

2024/03/06

實驗二

組合邏輯練習

姓名：黃文祺 學號：01057013

班級：資工 3A

E-mail：wenchi971244202@gmail.com

注意

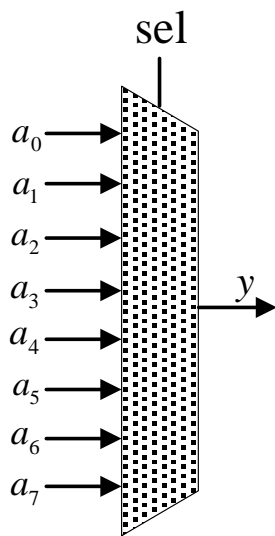
1. 一律將此檔轉成 PDF 檔
繳交
2. 繳交期限為下周上課前
3. 一人繳交一份
4. 檔名：學號_HW?.pdf
檔名請按照作業檔名格式進行填寫
未依照格式不予批改

一、多工器

● 實驗說明：

1. 實作多工器及 testbench。
2. 輸入資料 a_n [2:0]： $a_0=0$ 、 $a_1=1$ 、 $a_2=2$ 、 $a_3=3$ 、 $a_4=4$ 、 $a_5=5$ 、 $a_6=6$ 、 $a_7=7$
3. 輸出： y [2:0]
4. Testbench 內容為 sel 由 0 到 7

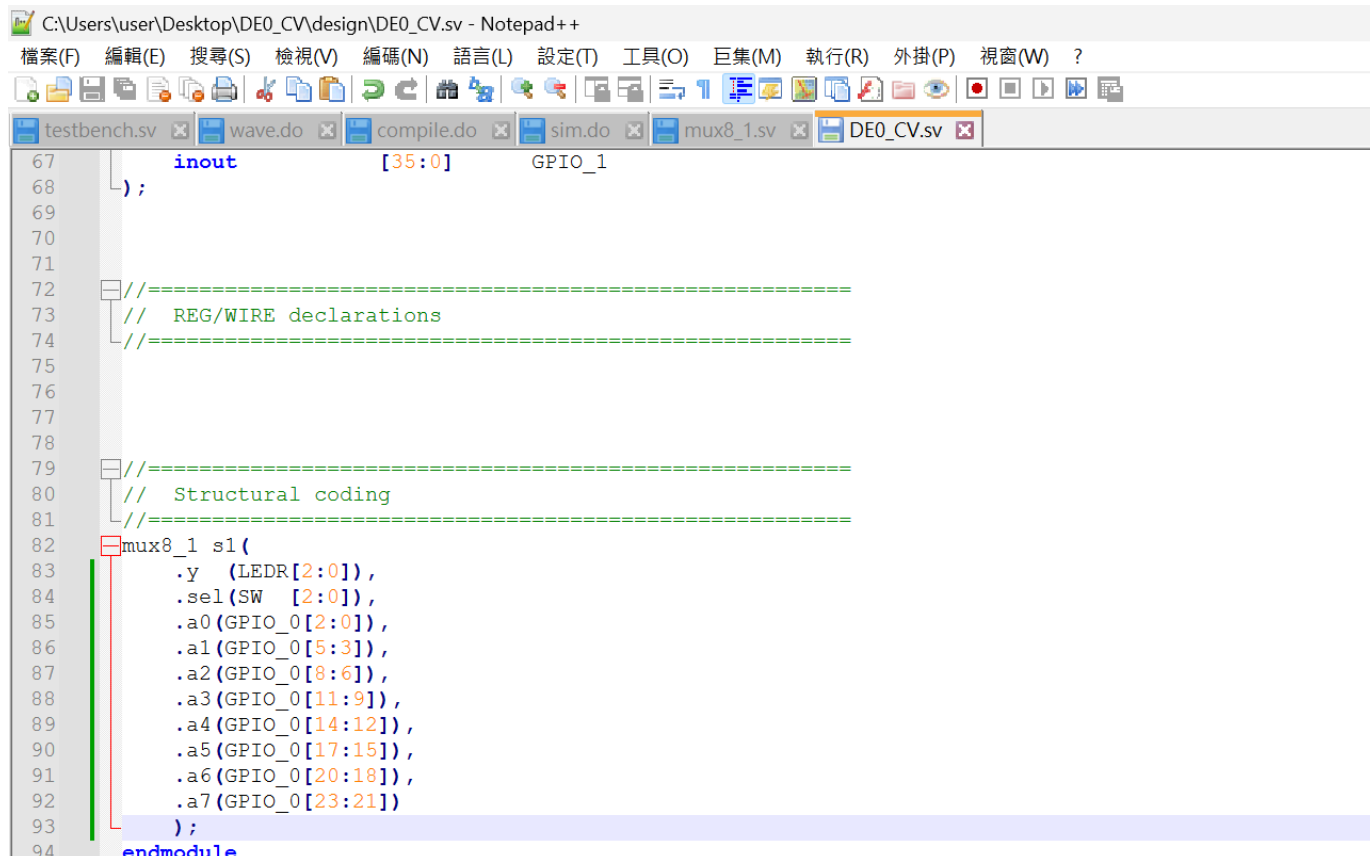
● 系統硬體架構方塊圖（接線圖）：



● 系統架構程式碼、測試資料程式碼與程式碼說明

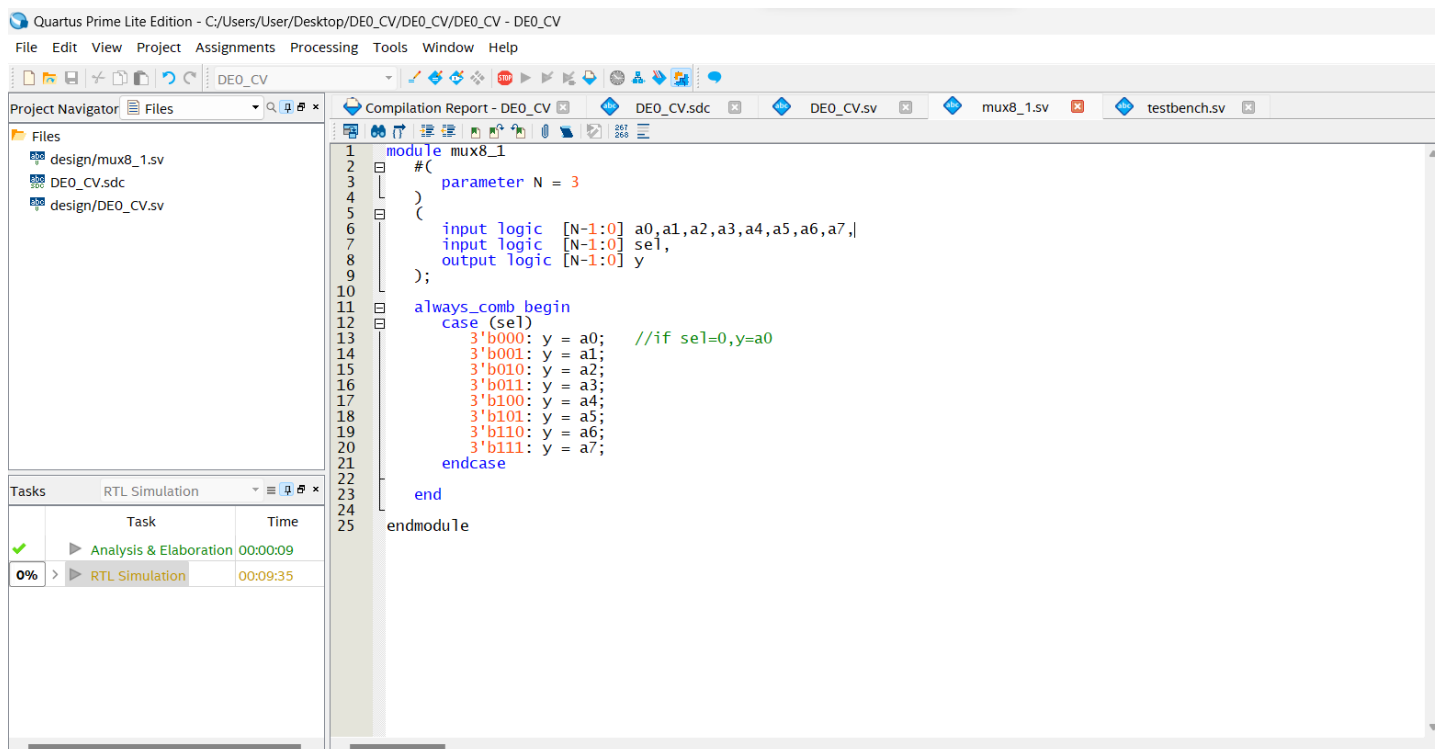
截圖請善用 win+shift+S

DE0_CV.sv (只有改動 mux8_1 的部分) :



