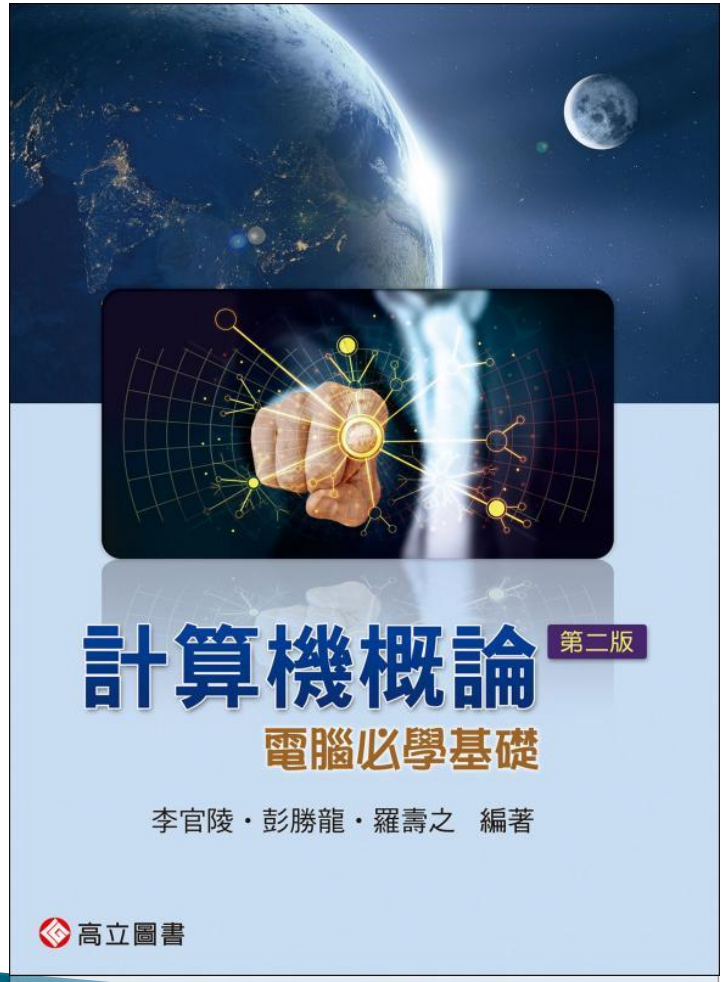


Ch04
計算機組織與數位邏輯設計

李官陵 彭勝龍 羅壽之

- 現今電腦的通用架構
  - 基於范紐曼模式 (von Neumann Model) 架構
  - 將程式指令記憶體和資料記憶體合併在一起的電腦設計概念架構

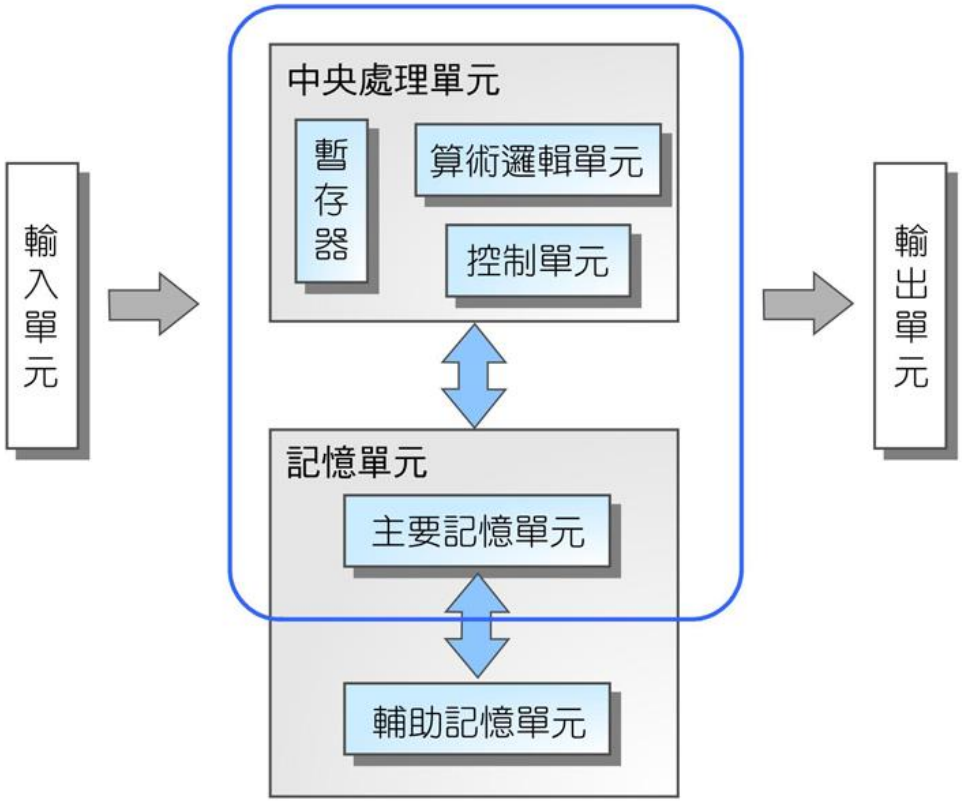
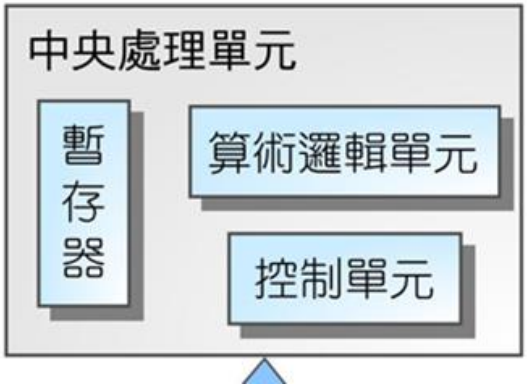


圖 4.1 電腦基礎架構

中央處理單元

- 中央處理單元或中央處理器
  - CPU (Central Processing Unit)
  - 負責程式的執行、控制以及數位資料的處理與運算
  - 主要包含
    - 算術邏輯單元 (Arithmetic Logic Unit, ALU)
    - 控制單元 (Control Unit, CU)
    - 暫存器 (Register)
      - 極小的儲存裝置，可以暫時儲存程式或資料



