組合邏輯電路設計

— 算術運算電路

概　　述

- 加、減、乘與除等四種算術運算在我們日常生活中，用來處理數值運算之基本方法，因此數位電路 (Digital Circuit) 亦應具有這些基本算術運算功能，才能用來幫助人類處理各種繁瑣之數值運算問題。
- 數位電路僅能用來處理二進位 (二元性) 資料，若引入補數 (Complement) 之觀念，則減法運算可用加法來取代，又乘法運算可用連續加法運算代替，而除法運算亦可用連續減法運算來取代，故加法器可為算術運算之基本電路。
- 本章首先將討論 1 位元加法器 (Adder) 之設計方法，再進一步說明如何串接 n 個 1 位元加法器來設計 n 位元二進位數並行加法電路。接下來討論使用補數之觀念，使用加法器來取代減法電路之設計方法。
- 接著討論如何直接設計減法器 (Subtractor) 與乘法器 (Multiplier)，以提所高算術運算電路之運算速度。最後討論將算術運算 (Arithmetic Operation) 與邏輯運算 (Logic Operation) 合併在同一個邏輯電路，稱為算術邏輯單元 (Arithmetic Logic Unit; ALU)，以實現多功能之組合邏輯函數。

加法器 (概述)

- 在數位系統中，加法器 (Adder) 可用來執行兩組二進位數之加法運算。
- 首先討論一種沒有考慮到進位傳輸 (Carry Propagation) 問題之加法電路，因此種加法器僅可執行兩個 1 位元之二進位數相加，故被稱為半加法器 (Half Adder)。若有考慮進位傳輸之問題，便可將加法器串接起來，以便執行更多位元加法運算，故這種加法器被冠以全加法器 (Full Adder) 之名稱。
- 全加法器亦僅能執行兩個 1 位元之二進位數相加，接著討論全加器串接方法，以設計可執行兩組兩個位元以上相加之加法電路，稱為並行加法器 (Parallel Adder)。若使用並行加法器直接執行 n 位元之加法運算，雖此種加法電路具構造簡單之優點，但運算速度較慢為最大缺點，故本節將提出前視進位加法器 (Look Ahead Carry Adder)，以改善並行加法器因進位造成運算速度延遲之缺點。
- 因人類習慣使用十進位執行算數運算，因此亦將於本節討論十進位加法器之設計方法，最後提出採用補數之觀念，以設計出可同時執行加、減法運算之組合邏輯電路。

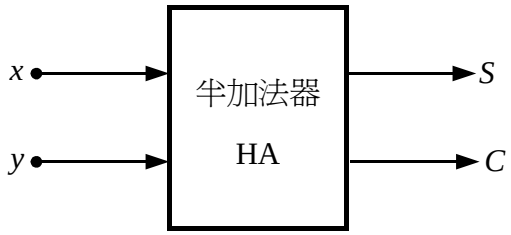
半加法器

◆ **半加法器** (Half Adder) 是一種組合邏輯電路，此電路僅可執行**兩組 1 位元**之二進位數的加法運算。接著列出兩個二進位數相加之**運算規則**如下：

$$\begin{array}{rrrr} 0 & 0 & 1 & 1 \leftarrow \text{被加數} \\ +\ 0 & +\ 1 & +\ 0 & +\ 1 \leftarrow \text{加數} \\ \hline 0\,0 & 0\,1 & 0\,1 & 1\,0 \end{array}$$

進位　和

◆ 半加法器有**加數與被加數**等**兩個輸入變數**，分別用 x 與 y 來標示，而兩數相加後會產生一個**和** (Sum) 與可能產生之**進位** (Carry) 等**兩個輸出變數**，分別標示為 S 與 C。



(a) 方塊圖

x	y	C	S
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0

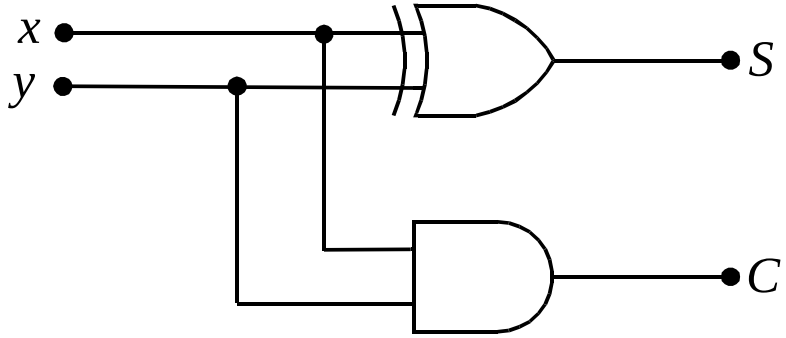
(b) 真值表

◆ 利用卡諾圖法對以上真值表進行邏輯化簡，可得化簡後之布林函數式為

$$S = \overline{x} \cdot y + x \cdot \overline{y} = x \oplus y$$

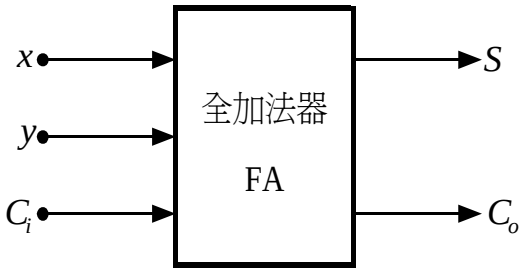
$$C = x \cdot y$$

◆ 最後使用邏輯閘來實現半加法器之邏輯電路，如下圖所示。



全加法器

- 因半加法器沒有考慮進位傳輸的問題，故無法執行兩組 2 位元以上的二進位數相加，即半加法器缺少一個進位輸入端，當兩個較低位元之二進位數相加後，若有進位產生時，進位輸入可作為傳送進位與上一位元相加之用。
- 若在半加法器上多加一個輸入變數，就便成了全加法器 (Full Adder)，故全加法器有加數、被加數與進位輸入等 3 個輸入變數，分別標示為 x 、y 與 C_i ，而此 3 個二進位數相加後，亦會產生一個和 (Sum) 與可能的進位 (Carry) 等 2 個輸出變數，分別標示為 S 與 C_o 。



(a) 方塊圖

x	y	C_i	C_o	S
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1

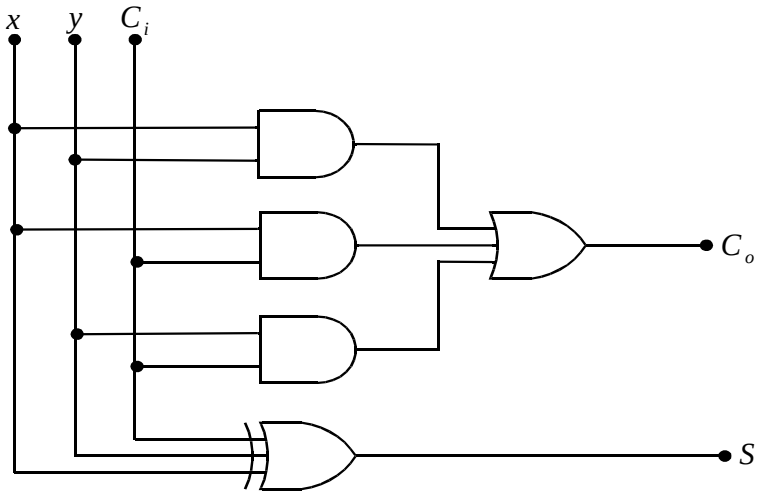
(b) 真值表

◆ 利用卡諾圖法對以上真值表進行邏輯化簡，可得簡化後之布林函數式為

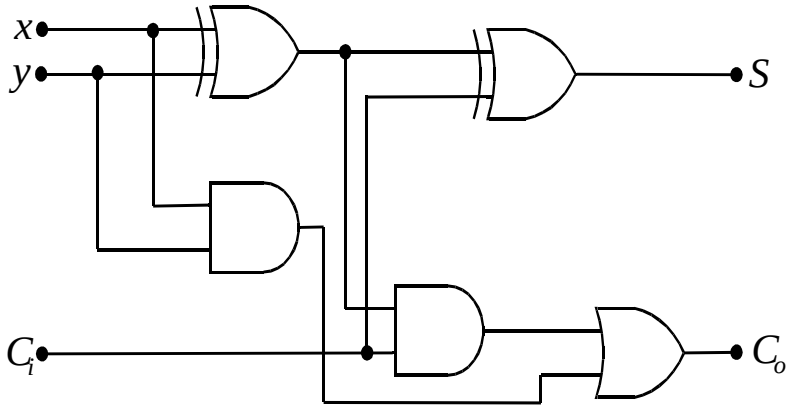
$$S = \bar{x} \cdot \bar{y} \cdot C_i + \bar{x} \cdot y \cdot \bar{C}_i + x \cdot \bar{y} \cdot \bar{C}_i + x \cdot y \cdot C_i = x \oplus y \oplus C_i$$

$$C_o = x \cdot y + x \cdot y \cdot C_i + \bar{x} \cdot y \cdot C_i = (x \oplus y) \cdot C_i + x \cdot y$$

◆ 最後將化簡所得之布林函數，使用邏輯閘來實現全加法器之邏輯電路，如下圖(a)所示。



(a)



(b)

註：上圖 (a) 與 (b) 皆可用來執行全法加器運算功能之邏輯電路圖，而這兩個電路之輸出 S 皆為 $S = x \oplus y \oplus C_i$，而為使全加法器之邏輯電路圖有不同的表示方式，進位輸出 C_o 採用邏輯功能相同，但表示形式不同之布林函數式，即圖 (a) 之進位輸出 $C_o = x \cdot y + x \cdot C_i + y \cdot C_i$，而圖 (b) 之進位輸出 $C_o = (x \oplus y) \cdot C_i + x \cdot y$。

二進位並行加法器

- 欲對兩組 n 位元之二進位數相加，必須串接 n 個全加器才能完成。若欲將兩組 4 位元之二進位數，被加數 $x = x_4x_3x_2x_1$ 與加數 $y = y_4y_3y_2y_1$ 執行加法運算規則如下：

$$\begin{array}{cccccc} & C_4 & C_3 & C_2 & C_1 & C_0 \\ & & x_4 & x_3 & x_2 & x_1 \\ + & & y_4 & y_3 & y_2 & y_1 \\ \hline & C_4 & S_4 & S_3 & S_2 & S_1 \end{array}$$

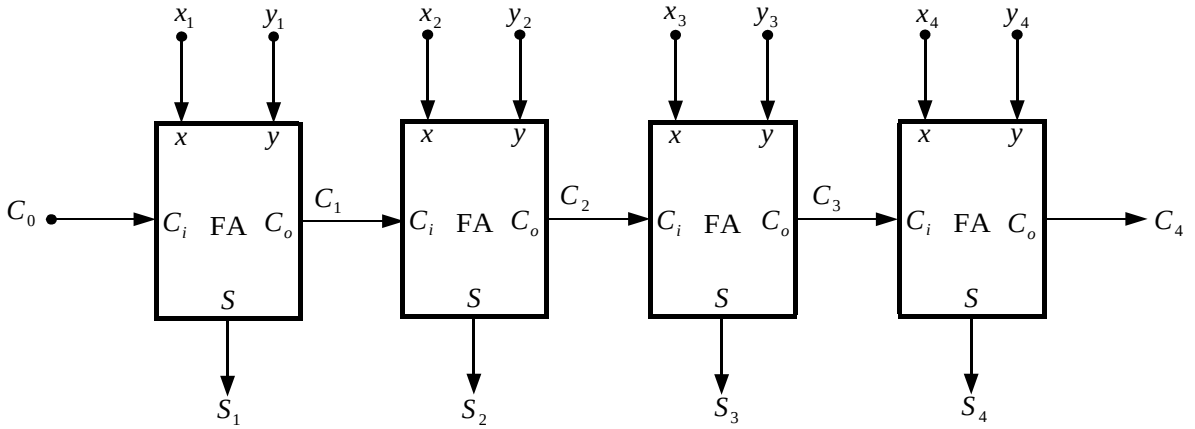
進位 ← $C_4\ C_3\ C_2\ C_1\ C_0$

被加數 ← $x_4\ x_3\ x_2\ x_1$

加數 ← $y_4\ y_3\ y_2\ y_1$

和 ← $C_4\ S_4\ S_3\ S_2\ S_1$

- 根據上面之運算式可知，並行加法電路之原理是將較低位元之加數與被加數相加所得之進位，直接傳送至較高位元之進位輸入端，再與較高位元之加數與被加數相加所得之進位，再直接傳送至更高位元之進位輸入端，依此類推，便可執行 n 個位元之加法運算，此種加法器稱漣波進位加法器 (Ripple Carry Adder)，亦可稱為並行加法器 (Parallel Adder)。

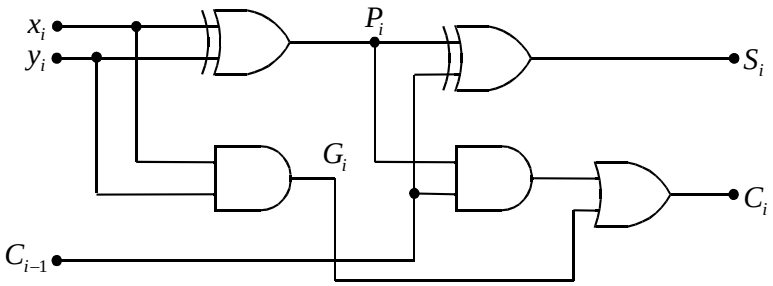


二進位前視進位加法器

- 並行加法器在進行加法運算時，雖然所有加數與被加數皆為已知，但較高位元之加法運算，必須等較低位元相加，得到穩定的進位訊號後，才能得到正確之結果，即加數與被加數通過邏輯閘之層數與加數、被加數之位元數成正比關係，因此會造成嚴重進位傳輸延遲之問題。雖然並行加法器之電路構造較為簡單，但運算速度會受到進位傳輸時間的限制而變慢。

