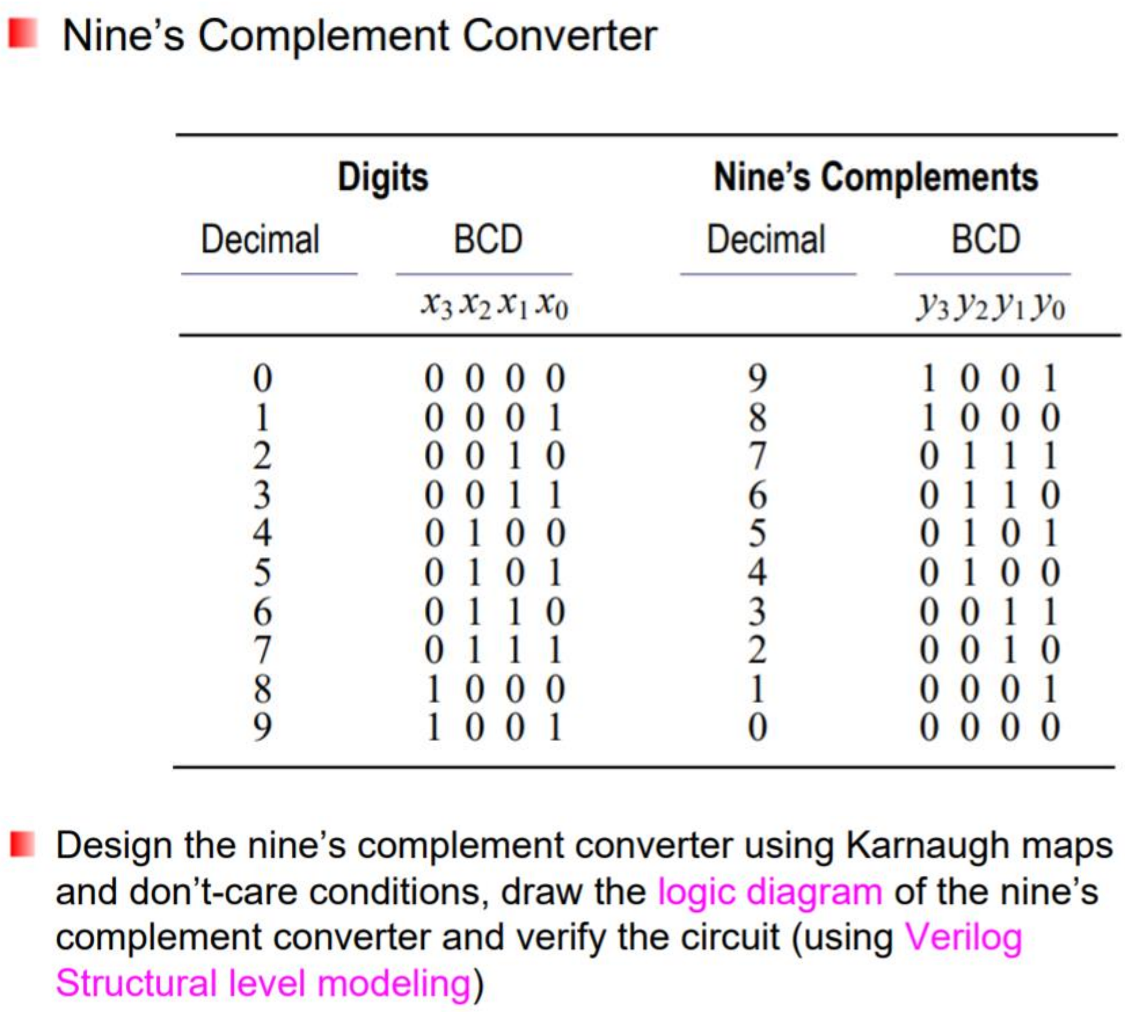
實驗主題(Lab06)

B083022053

黃啟桓

Exercise 1:

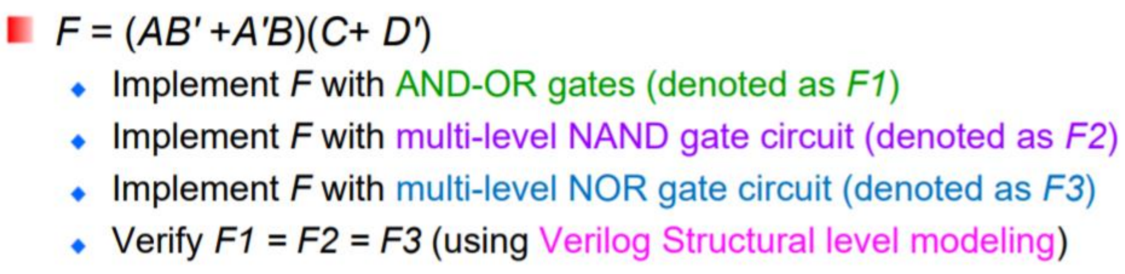


- Nine’s Complement Converter

Digits		Nine’s Complements	
Decimal	BCD	Decimal	BCD
	$x_3\,x_2\,x_1\,x_0$		$y_3\,y_2\,y_1\,y_0$
0	0 0 0 0	9	1 0 0 1
1	0 0 0 1	8	1 0 0 0
2	0 0 1 0	7	0 1 1 1
3	0 0 1 1	6	0 1 1 0
4	0 1 0 0	5	0 1 0 1
5	0 1 0 1	4	0 1 0 0
6	0 1 1 0	3	0 0 1 1
7	0 1 1 1	2	0 0 1 0
8	1 0 0 0	1	0 0 0 1
9	1 0 0 1	0	0 0 0 0

