第8章

多工器、解碼器及可程式規劃邏輯元件

CENGAGE Learning

8.1 簡介

- 小型積體電路（SSI）：功能包括NAND、NOR、AND 和OR閘、反相器及正反器。SSI積體電路內典型的包裝通常具有一到四個閘、六個反相器，或是一或兩個正反器。
- 中型積體電路（MSI）：包括如加法器、多工器、解碼器、暫存器及計數器，可以執行較為複雜的函數。這種積體電路典型的包裝大概含有相當於12到100個閘。
- 大型積體電路（LSI）：單一個包裝裡約含有100到幾千個閘。
- 超大型積體電路（VLSI）：含有數千個以上的閘。

8.2 多工器

- **多工器**（multiplexer，或資料選擇器，縮寫為MUX）：具有一群的資料輸入和一群的控制輸入。控制輸入是用來選擇一條資料輸入連接到輸出端。

8.2 多工器

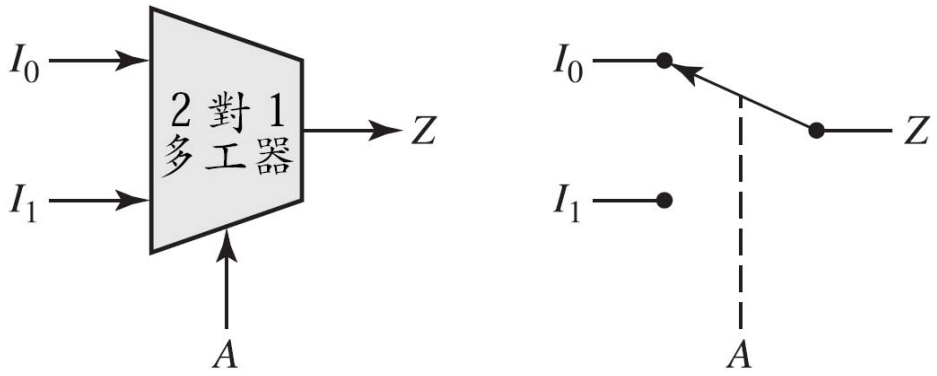


圖 8-1　2 對 1 多工器及開關類型

- 當控制輸入A 為0 時，則開關在上面的位置且多工器輸出為$Z = I_0$。
- 2 對1 多工器的邏輯等式為：

$$Z = A'I_0 + AI_1$$

8.2 多工器

- 圖8-2所示為一個4對1多工器、8對1多工器及2^n對1多工器。
- 8 對1 多工器利用三個控制輸入選擇八個輸入中的一個，可用下面的等式來描述：

$$Z = A'B'C'I_0 + A'B'CI_1 + A'BC'I_2 + A'BCI_3 + AB'C'I_4 + AB'CI_5 + ABC'I_6 + ABCI_7 \quad (8\text{-}2)$$

8.2 多工器

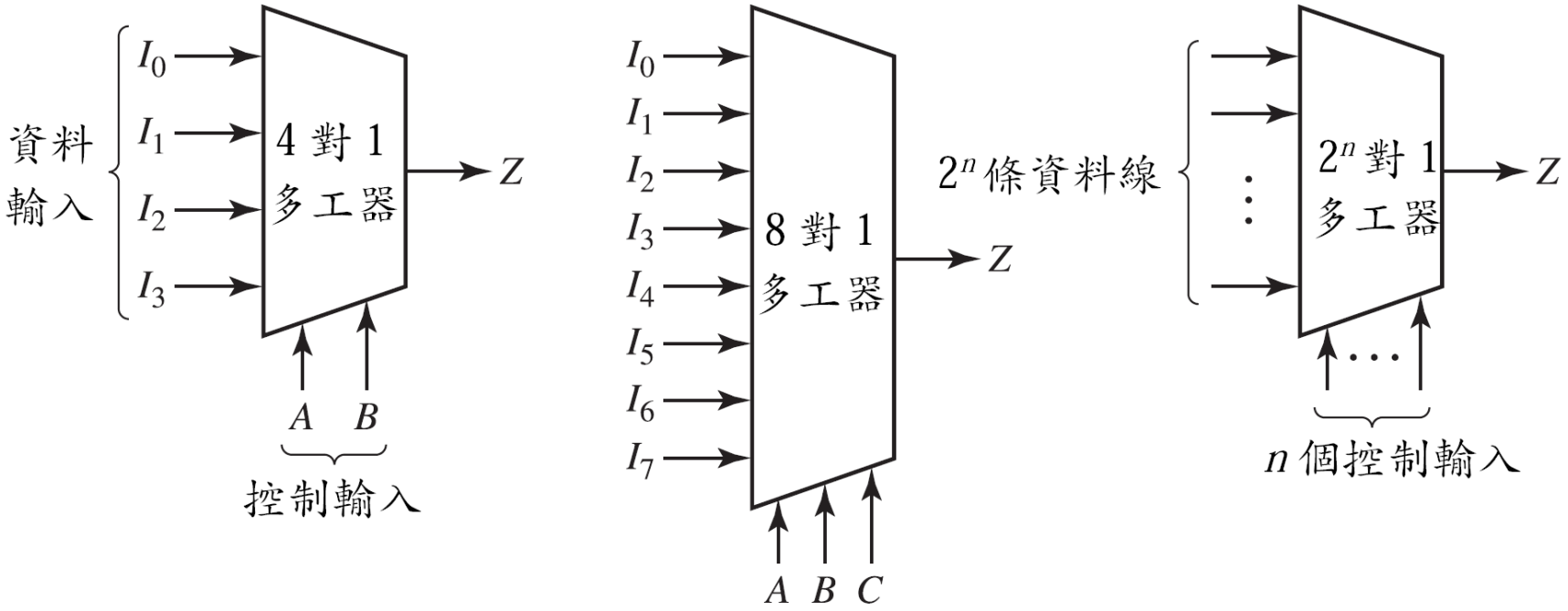


圖 8-2 多工器

8.2 多工器

❖圖8-3 所示是一個8 對1 多工器的內部邏輯圖。

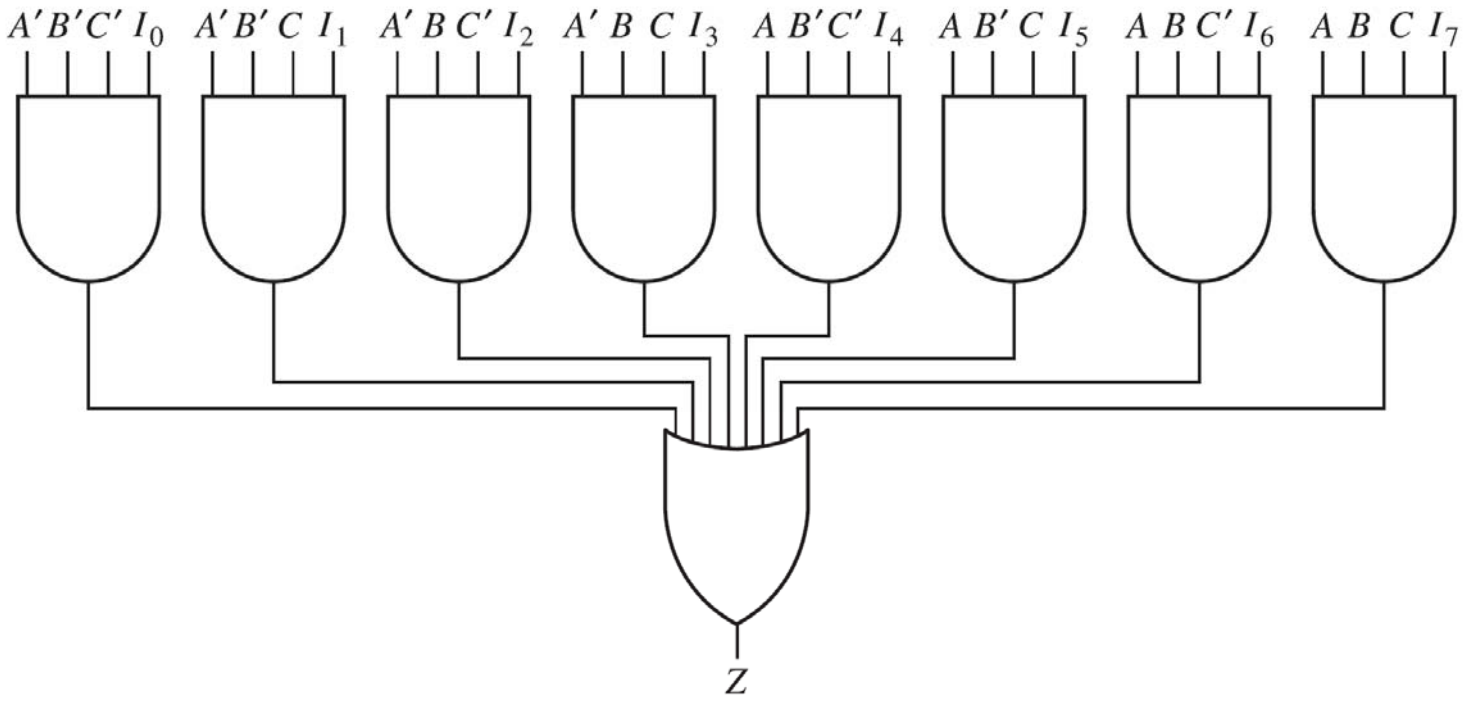


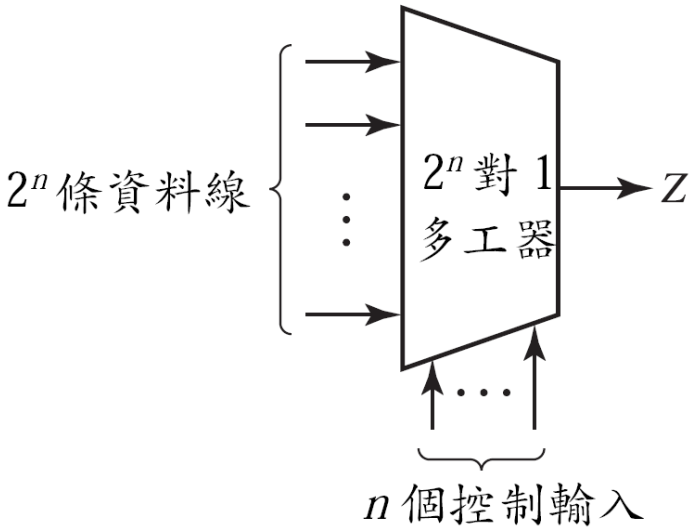
圖 8-3 8 對 1 多工器的邏輯圖

8.2 多工器

- 對於一個具有n個控制輸入和2^n個資料輸入之多工器輸出的通式為：

$$Z = \sum_{k=0}^{2^n-1} m_k I_k$$

- 其中m_k是n個控制變數的一個全及項，且I_k是相對的資料輸入。



8.2 多工器

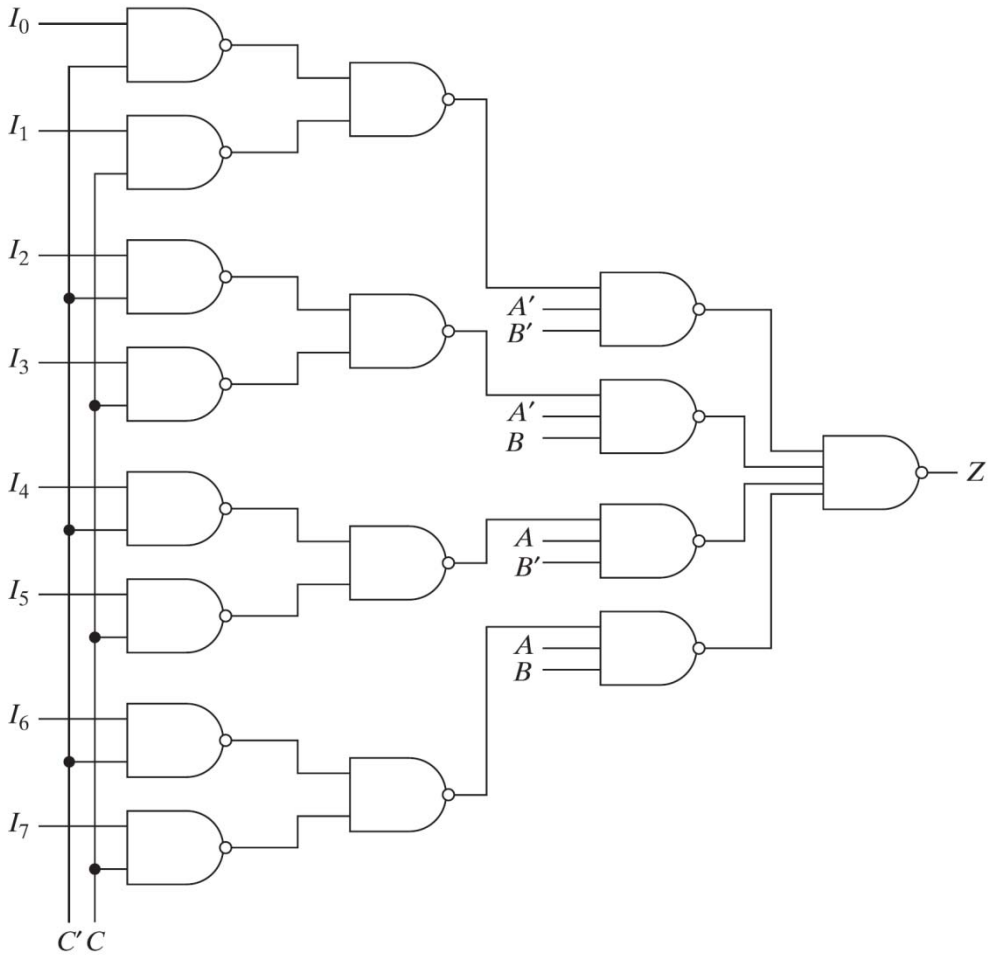


圖 8-4 一個 8 對 1 多工器之多階電路