- 為了加快算術運算速度（假設每個邏輯閘之傳輸延遲時間皆相同之條件下），可從改良電路著手，以減少等待進位所造成之傳輸延遲時間。而目前於減少進位傳輸延遲之方法非常多，其中最廣範使用之方法為前視進位加法 (Look-Ahead Carry) 原理。

- 首先根據下圖(全加法器)來定義與進位 (Carry) 有關的兩個變數，分別為進位產生 G_i (Carry Generator) 和進位傳輸 P_i (Carry Propagation) 於下：



$$G_i = x_i \cdot y_i \qquad P_i = x_i \oplus y_i$$

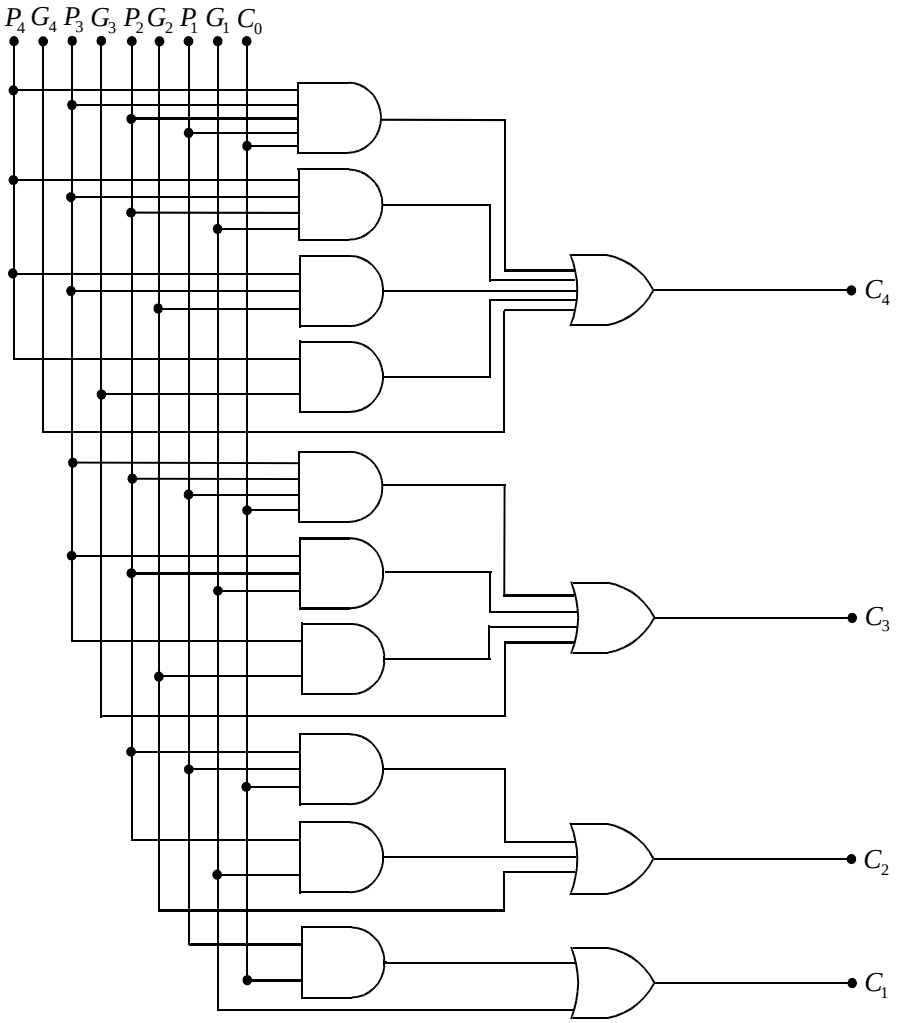
觀察左圖可得，全加法器之輸出 S_i 與 C_i 之布林函數式於下：

$$S_i = P_i \oplus C_{i-1} \qquad C_i = G_i + P_i \cdot C_{i-1}$$

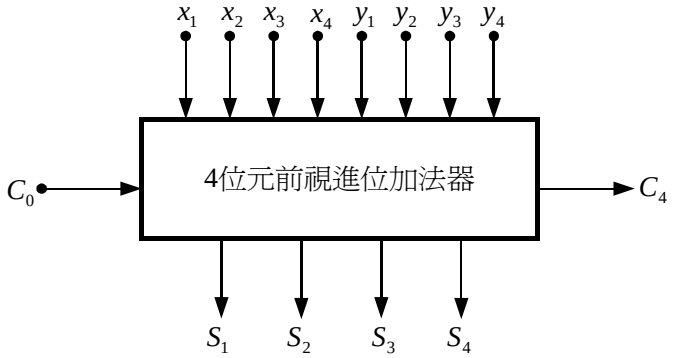
- 若以設計 4 位元加法器為例，將每一級的進位 C_i 用被加數 x_i、加數 y_i 與進位 C_0 表示如下：

$$C_1 = G_1 + P_1 \cdot C_0$$
$$C_2 = G_2 + P_2 \cdot C_1 = G_2 + P_2 \cdot G_1 + P_2 \cdot P_1 \cdot C_0$$
$$C_3 = G_3 + P_3 \cdot C_2 = G_3 + P_3 \cdot G_2 + P_3 \cdot P_2 \cdot G_1 + P_3 \cdot P_2 \cdot P_1 \cdot C_0$$
$$C_4 = G_4 + P_4 \cdot C_3 = G_4 + P_4 \cdot G_3 + P_4 \cdot P_3 \cdot G_2 + P_4 \cdot P_3 \cdot P_2 \cdot G_1 + P_4 \cdot P_3 \cdot P_2 \cdot P_1 \cdot C_0$$

- 若 被 加 數 $x = x_4x_3x_2x_1$ 與 加 數 $y = y_4y_2y_2y_1$ 分別為 4 位元之二進位數，觀察上面之進位C_i（i表示加數、被加數或進位位元之權位）之布林函數式，僅與被加數x_i、加數y_i與C_o有關。
- 對兩數之加法運算而言，加數與被加數應為已知數值，藉由較複雜之邏輯運算，先將每一級之進位C_i計算出來，便可省掉執行加法運算時，等待進位之時間，故可加快電路之運算速度。
- 接著利用基本邏輯閘，配合上述之布林函數式，即可繪出 4 位元之前視進位產生器之邏輯電路，如右圖所示。

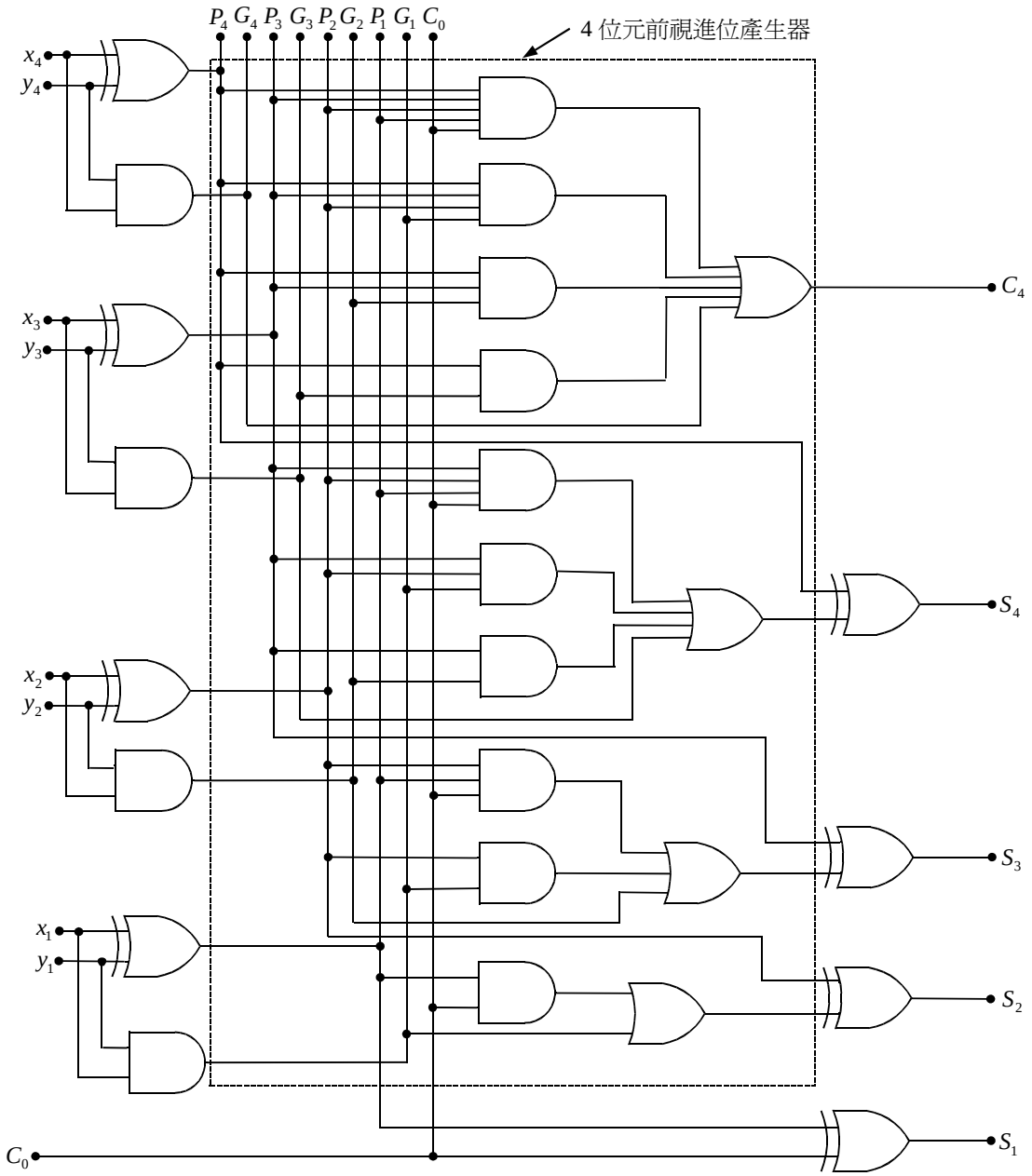


◆ 若組合進位產生 G_i 和進位傳輸 P_i 之邏輯電路，即可繪出 4 位元前視進位加法器之方塊圖與邏輯電路，分別如下、右圖所示。



(a) 方塊圖

◆ 雖此右圖之加法電路看起來相當複雜，在執行二進位數加法運算時，較高位元之相加，不必再等較低位元相加後之進位，便能得到正確相加結果，故可大大提高執行多位元加法電路之運算速度。



十進位加法器

◆ 一種以 4 個位元的二進位數來代表 1 個位元的十進位數之數碼稱為 BCD 碼 (Binary Code Decimal)，首先說明 BCD 碼與二進位碼間之差異，如下表所示。

十進位數	二進位碼	BCD 碼
0	0 0 0 0	0 0 0 0
1	0 0 0 1	0 0 0 1
2	0 0 1 0	0 0 1 0
3	0 0 1 1	0 0 1 1
4	0 1 0 0	0 1 0 0
5	0 1 0 1	0 1 0 1
6	0 1 1 0	0 1 1 0
7	0 1 1 1	0 1 1 1
8	1 0 0 0	1 0 0 0
9	1 0 0 1	1 0 0 1
10	1 0 1 0	1 0 0 0 0
11	1 0 1 1	1 0 0 0 1
12	1 1 0 0	1 0 0 1 0
13	1 1 0 1	1 0 0 1 1
14	1 1 1 0	1 0 1 0 0
15	1 1 1 1	1 0 1 0 1

◆ 因人類習慣用十進位數 (Decimal)來做算術運算，且大多數的數字顯示裝置，亦是以十進位數來顯示，以方便大眾閱讀。

◆ 觀察左表可知，當十進位數小於或等於 9 時，BCD 碼和二進位碼完全相同，而當十進位數大於 9 時，BCD 碼和二進位碼便不相同了。

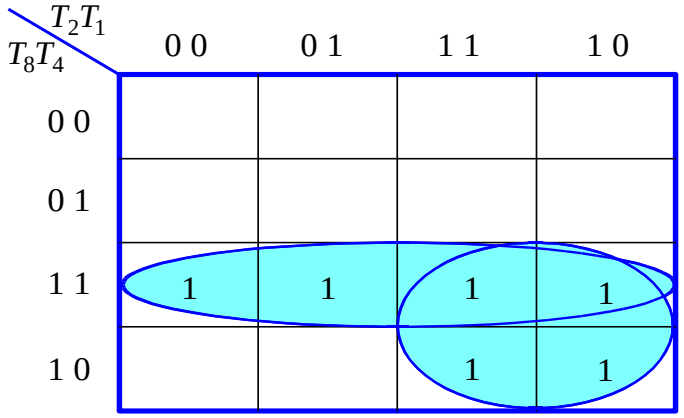
◆ 本節所討論之十進位加法器 (Decimal Adder) (亦可稱為 BCD 加法器) 是直接將兩個十進位數 (BCD 碼) 做加法運算，且相加後之結果亦直接以十進位數 (BCD 碼) 來表示，故兩 BCD 數相加後之和若超過 9，必須進行進位修正，才不會產生錯誤之 BCD 和。