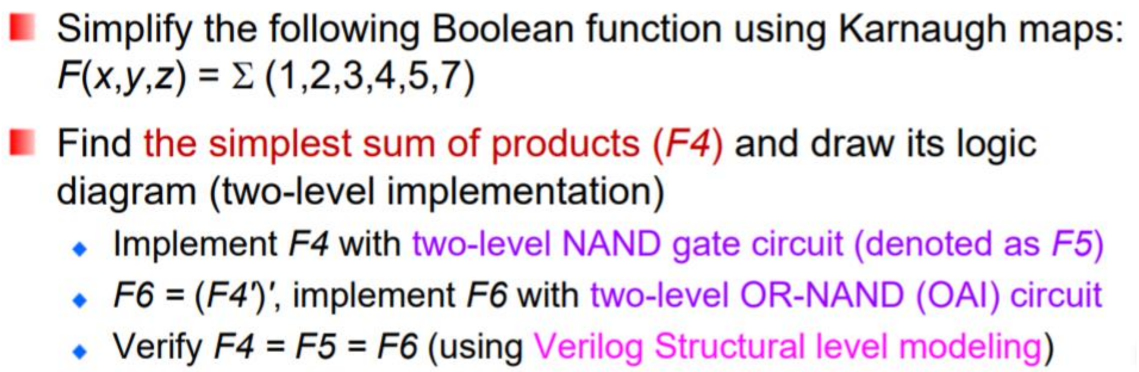
- Design the nine’s complement converter using Karnaugh maps and don’t-care conditions, draw the logic diagram of the nine’s complement converter and verify the circuit (using Verilog Structural level modeling)

Exercise 2:



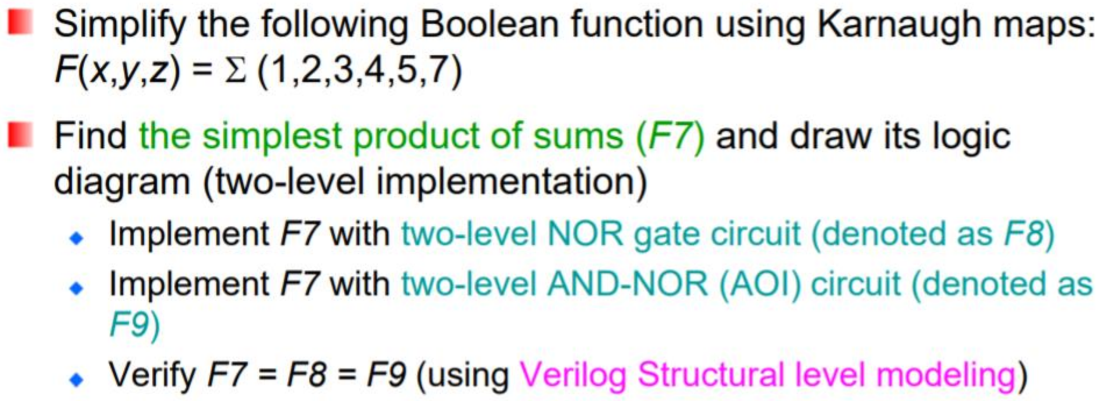
- $F = (AB' + A'B)(C + D')$
  - Implement F with AND-OR gates (denoted as $F1$)
  - Implement F with multi-level NAND gate circuit (denoted as $F2$)
  - Implement F with multi-level NOR gate circuit (denoted as $F3$)
  - Verify $F1 = F2 = F3$ (using Verilog Structural level modeling)

Exercise 3:



- Simplify the following Boolean function using Karnaugh maps: $F(x,y,z) = \Sigma\,(1,2,3,4,5,7)$
- Find the simplest sum of products (*F4*) and draw its logic diagram (two-level implementation)
  - Implement *F4* with two-level NAND gate circuit (denoted as *F5*)
  - *F6* = (*F4'*)', implement *F6* with two-level OR-NAND (OAI) circuit
  - Verify *F4* = *F5* = *F6* (using Verilog Structural level modeling)

Exercise 4:

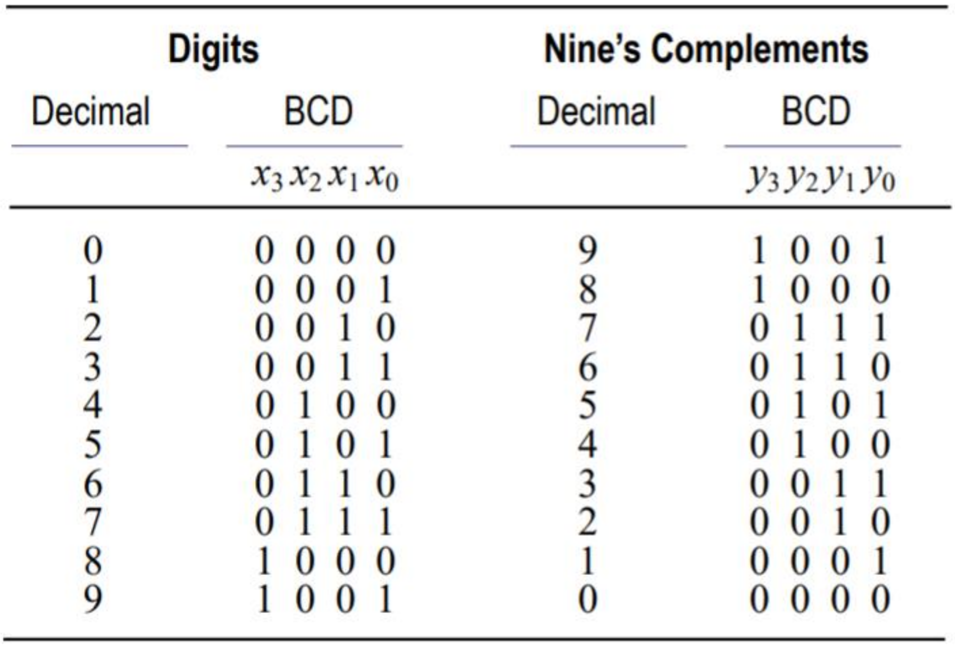


- Simplify the following Boolean function using Karnaugh maps: $F(x,y,z) = \Sigma\,(1,2,3,4,5,7)$
- Find the simplest product of sums (*F7*) and draw its logic diagram (two-level implementation)
  - Implement *F7* with two-level NOR gate circuit (denoted as *F8*)
  - Implement *F7* with two-level AND-NOR (AOI) circuit (denoted as *F9*)
  - Verify *F7* = *F8* = *F9* (using Verilog Structural level modeling)

實驗內容

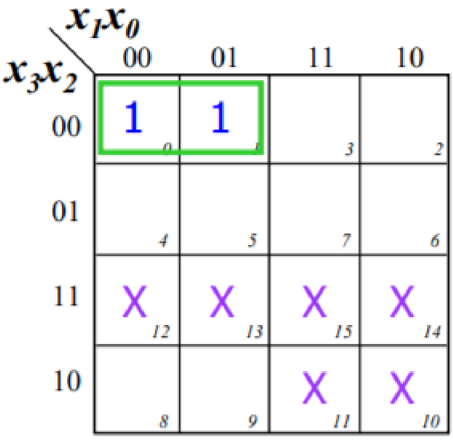
一、Exercise 1:

- Nine’s Complement Converter



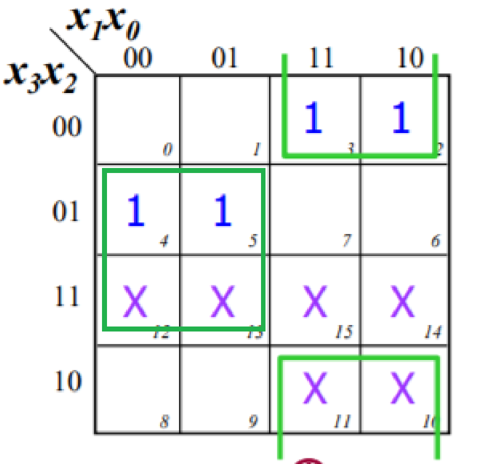
Digits		Nine’s Complements	
Decimal	BCD	Decimal	BCD
	$x_3 x_2 x_1 x_0$		$y_3 y_2 y_1 y_0$
0	0 0 0 0	9	1 0 0 1
1	0 0 0 1	8	1 0 0 0
2	0 0 1 0	7	0 1 1 1
3	0 0 1 1	6	0 1 1 0
4	0 1 0 0	5	0 1 0 1
5	0 1 0 1	4	0 1 0 0
6	0 1 1 0	3	0 0 1 1
7	0 1 1 1	2	0 0 1 0
8	1 0 0 0	1	0 0 0 1
9	1 0 0 1	0	0 0 0 0

- Design the nine’s complement converter using Karnaugh maps and don’t-care conditions, draw the logic diagram of the nine’s complement converter and verify the circuit (using Verilog Structural level modeling)



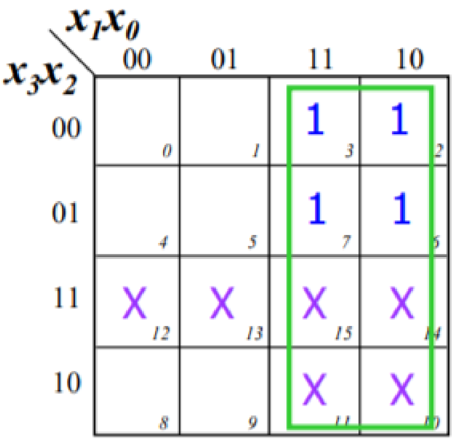
$x_3x_2 \backslash x_1x_0$	00	01	11	10
00	1	1		
01				
11	X	X	X	X
10			X	X

