第三章

閘階數之簡化

3-1 圖示法

★二變數卡諾圖

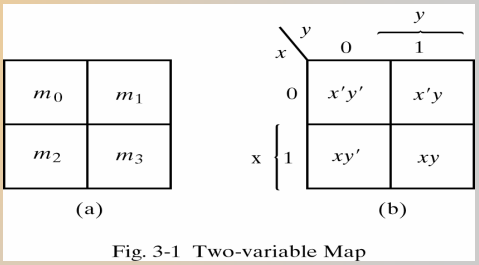


Fig. 3-1 Two-variable Map

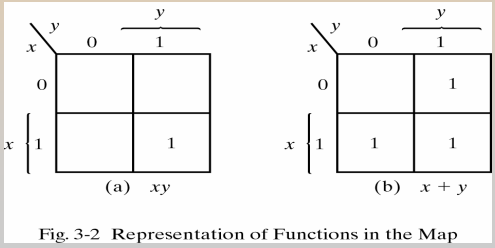


Fig. 3-2 Representation of Functions in the Map

圖示法

★ 三變數卡諾圖

m_0	m_1	m_3	m_2
m_4	m_5	m_7	m_6

(a)

x \ yz	00	01	11	10
0	$x'y'z'$	$x'y'z$	$x'yz$	$x'yz'$
1	$xy'z'$	$xy'z$	xyz	xyz'

(b)

Fig. 3-3 Three-variable Map

3-2 四變數卡諾圖

★ 四變數卡諾圖

m_0	m_1	m_3	m_2
m_4	m_5	m_7	m_6
m_{12}	m_{13}	m_{15}	m_{14}
m_8	m_9	m_{11}	m_{10}

(a)

wx \ yz	00	01	11	10
00	$w'x'y'z'$	$w'x'y'z$	$w'x'yz$	$w'x'yz'$
01	$w'xy'z'$	$w'xy'z$	$w'xyz$	$w'xyz'$
11	$wxy'z'$	$wxy'z$	$wxyz$	$wxyz'$
10	$wx'y'z'$	$wx'y'z$	$wx'yz$	$wx'yz'$

(b)

Fig. 3-8 Four-variable Map

範例

例題3-5 化簡布林函數

$$F(w,x,y,z)=\sum(0,1,2,4,5,6,8,9,12,13,14)$$

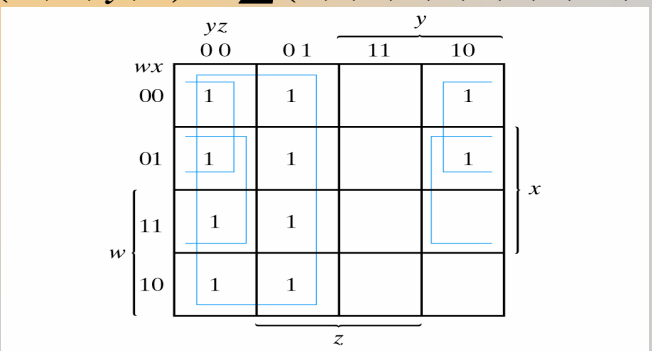
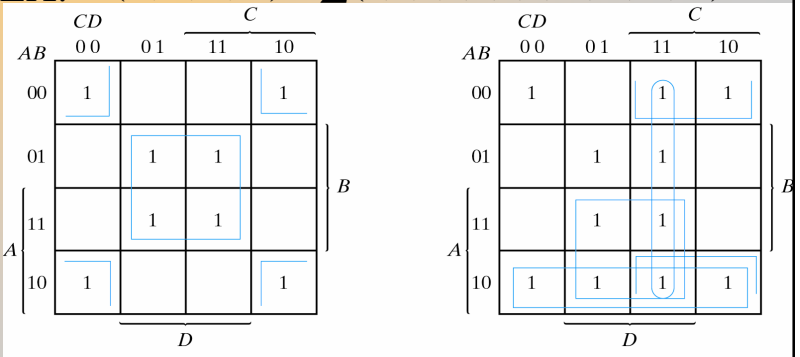


Fig. 3-9 Map for Example 3-5; $F(w, x, y, z)$
$= \Sigma\,(0, 1, 2, 4, 5, 6, 8, 9, 12, 13, 14) = y' + w'z' + xz'$

