CHAPTER

5 組合邏輯電路

5-1 組合邏輯電路分析

5-1-1 組合邏輯電路概論

邏輯電路分為組合邏輯電路 (Combinational Logic Circuit) 與序向邏輯電路 (Sequential Logic Circuit)。組合邏輯電路是由基本邏輯閘組成，且輸出是所有輸入的組合形式，如圖 5.1 (a) 所示。序向邏輯電路則是由組合邏輯電路與記憶電路組成，且其輸出是所有輸入與前次輸出的組合形式，如圖 5.1 (b) 所示。

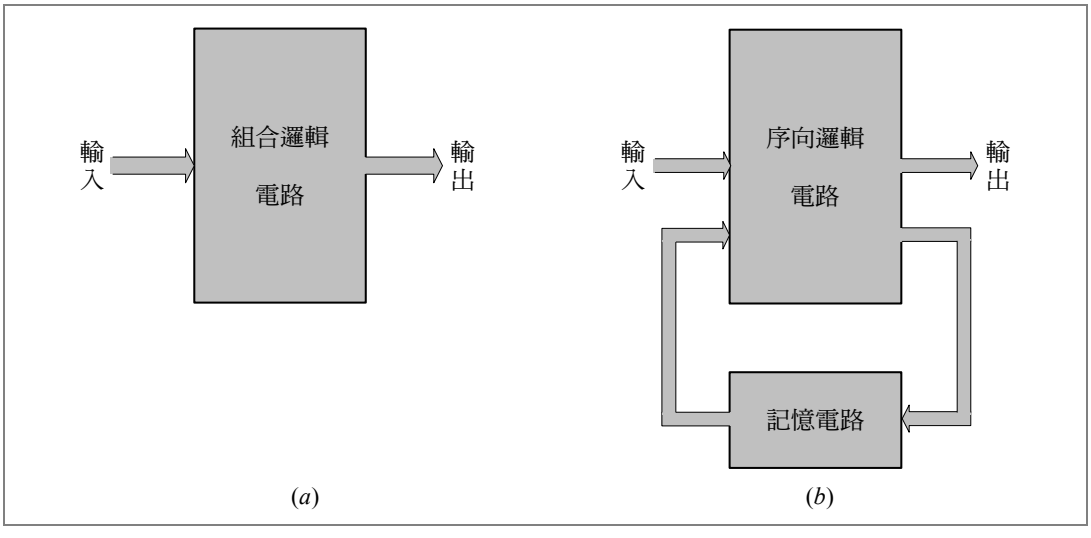


圖 5.1 組合邏輯與序向邏輯簡圖

5-1-2 組合邏輯分析方法

組合邏輯電路分析是以等效布林函數與真值表來表示該邏輯電路輸入與輸出的關係，以便技術人員進行檢查或維修。布林函數可以表示邏輯電路輸入與輸出之間的邏輯運算關係，真值表則列出輸入組合與輸出的對應狀態。

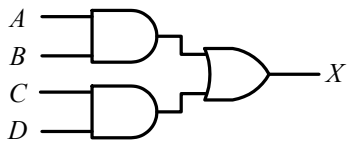
邏輯電路分析步驟

1. 由輸入端向輸出端，依次寫出電路中各個邏輯閘輸出的布林運算式。
2. 將最後輸出的布林運算式，寫成布林函數表示式。
3. 將布林函數化成積項之和或和項之積。
4. 將積項之和或和項之積化成最小項之和或最大項之積。
5. 將最小項之和或最大項之積發展成真值表。

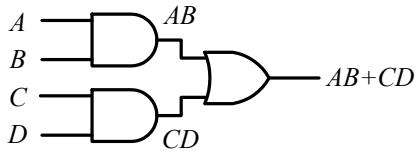
5-1-3 組合邏輯分析範例

例 5.1 寫出下列邏輯電路的布林函數積項之和，並發展成真值表。

電路 AND-OR 邏輯電路如下。



輸出 寫出各個邏輯閘輸出的布林運算式。



函數 邏輯電路的等效布林函數與化成最小項之和如下。

$$X = AB + CD$$

11⓪⓪ ⓪⓪11
11⓪① ⓪①11
11①⓪ ①⓪11
11①① ①①11

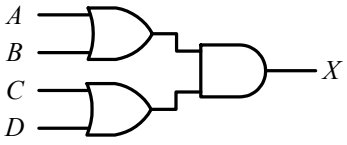
$$= AB\overline{C}\,\overline{D} + AB\overline{C}D + ABC\overline{D} + ABCD + \overline{A}\,\overline{B}CD + \overline{A}BCD + A\overline{B}CD + ABCD$$

真值表 利用最小項之和發展出下列的真值表。

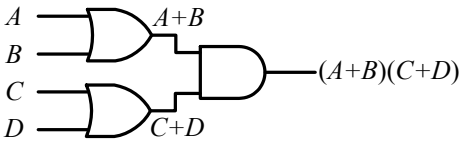
A	B	C	D	X	最小項
0	0	0	0	0	
0	0	0	1	0	
0	0	1	0	0	
0	0	1	1	1	$\overline{A}\overline{B}CD$
0	1	0	0	0	
0	1	0	1	0	
0	1	1	0	0	
0	1	1	1	1	$A\overline{B}\overline{C}\overline{D}$
1	0	0	0	0	
1	0	0	1	0	
1	0	1	0	0	
1	0	1	1	1	$A\overline{B}CD$
1	1	0	0	1	$AB\overline{C}\overline{D}$
1	1	0	1	1	$AB\overline{C}D$
1	1	1	0	1	$ABC\overline{D}$
1	1	1	1	1	$ABCD$

例 5.2 寫出下列邏輯電路的布林函數和項之積，並發展成真值表。

電路 OR-AND 邏輯電路如下。



輸出 寫出各個邏輯閘輸出的布林運算式。



函數 邏輯電路的等效布林函數與化成最大項之積如下。

$X=(A+B)\quad\cdot\quad(C+D)$

0+0+⓪+⓪　⓪+⓪+0+0

0+0+⓪+①　⓪+①+0+0

0+0+①+⓪　①+⓪+0+0

0+0+①+①　①+①+0+0

$=(A+B+C+D)(A+B+C+\overline{D})(A+B+\overline{C}+D)(A+B+\overline{C}+\overline{D})$

