第三章 研究背景

在本章，我們將介紹 Baugh-Wooley 乘法演算法、陣列乘法器與 Wallace 乘法器以及 Dadda 乘法器之比較分析、以及用於乘法器的 Brent-Kung 加法器。然後綜合以上的作法，產生管線正負號 Dadda 乘法器。

3.1 Baugh-Wooley 乘法演算法

DSP 的乘法器通常都是正負號(signed) 乘法，因為 DSP 的係數有正值跟負值。Baugh-Wooley 演算法是最常使用的正負號乘法[22]，它是用 2 的補數代表正負號。以下將介紹 Baugh-Wooley 演算法的運算方式。

圖 21 為 8 × 8 bit 無正負號(unsigned)乘法。其中 X 為被乘數，共有 X0、X1、……、X7　8 個 bits。Y 為乘數，共有 Y0、Y1、……、Y7　8 個 bit。S 為部分乘積(partial product)，Si, j＝ Xi AND Yj，共有 8×8＝64 個部分乘積。P 為將部分乘積相加而得到的乘積，共有 P0、P1、……、P15　16 個 bit，其中 P15 為 P14 的進位(carry)。

圖 22 為 8 × 8 bit Baugh-Wooley 正負號乘法，其中 NSi, j＝NOT (Xi AND Yj)。與 8×8 bit 無正負號乘法比較，Baugh -Wooley 演算

法僅需要將 S7,0、S7,1、……、S7,6 以及 S0,7、S1,7、……、S6,7 共 14 個部分乘積做反向，並且在 P8 列的位置加一，即可將無正負號乘法變成正負號乘法，須注意 P15 為 P14 的進位值做反向。

Baugh-Wooley 演算法大幅降低正負號延長(signed extension)造成的功率消耗，因此是使用相當普遍的正負號乘法演算法。

								X7	X6	X5	X4	X3	X2	X1	X0
								Y7	Y6	Y5	Y4	Y3	Y2	Y1	Y0
								S7,0	S6,0	S5,0	S4,0	S3,0	S2,0	S1,0	S0,0
							S7,1	S6,1	S5,1	S4,1	S3,1	S2,1	S1,1	S0,1	
						S7,2	S6,2	S5,2	S4,2	S3,2	S2,2	S1,2	S0,2		
					S7,3	S6,3	S5,3	S4,3	S3,3	S2,3	S1,3	S0,3			
				S7,4	S6,4	S5,4	S4,4	S3,4	S2,4	S1,4	S0,4				
			S7,5	S6,5	S5,5	S4,5	S3,5	S2,5	S1,5	S0,5					
		S7,6	S6,6	S5,6	S4,6	S3,6	S2,6	S1,6	S0,6						
	S7,7	S6,7	S5,7	S4,7	S3,7	S2,7	S1,7	S0,7							
P15	P14	P13	P12	P11	P10	P9	P8	P7	P6	P5	P4	P3	P2	P1	P0

圖 21　8 × 8 bit 無正負號(unsigned)乘法

								X7	X6	X5	X4	X3	X2	X1	X0
								Y7	Y6	Y5	Y4	Y3	Y2	Y1	Y0
							1	NS7,0	S6,0	S5,0	S4,0	S3,0	S2,0	S1,0	S0,0
							NS7,1	S6,1	S5,1	S4,1	S3,1	S2,1	S1,1	S0,1	
						NS7,2	S6,2	S5,2	S4,2	S3,2	S2,2	S1,2	S0,2		
					NS7,3	S6,3	S5,3	S4,3	S3,3	S2,3	S1,3	S0,3			
				NS7,4	S6,4	S5,4	S4,4	S3,4	S2,4	S1,4	S0,4				
			NS7,5	S6,5	S5,5	S4,5	S3,5	S2,5	S1,5	S0,5					
		NS7,6	S6,6	S5,6	S4,6	S3,6	S2,6	S1,6	S0,6						
	S7,7	NS6,7	NS5,7	NS4,7	NS3,7	NS2,7	NS1,7	NS0,7							
P15	P14	P13	P12	P11	P10	P9	P8	P7	P6	P5	P4	P3	P2	P1	P0

圖 22　8 × 8 bit Baugh-Wooley 正負號(signed)乘法

3.2 各種乘法器之架構

在這一節我們將分別介紹陣列(array)乘法器、Wallace 乘法器與 Dadda 乘法器，以及比較三者的優缺點。

3.2.1 陣列(array)乘法器

陣列乘法器為使用相當普遍的乘法器，它的加法器的排列較有規則性，但是速度卻較慢。圖 23 為 8 × 8 bit 陣列乘法器的總和圖解。╳代表一個半加器(half adder)，／ 代表一個全加器(full adder)。

假設半加器的 2 個輸入為 A、B，則輸出為和(sum) S 與進位(carry) C，其邏輯等式為

$$S = A \oplus B$$

$$C = AB$$

全加器比半加器多了一個輸入，假設全加器的 3 個輸入為 A、B、Cin，則輸出為和 S 與進位 C，其邏輯等式為

$$S = A \oplus B \oplus Cin \quad (9)$$

$$C = AB + BCin + ACin \quad (10)$$

╳線右上角的點代表加法器的和 S，左下角的點代表加法器的進位 C，／線亦同。

圖 23 最上層是部分乘積，共有 8 × 8＝64 個部分乘積。Stage 1

是第一階層加法器，共有 7 個半加器。Stage 2 至 Stage 7 的加法器皆為全加器，因此共有 42 個全加器。

n × n bit 陣列乘法器 Stage 1 的列數為 n，Stage 2 的列數為 n－1，Stage 3 的列數為 n－2，…依此類推，最後 Stage 的列數為 2。以 8 × 8 bit 陣列乘法器為例，由下到上 Stage 的列數為 2, 3, 4, 5, 6, 7, 8。