	十進位數	二 進 位 和					BCD 和				
		K	T_8	T_4	T_2	T_1	C_4	S_8	S_4	S_2	S_1
A 部份	0	0	0	0	0	0	0	0	0	0	0
	1	0	0	0	0	1	0	0	0	0	1
	2	0	0	0	1	0	0	0	0	1	0
	3	0	0	0	1	1	0	0	0	1	1
	4	0	0	1	0	0	0	0	1	0	0
	5	0	0	1	0	1	0	0	1	0	1
	6	0	0	1	1	0	0	0	1	1	0
	7	0	0	1	1	1	0	0	1	1	1
	8	0	1	0	0	0	0	1	0	0	0
	9	0	1	0	0	1	0	1	0	0	1
B 部份	10	0	1	0	1	0	1	0	0	0	0
	11	0	1	0	1	1	1	0	0	0	1
	12	0	1	1	0	0	1	0	0	1	0
	13	0	1	1	0	1	1	0	0	1	1
	14	0	1	1	1	0	1	0	1	0	0
	15	0	1	1	1	1	1	0	1	0	1
C 部份	16	1	0	0	0	0	1	0	1	1	0
	17	1	0	0	0	1	1	0	1	1	1
	18	1	0	0	1	0	1	1	0	0	0
	19	1	0	0	1	1	1	1	0	0	1

- 每一個位元之十進位數值不會超過 9，所以兩個十進位數相加，再加上可能的進位輸入值，可能之最大值為 $9+9+1=19$ 。如果利用 4 位元之二進位加法器來執行兩個十進位數字相加，所得十進數之範圍為 0~19，如左表所示。
- 一個完整的 BCD 加法器，必須考慮 9 個輸入變數(包括兩組 4 位元之 BCD 碼與 1 位元之進位輸入)與 5 個輸出變數(包括 4 個位元之和與 1 個進位輸出)，若採用組合邏輯電路之設計步驟，以設計十進位加法器，則會變得相當複雜且不可行，故必須尋找兩組 BCD 數碼之加法規則，以演算法 (Algorithm) 來設計電路，才能簡化 BCD 加法器之設計問題。

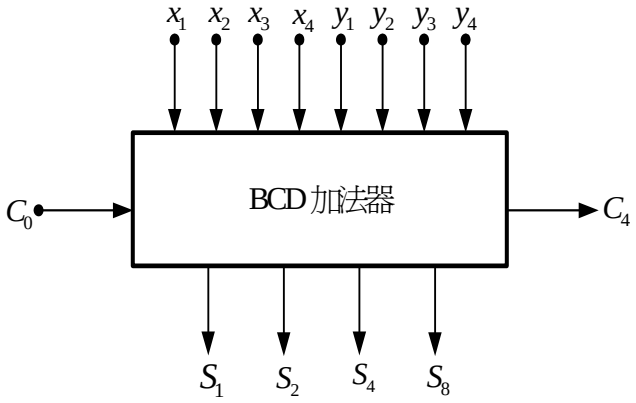
◆ 根據 BCD 碼與二進位碼之關係，分成 3 個部份來說明設計十進位加法器之演算法如下：

1. 當二組 BCD 和 $(T_8T_4T_2T_1)$ 小於或等於 9 時，BCD 碼與二進位碼相同，故所得之二進位和不必做任何進位修正。
2. 當二組 BCD 和 $(T_8T_4T_2T_1)$ 大於 9，且小於或等於 15 時，因這部分之 BCD 碼與二進位碼不同，必須將所得之二進位和再加十進位數之 6 (二進位數之 0110)，以跳過 6 個 BCD 碼沒有使用之二進位數，以做為進位修正之用。而需要進行進位修正之情況，可經由右邊之卡諾圖(即將 1010~1111 方格設定為邏輯 1) 化簡得到之布林函數式為 $f' = T_8 \cdot T_4 + T_8 \cdot T_2$。

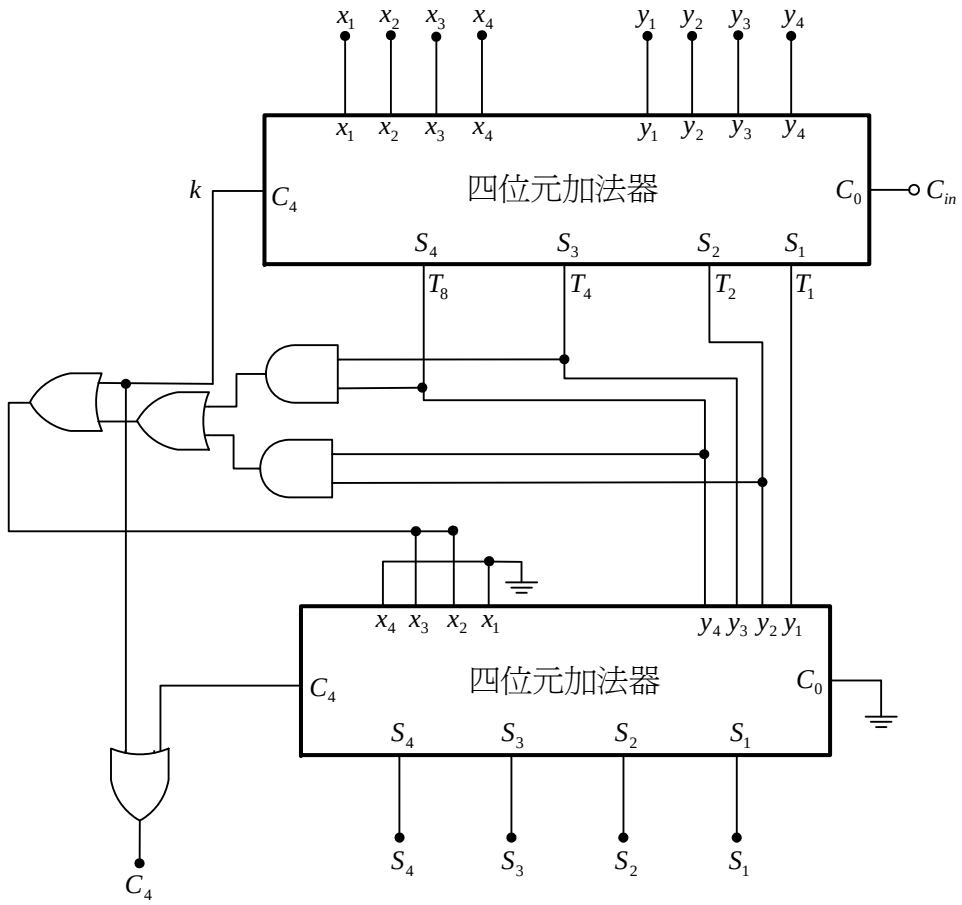


3. 當二組 BCD 和 $(T_8T_4T_2T_1)$ 大於 15 時，因此部分之 BCD 碼亦與二進位碼不同，故亦必須將相加所得之二進位和再加十進位數之 6 (二進位數之 0110)，來跳過 6 個 BCD 碼沒有使用之二進位碼，以做為進位修正之用。觀察上表可知，當 $K = 1$ (二進位和大於 15 之部分) 時，需要進行進位修正，故可得布林函數式為 $f'' = K$ 。

◆ 綜合以上 3 點之討論可得需要進位修正之布林函數式為 $K + T_8T_4 + T_8T_2$ 。經由以上之討論，即可繪出十進位加法器之方塊圖與邏輯電路，如下圖所示。



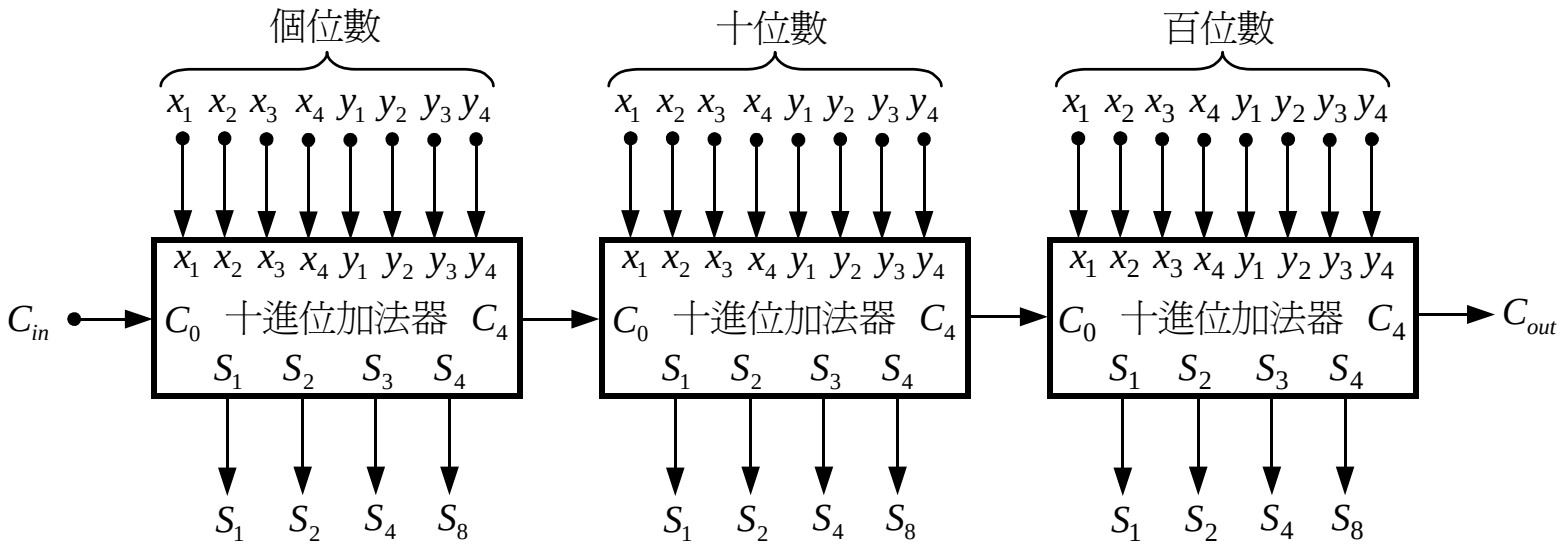
(a) 方塊圖



(b) 邏輯電路圖

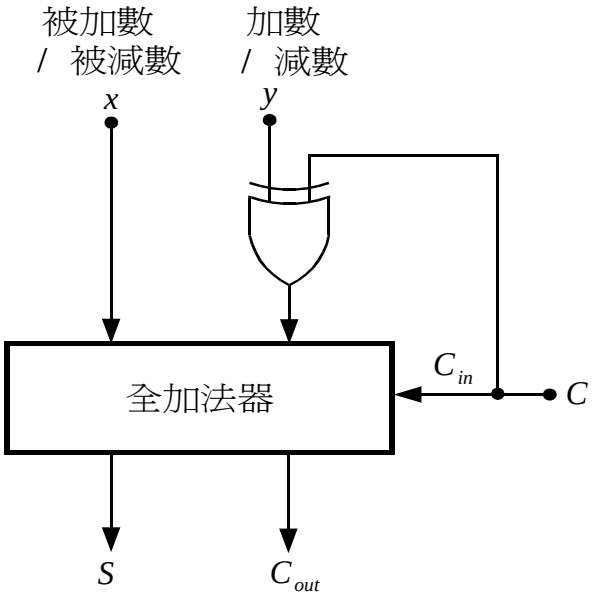
◆ 上圖僅能執行兩組 1 位元之十進位數的加法運算，而需要執行兩個位元以上之十進位數加法運算時，常常需串接兩個或兩個以上之十進位加法器，以便執行更多十進位數加法運算。

◆ 將兩個以上之十進位加法器進行串接時，可將較低位元相加後之進位輸出，直接傳輸至較高位元加法器的進位數入，即可完成串接工作。而下圖的十進位加法電路，便可用來執行兩組百位以下之十進位數的加法運算。依此類推，便可得到更多位數之十進位加法運算電路。



二進位加 / 減法器(單一位元)

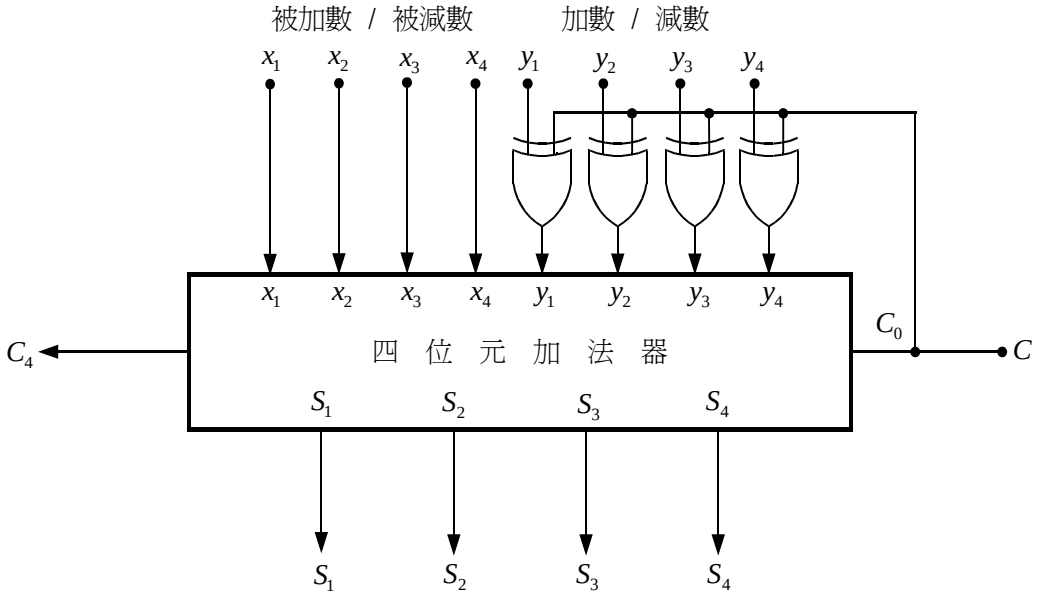
- 對兩組二進位數 (x 與 y) 執行加法 ($x+y$) 或減法 ($x-y=x+y_{2'S}$) 運算，皆需使用到加法運算，故全加法器 (Full Adder) 為必要使用之邏輯元件。
- 設計之二進位加 / 減法算術運算電路，欲執行加法運算時，則加數亦須保持不變，直接使用加法器將被加數 x 與加數 y 相加；當欲執行減法運算時必須對減數取 2 的補數 ($2's=1's+1$，可使用 XOR 閘來取得 1 的補數) 後，再使用被加數 x 與加輸 y 之 2 的補數相加。根據以上之討論，即可實現執行兩個 1 位元之二進位加 / 減法運算電路，如下圖所示。