$(A+B+C+D)(A+\overline{B}+C+D)(\overline{A}+B+C+D)(\overline{A}+\overline{B}+C+D)$

真值表 利用最大項之積發展出下列的真值表。

A	B	C	D	X	最大項
0	0	0	0	0	$A+B+C+D$
0	0	0	1	0	$A+B+C+\overline{D}$
0	0	1	0	0	$A+B+\overline{C}+D$
0	0	1	1	0	$A+B+\overline{C}+\overline{D}$
0	1	0	0	0	$A+\overline{B}+C+D$
0	1	0	1	1	
0	1	1	0	1	
0	1	1	1	1	
1	0	0	0	0	$\overline{A}+B+C+D$
1	0	0	1	1	
1	0	1	0	1	
1	0	1	1	1	
1	1	0	0	0	$\overline{A}+\overline{B}+C+D$
1	1	0	1	1	
1	1	1	0	1	
1	1	1	1	1	

5-2 組合邏輯電路設計

5-2-1 組合邏輯設計方法

組合邏輯電路設計是將一個文字敘述問題換成組合邏輯電路。例如，設計一個加法器或設計一個投票器，設計步驟如下。

邏輯電路設計步驟

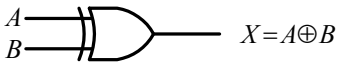
1. 由問題中，確定輸入變數個數與輸出變數的個數。
2. 將問題轉換成輸入變數與輸出變數關係的真值表。
3. 將真值表轉換成布林函數最小項之和 (或最大項之積)。
4. 利用布林定理或卡諾圖，化簡布林函數為積項之和 (或和項之積)。
5. 畫出布林函數的等效邏輯電路。

5-2-2 設計 XOR 電路

XOR 的運算定義為當偶數個輸入為 1 時則輸出為 0，而當奇數個輸入為 1 時則輸出為 1。在 3-1-7 節曾經介紹過 XOR 邏輯閘符號、運算符號與運算真值表，而本節將利用 NOT、AND、OR、NAND 閘的組合邏輯來設計二輸入或三輸入 XOR 閘。

例 5.5 使用 NOT、AND、OR、NAND，設計二輸入 XOR 電路。

符號 二輸入 XOR 邏輯閘符號與布林表示式。

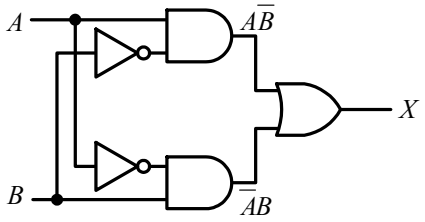


真值表 Exclusive-OR 的真值表與最小項、最大項定義如下。

A	B	X	最小項	最大項
0	0	0		$A+B$
0	1	1	$\overline{A}B$	
1	0	1	$A\overline{B}$	
1	1	0		$\overline{A}+\overline{B}$

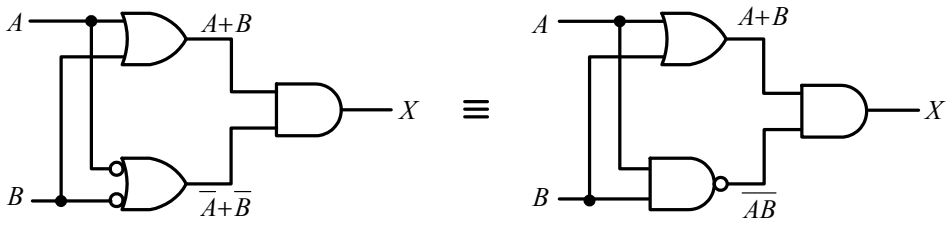
最小值 真值表中最小項之和的布林函數與等效邏輯電路如下。

$X = \overline{A}B + A\overline{B}$



最大值 真值表中最大項之積的布林函數與等效邏輯電路如下。

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

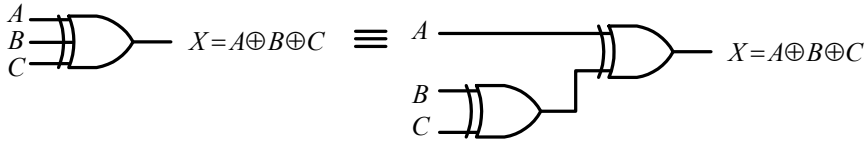


三輸入 XOR 邏輯閘的邏輯運算，除了遵循『偶數個輸入為 1 時輸出為 0，奇數個輸入為 1 時輸出為 1』的運算原則，還可使用結合律先執行其中二個輸入的 XOR 運算，再與第三個輸入執行 XOR 運算，如下；

$$X = A \oplus B \oplus C = A \oplus (B \oplus C) = (A \oplus B) \oplus C$$

例 5.6 **設計三輸入 XOR 電路。**

符號 三輸入 XOR 邏輯閘符號與布林表示式。

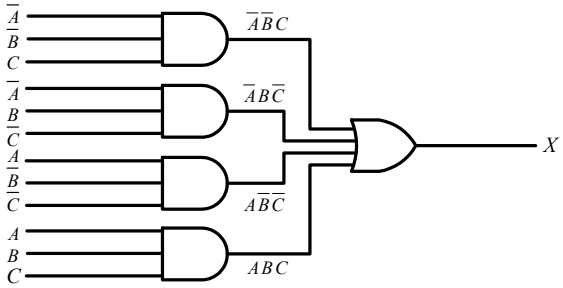


真值表 三輸入 XOR 的真值表與最小項定義如下。

A	B	C	B⊕C	X=A⊕(B⊕C)	最小項
0	0	0	0	0	
0	0	1	1	1	$\overline{A}\,\overline{B}C$
0	1	0	1	1	$\overline{A}B\overline{C}$
0	1	1	0	0	
1	0	0	0	1	$A\overline{B}\overline{C}$
1	0	1	1	0	
1	1	0	1	0	
1	1	1	0	1	ABC