質含項

EX: $F(A,B,C,D)=\sum(0,2,3,5,7,8,9,10,11,13,15)$



(a) Essential prime implicants BD and B′D′

(b) Prime implicants CD, B′C AD, and AB′

Fig. 3-11 Simplification Using Prime Implicants

3-3 五變數卡諾圖

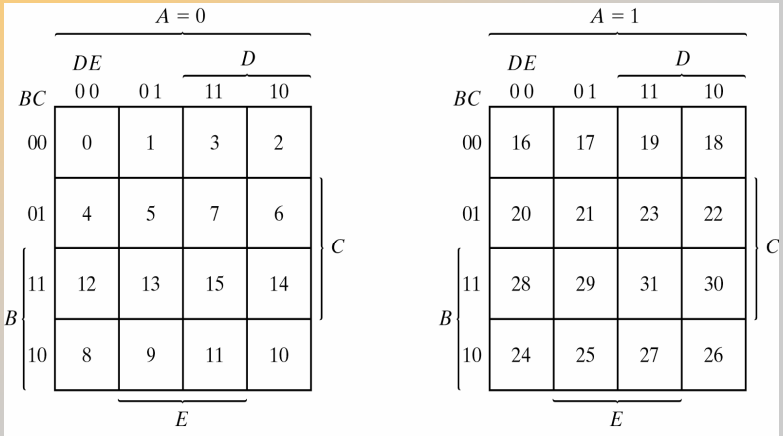


Fig. 3-12 Five-variable Map

相鄰方格數與每項中字元數的關係

表 3-1 相鄰方格數與每項中字元數的關係

	相鄰的方格數	在一個 n 變數卡諾圖中每項的字元數			
K	2^k	$n=2$	$n=3$	$n=4$	$n=5$
0	1	2	3	4	5
1	2	1	2	3	4
2	4	0	1	2	3
3	8		0	1	2
4	16			0	1
5	32				0

範例

EX: $F(A,B,C,D,E)=\sum(0,2,4,6,9,13,21,23,25,29,31)$

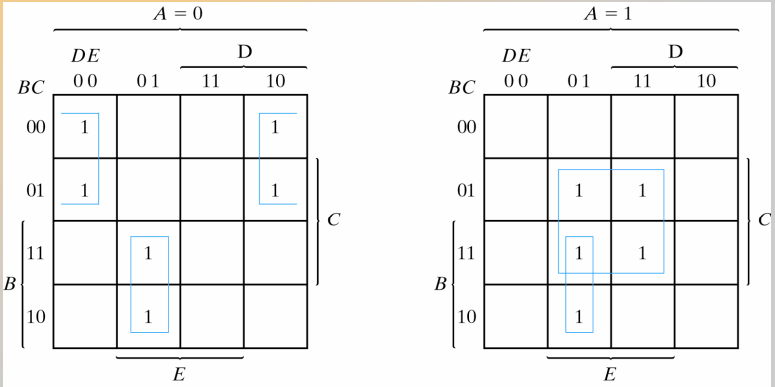


Fig. 3-13 Map for Example 3-7; $F = A'B'E' + BD'E + ACE$

3-4 和項積的化簡

★ 例題3-8 化簡布林函數爲(a)積項和的形式與(b)和項積的形式 $F(A,B,C,D)=\sum(0,1,2,5,8,9,10)$

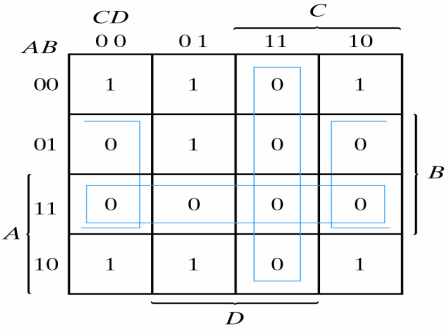
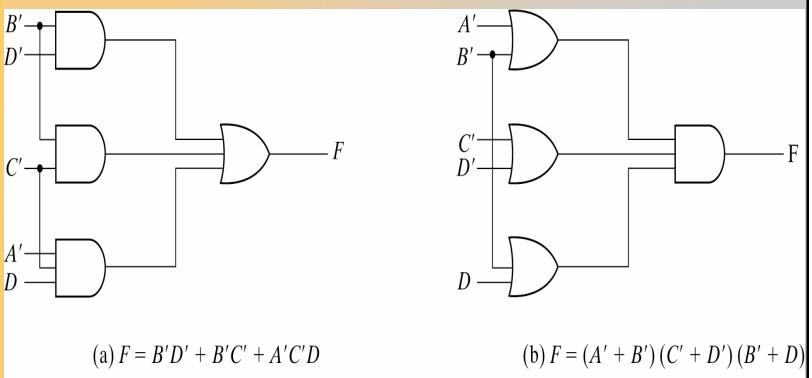


Fig. 3-14 Map for Example 3-8; $F(A,B,C,D) = \Sigma(0,1,2,5,8,9,10)$
$= B'D' + B'C' + A'C'D = (A'+B')(C'+D')(B'+D)$

