FPGA 系統設計實務

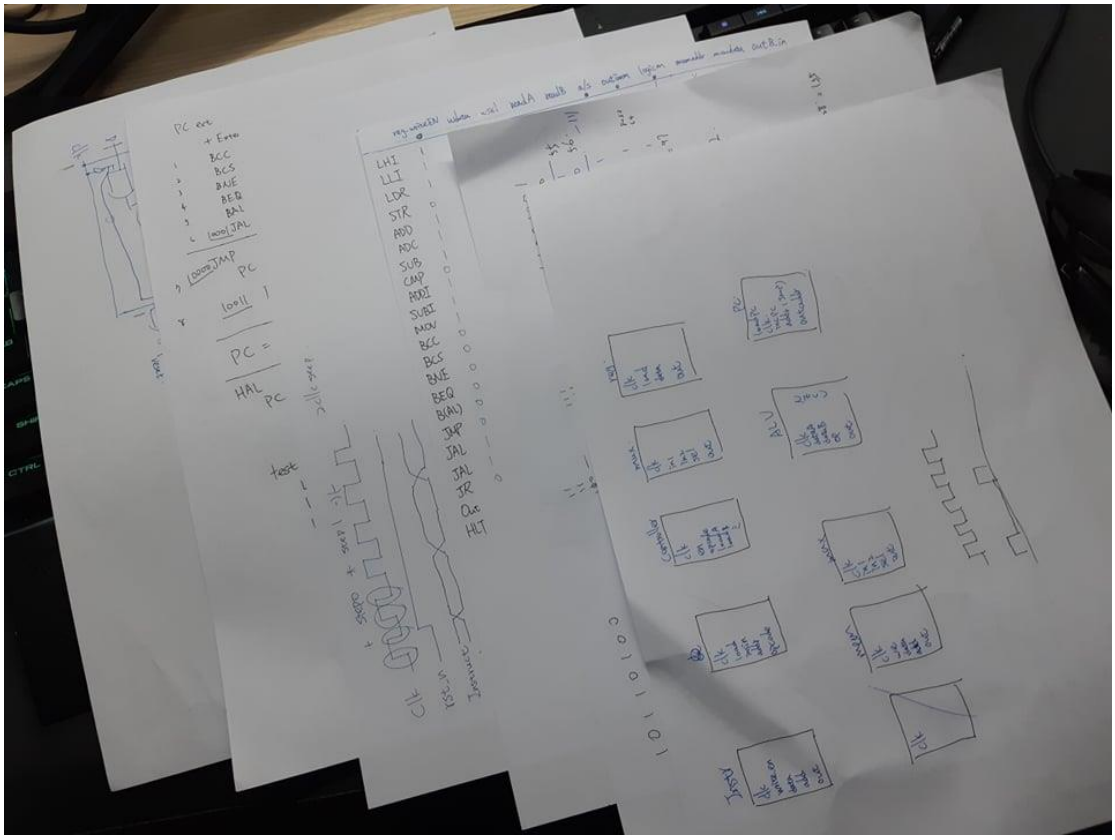
期中報告

B10702226

李季鴻

Design

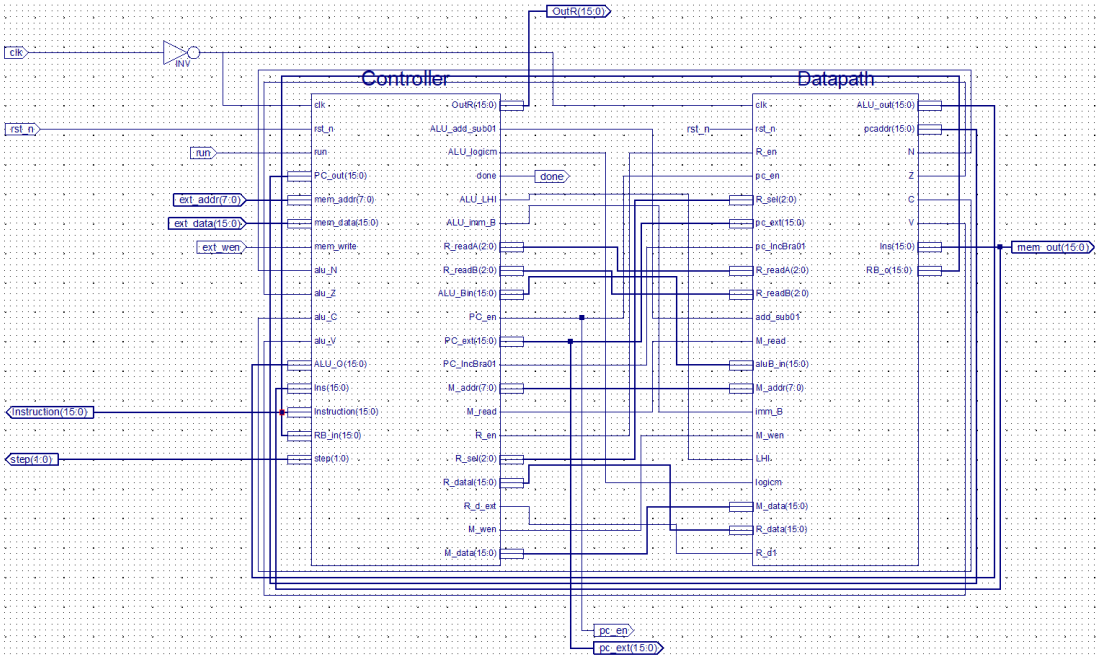
在此部分，我不確定需要放上繳交的內容，是否要完全完整每個部份的先行設計，都有條理地列出來在這報告上，所以我將我在設計過程中的草稿內容稍微重新寫過放在此報告上。



