正確學會 改訂新版 ディジタル回路と Verilog HDL

Verilog 的16堂課

序章

邏輯代數

本章重點

- 0.1 基本邏輯運算
- 0.2 基本規則與定律
- 0.3 求出邏輯式
- 0.4 邏輯式的化簡
- 0.5 延伸學習

0.1 基本邏輯運算

- 電路設計的重點在用最少的元件實現想要的功能

0.1.1 邏輯與命題

- 判斷一個句子內容的真假, 稱為命題 (proposition)
- 結合兩種以上的命題稱為複合命題 (compound proposition)

0.1.2 布林代數

- 忽略數字的大小，只判斷屬於真或偽的命題，稱為 **2 元邏輯** (binary logic)
- 根據 2 元邏輯來處理命題的手法稱為 **2 元邏輯運算**
- 2 元邏輯運算的基本形式
  - 邏輯積 (AND)
  - 邏輯和 (OR)
  - 邏輯否定 (NOT)
- 以上述 3 種邏輯運算構成的代數稱為**布林代數** (Boolean algebra)

0.1.3 複合命題與邏輯運算

- 複合命題

  那個人是男性, 且 18 歲以上

- 將命題當成 A、B，複合命題當作 Y，可改成

  A AND B

- 複合命題 'Y' 可稱為邏輯函數 (logical function)
- 各個命題 'A', 'B' 稱為邏輯變數 (logical variable)
- 考慮邏輯運算的結果來決定每個命題的動作稱為命題邏輯 (proposition logic)

0.1.4 邏輯積 (AND)

- 意指：構成複合命題的各命題全為真的時候
  → 結果為真, 其他都為偽
- 邏輯積 (AND) 在布林代數裡面以 '・' 表示
- 其邏輯式：
  - Y = A・B

0.1.5 邏輯和 (OR)

- 意指：構成複合命題的各命題只要其中一個為真 → 結果就為真
  各命題全為偽的時候 → 結果就為偽
- 在布林代數裡面以 '**+**' 表示
- 其邏輯式：
  - Y = A + B

0.1.6 邏輯否定 (NOT)

- 意指：
  結果為真→判斷為偽
  結果為偽→判斷為真
- 在布林代數裡面以‘—’或‘’’表示
- '$\overline{A}$' 不讀做「not A」而讀做「A bar」

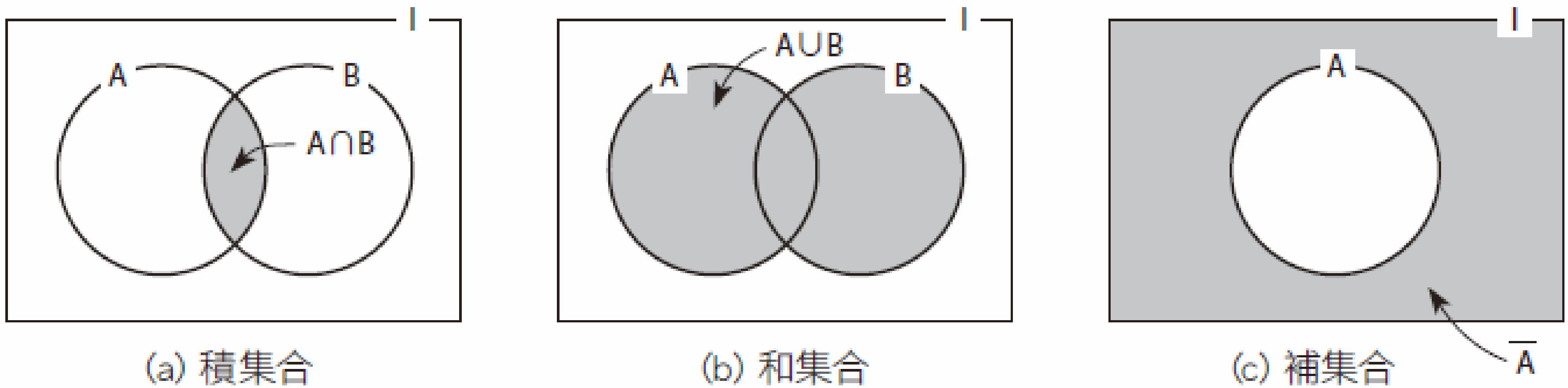
- 邏輯積(AND) → '·'
- 邏輯和(OR) → '+'
- 邏輯否定(NOT) → '¯'