例題3-8函數的閘電路



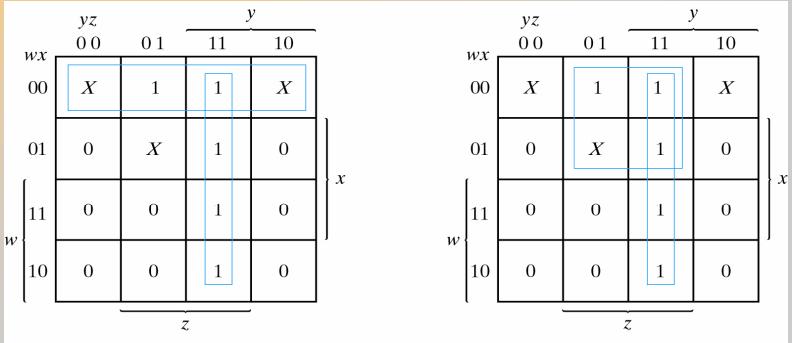
(a) $F = B'D' + B'C' + A'C'D$ (b) $F = (A' + B')(C' + D')(B' + D)$

Fig. 3-15 Gate Implementation of the Function of Example 3-8

3-5 不理會條件

例題3-9 化簡布林函數 $F(w,x,y,z) = \sum(1,3,7,11,15)$

不理會條件是 $d(w,x,y,z) = \sum(0,2,5)$



(a) $F = yz + w'x'$ (a) $F = yz + w'z$

Fig. 3-17 Example with don't-care Conditions

3-6 NAND及NOR閘的應用

✶NAND電路

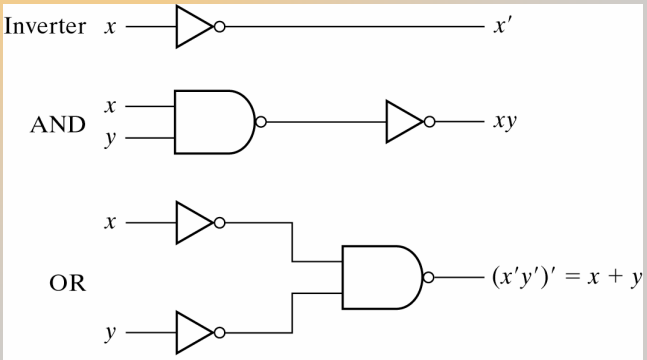
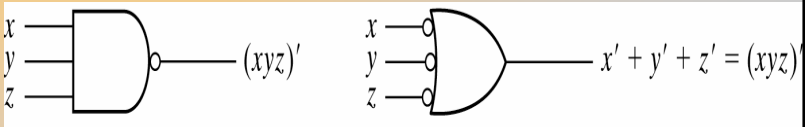


