLW = (ir_q[13:8] == 6'b110000);
62     assign ADDLW = (ir_q[13:8] == 6'b111110);
63     assign SUBLW = (ir_q[13:8] == 6'b111100);
64     assign ANDLW = (ir_q[13:8] == 6'b111001);
65     assign IORLW = (ir_q[13:8] == 6'b111000);
66     assign XORLW = (ir_q[13:8] == 6'b111010);
68     always_comb
69     begin
70        if(reset)
71           op <= #1 0;
72        else
73           begin
74              if(MOVLW) op = 5;
75              else if(ADDLW) op = 0;
76              else if(SUBLW) op = 1;
77              else if(ANDLW) op = 2;
78              else if(IORLW) op = 3;
79              else if(XORLW) op = 4;
80              else op = 6;
81           end
82     end
```

```
84      //--------alu--------- 用op決定計算結果
85      always_comb
86      begin
87         if(reset)
88            alu_q <= #1 0;
89         else
90            begin
91               case(op)
92                  0: alu_q = ir_q[7:0] + w_q;
93                  1: alu_q = ir_q[7:0] - w_q;
94                  2: alu_q = ir_q[7:0] & w_q;
95                  3: alu_q = ir_q[7:0] | w_q;
96                  4: alu_q = ir_q[7:0] ^ w_q;
97                  5: alu_q = ir_q[7:0];
98               endcase
99            end
100     end
103     //--------fsm---------   有限狀態機
104     parameter T0 = 0;
105     parameter T1 = 1;
106     parameter T2 = 2;
107     parameter T3 = 3;
108     parameter T4 = 4;
109     parameter T5 = 5;
110     parameter T6 = 6;
111
112     always_ff @(posedge clk)
113     begin
114        if(reset) ps <= #1 0;
115        else ps <= #1 ns;
116     end
117
118     always_comb
119     begin                              //初始化
120     load_mar = 0;
121     load_pc = 0;
122     reset_ir_q = 0;
123     load_ir_q = 0;
124     load_w = 0;
125     ns = 0;
126        case(ps)
127        T0:                             //初始化ir_q
128           begin
129              reset_ir_q = 1;
130              ns = T1;
131           end
132
133        T1:
134           begin
135              load_mar = 1;             //load mar
136              ns = T2;
137           end
138
139        T2:
140           begin
141              load_pc = 1;              //load pc
142              ns = T3;
143           end
144
145        T3:
146           begin                        //load ir_q
147              load_ir_q = 1;
148              ns = T4;
149           end
151        T4:                             //load w
152           begin
153              load_w = 1;
154              ns = T5;
155           end
156
157        T5:                             //空狀態
158           begin
159              ns = T6;
160           end
161        T6:
162           begin
163              ns = T1;
164           end
165     endcase
166     end
167  endmodule
```

- program_Rom.sv

```
module Program_Rom(
    output logic [13:0]Rom_data_out,
    input [10:0]Rom_addr_in
);

     logic [13:0] data;

     always_comb
         begin
             case (Rom_addr_in)
                 11'h0 : data = 14'h3044;  //MOVLX 1
                 11'h1 : data = 14'h3E01;  //ADDLXW 12
                 11'h2 : data = 14'h3802;  //MOVLW 2
                 11'h3 : data = 14'h39FE;  //ADDLW 2
                 11'h4 : data = 14'h3C47;  //MOVLX 6
                 11'h5 : data = 14'h3A55;  //ADDLW 7
                 11'h6 : data = 14'h3AAA;  //ANDLW 7
                 default: data = 14'h0;
             endcase
         end

      assign Rom_data_out = data;

endmodule

```

- testbench.sv

simulation > tb > testbench.sv

```
`timescale 1ns/10ps
module testbench;

    logic reset;                    //重置
    logic clk;                      //時脈
    logic [7:0] w_q;                    //輸出

    hw_1107 hw_1107_1(
        .reset(reset), //()內的變數為tb的變數，"."後面為hw_1107.sv的變數，將2者對應起來
        .clk(clk),
        .w_q(w_q)
    );

    always #10 clk = ~clk;

    initial begin
        reset = 1;clk = 0; //一開始先reset，將時脈歸0
        #15 reset = 0;
        #1000 $stop;
    end
endmodule
```

- compile.do

simulation > modelsim > compile.do

```
#vlib work



# ----------------------------------------------------------------
vlog ../tb/testbench.sv
vlog ../../design/hw_1107.sv
vlog ../../design/Program_Rom.sv

```

- sim.do

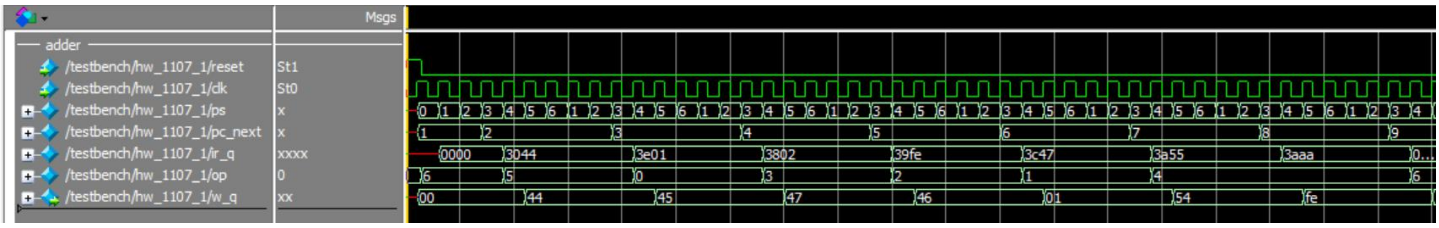
simulation > modelsim > sim.do

```
vsim -voptargs=+acc work.testbench
view structure wave signals

do wave.do

log -r *
run -all
```

- wave.do

```
simulation > modelsim > ≡ wave.do
1   onerror {resume}
2   quietly WaveActivateNextPane {} 0
3
4   #add wave -noupdate -divider {TOP LEVEL INPUTS}
5
6   #add wave -noupdate -format Logic /testbench/clk
7   #add wave -noupdate -format Logic /testbench/rst
8
9
10
11  add wave -noupdate -divider {adder}
12
13  add wave -noupdate -format logic    /testbench/hw_1107_1/reset
14  add wave -noupdate -format logic    /testbench/hw_1107_1/clk
15  add wave -noupdate -format Literal -radix Unsigned      /testbench/hw_1107_1/ps
16  add wave -noupdate -format Literal -radix Unsigned      /testbench/hw_1107_1/pc_next
17  add wave -noupdate -format Literal -radix Hexadecimal   /testbench/hw_1107_1/ir_q
18  add wave -noupdate -format Literal -radix Hexadecimal   /testbench/hw_1107_1/op
19  add wave -noupdate -format Literal -radix Hexadecimal   /testbench/hw_1107_1/w_q
```

● 模擬結果與結果說明：



ppt 可結果圖是在狀態 6 才 load w，但是前面是寫狀態 4 load w，我是按照 ppt 前面介紹的那邊寫的。

● 結論與心得：

感覺東西有越來越難的趨勢，但是隨著教材變難，也更了解了整個的運作流程。寫完後，覺得一開始設計這架構的人還蠻聰明的，他用很巧妙的方式決定變數，在組合邏輯和序向邏輯裡轉換，就可以比一般軟體還要快很多，而且寫法也簡單很多。真是佩服佩服!