函數 真值表中最小項之和的布林函數與等效邏輯電路如下。

$$X = \overline{A}\,\overline{B}C + \overline{A}B\overline{C} + A\overline{B}\overline{C} + ABC$$

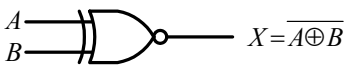


5-2-3 設計 XNOR 電路

XNOR 的運算定義為當偶數個輸入為 1 時則輸出為 1，而當奇數個輸入為 1 時則輸出為 0。在 3-1-8 節曾經介紹過 XNOR 邏輯閘符號、運算符號與運算真值表，而本節將利用 NOT、AND、OR、NOR 閘的組合邏輯來設計二輸入或三輸入 XNOR 閘。

例 5.7 **使用 NOT、AND、OR、NOR，設計二輸入 XNOR 電路。**

符號 二輸入 XNOR 邏輯閘符號與布林表示式。

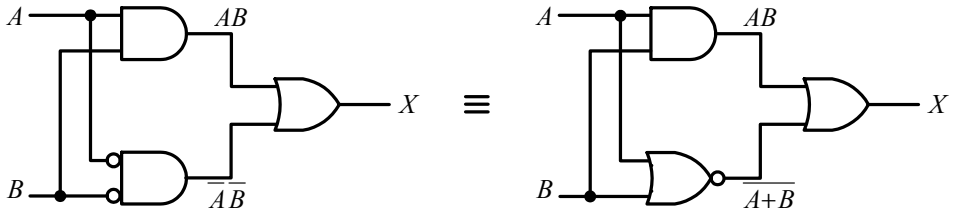


真值表 二輸入 Exclusive-NOR 的真值表與最小項、最大項定義如下。

A	B	X	最小項	最大項
0	0	1	$\overline{A}\,\overline{B}$	
0	1	0		$A+\overline{B}$
1	0	0		$\overline{A}+B$
1	1	1	AB	

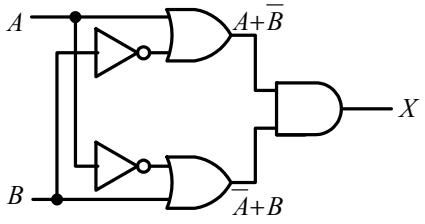
最小值 真值表中最小項之和的布林函數與等效邏輯電路如下。

$$X = AB + \overline{A}\,\overline{B} = AB + (\overline{A+B})$$



最大值 真值表中最大項之積的布林函數與等效邏輯電路如下。

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



三輸入 XNOR 邏輯閘的邏輯運算，除了遵循『偶數個輸入為 1 時輸出為 1，奇數個輸入為 1 時輸出為 0』的運算原則，還可使用結合律先執行其中二個輸入的 XOR 運算，再與第三個輸入執行 XNOR 運算，如下；

$$X = \overline{A \oplus B \oplus C} = \overline{A \oplus (B \oplus C)} = \overline{(A \oplus B) \oplus C}$$

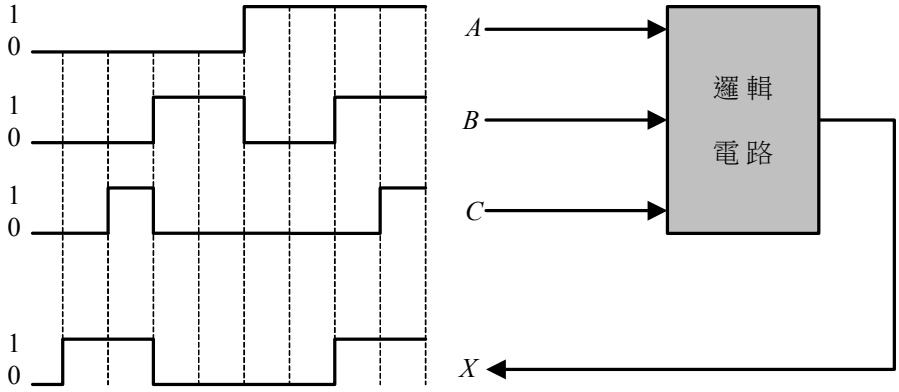
5-2-6 利用波形設計電路

利用波形設計邏輯電路步驟

1. 將輸入波形與輸出波形轉換成輸入變數與輸出變數關係的真值表。
2. 將真值表轉換成布林函數最小項之和 (或最大項之積)。
3. 利用布林定理或卡諾圖，化簡布林函數為積項之和 (或和項之積)。
4. 畫出布林函數的等效邏輯電路。

例 5.11 求適合於下列輸入波形與輸出波形的邏輯電路。

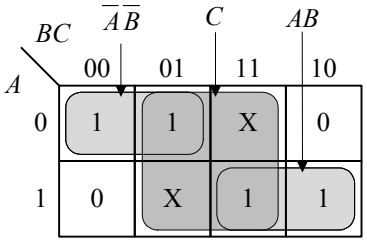
波形 已知輸入波形與輸出波形如下。



真值表 找出已知的輸入與輸出關係，並發展成真值表，而未知的關係則以 X (Don‘t Care) 代替。

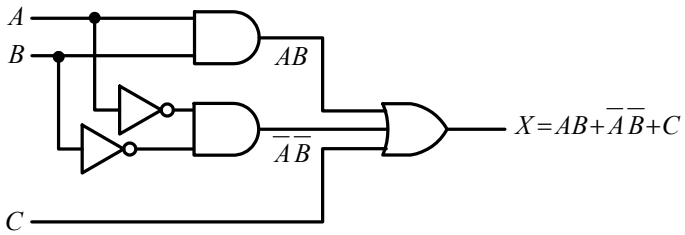
A	B	C	X
0	0	0	1
0	0	1	1
0	1	0	0
0	1	0	X
1	0	0	0
1	0	0	X
1	1	0	1
1	1	1	1

