今天利用 3 種方式實驗數位電路驗證

第一個實驗是使用 minterm,maxterm 的方式來找回原本的邏輯電路

歸納 minterm:

想辦法將各個 input(包含 invert 反轉)and 運算後產出 1

各個 minterm 合併利用 or

歸納 maxterm:

想辦法將各個 input(包含 invert 反轉)or 運算後產出 0

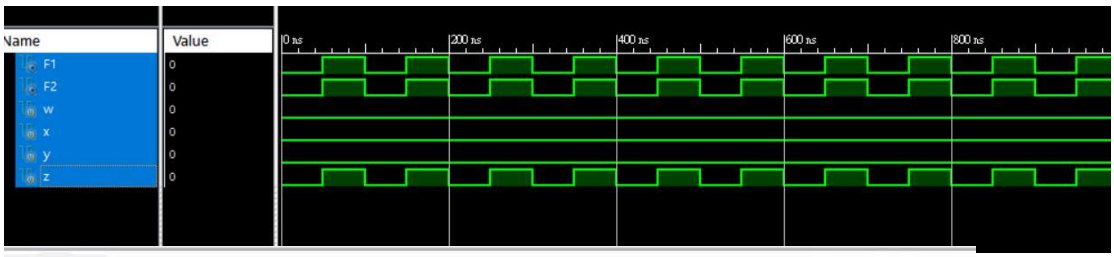
各個 maxterm 合併利用 and

```
`timescale 1ns / 1ps
//////////////////////////////////////////////////////////////////////////////////
// Company:
// Engineer:
//
// Create Date:    22:36:19 09/29/2019
// Design Name:
// Module Name:    veriloga
// Project Name:
// Target Devices:
// Tool versions:
// Description:
//
// Dependencies:
//
// Revision:
// Revision 0.01 - File Created
// Additional Comments:
//
//////////////////////////////////////////////////////////////////////////////////
module veriloga(output F1 , output F2 ,input w , input x ,input y ,input z);
   assign F1 = ((~w)&(~x)&(~y)&(z))|((~w)&(~x)&(y)&(~z))|((~w)&(~x)&(y)&(z))|((~w)&(x)&(y)&(~z))|((w)&(~x)&(y)&(z))
   assign F2 = ((w)|(x)|(y)|(z))&((w)|(~x)|(y)|(z))&((w)|(~x)|(y)|(~z))&((w)|(~x)|(~y)|(~z))&((~w)|(x)|(y)|(z))&((~

endmodule
```

以 Verilog 實現

在使用 testbench 時遇到了一些狀況:




```
46
47      initial begin
48          // Initialize Inputs
49          /*
50          for (w = 0 ;w <= 1 ;w=w+1)begin
51              for (x = 0 ;x<= 1; x=x+1)begin
52                  for(y = 0; y <= 1 ;y=y+1)begin
53                      for(z = 0 ;z <= 1 ;z= z+1)begin
54                          #50;
55                      end
56                  end
57              end
58          end
59          */
60
```