0.2 基本規則與定律

- 數位電路設計就是用「AND」,「OR」,「NOT」三個基本邏輯來表現電路的動作
- 把複雜的邏輯變換成比較簡單的邏輯稱為化簡

0.2.1 范氏圖

- 范氏圖 (Venn diagram) 是根據 AND 或 OR 這種邏輯運算的集合理論, 用圖示的方式來做視覺上表現的方法
- 集合 (set) 就是把符合某種條件的「東西」全部集合在一起, 相當於布林代數的「變數」
- 集合裡面的每個「東西」稱作元素(element)
- 沒有元素的集合稱之為空集合 (empty set)



(a) 積集合　(b) 和集合　(c) 補集合

- 積集合： 當有兩個以上的集合時, 符合所有集合的元素所在的集合稱之為積集合(交集)。符號為 ∩, 在布林代數裡面等同於 AND。
- 和集合： 當有兩個以上的集合時, 符合至少一個集合的元素所在的集合稱之為和集合(聯集)。符號為∪, 在布林代數裡面等同於 OR。
- 補集合： 不屬於任何一個集合的元素所在的集合稱之為補集合。在布林代數裡面等同於 NOT。

0.2.2 布林代數的基本規則

交換律	$A \cdot B = B \cdot A$ $A + B = B + A$	分配律	$A \cdot (B + C) = A \cdot B + A \cdot C$ $A + B \cdot C = (A + B) \cdot (A + C)$
邊界定律	$A \cdot 1 = A$ $A \cdot 0 = 0$ $A + 1 = 1$ $A + 0 = A$	自補律 互補律	$\overline{\overline{A}} = A$ $A \cdot \overline{A} = 0$ $A + \overline{A} = 1$
結合律	$(A \cdot B) \cdot C = A \cdot (B \cdot C)$ $(A + B) + C = A + (B + C)$	等冪律	$A \cdot A = A$ $A + A = A$

0.2.3 笛摩根定律

$$\overline{(A + B)} = \overline{A} \cdot \overline{B}$$

$$\overline{(A \cdot B)} = \overline{A} + \overline{B}$$

- 把某邏輯式全部取否定, 則各變數都變成否定
- OR 與 AND 做交換之後仍跟原式相等

0.3 求出邏輯式

- 導出邏輯式的步驟：
  - (1) 做成可以表示電路動作的真值表
  - (2) 根據真值表導出邏輯式
  - (3) 把求出的邏輯式化簡

0.3.1 真值與真值表

- 在邏輯過程,我們把 '真' 當作 '1', '偽' 當作 '0' ,此時的 '1'/'0' 稱為**真值**
- 使用這兩個真值整理出各種邏輯結果的表格則稱為真值表 (truth table)
- 當輸入有 n 個的時候組合就有 2^n 種

做成真值表的步驟

我們用三個輸入變數與一個輸出作為例子來說明

- 決定真值表的行數與列數(橫向為列,縱向為行)
  - 3 個輸入、1 個輸出, 因此有 4 欄(行)
  - 輸入變數的組合有 2^3 (8) 種, 加上變數欄位名稱共9列
- 將各輸入變數的組合填入表中
- 依據輸入組合, 填入輸出值

本書 0-14 頁, 關於行、列的描述誤植, 正確應為 4 行(欄)、9 列

(a)

3 輸入 (n=3)

C	B	A	Y

(b)

C	B	A	Y
0	0	0	
0	0	1	
0	1	0	
0	1	1	
1	0	0	
1	0	1	
1	1	0	
1	1	1	

組合有
2^n 種 = 8 種 (n=3)

0 與 1 一個一個交替填入

0 與 1 兩個兩個交替填入

0 與 1 四個四個交替填入

(c)

C	B	A	Y
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	0
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1

填入輸入組合
的對應輸出值

0.3.2 加法標準型

1. 把輸出Y為‘1'的所有輸入組合挑出來
2. 根據 (1) 的組合，各輸入的值為‘1'的時候維持原狀，為 '0' 的時候就做 NOT 且用 AND 做結合。
3. 把 (2) 的各 AND 項用 OR 做結合。