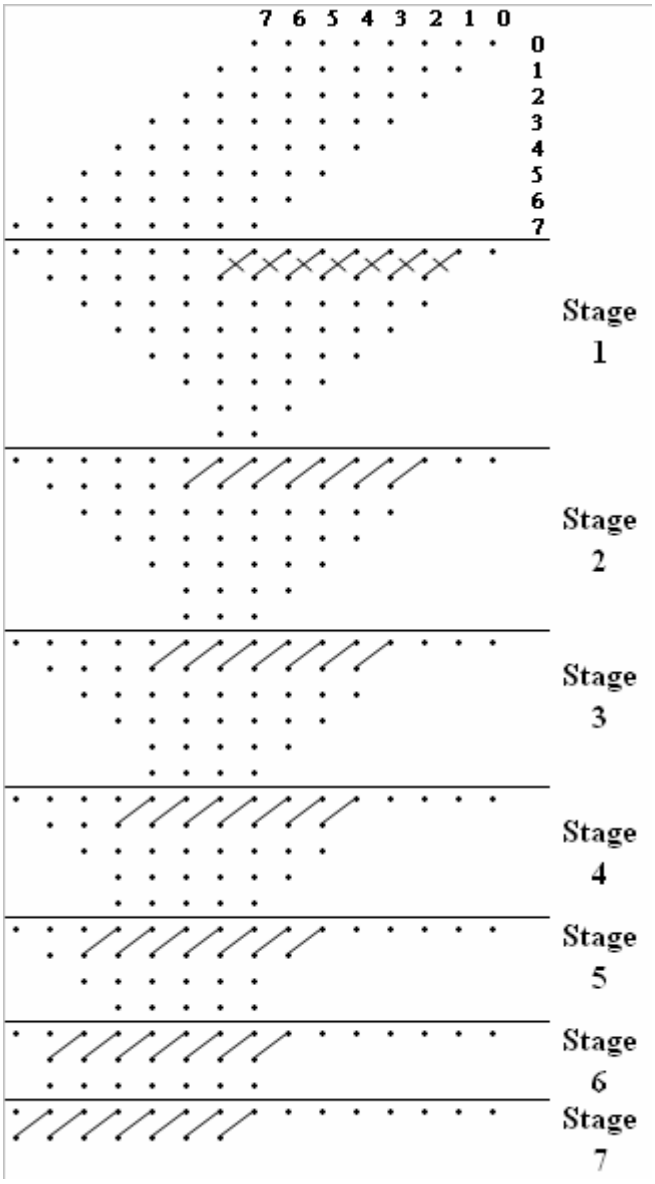

圖 23　8 × 8 bit 陣列乘法器的總和圖解

根據圖 23 可畫出圖 24 8 × 8 bit 陣列乘法器線路圖。圖 40 中 F 代表全加器，H 代表半加器，加法器右下方的輸出為和(sum)，左下方的輸出為進位(carry)。S 為部分乘積(partial product)，Si, j＝ Xi AND Yj 。

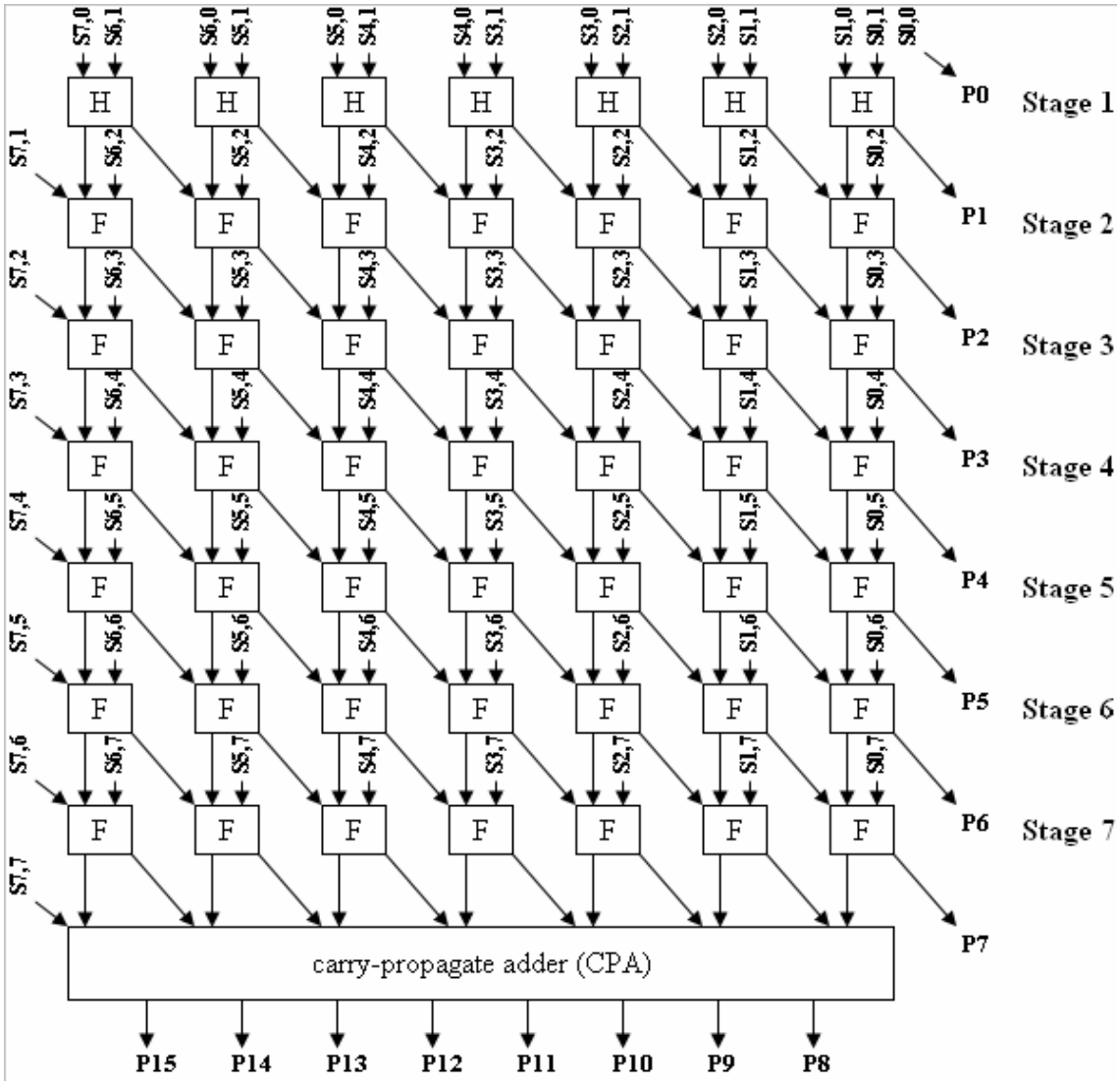

圖 24　8 × 8 bit 陣列乘法器線路圖

在 Stage 7 之後還必須接進位-增殖加法器(carry-propagate adder, CPA)，CPA 的字元長度(word length)為輸入 bit 除以 2。以