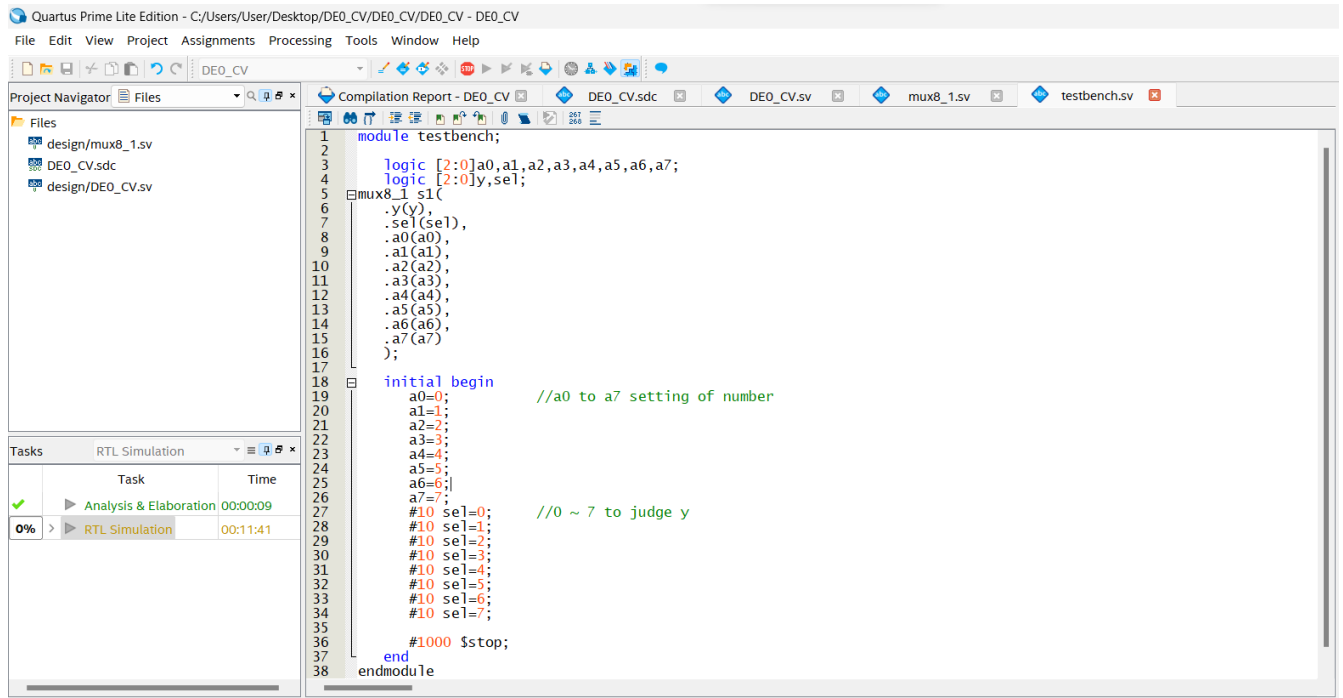
```
67      inout          [35:0]      GPIO_1
68  );
69
70
71
72  //=====
73  //  REG/WIRE declarations
74  //=====
75
76
77
78
79  //=====
80  //  Structural coding
81  //=====
82
83  mux8_1 s1(
84      .y (LEDR[2:0]),
85      .sel (SW [2:0]),
86      .a0 (GPIO_0[2:0]),
87      .a1 (GPIO_0[5:3]),
88      .a2 (GPIO_0[8:6]),
89      .a3 (GPIO_0[11:9]),
90      .a4 (GPIO_0[14:12]),
91      .a5 (GPIO_0[17:15]),
92      .a6 (GPIO_0[20:18]),
93      .a7 (GPIO_0[23:21])
94  );
95  endmodule
```

mux8_1.sv:



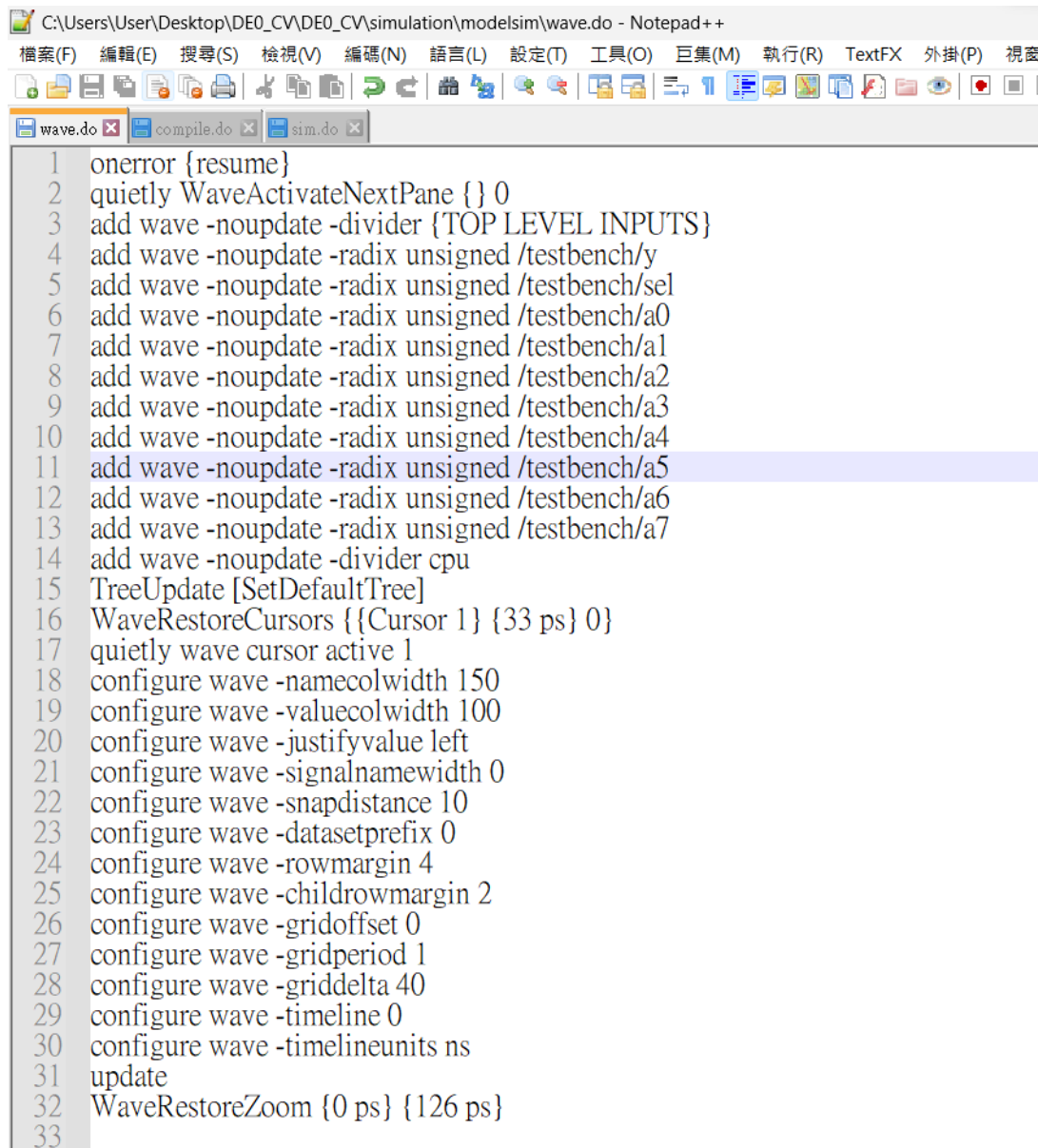
```
1  module mux8_1
2  #
3      parameter N = 3
4  )
5  (
6      input logic [N-1:0] a0,a1,a2,a3,a4,a5,a6,a7,
7      input logic [N-1:0] sel,
8      output logic [N-1:0] y
9  );
10
11  always_comb begin
12      case (sel)
13          3'b000: y = a0; //if sel=0,y=a0
14          3'b001: y = a1;
15          3'b010: y = a2;
16          3'b011: y = a3;
17          3'b100: y = a4;
18          3'b101: y = a5;
19          3'b110: y = a6;
20          3'b111: y = a7;
21      endcase
22  end
23
24  end
25  endmodule
```