Fig. 3-18 Logic Operations with NAND Gates

NAND閘的圖示符號



(a) AND-invert (b) Invert-OR

Fig. 3-19 Two Graphic Symbols for NAND Gate

二階電路

例：$F = AB + CD$

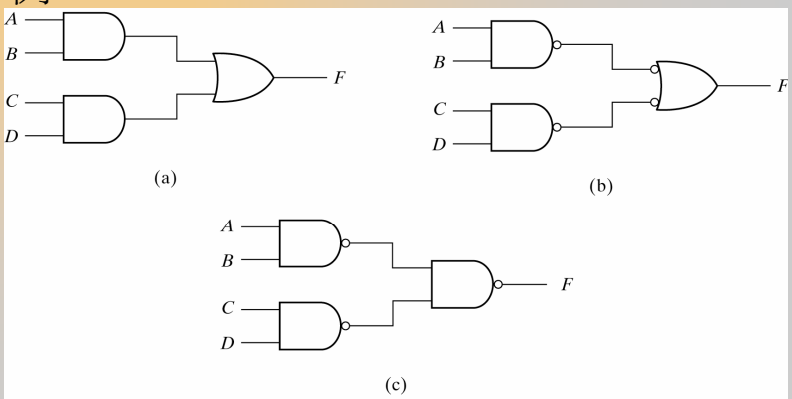


Fig. 3-20 Three Ways to Implement $F = AB + CD$

例題3-10利用NAND閘完成布林函數

$F(x, y, z) = \sum(1,2,3,4,5,7)$

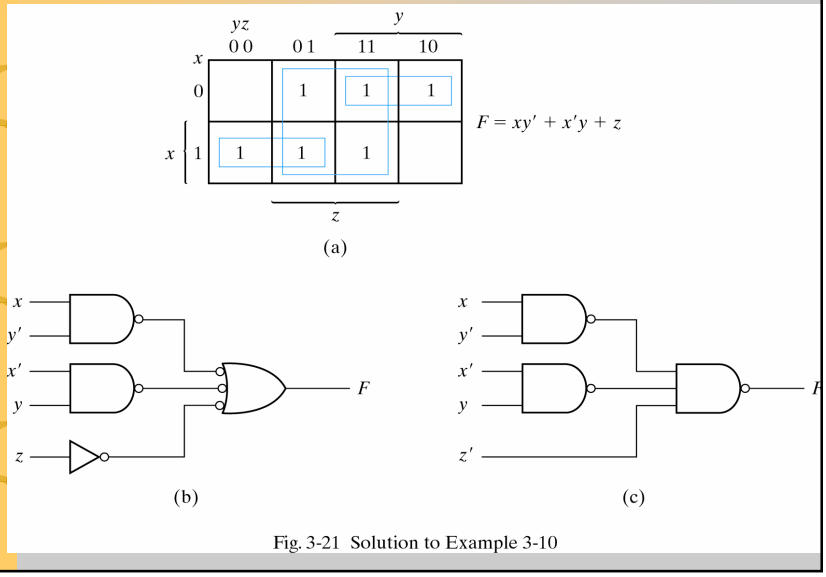


Fig. 3-21 Solution to Example 3-10

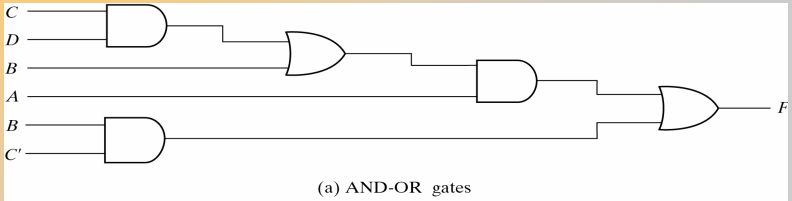
布林函數以二階NAND閘實現之步驟

- 將函數簡化成積項和的形式。
- 將表示式中至少含有2個字元的每一個積項用一個NAND閘來表示。每個NAND閘的輸入為每一項中的變數，由此構成邏輯電路圖的第一階邏輯閘。

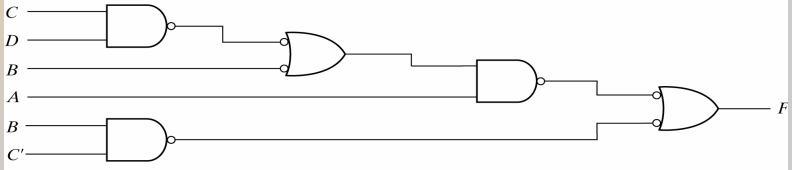
布林函數以二階NAND閘實現之步驟

- 在第二階的部分使用單一個AND-invert或invert-OR的閘圖示符號來表示，而其輸入來自第一階閘的輸出。
- 若函數中有任一項只含一個文字(變數)，則在第一階需要一個反相器，然而，若將單一個字元取補數，則可以直接將它連接到第二階NAND閘的一個輸入。

多階NAND閘



(a) AND-OR gates



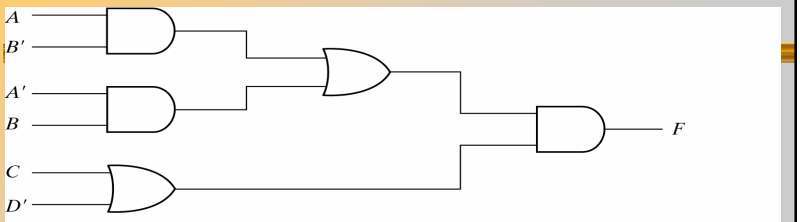
(a) NAND gates

Fig. 3-22 Implementing $F = A(CD + B) + BC$

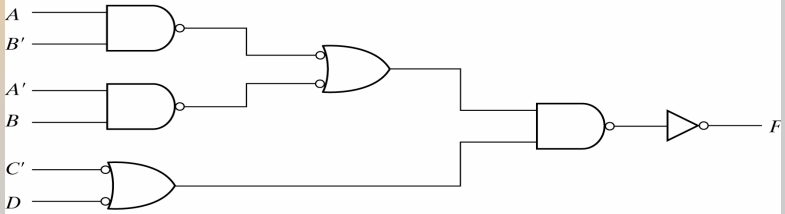
多階AND－OR電路轉換成多階NAND閘之轉換步驟

- 將所有的AND閘使用AND-invert圖形符號的NAND閘來取代。
- 將所有的OR閘使用invert-OR圖形符號的NAND閘來取代。
- 檢查圖中所有的小圈，在沿著同一條線上的每一個小圈若不是用來補償其它小圓的的話，則插入一個反相器(一個輸入的NAND閘)或是將輸入字元取補數。