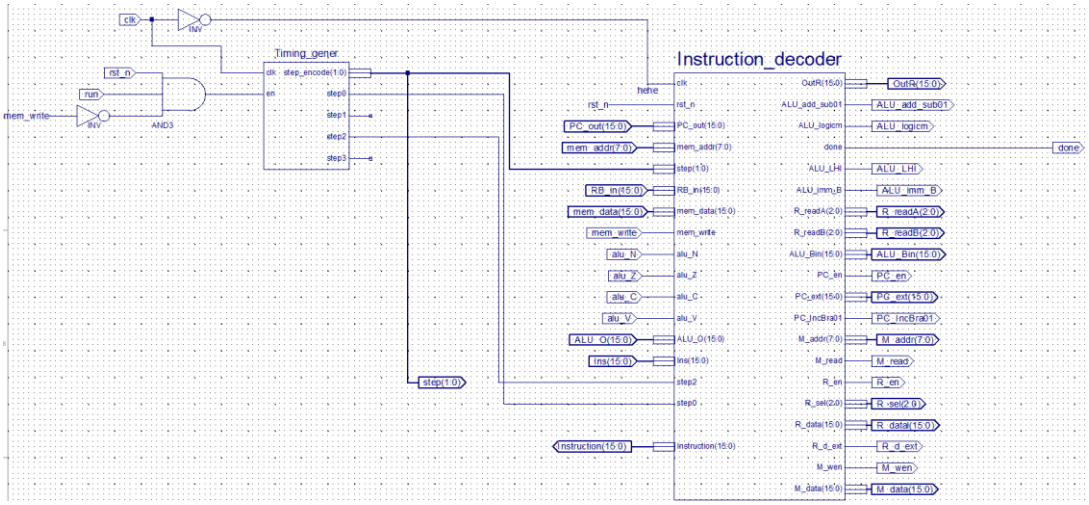
B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V
		v	v	v	v	v	v	v	v	v	v	v	v	v	v	v		x	v	v
		regEn	regSel	regData	reg 外	readA	readB	outB_in	outImm	LHI	add/sub	logicm	memread	memEn	memAddr	memData	done	pcen	pcext	pc inc bra
00001	LHI	1	instr[10:8]	ALUout	0	instr[10:8]		instr[7:0]	1	1	0	1	0	0	x	x	0		1	0
00010	LLI	1	instr[10:8]	ALUout	0	x		instr[7:0]	1	0	0	1	0	0	x	x	0		1	0
00011	LDR	1	instr[10:8]	MEMout	1	instr[7:5]		instr[4:0]	1	0	0	0	1	0	ALUout	x	0		1	0
00101	STR	0	x	x	0	instr[7:5]	instr[10:8]	instr[4:0]	1	0	0	0	0	1	ALUout	RB	0		1	0
00000	ADD	1	instr[10:8]	ALUout	0	instr[7:5]	instr[4:2]		0	0	0	0	0	0	x	x	0		1	0
00000	ADC	1	instr[10:8]	ALUout	0	instr[7:5]	instr[4:2]		0	0	0	0	0	0	x	x	0		1	0
00000	SUB	1	instr[10:8]	ALUout	0	instr[7:5]	instr[4:2]		0	0	1	0	0	0	x	x	0		1	0
00000	SBB	1	instr[10:8]	ALUout	0	instr[7:5]			0	0	1	0	0	0	x	x	0		1	0
00110	CMP	0	x	x	0	instr[7:5]	instr[4:2]		0	0	1	0	0	0	x	x	0		1	0
00111	ADDI	1	instr[10:8]	ALUout	0	instr[7:5]		instr[4:0]	1	0	0	0	0	0	x	x	0		1	0
01000	SUBI	1	instr[10:8]	ALUout	0	instr[7:5]		instr[4:0]	1	0	1	0	0	0	x	x	0		1	0
01011	MOV	1	instr[10:8]	ALUout	0	instr[7:5]		0x0000	1	0	0	0	0	0	x	x	0		1	0
C3	BCC	0	x	x	0	x	x		0	0	0	0	0	0	x	x	0		disp	0
C2	BCS	0	x	x	0	x	x		0	0	0	0	0	0	x	x	0		disp	0
C1	BNE	0	x	x	0	x	x		0	0	0	0	0	0	x	x	0		disp	0
C0	BEQ	0	x	x	0	x	x		0	0	0	0	0	0	x	x	0		disp	0
CE	B[AL]	0	x	x	0	x	x		0	0	0	0	0	0	x	x	0		disp	0
10000	JMP	0	x	x	0	x	x		0	0	0	0	0	0	x	x	0		instr[10:0]	1
10001	JAL	1	instr[10:8]	pcaddr	1	x	x		0	0	0	0	0	0	x	x	0		disp	0
10010	JAL	1	instr[10:8]	pcaddr	1	instr[7:5]		0x0000	1	0	0	0	0	0	x	x	0		ALUout	1
10011	JR	0	x	x	0	instr[7:5]		0x0000	1	0	0	0	0	0	x	x	0		ALUout	1
11100	OutR	0	x	x	0	instr[7:5]		0x0000	1	0	0	0	0	0	x	x	0		1	0
11100	HLT	0	x	x	0	x	x	0	0	0	0	0	0	0	x	x	1		0	0