- 1 位元之二進位加 / 減法運算電路之邏輯功能如下：
  1. 當控制 $C=0$ 時，全加法器之輸入端 (x、y 與 C_{in}) 分別為 x、y 與 0，則此電路將執行 $x+y$ (當 $C=0$ 時，$y\oplus 0=y$)之算術運算。
  2. 當控制 $C=1$ 時，全加法器之輸入端 (x、y 與 C_{in}) 分別為 x、$\bar{y}$ 與 1，則此電路將執行 $x+y_{2'S}$ (當 $C=1$ 時，$y\oplus 1=\bar{y}$，再加上進位輸入 $C_{in}=1$，即 $y_{2'S}=\bar{y}+1$) 之算術運算，即此時電路將執行 $x-y$ 之算術運算。

二進位加 / 減法器(多位元)

◆ 前面所討論之 1 位元加 / 減法算術運算電路，亦可擴充至 n 位元加 / 減法算術運算電路。假設某兩組二進位數分別為 $x = x_4x_3x_2x_1$ 與 $y = y_4y_3y_2y_1$，則可執行兩組 4 位元(x 與 y)加 / 減法算術運算電路，如下圖所示。



◆ 觀察右圖可知，4 位元(x 與 y) 加 / 減法算術運算電路之邏輯功能如下：

1. 當控制 $C = 0$ 時，4 位元加法器之輸入端 (x、y 與 C_{in}) 分別為 $x = x_4x_3x_2x_1$ 、 $y = y_4y_3y_2y_1$ 與 0，則此電路將執行 $x + y$ (當 $C = 0$ 時， $y \oplus 0 = y$)之算術運算。

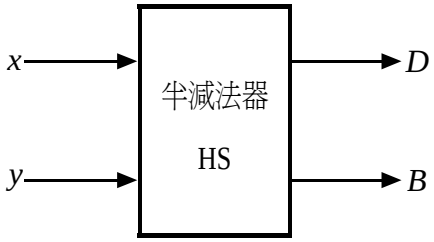
2. 當控制 $C = 1$ 時，4 位元加法器之輸入端 (x、y 與 C_{in}) 分別為 $x = x_4x_3x_2x_1$ 、 $\bar{y} = \bar{y}_4\bar{y}_3\bar{y}_2\bar{y}_1$ 與 1，則此電路將執行 $x + y_{2'S}$ (當 $C = 1$ 時， $y \oplus 1 = \bar{y}$，再加上進位輸入 $C_{in} = 1$，即 $y_{2'S} = \bar{y} + 1$) 算術運算，此時電路將執行 $x - y$ 之算術運算。

減法器(半減法器)

- **減法器**可用來執行兩個二進位數相減之算術運算，而沒有考慮到**借位傳輸** (Borrow Propagation) 問題之減法電路，因這種減法器僅可執行 1 位元之二進位數相減，故被稱為**半減器** (Half Subtractor)。而兩組二進位數相減之運算規則如下：

0	0	1	1 ← 被減數
− 0	− 1	− 0	− 1 ← 減數
0 0	1 1	0 1	0 0
			借位 差

- 半減法器有**減數**與**被減數**等兩個輸入變數，分別用 *x* 與 *y* 來標示，兩數相減後會產生一個**差** (Difference) 與可能產生之**借位** (Borrow) 等兩個輸出變數，分別標示為 *D* 與 *B*。



(a) 方塊圖

x	*y*	*B*	*D*
0	0	0	0
0	1	1	1
1	0	0	1
1	1	0	0

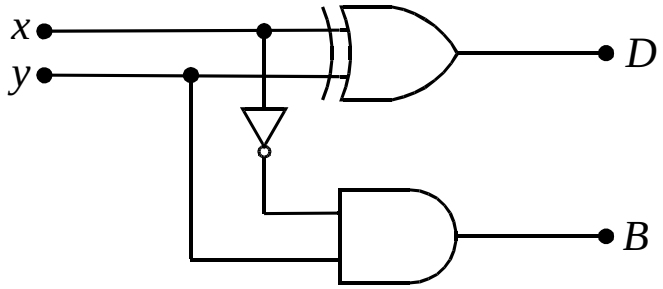
(b) 真值表

◆ 利用卡諾圖法對半減器之真值表進行邏輯化簡，可得簡化後之布林函數式如下：

$$D = \overline{x} \cdot y + x \cdot \overline{y} = x \oplus y$$

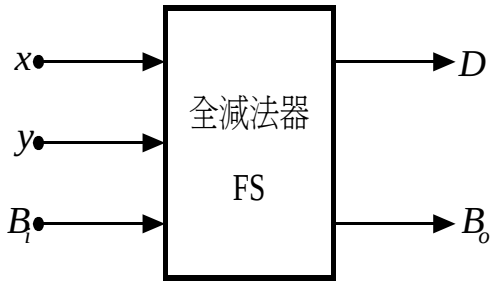
$$B = \overline{x} \cdot y$$

◆ 使用邏輯閘來實現半減法器之邏輯電路，如下圖所示。



減法器(全減法器)

◆ 因考慮借位傳輸的問題，全減法器有被減數、減數與借位輸入等 3 個輸入變數，分別標示為 x、y 與 B_i，而這 3 個輸入之二進位數相減後，會產生一個差 (Difference) 與可能產生一個借位 (Borrow) 等兩個輸出變數，分別標示為 D 與 B_o。當得知輸入與輸出之變數後，即可繪出全減法器之方塊圖與真值表，如下圖所示。



(a) 方塊圖

x	y	B_i	D	B_o
0	0	0	0	0
0	0	1	1	1
0	1	0	1	1
0	1	1	0	1
1	0	0	1	0
1	0	1	0	0
1	1	0	0	0
1	1	1	1	1

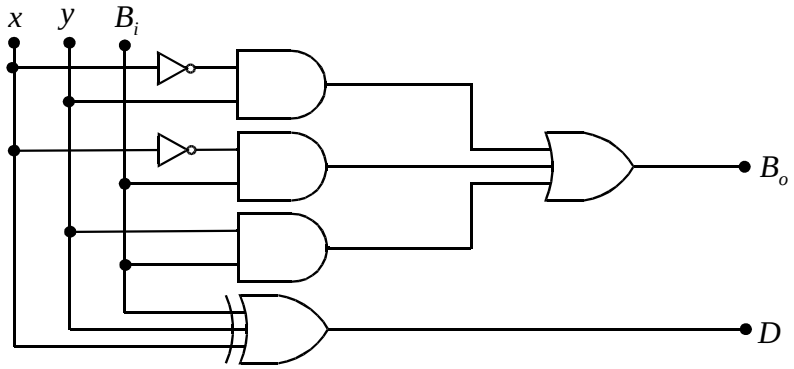
(b) 真值表

◆ 利用卡諾圖法對上圖之真值表進行**邏輯化簡**，可得簡化後之**布林函數式**為

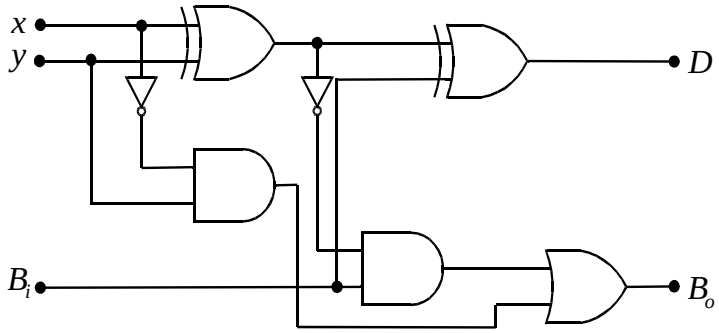
$$D = \bar{x} \cdot \bar{y} \cdot B_i + \bar{x} \cdot y \cdot \overline{B_i} + x \cdot \bar{y} \cdot \overline{B_i} + x \cdot y \cdot B_i = x \oplus y \oplus B_i$$

$$B_o = \bar{x} \cdot y + \bar{x} \cdot B_i + y \cdot B_i = \overline{(x \oplus y)} \cdot B_i + \bar{x} \cdot y$$

◆ 使用**邏輯閘**來實現全加法器之**邏輯電路**，如下圖所示。



(a)



(b)

註：上圖 (a) 與 (b) 皆可用來執行全減器運算功能之邏輯電路圖，而這兩個電路之輸出 D 皆為 $D = x \oplus y \oplus B_i$，而為使全減法器之邏輯電路圖有不同之表示方式，輸出 B_o 採用邏輯功能相同，但**表示形式不同之布林函數式**，即圖 (a) 之輸出 $B_o = \bar{x} \cdot y + \bar{x} \cdot B_i + y \cdot B_i$，而圖(b) 之輸出 $B_o = \overline{(x \oplus y)} \cdot B_i + \bar{x} \cdot y$ 。

十進位加減法器(9 的補數產生器)

- 使用 10 的補數之運算規則，以設計使用十進位加法器來取代十進位減法運算之方法。假設被減數 $x = x_4x_3x_2x_1$ 與減數 $y = y_4y_3y_2y_1$ 均為 4 位元之 BCD 碼，欲使用十進位加法器來取代十進位減法運算須對減數 y 取 10 的補數後，再與被減數 x 相加，即可執行 $x - y = x + y_{10's}$ 之算術運算。

- 為實現十進位加 / 減法運算電路，首先應設計 9 的補數產生器，因 9 的補數產生器是一種組合邏輯電路，故其設計步驟如下：

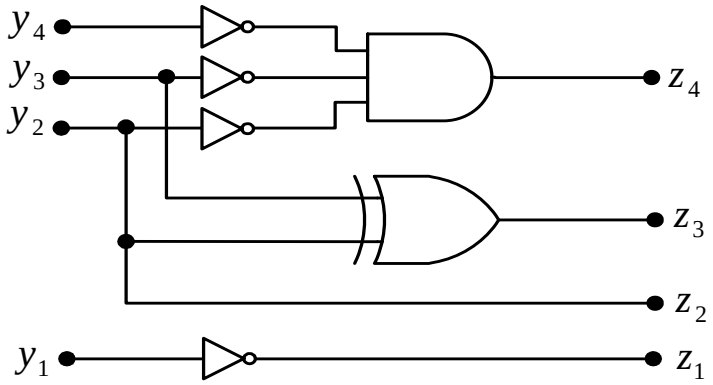
  1. 因電路之輸入為 BCD 碼，故所設計之電路有 4 個輸入變數，分別標示為 y_4 、 y_3 、 y_2 與 y_1 等 4 個符號；而輸出為輸入之 9 的補數，故輸出亦應有 4 個輸出變數，分別以 z_4 、 z_3 、 z_2 與 z_1 等符號來標示。

  2. 因輸出為輸入之 9 的補數，故可用真值表表來定義輸出與輸入之關係。

3. 利用卡諾圖化簡所得之真值表，以求得 z_4 、 z_3 、 z_2 與 z_1 之最簡布林函數式如下：

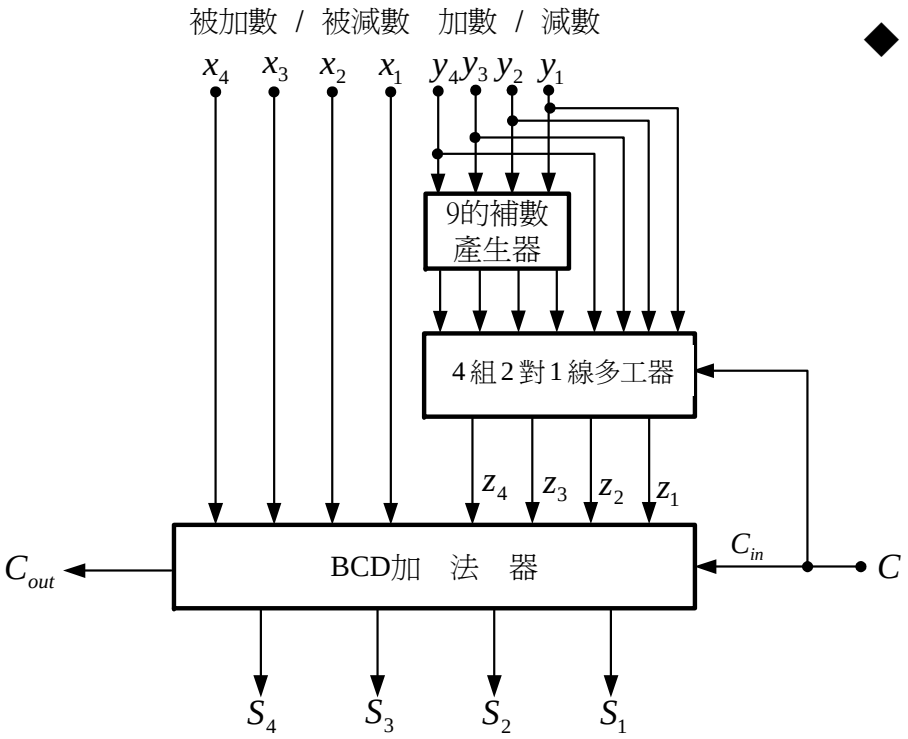
$$z_4 = \bar{y}_4 \cdot \bar{y}_3 \cdot \bar{y}_2$$
$$z_3 = \bar{y}_3 \cdot y_2 + y_3 \cdot \bar{y}_2 = y_3 \oplus y_2$$
$$z_2 = y_2$$
$$z_1 = \bar{y}_1$$