8.2 多工器

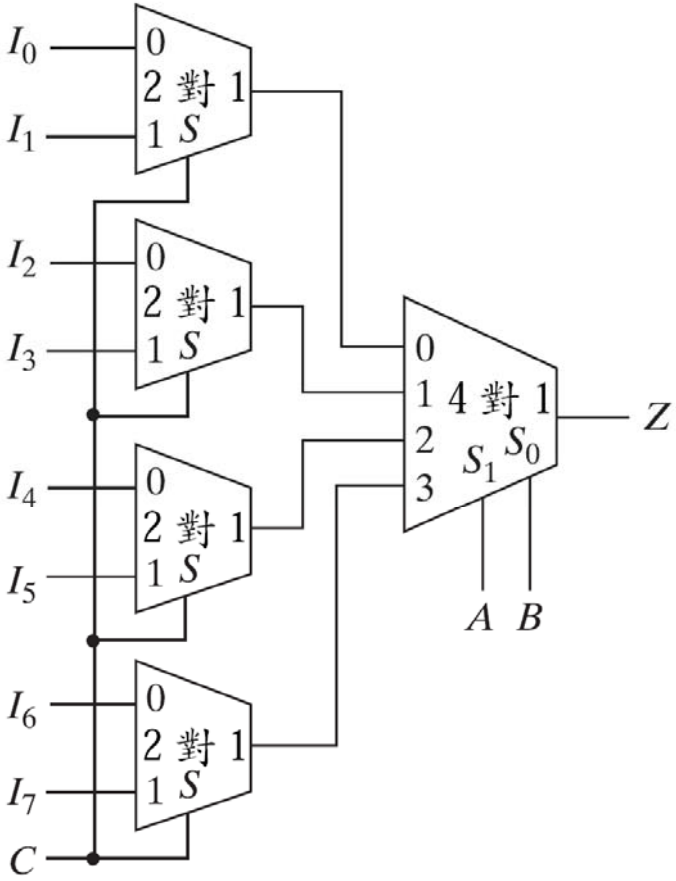


圖 8-5　圖 8-4 之多工器元件

8.2 多工器

- 圖8-6所示為一個四連2對1多工器

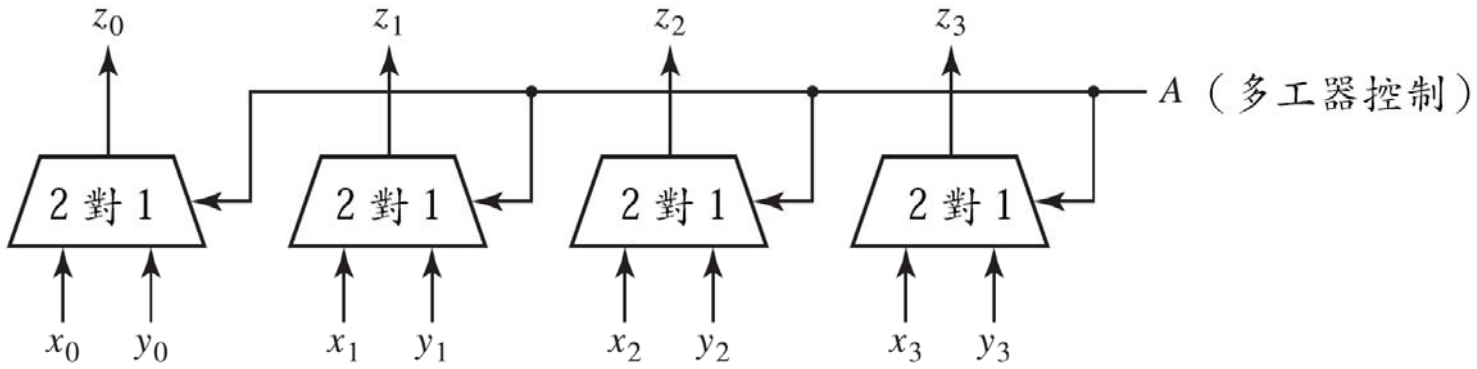


圖 8-6 用來選擇資料的四連多工器

- 若控制信號$A=0$，則x_0、x_1、x_2和x_3會出現在輸出端z_0、z_1、z_2和z_3；若$A=1$，則y_0、y_1、y_2和y_3會出現在輸出端。

8.2 多工器

- 具有匯流排輸入和輸出的四連多工器：
  - 當$A = 0$時，匯流排X的信號會出現在匯流排Z；否則，匯流排Y的信號會出現。
  - 在匯流排上一個旁邊帶有數字之對角的斜線是定義此匯流排的位元數目。

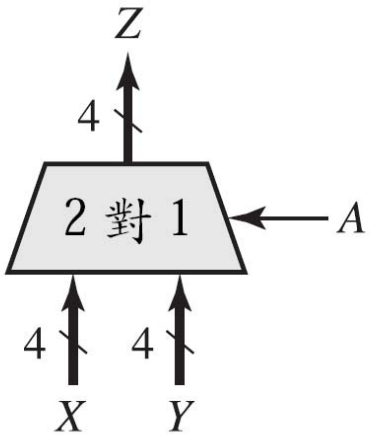


圖 8-7 具有匯流排輸入和輸出的四連多工器

8.2 多工器

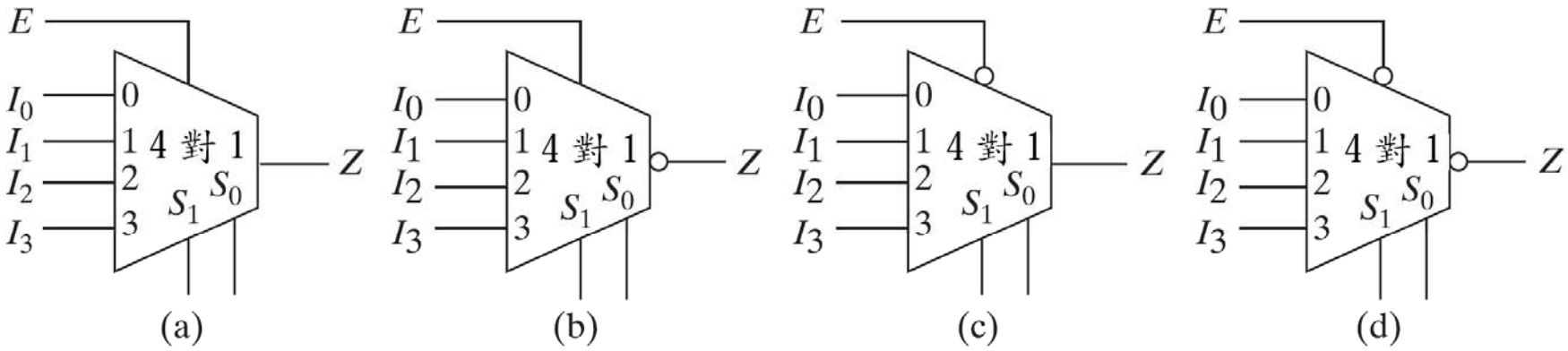


圖 8-8 主動高、主動低致能及輸出組合

8.2 多工器

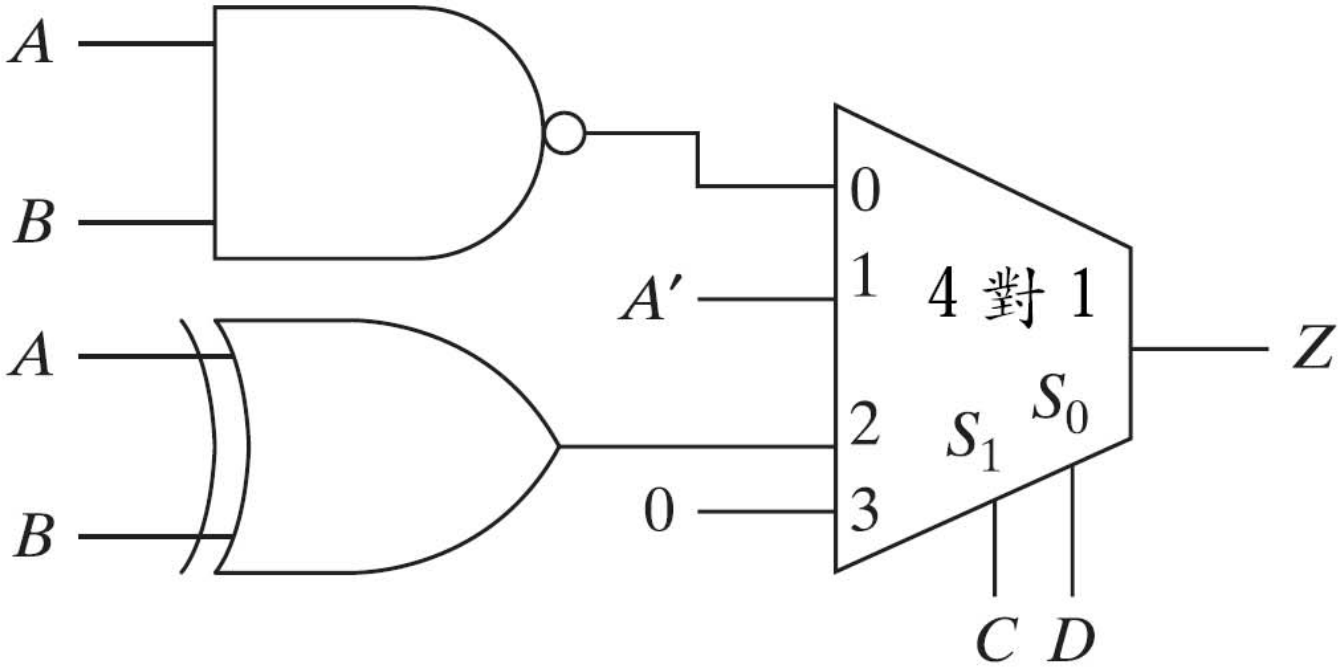


圖 8-9 具有一個 4 對 1 多工器之 4 變數函數電路

8.3 三態緩衝器

❖緩衝器：用來增加閘輸出的推動能力。

$$F = C$$

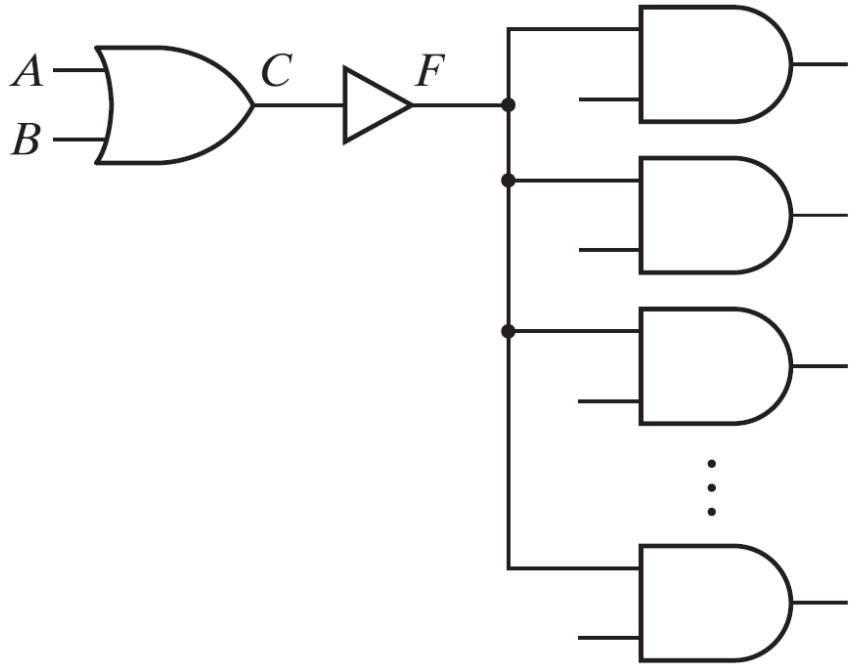


圖 8-10 加入緩衝器的閘電路

8.3 三態緩衝器

- 如果將兩個或兩個以上的閘或其他邏輯元件的輸出直接連接在一起，會使得該邏輯電路無法正常的動作。
- 使用三態邏輯，將允許兩個或兩個以上的閘或其他邏輯元件的輸出可以連接在一起。
- 圖8-11 所示為三態緩衝器及其邏輯等效電路。

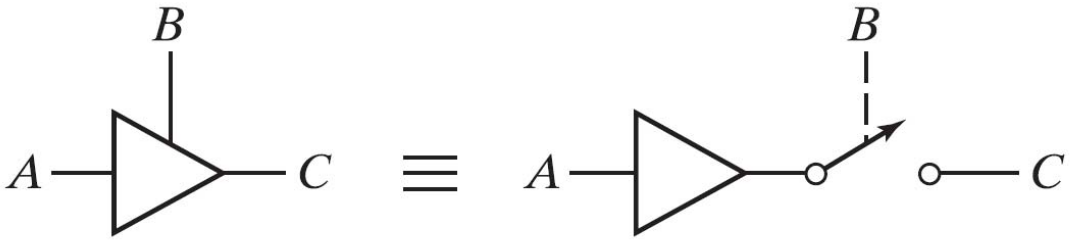


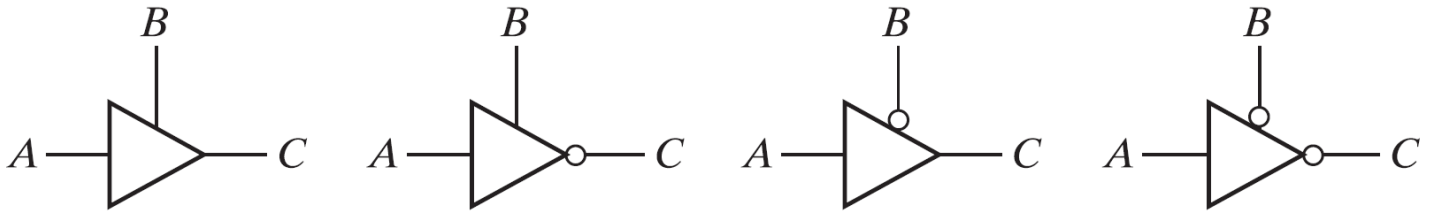
圖 8-11　三態緩衝器

8.3 三態緩衝器

- 當**致能輸入**（enable input）B為1時，輸出C等於A；當B為0時，則C就像是開路。換句話說，當B為0時，輸出C被有效地與緩衝器隔絕，所以不會有電流流通。這就是所謂輸出的高阻抗狀態，因為對電流來說，該電路提供一個很高的電阻或阻抗值，**三態緩衝器**（three-state buffers也稱為tri-state buffers）。