Schematic

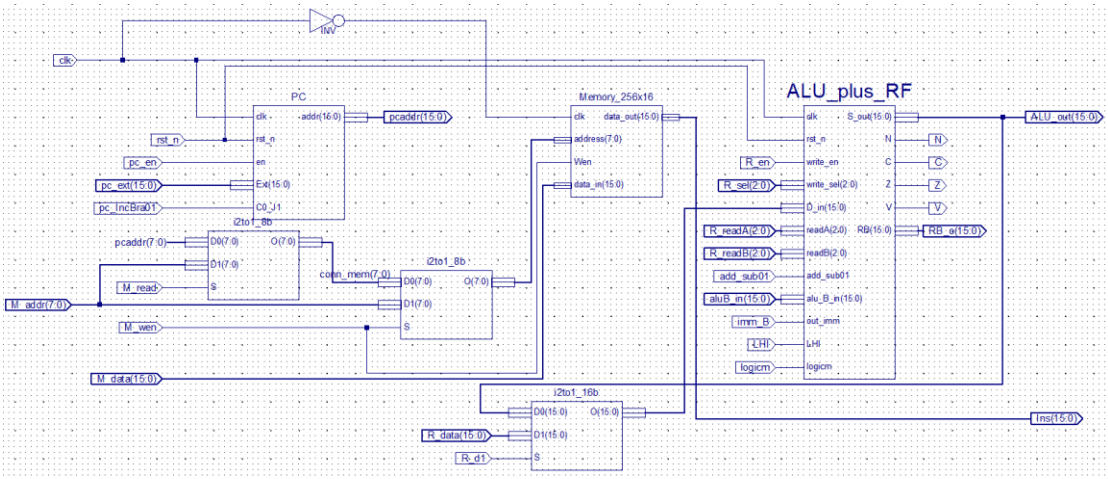
1. Top-level CPU



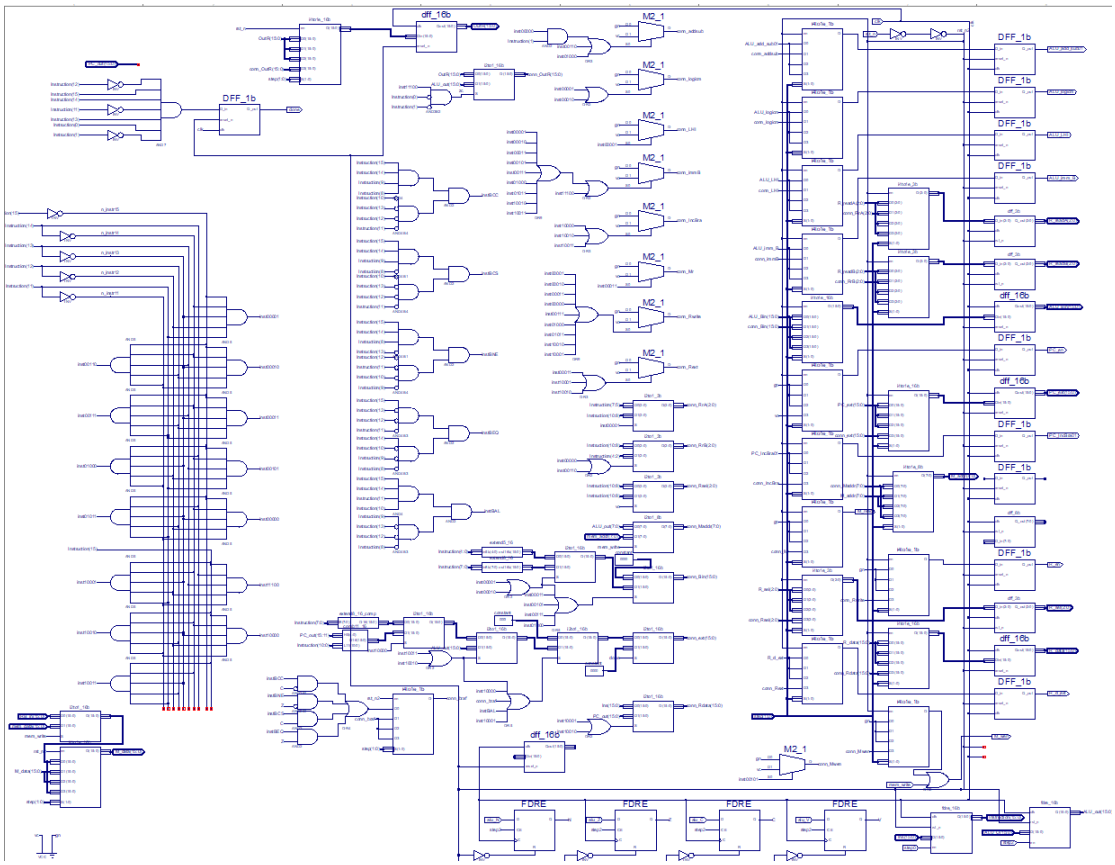
2. Controller



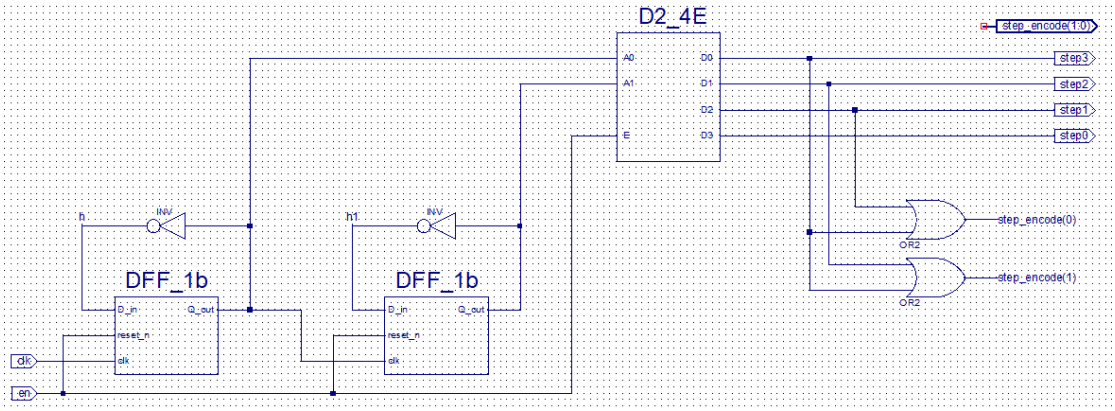
3. Datapath



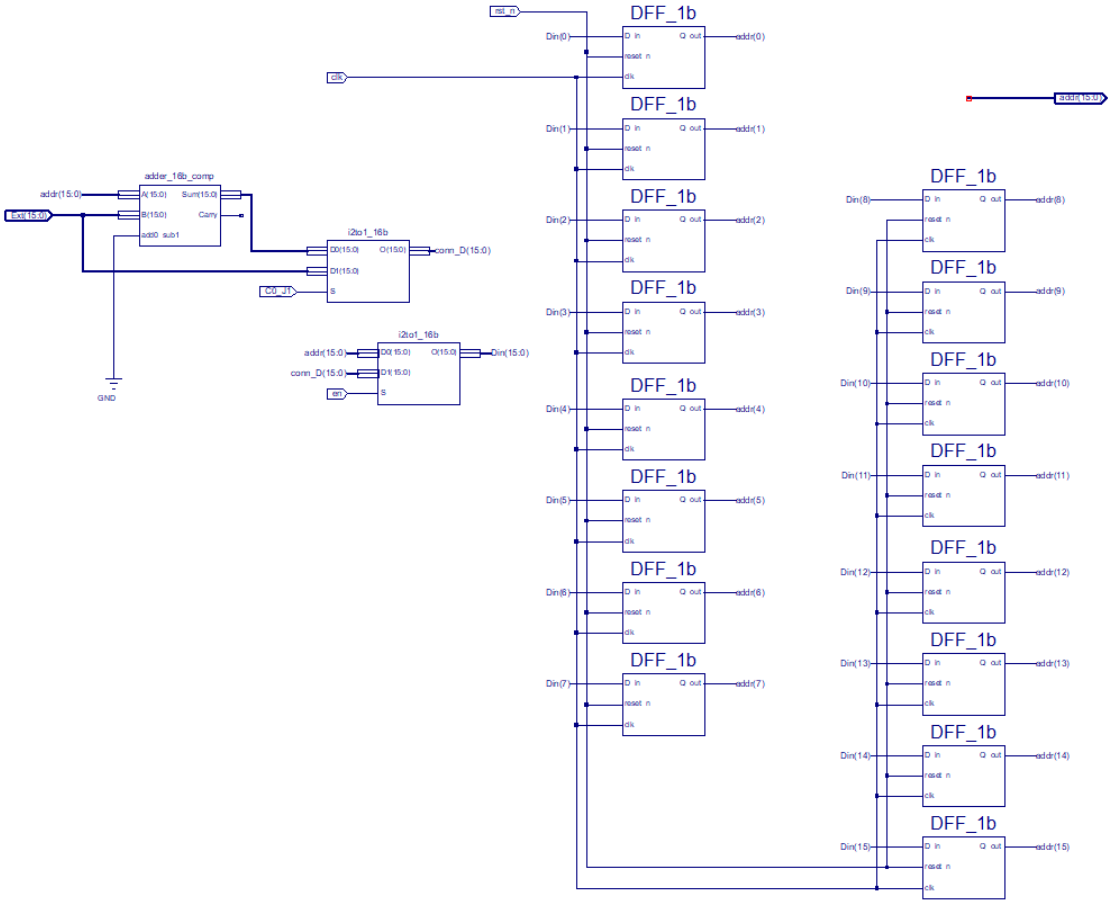
4. Instruction Decoder



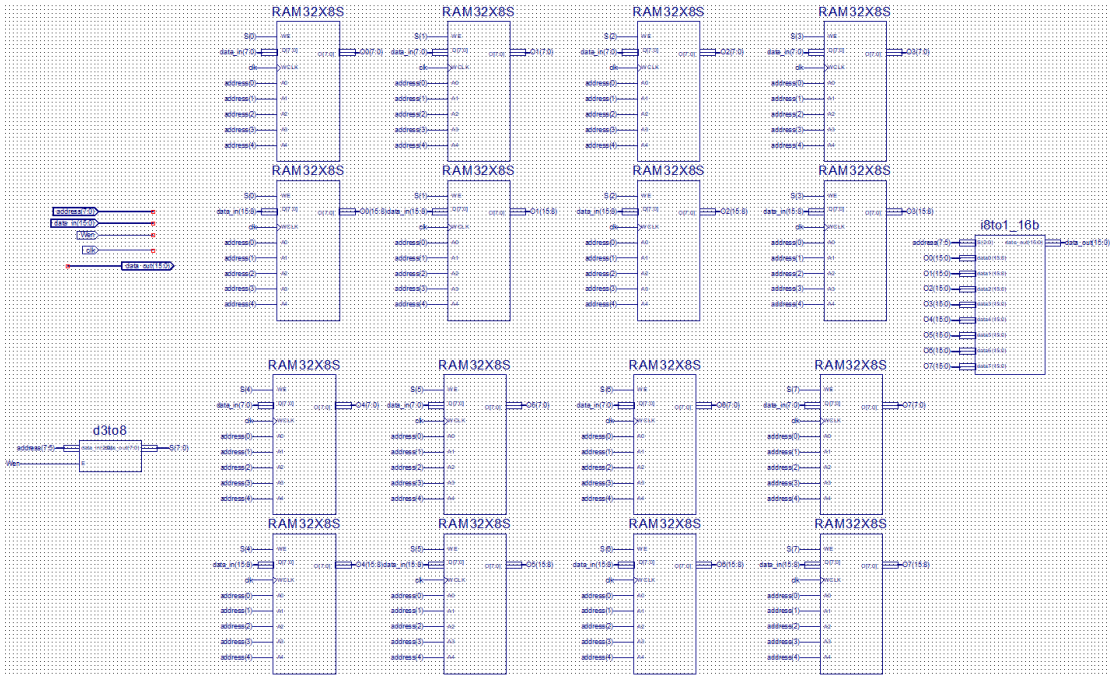
5. Timing Generator



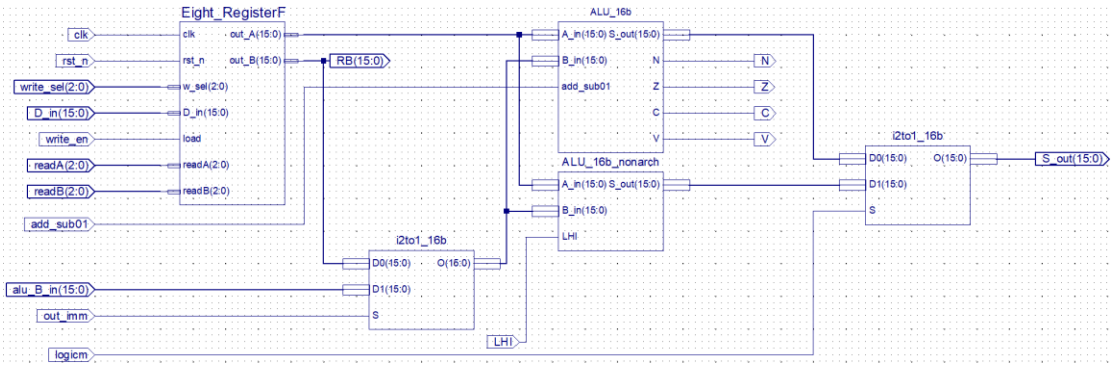
6. Program Counter



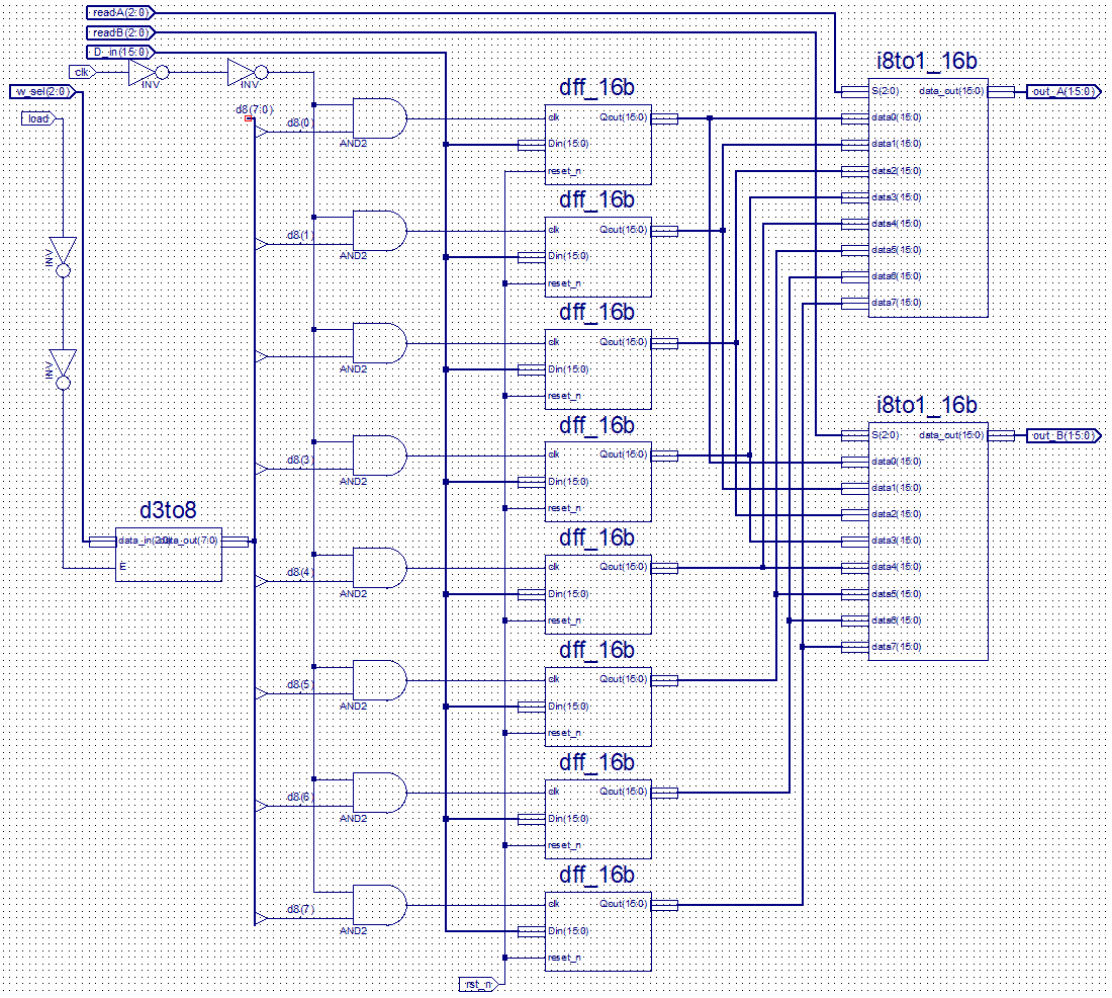
7. 256x16 Memory



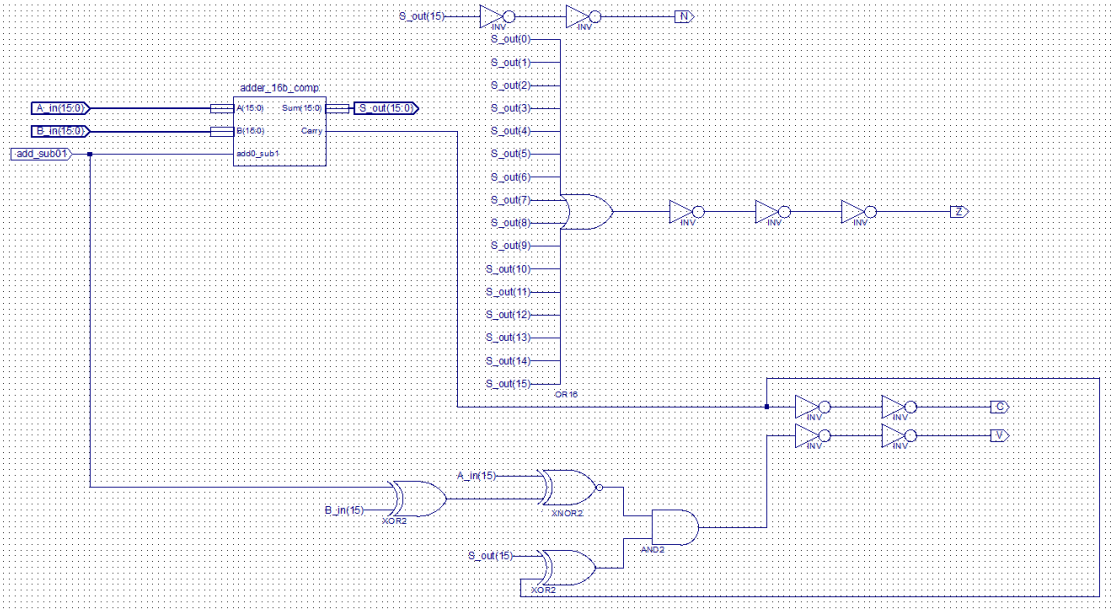
8. Register File + ALU



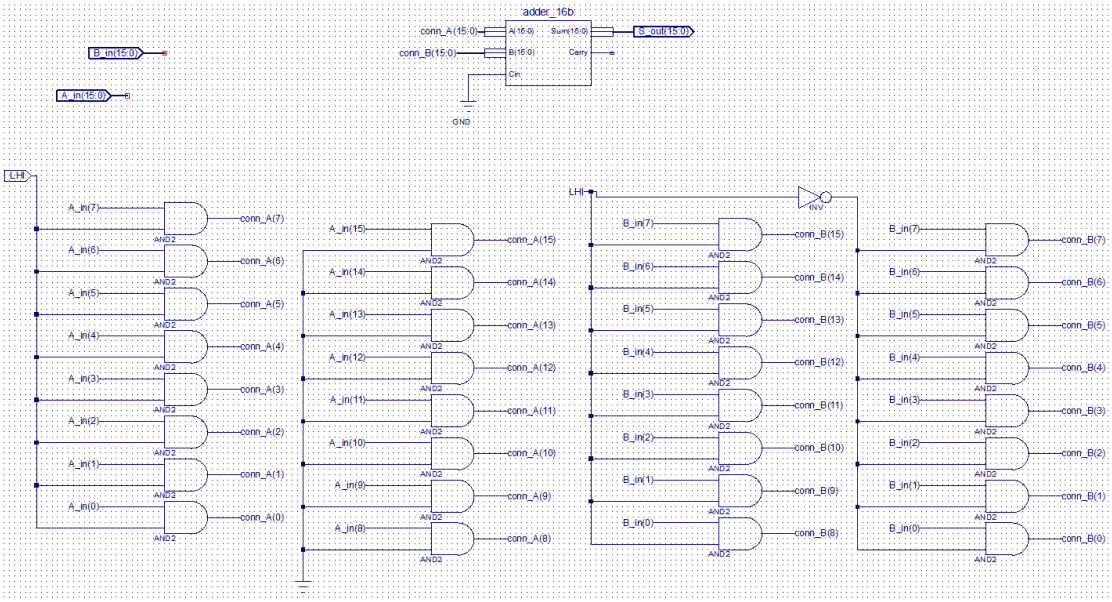
9. Eight Register File



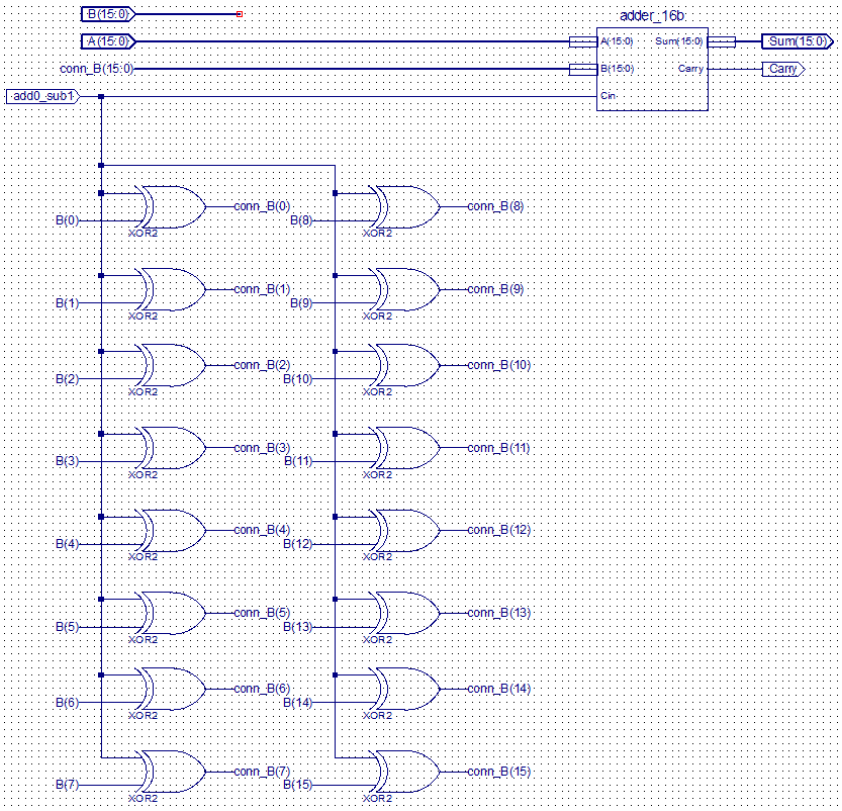
10. ALU



11. Logicm ALU



12.16-bits complement Adder



避免太長其他設計的小 module 將不放此報告。

Verification

1. Register File

```
    $monitor($time," readA=%1h,out_A=%4h,readB=%1h,out_B=%4h",readA,out_A,readB,out_B);
 end
 initial begin
    #5;
    for(i=0;i<8;i=i+1)begin
       readA = i[2:0];
       readB = i[2:0];
       #2;
    end
    write_register(3'b010,16'h1278);
    write_register(3'b011,16'h2401);
    write_register(3'b101,16'h2121);
    write_register(3'b110,16'h6792);
    write_register(3'b111,16'h6662);
    for(i=0;i<8;i=i+1)begin
       readA = i[2:0];
       readB = i[2:0];
       #5;
    end
    $finish;
 end
 task write_register();
    input [2:0] sel;
    input [15:0] D;
    begin
       w_sel <= sel;
       D_in <= D;
       load <= 1;
       //$display("R%1h = %4h",w_sel,D_in);