4. 最後使用邏輯閘來實現 z_4 、 z_3 、 z_2 與 z_1 之布林函數式，即可得 9 的補數產生器之組合邏輯電路，如下圖所示。



十進位加 / 減法器

◆ 設計之十進位加 / 減法算術運算電路，欲執行加法運算時，則被加數 x 與加數 y 必須保持不變；而欲執行減法運算時，被減數 x 必須保持不變，而對減數 y 取 10 的補數。根據以上之討論，即可得執行兩個 BCD 碼 (x 與 y) 之十進位加 / 減法算術運算電路，如下圖所示。



◆ 觀察左圖可知，十進位加 / 減法算術運算電路之邏輯運算功能如下：

1. 當選擇 $C=0$ 時，BCD 加法器之輸入端 (x、z 與 C_{in}) 分別為 x、y 與 0，則此電路將執行 $x+y$ 之十進位 (BCD 碼) 算術運算。

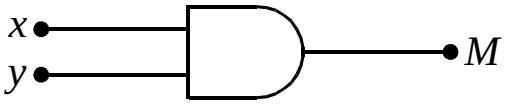
2. 當選擇 $C=1$ 時，BCD 加法器之輸入 (x、y 與 C_{in})分別為 x、$y_{9's}$ 與 1，則此電路將執行 $x+y_{10's}$ (當 $C=1$ 時，$z=y_{9's}$ 再加上進位輸入 $C_{in}=1$，即 $y_{10's}=y_{9's}+1$) 算術運算，此時電路將執行 $x-y$ 之十進位 (BCD 碼)算術運算。

二進位乘法器

- 由於積體電路製造技術的持續進步，目前算術電路中之**乘法運算**，皆直接用**乘法器** (Multiplier) 來完成，而**不再使用加法器來代替**，以加快算術電路之運算速度。
- 二進位數之乘法原理與十進位相同，必須考慮各個位元相乘後之**加權位置**，因**乘法可視為連續之加法**，二進位乘法器亦是依此原理設計出來的。
- **乘法器** (Multiplier) 是一種**組合邏輯電路**，而兩個 1 位元之二進位數 x 與 y 之乘法運算規則如下：

$$\begin{array}{r} 0 \\ \times\ 0 \\ \hline 0 \end{array} \qquad \begin{array}{r} 0 \\ \times\ 1 \\ \hline 0 \end{array} \qquad \begin{array}{r} 1 \\ \times\ 0 \\ \hline 0 \end{array} \qquad \begin{array}{r} 1 \\ \times\ 1 \\ \hline 1 \end{array}$$

- 觀察上面之乘法運算規則，**兩個 1 位元之二進位數乘法運算與兩個輸入的 AND 閘之邏輯運算（真值表）完全相同**，故兩個 **1 位元**之二進位（分別標示為 x 與 y 之符號）**乘法電路**，可用 **AND 閘**來實現，且兩數相乘後之輸出僅有 1 個，使用 M 之符號來標示，如下圖所示。



2 位元與 2 位元之乘法器

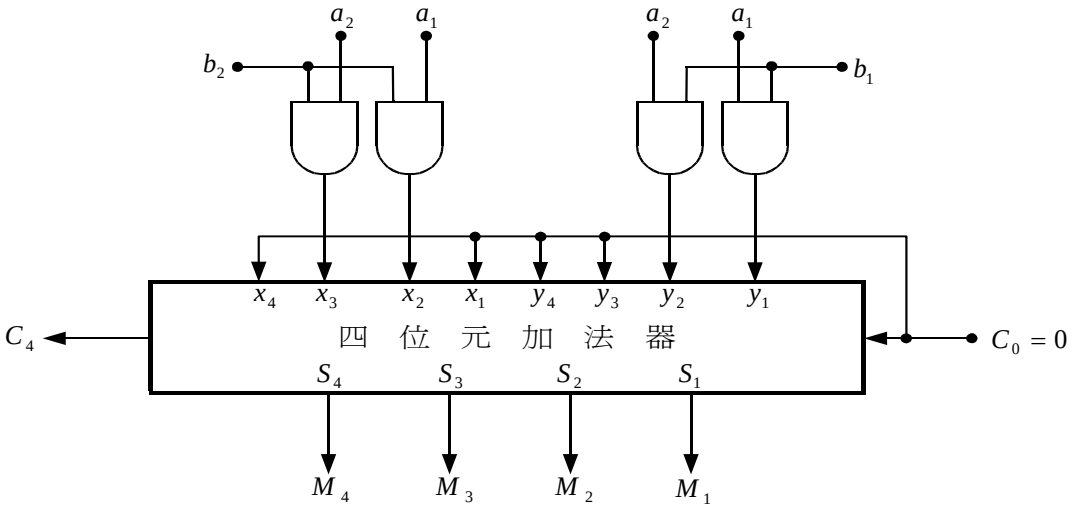
◆ 兩組 n 個位元之二進位數相乘的運算規則與兩組十進位數相乘一樣，欲執行被乘數 $a = a_2a_1$ 與乘數 $b = b_2b_1$ 之 $2bit \times 2bit$ 乘法運算，依二進位乘法之運算規則，可得以下之計算式：

$$
\begin{array}{rcccl}
 & & a_2 & a_1 & \leftarrow \text{被乘數} \\
 & \times & b_2 & b_1 & \leftarrow \text{乘數} \\
\hline
 & & a_2 \cdot b_1 & a_1 \cdot b_1 & \\
+ & C_3 & a_2 \cdot b_2 & a_1 \cdot b_2 & \\
\hline
 & M_4 & M_3 & M_2 & M_1 \leftarrow \text{乘積}
\end{array}
$$

其中

$$
\begin{aligned}
M_1 &= a_1 \cdot b_1 \\
M_2 &= a_2 \cdot b_1 + a_1 \cdot b_2 \\
M_3 &= a_2 \cdot b_2 \\
M_4 &= C_3
\end{aligned}
$$

◆ 觀察 $M_1 \sim M_3$ 之乘積可知，欲執行 $2bit \times 2bit$ 乘法器之邏輯電路，至少需 4 個 AND 閘與 2 個全加法器，故可用 4 位元加法器來實現較為簡單，如下圖所示。



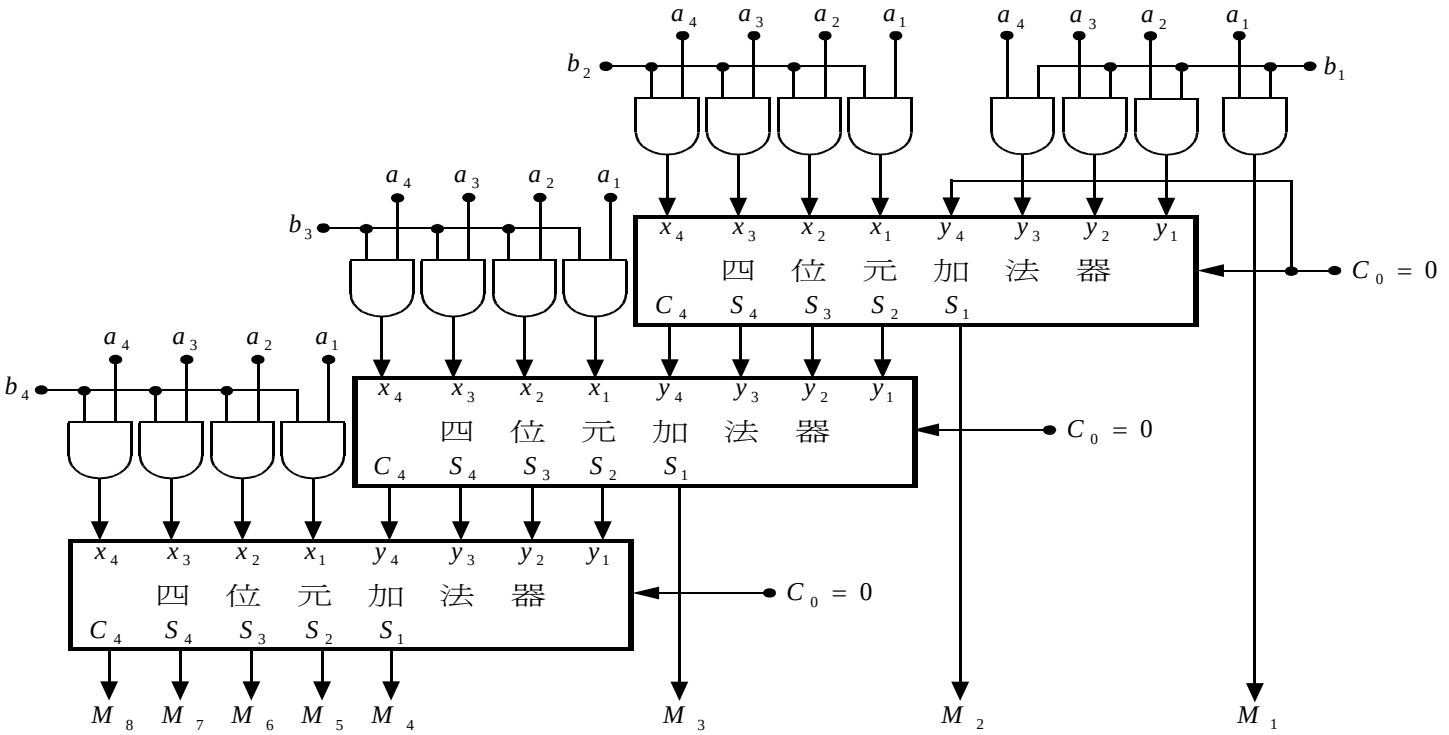
註 1： 上面之方程式為數學運算式，而非布林函數式，其中「+」之符號為算術運算之加法，而非 OR 之邏輯運算

註 2： 4 位元加法器之電路，已經考慮進位傳輸之問題，故在設計乘法電路時，可省去考慮進位傳輸之困擾。

4 位元與 4 位元之乘法器

◆ 被乘數 $a = a_4a_3a_2a_1$ 與乘數 $b = b_4b_3b_2b_1$，依**乘法之運算規則**，可得以下之計算式：

				a_4	a_3	a_2	a_1	← 被乘數	
			×	b_4	b_3	b_2	b_1	← 乘數	
				$a_4 \cdot b_1$	$a_3 \cdot b_1$	$a_2 \cdot b_1$	$a_1 \cdot b_1$		
			$a_4 \cdot b_2$	$a_3 \cdot b_2$	$a_2 \cdot b_2$	$a_1 \cdot b_2$			
		$a_4 \cdot b_3$	$a_3 \cdot b_3$	$a_2 \cdot b_3$	$a_1 \cdot b_3$				
+	C_4	$a_4 \cdot b_4$	$a_3 \cdot b_4$	$a_2 \cdot b_4$	$a_1 \cdot b_4$				
	M_8	M_7	M_6	M_5	M_4	M_3	M_2	M_1	← 乘積



算術邏輯單元(概述)

- 將算術運算(Arithmetic Operation)與邏輯運算(Logic Operation)合併在同一個邏輯電路，稱為算術邏輯單元 (Arithmetic Logic Unit; ALU)，藉著選擇變數之改變，不但可用來執行各種加、減法之算術運算，亦可用來實現一些 AND、OR、NOT 與 XOR 等基本邏輯運算。

- 設計算術邏輯單元需遵循一定的設計步驟：

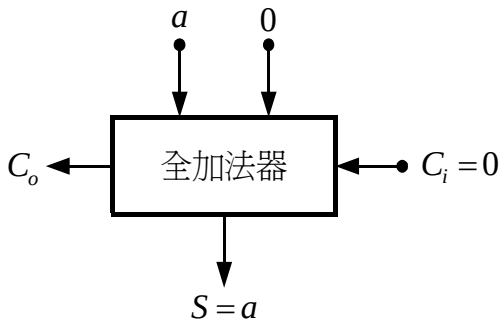
  1、 先考慮算術運算電路。

  2、 藉由改變算術運算之邏輯功能，以實現所需之邏輯運算。

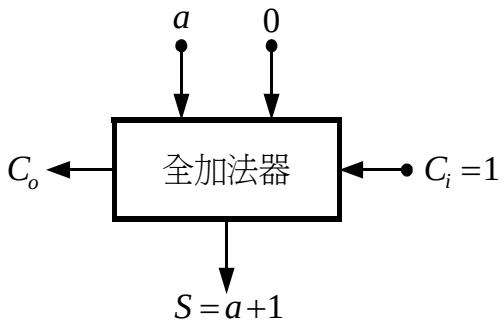
  3、 若尚有未能實現之邏輯運算，則再修正算術運算電路，以得到符合實際所需之算術邏輯運算單元。

算術運算電路之設計

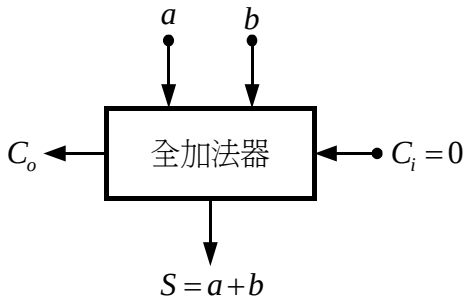
◆ 實現算術運算電路的基本元件為全加法器(FA)，首先假設 FA 之被加數與加數分別為 a 與 b，當給予 FA 之被加數 a、加數 b 與進位輸入端 C_i 不同之邏輯值時，便可得到各種不同功能之算術運算電路，如下圖所示。