8 × 8 bit 陣列乘法器為例，其 CPA 的字元長度為 7。通常 CPA 使用漣波-進位加法器(ripple-carry adder, RCA)或是速度較快的進位-向前看加法器(carry look-ahead adder, CLA)[23]，所謂漣波-進位加法器是全加器以串連方式連接在一起，如圖 25 所示。圖 25 為 4 bit 漣波-進位加法器，A(3:0)與 B(3:0)為相加的二個 4 bit 數，S(4:0)為相加之後 5 bit 的結果，FA 代表全加器，和 S 與進位 C 由(9)(10)式產生，須注意 Cin＝0 而 C(3)＝S(4)。每一級的加法器提供一個進位位元 C 給左邊的下一級，因為進位信號是串連從加法器右邊一級一級傳至左邊，所以速度較慢。

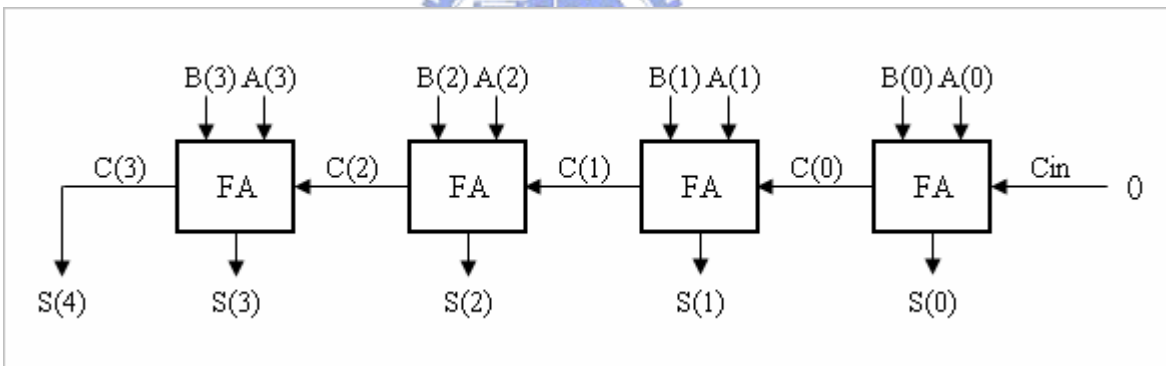

圖 25　4 bit 漣波-進位加法器

另外 CPA 可以使用速度較快的進位-向前看加法器，圖 26 為 4 bit 進位-向前看加法器，A(3:0)與 B(3:0)為相加的二個 4 bit 數，S(4:0)為相加之後 5 bit 的結果，FA 代表全加器，全加器的進位信號 C(i)是由產生(generate)G(i)與傳遞(propagate)P(i)所取代，其等式為

$$G(i)=A(i)B(i)$$

$$P(i)=A(i)+B(i)$$

則

$$C(i)=G(i)+P(i)C(i-1) \qquad (11)$$

同樣地，C(i＋1)可以寫成

$$C(i+1)=G(i+1)+P(i+1)C(i) \qquad (12)$$

將(11)式代入(12)式可以得到

$$C(i+1)=G(i+1)+P(i+1)G(i)+P(i+1)P(i)C(i-1)$$

繼續這樣下去，C(i)可以表示成G與P乘積和的函數。以圖26的4 bit加法器為例，進位C是由下列式子所表示

$$C(0)=G(0)+P(0)Cin$$

$$C(1)=G(1)+P(1)C(0)$$

$$=G(1)+P(1)G(0)+P(1)P(0)Cin$$

$$C(2)=G(2)+P(2)C(1)$$

$$=G(2)+P(2)G(1)+P(2)P(1)G(0)+P(2)P(1)P(0)Cin$$

$$C(3)=G(3)+P(3)C(2)$$

$$=G(3)+P(3)G(2)+P(3)P(2)G(1)+P(3)P(2)P(1)G(0)+$$

$$P(3)P(2)P(1)P(0)Cin$$

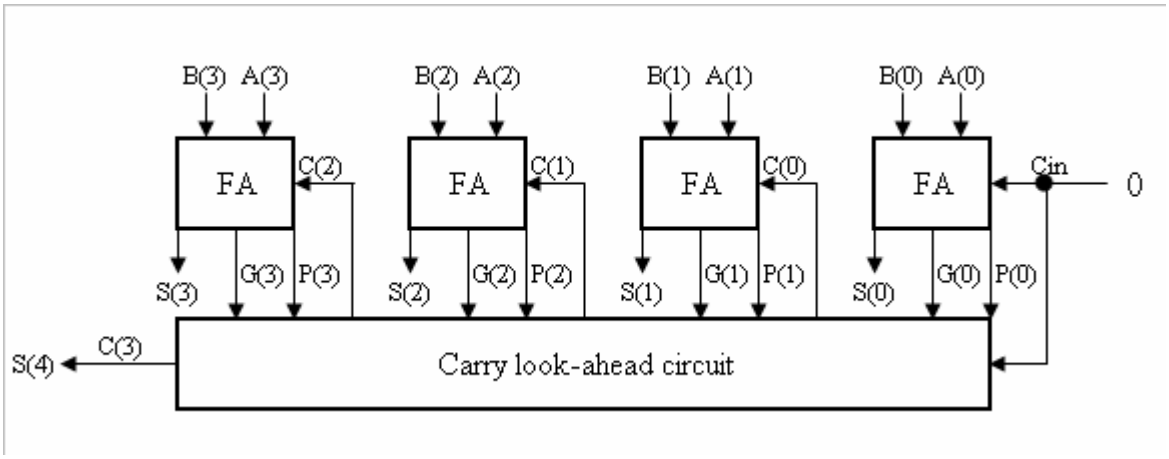

圖 26　4 bit 進位-向前看加法器

須注意圖 26 中 Cin＝0 而 C(3)＝S(4)。因為進位-向前看加法器是直接由前級 Cin 來產生進位位元，而不是將進位一級一級傳送，因此速度較漣波-進位加法器快。

