

用 Block Design 與 Verilog 完成 OR 閘以及 NOT 閘。

完成 TestBench。

實驗內容

一、and gate

(一) and_gate.v :and 邏輯閘運算的程式碼

第 6、13 行: `module` `endmodule`

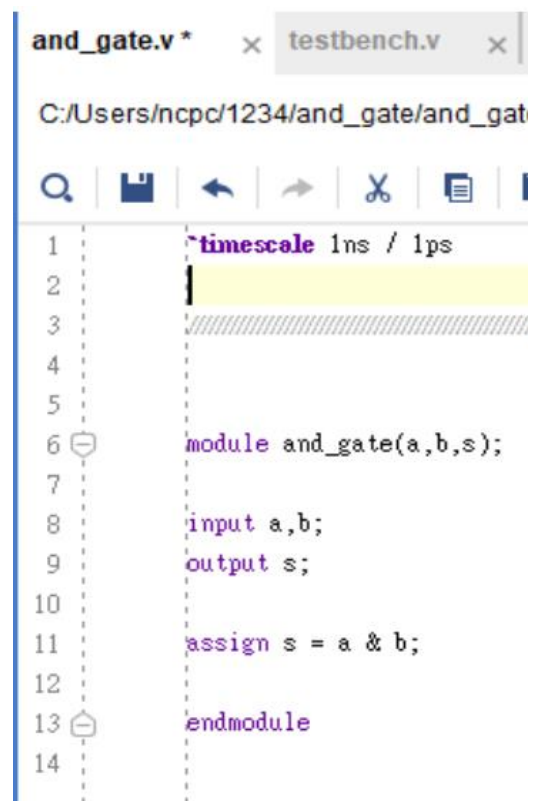
// and_gate 的開始及結束

第 8、9 行: `input a,b;` `output s;`

//電流輸入端 a、b，輸出端 s

第 11 行: `assign s = a & b;`

//assign：要求指定的訊號線要做什麼運算。a 和 b 執行 and 運算的結果指定給 s



```
1  timescale 1ns / 1ps
2
3
4
5
6  module and_gate(a,b,s);
7
8      input a,b;
9      output s;
10
11     assign s = a & b;
12
13 endmodule
14
```

(二) testbench.v :對 HDL 的電路進行仿真驗證

第 24、25 行: **reg** a,b; **wire** s;
// wire 和 reg 都類似於 C 的變數，但若此變數要放在 begin...end 內，該變數就須使用 reg。即 reg 變數表示輸入，wire 表示輸出。

第 27 行: and_gate U0(.a(a).b(b).s(s))
// and_gate 對應 and_gate.v 檔，而.a(a)中前面的 a 對應 and_gate.v 檔的變數 a，後面的 a 則對應到 testbench.v 檔的變數 a

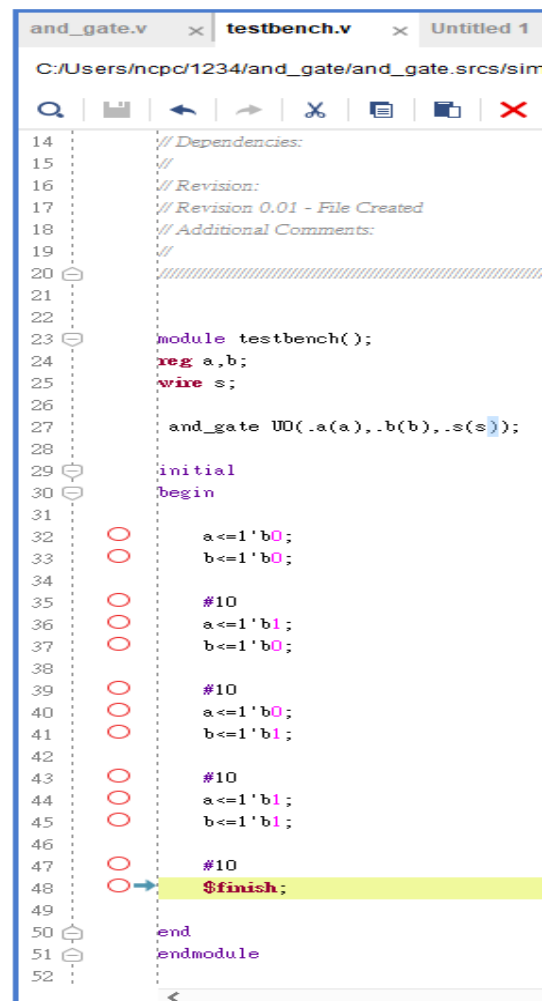
第 29、30、50 行: **initial** **begin** **end**
// 以 initial 為主的程式區塊，只會在一開始時執行一次。從 begin 開始到 end 結束，所有的 initial block 皆同時執行。initial 通常用於 Testbench