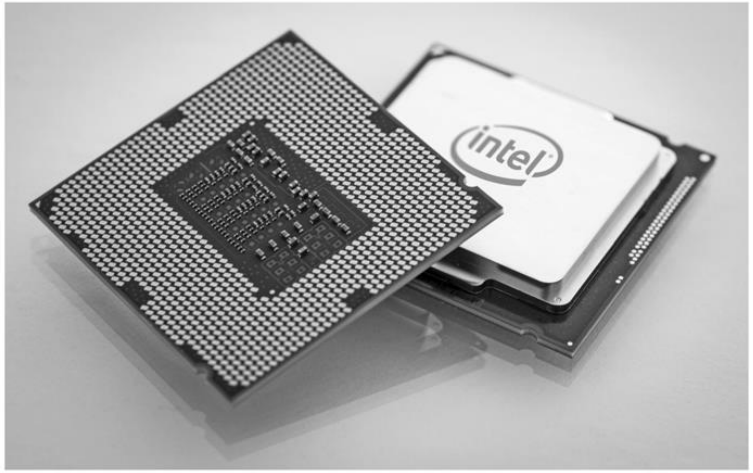


圖 4.2 中央處理器（圖片來源：Intel 官網）

中央處理單元 (續)

1. 控制單元

   負責控制電腦執行程式的流程

   協調輸入、輸出、記憶以及算術邏輯等部門的運作，使得電腦能自動化的處理資料。

2. 算術邏輯單元

   算術邏輯單元是算術單元 (Arithmetic Unit, AU) 與邏輯單元 (Logic Unit, LU) 的合稱

   負責程式中各類算術運算與邏輯運算

中央處理單元 (續)

3. 暫存器
   - 在中央處理器中暫時存放指令或資料的地方
   - 存取速度比主記憶體快很多
     - 能大幅的增加中央處理器的效能
   - 暫存器依用途可大致區分為三類
     - 指令暫存器 (Instruction Register, IR)
       - 儲存準備要被執行的指令
     - 程式指標暫存器 (Program Counter, PC)
       - 記錄下一個所要執行的指令位址
     - 通用暫存器 (General Purpose Register, GPR)
       - 可以由程式設計師指定程式使用的暫存器
       - 可用來儲存程式執行時暫存的資料或運算結果，通常這種暫存器越多，程式執行的效率越高。

記憶單元

- 是電腦執行運算時所使用的記憶體，稱為主要記憶體

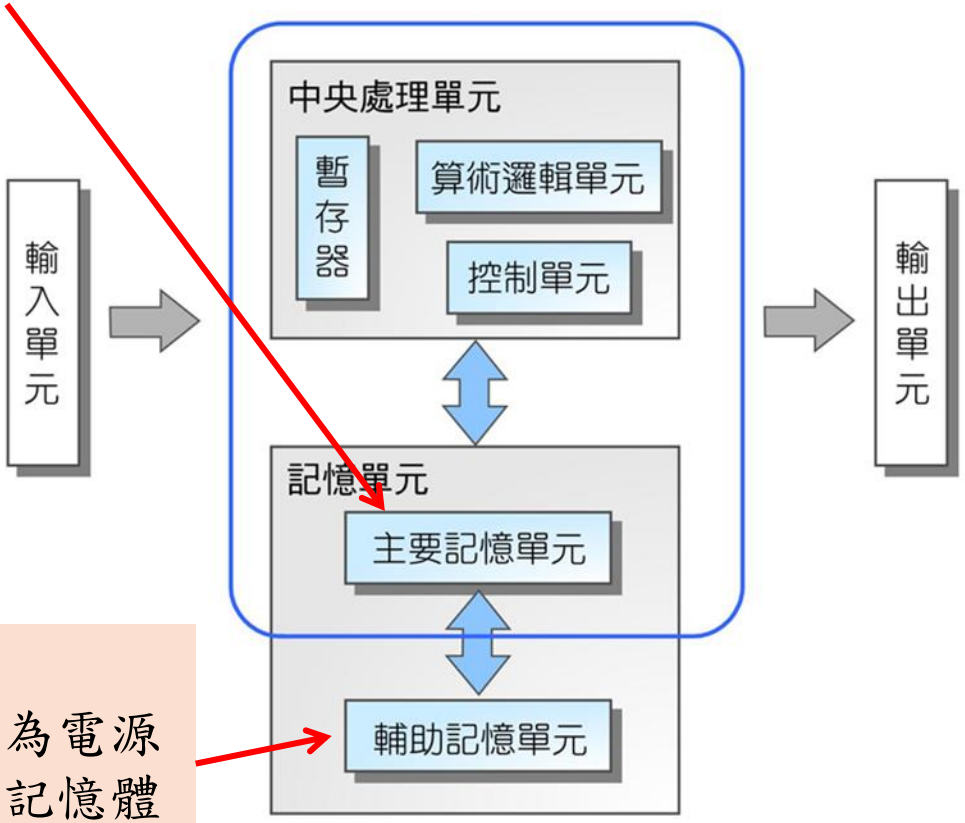


圖 4.1 電腦基礎架構

外部儲存裝置中
存放在外部儲存裝置的資料不會因為電源的消失而消失，因此可隨時讀至主記憶體中進行後續的計算與處理

記憶單元(續)

- 暫存器位於記憶體階層的最上層，是速度最快且單位價格最高的記憶體
- 介於暫存器與主記憶體之間的是快取記憶體，就如同在階層中的位置，快取記憶體乃是一種比暫存器速度慢，但比主記憶體速度快的記憶體

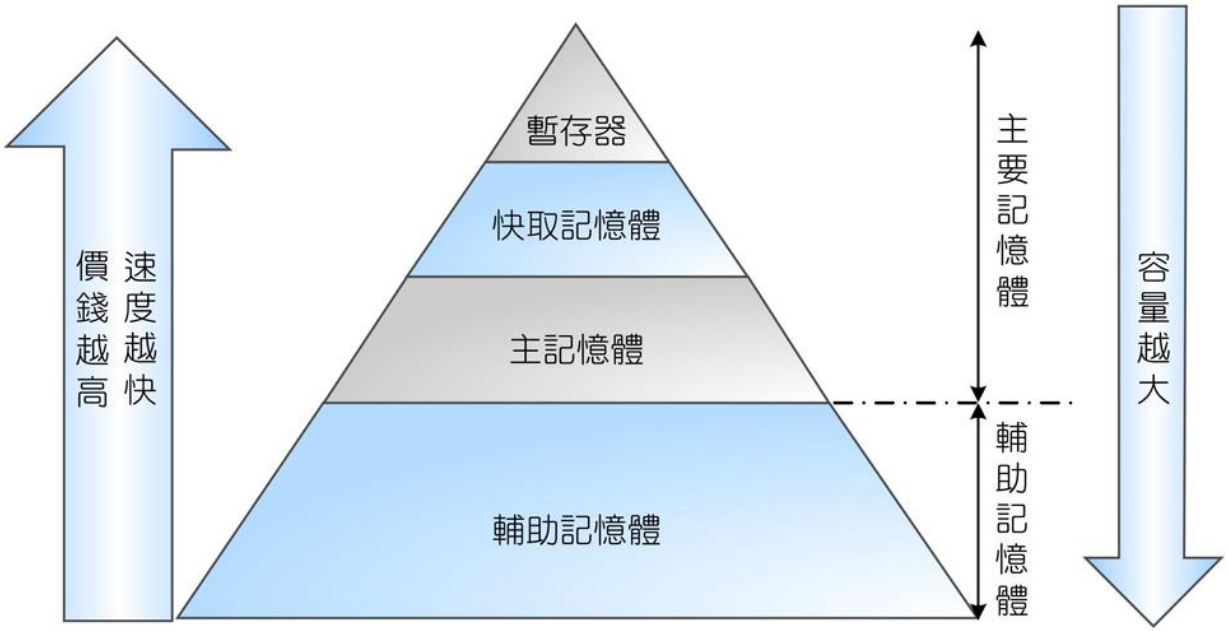


圖 4.3　記憶體階層圖

記憶單元 (續)