3.2.2 Wallace 乘法器

Wallace 乘法器是由輸入往前(forward)做壓縮[24]，全加器可視為 3:2 壓縮器(compressor)或(3,2)計數器(counter)，亦即將 3 bits 壓縮成 2 bits，半加器則為 2:2 壓縮器或(2,2) 計數器，亦即將 2 bits 壓縮成 2 bits。

n × n bit的Wallace乘法器在最上層的部分乘積列數R_0為n，在 Stage j＋1 的列數R_{j+1}為

$$R_{j+1}=2\cdot floor(R_j/3)+R_j \bmod 3$$

其中floor(R_j/3)是取整數，而最下層的列數為 2。以 8 × 8 bit Wallace乘法器為例，R_0＝8，Stage 1 至Stage 4 的列數R_1至R_4為

$$R_1 = 2 \cdot \text{floor}(8/3) + 8 \bmod 3 = 2 \cdot 2 + 2 = 6$$

$$R_2 = 2 \cdot \text{floor}(6/3) + 6 \bmod 3 = 2 \cdot 2 = 4$$

$$R_3 = 2 \cdot \text{floor}(4/3) + 4 \bmod 3 = 2 \cdot 1 + 1 = 3$$

$$R_4 = 2 \cdot \text{floor}(3/3) + 3 \bmod 3 = 2 \cdot 1 = 2$$

因此 8 × 8 bit Wallace 乘法器由下到上 Stage 的列數分局為 2,3,4,6。圖 27 為 8 × 8 bit Wallace 乘法器的總和圖解。全加器的數目為 38，半加器為 15，而 CPA 的字元長度為 11。

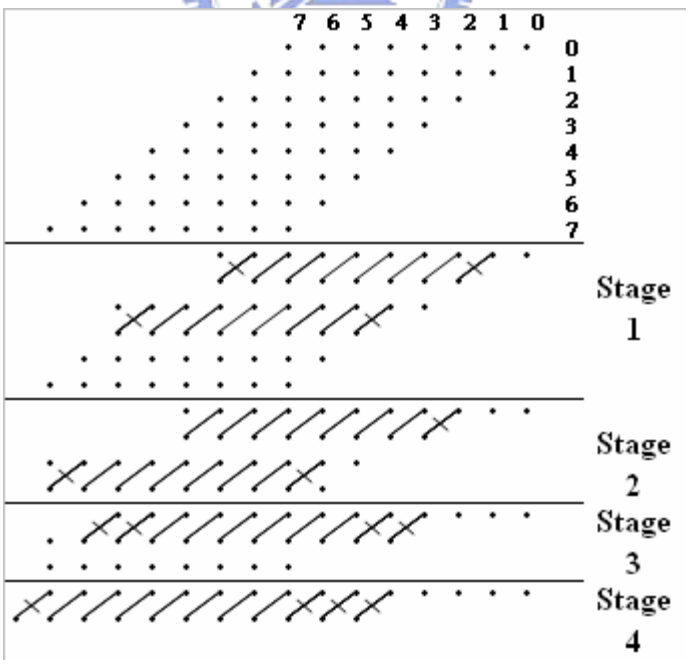

圖 27　8 × 8 bit Wallace 乘法器的總和圖解

2.2.3 Dadda 乘法器

Dadda 乘法器是由 L. Dadda 於 1965 年所提出[25],如同 Wallace 乘法器，它減少了加法器的階層數目，因此速度較快。但做法不同於 Wallace 乘法器，它是往後(backward)做壓縮。

圖 28 為 8 × 8 bit Dadda 乘法器的總和圖解，最下階層的列數為 2，往上一階層的列數是下面階層的 3/2 倍取整數，因此由下到上每一階層的列數分別為 2,3,4,6,9,13,19,28,42,63,……。以 8 × 8 bit Dadda 乘法器為例，其列數為 2,3,4,6。

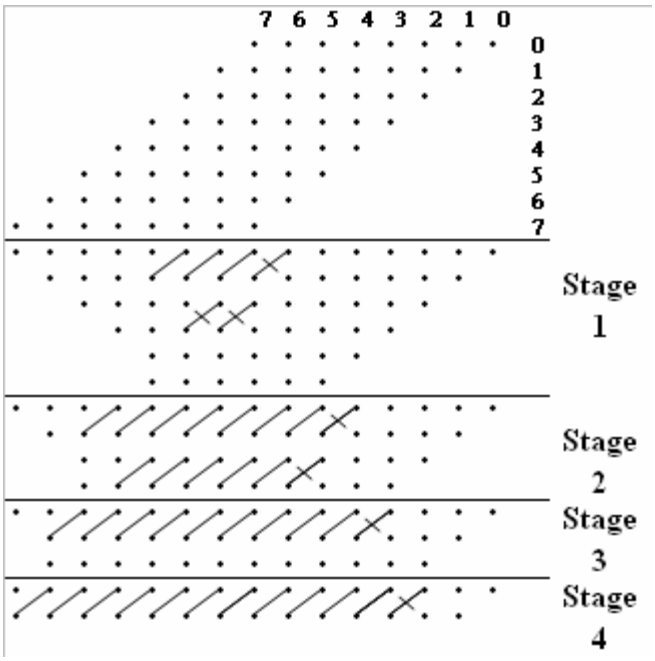

圖 28　8 × 8 bit Dadda 乘法器的總和圖解

根據圖 28 可畫出圖 29　8 × 8 bit Dadda 乘法器線路圖。H 代表半加器，F 代表全加器，右下方的輸出為和(sum)，左下方的輸出

為進位(carry)。H 與 F 底下的數字則代表該加法器的和所構成乘積(product)的位元數，任一加法器的 sum 一定接到相同乘積位元的加法器，而 carry 一定接到下一個乘積位元的加法器。舉例來說，在 stage 1 最右邊的加法器 H6，其 sum 接到 stage 2 相同乘積位元為 6 的加法器 F6，而 carry 接到 stage 2 乘積位元為 7 的加法器 F7。