範例 $F = (AB' + A'B)(C + D')$



(a) AND-OR gates



(b) NAND gates

Fig. 3-23 Implementing $F = (AB' + A'B)(C + D')$

NOR電路

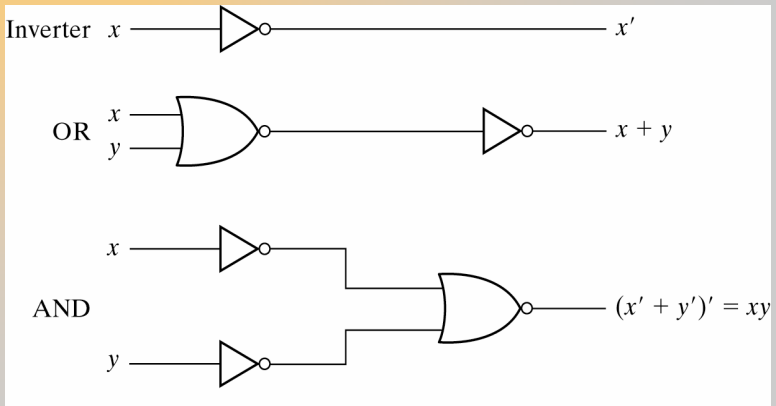
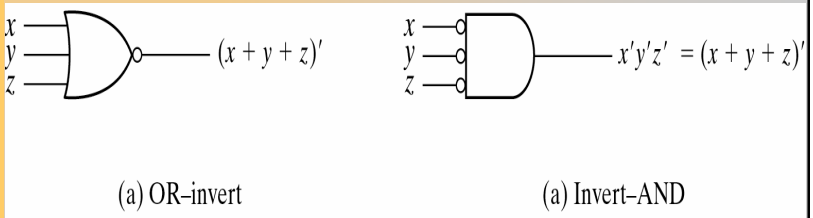


Fig. 3-24 Logic Operations with NOR Gates

NOR閘的圖示符號



(a) OR-invert (a) Invert-AND

Fig. 3-25 Two Graphic Symbols for NOR Gate

範例

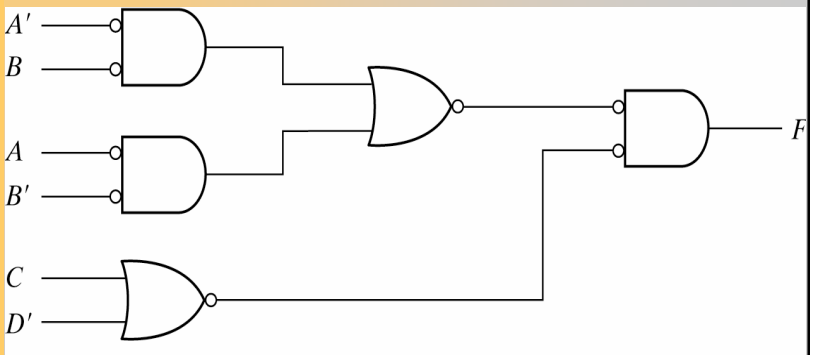
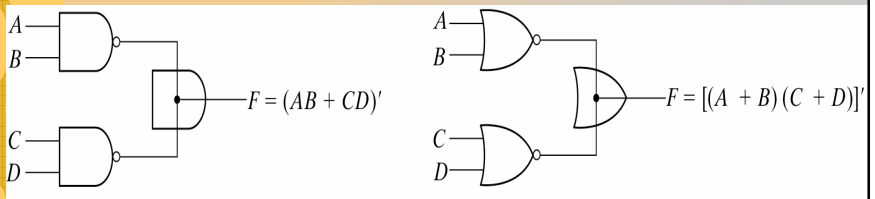


Fig. 3-27 Implementing $F = (AB' + A'B)(C + D')$ with NOR Gates

3-7 其他二階電路

- 線結邏輯



(a) Wired-AND in open-collector TTL NAND gates.

(AND-OR-INVERT)

(b) Wired-OR in ECL gates

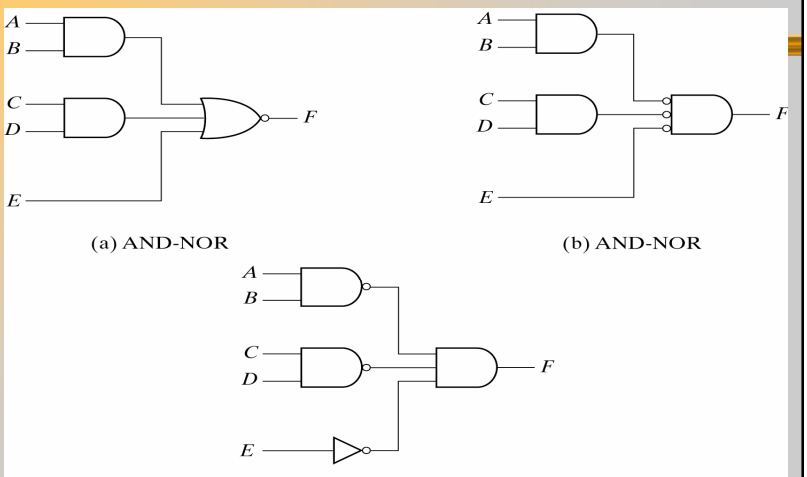
(OR-AND-INVERT)