- $Y=\overline{A}\cdot B\cdot\overline{C}+\overline{A}\cdot\overline{B}\cdot C+\overline{A}\cdot B\cdot C+A\cdot B\cdot C$

組合 1　　組合 2　　組合 3　　組合 4

C	B	A	Y	
0	0	0	0	
0	0	1	0	
0	1	0	1	⋯ 組合 1
0	1	1	0	
1	0	0	1	⋯ 組合 2
1	0	1	0	
1	1	0	1	⋯ 組合 3
1	1	1	1	⋯ 組合 4

0.3.3 乘法標準型

1. 輸出為‘0’的所有輸入組合挑出來。
2. 根據 (1) 的組合, 各輸入值為‘0'的話保持原狀, 為 '1' 的時候就做 NOT 且用 OR 做結合。
3. 把 (2) 的各 OR 項用 AND 做結合。
   - 不過因為在運算的時候, AND 的優 (3) 先順序比 OR 高, 所以要先用括號之後才能跟 AND 做結合。

- $Y = (A + B + C) \cdot (\overline{A} + B + C) \cdot (\overline{A} + \overline{B} + C) \cdot (\overline{A} + B + \overline{C})$

  組合1'　　組合2'　　組合 3'　　組合 4'

C	B	A	Y	
0	0	0	0	… 組合 1'
0	0	1	0	… 組合 2'
0	1	0	1	
0	1	1	0	… 組合 3'
1	0	0	1	
1	0	1	0	… 組合 4'
1	1	0	1	
1	1	1	1	

0.4 邏輯式的化簡

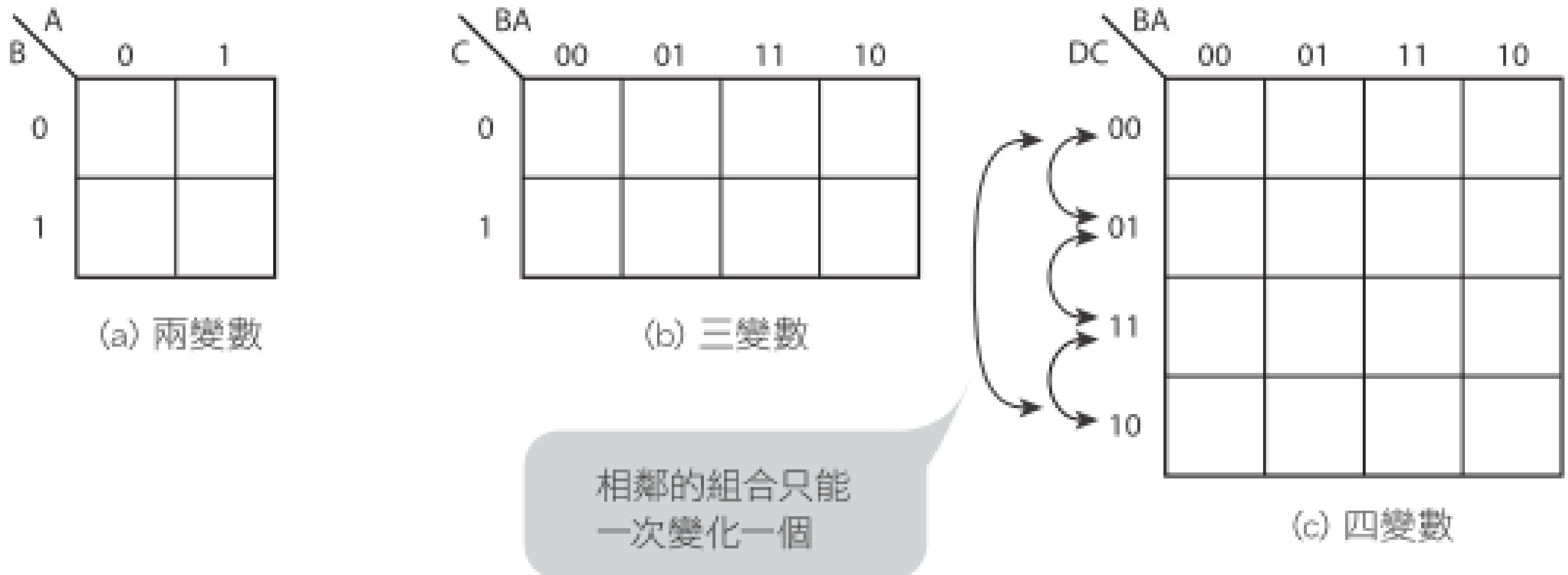
- 前面導出來的邏輯式常有多餘的式子, 要設計出有效率的數位電路,需要做邏輯式的化簡(simplification)