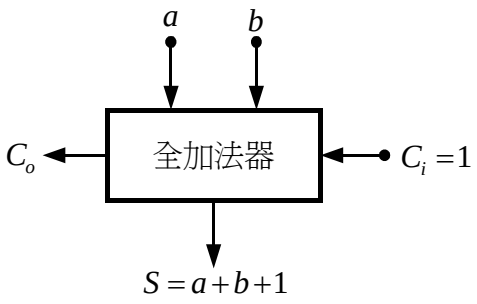
(a)通過 a



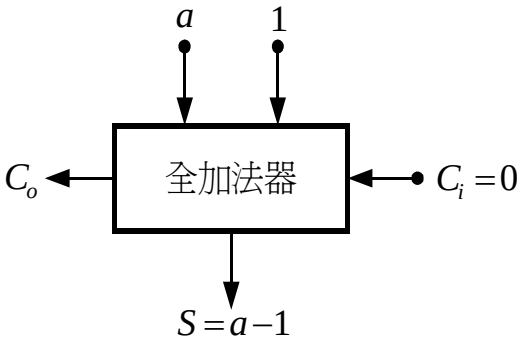
(b)遞增 a



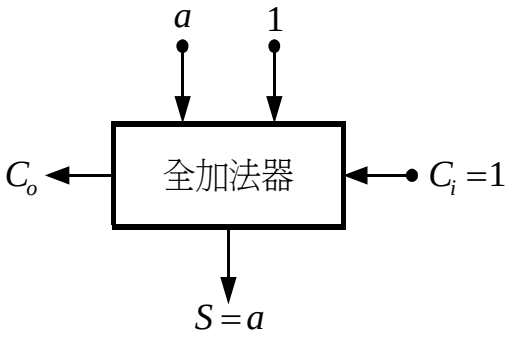
(c)加法運算



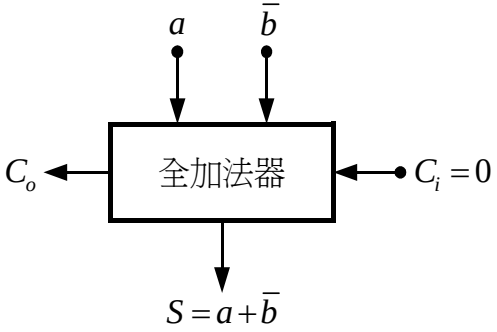
(d)有進位輸入之加法運算



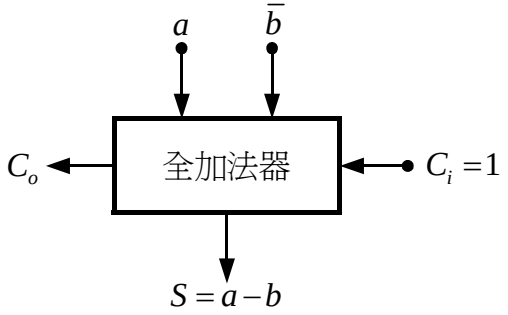
(e) 遞減 *a*



(*f*)通過 *a*

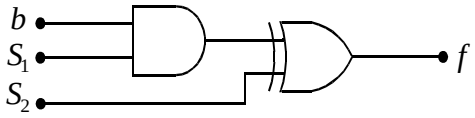


(g) a 加 b 之 1 的補數



(h)減法運算

◆ **藉由兩個選擇變數 S_2 與 S_1，即可用來控制輸入變數 *b* 之真 / 補、1/0 (True/Complement、One/Zero Element) 之組合邏輯電路，如下圖所示。**

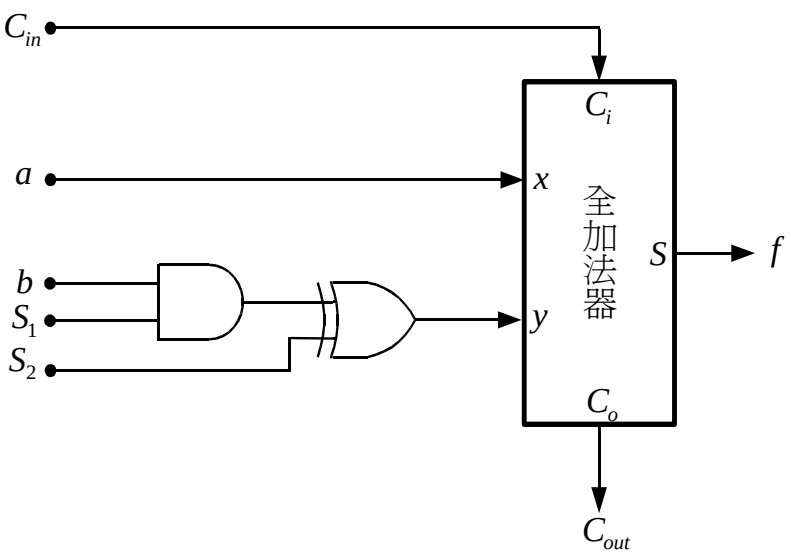


S_2	S_1	輸出 f
0	0	0
0	1	b
1	0	1
1	1	$\overline{b}$

(a) 邏輯電路圖　　(b) 邏輯運算功能表

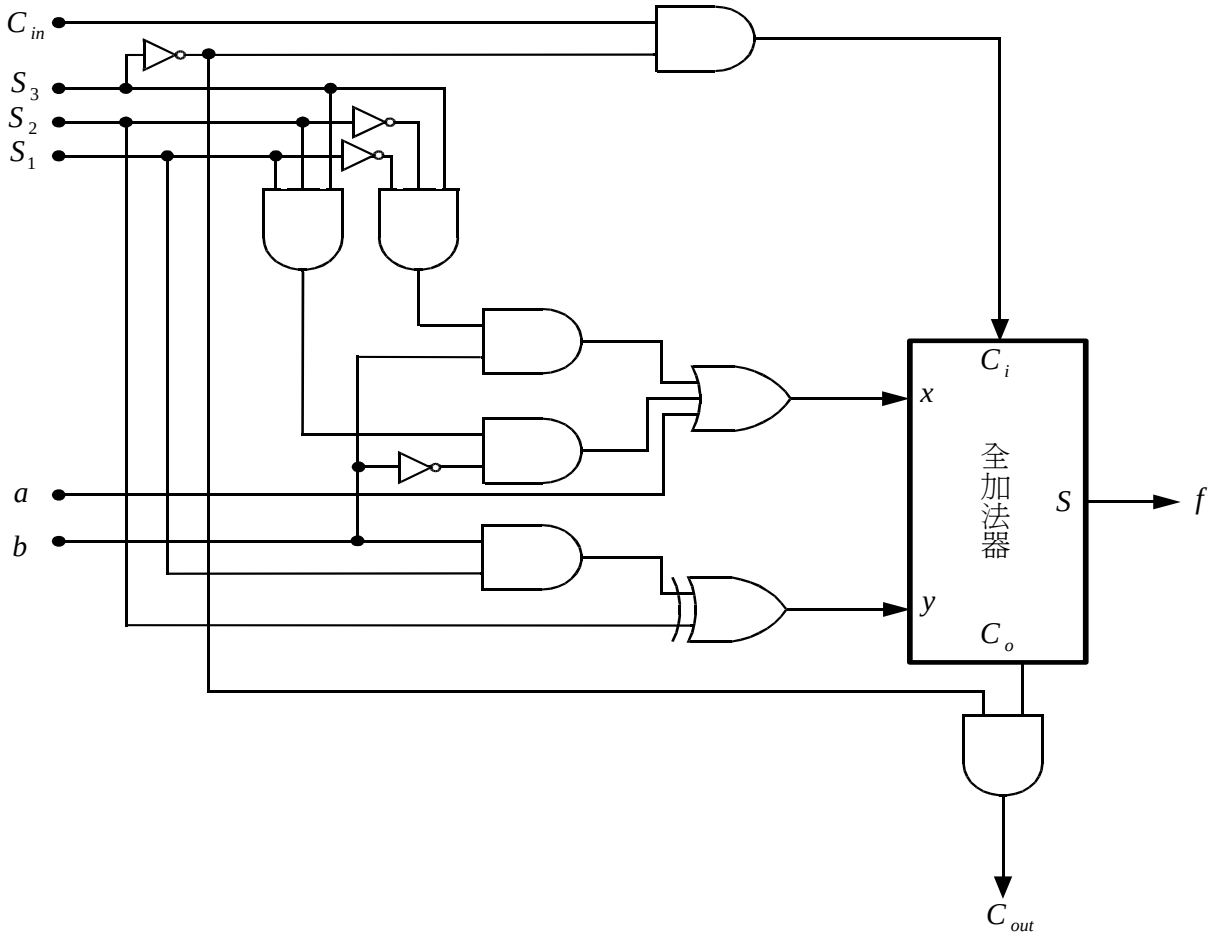
◆ 若分別加入被加數 a、加數 b 之輸入端與進位輸入 C_{in}，便可結合算術運算功能在同一個邏輯電路，而此電路圖與功能表如下圖所示。

S_2	S_1	C_{in}	y	輸出 f	算 術 運 算 功 能
0	0	0	0	a	通過 a
0	0	1	0	$a+1$	遞增 a
0	1	0	b	$a+b$	加法運算
0	1	1	b	$a+b+1$	有進位輸入之加法運算
1	0	0	1	$a-1$	遞減 a
1	0	1	1	a	通過 a
1	1	0	$\overline{b}$	$a+\overline{b}$	a 加 b 之 1 的補數
1	1	1	$\overline{b}$	$a+\overline{b}+1$	減法運算 $(a+b_{2'S}=a-b)$