Fig. 3-28 Wired Logic

非退化型式

- AND-OR
- NAND-NAND
- NOR-OR
- OR-NAND
- OR-AND
- NOR-NOR
- NAND-AND
- AND-NOR

AND－OR－INVERT電路

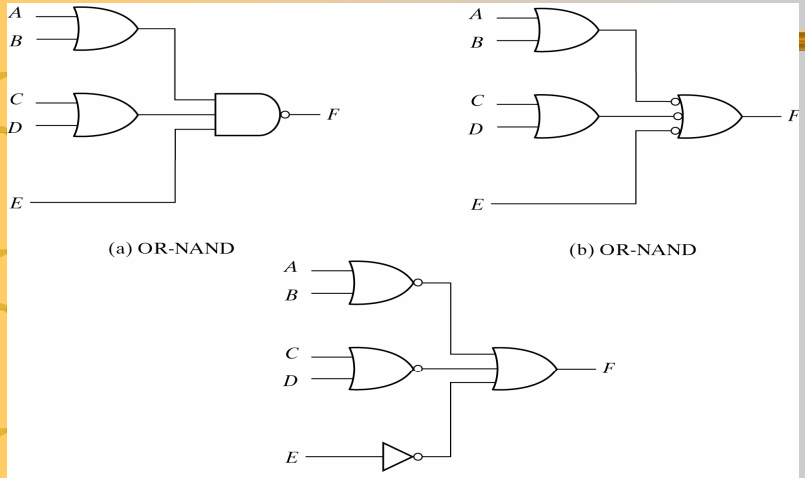


(a) AND-NOR　　(b) AND-NOR

(c) NAND-AND

Fig. 3-29 AND-OR-INVERT Circuits; $F = (AB + CD + E)'$

OR－AND－INVERT電路



(a) OR-NAND　　(b) OR-NAND

(c) NOR-OR

Fig. 3-30 OR-AND-INVERT Circuits; $F = [(A + B)(C + D)E]'$

其他二階型式實現電路

表 3-3　利用其他二階形式實現電路

等效的非退化形式 (a)	等效的非退化形式 (b)*	實現的函數	化簡 F' 為	得到的輸出
AND-NOR	NAND-AND	AND-OR-INVERT	積項和利用合併卡諾圖中的 0	F
OR-NAND	NOR-OR	OR-AND-INVERT	和項積利用合併卡諾圖中的 1 然後取補數	F

*對於一個單一字元項在形式(b)需要一個反相器

其他二階型式實現電路

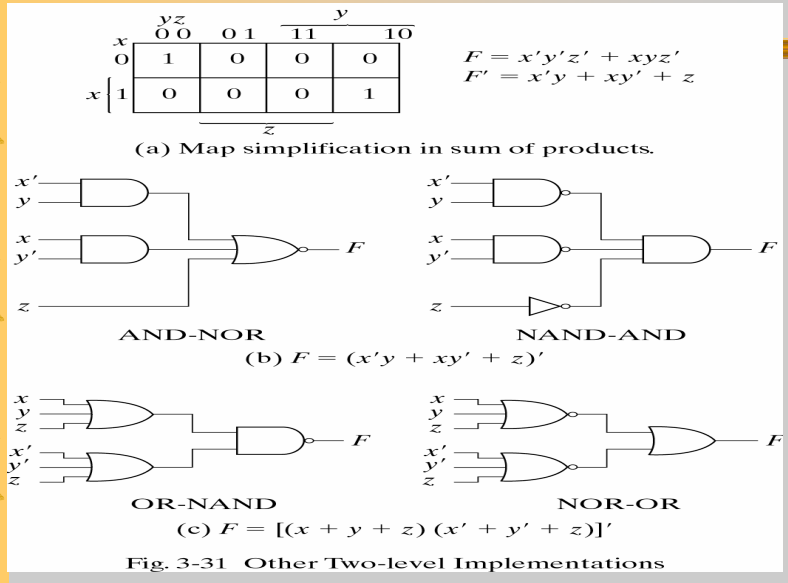


Fig. 3-31　Other Two-level Implementations

3-8 互斥-OR函數

★ 互斥-OR

$x \oplus y = xy' + x'y$

$x \oplus 0 = x$

$x \oplus 1 = x'$

$x \oplus x = 0$

$x \oplus x' = 1$

$x \oplus y' = x' \oplus y = (x \oplus y)'$

★ 全等

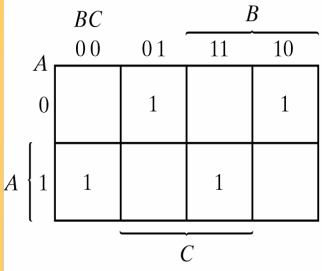
$$(x \oplus y)' = (xy' + x'y)'$$
$$= (x' + y)(x + y')$$
$$= xy + x'y'$$