第 32 行: a<=1'b0;
//將 a 宣告為一位元二進制之值為 0
** b=二進制 o=八進制**
** d=十進制 h=十六進制**

第 35 行: **#10**
//等待 10 毫秒(0.01s)

第 48 行: **\$finish;**
//結束 begin 到 end 的程式，因為此程式僅存在單個 begin 到 end，即可視為程式結束


第 51 行: **endmodule**
//名為 testbench()的 module 從 23 行開始到 endmodule 結束



```
and_gate.v x testbench.v x Untitled 1
C:/Users/ncpc/1234/and_gate/and_gate.srcs/sim

14 // Dependencies:
15 //
16 // Revision:
17 // Revision 0.01 - File Created
18 // Additional Comments:
19 //
20 ///////////////////////////////////////////////////////////////////
21
22
23 module testbench();
24     reg a,b;
25     wire s;
26
27     and_gate U0(.a(a),.b(b),.s(s));
28
29     initial
30     begin
31
32         a<=1'b0;
33         b<=1'b0;
34
35         #10
36         a<=1'b1;
37         b<=1'b0;
38
39         #10
40         a<=1'b0;
41         b<=1'b1;
42
43         #10
44         a<=1'b1;
45         b<=1'b1;
46
47         #10
48         $finish;
49
50     end
51 endmodule
52
```

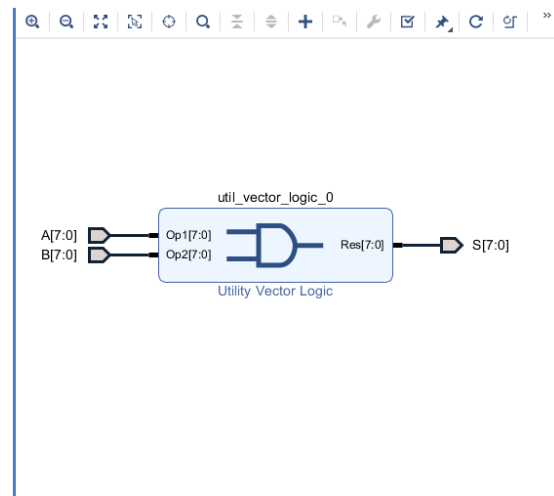
(三) FPGA : 電路圖

 :and 運算邏輯閘，此為 2 輸入 1 輸出的邏輯閘

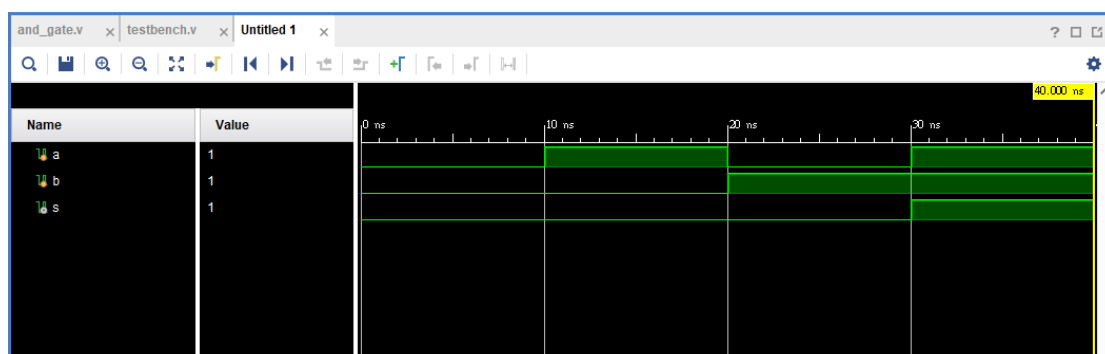
A[7:0]-----Op1[7:0]
// 8 位元的 data 由 A 輸往輸入端 Op1

B[7:0]-----Op2[7:0]
// 8 位元的 data 由 B 輸往輸入端 Op2

Res[7:0]-----S[7:0]
// 8 位元的 data 由輸出端 Res 輸往 S 端



(四) 仿真驗證圖



a、b 電壓根據 testbench 而變，s 亦根據 and_gate.v 而變

由此可知程式碼沒錯。

而此次實驗有幾個要注意的點:

1. and_gate U0(.a(a).b(b).s(s))