       #15 load = 0;
    end
 endtask
 always@(posedge clk)
    if (load==1)
       $display("R%1h = %4h",w_sel,D_in);
 always #5 clk = ~clk;
dmodule
```

```
Finished circuit initialization process.
           1 readA=x,out_A=0000,readB=x,out_B=0000
           5 readA=0,out_A=0000,readB=0,out_B=0000
           7 readA=1,out_A=0000,readB=1,out_B=0000
           9 readA=2,out_A=0000,readB=2,out_B=0000
          11 readA=3,out_A=0000,readB=3,out_B=0000
          13 readA=4,out_A=0000,readB=4,out_B=0000
          15 readA=5,out_A=0000,readB=5,out_B=0000
          17 readA=6,out_A=0000,readB=6,out_B=0000
          19 readA=7,out_A=0000,readB=7,out_B=0000
R2 = 1278
R2 = 1278
R3 = 2401
R5 = 2121
R5 = 2121
R6 = 6792
R7 = 6662
          85 readA=7,out_A=6662,readB=7,out_B=6662
R7 = 6662
          95 readA=7,out_A=6662,readB=7,out_B=6662
          96 readA=0,out_A=0000,readB=0,out_B=0000
         101 readA=1,out_A=0000,readB=1,out_B=0000
         106 readA=2,out_A=1278,readB=2,out_B=1278
         111 readA=3,out_A=2401,readB=3,out_B=2401
         116 readA=4,out_A=0000,readB=4,out_B=0000
         121 readA=5,out_A=2121,readB=5,out_B=2121
         126 readA=6,out_A=6792,readB=6,out_B=6792
         131 readA=7,out_A=6662,readB=7,out_B=6662
```

2. 16-bits ALU

```
initial begin
   test_adder(16'd30000,16'd20000,1); //1
   test_adder(16'd30000,16'd20000,0); //2