化簡 利用卡諾圖化簡布林函數如下。



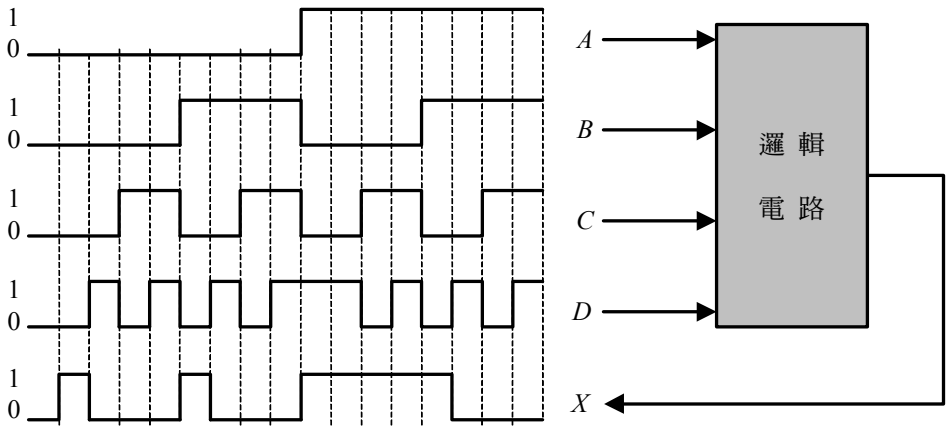
$$X = \overline{A}\,\overline{B} + AB + C$$

電路 將布林函數畫成等效邏輯電路如下。



例 5.12 求適合於下列輸入波形與輸出波形的邏輯電路。

波形 已知輸入波形與輸出波形如下。

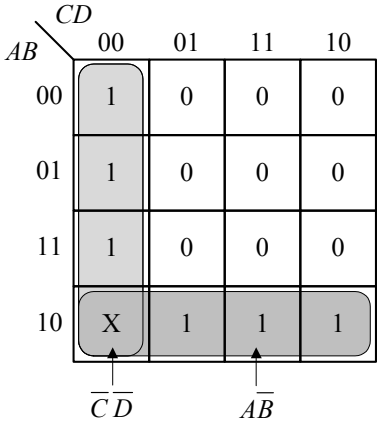


真值表 找出已知的輸入與輸出關係，並發展成真值表，而未知的關係則以 X (Don't Care) 代替。

A	B	C	D	X
0	0	0	0	1
0	0	0	1	0
0	0	1	0	0
0	0	1	1	0
0	1	0	0	1
0	1	0	1	0
0	1	1	0	0
0	1	1	1	0

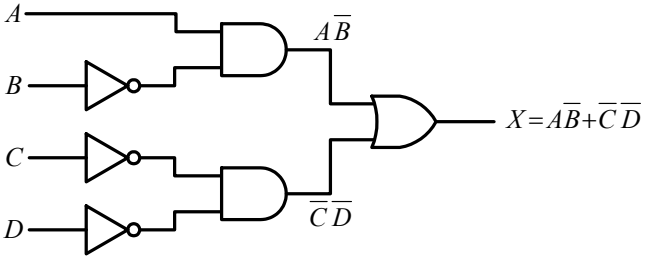
A	B	C	D	X
1	0	0	1	X
1	0	0	1	1
1	0	1	0	1
1	0	1	1	1
1	1	0	0	1
1	1	0	1	0
1	1	0	0	0
1	1	1	1	0

化簡 利用卡諾圖化簡布林函數如下。

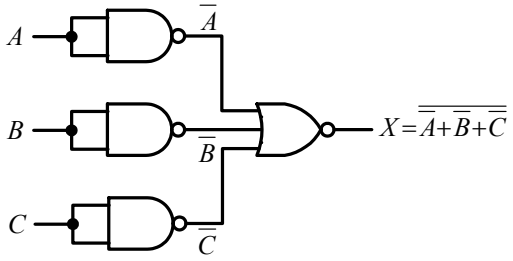


$$X = A\overline{B} + \overline{C}\,\overline{D}$$

電路 將布林函數畫成等效邏輯電路如下。

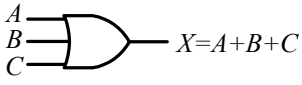


解答 $X = ABC = \overline{\overline{ABC}} = \overline{\overline{A} + \overline{B} + \overline{C}}$

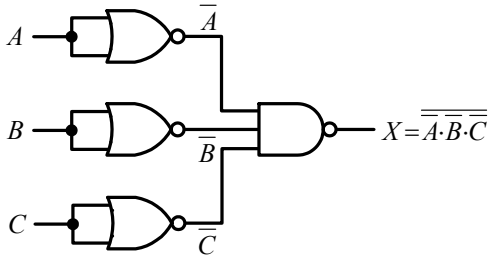


例 5.18 利用 NOR-NAND 組合邏輯，取代下列布林函數與邏輯電路。

題目 $X = A + B + C$



解答 $X = A + B + C = \overline{\overline{A + B + C}} = \overline{\overline{A} \cdot \overline{B} \cdot \overline{C}}$



5-4 組合邏輯電路的實現方式

從 1947 年貝爾實驗室研發的電晶體取代了真空管之後，由於具備體積更小、性能更穩定、而且成本低廉的優點，電晶體不僅引發了一連串固態物理學的革命，數位電子產業也因此得以蓬勃發展，更為今日的半導體產業奠定良好的基礎，電晶體對現今科技產業有著難以衡量的重要性。

在第一章曾經提到，數位系統設計的演進過程大致可分為「電晶體層次」、「邏輯閘層次」、「暫存器轉移層次」和「系統層次」等四個階段。

現階段的數位系統設計環境仍以「暫存器轉移層次」為主。RTL 層次設計主要的特色就是大量的使用暫存器與算數邏輯元件(ALU 或 FU)來描述電路，目前最熱門

的硬體電路描述語言莫過於 VHDL 與 Verilog。由於 VHDL 程式語言的架構較為嚴謹，本書後面的教學和專案做中學都是採用 VHDL 程式語言。

硬體描述語言是屬於設計的工具或平台，最後仍須將設計的成果燒錄至數位 IC 內(FPGA 或 CPLD 晶片)，數位系統的功能才得以實現。所以除了認識硬體描述語言之外，系統開發者也必須對數位 IC 的分類有基本的認識。

5-4-1 數位 IC 分類

數位 IC 大致可分為標準邏輯 IC(Standard Logic IC)和客製化 IC(ASIC)兩大類，圖 5.8 為數位 IC 的家族分類。數位系統設計開發者可以依據需要從中選擇適合的晶片來實現電路。