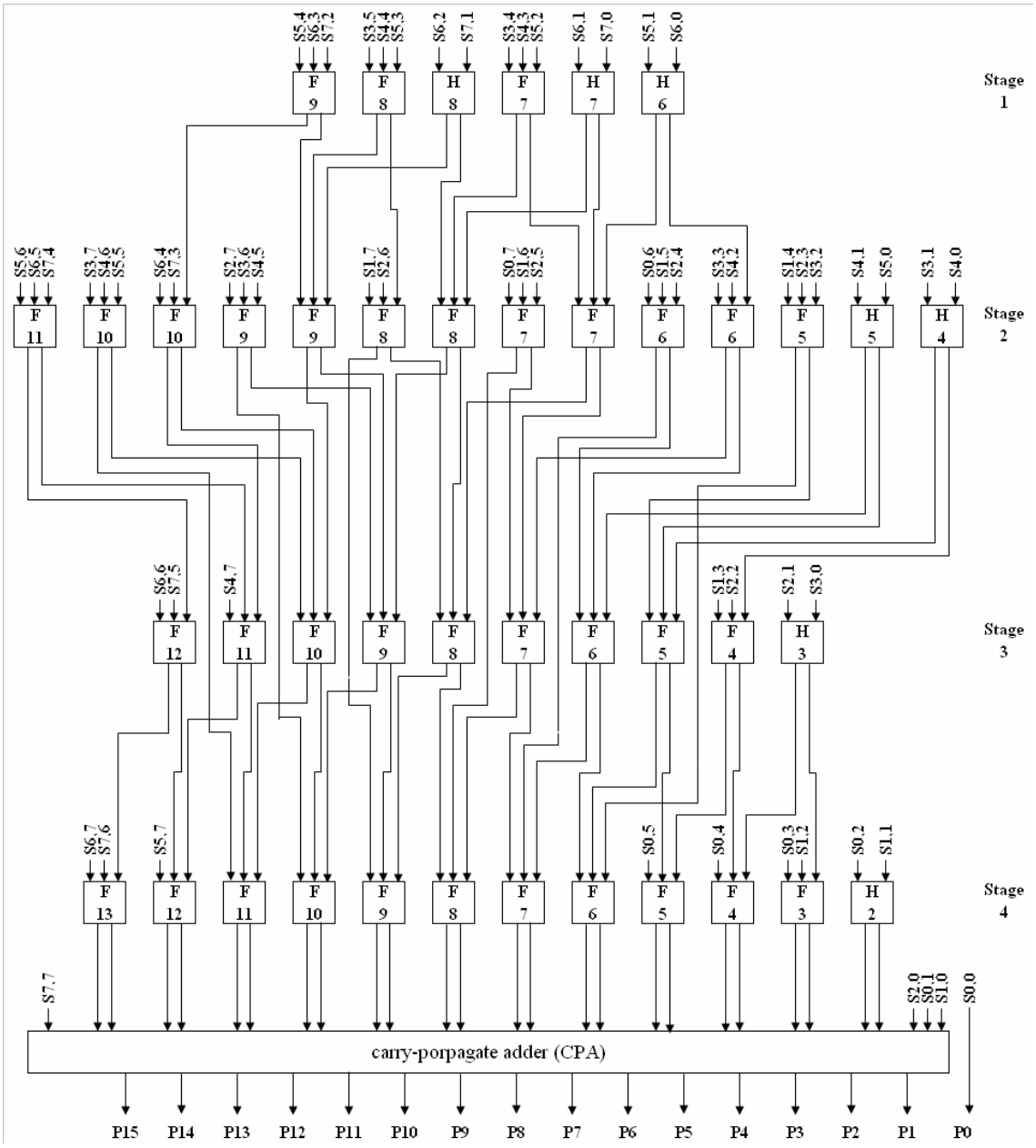

圖 29　8 × 8 bit Dadda 乘法器線路圖

以 8 × 8 bit Dadda 乘法器的全加器的數目為 35，半加器為 7，CPA 的字元長度為 14，而階層數跟 Wallace 乘法器一樣皆為 4。

3.2.4 比較分析

我們比較這三種乘法器的速度與面積，表 6 分別列出 8 × 8，16 × 16，32 × 32，以及 64 × 64 bit 三種乘法器的比較。

Multiplier	#(3,2)	#(2,2)	CPA length	# Stage
8 x 8 Array	42	7	7	7
8 x 8 Wallace	38	15	11	4
8 x 8 Dadda	35	7	14	4
16 x 16 Array	210	15	15	15
16 x 16 Wallace	200	54	25	6
16 x 16 Dadda	195	15	30	6
32 x 32 Array	930	31	31	31
32 x 32 Wallace	906	164	55	8
32 x 32 Dadda	899	31	62	8
64 x 64 Array	3906	63	63	63
64 x 64 Wallace	3850	459	117	10
64 x 64 Dadda	3843	63	126	10

表 6　各種乘法器的比較

1.比較速度

Dadda乘法器與Wallace乘法器的階層數目較Array乘法器少，但是Dadda乘法器與Wallace乘法器的CPA字元長度較Array乘法器多，我們可以將CPA使用快速的進位-向前看加法器(carry look-ahead

adder, CLA)以提高速度。若三種乘法器的CPA皆使用CLA，隨著位元的增加，Dadda乘法器與Wallace乘法器的閘延遲(gate delays)並沒有太大的增加，但是陣列乘法器呈現線性增加[26]，這是因為陣列乘法器的階層數目大約等於輸入位元，而Dadda乘法器與Wallace乘法器的階層數目大約等於輸入位元再取$\log_2$，因此Dadda乘法器與Wallace乘法器的速度較Array乘法器快。