$y_3 = x_3' x_2' x_1'$



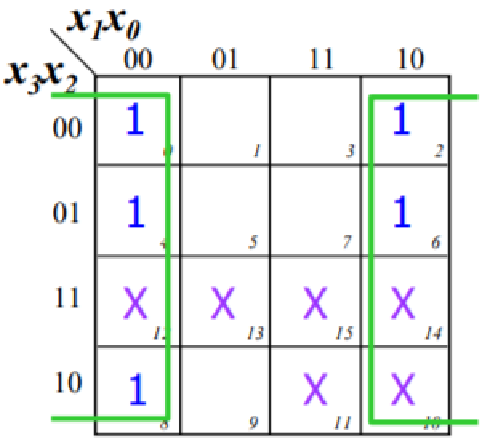
$x_3x_2 \backslash x_1x_0$	00	01	11	10
00			1	1
01	1	1		
11	X	X	X	X
10			X	X

$y_2 = x_2 \oplus x_1$



$x_3x_2 \backslash x_1x_0$	00	01	11	10
00			1	1
01			1	1
11	X	X	X	X
10			X	X

$y_1 = x_1$

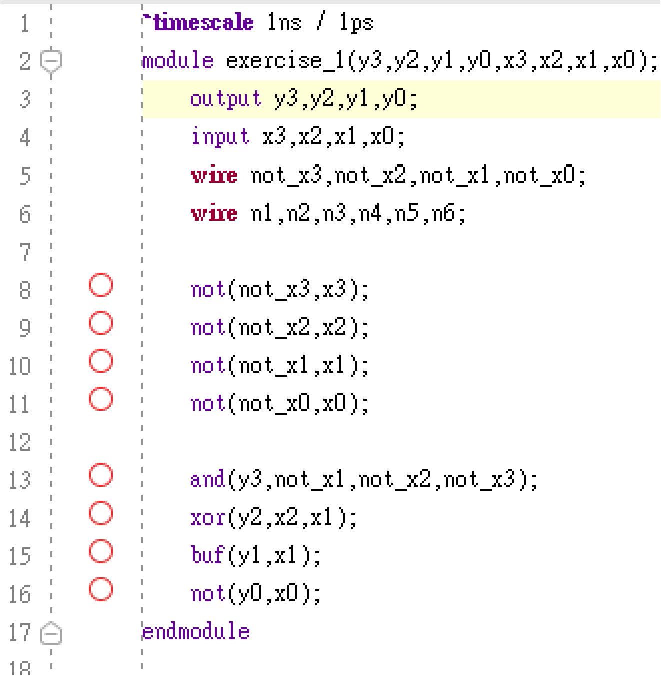


$x_3x_2 \backslash x_1x_0$	00	01	11	10
00	1			1
01	1			1
11	X	X	X	X
10	1		X	X

$y_0 = x_0'$

因為不可能輸入 m_{12}~m_{16}，所以 m_{12}~m_{16} 為 don’t care 項，don’t care 項可以視為 0 亦可視為 1。

Structural level modeling 程式如右:



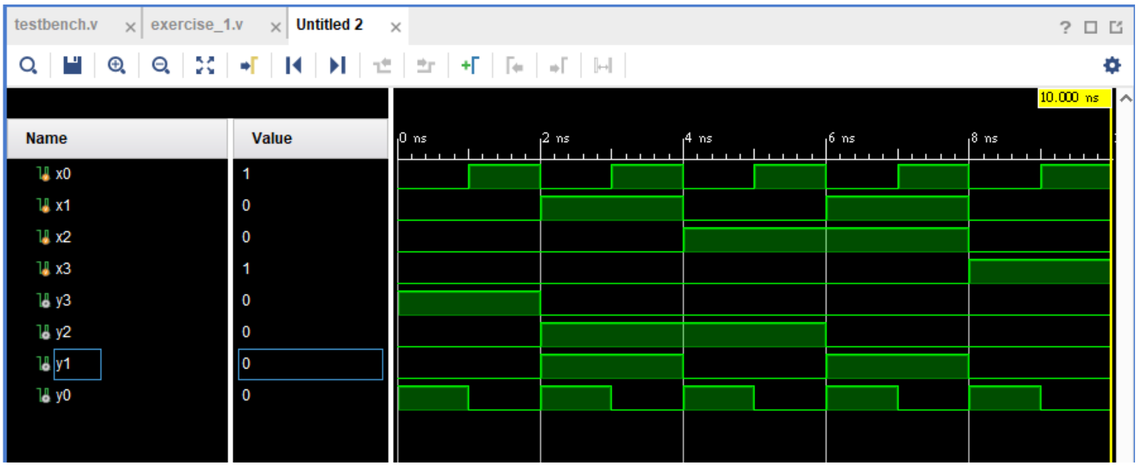
```
1   `timescale 1ns / 1ps
2   module exercise_1(y3,y2,y1,y0,x3,x2,x1,x0);
3       output y3,y2,y1,y0;
4       input x3,x2,x1,x0;
5       wire not_x3,not_x2,not_x1,not_x0;
6       wire n1,n2,n3,n4,n5,n6;
7
8       not(not_x3,x3);
9       not(not_x2,x2);
10      not(not_x1,x1);
11      not(not_x0,x0);
12
13      and(y3,not_x1,not_x2,not_x3);
14      xor(y2,x2,x1);
15      buf(y1,x1);
16      not(y0,x0);
17  endmodule
18
```