   test_adder(16'd6666,16'd6666,1);   //3
   test_adder(16'd7777,16'd7777,0);   //4
   test_adder(16'd32767,16'd1,0);     //5
   test_adder(16'h8000,16'h0001,1);   //6
   test_adder(16'h8000,16'h0001,0);   //7
   $finish;
end
reg [16:0] S;
reg tn,tz,tc,tv;
task test_adder();
   input [15:0] a;
   input [15:0] b;
   input op;
   begin
      $display($time);
      A=a;
      B=b;
      add_sub01=op;
      if(op==0)begin
         S=a+b;
         #5 $display("a+b=%5h, C=%1b,S_out=%4h",S,C,S_out);
      end
      else begin
         b=~b;
         S=a+b+op;
         #5 $display("a-b=%5h, C=%1b,S_out=%4h",S,C,S_out);
      end
      tn = S[15];
      tz = (S[15:0]==0)? 1:0;
      tc = S[16];
      tv = (a[15]!=b[15]||S[15]==a[15])? 0:1;
      $display("tb:NZCV=%1b%1b%1b%1b",tn,tz,tc,tv);
      $display("sch:NZCV=%1b%1b%1b%1b",N,Z,C,V);
   end
endtask
```

```
Simulator is doing circuit initialization process.
           0
Finished circuit initialization process.
a-b=12710, C=1,S_out=2710
tb:NZCV=0010
sch:NZCV=0010
           5
a+b=0c350, C=0,S_out=c350
tb:NZCV=1001
sch:NZCV=1001
          10
a-b=10000, C=1,S_out=0000
tb:NZCV=0110
sch:NZCV=0110
          15
a+b=03cc2, C=0,S_out=3cc2
tb:NZCV=0000
sch:NZCV=0000
          20
a+b=08000, C=0,S_out=8000
tb:NZCV=1001
sch:NZCV=1001
          25
a-b=17fff, C=1,S_out=7fff
tb:NZCV=0011
sch:NZCV=0011
          30
a+b=08001, C=0,S_out=8001
tb:NZCV=1000
sch:NZCV=1000
```