8.3 三態緩衝器

❖圖8-12是四種形式之三態緩衝器的真值表。



B	A	C
0	0	Z
0	1	Z
1	0	0
1	1	1

(a)

B	A	C
0	0	Z
0	1	Z
1	0	1
1	1	0

(b)

B	A	C
0	0	0
0	1	1
1	0	Z
1	1	Z

(c)

B	A	C
0	0	1
0	1	0
1	0	Z
1	1	Z

(d)

圖 8-12 四種類型的三態緩衝器

8.3 三態緩衝器

- 在圖8-12(a) 及(b)，致能輸入B 並未反相，所以緩衝器的輸出在B=1 時被**致能**（enable）；而當$B = 0$ 時為**禁能**（disable）。也就是當$B = 1$ 時，緩衝器會正常動作；而當$B = 0$ 時，緩衝器輸出為開路狀態。我們使用符號Z來表示這個高阻抗狀態。在圖8-12(b)，緩衝器的輸出被反相，所以當緩衝器被致能時，輸出為$C = A'$。在圖8-12(c)及(d)之緩衝器與在(a)和(b)的動作相同，差別只在於致能輸入為反相，所以當$B = 0$ 時緩衝器會被致能。

8.3 三態緩衝器

❖在圖8-13 中，兩個三態緩衝器的輸出被連接在一起。

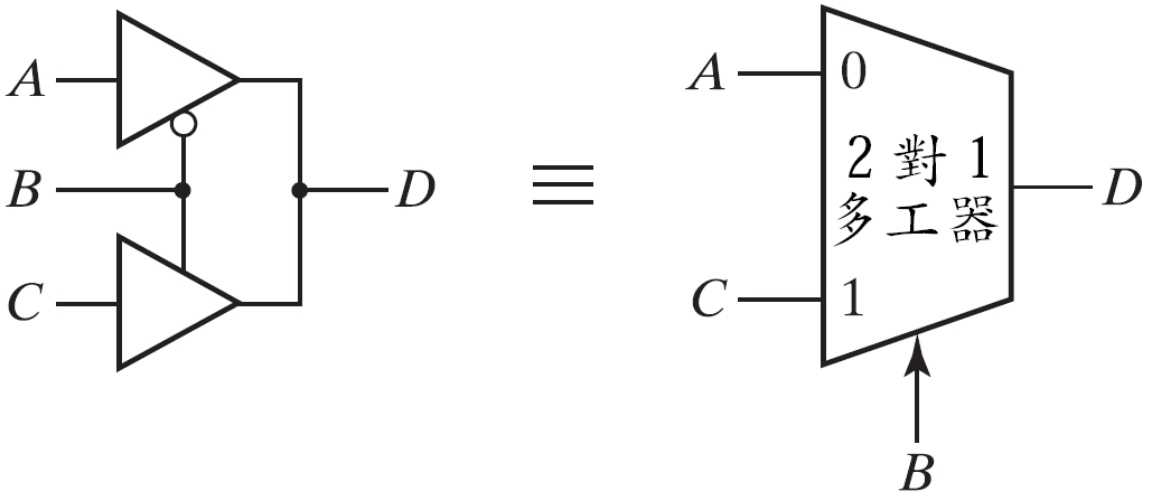
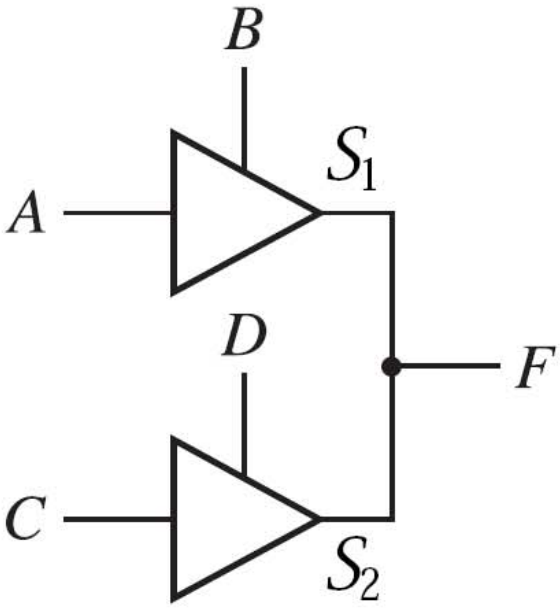


圖 8-13　利用三態緩衝器作資料選取

8.3 三態緩衝器

- 當$B=0$時，上方的緩衝器被致能，所以$D=A$；當$B=1$時，下方的緩衝器被致能，所以$D=C$。因此，$D=B'A+BC$，這在邏輯上相當於是以2對1多工器來作選擇，當$B=0$時，選取輸入A；當$B=1$時，則選取輸入C。

8.3 三態緩衝器



S_1	S_2 = X	S_2 = 0	S_2 = 1	S_2 = Z
X	X	X	X	X
0	X	0	X	0
1	X	X	1	1
Z	X	0	1	Z

圖 8-14 具有兩個三態緩衝器的電路

8.3 三態緩衝器

- 如圖8-14 所示，當我們將兩個三態緩衝器的輸出連接在一起時，假如其中一個緩衝器被禁能（輸出＝Z），則合併的輸出F 會與另外一個緩衝器的輸出相同；假如兩個緩衝器都被禁能，則輸出為Z；假如兩個緩衝器都被致能，則會發生混淆。
- 當一個匯流排是利用三態緩衝器來推動時，則稱為三態匯流排。在匯流排上的信號有0、1、Z和假定的X 的值。

8.3 三態緩衝器

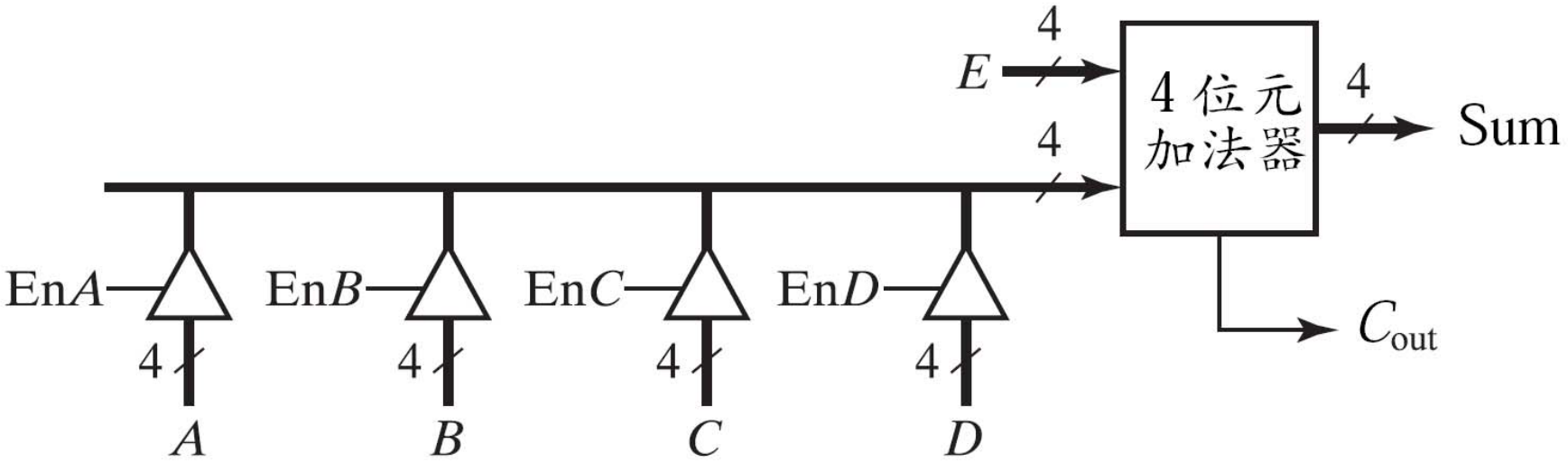


圖 8-15 對於一個運算元具有四個來源之 4 位元加法器

8.3 三態緩衝器

- 具有雙向輸入／輸出接腳的積體電路：

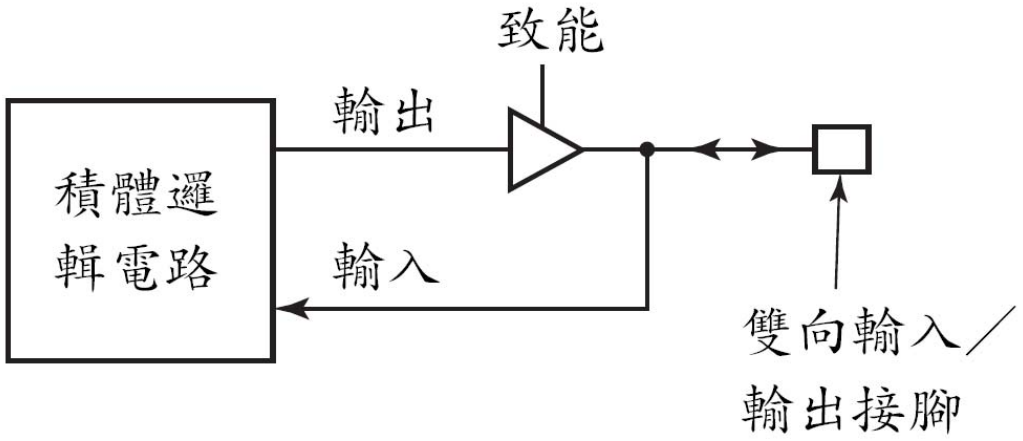
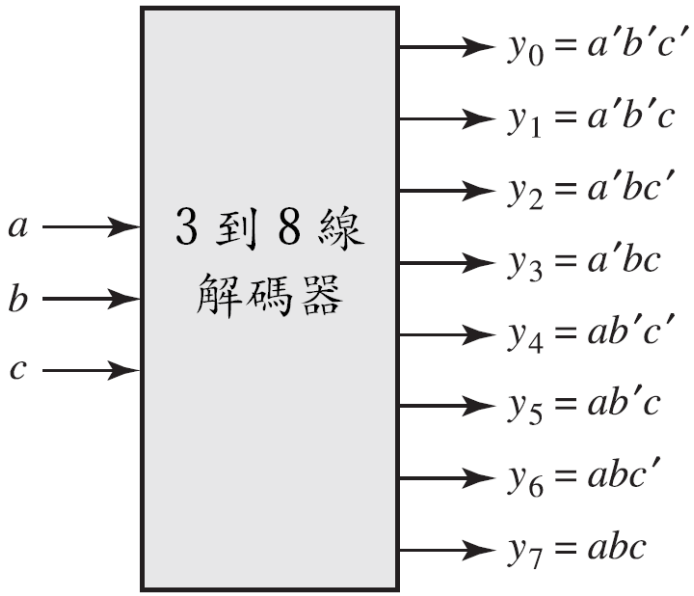


圖 8-16 具有雙向輸入／輸出接腳的積體電路

- 當緩衝器被致能時，接腳被輸出信號推動，當緩衝器被禁能時，外部的來源可以推動輸入接腳。

8.4 解碼器與編碼器

- 3到8線解碼器：對於輸入變數值的每一種組合方式都恰好使得一條輸出線的值是1。

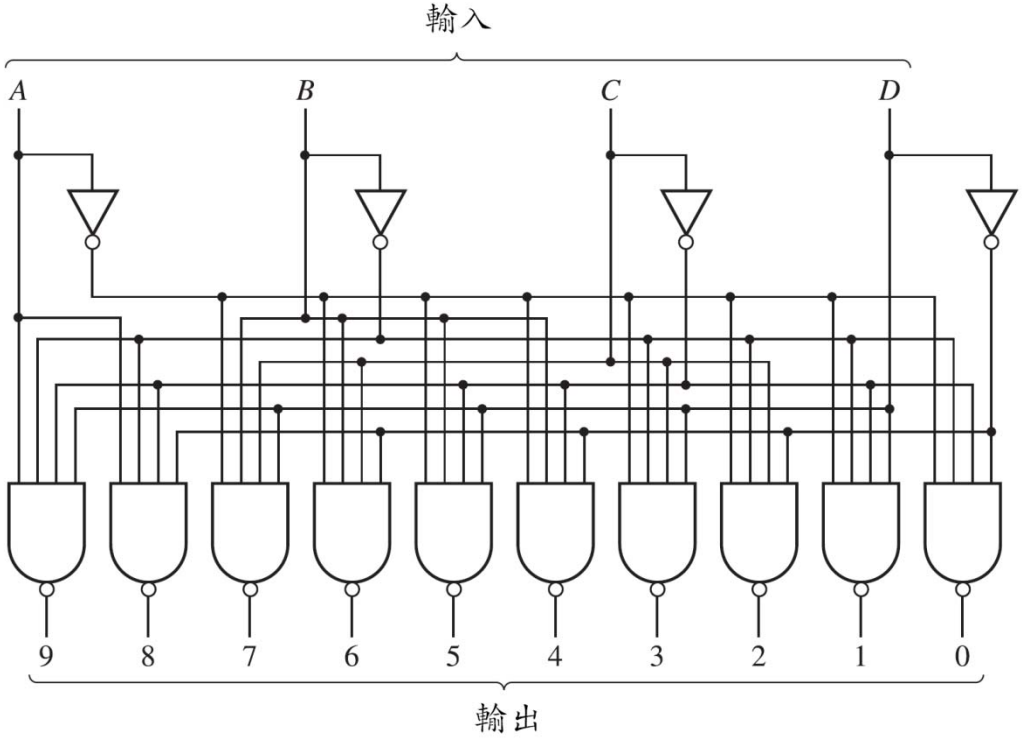


a b c	y_0	y_1	y_2	y_3	y_4	y_5	y_6	y_7
0 0 0	1	0	0	0	0	0	0	0
0 0 1	0	1	0	0	0	0	0	0
0 1 0	0	0	1	0	0	0	0	0
0 1 1	0	0	0	1	0	0	0	0
1 0 0	0	0	0	0	1	0	0	0
1 0 1	0	0	0	0	0	1	0	0
1 1 0	0	0	0	0	0	0	1	0
1 1 1	0	0	0	0	0	0	0	1