第 13~16 行:

y3=x1’x2’x3’

y2=x1’x2+x1x2’=x1⊕x2

y1=x1
y0=x0’



Name	Value
x0	1
x1	0
x2	0
x3	1
y3	0
y2	0
y1	0
y0	0

電壓圖完全符合 BCD 碼 9 補數(上到下: 輸入:x0,x1,x2,x3 輸出:y3,y2,y1,y0)

二、Exercise 2:

- $F = (AB' + A'B)(C + D')$
  - Implement *F* with AND-OR gates (denoted as *F1*)
  - Implement *F* with multi-level NAND gate circuit (denoted as *F2*)
  - Implement *F* with multi-level NOR gate circuit (denoted as *F3*)
  - Verify *F1* = *F2* = *F3* (using Verilog Structural level modeling)

程式如右:

(1)

$F=(AB'+A'B)(C+D')$

$=AB'C+AB'D'+A'BC+A'BD'$

$=F1$

(2)

$F1=AB'C+AB'D'+A'BC+A'BD'$

$=(AB'C)''+(AB'D')''+(A'BC)''+(A'BD')''$

$=(AB'C)'(AB'D')'(A'BC)'(A'BD')')'$

$=F2$

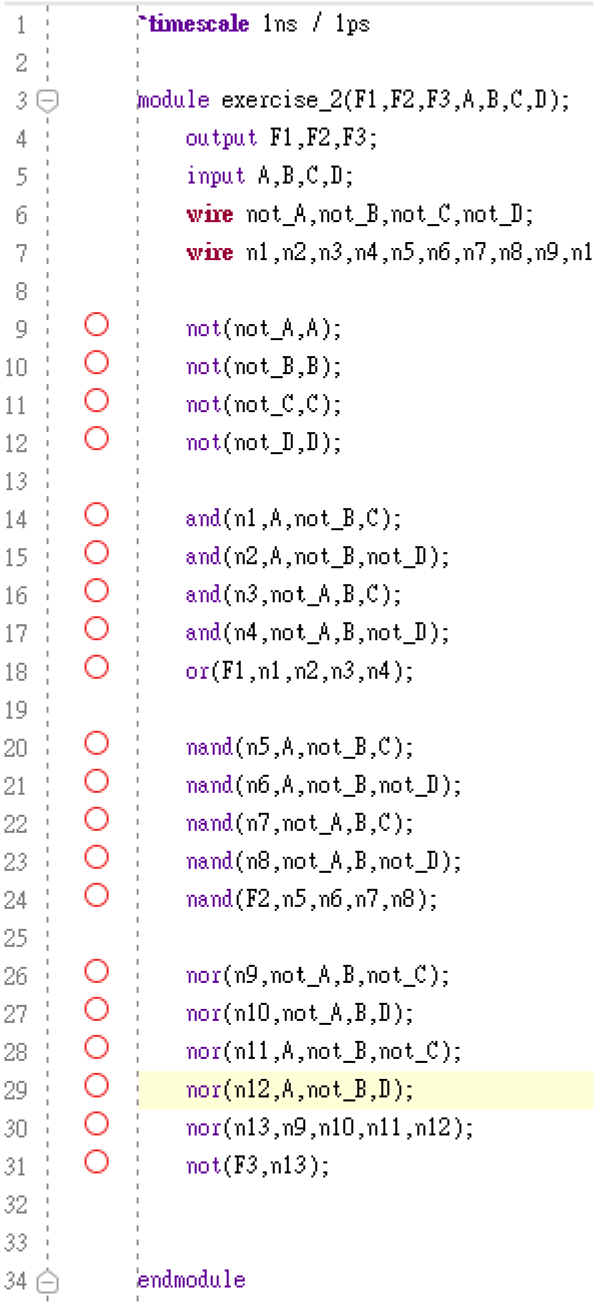
(3)

$F1=AB'C+AB'D'+A'BC+A'BD'$

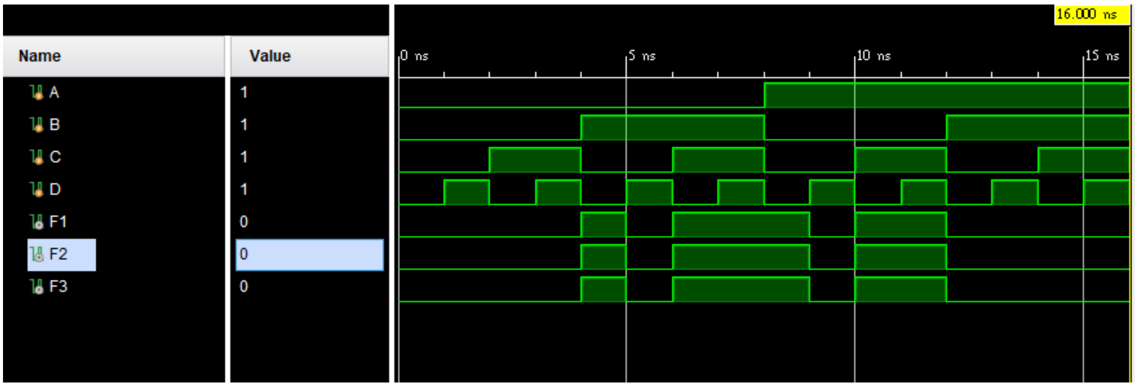
$=(AB'C)''+(AB'D')''+(A'BC)''+(A'BD')''$