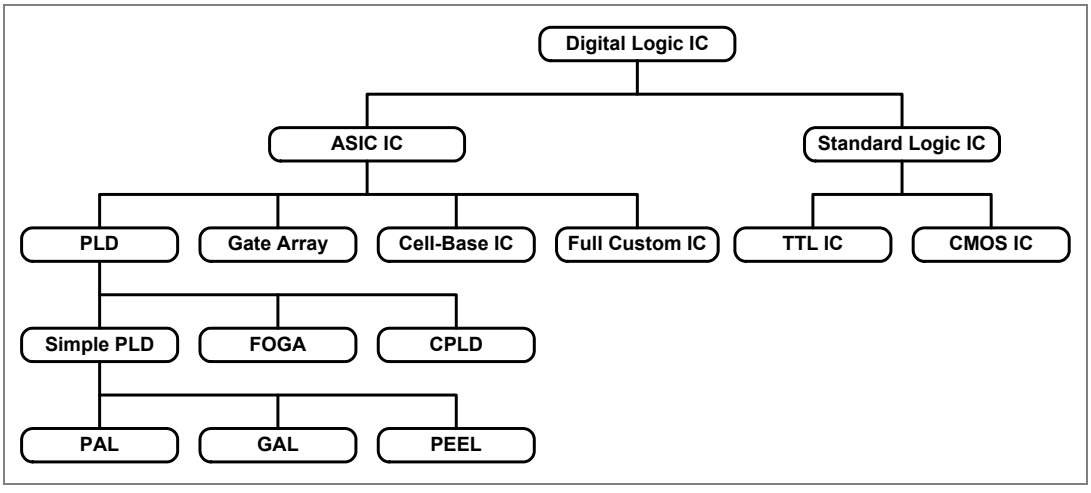


圖 5.8 數位 IC 的家族分類

5-4-2 FPGA 和 CPLD 晶片

由於市場需求變化迅速，市場規模也從大量製造演變成少量多樣的型態，因此並非所有的系統設計開發者都可以負擔得起客製化 IC 的費用。FPGA 或 CPLD IC 便在這種需求下出現，由於具備體積小、容量大、可程式規劃可以大幅降低開發成本，更能夠在短時間內完成產品設計並讓商品上市的優點，系統設計開發者可依據其電路規模的大小或規格的要求，選擇功能、價格適合的可程式規劃的 FPGA 晶片來實現電路，因此很快就獲得青睞，擴展出相當的市場規模。表 5.1 列出常用可程式晶片的特性比較。

表 5.1　常用可程式晶片的特性比較

	Full-Custom IC	Cell-Based IC	Gate Arrays	FPGA/CPLD
Speed	★★	★	★	★
Integration Density	★★	★	★	★
Volume device cost	HIGH	HIGH	LOW	LOW
Risk Reduction			★	★★
Time to Market			★	★★

當然，一項設計的好壞很難從單一角度來評論其好壞，一個設計通常是許多條件下取捨後的結果，基本上能夠滿足客戶的需求就是一個好的設計。好比在都會區的交通規劃，打算從 A 點前往 B 點，開車、搭捷運還是騎機車，其選擇會因人、因地、因時而異。

5-4-3　硬體描述語言(HDL)介紹

近年來隨著科技的進步，各種 EDA 工具應運而生，使用硬體描述語言搭配 FPGA 或 CPLD 這類的晶片來進行數位系統的設計逐漸成為近年來的主要趨勢，硬體描述語言的問世，甚至可以視為數位系統設計發展的一個里程碑，目前市面上較為通用的硬體描述語言有 VHDL 和 Verilog 兩種。由於 VHDL 的架構比較嚴謹，本書接下來的章節中都將以 VHDL 這款硬體描述語言的基礎架構來描述電路。

5-4-4　VHDL 的由來

VHDL 是 Very high speed integrated circuit Hardware Description Language 的縮寫，意思是非常高速積體電路的硬體描述語言。這是一項由美國國防部所支持的研究計畫，目的是為了把電子電路的設計意義以文字或文件的方式保存下來，以便其他人能輕易地了解電路的設計意義。

1985 年完成第一版的硬體描述語言，兩年後(1987)成為 IEEE 標準，即 IEEE1076 標準。1988 年，美國國防部規定所有官方的 ASIC 設計都必須以 VHDL 為設計描述語言，所以 VHDL 就漸漸地成為工業界的標準。之後於 1993 年增修為眾所週知的

IEEE1164 標準，1996 年，IEEE 再將電路合成的標準程式與規格加入至 VHDL 硬體描述語言之中，稱之為 IEEE 1076.3 標準。

由於半導體製程技術和電腦輔助設計工具的快速進步，VHDL 能夠以高階電路描述語言的方式，讓複雜的電路可以透過 VHDL 編輯合成器快速地完成電路的合成，並且可立即將設計燒錄在 FPGA 或 CPLD 等可程式規劃晶片上，不但大大地縮短數位系統設計開發的時程，更可降低開發過程中的 Non Return Cost，因此很快就受到市場的注目。

由於 VHDL 電路描述語言所能涵蓋的範圍相當廣，能適用於各種不同階層的設計工程師之需求。從 ASIC 的設計到 PCB 系統的設計，VHDL 電路描述語言都能夠派上用場，所以 VHDL 電路設計很快地成為硬體設計工程師的必備工具之一。