奇函數、偶函數

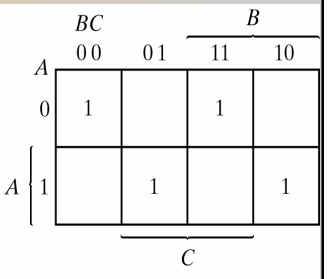
★ 奇函數---三個或更多個變數中，奇數個變數等於1則函數等於1

★ 偶函數---三個或更多個變數中，偶數個變數等於1則函數等於1

三變數奇函數、偶函數



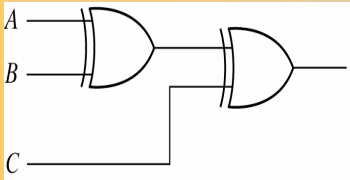
(a) Odd function
$F = A \oplus B \oplus C$



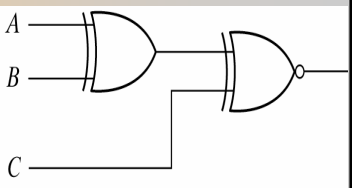
(a) Even function
$F = (A \oplus B \oplus C)'$

Fig. 3-33 Map for a Three-variable Exclusive-OR Function

三變數奇函數、偶函數



(a) 3-input odd function



(b) 3-input even function

Fig. 3-34 Logic Diagram of Odd and Even Functions

四變數奇函數、偶函數

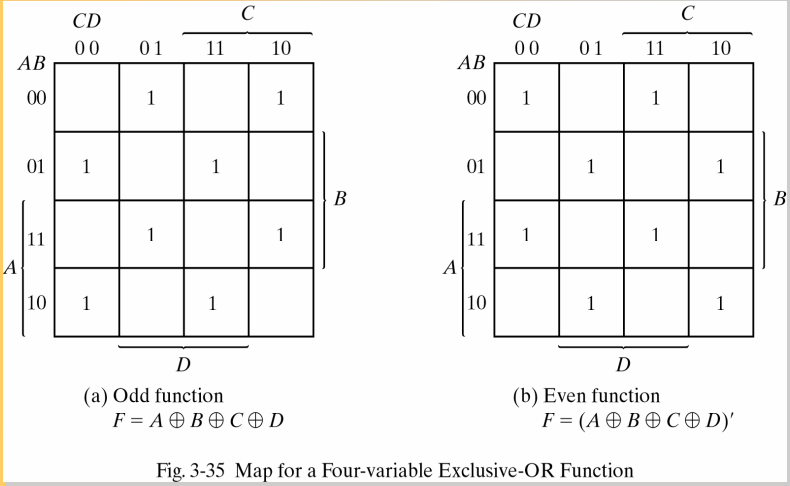


Fig. 3-35 Map for a Four-variable Exclusive-OR Function

同位產生器與檢查器

- 同位產生器
- 同位檢查器
- 偶同位產生器

表 3-4 偶同位產生器之真值表

3 位元訊息			同位位元
x	*y*	*z*	*P*
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	0
1	1	1	1

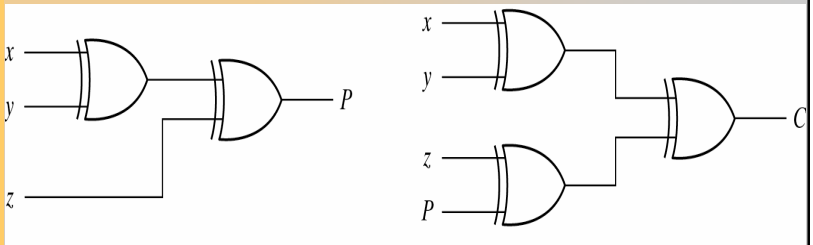
偶同位檢查器

表 3-5　偶同位檢查器之真值表

接收到的 4 個位元				同位錯誤檢查
x	y	z	P	C
0	0	0	0	0
0	0	0	1	1
0	0	1	0	1
0	0	1	1	0
0	1	0	0	1
0	1	0	1	0
0	1	1	0	0
0	1	1	1	1
1	0	0	0	1
1	0	0	1	0
1	0	1	0	0
1	0	1	1	1
1	1	0	0	0
1	1	0	1	1
1	1	1	0	1
1	1	1	1	0