- 主記憶體是以半導體元件製成，與輔助記憶體相比，其存取的速度快但成本高。
- 依照其揮發的特性
  - 唯讀記憶體 (Read Only Memory, ROM)
    - 成本較高，屬於非揮發性的記憶體
    - 在無電力的狀態下亦可保留資料
    - 傳統的 ROM 只能寫入資料一次，因此用來儲存開機使用的程式
  - 隨機存取記憶體 (Random Access Memory, RAM)

記憶單元 (續)

RAM 的分類

- 動態隨機存取記憶體 (Dynamic RAM, DRAM)
  - 以電容器 (capacitor) 的方式儲存資料，隨著時間的流逝，電容器會逐漸地失去它的電容量
  - 必須週期性的更新內容以保存其儲存的資料
- 靜態隨機存取記憶體 (Static RAM, SRAM)
  - 以正反器 (flip-flop gate) 的方式設計，在電源維持的狀態下，不須要做更新 (reflesh) 的動作即可保留儲存其內的資料，因此速度較 DRAM 快得許多
  - 因為電路設計的關係，相對於 DRAM 而言，其成本較高，且所需的晶片面積也大得許多

原來的主要用途是在主機板上，作為快取記憶體，但因L2（第二階快取）逐漸內建於CPU中，使SRAM在個人電腦市場的應用空間漸漸縮小，未來的發展以通訊市場為重點，主要是手機市場

記憶單元(續)

- 輔助記憶體
  - 位於記憶體階層的最底層
  - 相較於上層的記憶體，其存取速度慢得許多，但價格便宜且儲存空間大

圖 4.4　硬碟、DVD、隨身碟與記憶卡（圖片來源：林士凱）

記憶單元(續)

- 主記憶體的功能是儲存正在執行的程式與資料
- 輔助記憶體則是用來儲存在電源消失後仍需保留的資料。

表 4.1　主要記憶體與輔助記憶體的比較

	主要記憶體	輔助記憶體
與 CPU 的關係	可以由 CPU 直接存取資料，屬於線上儲存裝置。	必須透過輸入 / 輸出介面存取資料，屬於離線儲存裝置。
揮發性	資料在電源消失後無法保存，為揮發性記憶體。	資料在電源消失後仍可以保存，為非揮發性記憶體。
存取速度與成本	存取速度快、容量較小、成本較高。	存取速度慢、容量大、成本較低。

輸入與輸出單元

- 從現在的觀點，輸入裝置泛指透過使用者的操作將資料輸入至電腦內部的設備；輸出裝置則是將電腦的處理結果與訊息呈現給使用者的設備。

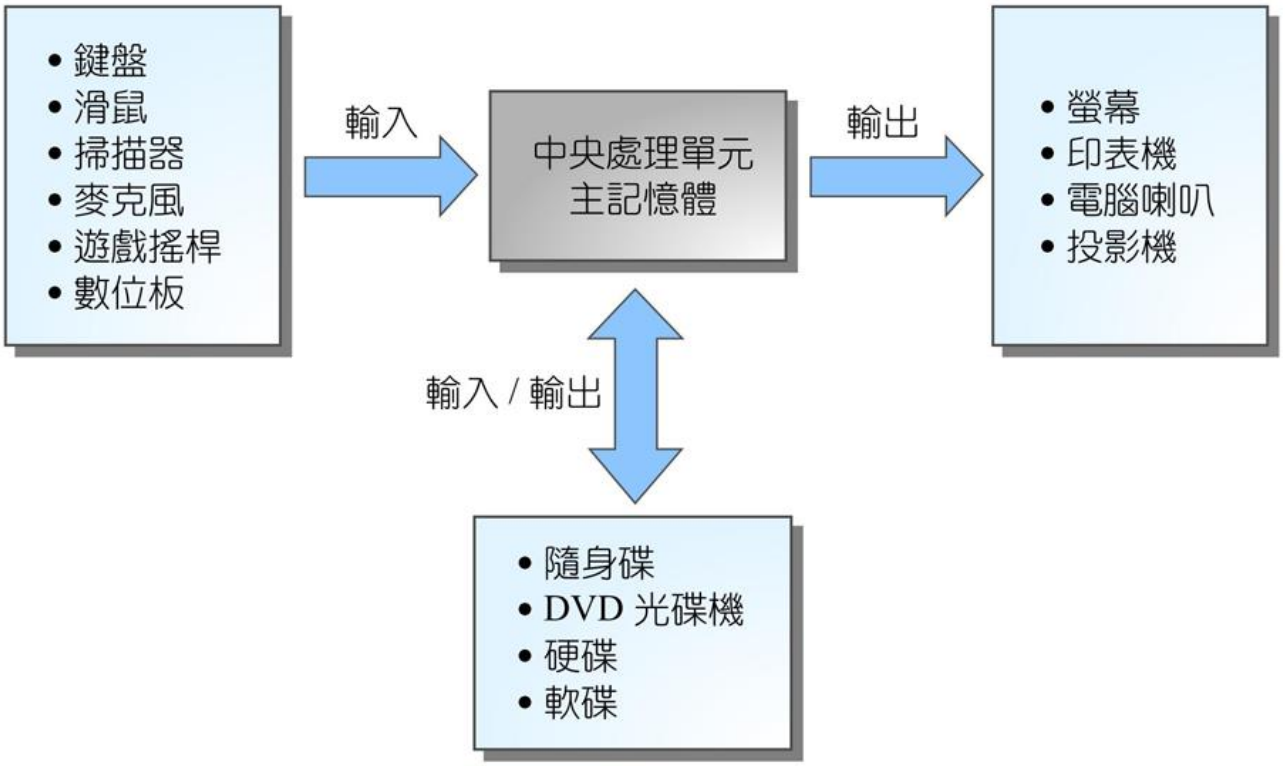


圖 4.5　輸入與輸出

鍵 盤

- 鍵盤是輔助我們將資料輸入電腦的標準輸入裝置，與主機板連接的介面規格主要為 PS2 與 USB。

(a)