testbench.sv:



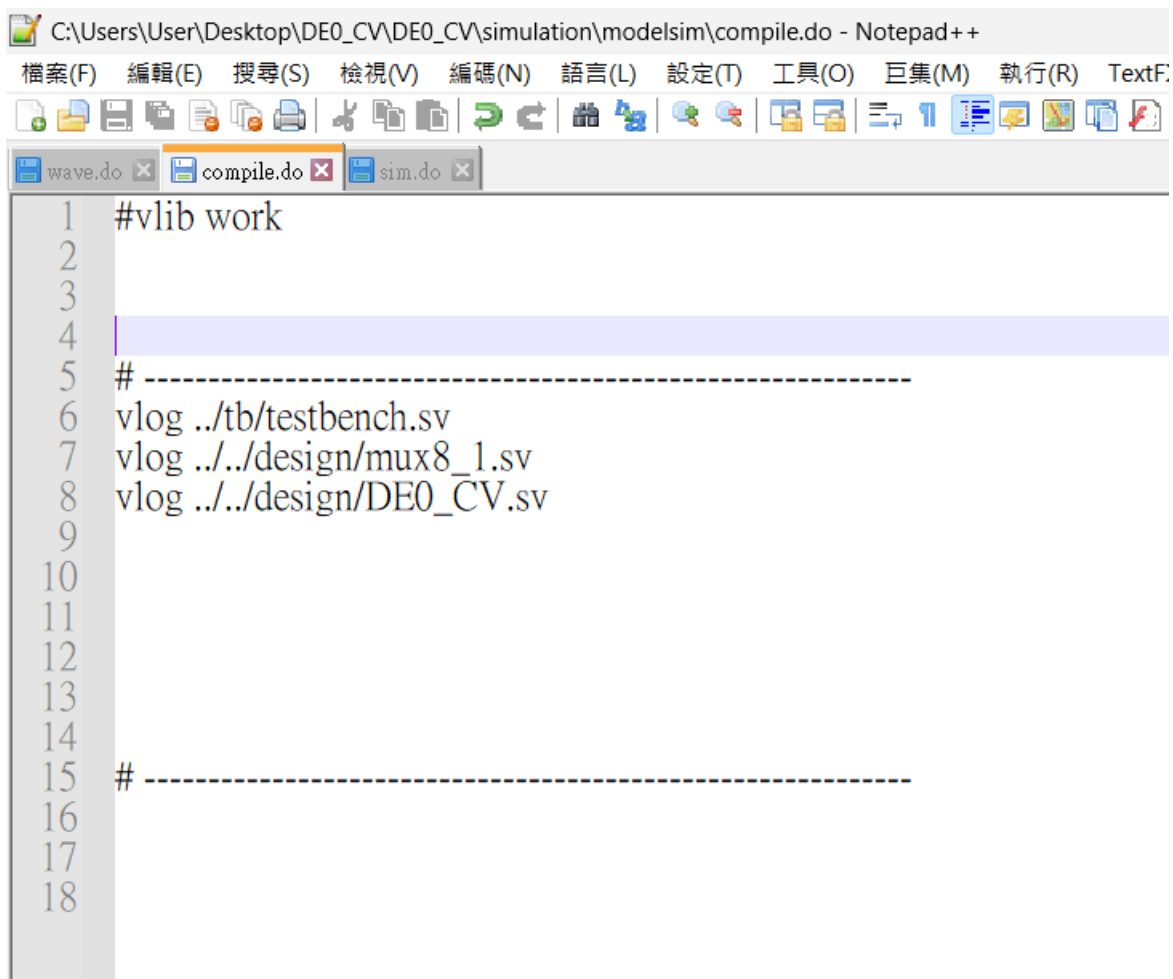
```
1 module testbench;
2
3     logic [2:0] a0,a1,a2,a3,a4,a5,a6,a7;
4     logic [2:0] y,sel;
5     mux8_1 s1(
6         .y(y),
7         .sel(sel),
8         .a0(a0),
9         .a1(a1),
10        .a2(a2),
11        .a3(a3),
12        .a4(a4),
13        .a5(a5),
14        .a6(a6),
15        .a7(a7)
16    );
17
18    initial begin
19        //a0 to a7 setting of number
20        a0=0;
21        a1=1;
22        a2=2;
23        a3=3;
24        a4=4;
25        a5=5;
26        a6=6;
27        a7=7;
28        #10 sel=0; //0 ~ 7 to judge y
29        #10 sel=1;
30        #10 sel=2;
31        #10 sel=3;
32        #10 sel=4;
33        #10 sel=5;
34        #10 sel=6;
35        #10 sel=7;
36        #1000 $stop;
37    end
38 endmodule
```

wave.do: (只有 add y,sel,a0~a7)



```
1 onerror {resume}
2 quietly WaveActivateNextPane {} 0
3 add wave -noupdate -divider {TOP LEVEL INPUTS}
4 add wave -noupdate -radix unsigned /testbench/y
5 add wave -noupdate -radix unsigned /testbench/sel
6 add wave -noupdate -radix unsigned /testbench/a0
7 add wave -noupdate -radix unsigned /testbench/a1
8 add wave -noupdate -radix unsigned /testbench/a2
9 add wave -noupdate -radix unsigned /testbench/a3
10 add wave -noupdate -radix unsigned /testbench/a4
11 add wave -noupdate -radix unsigned /testbench/a5
12 add wave -noupdate -radix unsigned /testbench/a6
13 add wave -noupdate -radix unsigned /testbench/a7
14 add wave -noupdate -divider cpu
15 TreeUpdate [SetDefaultTree]
16 WaveRestoreCursors {{Cursor 1} {33 ps} 0}
17 quietly wave cursor active 1
18 configure wave -namecolwidth 150
19 configure wave -valuecolwidth 100
20 configure wave -justifyvalue left
21 configure wave -signalnamewidth 0
22 configure wave -snapdistance 10
23 configure wave -datasetprefix 0
24 configure wave -rowmargin 4
25 configure wave -childrowmargin 2
26 configure wave -gridoffset 0
27 configure wave -gridperiod 1
28 configure wave -griddelta 40
29 configure wave -timeline 0
30 configure wave -timelineunits ns
31 update
32 WaveRestoreZoom {0 ps} {126 ps}
33
```