同位產生器與檢查器

$P = x \oplus y \oplus z$　　$C = x \oplus y \oplus z \oplus P$



(a) 3-bit even parity generator　　(a) 4-bit even parity checker

Fig. 3-36 Logic Diagram of a Parity Generator and Checker

3-9 硬體描述語言

- 邏輯模擬(logic simulation)
- 邏輯合成(logic synthesis)
- 測試平台(test bench)

模組表示法

- HDL 範例3 -1

```
//Description of the simple circuit of Fig. 3-37
module  smpl_circuit(A,B,C,x,y);
   input  A,B,C;
   output  x,y;
   wire  e;
   and g1(e,A,B);
   not  g2(y, C);
   or    g3(x,e,y);
endmodule
```

HDL電路

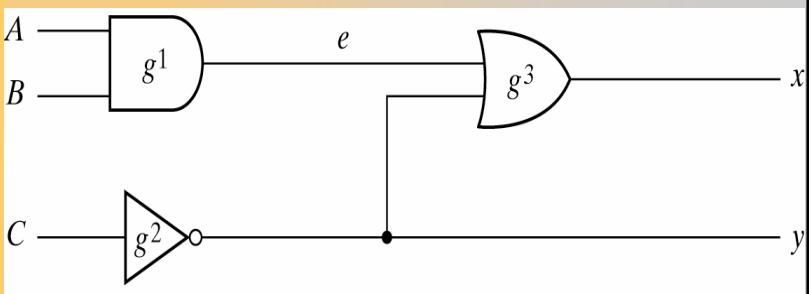


Fig. 3-37 Circuit to Demonstrate HDL

閘延遲

✶HDL 範例 3-2

```
//Description of circuit with delay
module  circuit_with_delay (A,B,C,x,y);
   input  A,B,C;
   output  x,y;
   wire e;
   and  #(30)  g1(e,A,B);
   or     #(20)  g3(x,e,y);
   not   #(10)  g2(y,C);
endmodule
```

HDL 範例 3-3
(模擬一個具有延遲電路的測試平台)

```
//Stimulus for simple circuit
module  stimcrct;
reg A,B,C;
wire  x,y;
circuit_with_delay cwd(A,B,C,x,y);
initial
```

HDL 範例 3-3
(模擬一個具有延遲電路的測試平台)

```
  begin
      A = 1'b0; B = 1'b0; C = 1'b0;
   #100
      A = 1'b1; B = 1'b1; C = 1'b1;
   #100  $finish;
  end
endmodule
```

HDL 範例 3-3
(模擬一個具有延遲電路的測試平台)

```
//Description of circuit with delay
module  circuit_with_delay (A,B,C,x,y);
   input  A,B,C;
   output  x,y;
   wire  e;
   and  #(30) g1(e,A,B);
   or   #(20) g3(x,e,y);
   not  #(10) g2(y,C);
endmodule
```

布林表示式

$$x = A + BC + B'D$$
$$y = B'C + BC'D'$$

✶HDL 範例 3-4

```
//Circuit specified with Boolean equations
module  circuit_bln (x,y,A,B,C,D);
   input  A,B,C,D;
   output  x,y;
   assign  x = A | (B & C) | (~B & C);
   assign  y = (~B & C) | (B & ~C & ~D);
endmodule
```

使用者定義的原始值

- 使用關鍵字 primitive 作宣告
- 只可以有一個輸出且此輸出必須列在埠名單的第一個還有必須用關鍵字output來宣告。
- 輸入數目不限制，至於它們在input宣告中的順序則必須與它們在下面表中所給值的順序相同。

使用者定義的原始值(UDP)

- 真值表必須在關鍵字table及endtable之間。
- 輸入值依順序列出，用冒號(：)代表結束，輸出通常是每一列的最後一個記錄，後面跟著是一個分號(；)。
- 最後用關鍵字endprimitive作結束。

HDL 範例 3-5
(利用一個真值表來定義UDP)

```
//User defined primitive(UDP)
primitive  crctp (x,A,B,C);
   output  x;
   input A,B,C;
//Truth table for x(A,B,C) = Minterms (0,2,4,6,7)
   table
```

HDL 範例 3-5
(利用一個真值表來定義UDP)

```
//    A  B  C  :  x  (Note that this is only a comment)
      0  0  0  :  1;
      0  0  1  :  0;
      0  1  0  :  1;
      0  1  1  :  0;
      1  0  0  :  1;
      1  0  1  :  0;
      1  1  0  :  1;
      1  1  1  :  1;
   endtable
endprimitive
```

HDL 範例 3-5
(利用一個真值表來定義UDP)

```
//Instantiate primitive
module declare_crctp;
    reg  x,y,z;
    wire w;
    crctp (w,x,y,z);
endmodule
```