(b)

圖 4.6 (a) 101 鍵盤；(b) Windows 專用鍵盤（圖片來源：維基共享資源）

滑　鼠

- 滑鼠它與主機板連接的介面主要為 PS2 與 USB，滑鼠的種類繁多，大致上有二鍵、三鍵、軌跡球與電競用滑鼠。

圖 4.7　二鍵、三鍵、軌跡球與電競用滑鼠（圖片來源：維基共享資源）

掃描器

- 掃描器 (scanner) 基本上就像是影印機，可以將靜態的文件、圖片等複製下來
  - 先將光線投射至待掃描的文件上，光線反射回來後經由感光元件接收
  - 因為文件上較暗的地方反射的光較少，亮的地方反射的光較多，因此感光元件將反射回來的光的強弱轉換為數位資料，最後即可組成數位影像。

圖 4.8 掃描器（圖片來源：高立圖書）

掃描器 (續)

掃描器的解析度

- 掃描後文件的效果與掃描器的解析度有關
- 解析度的單位是DPI (dot per inch)
  - 掃描器在掃描時每英吋的取樣點數
    - 300 dpi 就表示所掃描的文件每英吋將產生 300 點的資料
  - 掃描時設定的 DPI 越高，掃描的結果越精密，但所需的儲存空間也越大
- 掃描器在包裝上通常會標明兩種解析度
  - 光學解析度
    - 掃描器感光元件實際的感測能力，代表掃描器本身硬體的條件
  - 軟體解析度
    - 掃描器將圖檔掃進來後，利用驅動程式運算後所獲得的解析度

螢 幕

- 螢幕是電腦最主要的輸出設備

(a) (b)

圖 4.9 (a) 映像管螢幕；(b) 液晶螢幕 [圖片來源：高立圖書 (a、b)]

印表機

- 在選購印表機時兩個重要的判斷指標
  - 列印速度
    - 單位是 PPM (page per minute)，指得是每分鐘能連續列印的英文文件最大的頁數
  - 列印品質
    - 以解析度標示 (單位是 DPI)
    - 解析度越高，列印的品質越好，然而耗費的碳粉或墨水也越多

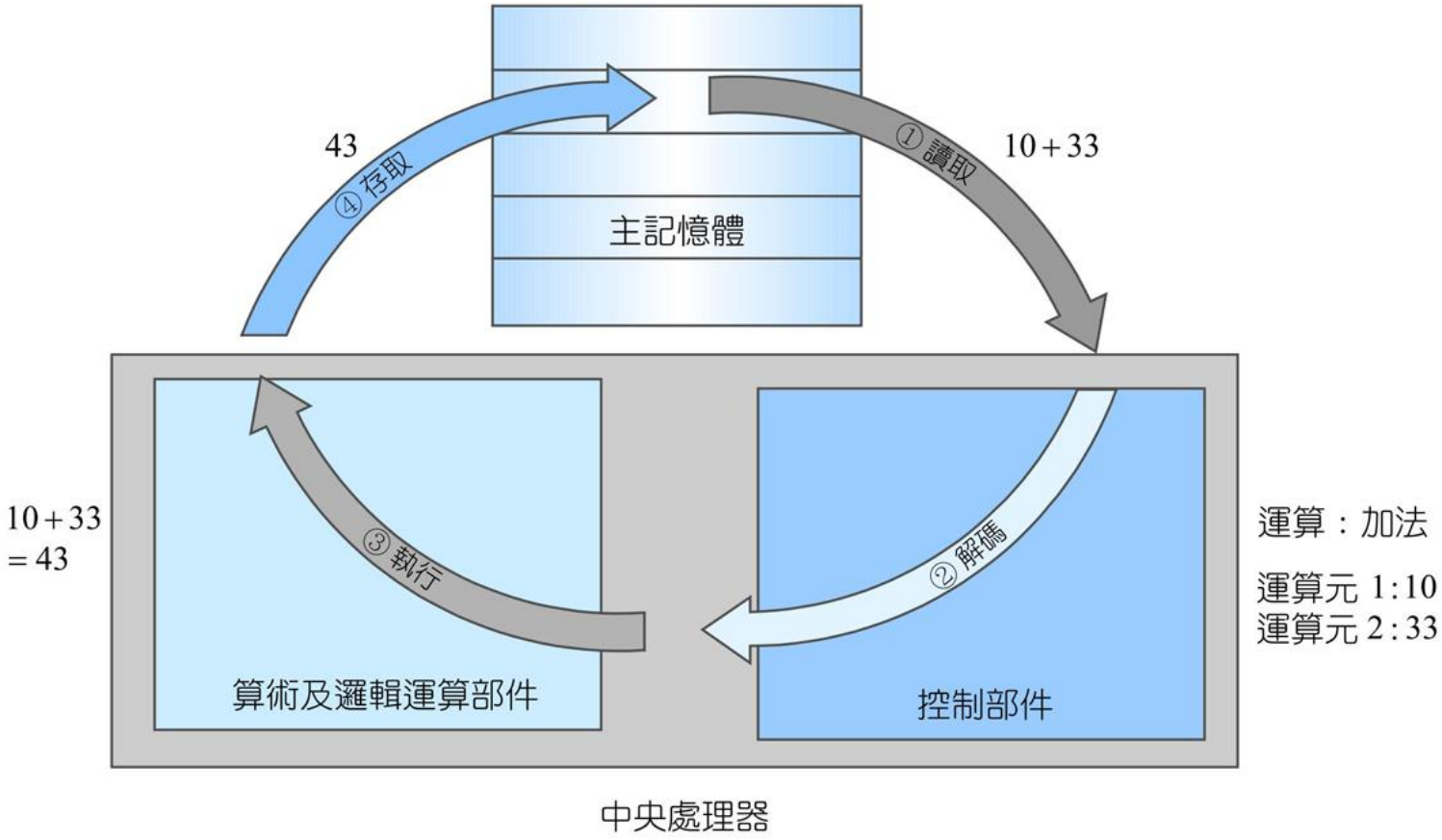
圖 4.10 (a) 點陣式印表機；(b) 噴墨印表機；(c) 雷射印表機 [圖片來源：維基共享資源 (a)、高立圖書 (b、c)]

程式執行流程

- 電腦的運作就是程式的執行
  - 程式要被執行之前都必須先載入到主記憶體內
  - 中央處理器會從主記憶體內將須執行的程式指令搬到暫存器中，並對程式指令進行指令解碼、執行與結果儲存的動作
  - 執行完畢後，會再重複上述的動作，至主記憶體中，將下一個須執行的指令搬到暫存器中並開始執行