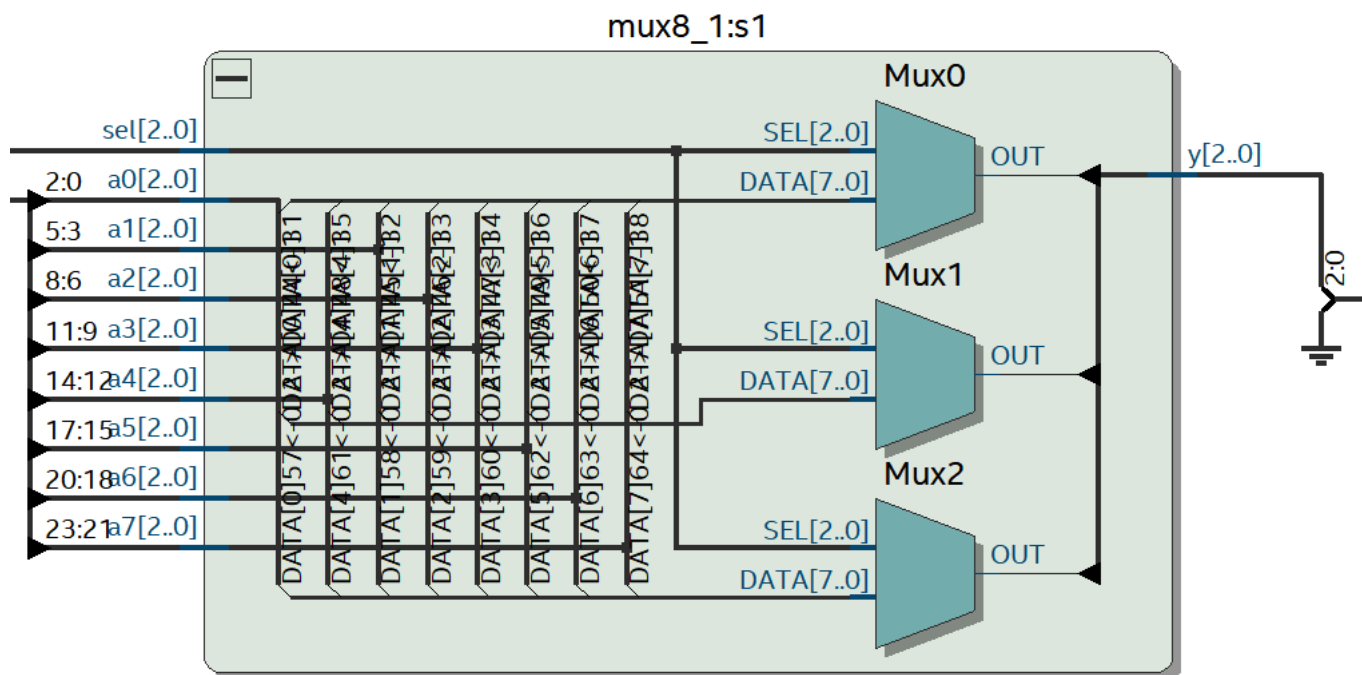
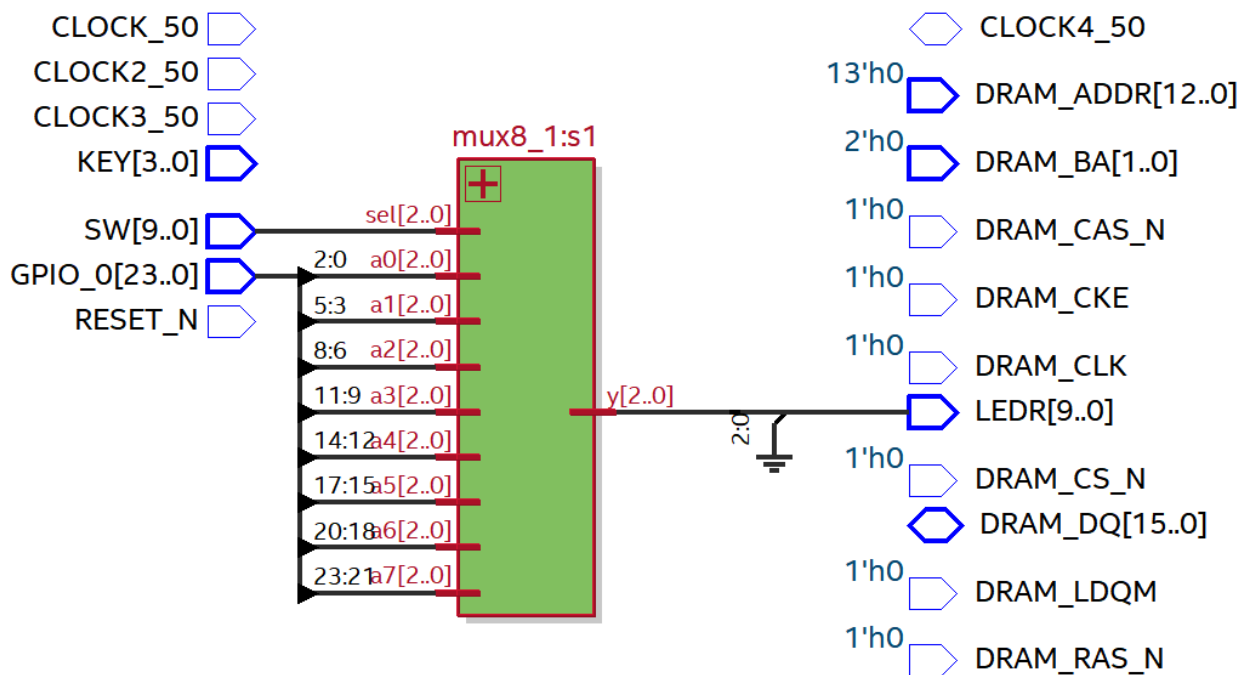
compile.do: (只有更改為 mux8_1.sv)