3. Register File + ALU

```
task test_adder();
   input [2:0] a;
   input [2:0] b;
   input op;
   begin
      LHI=0;
      imm_B=0;
      logicm=0;
      $display($time);

      readA=a;
      readB=b;
      add_sub01=op;

      if(op==0)begin
         S = testarr[a]+testarr[b];
         #5 $display("R%1d+R%1d,S_tb=%4h, S_out=%4h",a,b,S,S_out);
         tV2 = testarr[b];
      end
      else begin
         bf = ~testarr[b];
         S = testarr[a]+bf+1;
         tV2 = bf;
         #5 $display("R%1d-R%1d,S_tb=%4h, S_out=%4h",a,b,S,S_out);
      end
      tn = S[15];
      tz = (S[15:0]==0)? 1:0;
      tc = S[16];
      tV1 = testarr[a];

      tv = (tV1[15]!=tV2[15]||S[15]==tV1[15])? 0:1;
      $display("tb:NZCV=%1b%1b%1b%1b",tn,tz,tc,tv);
      $display("sch:NZCV=%1b%1b%1b%1b",N,Z,C,V);
   end
endtask
```

```
Finished circuit initialization process.
R2 = 1278
R3 = 2401
R3 = 2401
R5 = 2121
R6 = 6792
R6 = 6792
R7 = 6662
          101
R1-R5,S_tb=0dedf, S_out=dedf
tb:NZCV=1000
sch:NZCV=1000
          106
R0+R7,S_tb=06662, S_out=6662
tb:NZCV=0000
sch:NZCV=0000
          111
R5-R6,S_tb=0b98f, S_out=b98f
tb:NZCV=1000
sch:NZCV=1000
          116
R3+R1,S_tb=02401, S_out=2401
tb:NZCV=0000
sch:NZCV=0000
          121
R3-R5,S_tb=102e0, S_out=02e0
tb:NZCV=0010
sch:NZCV=0010
          126
R1+R6,S_tb=06792, S_out=6792
tb:NZCV=0000
sch:NZCV=0000
0521
0578
0501
```

4. 256x16 Memory

```
Finished circuit initialization process.
read: addr=00,data_out=   0
read: addr=01,data_out=   1
read: addr=02,data_out=   2
read: addr=03,data_out=   3
read: addr=04,data_out=   4
read: addr=05,data_out=   5
read: addr=06,data_out=   6
read: addr=07,data_out=   7
read: addr=08,data_out=   8
read: addr=09,data_out=   9
read: addr=0a,data_out=  10
read: addr=0b,data_out=  11
read: addr=0c,data_out=  12
read: addr=0d,data_out=  13
read: addr=0e,data_out=  14
read: addr=0f,data_out=  15
read: addr=10,data_out=  16
read: addr=11,data_out=  17
read: addr=12,data_out=  18
read: addr=13,data_out=  19
read: addr=14,data_out=  20
read: addr=15,data_out=  21
read: addr=16,data_out=  22
read: addr=17,data_out=  23
read: addr=18,data_out=  24
read: addr=19,data_out=  25
read: addr=1a,data_out=  26
read: addr=1b,data_out=  27
read: addr=1c,data_out=  28
read: addr=1d,data_out=  29
read: addr=1e,data_out=  30
read: addr=1f,data_out=  31
read: addr=20,data_out=  32
read: addr=21,data_out=  33
read: addr=22,data_out=  34
read: addr=23,data_out=  35
read: addr=24,data_out=  36
read: addr=25,data_out=  37
read: addr=26,data_out=  38
read: addr=27,data_out=  39
read: addr=28,data_out=  40
read: addr=29,data_out=  41
read: addr=2a,data_out=  42
read: addr=2b,data_out=  43
read: addr=2c,data_out=  44
read: addr=2d,data_out=  45
read: addr=2e,data_out=  46
read: addr=2f,data_out=  47
read: addr=30,data_out=  48
read: addr=31,data_out=  49
read: addr=32,data_out=  50
read: addr=33,data_out=  51
read: addr=34,data_out=  52
read: addr=35,data_out=  53
read: addr=36,data_out=  54
read: addr=37,data_out=  55
read: addr=38,data_out=  56
read: addr=39,data_out=  57
read: addr=3a,data_out=  58
read: addr=3b,data_out=  59
read: addr=3c,data_out=  60
read: addr=3d,data_out=  61
read: addr=3e,data_out=  62
read: addr=3f,data_out=  63
read: addr=40,data_out=  64
read: addr=bd,data_out= 189
read: addr=be,data_out= 190
read: addr=bf,data_out= 191
read: addr=c0,data_out= 192
read: addr=c1,data_out= 193
read: addr=c2,data_out= 194
read: addr=c3,data_out= 195
read: addr=c4,data_out= 196
read: addr=c5,data_out= 197
read: addr=c6,data_out= 198
read: addr=c7,data_out= 199
read: addr=c8,data_out= 200
read: addr=c9,data_out= 201
read: addr=ca,data_out= 202
read: addr=cb,data_out= 203
read: addr=cc,data_out= 204
read: addr=cd,data_out= 205
read: addr=ce,data_out= 206
read: addr=cf,data_out= 207
read: addr=d0,data_out= 208
read: addr=d1,data_out= 209
read: addr=d2,data_out= 210
read: addr=d3,data_out= 211
read: addr=d4,data_out= 212
read: addr=d5,data_out= 213
read: addr=d6,data_out= 214
read: addr=d7,data_out= 215
read: addr=d8,data_out= 216
read: addr=d9,data_out= 217
read: addr=da,data_out= 218
read: addr=db,data_out= 219
read: addr=dc,data_out= 220
read: addr=dd,data_out= 221
read: addr=de,data_out= 222
read: addr=df,data_out= 223
read: addr=e0,data_out= 224
read: addr=e1,data_out= 225
read: addr=e2,data_out= 226
read: addr=e3,data_out= 227
read: addr=e4,data_out= 228
read: addr=e5,data_out= 229
read: addr=e6,data_out= 230
read: addr=e7,data_out= 231
read: addr=e8,data_out= 232
read: addr=e9,data_out= 233
read: addr=ea,data_out= 234
read: addr=eb,data_out= 235
read: addr=ec,data_out= 236
read: addr=ed,data_out= 237
read: addr=ee,data_out= 238
read: addr=ef,data_out= 239
read: addr=f0,data_out= 240
read: addr=f1,data_out= 241
read: addr=f2,data_out= 242
read: addr=f3,data_out= 243
read: addr=f4,data_out= 244
read: addr=f5,data_out= 245
read: addr=f6,data_out= 246
read: addr=f7,data_out= 247
read: addr=f8,data_out= 248
read: addr=f9,data_out= 249
read: addr=fa,data_out= 250
read: addr=fb,data_out= 251
read: addr=fc,data_out= 252
read: addr=fd,data_out= 253
read: addr=fe,data_out= 254
read: addr=ff,data_out= 255
```

5. Program Counter

```
initial begin
   $monitor("1:%h,2:%h,3:%h,4:%h",Ext,addr,en,C0_J1);
   #1;
   rst_n=1;
   testPC(16'h0000,0);
   testPC(16'h0001,0);
   testPC(16'h0001,0);
   testPC(16'h0001,0);
   testPC(16'h0001,0);
   testPC(16'h0000,1);
   testPC(16'h0001,0);
   $finish;
end
task testPC();
   input [15:0] ad;
   input inc_jmp;

   begin
      //wait(!clk);
      @(posedge clk)en=1;
         C0_J1 = inc_jmp;
         Ext = ad;
         en=1;
      @(posedge clk);
      //wait(!clk);
         #1 en=0;

      #30;
   end
endtask
always #5 clk = ~clk;
```

```
Simulator is doing circuit initialization process.
Finished circuit initialization process.
1:0000,2:0000,3:0,4:0
1:0000,2:0000,3:1,4:0
1:0000,2:0000,3:1,4:0
1:0000,2:0000,3:0,4:0
1:0000,2:0000,3:0,4:0
1:0000,2:0000,3:0,4:0
1:0000,2:0000,3:0,4:0
1:0001,2:0000,3:1,4:0
1:0001,2:0001,3:1,4:0
1:0001,2:0001,3:0,4:0
1:0001,2:0001,3:0,4:0
1:0001,2:0001,3:0,4:0
1:0001,2:0001,3:0,4:0
1:0001,2:0001,3:1,4:0
1:0001,2:0002,3:1,4:0
1:0001,2:0002,3:0,4:0
1:0001,2:0002,3:0,4:0
1:0001,2:0002,3:0,4:0
1:0001,2:0002,3:0,4:0
1:0001,2:0002,3:1,4:0
1:0001,2:0003,3:1,4:0
1:0001,2:0003,3:0,4:0
1:0001,2:0003,3:0,4:0
1:0001,2:0003,3:0,4:0
1:0001,2:0003,3:0,4:0
1:0001,2:0003,3:1,4:0
1:0001,2:0004,3:1,4:0
1:0001,2:0004,3:0,4:0
1:0001,2:0004,3:0,4:0
1:0001,2:0004,3:0,4:0
1:0001,2:0004,3:0,4:0
1:0000,2:0004,3:1,4:1
1:0000,2:0000,3:1,4:1
1:0000,2:0000,3:0,4:1
1:0000,2:0000,3:0,4:1
1:0000,2:0000,3:0,4:1
1:0000,2:0000,3:0,4:1
1:0001,2:0000,3:1,4:0
1:0001,2:0001,3:1,4:0
1:0001,2:0001,3:0,4:0
1:0001,2:0001,3:0,4:0
1:0001,2:0001,3:0,4:0
1:0001,2:0001,3:0,4:0
Stopped at time : 346 ns : File "E:/FPGA_11001/RISC_16bit_
```