- 中央處理器執行一個指令的過程稱為機器週期(machine cycle)，一個機器週期可分為讀取、解碼、執行與結果回存四個步驟

程式執行流程 (續)



- 指令讀取與指令解碼的總花費時間稱為指令時間(Instruction Time, I-Time)
- 指令執行與結果回存的總花費時間稱為執行時間 (Execution Time, E-Time)
- 一個機器週期即為 I-Time + E-Time。

數位邏輯設計簡介

- 布林代數 (Boolean Algebra)
  - 與一般代數的不同在於它只專注於處理 0 與 1 的數值
  - 代表著「真 (True)」與「偽 (False)」的觀念

1. 常數 (constant)：0 (偽) 與 1 (真)
2. 二元變數 (binary variable)：變數的值只能為 0 或 1
3. 運算子 (operator)：AND、OR、NOT 是三種最基本的運算子，AND 與 OR 是二元運算子、NOT 是單元運算子，其優先權為 NOT > AND > OR。

運算子	運算符號	運算式
NOT	—	$\overline{X}$
AND	·	$X \cdot Y$
OR	+	$X + Y$

數位邏輯設計簡介(續)

- 真值表
  - X、Y代表著變數
  - 最後一欄則代表著運算式(或稱布林函數)

表 4.3　NOT、AND、OR 的真值表

X	$\overline{X}$
0	1
1	0

X	Y	$X \cdot Y$
0	0	0
0	1	0
1	0	0
1	1	1

X	Y	$X+Y$
0	0	0
0	1	1
1	0	1
1	1	1

數位邏輯設計簡介 (續)

範例一

- 假設布林函數 $F(X,Y)=\overline{X}\cdot Y+X\cdot\overline{Y}$，那麼 $F(1, 0)$ 的值為何？
  - 將 $X=1$、$Y=0$ 帶入 $F(X,Y)$ 中

$$F(1,0)=\bar{1}\cdot 0+1\cdot\bar{0}$$

  - 運算子的優先權大小為 NOT > AND > OR

$$F(1,0)=0\cdot 0+1\cdot 1=0+1=1$$

數位邏輯設計簡介 (續)

範例二

- 請寫出布林函數 $F(X,Y,Z)=\overline{X}\cdot Y\cdot Z+\overline{Y}\cdot\overline{Z}$ 的真值表。
  - 將所有可能的輸入組合帶入此布林函數並運算出結果，再製作成對應的表格即可完成
  - 要算出 $F(0,0,0)\sim F(1,1,1)$ 共八種結果

$$F(0,0,0)=\overline{0}\cdot 0\cdot 0+\overline{0}\cdot\overline{0}=1\cdot 0\cdot 0+1\cdot 1$$
$$=0\cdot 0+1=0+1=1$$

表 4.4 $\overline{X}\cdot Y\cdot Z+\overline{Y}\cdot\overline{Z}$ 的真值表

X	Y	Z	F(X,Y,Z)
0	0	0	1
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	1
1	0	1	0
1	1	0	0
1	1	1	0

數位邏輯設計簡介 (續)

隨堂練習

- 請寫出布林函數 $F(X,Y,Z)=X\cdot Y\cdot \overline{Z}+X\cdot \overline{Y}\cdot Z+\overline{X}\cdot Y\cdot Z$ 的真值表。

解答：

- 依序算出 $F(0,0,0)$ ~ $F(1,1,1)$ 八種結果再製作成對應表格即可完成

$$F(0,0,0)=0\cdot 0\cdot \overline{0}+0\cdot \overline{0}\cdot 0+\overline{0}\cdot 0\cdot 0$$
$$=0\cdot 0\cdot 1+0\cdot 1\cdot 0+1\cdot 0\cdot 0$$
$$=0+0+0$$
$$=0$$

表 4.5　$X\cdot Y\cdot \overline{Z}+X\cdot \overline{Y}\cdot Z+\overline{X}\cdot Y\cdot Z$ 的真值表

X	Y	Z	F(X,Y,Z)
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	0

數位邏輯設計簡介 (續)

- 在邏輯設計中，我們可利用各式各樣的邏輯閘設計出電路

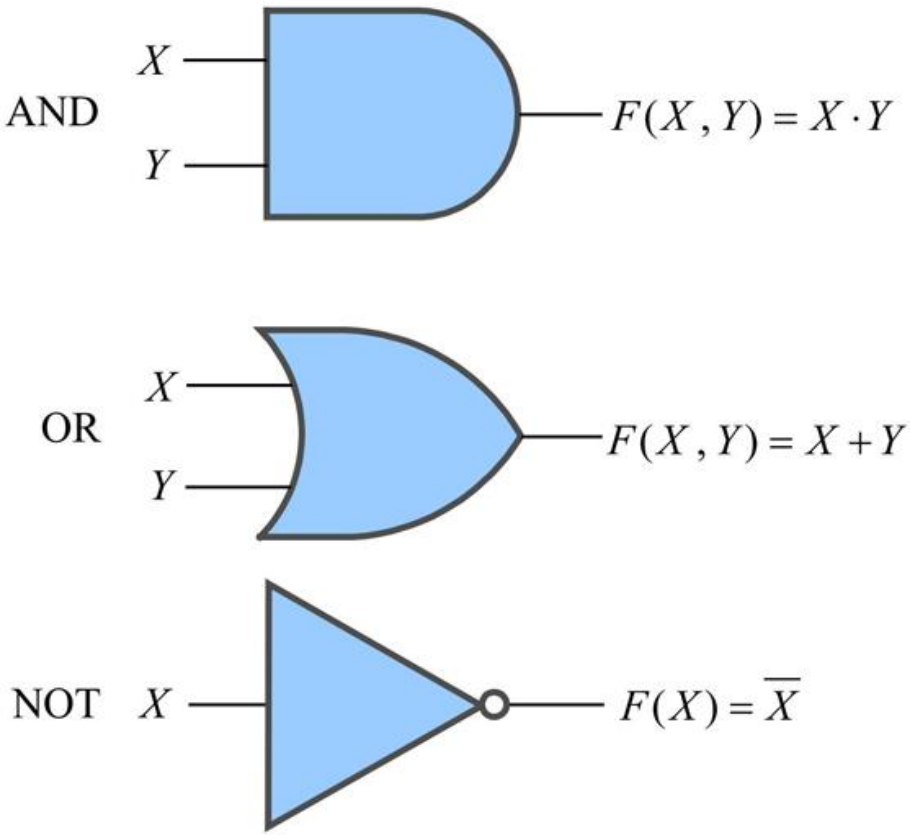


圖 4.12　AND、OR、NOT 邏輯閘

數位邏輯設計簡介 (續)