圖 8-17　3 到 8 線解碼器

8.4 解碼器與編碼器

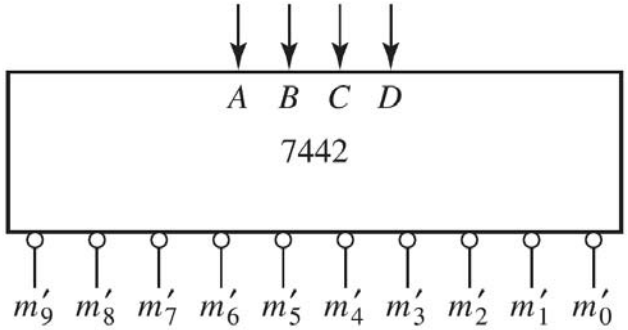
❖4到10線解碼器(輸出為反相式)：對於輸入值的每一種組合方式都恰好使得一條輸出線是0。



(a) 邏輯圖

圖 8-18 4 到 10 線解碼器

8.4 解碼器與編碼器



(b) 方塊圖

BCD 輸入				十進位數輸出									
A	*B*	*C*	*D*	0	1	2	3	4	5	6	7	8	9
0	0	0	0	0	1	1	1	1	1	1	1	1	1
0	0	0	1	1	0	1	1	1	1	1	1	1	1
0	0	1	0	1	1	0	1	1	1	1	1	1	1
0	0	1	1	1	1	1	0	1	1	1	1	1	1
0	1	0	0	1	1	1	1	0	1	1	1	1	1
0	1	0	1	1	1	1	1	1	0	1	1	1	1
0	1	1	0	1	1	1	1	1	1	0	1	1	1
0	1	1	1	1	1	1	1	1	1	1	0	1	1
1	0	0	0	1	1	1	1	1	1	1	1	0	1
1	0	0	1	1	1	1	1	1	1	1	1	1	0
1	0	1	0	1	1	1	1	1	1	1	1	1	1
1	0	1	1	1	1	1	1	1	1	1	1	1	1
1	1	0	0	1	1	1	1	1	1	1	1	1	1
1	1	0	1	1	1	1	1	1	1	1	1	1	1
1	1	1	0	1	1	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1	1	1	1	1	1

(c) 真值表

圖 8-18 4 到 10 線解碼器

8.4 解碼器與編碼器

- 一般而言，一個n輸入變數之n到2^n線解碼器可以產生2^n個全及項（或全或項）。其輸出可以由下面的等式來定義：

$$y_i = m_i = M'_i \qquad i = 0 \text{ 到 } 2^n - 1 \quad \text{(非反相輸出)} \quad (8\text{-}5)$$

或

$$y_i = m'_i = M_i \qquad i = 0 \text{ 到 } 2^n - 1 \quad \text{(反相輸出)} \quad (8\text{-}6)$$

其中m_i是n輸入變數的全及項且M_i是全或項。

8.4 解碼器與編碼器

- 一個n輸入解碼器可以產生n變數之所有全及項，因此n變數的函數可以藉由選擇一個解碼器的全及項輸出，然後OR在一起來實現。若解碼器的輸出是反相，則可以用NAND閘來實現函數，如下面的例子所作的說明。
- 實現：

$$f_1(a,b,c,d) = m_1 + m_2 + m_4 \quad \text{且} \quad f_2(a,b,c,d) = m_4 + m_7 + m_9$$

8.4 解碼器與編碼器

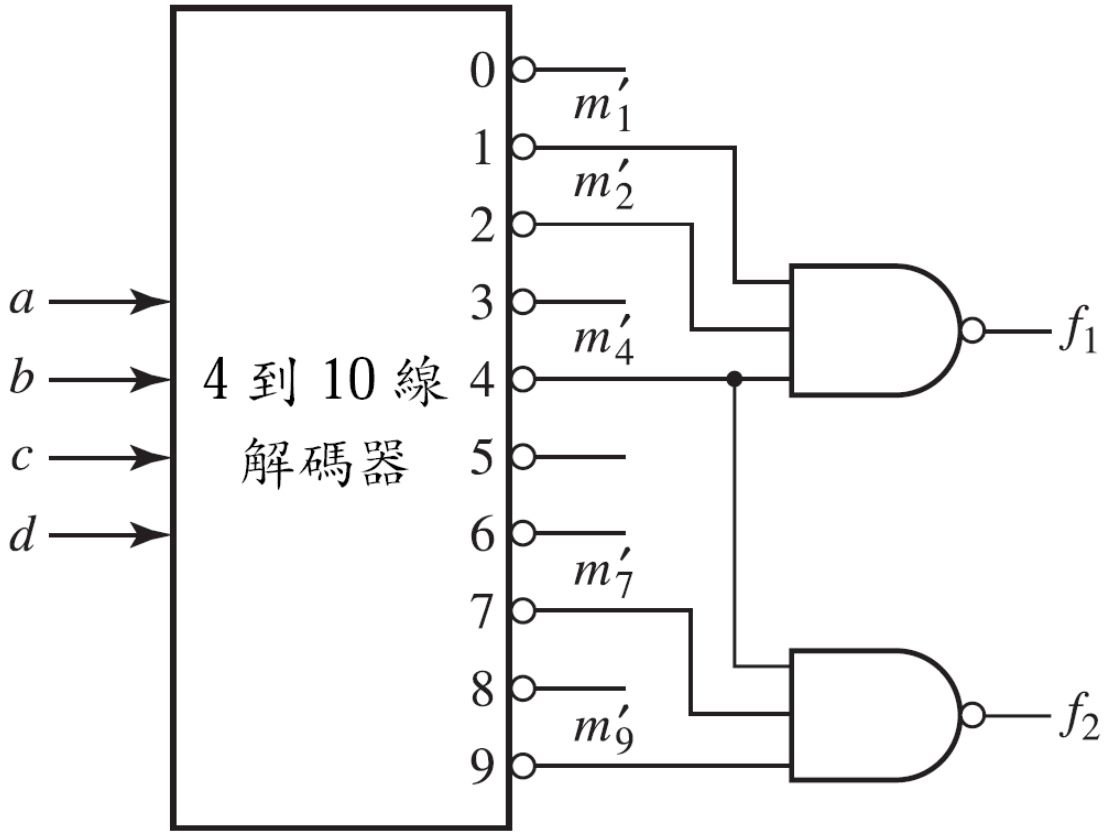
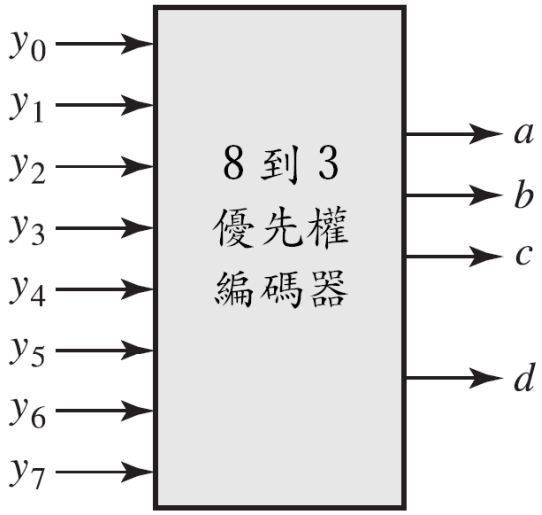


圖 8-19 利用一個解碼器實現多重輸出電路

8.4 解碼器與編碼器

編碼器

- 編碼器執行的是解碼器的反函數
- 8 到3優先權編碼器：



y_0	y_1	y_2	y_3	y_4	y_5	y_6	y_7	a	b	c	d
0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0	0	1
X	1	0	0	0	0	0	0	0	0	1	1
X	X	1	0	0	0	0	0	0	1	0	1
X	X	X	1	0	0	0	0	0	1	1	1
X	X	X	X	1	0	0	0	1	0	0	1
X	X	X	X	X	1	0	0	1	0	1	1
X	X	X	X	X	X	1	0	1	1	0	1
X	X	X	X	X	X	X	1	1	1	1	1

圖 8-20 8 到 3 優先權編碼器

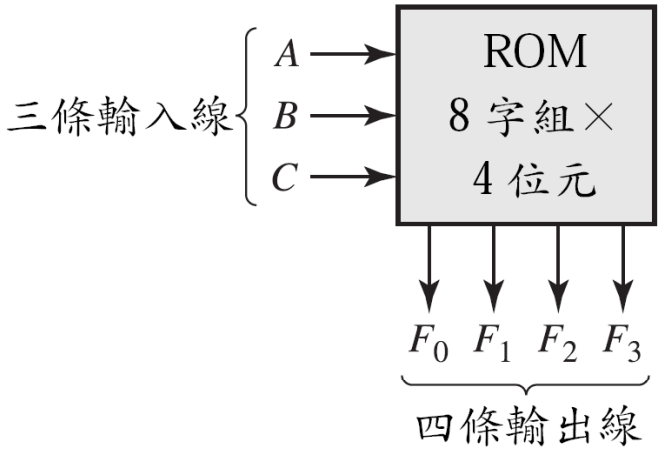
8.4 解碼器與編碼器

編碼器

- 若輸入y_i是1且其他輸入是0，則輸出abc代表一個等於i的二進位數，例如：若$y_3 = 1$，則$abc = 011$。如果同時有超過一個的輸入是1，則輸出可以利用優先權圖表來定義，在圖8-20的真值表使用下列的圖表：假如超過一個輸入是1，則最高數字的輸入決定輸出。例如：若輸入y_1、y_4和y_5是1，則輸出$abc = 101$，在真值表中的X表示不理會；例如：若y_5是1，我們不理會y_0到y_4的輸入為何，如果任何的輸入是1，則輸出d是1；否則，d是0。這個信號主要是用來區分所有輸入都是0，還是只有y_0是1的情況。

8.5 唯讀記憶體

❖ **唯讀記憶體**（read-only memory, ROM）是由彼此連接的半導體元件陣列所組成，用來儲存二進位資料陣列。



(a) 方塊圖

A B C	F_0 F_1 F_2 F_3
0 0 0	1 0 1 0
0 0 1	1 0 1 0
0 1 0	0 1 1 1
0 1 1	0 1 0 1
1 0 0	1 1 0 0
1 0 1	0 0 0 1
1 1 0	1 1 1 1
1 1 1	0 1 0 1

儲存在 ROM 內典型的資料（2^3 個字組，每個字組 4 位元）

(b) ROM 的真值表

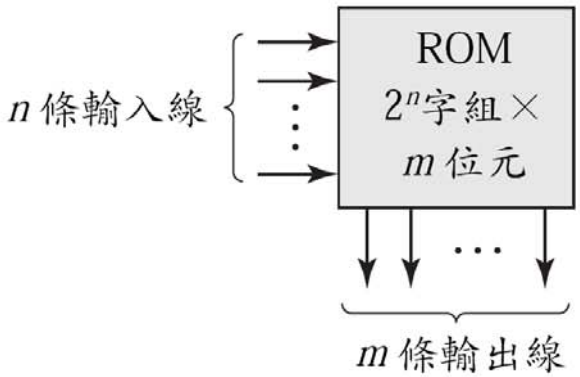
圖 8-21 一個 8 字組×4 位元 ROM

8.5 唯讀記憶體

- 儲存在ROM內的每一個輸出樣式稱為一個**字組**（word）。
- 因為這個ROM有三條輸入線，所以有$2^3 = 8$個不同組合的輸入值。每一種輸入組合當作一個可以選擇儲存在記憶體內8個字組其中之一的**位址**（address）。
- 因為有四條輸出線，每個字組長度是4位元，所以這個ROM的大小為8字組×4位元。

8.5 唯讀記憶體

❖ 具有n 條輸入線和m 條輸出線的ROM：



n 輸入變數	m 輸出變數
00 · · · 00	100 · · · 110
00 · · · 01	010 · · · 111
00 · · · 10	101 · · · 101
00 · · · 11	110 · · · 010
⋮	⋮
11 · · · 00	001 · · · 011
11 · · · 01	110 · · · 110
11 · · · 10	011 · · · 000
11 · · · 11	111 · · · 101