$=((A'+B+C')' + (A'+B+D)' + (A+B'+C') + (A+B'D)')'$

$=F4$



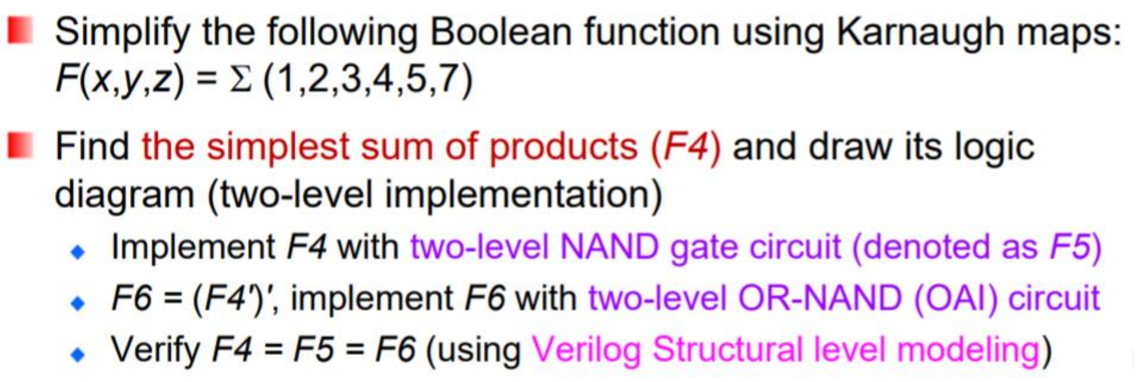
```
`timescale 1ns / 1ps

module exercise_2(F1,F2,F3,A,B,C,D);
    output F1,F2,F3;
    input A,B,C,D;
    wire not_A,not_B,not_C,not_D;
    wire n1,n2,n3,n4,n5,n6,n7,n8,n9,n1

    not(not_A,A);
    not(not_B,B);
    not(not_C,C);
    not(not_D,D);

    and(n1,A,not_B,C);
    and(n2,A,not_B,not_D);
    and(n3,not_A,B,C);
    and(n4,not_A,B,not_D);
    or(F1,n1,n2,n3,n4);

    nand(n5,A,not_B,C);
    nand(n6,A,not_B,not_D);
    nand(n7,not_A,B,C);
    nand(n8,not_A,B,not_D);
    nand(F2,n5,n6,n7,n8);

    nor(n9,not_A,B,not_C);
    nor(n10,not_A,B,D);
    nor(n11,A,not_B,not_C);
    nor(n12,A,not_B,D);
    nor(n13,n9,n10,n11,n12);
    not(F3,n13);


endmodule
```



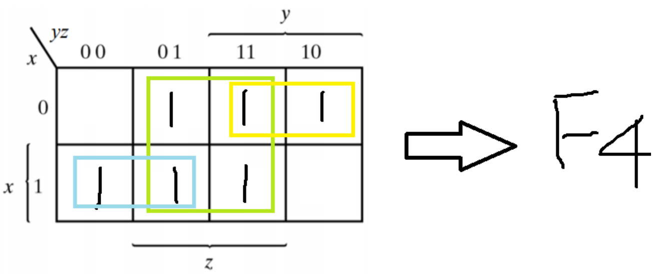
Name	Value
A	1
B	1
C	1
D	1
F1	0
F2	0
F3	0

由電壓圖 F1=F2=F3 (上到下: 輸入:A,B,C,D 輸出:F1,F2,F3)

三、Exercise　3:



- Simplify the following Boolean function using Karnaugh maps: *F*(*x*,*y*,*z*) = Σ (1,2,3,4,5,7)
- Find the simplest sum of products (*F4*) and draw its logic diagram (two-level implementation)
  - Implement *F4* with two-level NAND gate circuit (denoted as *F5*)
  - *F6* = (*F4*’)’, implement *F6* with two-level OR-NAND (OAI) circuit
  - Verify *F4* = *F5* = *F6* (using Verilog Structural level modeling)



程式碼如右:

(1)

The simplest sum of products 選越多相鄰越好

F4 = x’y+xy’+z

(2)

F4 = x’y+xy’+z = (x’y)”+(xy’)”+(z)”

= ((x’y)’ (xy’)’ (z)’)’

=F5

(3) 2 種方法

(a)　(F4’)’ =　(x’y’z’+xyz’)’

F4 = (x’y’z’+xyz’)’ = (x+y+z)(x’+y’+z)”

=((x+y+z)(x’+y’+z)’)’