範例三

- Exclusive OR (XOR) 是一個常用的二元邏輯運算子，它的運算符號為⊕，當輸入的運算元一個為 1 一個為 0 時，結果則為 1，否則則為 0

表 4.6　$X \oplus Y$ 的真值表

X	Y	$X \oplus Y$
0	0	0
0	1	1
1	0	1
1	1	0

$$F(X,Y) = \overline{X} \cdot Y + X \cdot \overline{Y}$$

當「$X = 0$ 而且 $Y = 1$」或者「$X = 1$ 而且 $Y = 0$」的時候，結果會是 1

數位邏輯設計簡介 (續)

$$F(X,Y) = \overline{X} \cdot Y + X \cdot \overline{Y}$$

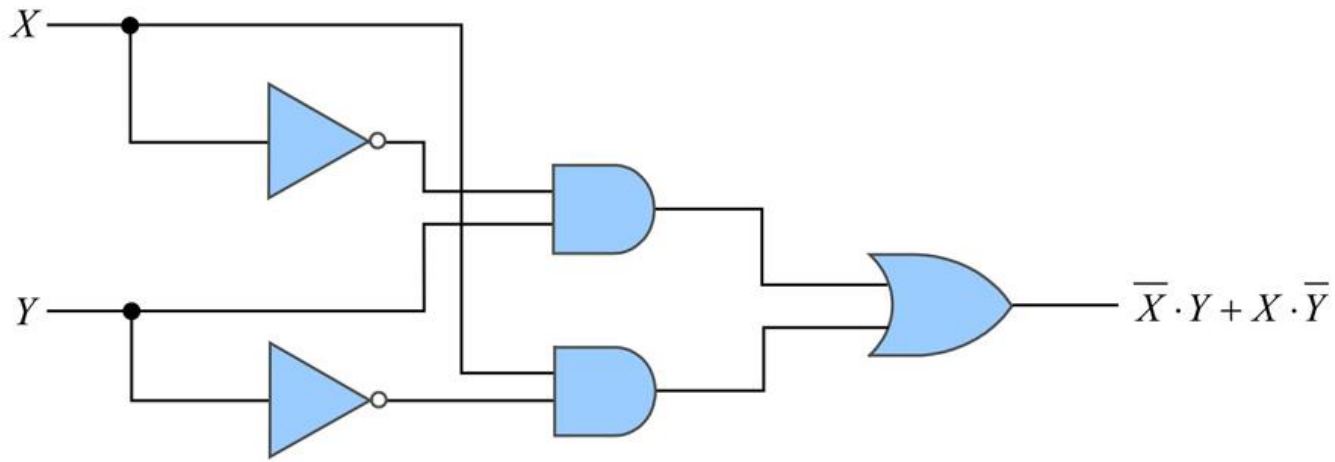


圖 4.13　利用 AND、OR、NOT 做出 XOR 的邏輯電路

數位邏輯設計簡介 (續)

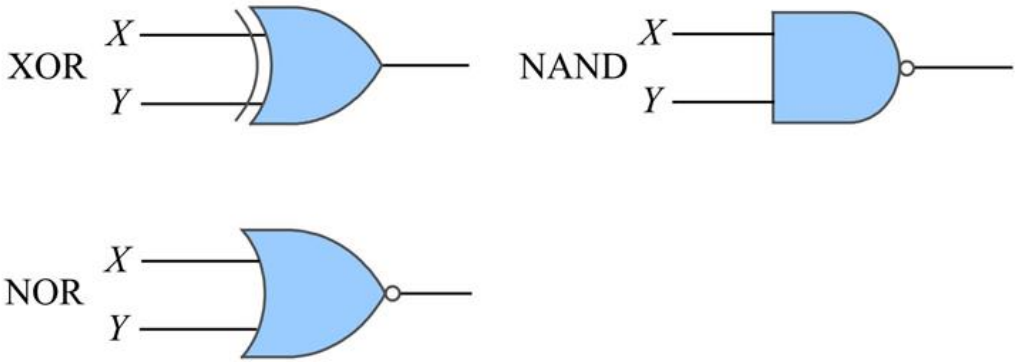


圖 4.14　XOR、NAND 與 NOR 邏輯閘

X	Y	$\overline{XY}$
0	0	1
0	1	1
1	0	1
1	1	0

X	Y	$\overline{X+Y}$
0	0	1
0	1	0
1	0	0
1	1	0

數位邏輯設計簡介 (續)

- 1 位元的加法

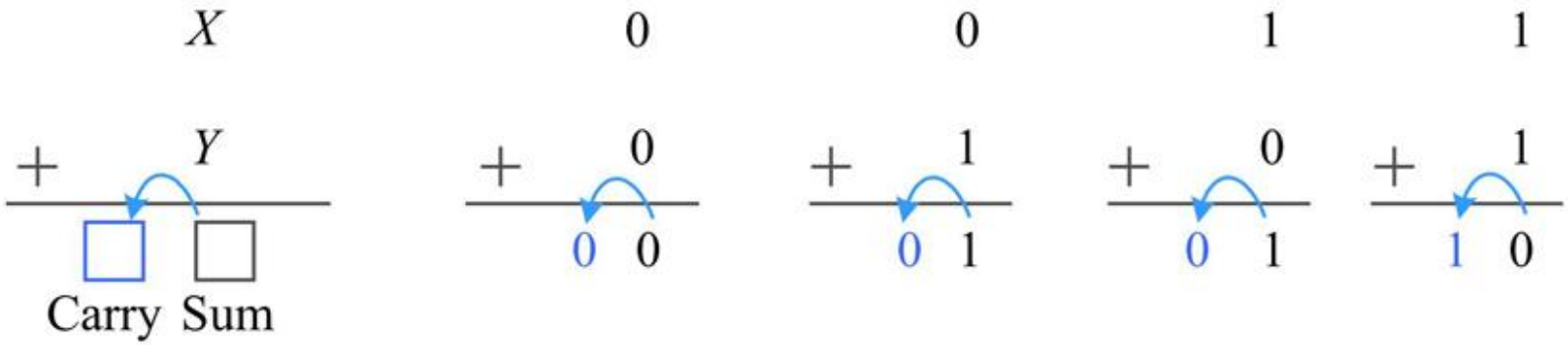


表 4.7　Sum 與 Carry 的真值表

X	Y	Sum	Carry
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

數位邏輯設計簡介 (續)