0.4.1 根據基本規則做化簡

- 最基本的就是使用 cut and try 法或稱 trial and error 法
  1. 真值表用加法標準型來導出邏輯式。
  2. 根據分配律將共通項一一化簡。
  3. 等冪律可以無數次複製同樣的 AND 項或 OR 項
  4. 利用互補律與邊界定律將 AND 項的輸入變數減少, 或者把 AND 項整個減去

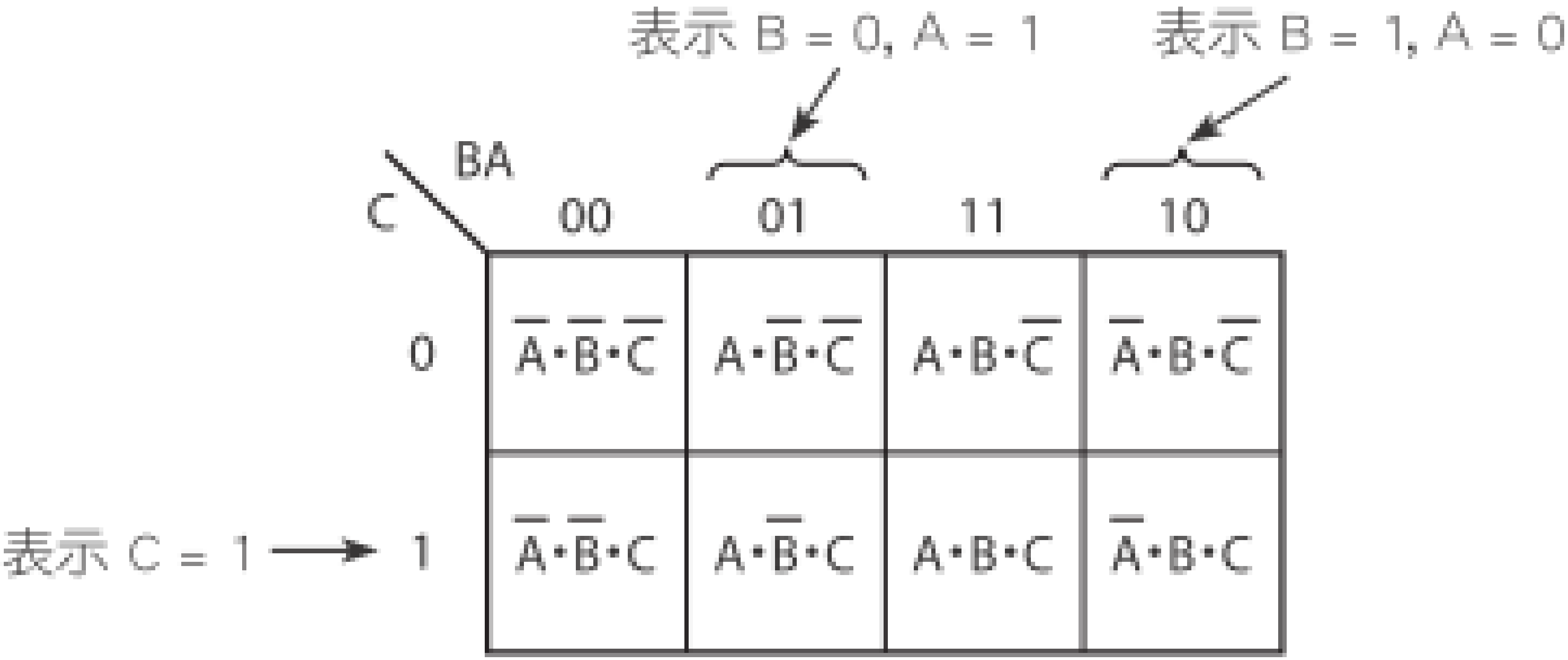
0.4.2 用卡諾圖化簡

- (1)卡諾圖的畫法
  - a 變數有 n 個的時候，會由 2^n 個被稱為 cell 的框框構成。
  - b 卡諾圖的形狀會根據變數的數目而有所改變。
  - c 把變數名稱填到卡諾圖的左上方。
  - d 外框上面與左面填入變數所有可能的值 '1'/'0'。
  - e 左上的 cell 一定要填入各變數都為 '0' 的組合。
  - f 相鄰的所有 cell，變數的狀態 ('1'/'0') 一次只能變動一個。



(a) 兩變數
(b) 三變數
(c) 四變數

- (2) 把真值表的輸出值寫到卡諾圖上
  - a 填到卡諾圖裡面的只需要是輸出為 '1' 的時候。不需要特別填入 '0'。
  - b 當填完卡諾圖的時候, 請對一下真值表輸出為 '1' 的數目是否跟卡諾圖上的 '1' 一樣。
  - c 填入 '1' 的 cell 務必不要搞錯。如果填到錯誤的 cell 裡面的話, 之後求出的邏輯式跟最後畫出來的電路圖就是錯誤的了。


表示 B = 0, A = 1
表示 B = 1, A = 0
BA
C
00
01
11
10
0
1
表示 C = 1
$\overline{A}\cdot\overline{B}\cdot\overline{C}$
$A\cdot\overline{B}\cdot\overline{C}$
$A\cdot B\cdot\overline{C}$
$\overline{A}\cdot B\cdot\overline{C}$
$\overline{A}\cdot\overline{B}\cdot C$
$A\cdot\overline{B}\cdot C$
$A\cdot B\cdot C$
$\overline{A}\cdot B\cdot C$