6. Controller

```
initial begin
   $monitor("PC_ext=%h,PC_IncBra",PC_ext,PC_IncBra01);
   mem_write <= 1;
   mem_addr <= 1;
   mem_data <= 16'h55;
   clk=0;
   rst_n=0;
   #1 rst_n = 1;
   write_mem(16'h0,16'b00010_001_00001010  ) ; //
   write_mem(16'h1,16'b00010_000_00110000  ) ; //
   write_mem(16'h2,16'b00000_001_000_001_00 ) ; //
   write_mem(16'h3,16'b00011_010_000_00000   ) ; //
   write_mem(16'h4,16'b00101_010_000_01010  ) ; //
   write_mem(16'h5,16'b00111_000_000_00001  ) ; //
   write_mem(16'h6,16'b00110_000_001_000_01) ; //
   write_mem(16'h7,16'b11000001_11111100  ) ; //
   run = 1;
   mem_write <= 1;
   mem_addr <= 1;
   mem_data <= 16'h72;
   Ins<=16'h0125;
   #50;
   mem_write<=0;


   #999 $finish;
end
```

```
PC_ext=xxxx,PC_IncBrax
PC_ext=xxxx,PC_IncBrax
PC_ext=xxxx,PC_IncBrax
PC_ext=xxxx,PC_IncBrax
PC_ext=xxxx,PC_IncBrax
PC_ext=xxxx,PC_IncBrax
PC_ext=xxxx,PC_IncBrax
PC_ext=xxxx,PC_IncBrax
PC_ext=0001,PC_IncBra0
PC_ext=0001,PC_IncBra0
PC_ext=0001,PC_IncBra0
PC_ext=0001,PC_IncBra0
PC_ext=0001,PC_IncBra0
PC_ext=0001,PC_IncBra0
PC_ext=0001,PC_IncBra0
PC_ext=0001,PC_IncBra0
PC_ext=0001,PC_IncBra0
```

在接下來 Complete Computer 部分 所使用的資料都是下面圖 1

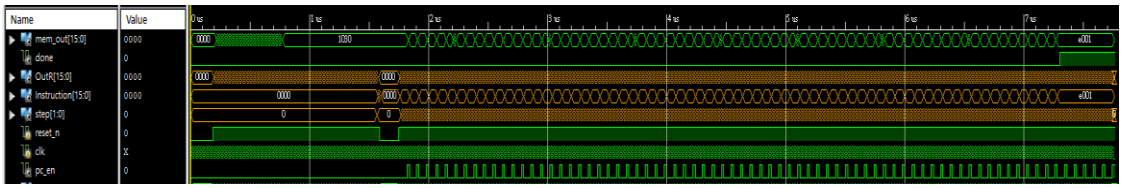
```
117         write_mem(16'h30,16'h47) ; // data (30h, 47h)
118         write_mem(16'h31,16'h80) ; // data (31h, 80h)
119         write_mem(16'h32,16'h42) ; // data (32h, 42h)
120         write_mem(16'h33,16'h77) ; // data (33h, 77h)
121         write_mem(16'h34,16'hf5) ; // data (34h, f5h)
122         write_mem(16'h35,16'h33) ; // data (35h, 33h)
123         write_mem(16'h36,16'h66) ; // data (36h, 66h)
124         write_mem(16'h37,16'h12) ; // data (37h, 12h)
125         write_mem(16'h38,16'h35) ; // data (38h, 35h)
126         write_mem(16'h39,16'h87) ; // data (39h, 87h)
```

▲圖一

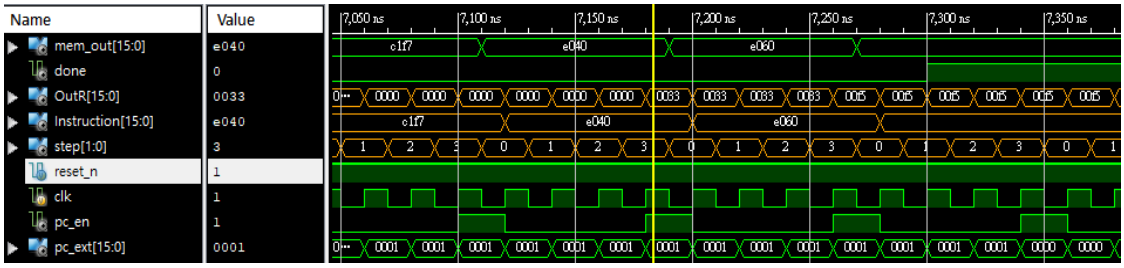
1. Find the minimum and maximum from two numbers in memoey

	A	B	C	D	E	F	G
1	HEX	label	ASSEMBLY		COMMAND		
2	0		LLI R0,#30H		00010_000_00110000		
3	1		LLI R1,#37H		00010_001_00110111		
4	2		LLI R2,#255		00010_010_11111111		
5	3		LHI R2,#7fH		00001_010_01111111		
6	4		LLI R3,#0		00010_011_00000000		
7	5	lab0	LDR R4,R0,#0		00011_100_000_00000		
8	6		CMP R2,R4		00110_000_010_100_01		
9	7		BCC lab1		11000011_00000010		
10	8		MOV R2,R4		01011_010_100_00000		
11	9	lab1	CMP R3,R4		00110_000_011_100_01		
12	a		BCS lab2		11000010_00000010		
13	b		MOV R3,R4		01011_011_100_00000		
14	c	lab2	ADDI R0,R0,#1		00111_000_000_00001		
15	d		CMP R1,R0		00110_000_001_000_01		
16	e		BNE lab0		11000001_11110111		
17	f		OutR R2		11100_000_010_000_00		
18	10		OurR R3		11100_000_011_000_00		
19	11		HLT		11100_00000000_01		

全圖貌



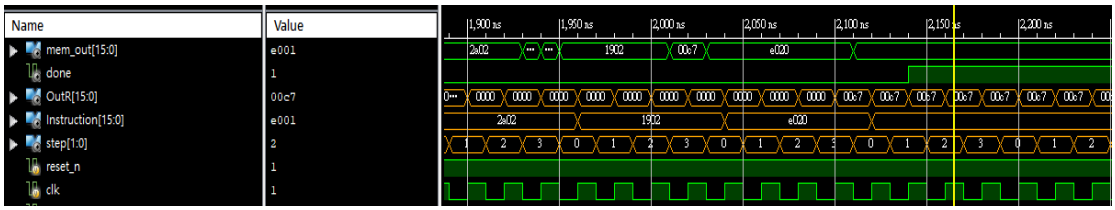
結果放大圖



此程式我將記憶體簡單讀 7 個 data 比較大小，47H，80H、42H、77H、F5H、33H、66H，做比大小，R2 為最小值，R3 為最大值，可以從波形圖看見 R2 為 33H，R3 為 F5H，結果正確。