儲存在 ROM 內典型的資料陣列（2^n個字組，每個字組 m 位元）

圖 8-22 具有 n 輸入及 m 輸出的 ROM

8.5 唯讀記憶體

- 一個$2^n \times m$ 的ROM可以實現n變數之m個函數。
- 一個ROM 基本上包含一個解碼器和一個記憶體陣列：

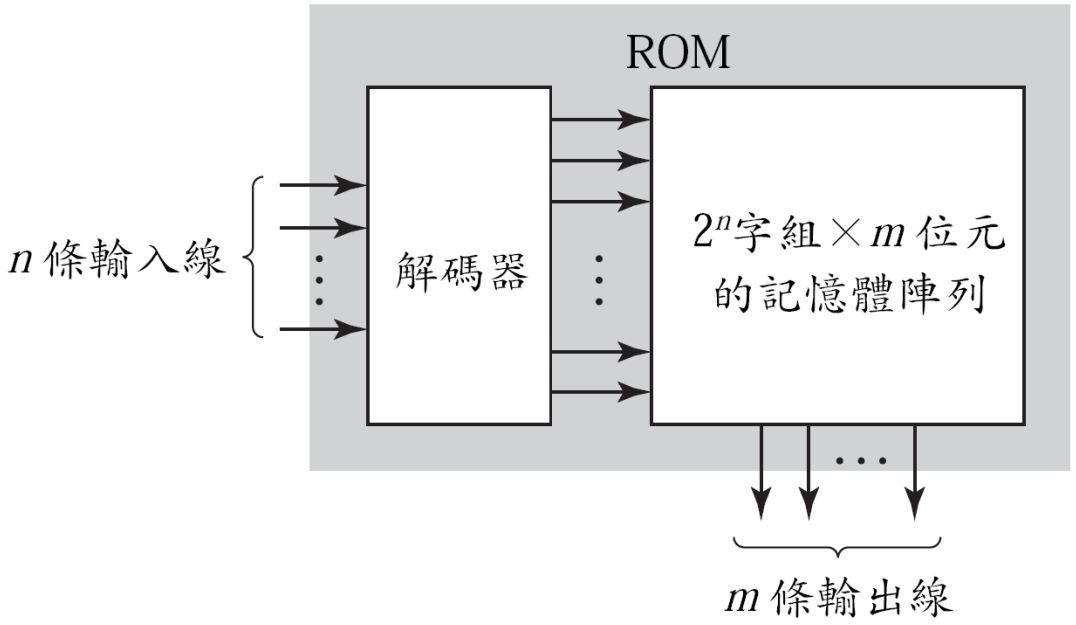


圖 8-23 基本 ROM 的結構

8.5 唯讀記憶體

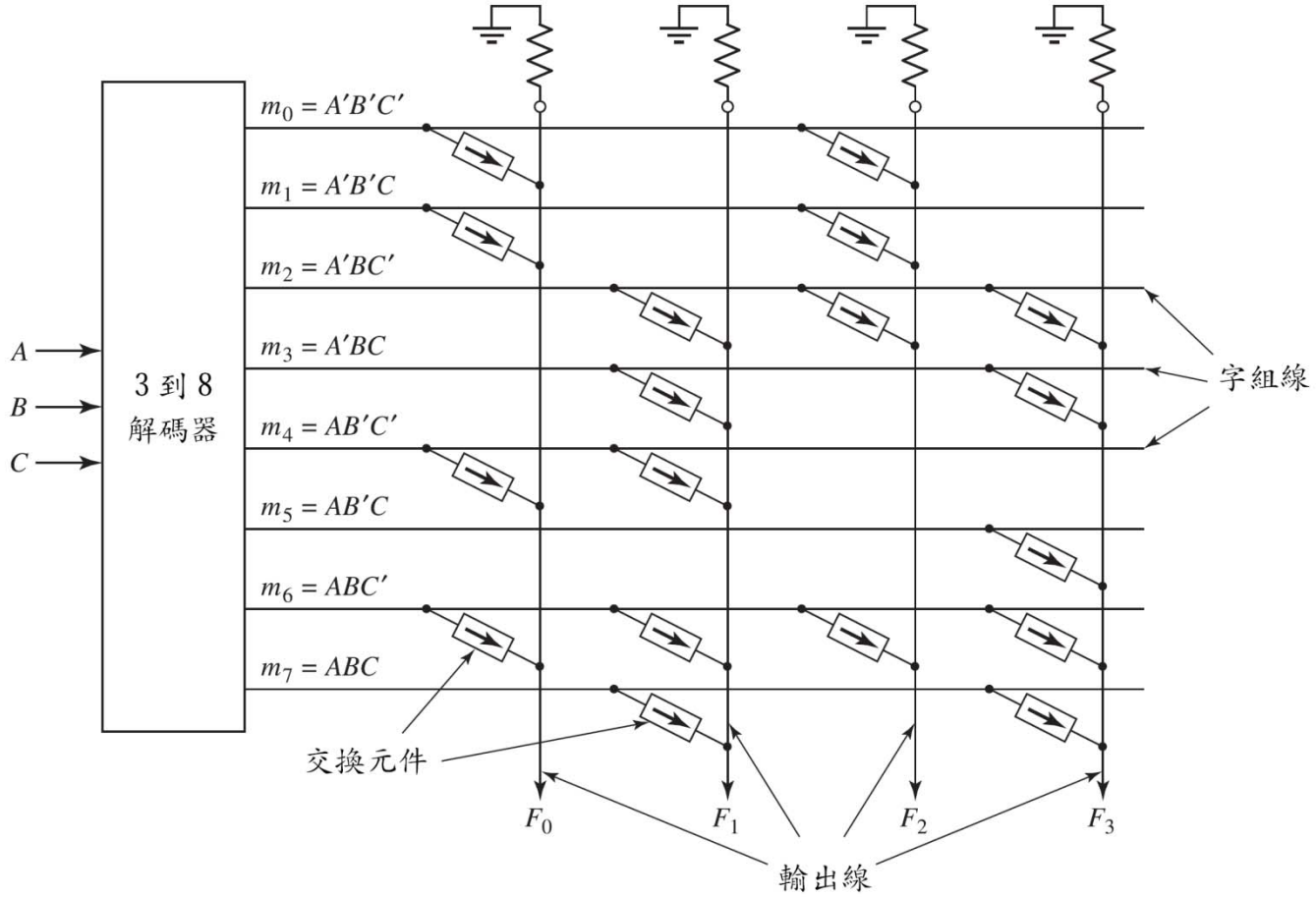


圖 8-24 一個 8 字組×4 位元的 ROM

8.5 唯讀記憶體

❖ 在圖8-24 的ROM可以產生下列函數：

$$F_0 = \Sigma m(0,\ 1,\ 4,\ 6) = A'B' + AC'$$
$$F_1 = \Sigma m(2,\ 3,\ 4,\ 6,\ 7) = AB + AC'$$
$$F_2 = \Sigma m(0,\ 1,\ 2,\ 6) = A'B' + BC'$$
$$F_3 = \Sigma m(2,\ 3,\ 5,\ 6,\ 7) = AC + B \qquad (8\text{-}7)$$

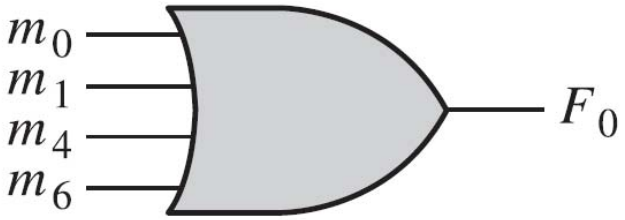


圖 8-25 F_0 的等效 OR 閘

8.5 唯讀記憶體

- 實現一個將4位元二進位數轉換為十六進位數數元且輸出是7位元ASCII 碼的碼轉換器。

輸入				十六進位	十六進位數元之 ASCII 碼						
W	X	Y	Z	數元	A_6	A_5	A_4	A_3	A_2	A_1	A_0
0	0	0	0	0	0	1	1	0	0	0	0
0	0	0	1	1	0	1	1	0	0	0	1
0	0	1	0	2	0	1	1	0	0	0	0
0	0	1	1	3	0	1	1	0	0	0	1
0	1	0	0	4	0	1	1	0	0	0	0
0	1	0	1	5	0	1	1	0	0	0	0
0	1	1	0	6	0	1	1	0	0	0	0
0	1	1	1	7	0	1	1	0	0	0	1
1	0	0	0	8	0	1	1	0	0	0	0
1	0	0	1	9	0	1	1	1	0	0	1
1	0	1	0	A	1	0	0	0	0	0	1
1	0	1	1	B	1	0	0	0	0	1	0
1	1	0	0	C	1	0	0	0	0	1	1
1	1	0	1	D	1	0	0	0	1	0	0
1	1	1	0	E	1	0	0	0	1	0	1
1	1	1	1	F	1	0	0	0	1	1	0

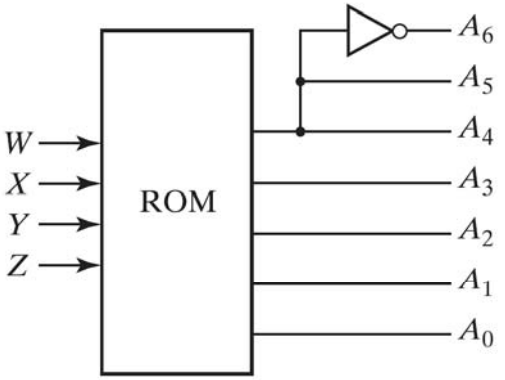


圖 8-26 十六進位到 ASCII 碼轉換器

8.5 唯讀記憶體

- 因為$A_5 = A_4$且$A_6 = A'_4$，ROM只需要五個輸出，因為有四條位址線，所以ROM的大小為16字組 x 5位元。

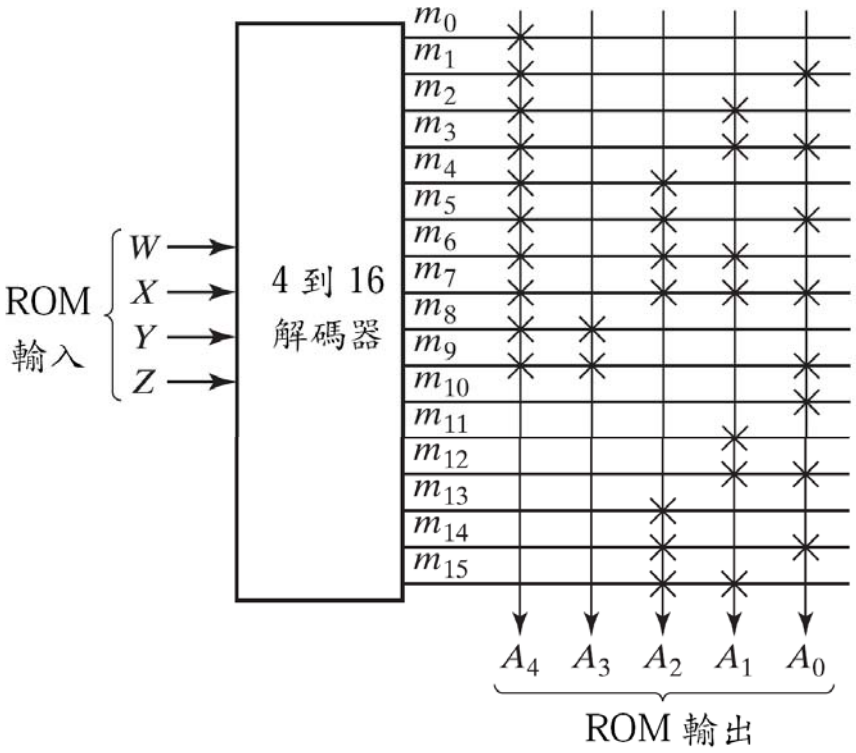


圖 8-27 以 ROM 實現的碼轉換器

8.5 唯讀記憶體

- 三種基本形式的ROM ：
  - **罩幕可程式規劃唯讀記憶體**（mask-programmable ROM）
  - **可程式規劃唯讀記憶體**（programmable ROM，通常稱為PROM）
  - **電氣可擦除可程式規劃唯讀記憶體**（electrically erasable programmable ROM, EEPROM）：利用一個特殊的電荷—儲存機構來致能或禁能在記憶體陣列中的交換元件，一個PROM燒錄器可以提供適當的電壓脈衝將電荷儲存在記憶體陣列的位置，以這種方式儲存的資料通常保持永久，直到它被擦除。

8.5 唯讀記憶體

- **快閃**（flash）記憶體類似EEPROM，差別的是它們使用不同的電荷－儲存機構，它們通常建立在可規劃及擦除能力，所以資料可被寫到放在一個電路裡面的快閃記憶體中，而不需要用一個分開的燒錄器。

8.6 可程式規劃邏輯元件

- **可程式規劃邏輯元件**（programmable logic device, PLD）是一個可被程式化用來提供很多不同邏輯功能的數位積體電路的一般名稱。
- **可程式規劃邏輯陣列**（programmable logic array, PLA）
  - 一個具有*n*個輸入和*m*個輸出的PLA（圖8-28）可以實現*n*變數之*m*個函數。