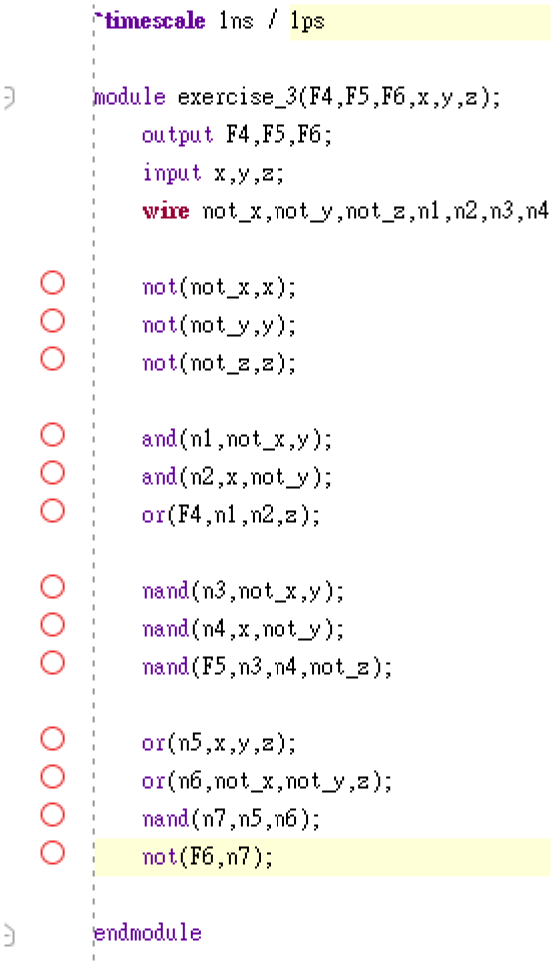
=F6

(b) 已知(AB)’ = (A’+B’) :

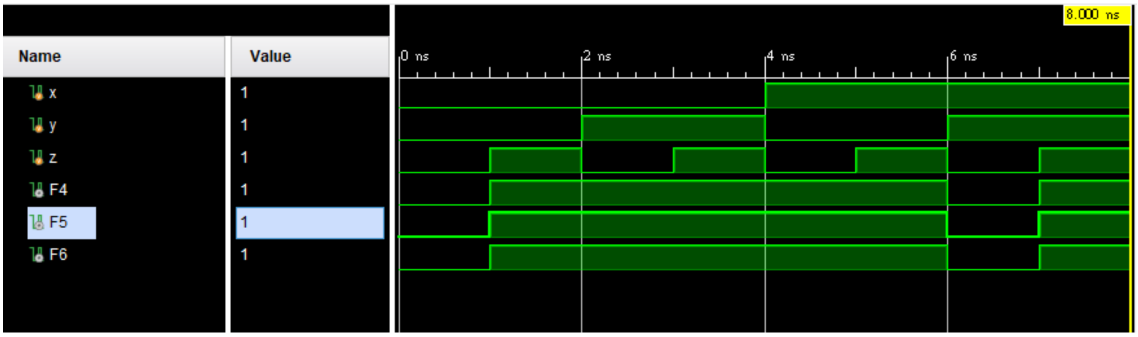
所以由 F5 可以得:

((x’y)’ (xy’)’ (z)’)’ = ((x+y’) (x’+y) (z)’)’

=F6



```
`timescale 1ns / 1ps

module exercise_3(F4,F5,F6,x,y,z);
    output F4,F5,F6;
    input x,y,z;
    wire not_x,not_y,not_z,n1,n2,n3,n4

    not(not_x,x);
    not(not_y,y);
    not(not_z,z);

    and(n1,not_x,y);
    and(n2,x,not_y);
    or(F4,n1,n2,z);

    nand(n3,not_x,y);
    nand(n4,x,not_y);
    nand(F5,n3,n4,not_z);

    or(n5,x,y,z);
    or(n6,not_x,not_y,z);
    nand(n7,n5,n6);
    not(F6,n7);

endmodule
```

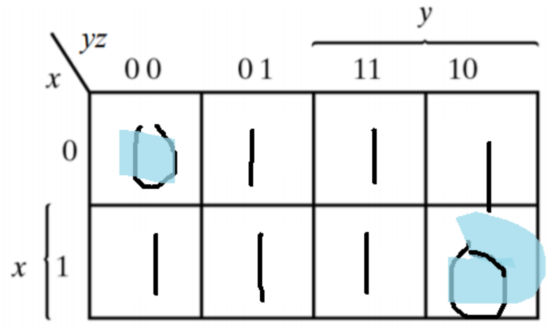


Name	Value
x	1
y	1
z	1
F4	1
F5	1
F6	1

由電壓圖 F4=F5=F6(上到下: 輸入:x,y,z 輸出:F4,F5,F6)

四、Exercise 4:

- Simplify the following Boolean function using Karnaugh maps: *F*(*x*,*y*,*z*) = Σ (1,2,3,4,5,7)
- Find the simplest product of sums (*F7*) and draw its logic diagram (two-level implementation)
  - Implement *F7* with two-level NOR gate circuit (denoted as *F8*)
  - Implement *F7* with two-level AND-NOR (AOI) circuit (denoted as *F9*)
  - Verify *F7* = *F8* = *F9* (using Verilog Structural level modeling)



x \ yz	00	01	11	10
0	0	1	1	1
1	1	1	1	0

程式碼如右:

(1) F7 ’ = (x’y’z’+xyz’)

F7 = (x’y’z’+xyz’)’ = (x+y+z)(x’+y’+z)

(2) F7= (x+y+z)(x’+y’+z)

= ((x+y+z)’ + (x’+y’+z)’)’

=F8

(3) 2 種方法

(a) F7 = F4 = x’y+xy’+z

x’y+xy’+z = ((x’y+xy’+z)’)’

=F9

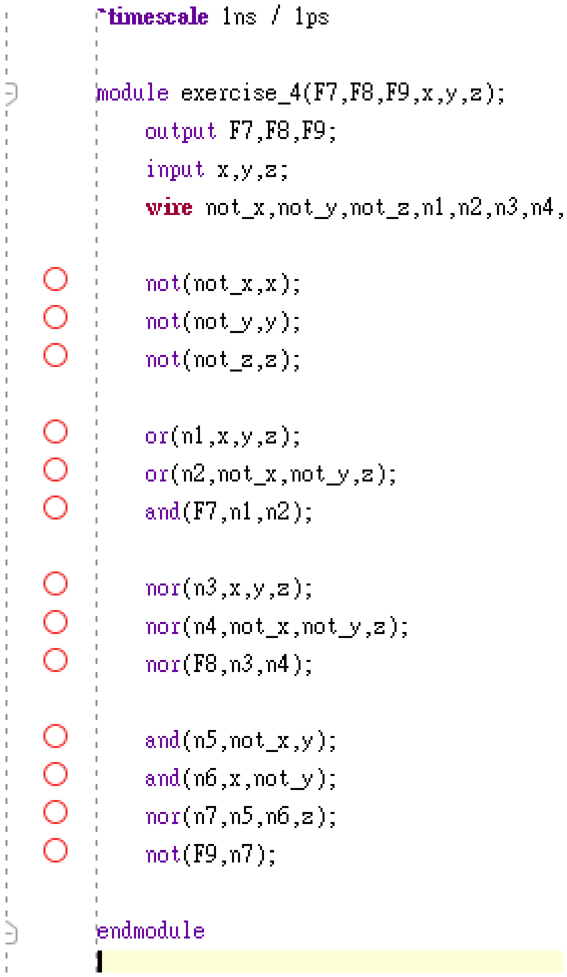
(b) 已知 (A+B)’ = (A’B’) :

所以由 F8 可以得:

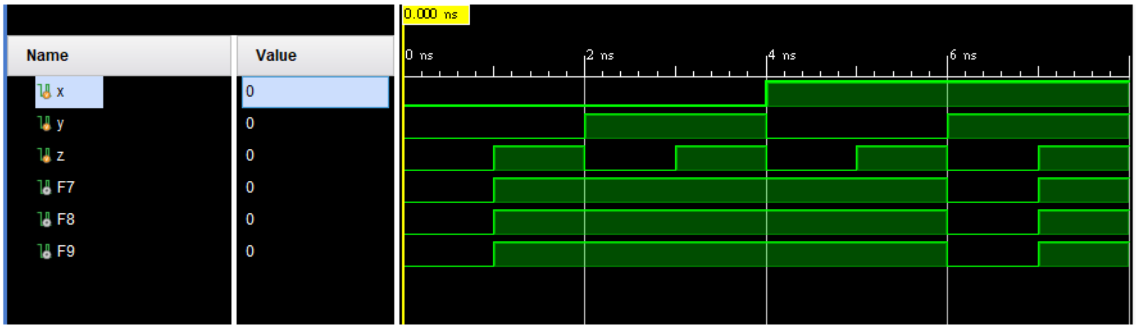
((x+y+z)’ + (x’+y’+z)’)’

= ((x’y’z’) + (xyz’))’

=F9



```
`timescale 1ns / 1ps

module exercise_4(F7,F8,F9,x,y,z);
    output F7,F8,F9;
    input x,y,z;
    wire not_x,not_y,not_z,n1,n2,n3,n4,

    not(not_x,x);
    not(not_y,y);
    not(not_z,z);

    or(n1,x,y,z);
    or(n2,not_x,not_y,z);
    and(F7,n1,n2);

    nor(n3,x,y,z);
    nor(n4,not_x,not_y,z);
    nor(F8,n3,n4);

    and(n5,not_x,y);
    and(n6,x,not_y);
    nor(n7,n5,n6,z);
    not(F9,n7);

endmodule
```



Name	Value
x	0
y	0
z	0
F7	0
F8	0
F9	0

由電壓圖 F7=F8=F9(上到下: 輸入:x,y,z 輸出:F7,F8,F9)

實驗心得

1. Structural level modeling 中邏輯閘使用函數表示
2. Structural level modeling 可以很直觀的轉換成 Schematic
3. Structural level modeling 需要宣告邏輯閘之間的線路
4. 所有的 modeling 都需要注意該邏輯閘是否有交換律或結合律
5. Exercise 1 的 don’t care conditions 很好用可以簡化電路
6. Exercise 2、3、4 電路的轉換用 Schematic 畫出來後比較簡單。

例如:exercise2:AND-OR gate 轉成 multi-level NAND gate、multi-level NOR gate…..。