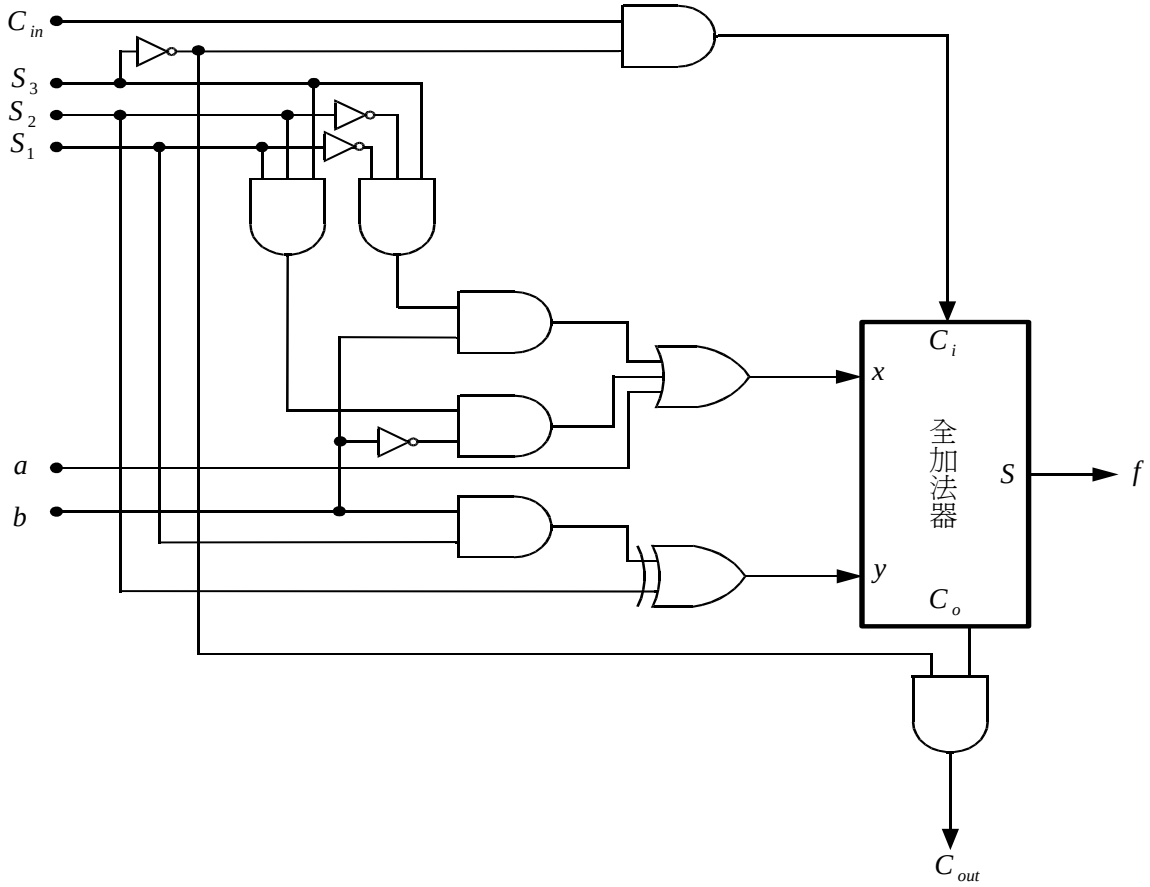
邏輯運算單元之設計

◆ 若再增加一條選擇線 S_3 與部分組合邏輯電路，便可將上圖改變為可產生 OR、AND、XOR 與 NOT 等 4 個邏輯運算之算術邏輯電路，如下圖所示。

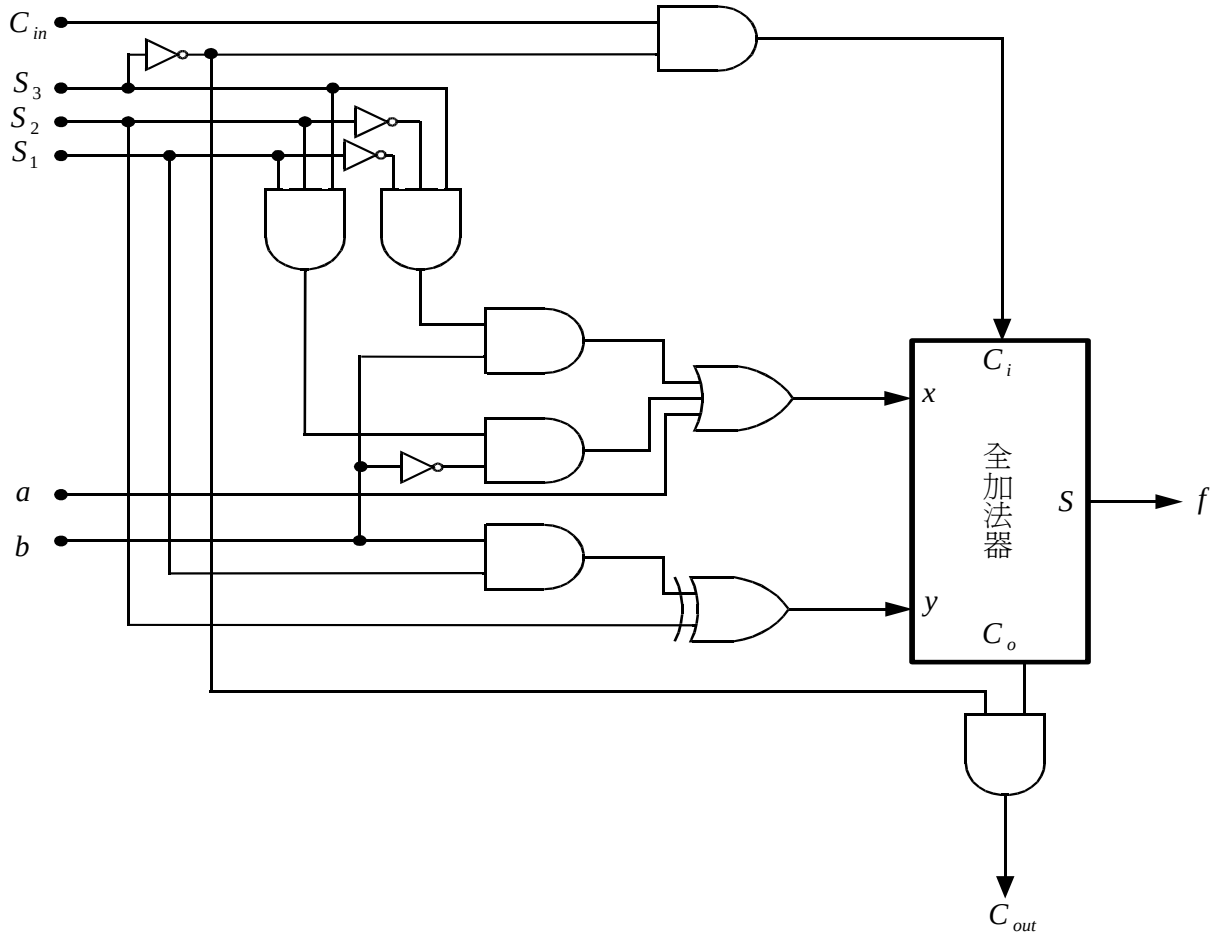


◆ 當選擇線S_3、S_2與S_1具不同值，且$C_{in}=0$時，產生下列邏輯運算功能：

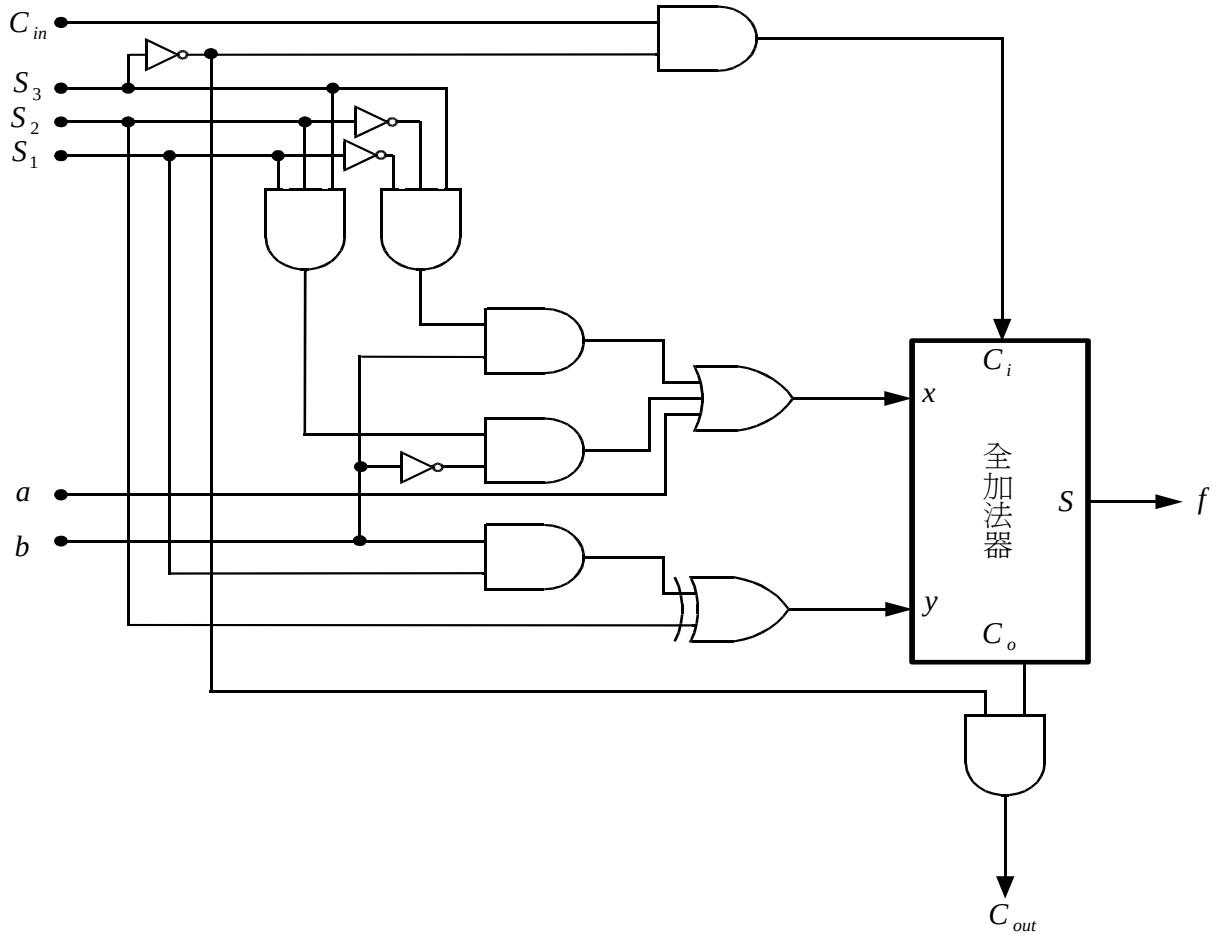
1.當 $S_3S_2S_1=100$ 時，全加法器之輸入端 $x=a+b$ 、 $y=0$ ，則全加法器之輸出 $f=x\oplus y=(a+b)\oplus 0=a+b$，故此時電路便可執行 OR 之邏輯運算。



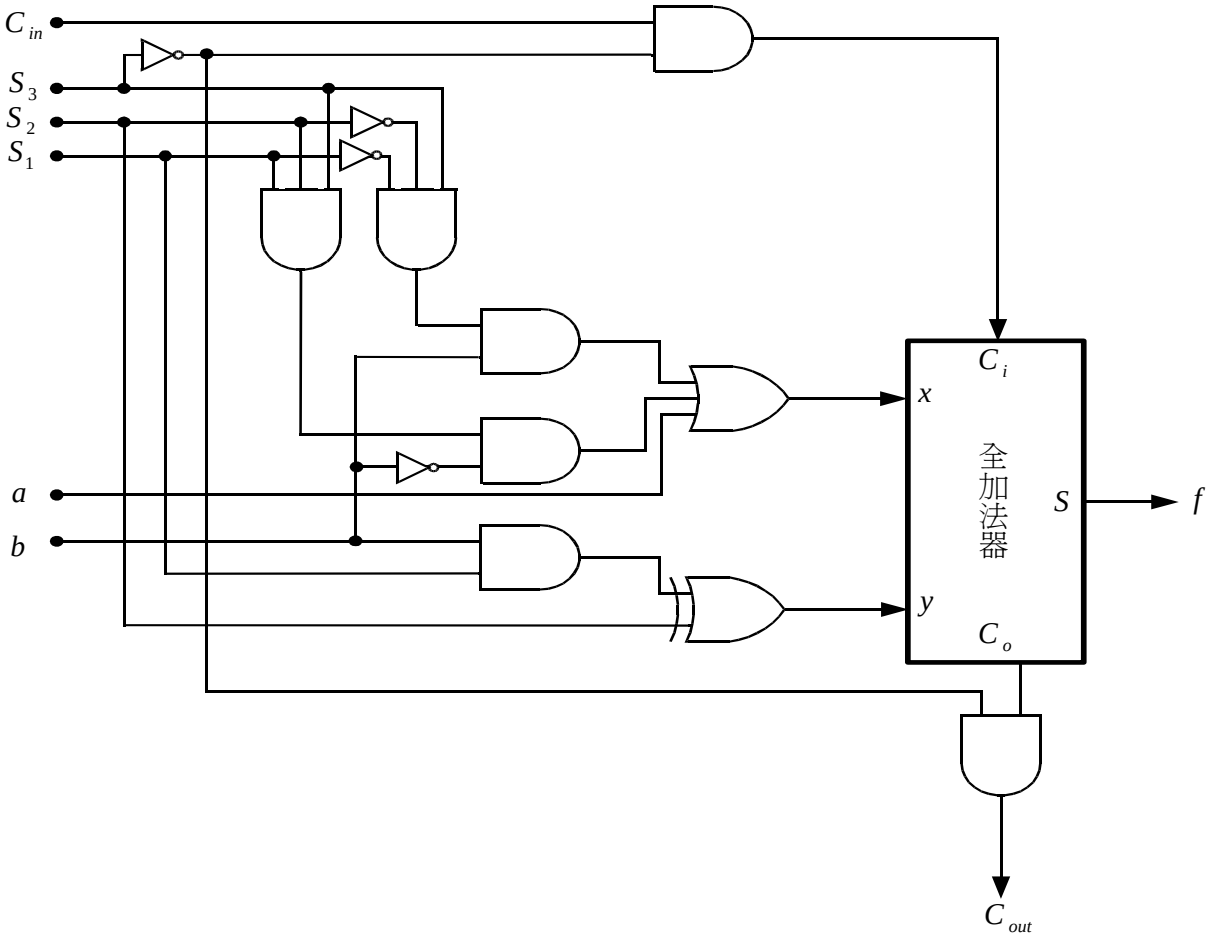
2. 當 $S_3S_2S_1 = 101$ 時，全加法器之輸入端 $x = a$ 、 $y = b$ ，則全加法器之輸出 $f = x \oplus y = a \oplus b$ ，故此時電路便可執行 XOR 之邏輯運算。



3. 當 $S_3S_2S_1 = 110$ 時，全加法器之輸入端 $x = a$ 、 $y = 1$ ，則全加法器之輸出 $f = x \oplus y = a \oplus 1 = \bar{a}$ ，故此時電路便可執行 NOT 之邏輯運算。



4. 當 $S_3S_2S_1 = 111$ 時，全加法器之輸入端 $x = a + \overline{b}$ 、 $y = \overline{b}$ ，則全加法器之輸出 $f = x \oplus y = (a + \overline{b}) \oplus \overline{b} = [\overline{(a + \overline{b})} \cdot \overline{b}] + [(a + \overline{b}) \cdot \overline{\overline{b}}] = a \cdot b$，故此時電路便可執行 AND 之邏輯運算。



算術邏輯單元之設計

◆ 若分別結合算術運算與邏輯運算功能，便可實現 1 位元算術邏輯單元之設計，如上圖所示，而結合後之整體算術邏輯運算功能表，如下表所示。

S_3	S_2	S_1	C_{in}	x	y	輸出 f	算術邏輯運算功能
0	0	0	0	a	0	a	通過 a
0	0	0	1	a	0	$a+1$	遞增 a
0	0	1	0	a	b	$a+b$	加法運算
0	0	1	1	a	b	$a+b+1$	有進位輸入之加法運算
0	1	0	0	a	1	$a-1$	遞減 a
0	1	0	1	a	1	a	通過 a
0	1	1	0	a	$\overline{b}$	$a+\overline{b}$	a 加 b 之 1 的補數
0	1	1	1	a	$\overline{b}$	$a+\overline{b}+1$	減法運算 $(a+b_{2's}=a-b)$
1	0	0	×	$a+b$	0	$a+b$	OR
1	0	1	×	a	b	$a\cdot b$	XOR
1	1	0	×	a	1	$a\oplus b$	NOT
1	1	1	×	$a+\overline{b}$	$\overline{b}$	$\overline{a}$	AND

數值比較器(1 位元之數值比較器)

◆ 數值比較器(Magnitude Comparator)是一種組合邏輯電路，它可決定一數值大於、等於或小於另一數值。首先討論一位元之比較器設計步驟列述如下：

1. 欲比較兩個 1 位元之二進位數之邏輯電路應有 2 個輸入變數，分別標示為 x 與 y 之符號；而兩數值比較結果，可指出兩數值間大於、等於或小於之關係，故此電路應有 3 個輸出變數，分別標示為 $f_1(x > y)$ 、 $f_2(x = y)$ 與 $f_3(x < y)$ 之符號。
2. 因輸出可指出兩數值 (x 與 y) 間之大於、等於或小於之關係，且兩個數值比較大小之結果，亦僅有一個輸出成立 (此情況可設定為邏輯 1)，其它兩個輸出不成立 (此情況可設定為邏輯 0)，故可用下面之真值表來定義輸出與輸入間的關係。

輸　入		輸　出		
x	y	$f_1(x > y)$	$f_2(x = y)$	$f_3(x < y)$
0	0	0	1	0
0	1	0	0	1
1	0	1	0	0
1	1	0	1	0