8.6 可程式規劃邏輯元件

❖ 可程式規劃邏輯陣列結構

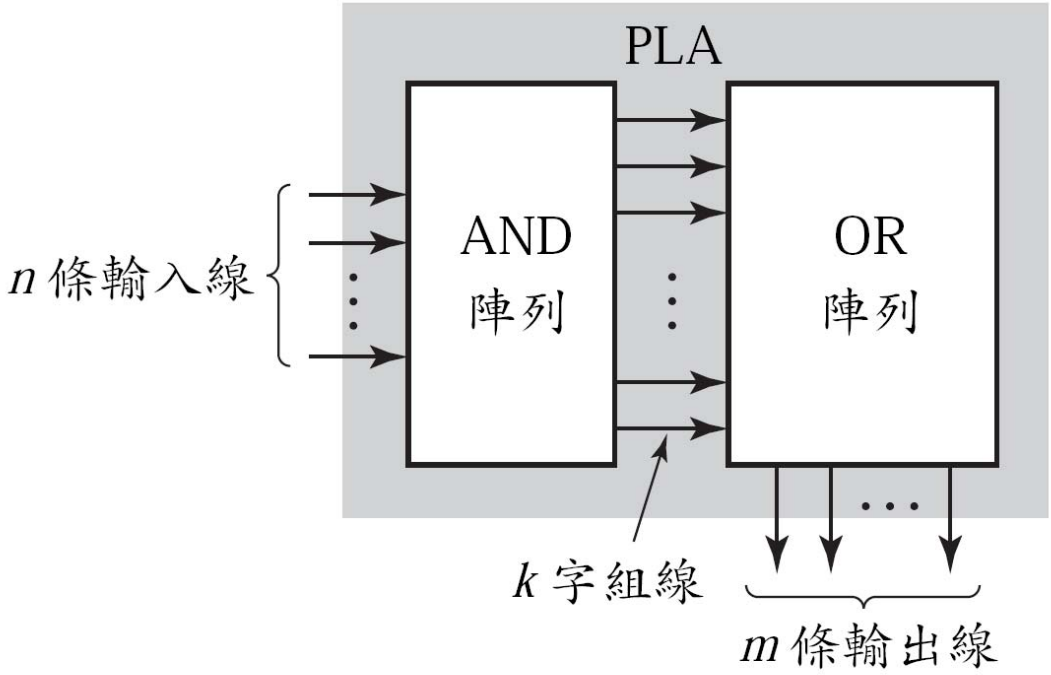


圖 8-28　可程式規劃邏輯陣列結構

8.6 可程式規劃邏輯元件

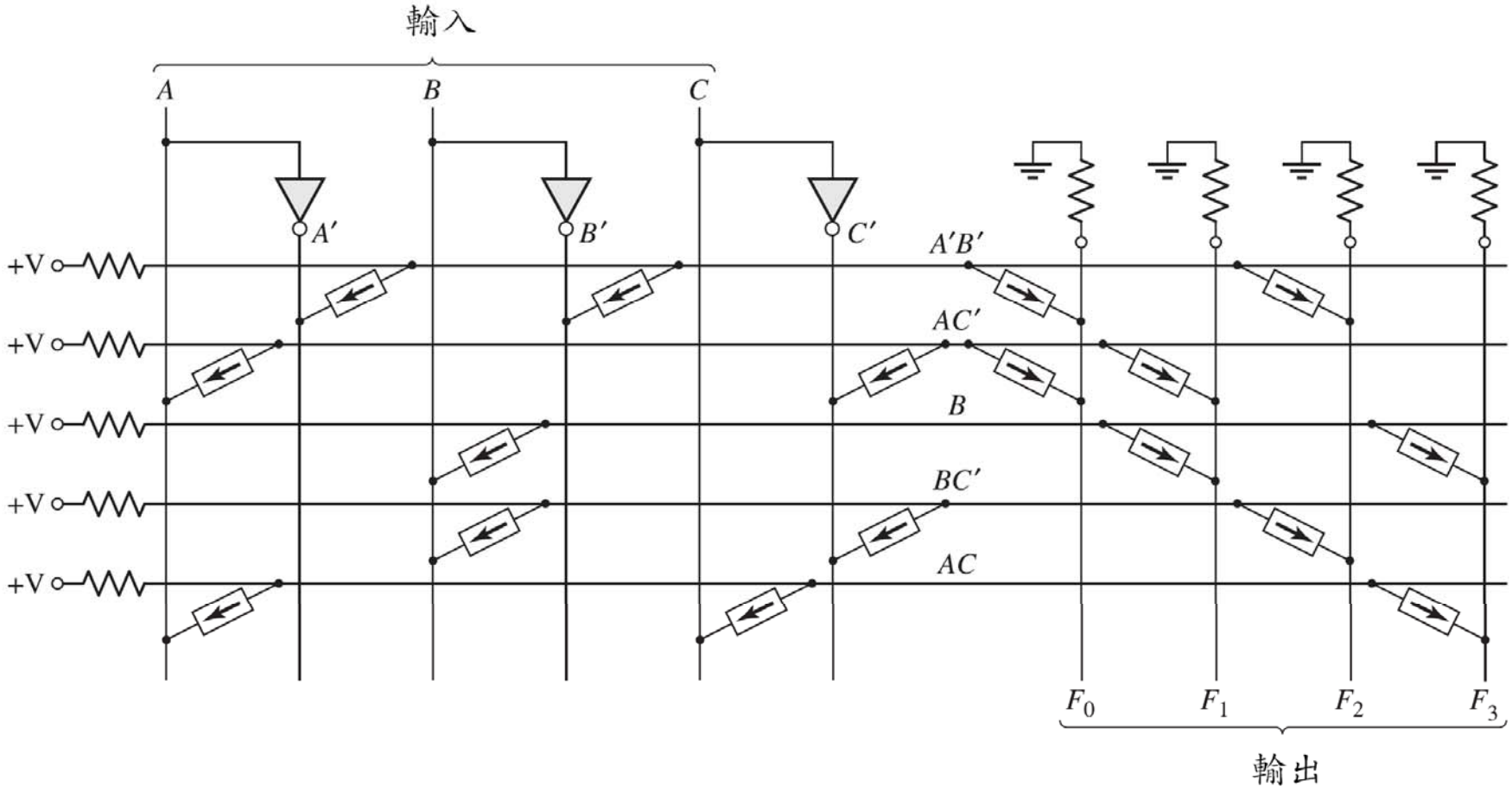


圖 8-29 具有三個輸入、五個積項及四個輸出的 PLA

8.6 可程式規劃邏輯元件

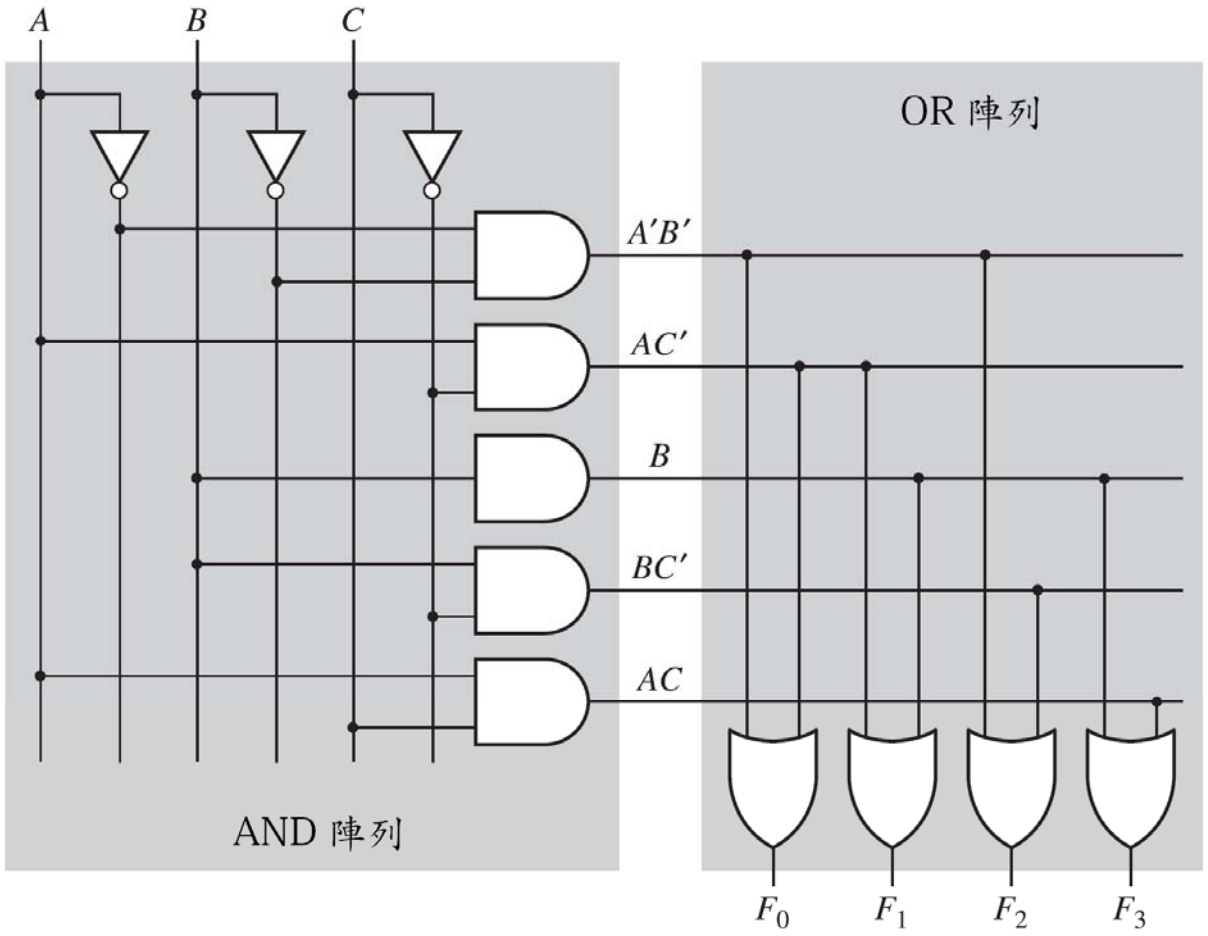


圖 8-30 與圖 8-29 等效的 AND-OR 陣列

8.6 可程式規劃邏輯元件

表 8-1　圖 8-29 之 PLA 表

積項	輸入 A B C	輸出 F_0 F_1 F_2 F_3
$A'B'$	0 0 –	1 0 1 0
AC'	1 – 0	1 1 0 0
B	– 1 –	0 1 0 1
BC'	– 1 0	0 0 1 0
AC	1 – 1	0 0 0 1

$F_0 = A'B' + AC'$

$F_1 = AC' + B$

$F_2 = A'B' + BC'$

$F_3 = B + AC$

8.6 可程式規劃邏輯元件

- PLA 的內容可以由PLA 表來指定，表8-1指定的是圖8-29的PLA。
- 表中的輸入欄指定積項，符號0、1 和 – 分別代表一個變數是補數形式、非補數形式或未出現在相對應的積項中。
- 表中的輸出欄指出哪個積項出現在哪個輸出函數中。1 或0 代表與其相對應的積項出現或未出現在相對應的函數中。
- 因此，由表8-1 的第1列可以看出$A'B'$ 項出現在輸出函數F_0 和F_2 中。且由第2列可知AC' 項出現在F_0 和F_1 中。

8.6 可程式規劃邏輯元件

- 使用PLA實現(6-25) 式。利用(6-25b) 式所給的最簡多重輸出的解，我們可以建立圖8-31(a) 的PLA 表，其中每一列代表每個不同的積項。圖8-31(b) 表示相對應之PLA 結構，其中有四個輸入、六個積項及三個輸出。在字組線和輸入線或輸出線交點上的黑點表示在此陣列有交換元件的存在。

8.6 可程式規劃邏輯元件

❖建立PLA 表

$a\ b\ c\ d$	$f_1\ f_2\ f_3$
0 1 – 1	1 1 0
1 1 – 1	1 0 1
1 0 0 –	1 0 1
– 0 1 –	1 0 0
– – 1 –	0 1 0
– 1 1 –	0 0 1

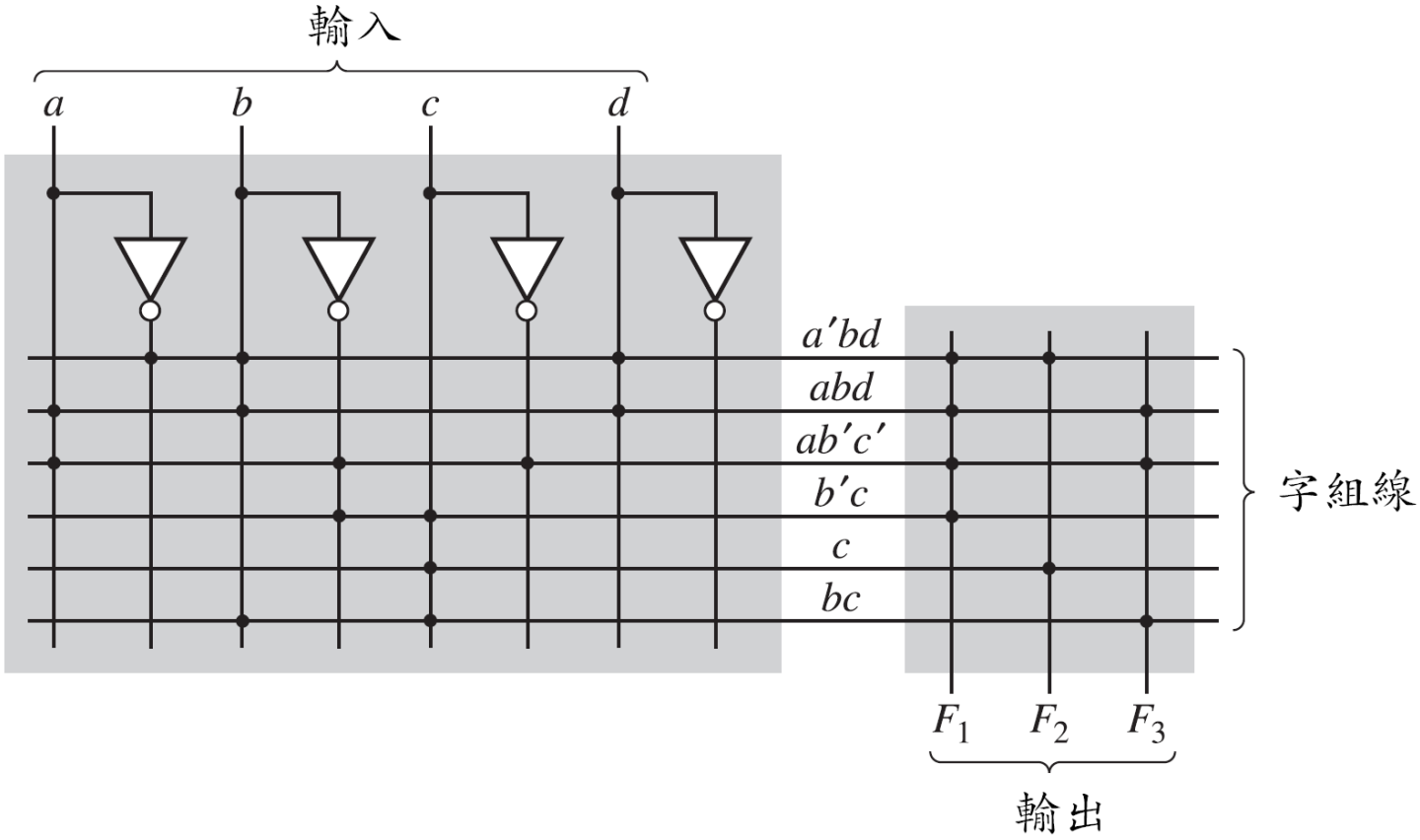
(a) PLA 表

圖 8-31 (6-25b) 式之 PLA 電路

8.6 可程式規劃邏輯元件

- 若$abcd = 0001$，則沒有選擇任一列；且所有f_i為0。若$abcd = 1001$，則只有第3 列被選出，且$f_1f_2f_3 = 101$；若$abcd = 0111$，則第1、5和6列被選出，因此$f_1 = 1 + 0 + 0 = 1$，$f_2 = 1 + 1 + 0 = 1$且$f_3 = 0 + 0 + 1 = 1$。

8.6 可程式規劃邏輯元件

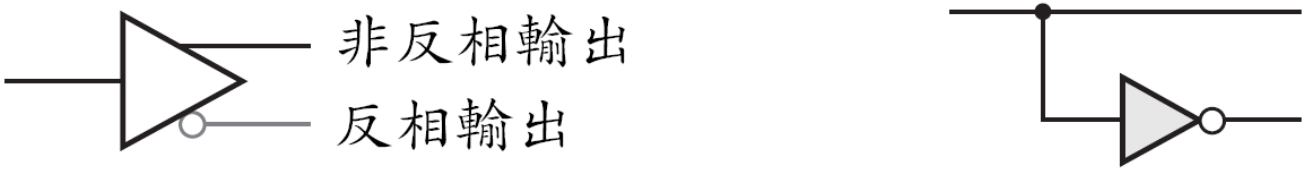


(b) PLA 結構

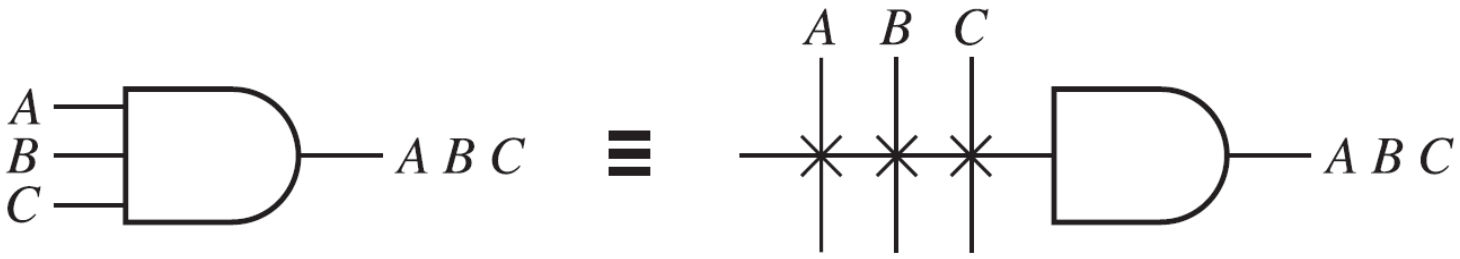
圖 8-31 (6-25b) 式之 PLA 電路

8.6 可程式規劃邏輯元件

- **可程式規劃陣列邏輯**（Programmable Array Logic, PAL）：AND 陣列可以作程式規劃，而OR 陣列則是固定的。
- **緩衝器**（buffer）：