5-4-5 設計流程

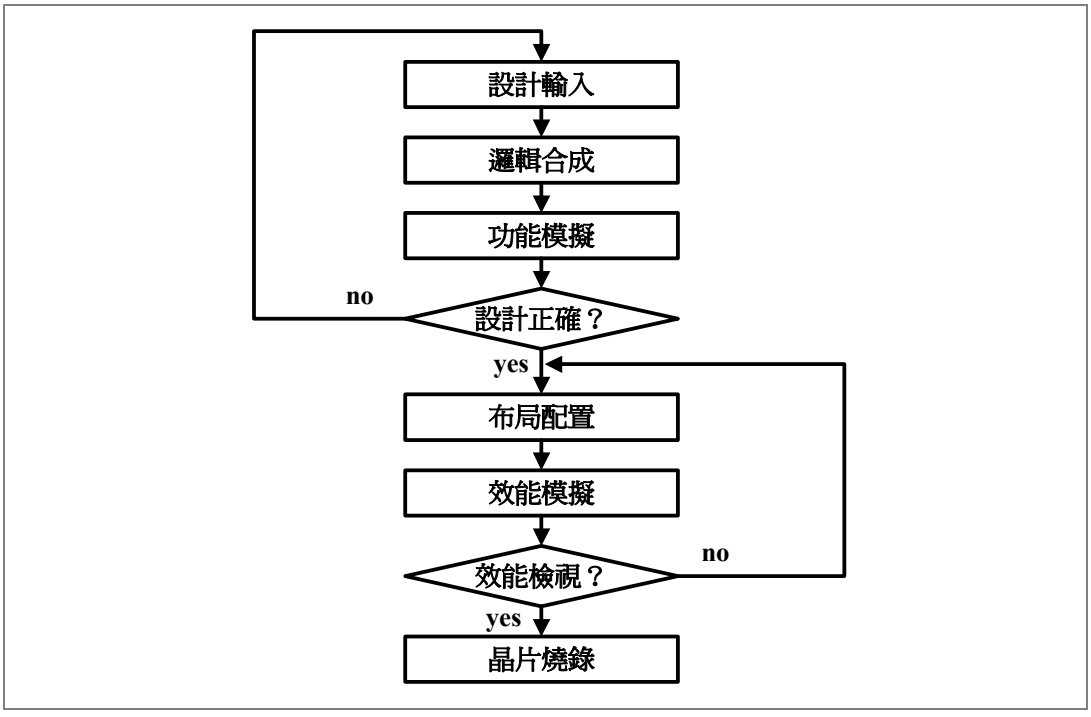


圖 5.9 VHDL 數位系統設計流程

5-4-6 VHDL 語言的程式結構

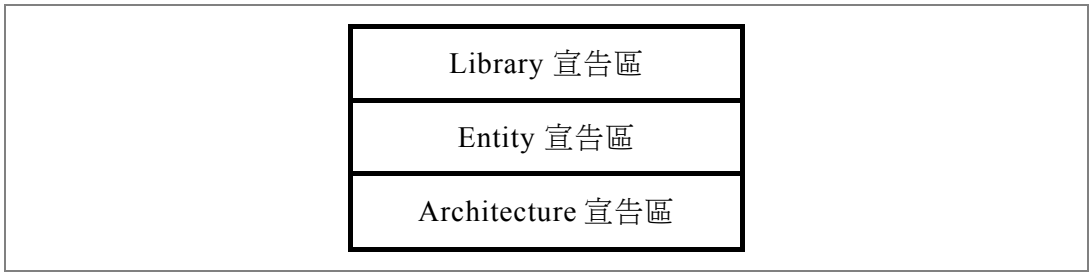


圖 5.10 VHDL 程式基本架構

圖 5.10 是一個 VHDL 程式基本架構，每個宣告區的用途說明如下：

LIBRARY 宣告區

宣告要使用的零件庫名稱。當程式在執行組譯(Compiler)或是邏輯電路合成(Logic Synthesize)時，如果找不到零件就會出現錯誤，無法執行電路合成的工作。

瞭解各個零件庫的用途對程式開發者是有其必要的，但過程卻很繁瑣，尤其對於初學者而言更是一項艱澀的工作。

建議初學者先行參考範例程式中的三種零件庫宣告，絕大多數的情況下，這三種零件庫應該可以滿足初學者的需求，至於其他的零件庫可以慢慢再行了解。

```
use ieee.std_logic_1164.all;          --定義基礎標準邏輯的資料型態和函數
use ieee.std_logic_arith.all;         --定義算數運算元及相關的資料型態
use ieee.std_logic_unsigned.all;      --用於邏輯常數和數值常數間的轉換
```

ENTITY 宣告區

ENTITY 宣告區的作用是宣告電路外觀的接腳，也就是輸入輸出的介面，包含每個輸入及輸出訊號的名稱、資料型態和資料的位元寬度。

```
ENTITY 電路名稱 IS
      PORT
(
            接腳名稱 1 : 輸出入情形  資料型態；
            接腳名稱 2 : 輸出入情形  資料型態；
            .......
            接腳名稱 n : 輸出入情形  資料型態(最後一個宣告不要打分號)
```

```
        );
END 電路名稱;
```

其實，VHDL 是一種使用文字的硬體描述語言，所以基本上它的描述方式就跟說話類似，只不過它說的是美語，而且必須符合 VHDL 的語法，邏輯合成器才能夠將電路合成出來；如果把上面的 ENTITY 宣告的架構用國語重說一次，他的大意如下：

```
宣告「ＸＸ電路」的介面
介面宣告，如下面的括弧所示
(
        根據語法來宣告每一個輸入和輸出訊號。說完記得要打分號(;)
        訊號宣告包含[接腳名稱][：][輸入或輸出][ 資料型態]四個部分
)
介面宣告結束
```

經過中英對照說明之後，有沒有覺得它其實跟說話沒有太大的差別呢？

Architecture 宣告區

Architecture 宣告區的作用是在做電路架構的行為描述。換句話說，就是在描述我們所要設計的電路行為與特性，或是電路提供了哪些功。Architecture 宣告區的語法如下：

```
ARCHITECTURE 架構名稱 OF 電路名稱 IS
        內部訊號/元件宣告區
        ……
BEGIN
        電路內部描述
        ……
END 架構名稱;
```

一個 VHDL 程式可以有好幾個 Architecture 宣告區，最後再利用 Configuration 宣告來指定哪一個 Architecture 會被啟用。這好比我們會在一部電腦同時安裝多個作業系統一樣，再利用開機時的組態配置功能來決定進入哪一種作業系統。不過對初學者而言，能寫出一個 Architecture 宣告就很厲害了。同時為了降低學習的困難度，本書中的程式範例中也只會出現一個 Architecture 宣告。

因為每一個電路的功能不盡相同，因此 Architecture 宣告區會是一個 VHDL 程式中難度較高的部分，不過雖然難度較高，但是往往困難之處也正是系統開發設計者展現功力與巧思的最佳場所。