// and_gate 對應 and_gate.v 檔，而.a(a)中前面的 a 對應 and_gate.v 檔的變數 a，
後面的 a 則對應到 testbench.v 檔的變數 a

2. assign

//assign：要求指定的訊號線要做什麼運算。

3. \$finish;

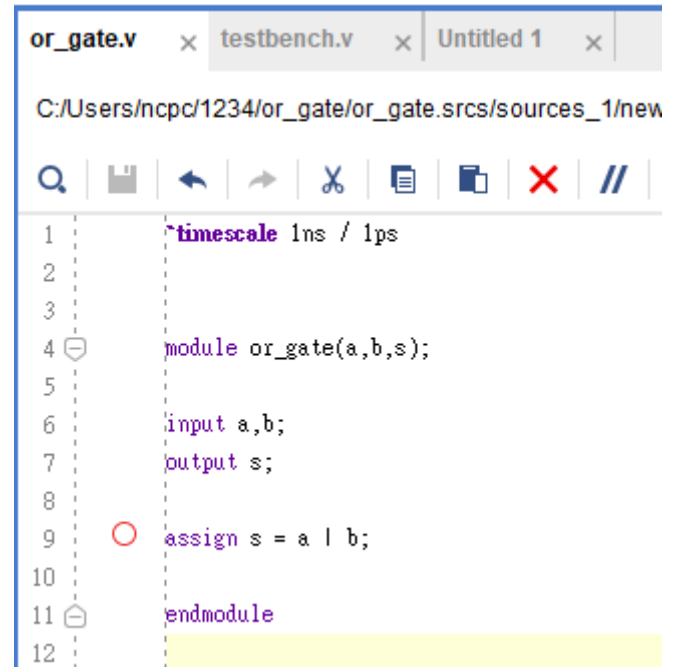
//結束 begin 到 end 的程式

二、or gate

(一) or_gate.v :or 邏輯閘運算的程式碼

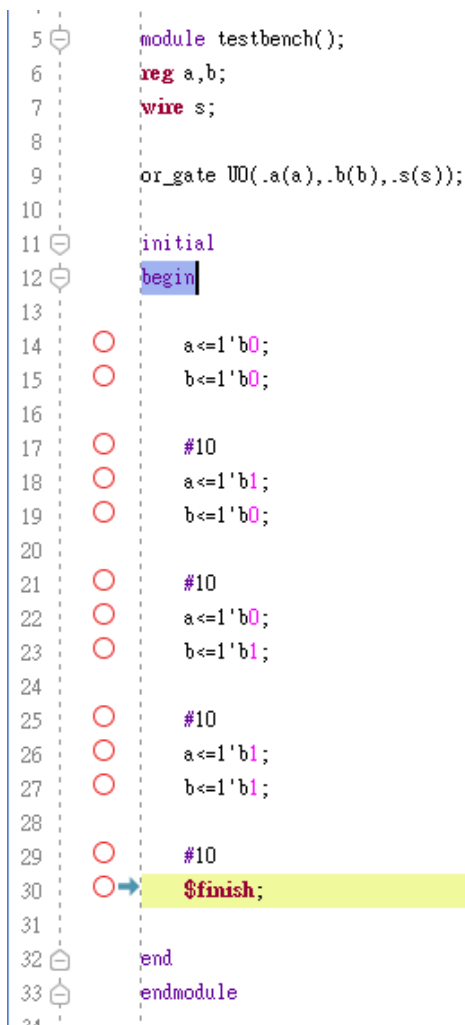
類似前面的 and_gate。唯一的不同在第 9 行

第 9 行: `assign s = a | b;`
// a 和 b 執行 or 運算的結果指定給 s



```
or_gate.v x testbench.v x Untitled 1 x
C:/Users/ncpc/1234/or_gate/or_gate.srscs/sources_1/new

1 timescale 1ns / 1ps
2
3
4 module or_gate(a,b,s);
5
6 input a,b;
7 output s;
8
9 assign s = a | b;
10
11 endmodule
12
```



```
5 module testbench();
6 reg a,b;
7 wire s;
8
9 or_gate U0(.a(a),.b(b),.s(s));
10
11 initial
12 begin
13
14 a<=1'b0;
15 b<=1'b0;
16
17 #10
18 a<=1'b1;
19 b<=1'b0;
20
21 #10
22 a<=1'b0;
23 b<=1'b1;
24
25 #10
26 a<=1'b1;
27 b<=1'b1;
28
29 #10
30 $finish;
31
32 end
33 endmodule
24
```