X	Y	Sum
0	0	0
0	1	1
1	0	1
1	1	0

$\overline{X} \cdot Y$
$X \cdot \overline{Y}$

也就是 $X \oplus Y$

$\Longrightarrow \text{Sum} = \overline{X} \cdot Y + X \cdot \overline{Y}$

X	Y	Carry
0	0	0
0	1	0
1	0	0
1	1	1

$X \cdot Y \Longrightarrow \text{Carry} = X \cdot Y$

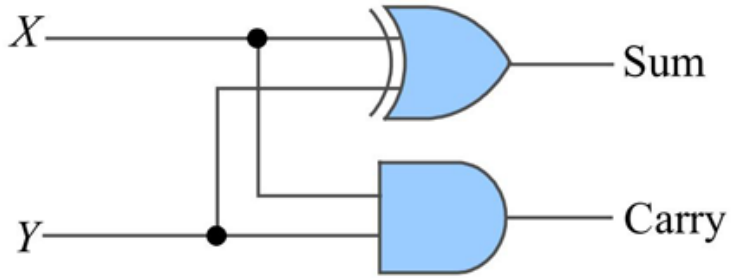


圖 4.16　1 位元加法的邏輯設計流程

數位邏輯設計簡介 (續)

- 全加法器。
  - 將前一個位元產生的進位稱為 C_{in}，加完後產生的進位稱為 C_{out}
  - 總共有三個輸入 (X、Y、C_{in}) 與兩個輸出 (Sum、C_{out})。

表 4.8 Sum 與 C_{out} 的真值表

X	Y	C_{in}	Sum	C_{out}
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

$$\text{Sum} = \bar{X}\cdot\bar{Y}\cdot C_{in} + \bar{X}\cdot Y\cdot\bar{C}_{in} + X\cdot\bar{Y}\cdot\bar{C}_{in} + X\cdot Y\cdot C_{in}$$

也就是 $X \oplus Y \oplus C_{in}$

$$C_{out} = \bar{X}\cdot Y\cdot C_{in} + X\cdot\bar{Y}\cdot C_{in} + X\cdot Y\cdot\bar{C}_{in} + X\cdot Y\cdot C_{in}$$

為了使 C_{out} 的線路更為精簡，我們利用一些化簡的方法，可得 $X\cdot Y + Y\cdot C_{in} + X\cdot C_{in}$ (化簡的方法不在此討論，讀者有興趣可參閱邏輯設計相關書籍)

數位邏輯設計簡介 (續)

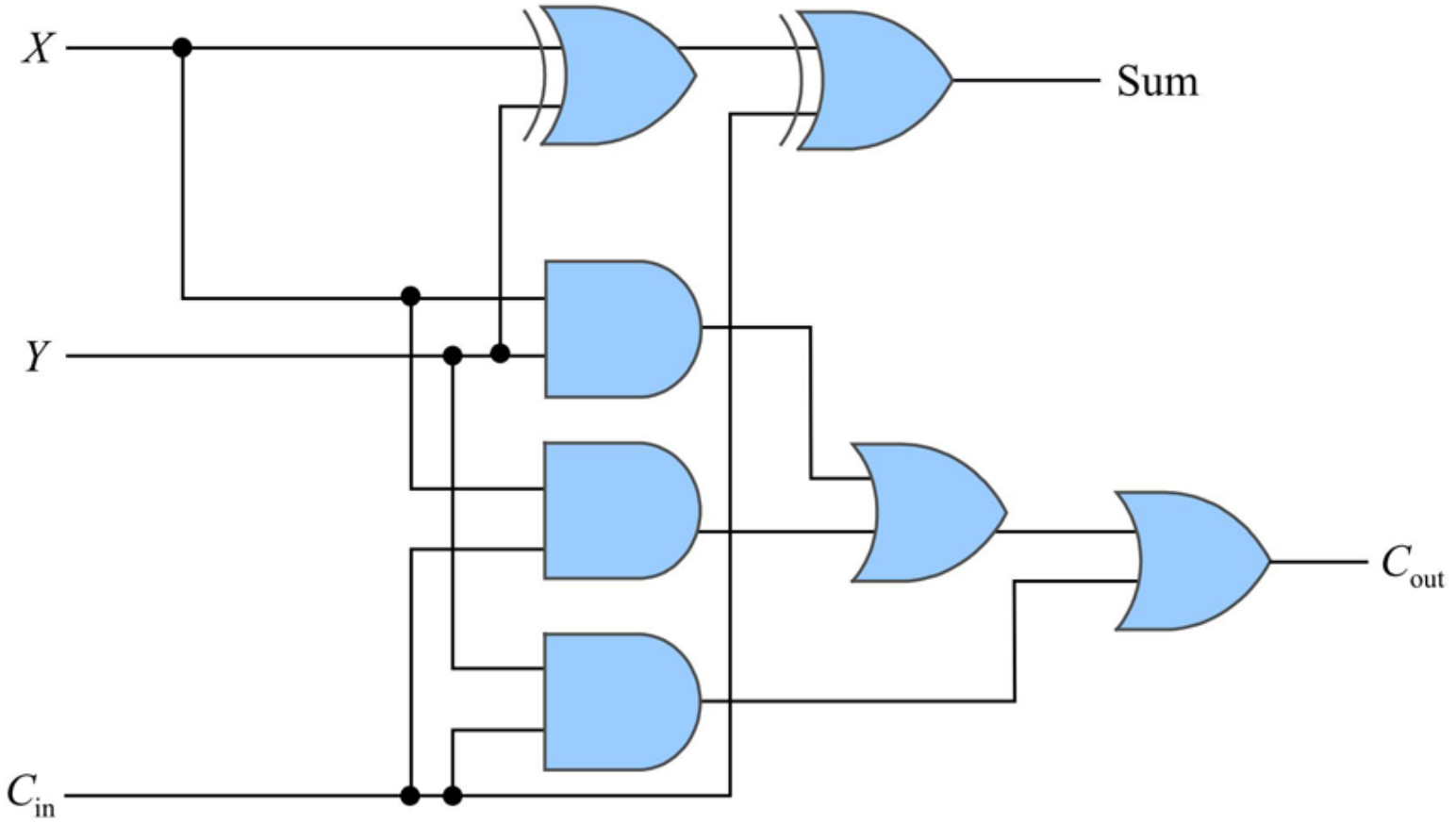


圖 4.17　全加法器的邏輯設計

數位邏輯設計簡介 (續)