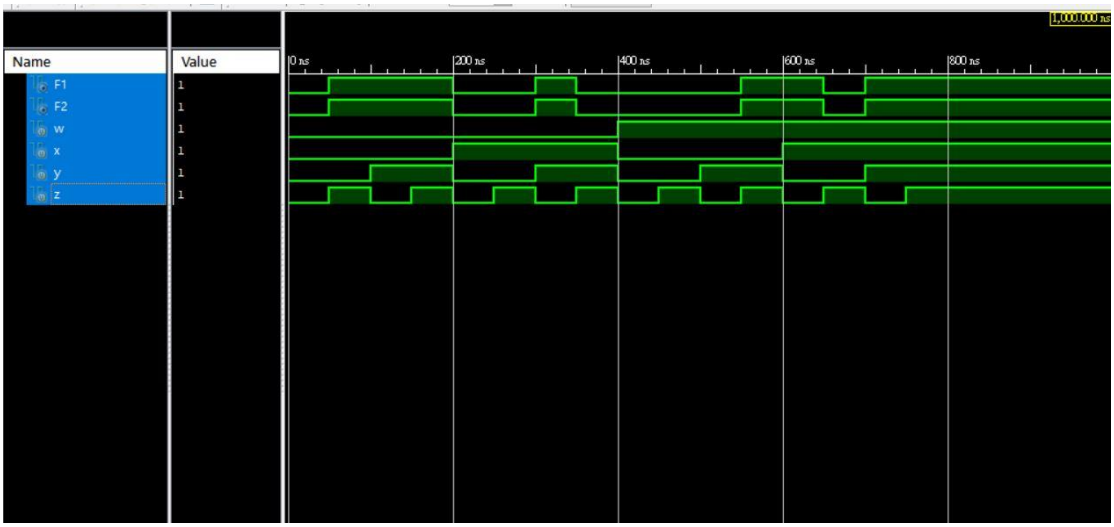
本來想要使 for 迴圈來模擬各個 Input 狀況
但模擬出來的結果只有迴圈內層的 Z 有震盪的情況
詢問教授的結果
應該是 for 迴圈內不行同時為 1.for 迴圈操縱的變數 2.input 的變數

```
for (u = 0 ;u <= 1 ;u=u+1)begin
   for (i = 0 ;i<= 1; i=i+1)begin
      for(o = 0; o <= 1 ;o=o+1)begin
         for(p = 0 ;p <= 1 ;p= p+1)begin
            w=u;
            x=i;
            y=o;
            z=p;
            #50;
         end
      end
   end
end
/*
```

指定其他的變數後狀況還是一樣，無法得到理想上的輸出
再次詢問教授後，教授有提到硬體描述語言的同步特性與 C 語言一條一條執行上是有差異，以後會在教 testbench 的其他用法
所以以後的狀況還是乖乖把 input 的資料寫入

```
60
61          w=0;
62          x=0;
63          y=0;
64          z=0;
65          #50;
66
67
68          w=0;
69          x=0;
70          y=0;
71          z=1;
72          #50;
73
74
75          w=0;
76          x=0;
77          y=1;
```

乖乖的打出 testbench 測試 Input



正常的輸出 F1=F2

第二個實驗 使用的是 structural level 的模擬

```
// Module Name:    veriloga
// Project Name:
// Target Devices:
// Tool versions:
// Description:
//
// Dependencies:
//
// Revision:
// Revision 0.01 - File Created
// Additional Comments:
//
//////////////////////////////////////////////////////////////////////////////////
module veriloga(output O1,output O2,output O3,output O4,input x,input y ,input z);
   wire O1_A;
   nor(O1_A,x,y);
   nor(O1,O1_A,z);

   wire O1_C;
   nor(O1_C,y,x);
   nor(O2,O1_C,z);

   wire O2_A;
   nor(O2_A,x,y);
   nor(O3,O2_A,z);

   wire O2_C;
   nor(O2_C,y,z);
   nor(O4,O2_C,x);

endmodule
```