若以 Dadda 乘法器與 Wallace 乘法器做比較，二者的階層數目相同，而 Wallace 乘法器的 CPA 字元長度較 Dadda 乘法器稍小，若將 CPA 使用 CLA，則 Wallace 乘法器跟 Dadda 乘法器的速度差不多。實際上以 Timemill 模擬的結果，二者的最差延遲(worst case delay)是很接近的[27]。

2.比較面積

比較三者的面積，Array 乘法器所需要的全加器最多，而 Wallace 乘法器所需要的半加器最多，相較之下 Dadda 乘法器的面積最小。以 Cadence 放置與繞線(place and route)工具做模擬，8 到 64 bits Wallace 乘法器的面積比 Dadda 乘法器大約多 4% 至 7% [27]，這是因為 Wallace 乘法器的半加器數量非常大的緣故。

3.結論

Array 乘法器的速度很慢，而且面積也較大，但好處是加法器的排列較規則。Dadda 乘法器的面積較 Wallace 乘法器小，而且它們的最差延遲非常接近，因此 Dadda 乘法器的速度快而面積也最小。這 3 種乘法器都可以加入管線暫存器(pipelined register)來提高速度，假設乘法器的每一階層皆加入一級管線暫存器，Dadda 乘法器因為階層數少所以管線暫存器的階層數也較少，而 Array 乘法器在每一階層的管線暫存器數量較少，但是總結來說，Array 乘法器的管線暫存器的階層數較多，因此 Array 乘法器的潛伏(latency)比 Dadda 乘法器較長，在二者每一階層的閘延遲都相同的情況下，Dadda 乘法器的速度還是較 Array 乘法器快。所以本論文選擇以 Dadda 乘法器做為實現功率意識之乘法器，並且 CPA 採用 Brent-Kung 進位-向前看加法器(carry look-ahead adder, CLA) ，以及加入管線暫存器(pipelined register)來提高速度。

3.3 管線正負號(pipelined signed) Dadda 乘法器之架構

在本節將介紹管線正負號 Dadda 乘法器之架構，以此架構產生功率意識乘法器。 Dadda 乘法器的 CPA，我們使用快速的 Brent-Kung 加法器[28]，首先我們先介紹用於 CPA 的 Brent－Kung 加法器。

3.3.1 Brent－Kung CLA 加法器

Brent-Kung 加法器可以減少 CLA 加法器的閘延遲，並且較有規則性[29]，因此我們選擇 Brent-Kung 加法器做為乘法器的 CPA。

以下將描述 Brent-Kung 加法器的數學運算式。令加法器的二個輸入為 A(i)與 B(i)，和(sum)為 S(i)，進位(carry)為 C(i)。其中 i 代表位元數，若是 n bit 加法，則 i＝0,1,2,……,n-1。一般加法可寫成

$$C(0)=0$$

$$C(i)=A(i)\cdot B(i)+A(i)\cdot C(i-1)+B(i)\cdot C(i-1)$$

$$S(i)=A(i)\oplus B(i)\oplus C(i-1)$$

$$S(n)=C(n-1)$$

其中・代表 AND 閘，＋代表 OR 閘，⊕代表 XOR 閘。

現在令產生(generate) G(i)與傳遞(propagate)P(i)分別為

$$G(i)=A(i)\cdot B(i) \qquad (13)$$

$$P(i)=A(i)\oplus B(i) \qquad (14)$$

則

$$C(i)=G(i)+P(i)\cdot C(i-1)$$

$$S(i)=P(i)\oplus C(i-1) \qquad (15)$$

Brent-Kung 定義運算子 o 為下式

[G(i),P(i)] o [G(i-1),P(i-1)]＝{G(i)＋[P(i)・G(i-1)], P(i)・P(i-1)}

圖 30 為運算子 o 之圖示，稱為進位-向前看產生元(carry look-ahead generator cell, CLG)。

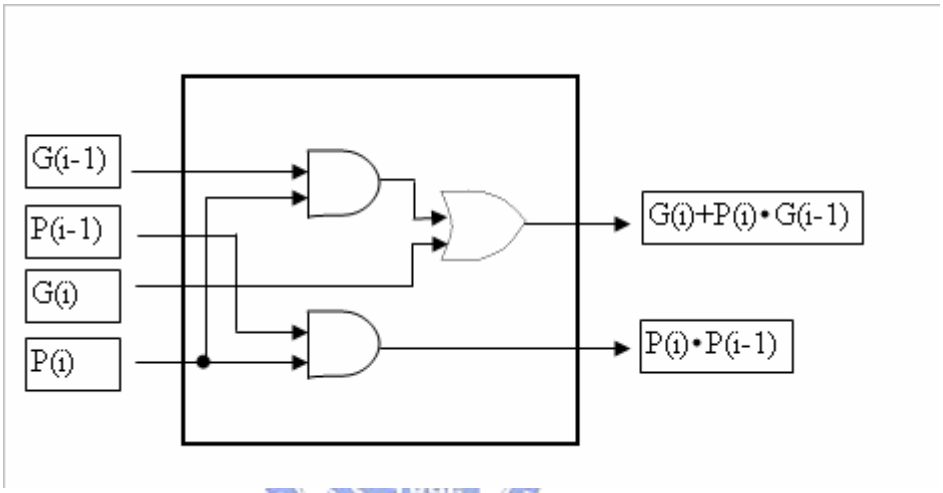

圖 30　進位-向前看產生元(CLG)

令

[g(i),p(i)]＝ [G(i),P(i)]　　if i＝0,

[G(i),P(i)] o [g(i-1),p(i-1)]　　if 1≦i≦n-1.