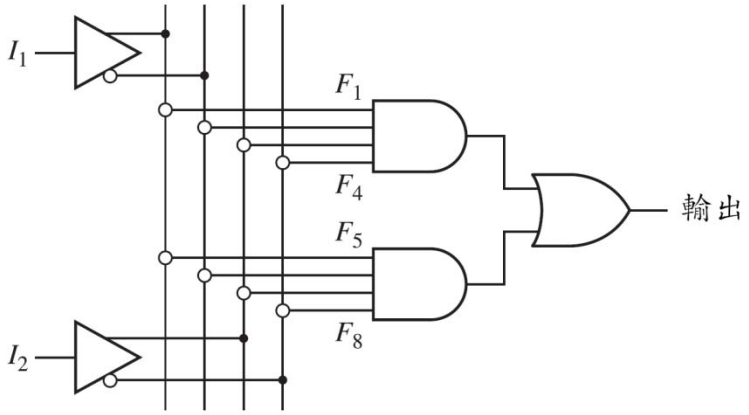


- 在PAL中與AND閘輸入端的連接用x來表示：

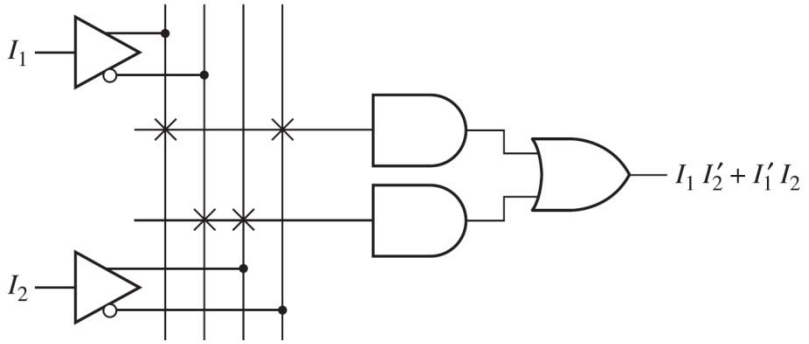


8.6 可程式規劃邏輯元件

❖ 舉例來說：



(a) 未程式規劃



(b) 已程式規劃

圖 8-32 PAL 片斷

8.6 可程式規劃邏輯元件

- 舉一個規劃PAL應用的例子，我們要實現一個全加法器，全加法器的邏輯方程式為：

$$\text{Sum} = X'Y'C_{\text{in}} + X'YC'_{\text{in}} + XY'C'_{\text{in}} + XYC_{\text{in}}$$

$$C_{\text{out}} = XC_{\text{in}} + YC_{\text{in}} + XY$$

8.6 可程式規劃邏輯元件

- 圖8-33所示為每一個OR 閘被四個AND閘所推動的PAL片斷。

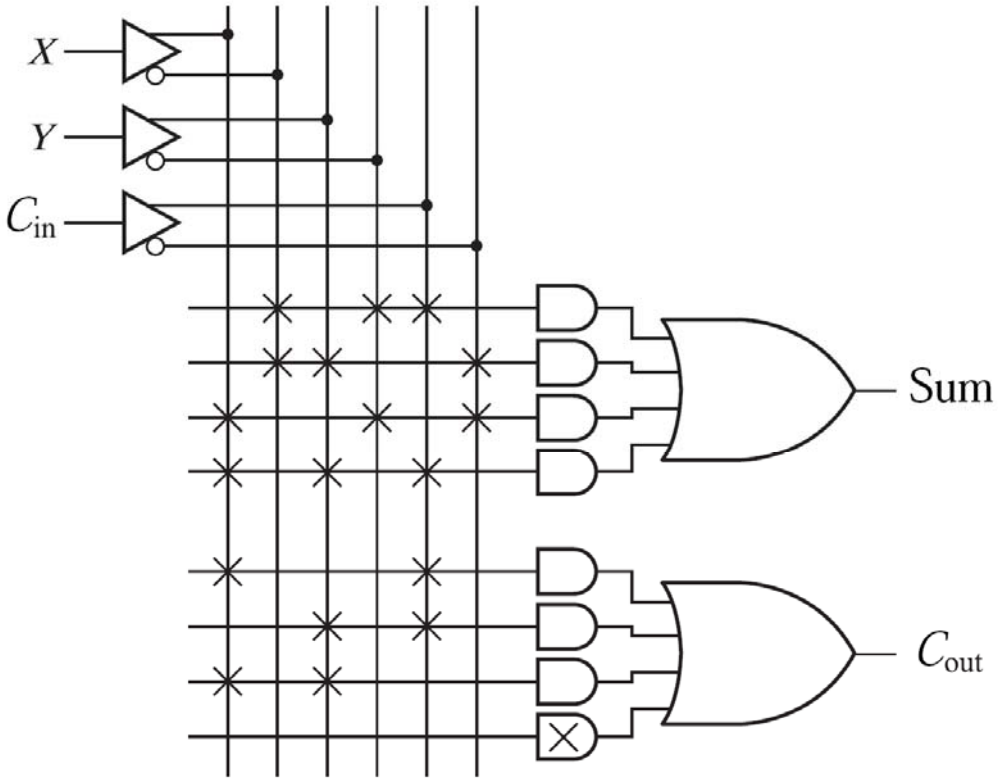


圖 8-33 利用 PAL 實現全加法器的電路

8.7 複雜的可程式規劃邏輯元件

❖ 複雜的可程式規劃邏輯元件（CPLDs）：

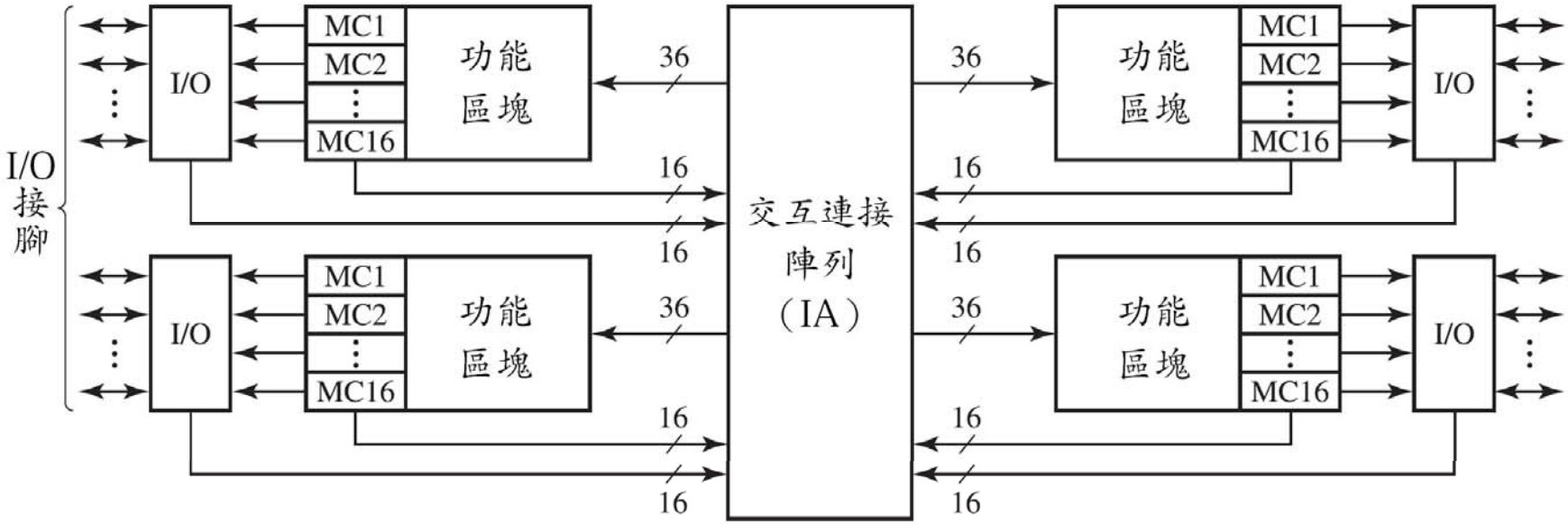


圖 8-34 Xilinx XCR3064XL CPLD 的架構

（圖及文字屬於 Xilinx 公司所有。）

8.7 複雜的可程式規劃邏輯元件

- 這個CPLD 有四個功能區塊，且每個區塊有16 個相關聯的巨集晶胞（MC1、MC2、……）。每一個功能區塊是一個可程式規劃的AND-OR 陣列，也就是形成PLA的結構，每一個巨集晶胞包含一個正反器及多工器，它們由功能區塊連線信號到輸入—輸出（I/O）區塊或是連線到交互連接陣列（IA），IA 由巨集晶胞或I/O 區塊的輸出選擇信號，並將它們連接回到功能區塊的輸入，因此在一個功能區塊所產生的信號可以被用來作為任何其他功能區塊的輸入，I/O區塊在IC上的雙向I/O 接腳以及CPLD內部之間提供一個介面。

8.7 複雜的可程式規劃邏輯元件

- 圖8-35所示為在PLA 中如何產生一個信號經由一個巨集晶胞連線到一個I/O 接腳。

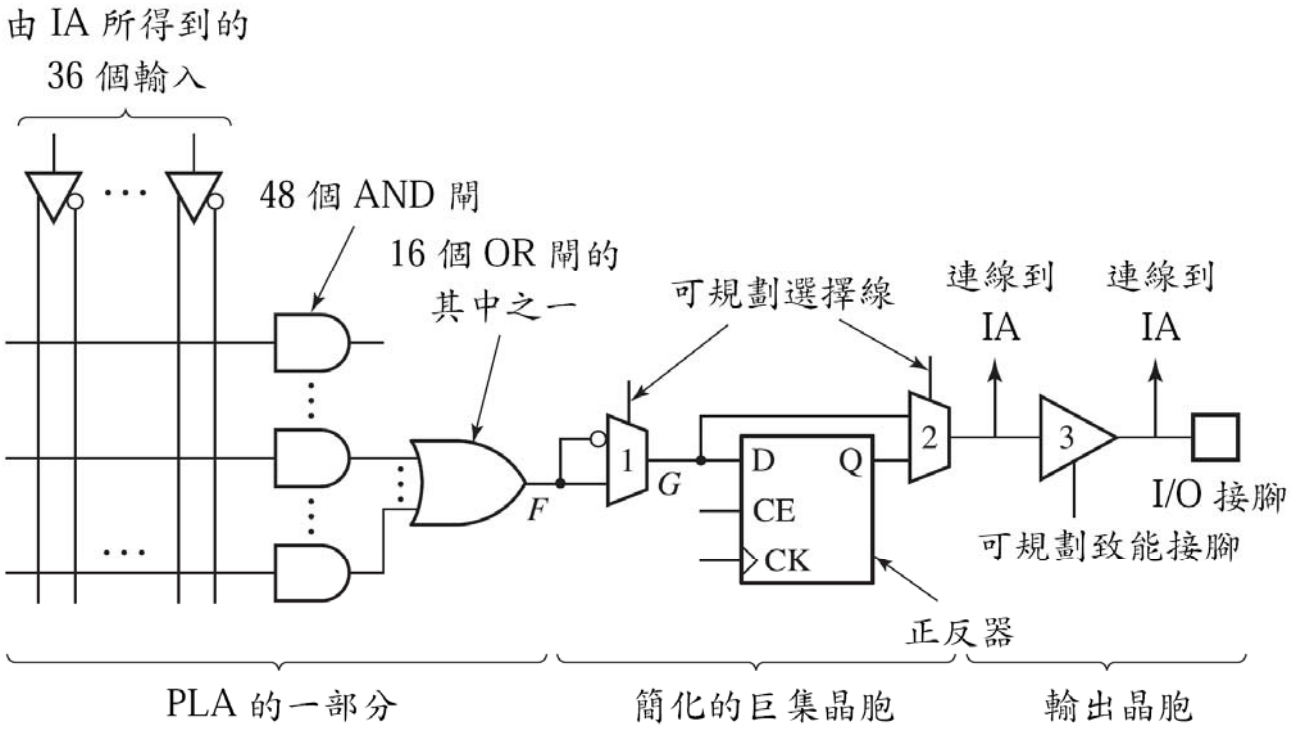


圖 8-35 CPLD 功能方塊及巨集晶胞（XCR3064XL 的簡化版）

8.8 場效可程式規劃閘陣列

- **場效可程式規劃閘陣列**（field-programmable gate array, FPGA）在組合邏輯設計的應用，FPGA是一個包含具有可程式規劃交互連接之相同的邏輯晶胞陣列的IC，使用者可以藉由實現每個邏輯晶胞以及晶胞之間的連接來程式規劃函數
- FPGA的內部包含邏輯晶胞陣列，也稱為**配置邏輯區塊**（configurable logic block, CLB），CLB陣列被一環的輸入－輸出的介面區塊所環繞，這些I/O區塊連接CLB的信號到IC的接腳，在CLB之間的空間用來作為介於CLB輸出及輸入間繞線的連接。