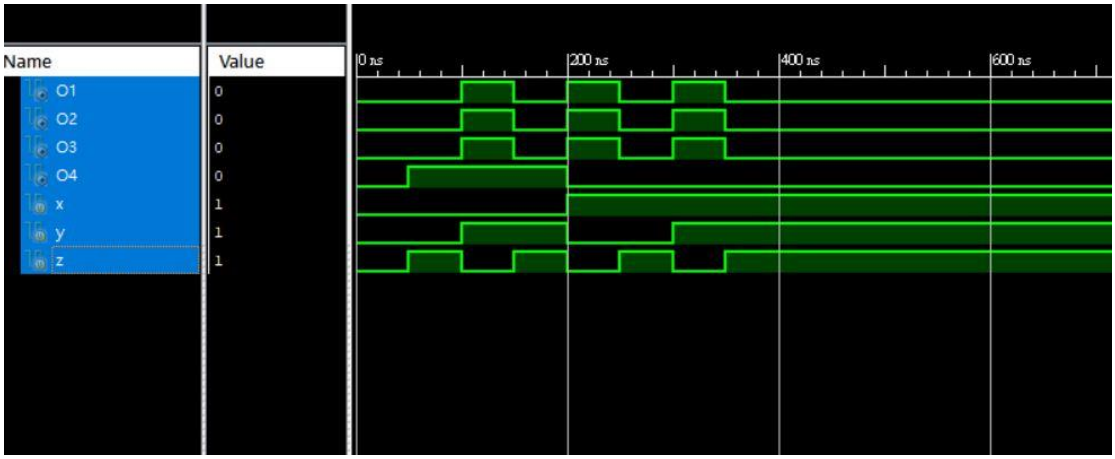
本來想利用 equivalence 方式檢查是否等式左邊等於等式右邊

但我不知道怎麼用 structural level 的方式表示，所以使用 nor 後在使用 not

詢問助教後可以使用 xnor 以檢查左式是否相等右式

```
22  //
23  //////////////////////////////////////////////////////////////////////////////////
24
25  module testbench;
26
27      // Inputs
28      reg x;
29      reg y;
30      reg z;
31
32      // Outputs
33      wire O1;
34      wire O2;
35      wire O3;
36      wire O4;
37
38      // Instantiate the Unit Under Test (UUT)
39      verilogx uut (
40          .O1(O1),
41          .O2(O2),
42          .O3(O3),
43          .O4(O4),
44          .x(x),
45          .y(y),
46          .z(z)
47      );
48
49      initial begin
50          // Initialize Inputs
51          x = 0;
52          y = 0;
53          z = 0;
54
55          #50;
56
57          x = 0;
58          y = 0;
59          z = 1;
60
61          #50;
62
63          x = 0;
64          y = 1;
65          z = 0;
66
67          #50;
68
69          x = 0;
70          y = 1;
71          z = 1;
72
73          #50;
74
75          x = 1;
76          y = 0;
77          z = 0;
78
79          #50;
80
81          x = 1;
82          y = 0;
83          z = 1;
84
85          #50;
```