從 VHDL 範本程式開始寫程式

下面是作者依據 VHDL 程式架構，先行寫好的一個 VHDL 範本程式，之後當讀者需要撰寫 VHDL 程式時，可以套用此一範本，再依據現況稍做修改，便能較輕鬆地完成一個 VHDL 程式的撰寫工作。其實學任何東西在一開始時，大多是從模仿開始的，有了基本認識和技能後，就可以跳脫模仿，自行完成所有的工作。

程式 5-01：VHDL 範本程式

```
1:  --LIBRARY DECLARATION(零件庫宣告)
2:  library ieee;
3:  use ieee.std_logic_1164.all;
4:  use ieee.std_logic_arith.all;
5:  use ieee.std_logic_unsigned.all;
6:
7:  --ENTITY DECLARATION(電路外部的輸入輸出介面宣告)
8:  entity __Entity_Name is
9:  port(
10:    __clk  :in  std_logic;
11:    __name :in  std_logic_vector(__width downto 0);
12:    ……
13:    __name :out std_logic_vector(__width downto 0)
14:      );
15: end __Entity_Name ;
16:
17: --ARCHITECTURE BODY DESCRIPTION(電路內部的功能/行為描述)
18: architecture a of __Entity_Name is
19:    signal temp1: std_logic_vector(__width downto 0);
20:    --(如果沒有信號需要宣告，這裡就不寫。)
21: begin
22:    __Behavior Description(內部的行為、功能描述)
23:    --(因為只是程式範本，功能描述留待程式開發設計人員自行完成)
24: end a ;
```

程式中有出現底線的地方是讀者需要按實際情況或要求來進行編輯的地方。接者，先來依樣畫葫蘆，嘗試完成一個 VHDL 程式。下面的範例想設計一個電路(半加法器)，可以完成 x + y = sum 的運算。

撰寫第一個 VHDL 程式

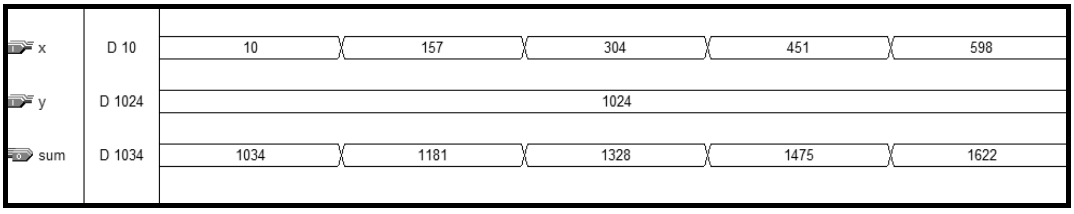
利用範本程式，稍加修改後，完成的第一個 VHDL 程式如下：

程式 5-02：第一個 VHDL 程式

```
1:  --LIBRARY DECLARATION
2:  library ieee;
3:  use ieee.std_logic_1164.all;
4:  use ieee.std_logic_arith.all;
5:  use ieee.std_logic_unsigned.all;
6:
7:  --ENTITY DECLARATION
8:  entity ADDER is
9:  port(
10:     x, y  :in  std_logic_vector(31 downto 0);
11:     sum:out std_logic_vector(31 downto 0)
12:     );
13: end ADDER ;
14:
15: --ARCHITECTURE BODY DESCRIPTION
16: architecture a of ADDER is
17: begin
18:     sum <= x + y;
19: end a ;
```

模擬的結果：

程式經過組譯合成後，模擬得到結果如下圖。仔細檢視一下圖中的 X + Y 執行加法運算後的結果得到 sum，其運算結果是正確的。嗯，是不是沒有很難呢？



補充說明：

更完整的 VHDL 程式架構共有五個區塊如圖 5.11 所示，它比圖 5.10 中的 VHDL 程式基本架構多出兩個區塊。增加的兩個區塊是：PACKAGE 宣告區和 CONFIGURATION 宣告區。

PACKAGE 宣告區可視為自訂零件庫宣告區。可以把一些常用的或是自行定義的零件寫在裡面。

CONFIGURATION 宣告區是負責選擇哪一種架構來和電路，它的前提是程式中必須包含有至少兩個以上的 Architecture 宣告。假設一個 VHDL 程式中描述了三個 ARCHITECTURE，分別是 ARCHITECTURE A、ARCHITECTURE B 和 ARCHITECTURE C，那麼就可以利用 CONFIGURATION 宣告區來決定選用哪一種架構來合成。舉一個類似的例子加以說明，他很類似如果一台 PC 安裝了 WINDOWS 和 LINUX 兩種作業系統，在開機時電腦會問你要使用哪一種 OS 來開啟電腦，這就是所謂的 CONFIGURATION。不過初學者通常都只有一個 Architecture 宣告，因此就不會有 CONFIGURATION 宣告區的出現。