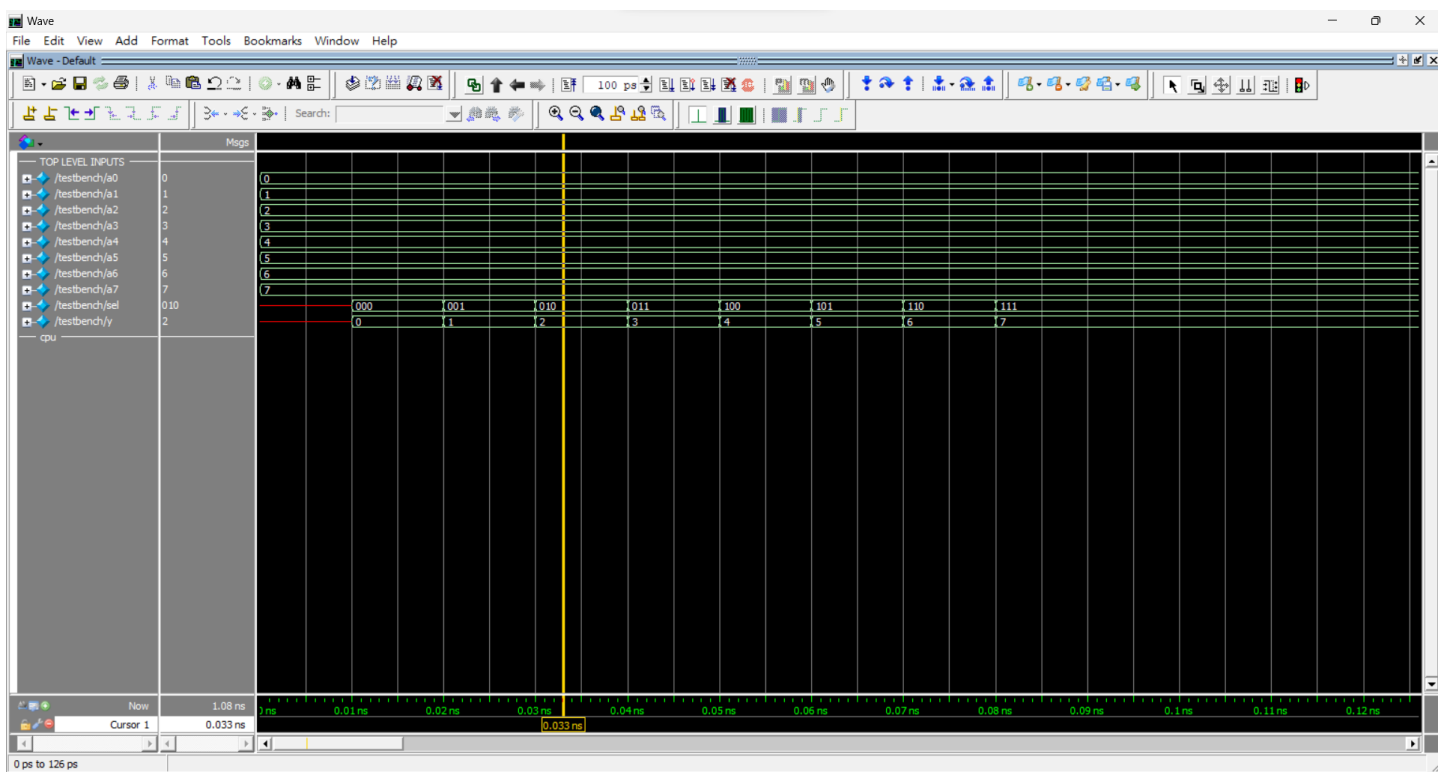
```
1 #vlib work
2
3
4
5 # -----
6 vlog ../tb/testbench.sv
7 vlog ../../design/mux8_1.sv
8 vlog ../../design/DE0_CV.sv
9
10
11
12
13
14
15 # -----
16
17
18
```

其它: DE0_CV.sdc 及 sim.do 和第一次作業一樣，沒有更改。

● RTL Viewer 截圖



● 模擬結果與結果說明：



(a0~a7 持續設為 0 到 7,而 sel 從 000 到 111，分別會使 y 對應到 a0~a7)

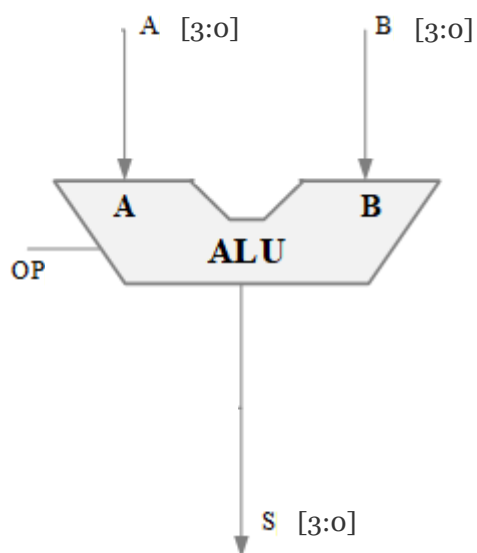
二、算術邏輯單元(ALU)

● 實驗說明：

1. 實作 ALU 及 testbench，跑模擬波形。
2. $S = 5 + B$
3. $S = D - 7$
4. A[3:0]、B[3:0]、op 分別輸入資料(16 進位)：5, B, 0。
5. A[3:0]、B[3:0]、op 分別輸入資料(16 進位)：D, 7, 1。
6. 輸入：A[3:0], B[3:0], op
7. 輸出：S[3:0]