乖乖打的測試資料



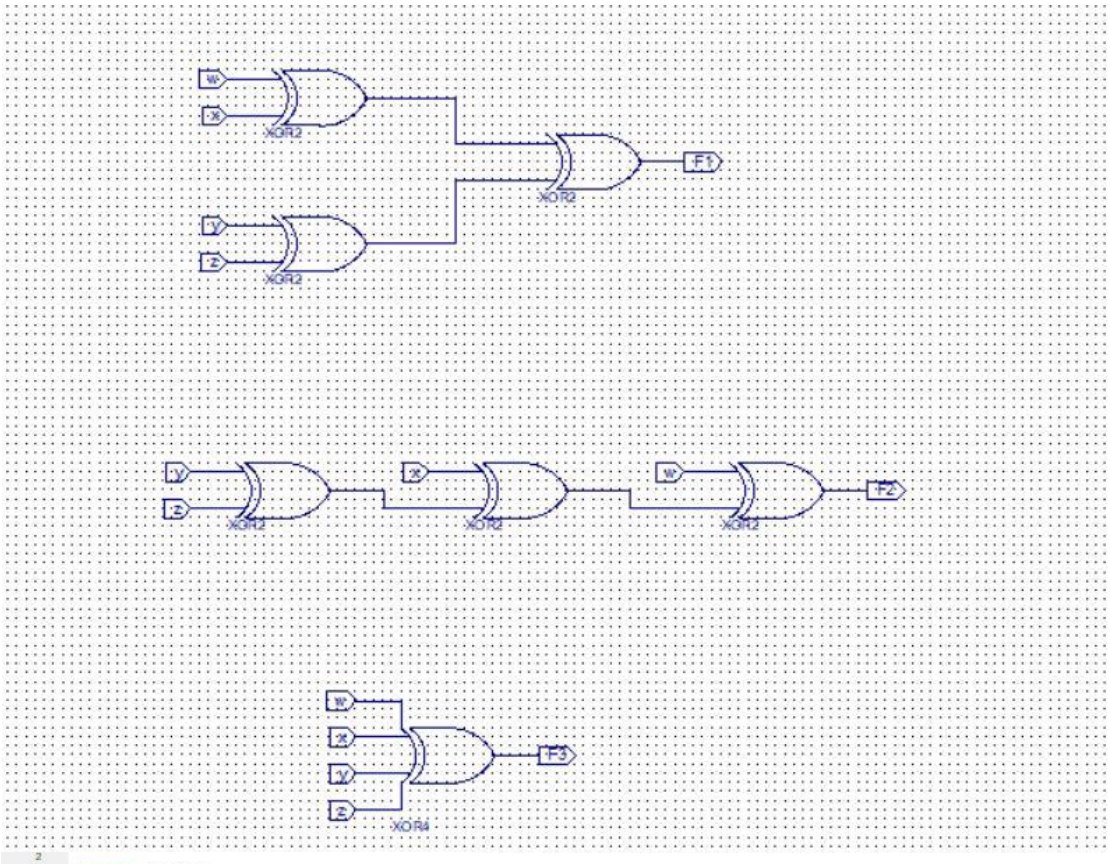
證實 O1=O2

O3≠O4

第三個實驗利用 schematic 方式

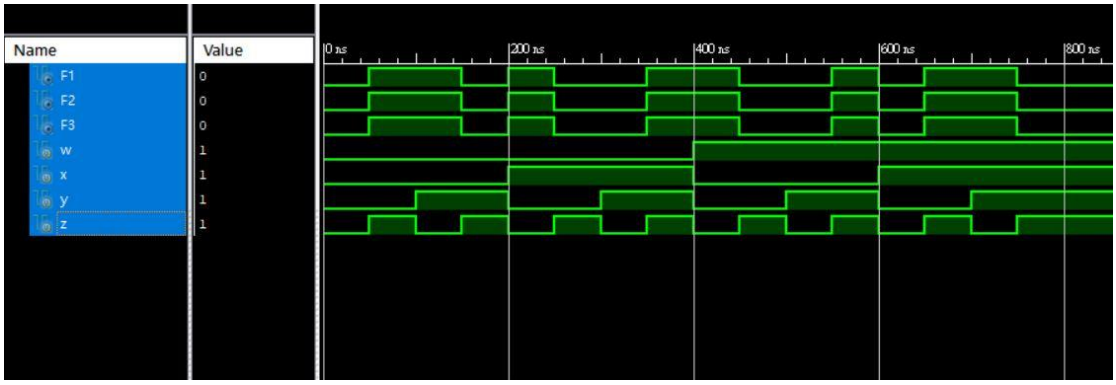
證明 XOR 的結合率

利用多種 XOR 模擬組合




```
2
3   `timescale 1ns / 1ps
4
5   module scha_scha_sch_tb();
6
7   // Inputs
8      reg w;
9      reg x;
10     reg y;
11     reg z;
12
13  // Output
14     wire F1;
15     wire F2;
16     wire F3;
17
18  // Bidirs
19
20  // Instantiate the UUT
21     scha UUT (
22        .w(w),
23        .x(x),
24        .y(y),
25        .z(z),
26        .F1(F1),
27        .F2(F2),
28        .F3(F3)
29     );
30  // Initialize Inputs
31        initial begin
32        w=0;
33        x=0;
34        y=0;
35        z=0;
36        #50;
37
38
39        w=0;
40        x=0;
41        y=0;
42        z=1;
43        #50;
44
45
46        w=0;
47        x=0;
```

測試資料



證實 F1=F2=F3