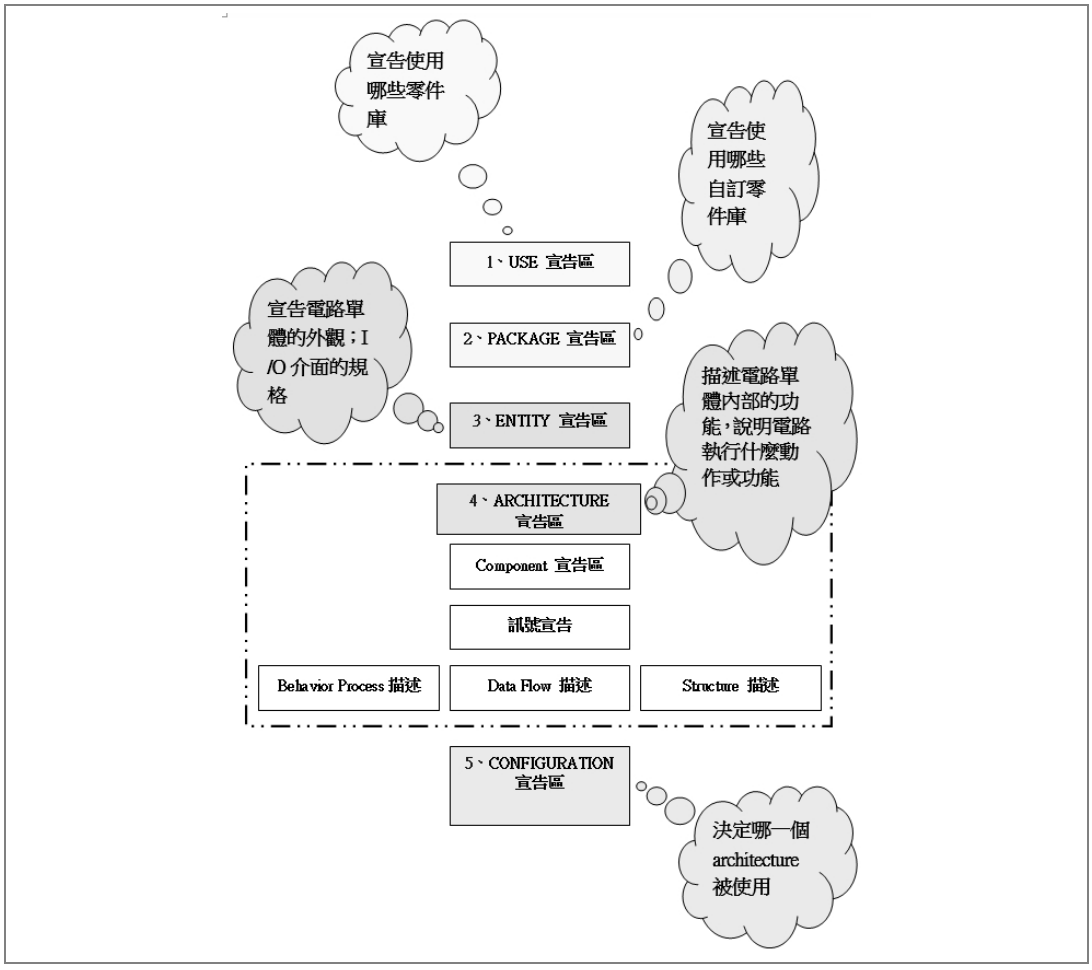


圖 5.11　VHDL 程式架構

通常第二個程式區塊(自製資料庫)和第五個程式區塊(組態配置)在初學階段通常會被省略，因為對初學者而言，還沒有能力完成自訂零件庫，同時開始學寫 VHDL 程式時通常只會寫一個 ARCHITECTURE 宣告，因此也就不會用到第五個程式區塊的組態配置了。

重點複習：

- VHDL 程式基本架構為何？
- ENTITY 宣告區的功能為何？
- ARCHITURER 宣告區的功能為何？
- 解讀下面的程式。

```
--資料庫宣告區(LIBRARY 宣告)
   library ieee;
   use ieee.std_logic_1164.all;
--電路外觀輸入輸出埠宣告區(entity 宣告)
   entity half_adder is
     port (
          x, y : in  std_logic;
          s, c : out std_logic
         );
   end half_adder;
--電路內部功能架構宣告區(ARCHITECTURE 宣告)
   architecture a of half_adder is
   begin
     s <= x xor y;
     c <= x and y;
   end a;
```

簡單看懂 VHDL 程式

想要看懂程式到底在幹什麼就必須用對方法。解讀程式時並不是一行一行的看程式在寫些什麼，而是應該先將程式區分成一個區塊一個區塊，然後再去解讀各個區塊的功能，弄清楚整個程式系統的架構之後，再進一步去探討區塊內的每一行程式在執行什麼功能，便能輕鬆地看懂程式。先將上面的程式分成三個區塊：

- 第一個區塊，程式的第 1 行～第 2 行，是 LIBRARY 宣告區：是在宣告零件資料庫，讓 compiler 可以讀懂程式，並讓邏輯合成契合成相關的電路。

- 第二個區塊，程式的第 3 行～第 8 行，是 ENTITY 宣告區：是在宣告電路的輸入輸出界面。根據程式的描述可以得知其外觀介面如圖 5.12 所示：

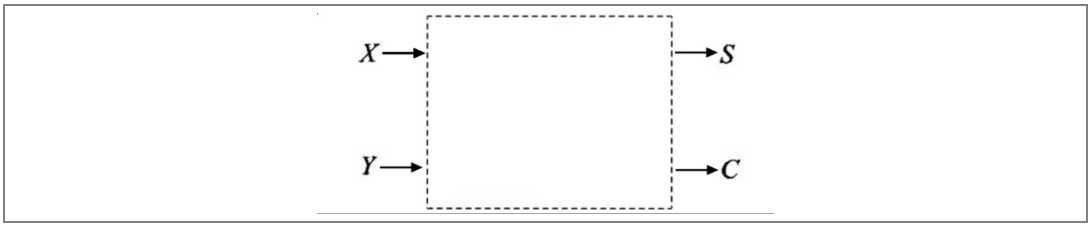


圖 5.12　電路的外觀介面

- 第三個區塊，程式的第 9 行～第 13 行，是 ARCHITECTURE 宣告區：是在描述電路的功能或特性。從第 11 行和第 12 行的程式可以得知電路內部的結構如圖 5.13 所示：

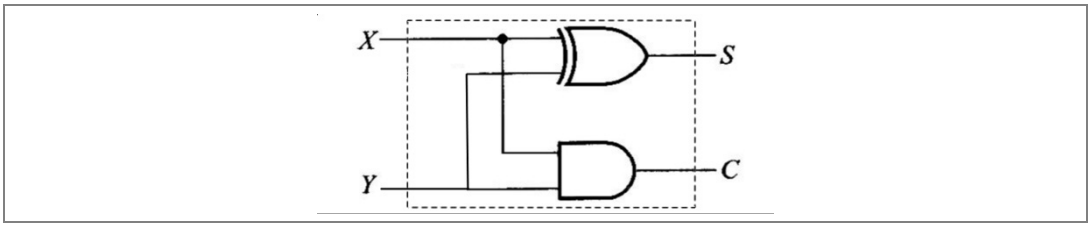


圖 5.13　電路的內部結構

因而得知這個程式是在描述一位元半加器。