則根據推導則得到

[g(i),p(i)]＝[G(i),P(i)] o [G(i-1),P(i-1)] o …… o [G(0),P(0)]　　(16)

並且　　C(i)＝g(i)　　for i＝0,1,…,n-1　　(17)

圖 31 為 16 bit Brent-Kung 加法器，A、B 為 16 bit 的二個輸入，S 為 17 bit 的結果。我們將 Brent-Kung 加法器分為 Block A、Block B、及 Block C 三個部分，其中 Block B 又包含 4 個 Stage，以下將 Block A、Block B、及 Block C 分別描述。

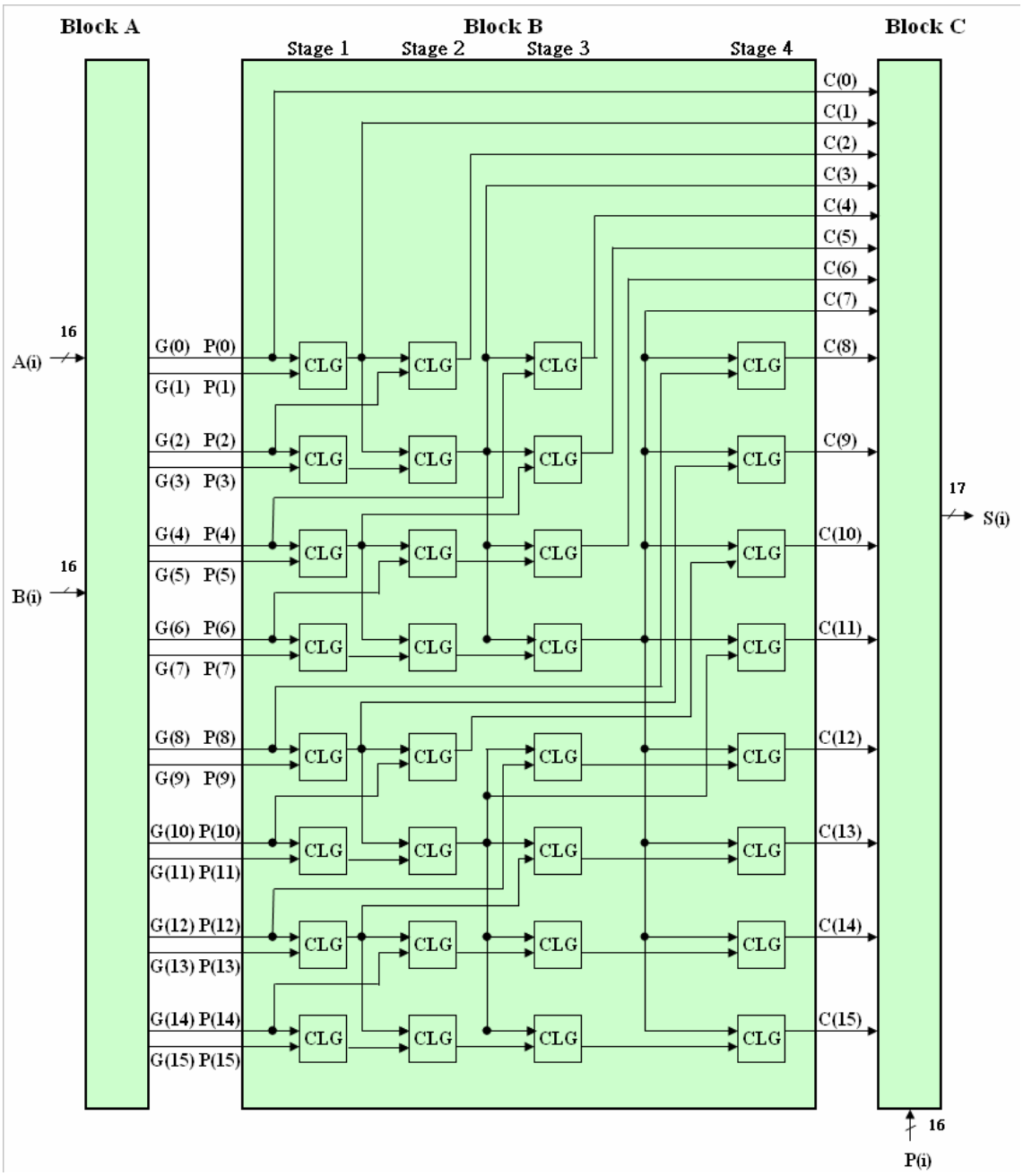

圖 31　16 bit Brent-Kung 加法器

Block A: 輸入 A 及 B，由(13),(14)式得到輸出 G 及 P。

Block B: 輸入 G 及 P，由(16),(17)式經過 CLG 而得到 C。圖 31 中 CLG 的每一條輸出及輸入皆是 2 bit，CLG 左上方的輸入為 G(i-1)與 P(i-1)，左下方的輸入為 G(i)與 P(i)，根據(16)式得到 CLG 輸出 g(i)與 p(i)，再由(17)式而得到 C(i)。Block B 所繪出的 CLG 組合具有最少的閘延遲，總共有 4 個 Stages。

Block C: 輸入 P 及 C，由(15)式求出最後的結果 S，這裡的輸入 P 是由 Block A 所產生，另外須注意的是 S(0)＝P(0)，S(16)＝C(15)。

3.3.2 管線暫存器之設置

根據前面幾節介紹的作法，可以產生正負號(signed) Dadda 乘法器。又為了達到高速的要求，在本節將介紹插入管線暫存器(pipeline register) 16 × 16 bit 正負號(signed) Dadda 乘法器。

圖 32 為 1 bit 管線暫存器，即 D 型正反器(Filp-Flop)，data_in 在經過一個 clock cycle 之後輸出至 data_out。

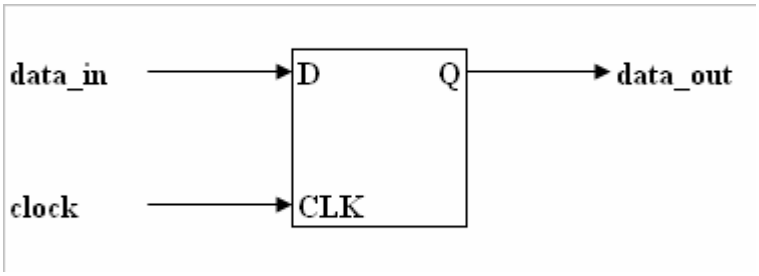

圖 32　1 bit 管線暫存器

管線暫存器 VHDL code 敘述如下:

```
library ieee;
use ieee.std_logic_1164.all;


entity reg is
 port( data_in: in std_logic;
       clk: in std_logic;
       data_out: out std_logic);
end reg;


architecture  beha of reg is
begin
 process(data_in,clk)
 begin
   if clk'event and clk='1' then
      data_out <= data_in;
   end if;
 end process;
end beha;
```