8.8 場效可程式規劃閘陣列

❖ 一個典型FPGA的佈局：

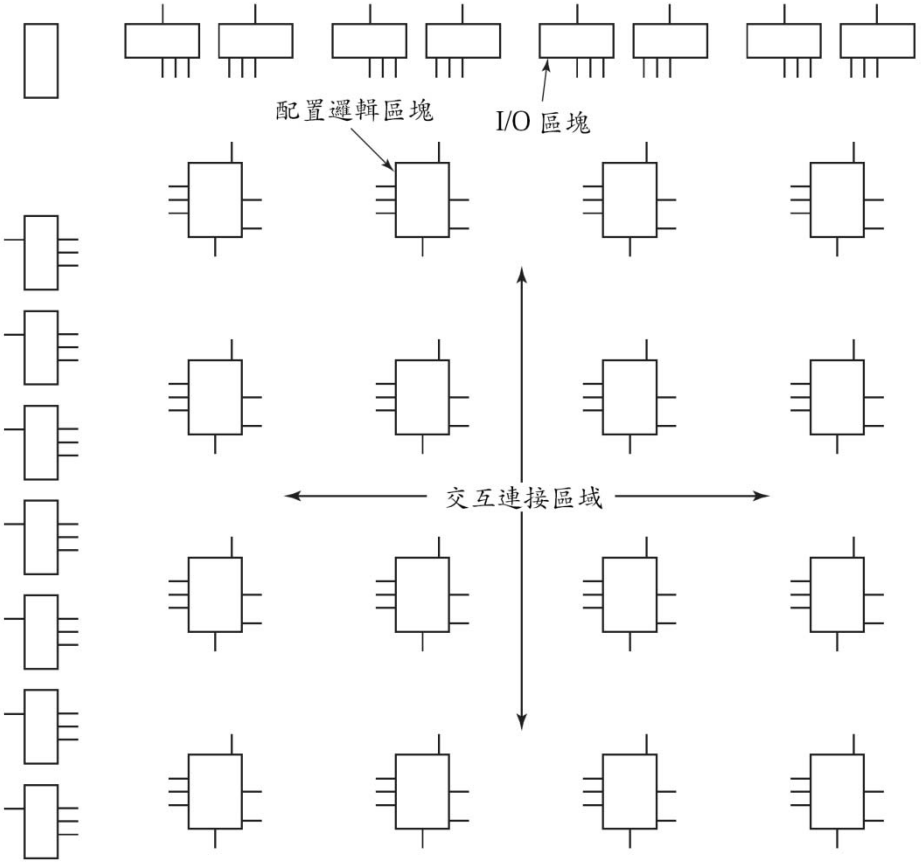


圖 8-36 一個典型 FPGA 的佈局

8.8 場效可程式規劃閘陣列

- 圖8-37所示為一個CLB的簡化版，這個CLB包含兩個函數產生器、兩個正反器及多個多工器，以提供在CLB 內部的繞線信號。

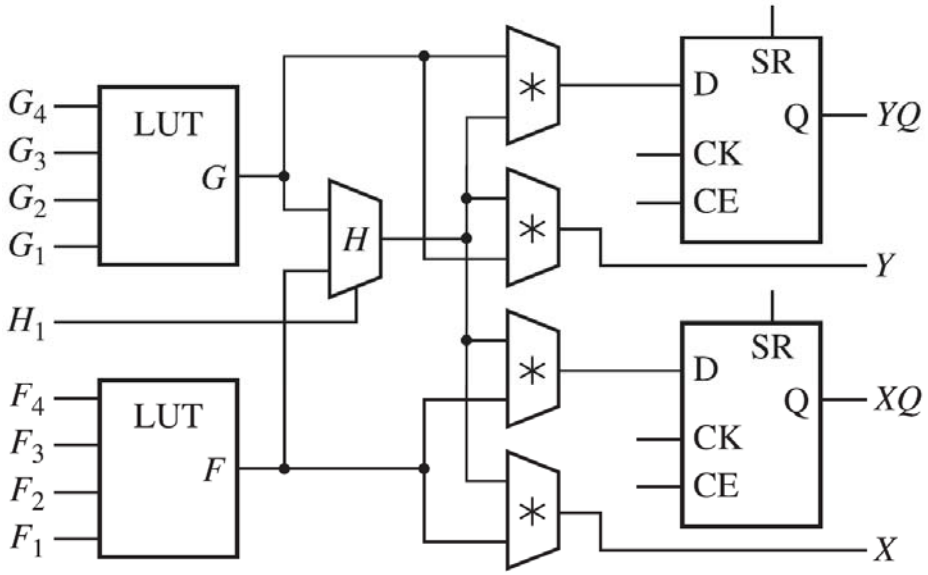


圖 8-37 簡化的配置邏輯區塊

8.8 場效可程式規劃閘陣列

- 一個四輸入的**對照表**（lookup table, LUT）本質上是一個具有16個1位元字組的再規劃ROM，此ROM儲存要被產生之函數的真值表。

a	b	c	d	F
0	0	0	0	0
0	0	0	1	1
⋮				⋮
1	1	1	1	1

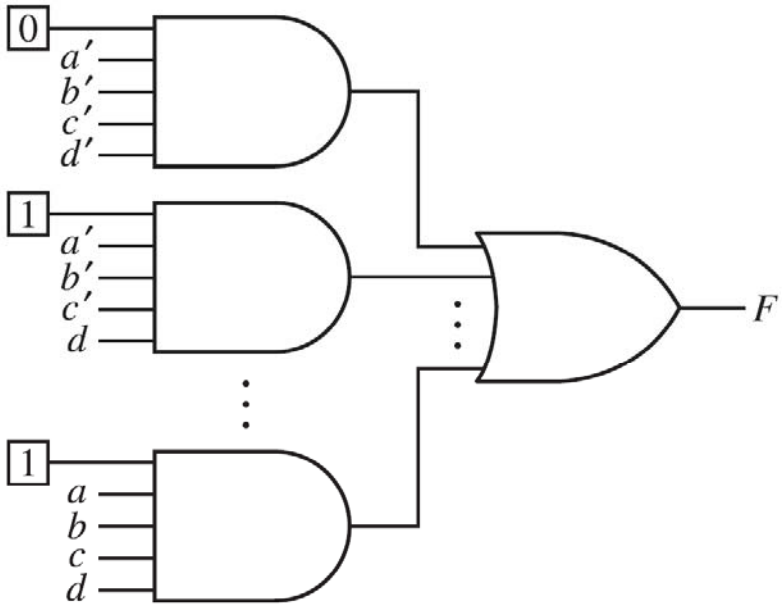


圖 8-38 對照表的實現

交換函數的分解

- 為了利用4 變數函數產生器來實現一個超過四個變數的交換函數，則函數必須被分解成只具有四個變數的子函數，一種分解的方法是利用Shannon的展開定理，首先我們經由一個具有變數a、b、c和d的函數來對變數a作展開來說明這個定理：

$$f(a,b,c,d) = a' f(0,b,c,d) + af(1,b,c,d) = a' f_0 + af_1 \quad (8\text{-}8)$$

交換函數的分解

❖ 一個應用(8-8) 式的例子如下所示：

$$f(a,b,c,d) = c'd' + a'b'c + bcd + ac'$$
$$= a'(c'd' + b'c + bcd) + a(c'd' + bcd + c')$$
$$= a'(c'd' + b'c + cd) + a(c' + bd) = a'\,f_0 + af_1 \qquad (8\text{-}9)$$

交換函數的分解

- 展開式也可以利用真值表或卡諾圖來完成，圖8-39所示為(8-9) 式的卡諾圖。

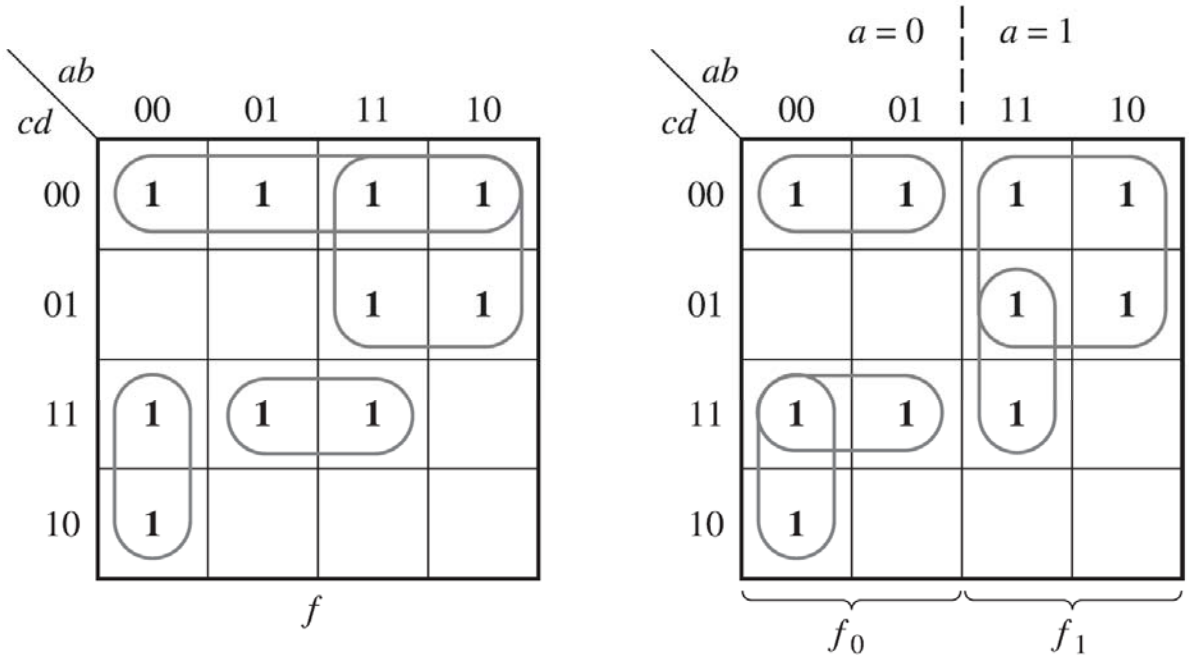


圖 8-39 利用卡諾圖作函數展開

$$f_0 = c'd' + b'c + cd \qquad f_1 = c' + bd$$

交換函數的分解

- 展開一個有關變數x_i的n 變數函數之Shannon 展開定理的一般化形式為：

$$
\begin{aligned}
&f(x_1, x_2, \ldots, x_{i-1}, x_i, x_{i+1}, \ldots x_n) \\
&= x'_i \, f(x_1, x_2, \ldots, x_{i-1}, 0, x_{i+1}, \ldots x_n) + x_i \, f(x_1, x_2, \ldots, x_{i-1}, 1, x_{i+1}, \ldots x_n) \\
&= x'_i \, f_0 + x_i \, f_1 \qquad (8\text{-}10)
\end{aligned}
$$

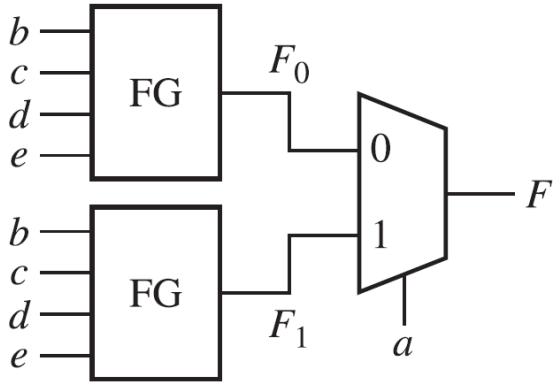
- 在此，f_0是在原始函數中令x_i為0所得到的$(n-1)$ 變數的函數，且f_1 是在原始函數中令x_i為1所得到的$(n-1)$ 變數的函數。

交換函數的分解

- 應用此展開式定理到5變數函數，則可得：

$$f(a,b,c,d,e) = a'f(0,b,c,d,e) + af(1,b,c,d,e) = a'f_0 + af_1 \quad (8\text{-}11)$$

- 這顯示出任何5 變數函數可以利用兩個4 變數函數產生器以及一個2 對1 多工器（圖8-40(a)）來實現。



(a) 5 變數函數

圖 8-40 具有函數產生器之 5 和 6 變數函數的實現

交換函數的分解

- 要利用4 變數函數產生器來實現一個6 變數函數，我們應用展開定理兩次：

$$G(a,b,c,d,e,f) = a'G(0,b,c,d,e,f) + aG(1,b,c,d,e,f) = a'G_0 + aG_1$$

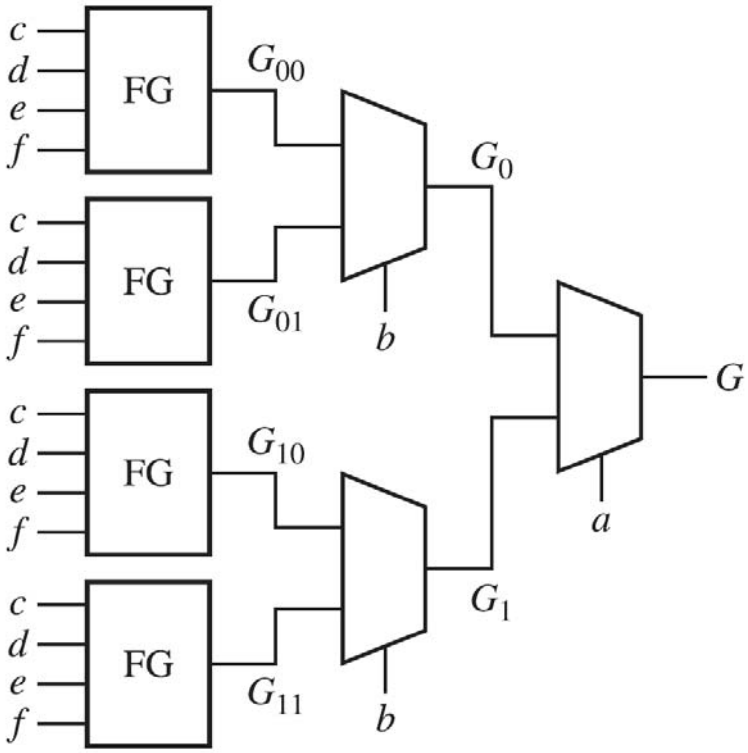
$$G_0 = b'G(0,0,c,d,e,f) + bG(0,1,c,d,e,f) = b'G_{00} + bG_{01}$$

$$G_1 = b'G(1,0,c,d,e,f) + bG(1,1,c,d,e,f) = b'G_{10} + bG_{11}$$

- 因為G_{00}、G_{01}、G_{10}和G_{11}都是4變數函數，所以我們可以利用四個4變數函數產生器，以及三個2 對1 多工器來實現任何6 變數函數。

交換函數的分解

$$G(a,b,c,d,e,f) = a'b'G_{00} + a'bG_{01} + ab'G_{10} + abG_{11} \quad (8\text{-}12)$$



(b) 6 變數函數

圖 8-40 具有函數產生器之 5 和 6 變數函數的實現