2. Add two numbers in memory and store the result in memory location.

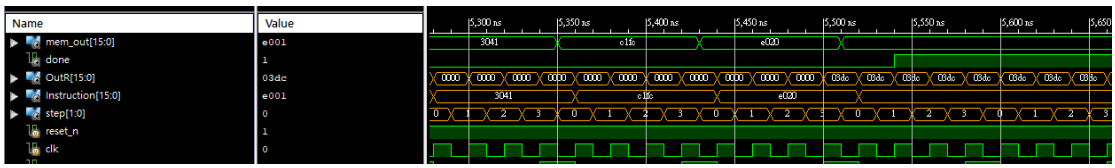
	A	K	L	M	N	O	P
1	HEX	label	ASSEMBLY		COMMAND		
2	0		LLI R0,#30H		00010_000_00110000		
3	1		LDR R3,R0,#0		00011_011_000_00000		
4	2		LDR R4,R0,#1		00011_100_000_00001		
5	3		ADD R2,R3,R4		00000_010_011_100_00		
6	4		STR R2,R0,#2		00101_010_000_00010		
7	5		LDR R1,R0,#2		00011_001_000_00010		
8	6		OutR R1		11100_000_001_000_00		
9	7		HLT		11100_00000000_01		



此程式將記憶體讀兩個數值出來相加後存於另一個記憶體位置，我將相加結果存入後，讀出並 OutR 觀看相加數值，可以從波形圖看見 47H add 80H，結果為 C7H，結果正確。

3. Add ten numbers in consecutive memory locations.

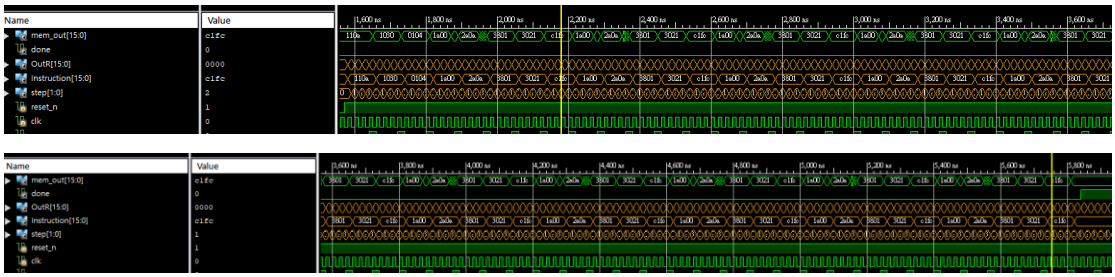
	A	T	U	V	W	X	Y
1	HEX	label	ASSEMBLY		COMMAND		
2	0		LLI R0,#30H		00010_000_00110000		
3	1		LLI R2,#39H		00010_010_00111001		
4	2		LDR R1,R0,#0		00011_001_000_00000		
5	3	lab0	ADDI R0,R0,#1		00111_000_000_00001		
6	4		LDR R3,R0,#0		00011_011_000_00000		
7	5		ADD R1,R1,R3		00000_001_001_011_00		
8	6		CMP R2,R0		00110_000_010_000_01		
9	7		BNE lab0		11000001_11111100		
10	8		OutR R1		11100_000_001_000_00		
11	9		HLT		11100_00000000_01		



此程式將記憶體裡的十個數字做相加，47H、80H、42H、77H、F5H、33H、66H、12H、35H、87H 的相加結果為 3DCH，可從波形圖看見結果正確。

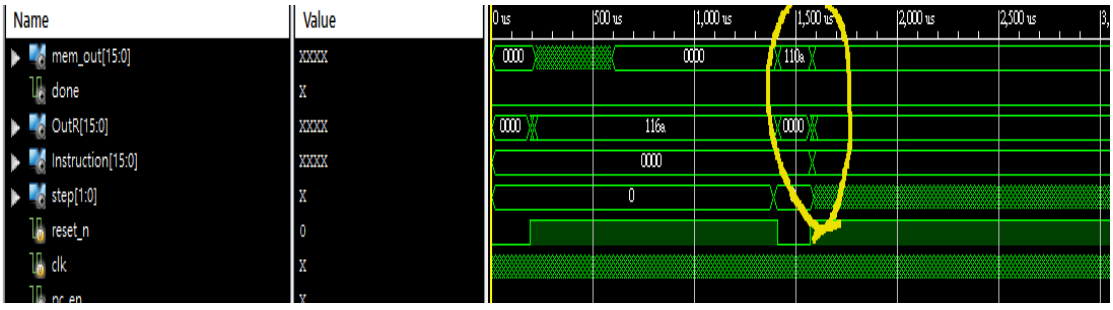
4. Mov a memory block of N words from one place to another.

AB	AC	AD	AE	AF	AG	AH
HEX	label	ASSEMBLY		COMMAND		
0		LLI R1,#10		00010_001_00001010		
1		LLI R0, #30H		00010_000_00110000		
2		ADD R1,R0,R1		00000_001_000_001_00		
3	lab0	LDR R2,R0,#0		00011_010_000_00000		
4		STR R2,R0,#10		00101_010_000_01010		
5		ADDI R0,R0,#1		00111_000_000_00001		
6		CMP R1,R0		00110_000_001_000_01		
7		BNE lab0		11000001_11111100		
8		HLT		11100_00000000_01		



此程式將記憶體中 N 個 word 當作 block 移動到其他位置，我讓組語 R1 為 N=10，波形圖可以看見每個 Instruction 的變化，來觀看確實完成正確的讀取寫入的迴圈。

這是其中一個 testbench 的 timing(post-route)的結果圖



我發現我的 post-route 部分的執行結果是錯誤的，在 deadline 之前都無法像 behavioral 一樣正確執行，所以上面部分沒有放入 timing(post-route)模擬。

Hardware Cost

Total number of LUTs 1545

Total number of FFs 34

16-bits RISC CPU hardware cost

Device Utilization Summary				[-]
Slice Logic Utilization	**Used**	**Available**	**Utilization**	**Note(s)**
Number of Slice Registers	34	93,120	1%	
Number used as Flip Flops	34			
Number used as Latches	0			
Number used as Latch-thrus	0			
Number used as AND/OR logics	0			
Number of Slice LUTs	1,545	46,560	3%	
Number used as logic	1,481	46,560	3%	
Number using O6 output only	1,480			
Number using O5 output only	0			
Number using O5 and O6	1			
Number used as ROM	0			
Number used as Memory	64	16,720	1%	
Number used as Dual Port RAM	0			
Number used as Single Port RAM	64			
Number using O6 output only	0			
Number using O5 output only	0			
Number using O5 and O6	64			
Number used as Shift Register	0			
Number used exclusively as route-thrus	0			
Number of occupied Slices	587	11,640	5%	
Number of LUT Flip Flop pairs used	1,545			

Discussion

1. Design entry

   原先我在設計作業的時候，我看到 Register File 需要 3to8 decoder 我就設計 3to8 decoder 然後去 schematic 上畫電路……。

   做到 ALU 的時候，我設計 adder 的結果與邏輯閘來表示 NZCV bit……。

   當我做到 Decoder 的時候總會發現，可能我 Datapath 哪裡需要輸出什麼，而去補一些腳位。在設計時序的控制會發現我的系統可能哪邊會有問題，必須回去修改，我認為這是我從這次報告中所需檢討的，我應當在清晰的腦袋下完成，整個系統的流程思路，而非一個一個想好就做了。

2. Timing simulation error

   在完成設計每個部份的時候，我都是只有跑 behavioral simulation，在完成整個 CPU 設計的時候跑 timing simulation 發現錯誤我無法在報告繳交時間內解決錯誤問題，另一部份是我不清楚要在 timing simulation 下有正確的功能，有什麼時序上的設計要做注意才不會出現錯誤結果，也希望老師可以做一些設計 CPU 要注意的事項檢討。

3. CPU timing

   原先我只知道 CPU 的架構組成，但是我無法細說每一個邏輯閘、線的動作如何組合，經過長時間的做作業的時間，我從作業參考內容，以及網路上的解說，來完成出 RISC_16b_CPU，讓我在這個作業中重新複習了 CPU 內的架構，也已經實際了解內部每個部份(wire)的運作，但是我在整個的架構中雖然做出了功能模擬正確，但是 timing(post-route)那部分錯誤令我非常擔心說，我的設計觀念有嚴重的錯誤，但我找不到錯誤而持續的錯誤下去，導致我未來其他設計或者在期末報告也出現錯誤，我希望且必須解決此問題，來提升自我，達到下一個境界。