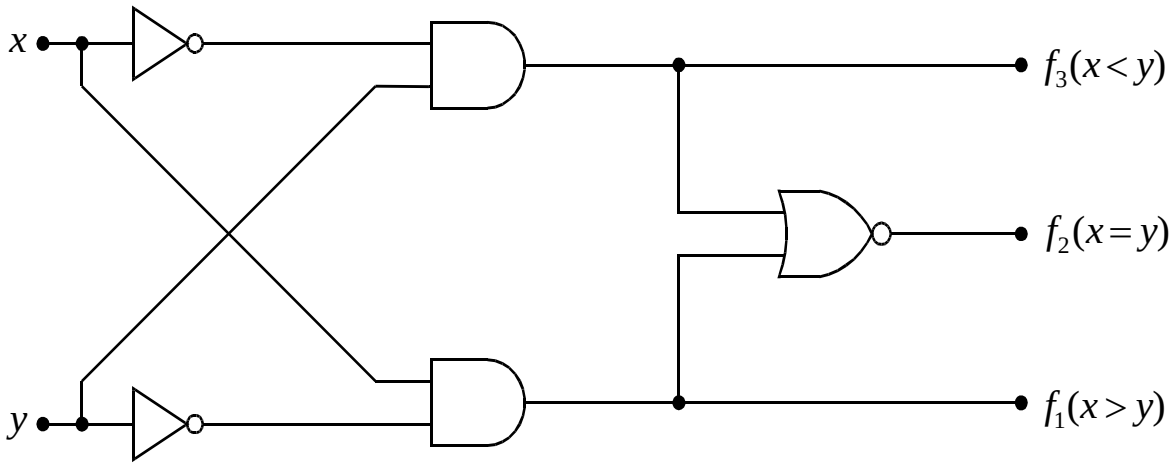
3. 利用卡諾圖對上面之真值表進行邏輯化簡，以求得輸出之最簡布林函數式

$$f_1(x>y)=x\cdot\bar{y}$$

$$f_2(x=y)=\bar{x}\cdot\bar{y}+x\cdot y=\overline{(x+y)\cdot(\bar{x}+\bar{y})}=\overline{\bar{x}\cdot y+x\cdot\bar{y}}$$

$$f_3(x<y)=\bar{x}\cdot y$$

4. 使用邏輯閘來實現以上 3 個輸出布林函數式，即可繪出 1 位元數值比較器之邏輯電路，如右圖所示。

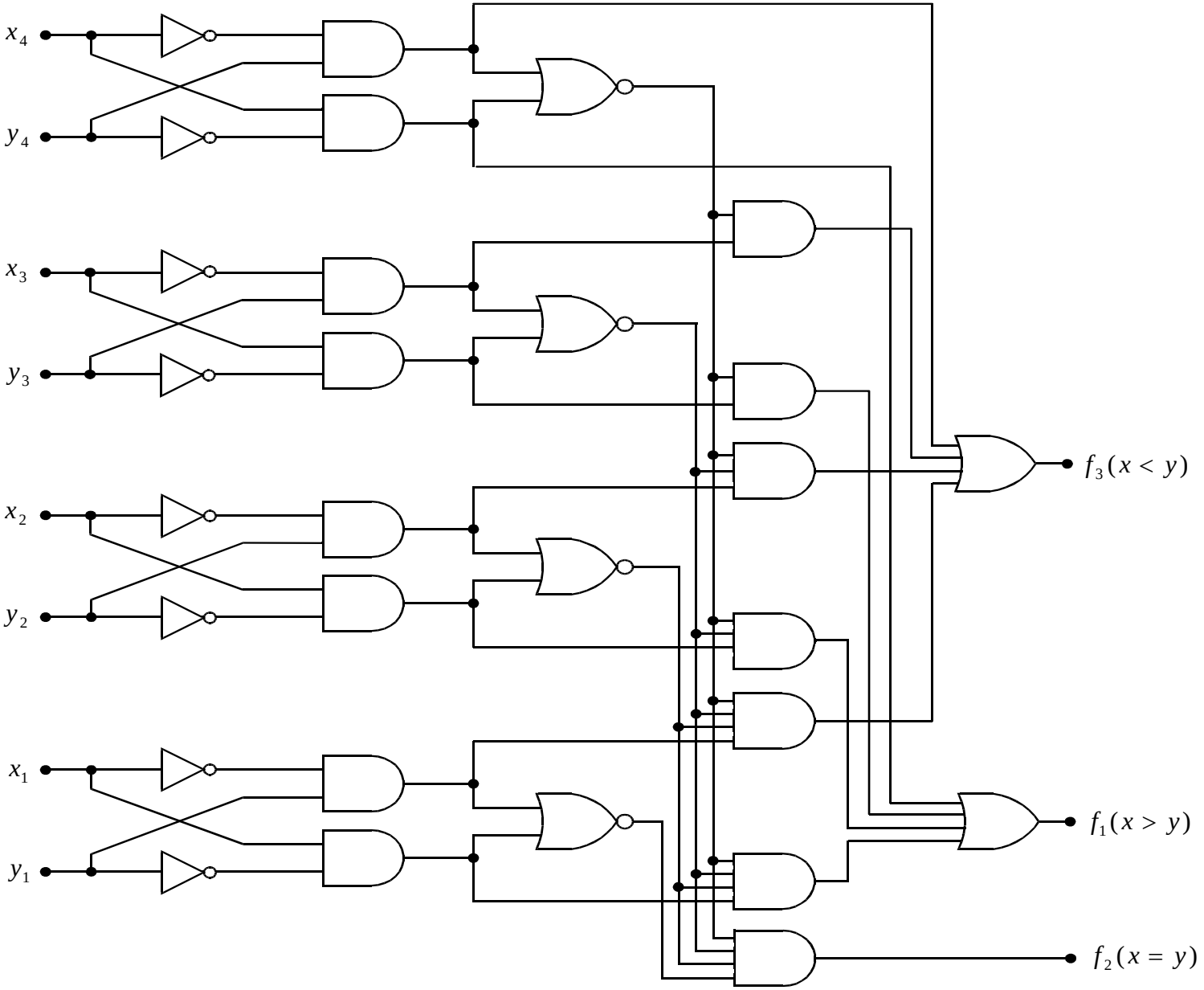


n 位元之數值比較器

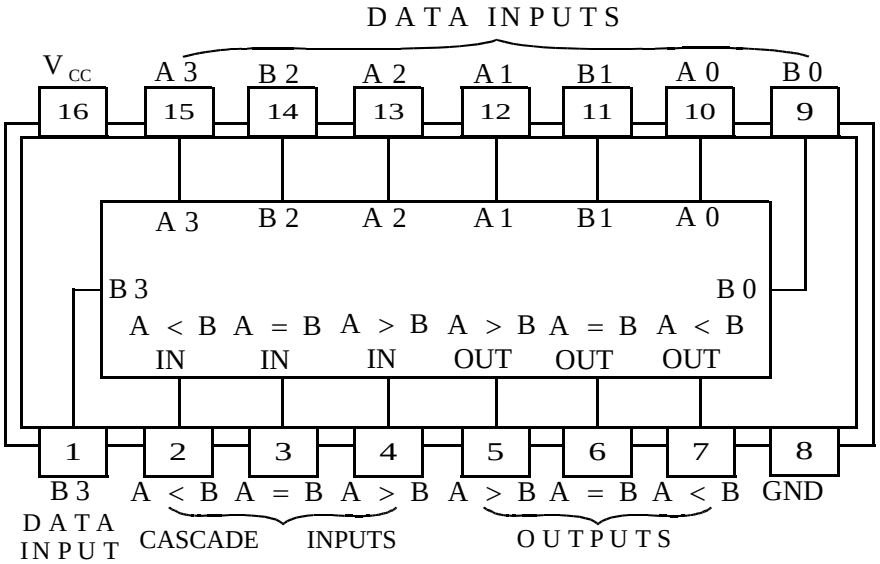
- 欲設計 n 位元之數值比較器，若使用真值表來表示，則可產生 2^{2n} 種組合，所以用傳統組合邏輯設計方法，則會變得相當不實用。因此必須法展一套直接、有效之程序 (Procedure) 來解決此類問題，而此種解決問題之程序稱為演算法 (Algorithm)。
- 以比較 $x = x_4x_3x_2x_1$ 與 $y = y_4y_3y_2y_1$ 兩組二進位數之大小為例，以說明使用演算法來設計 4 位元數值比較器之方法。首先使用演算法來發展出兩組 4 位元之二進位 (x 與 y) 數值比較器的真值表，如下表所示。

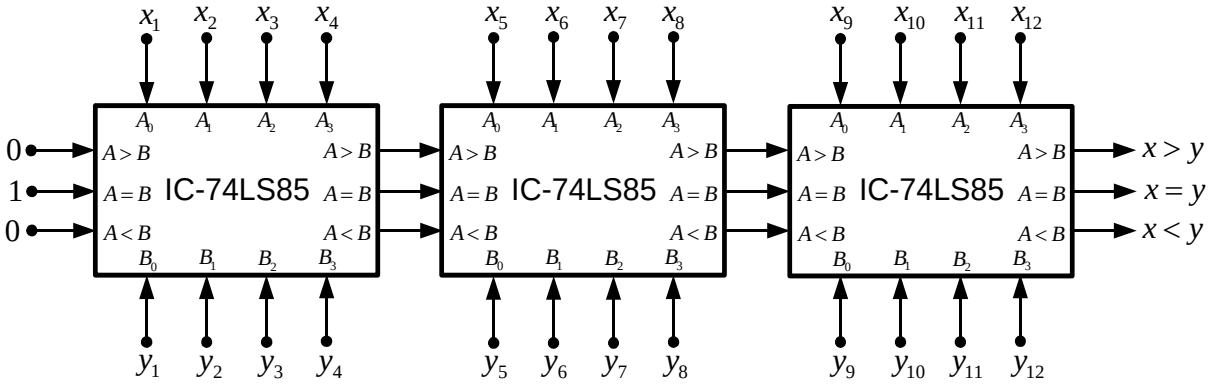
輸　　入				輸　　出		
x_4 、 y_4	x_3 、 y_3	x_2 、 y_2	x_1 、 y_1	$f_1(x > y)$	$f_2(x = y)$	$f_3(x < y)$
$x_4 > y_4$	×	×	×	1	0	0
$x_4 < y_4$	×	×	×	0	0	1
$x_4 = y_4$	$x_3 > y_3$	×	×	1	0	0
$x_4 = y_4$	$x_3 < y_3$	×	×	0	0	1
$x_4 = y_4$	$x_3 = y_3$	$x_2 > y_2$	×	1	0	0
$x_4 = y_4$	$x_3 = y_3$	$x_2 < y_2$	×	0	0	1
$x_4 = y_4$	$x_3 = y_3$	$x_2 = y_2$	$x_1 > y_1$	1	0	0
$x_4 = y_4$	$x_3 = y_3$	$x_2 = y_2$	$x_1 < y_1$	0	0	1
$x_4 = y_4$	$x_3 = y_3$	$x_2 = y_2$	$x_1 = y_1$	0	1	0

◆ 比較兩組多位元之二進位數之大小，需由兩個最高有效位元(MSB)開始比較，若此對位元不相等，則兩數值之大小立刻可以決定，而不必理會比 MSB 低之二進位數；若兩個 MSB 相等，再比較下一對權位較低之有效位元，直到有一對有效位元不相等為止。若 4 個有效位元皆相等，則表示 x 與 y 相等。根據以上之說明，即可設計出 4 位元數值比較器之邏輯電路，如下圖所示。


x_4
y_4
x_3
y_3
x_2
y_2
x_1
y_1
$f_3(x < y)$
$f_1(x > y)$
$f_2(x = y)$

◆ **IC-74LS85 除了提供 4 位元比較器之所有輸入與輸出接腳外，另外增加「$A > B$」(第 4 腳)、「$A = B$」(第 3 腳) 與「$A < B$」(第 2 腳)等 3 個串級(Cascade)接腳，利用此 3 個接腳，可用串接多個 IC-74LS85，以實現超過 4 個位元之數值比較器，如下圖所示。**





問題討論

1、試分別說明半加法器(Half Adder)與全加法器(Full Adder)在執行二進位加法運算時，並討論兩者之功能有何不同。

解：

(a)、半加法器：未考慮進位傳送之運算，僅可對兩個1個位元之2進位數進行加法運算。

(b)、全加法器：有考慮進位傳送之運算，因此可用串接之方式對超過兩個1個位元之2進位數進行加法運算。

2、試討論串接 4 個全加法器所設計之 4 位元並行加法器的缺點，並請說明改善之方法。

解：

(a) 使用並行加法器來執行二進位數的加法運算時，雖然所有加數與被加數之信號皆為已知，但較高位元之加法運算，必須等較低位元相加，得到穩定的進位訊號後，才能得到正確之結果，即加數與被加數通過邏輯閘之層數與加數、被加數之位元數成正比關係，因訊號通過邏輯閘會造成時間延遲之關係，因此會造成嚴重進位傳輸延遲之問題。

(b) 為了加快算術運算速度 (假設每個邏輯閘之傳輸延遲時間皆相同之條件下)，可從改良電路著手，以減少等待進位所造成之傳輸延遲時間。而目前於減少進位傳輸延遲之方法非常多，其中最廣泛使用之方法為前視進位加法 (Look-Ahead Carry) 原理。

3、試分別說明半減法器(Half Subtractor)與全減法器(Full Subtractor)在執行二進位加法運算時，並討論兩者之功能有何不同。

解：

(a) 半減法器：未考慮借位傳送之運算，僅可對兩個1個位元之2進位數進行減法運算。

(b) 全減法器：有考慮借位傳送之運算，因此可用串接之方式對超過兩個1個位元之2進位數進行減法運算。

4、試討論算術邏輯單元(ALU)之基本定義。

解：

將算術運算 (Arithmetic Operation) 與邏輯運算 (Logic Operation) 合併在同一個邏輯電路，此種電路稱為算術邏輯單元 (Arithmetic Logic Unit; ALU)，它可實現多種功能之組合邏輯函數，即藉著選擇變數之改變，不但可用來執行各種加、減法之算術運算，亦可用來實現一些 AND、OR、NOT 與 XOR 等基本邏輯運算。

5、試繪出十進位加法器 (Decimal Adder) 之邏輯電路圖，並說明電路進行 6 加 9 之運算過程 (請在邏輯電路上標示出每個邏輯閘輸入與輸出端之邏輯值)。

解：