(二) testbench.v :對 HDL 的電路進行仿真驗證

與前面的 and_gate 的 testbench 類似。唯一的不同
在第 9 行改成 or_gate

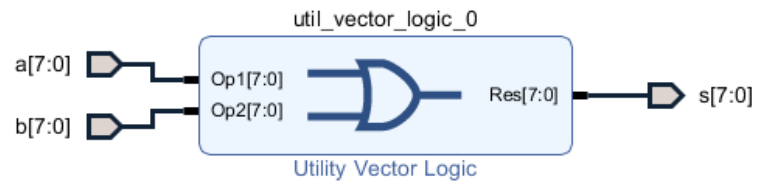
(三) FPGA : 電路圖



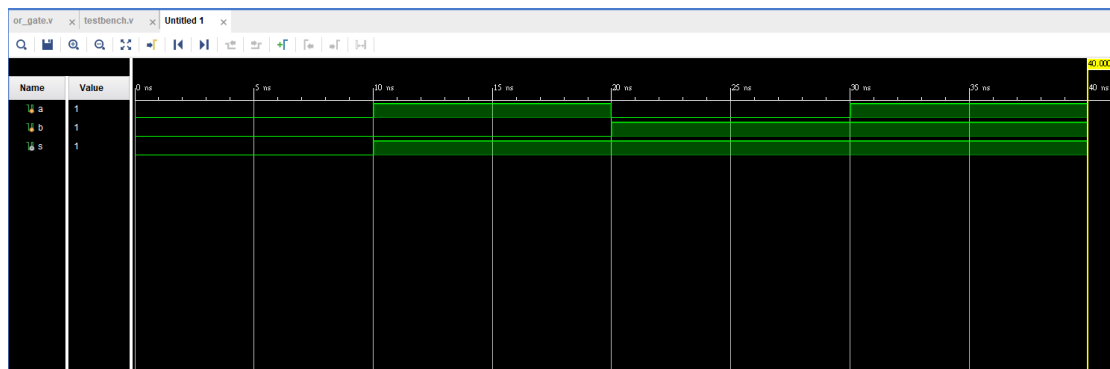
:or 運算邏輯閘，此為 2

輸入 1 輸出的邏輯閘

和 and gate 類似，不同之處在於 or gate 執行 or 運算



(四) 仿真驗證圖



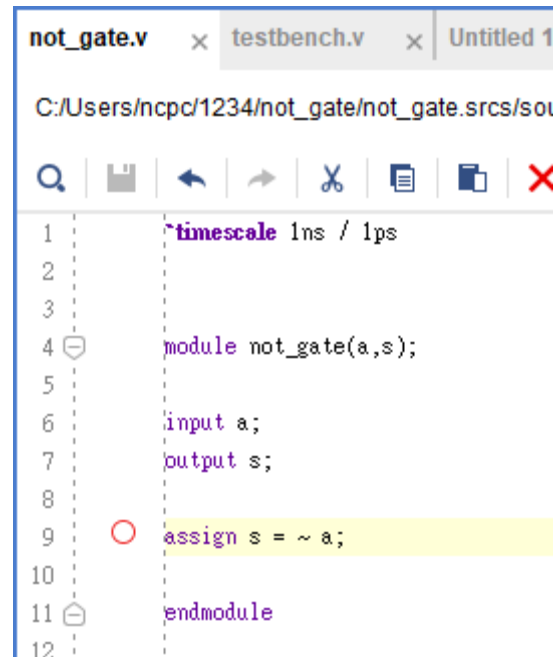
a、b 電壓根據 testbench 而變，s 亦根據 or_gate.v 而變
由此可知程式碼沒錯