C	B	A	Y
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	0

(a) 寫好真值表

BA ← 變數名稱

C \ BA				

(b) 劃出格子與寫出變數名稱

C \ BA	00	01	11	10
0				
1				

(c) 寫出各輸入的各種組合

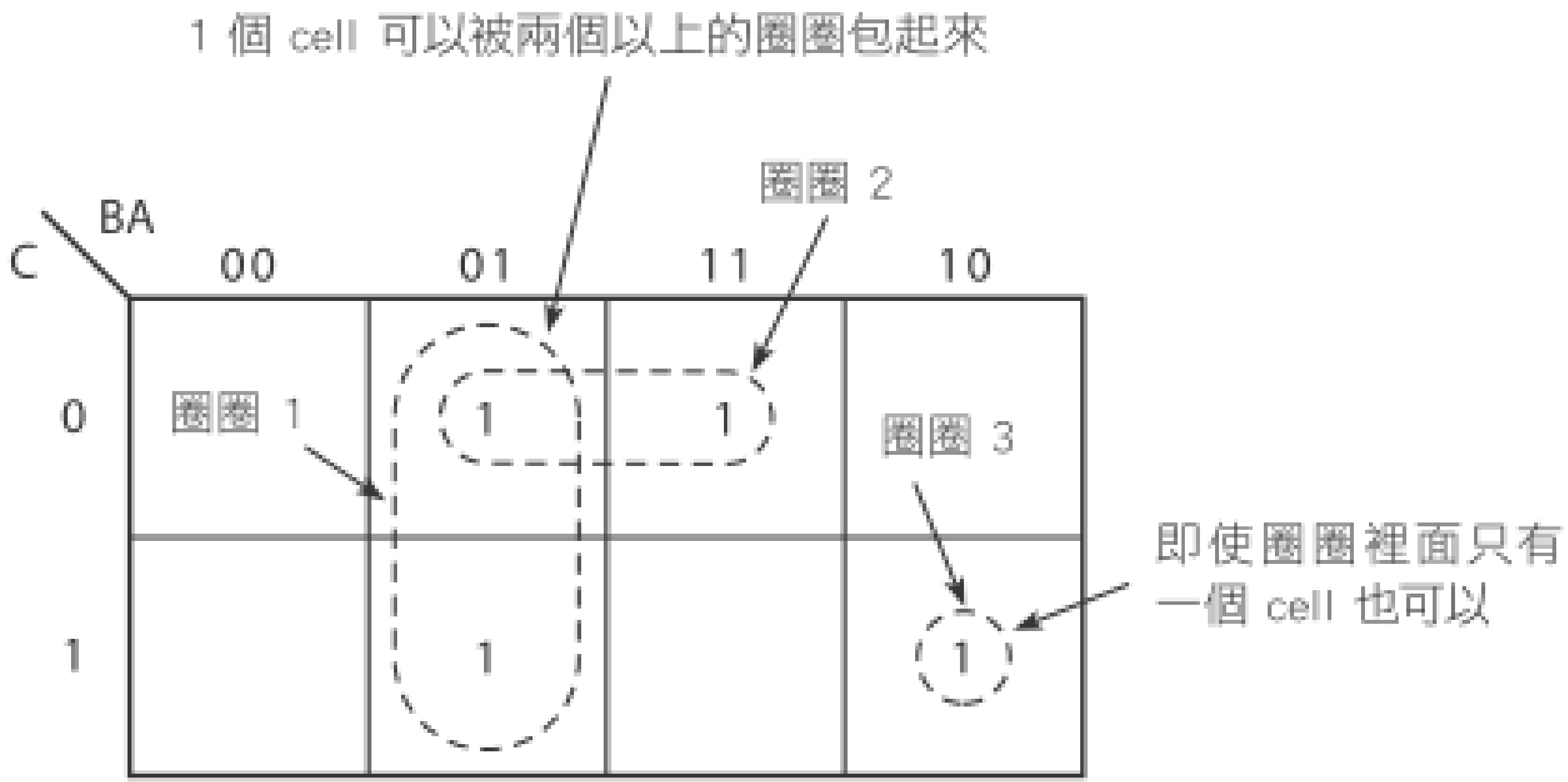
C \ BA	00	01	11	10
0		1	1	
1		1		1

A = 0, B = 1, C = 0 的時候 Y 為 0 所以這邊不用刻意填入 0

A = 0, B = 1, C = 1 的時候 Y 為 1 所以這邊要填入 1

(d) 填入對應輸入組合的輸出結果

- (3) 從卡諾圖導出邏輯式
  - 用圈圈把卡諾圖中有 '1' 的 cell 包起來, 根據各個包起來的部分來求出邏輯式



0.4.3 用列表法化簡

- 卡諾圖在變數增多時，化簡上不容易，改用列表法即使有很多變數，也可以一定步驟進行化簡
- **列表法**又稱為奎因－麥克拉斯基演算法 (Quine-Maclusky method：QM 演算法)

1. 先用加法標準型來變形, 擴充最小項。
2. 觀察 (1) 的邏輯式裡各個最小項裡有包含反向的變數的數目, 根據這個數目來做分組
3. 比較各組與隔壁組之間所有的組合, 根據互補律嘗試減少一個變數
4. 持續進行步驟 3, 直到變數不能再減少為止, 最後沒辦法再做化簡的項就是主項 (prime term)。
5. 主項有可能會重複, 因此需要檢查主項表

6. 把找出來主項的組合用 OR 連在一起，就可以得到最後的邏輯式了。

$$Y = A \cdot \overline{B} \cdot \overline{D} + \overline{A} \cdot C + \overline{C} \cdot D$$

最小項	主項					
	$A\overline{B}\,\overline{D}$	$A\overline{B}\,\overline{C}$	$\overline{B}C\overline{D}$	$\overline{A}C$	$\overline{C}D$	$\overline{A}D$
$A\overline{B}\,\overline{C}\,\overline{D}$	○	○				
$\overline{A}\,\overline{B}C\overline{D}$			○	○		
$\overline{A}\,\overline{B}\,\overline{C}D$					○	○
$A\overline{B}C\overline{D}$	○		○			
$A\overline{B}\,\overline{C}D$		○			○	
$\overline{A}BC\overline{D}$				◎		
$\overline{A}B\overline{C}D$					○	○
$\overline{A}\,\overline{B}CD$				○		○
$AB\overline{C}D$					◎	
$\overline{A}BCD$				○		○

○… 表示該最小項有被該主項包含

◎… 表示在各主項之中, 唯一含有該最小項的主項