OP	ALU 運算	註解
0	$S = A + B$	加法
1	$S = A - B$	減法

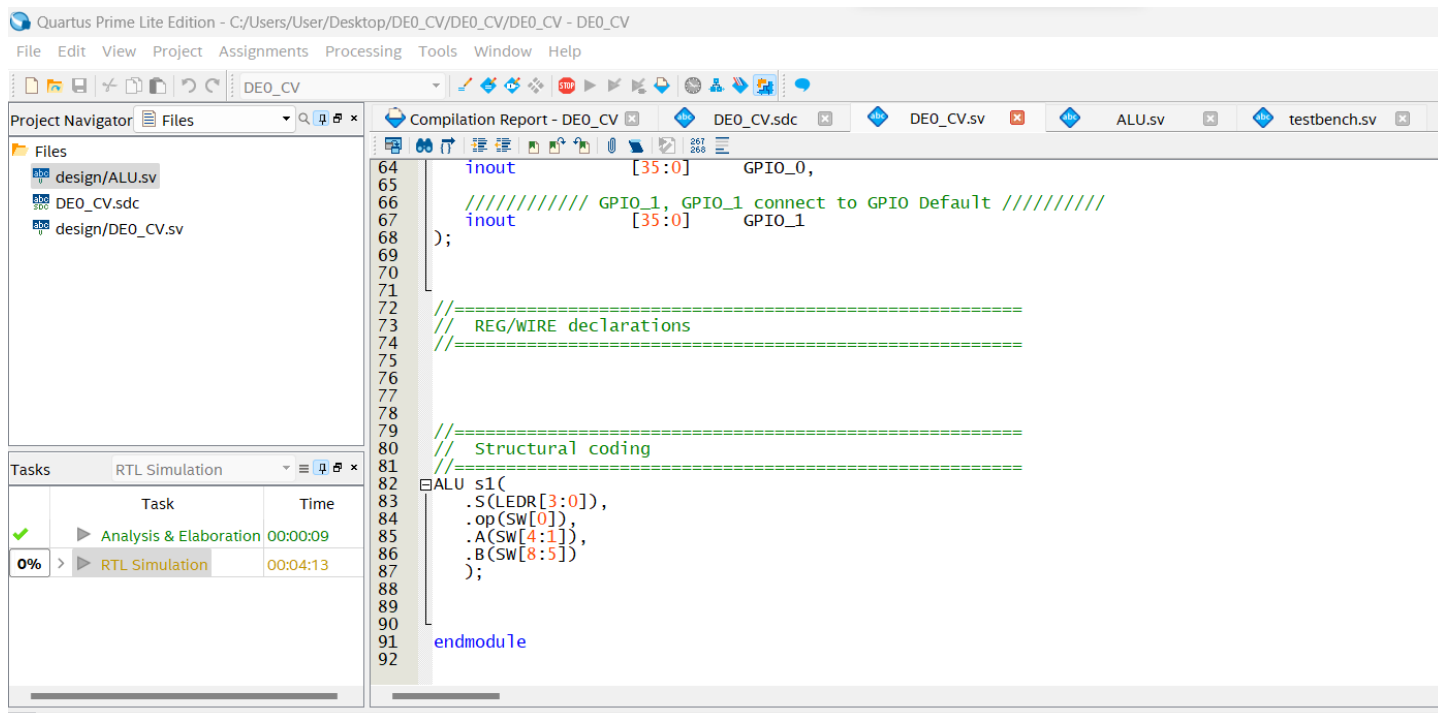
● 系統硬體架構方塊圖 (接線圖)：



● 系統架構程式碼、測試資料程式碼與程式碼說明

截圖請善用 win+shift+S

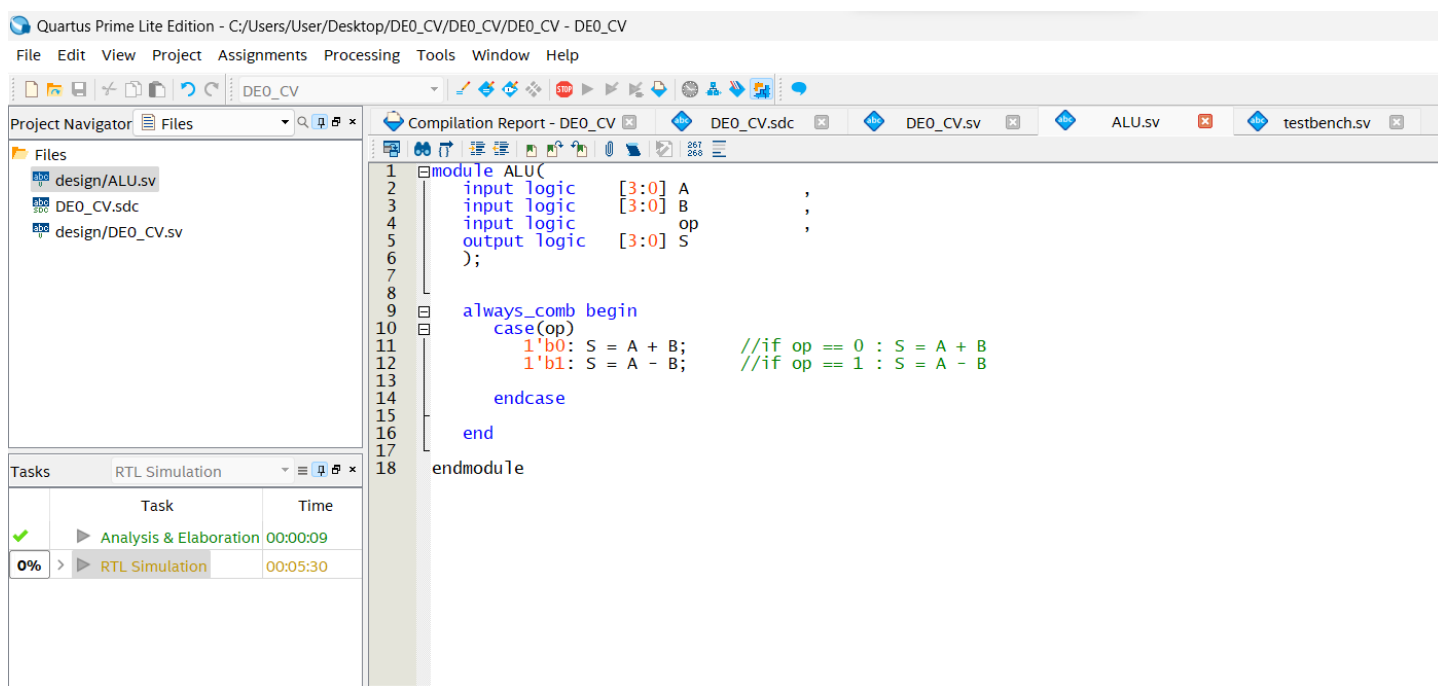
DE0_CV.sv: (只有改 ALU 的部分)



```
64      inout      [35:0]      GPIO_0,
65
66      //////////// GPIO_1, GPIO_1 connect to GPIO Default ////////////
67      inout      [35:0]      GPIO_1
68  );
69
70
71
72  //=====
73  // REG/WIRE declarations
74  //=====
75
76
77
78  //=====
79  // Structural coding
80  //=====
81
82  ALU s1(
83      .S(LED[3:0]),
84      .op(SW[0]),
85      .A(SW[4:1]),
86      .B(SW[8:5])
87  );
88
89
90
91  endmodule
92
```

Task	Time
Analysis & Elaboration	00:00:09
RTL Simulation	00:04:13

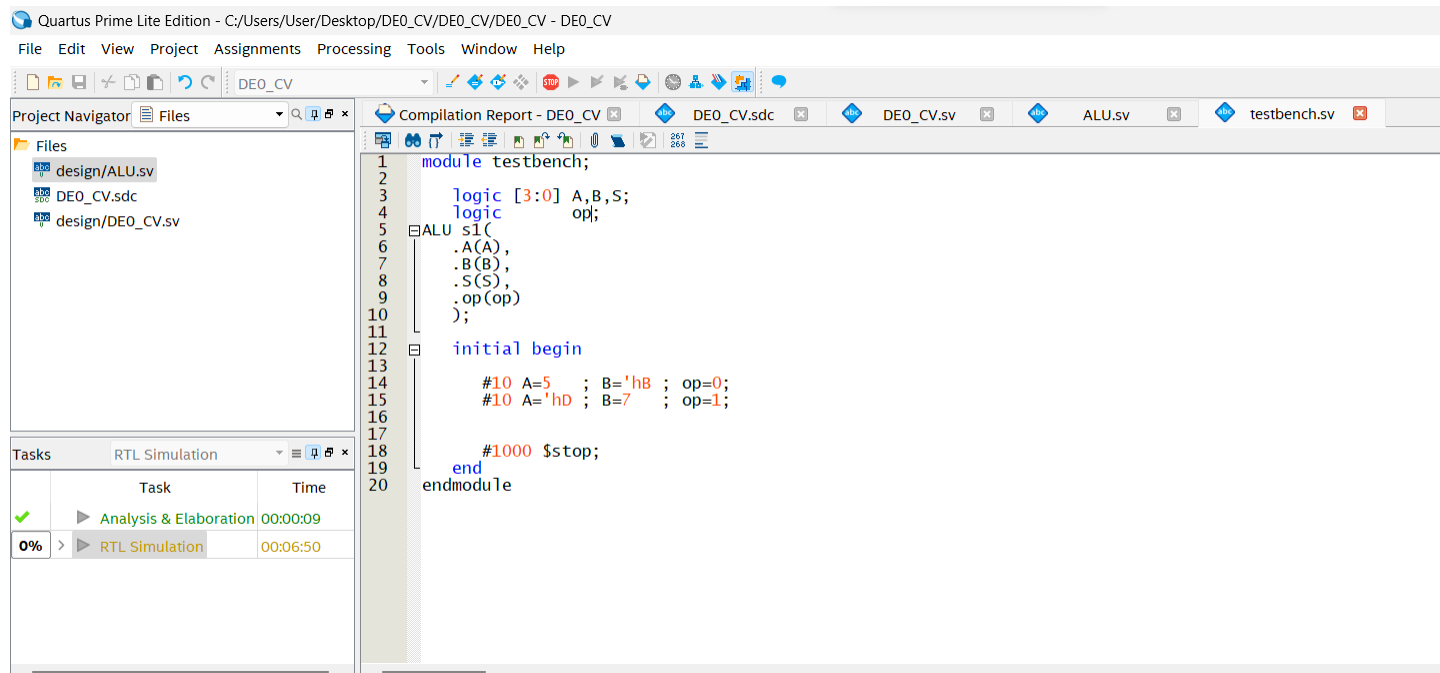
ALU.sv:



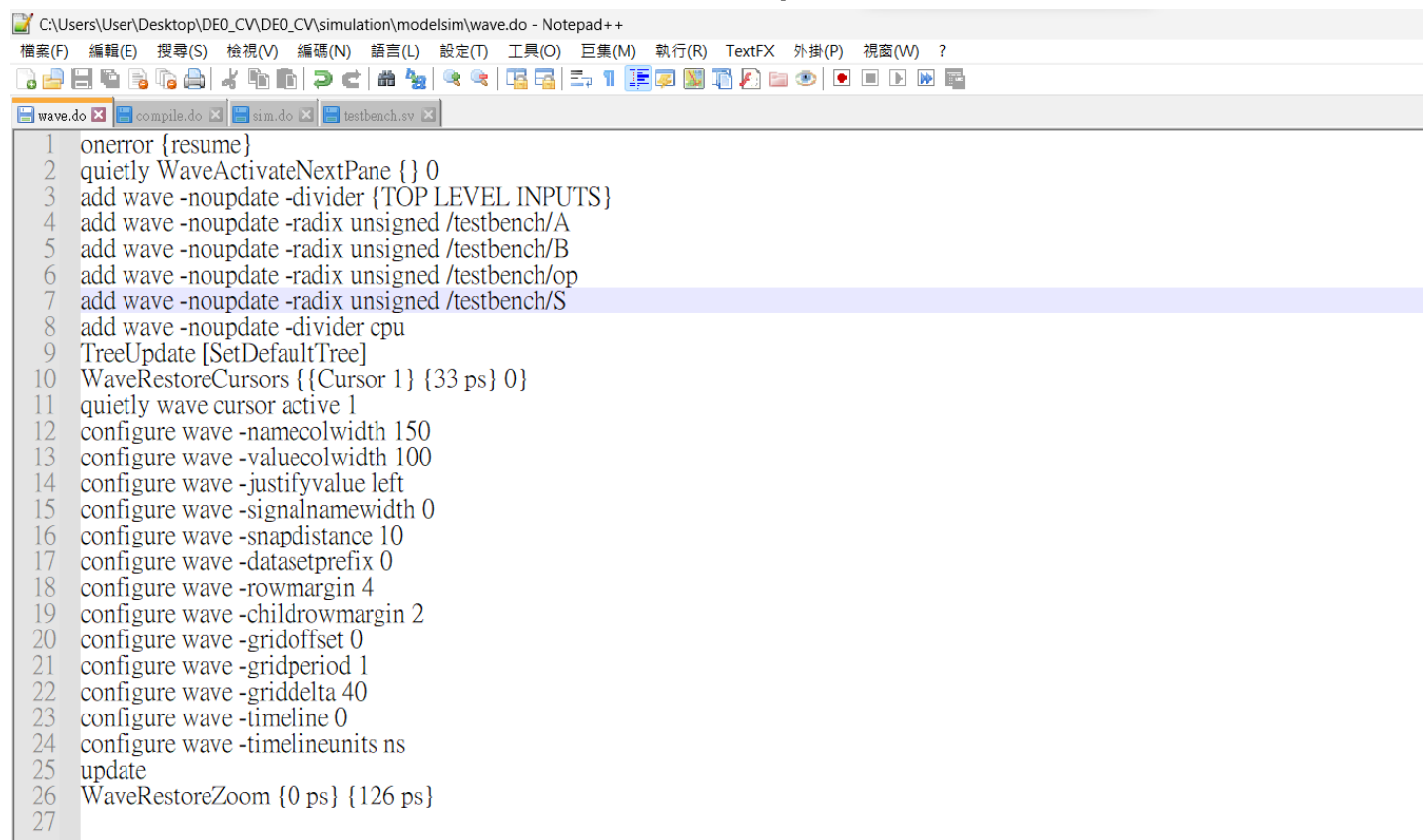
```
1  module ALU(
2      input logic [3:0] A,
3      input logic [3:0] B,
4      input logic op,
5      output logic [3:0] S
6  );
7
8
9      always_comb begin
10         case(op)
11             1'b0: S = A + B; //if op == 0 : S = A + B
12             1'b1: S = A - B; //if op == 1 : S = A - B
13         endcase
14     end
15
16 endmodule
17
18
```

Task	Time
Analysis & Elaboration	00:00:09
RTL Simulation	00:05:30

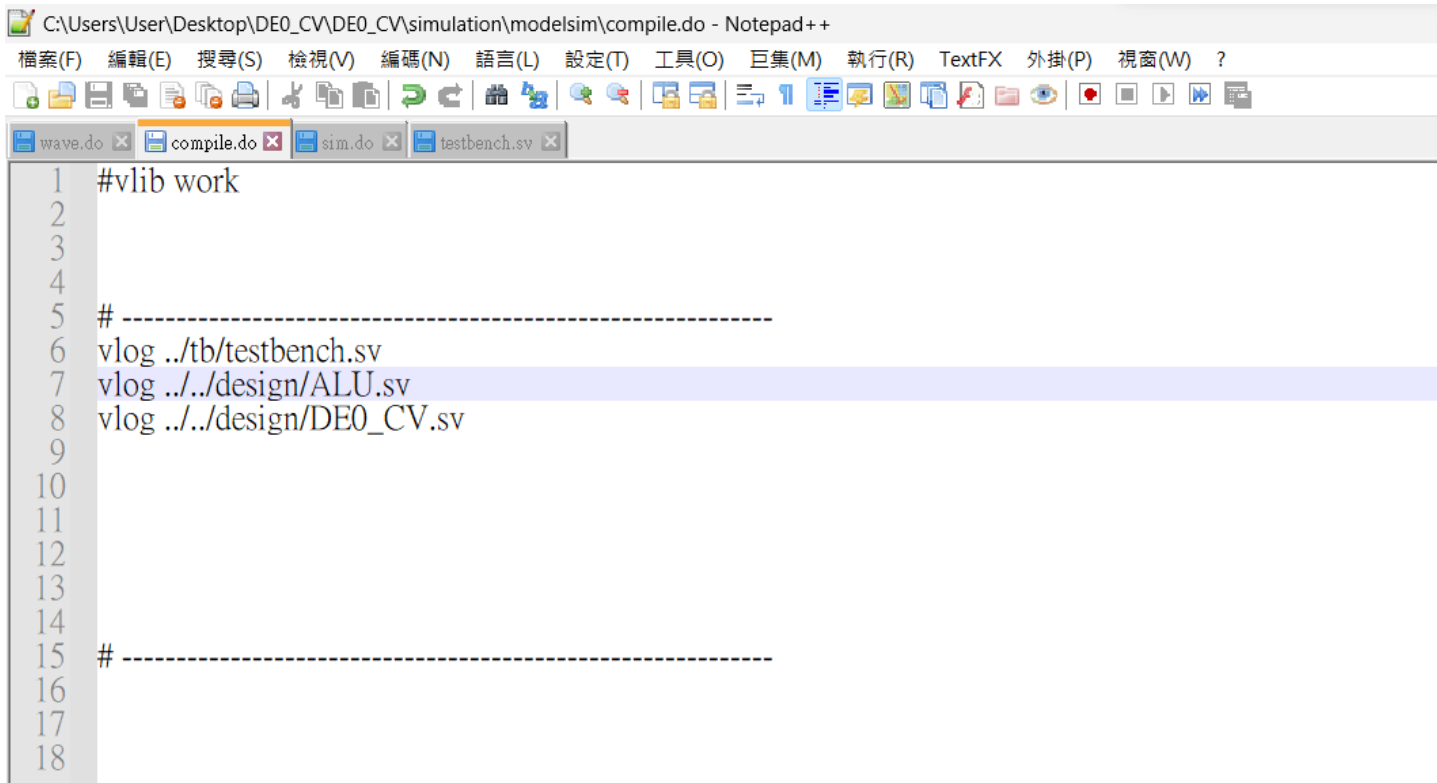
testbench.v:



wave.do : (更改 4 條 add 分別為 A,B,S,op)