三、not gate

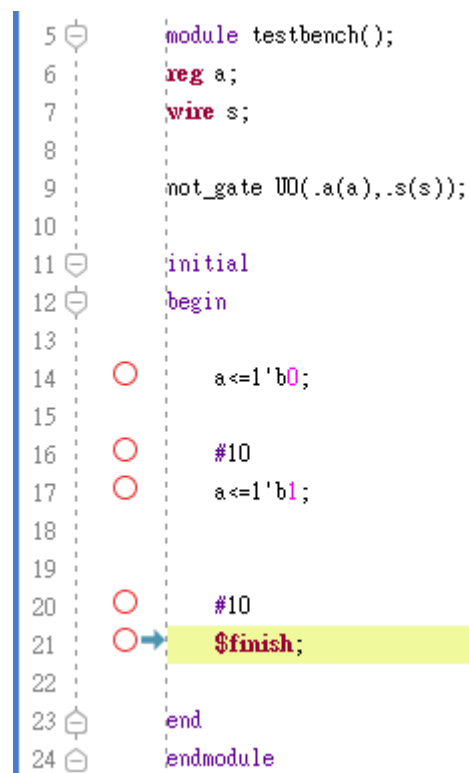
(一) not_gate.v :就是 not 邏輯閘運算的程式碼

not gate 僅有一個輸入和 and gate、or gate 不同，故程式碼中只需要有一個 input 變數

第 9 行: assign s = ~ a;
// a 執行 not 運算的結果指定給 s



```
not_gate.v x testbench.v x Untitled 1
C:/Users/ncpc/1234/not_gate/not_gate.srscs/sou
1 timescale 1ns / 1ps
2
3
4 module not_gate(a,s);
5
6 input a;
7 output s;
8
9 assign s = ~ a;
10
11 endmodule
12
```



```
1 module testbench();
2 reg a;
3 wire s;
4
5 not_gate U0(.a(a),.s(s));
6
7 initial
8 begin
9
10 a<=1'b0;
11
12 #10
13 a<=1'b1;
14
15 #10
16 $finish;
17
18 end
19 endmodule
20
```

(二) testbench.v :對 HDL 的電路進行仿真驗證

因為只有一個輸入所以第 14 行到第 17 行與 and_gate 不同

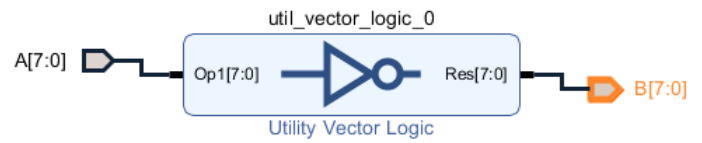
(三) FPGA : 電路圖



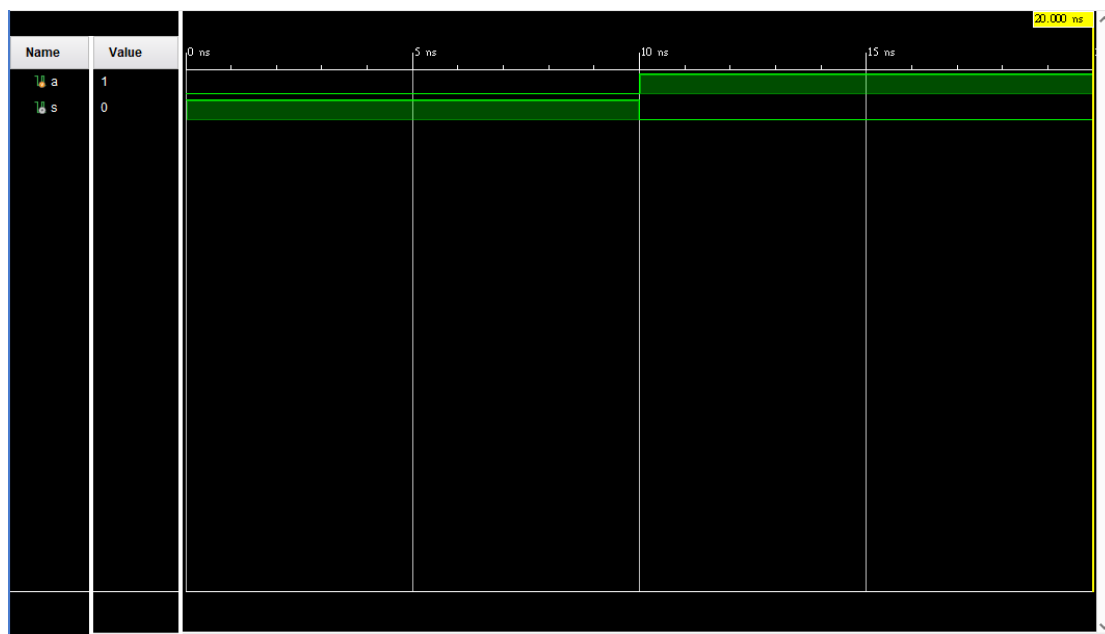
:not 運算邏輯閘，此為

1 輸入 1 輸出的邏輯閘

執行 not 運算



(四) 仿真驗證圖



a、b 電壓根據 testbench 而變，s 亦根據 not_gate.v 而變
由此可知程式碼沒錯

實驗心得

Verilog 文法與 C 相似都需要以分號(;)作為結尾。

C 語言中的大括弧({})在 Verilog 以 (begin、end) 或 (module、endmodule) 等，視情況而變。

assign 在 C 語言中沒有。在 Verilog 中：要求指定的訊號線要做什麼運算。