- 有了全加法器，現在只要將多個全加法器串聯起來，就可以設計出多位元的加法器。

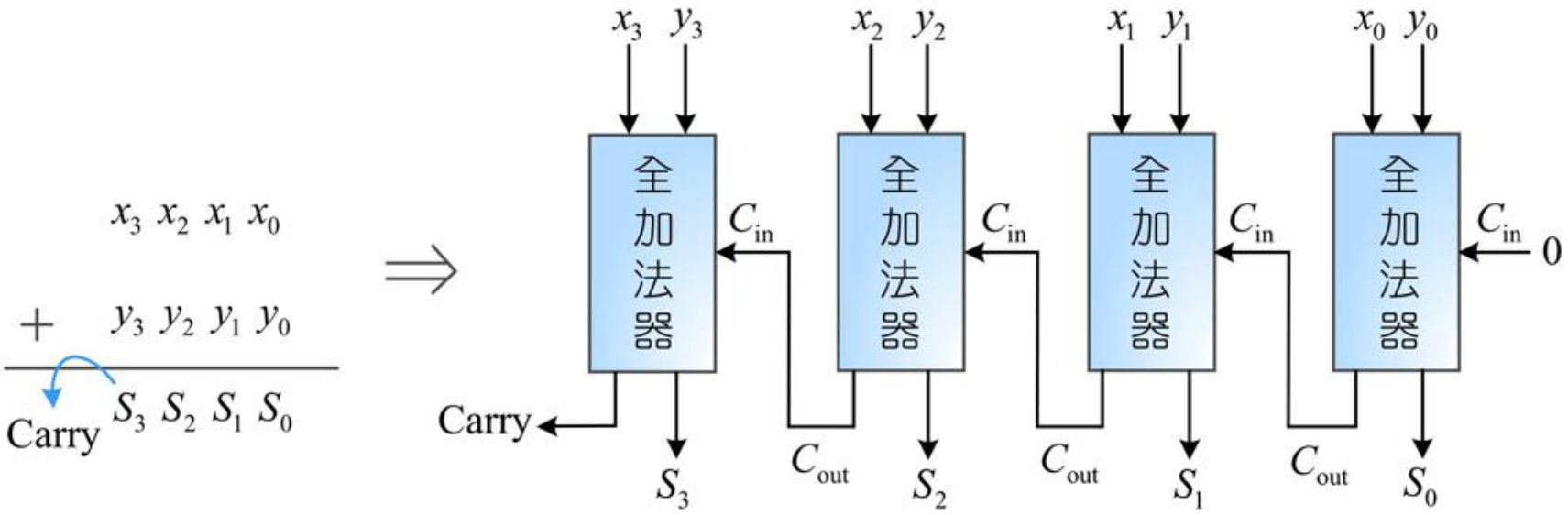


圖 4.18　4 位元加法器

數位邏輯設計簡介 (續)

隨堂練習

- 請設計出 4 位元的減法器。(提示：可以利用加法器與二的補數法觀念完成)

解答：

- 將 $x-y$ 改寫成 $x+(-y)$，也就是將 y 轉成負數後，再利用加法器完成加法即可將 $x-y$ 實踐。
- 一個數的二的補數就是「1 變 0、0 變 1、變完之後再加 1」
  - 將 y 轉成 $-y$，只需將每一個位元做反轉 (也就是 NOT)，並將加法器最低位的 C_{in} 設為 1 即可完成。

數位邏輯設計簡介 (續)

- **隨堂練習 (續)**

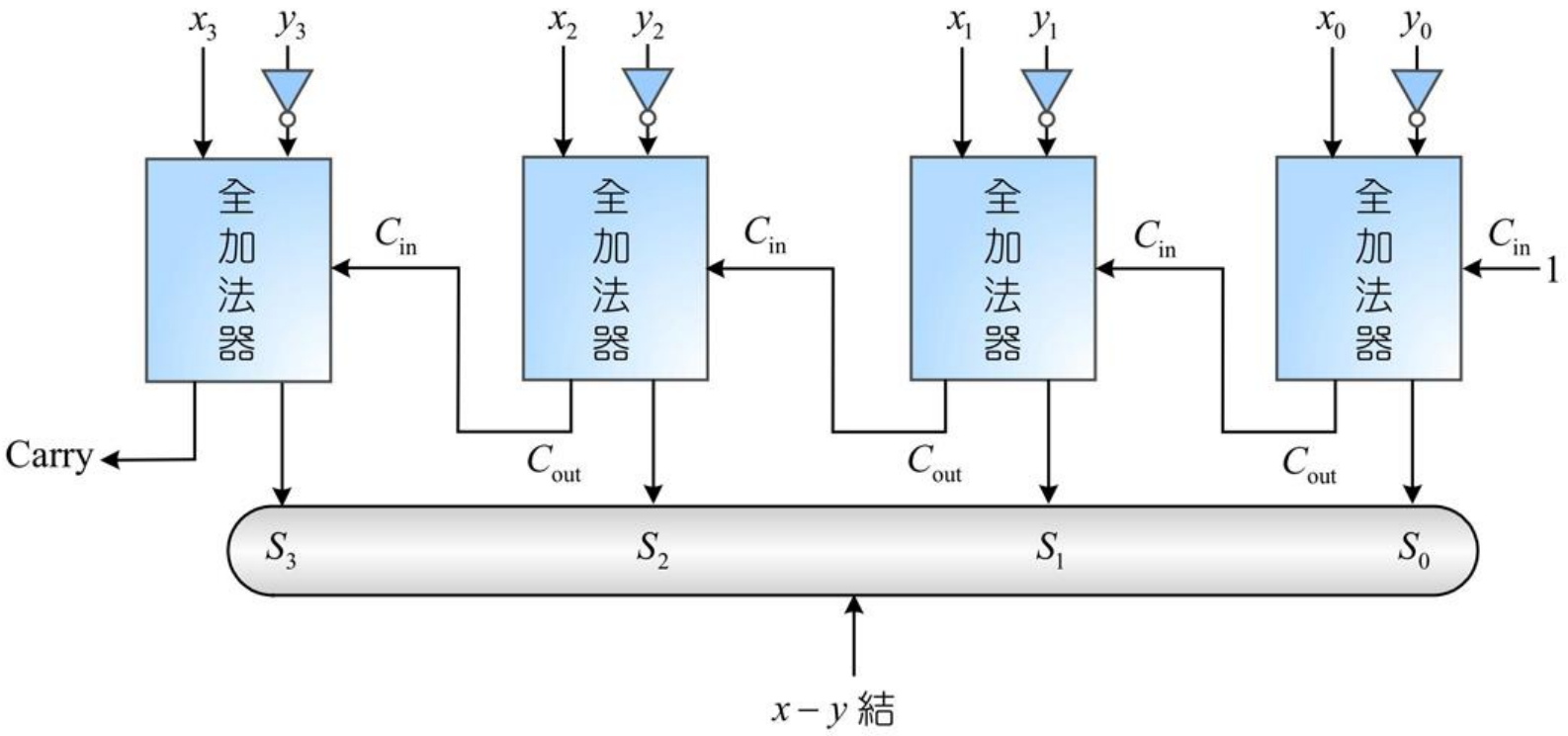


圖 4.19　4 位元減法器