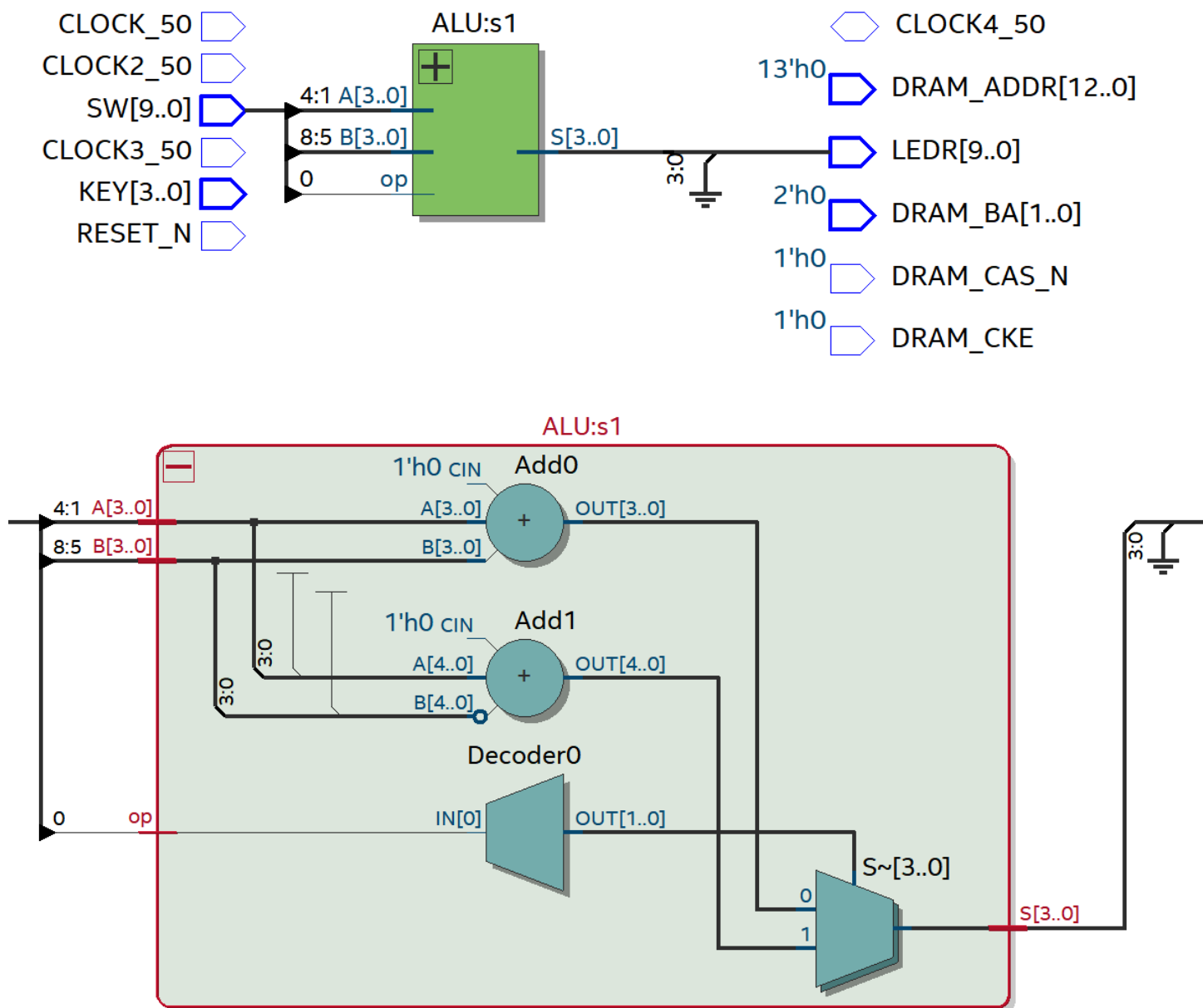
compile.do: (更改為 ALU.sv)



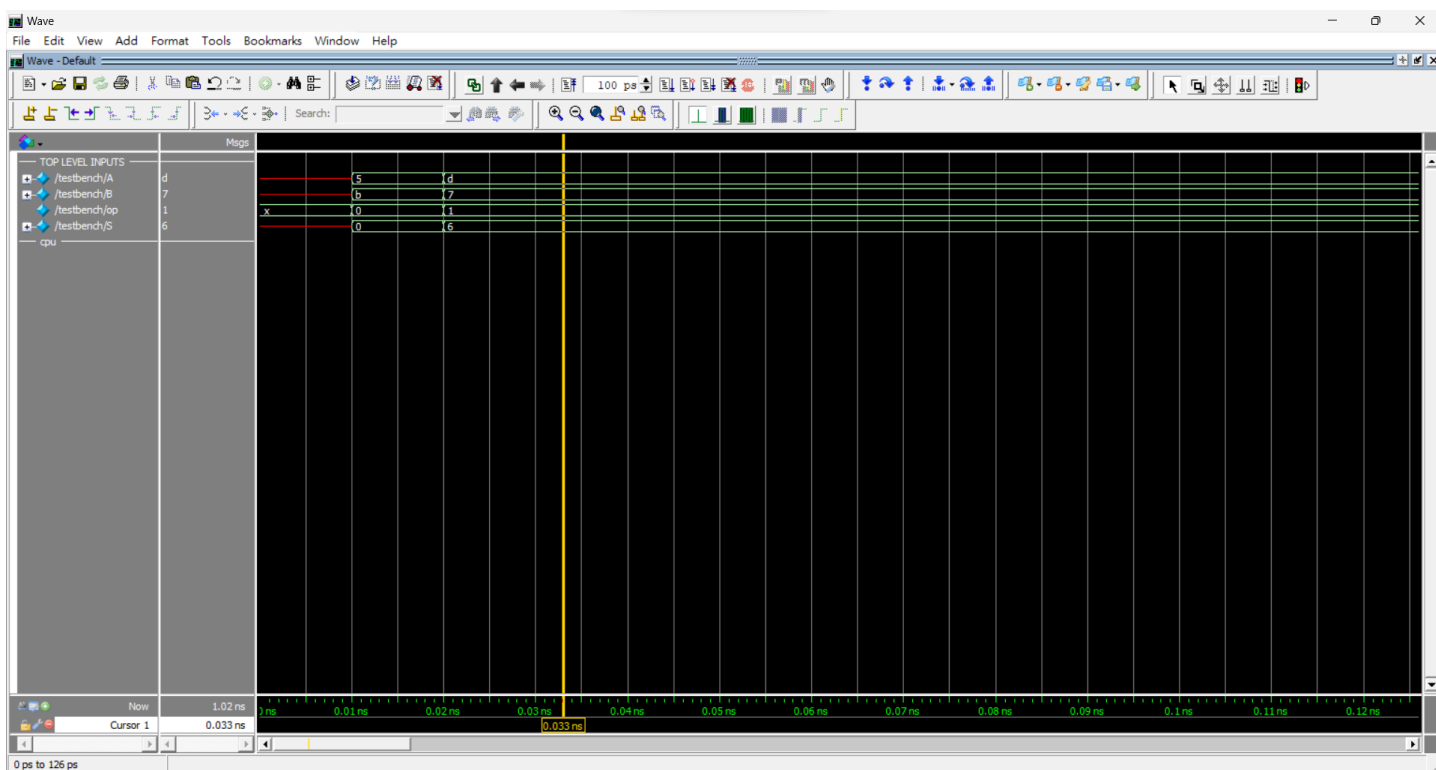
```
1 #vlib work
2
3
4
5 # -----
6 vlog ../tb/testbench.sv
7 vlog ../../design/ALU.sv
8 vlog ../../design/DE0_CV.sv
9
10
11
12
13
14
15 # -----
16
17
18
```

其它: DE0_CV.sdc 及 sim.do 和第一次作業一樣，沒有更改。

● RTL Viewer 截圖



● 模擬結果與結果說明：



(當 op 為 0 時，執行 $S=A+B$ ，但由於相加和為 16，溢位了，所以顯示為 0，若有做溢位偵測，則可令 $overflow=1$ ，而當 op 為 1 時，執行 $S=A-B$ 。)

● 結論與心得：

這次實驗相較上次只有執程式碼而言，實際編輯程式碼到完成作業，遇到問題再一一解決後，也讓我對 Quartus 及 ModelSim 的操作更加熟，而這次的作業主要用到 case 的語法，依樣畫葫蘆完其實沒有太大難度。