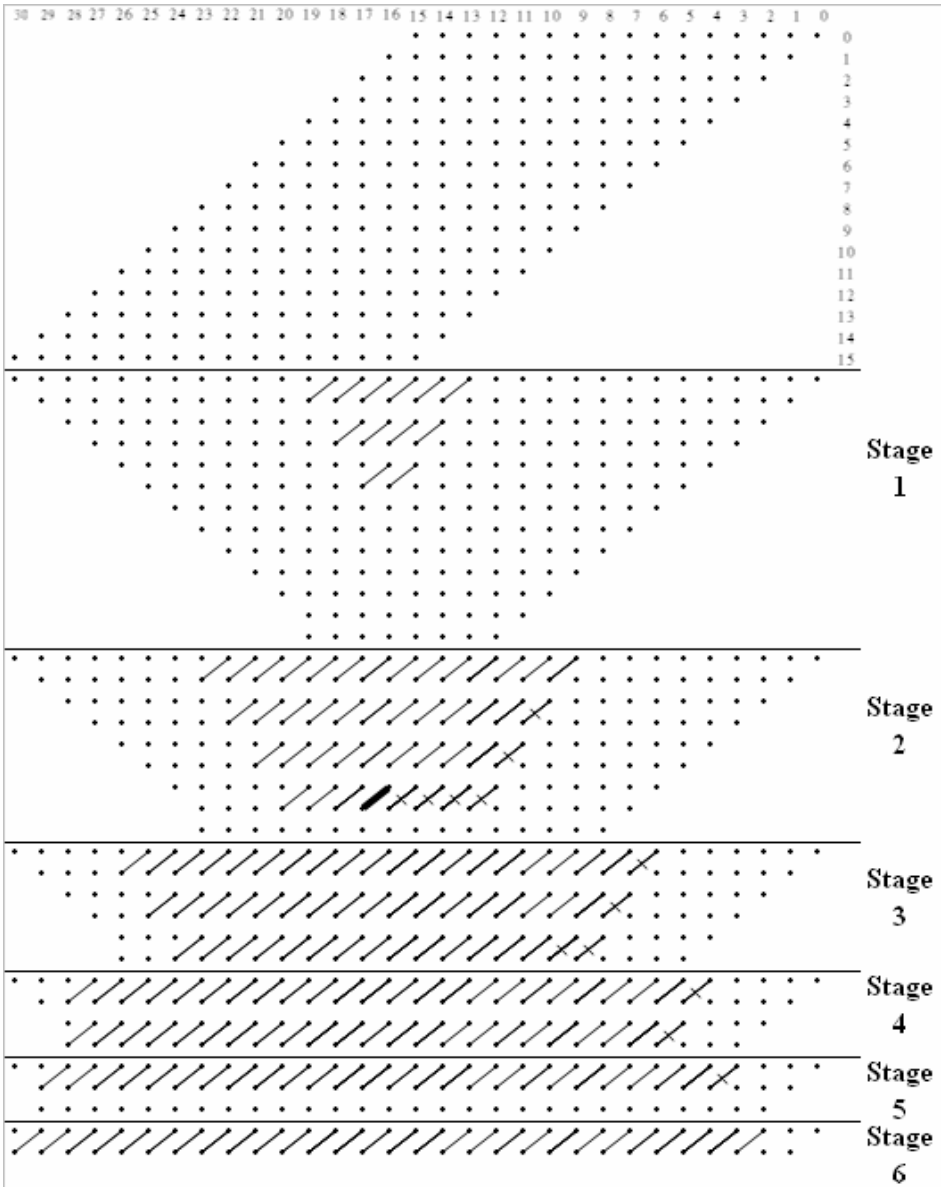

圖 33　16 × 16 bit 正負號 Dadda 乘法器的總和圖解

圖 33 為 16 × 16 bit 正負號 Dadda 乘法器的總和圖解，由下到上階層的列數分別為 2,3,4,6,9,13，總共有 6 個階層。原本位於 Stage 2　P16 列的加法器為半加器，根據 3.1 節 Baugh-Wooley 乘法演算法，欲將無正負號乘法改為正負號乘法而必須在 P16 列加 1，所以將該半加器改為全加器，如圖 33 Stage 2 中粗體斜線所示。

另外，需要將 S15,0、S15,1、……、S15,14 以及 S0,15、S1,15、……、S14,15 共 30 個部分乘積做反向，並且在 P30 的進位做反向產生 P31，即可將 16 × 16 bit 無正負號乘法變成正負號乘法。

圖 34 為 16 × 16 bit Dadda 乘法器的管線設置，X_in 與 Y_in 代表被乘數與乘數，Q 代表乘積。如表 6 所列，Dadda 乘法器的階層數目為 6，為了達到高速的要求，平均在 Dadda 乘法器每二層全加器插入管線暫存器。1st 管線暫存器置於產生部分乘積的 AND 閘與 Dadda 的第 1 層加法器之後，2nd 管線暫存器置於 Dadda 的第 2 層與第 3 層加法器之後，3rd 管線暫存器置於 Dadda 第 4 層與第 5 層加法器之後。

另外在 CPA 的管線設置方面，16 × 16 bit Dadda 乘法器的 CPA 字元長度為 30，亦即所使用的 Brent-Kung 加法器為 30 bit 的加法，其 Block B 的 CLG 階層數是 5。因為 CLG 的閘延遲(gate delay)與加法器相當，因此我們也是在每二層 CLG 插入管線暫存器。4th 管線暫存器置於 Dadda 的第 6 層加法器與 Brent-Kung 加法器的 Block A 之後，5th 管線暫存器置於 Block B 的第 1 層與第 2 層 CLG 之後，6th 管線暫存器置於 Block B 的第 3 層與第 4 層 CLG 之後。

因此 16 × 16 bit Dadda 乘法器的管線設置，總共有 6 個階層的管線暫存器，並且加入輸入及輸出暫存器。下面章節的實驗結果即由此 16 × 16 bit 管線(pipelined)正負號(signed) Dadda 乘法器產生。

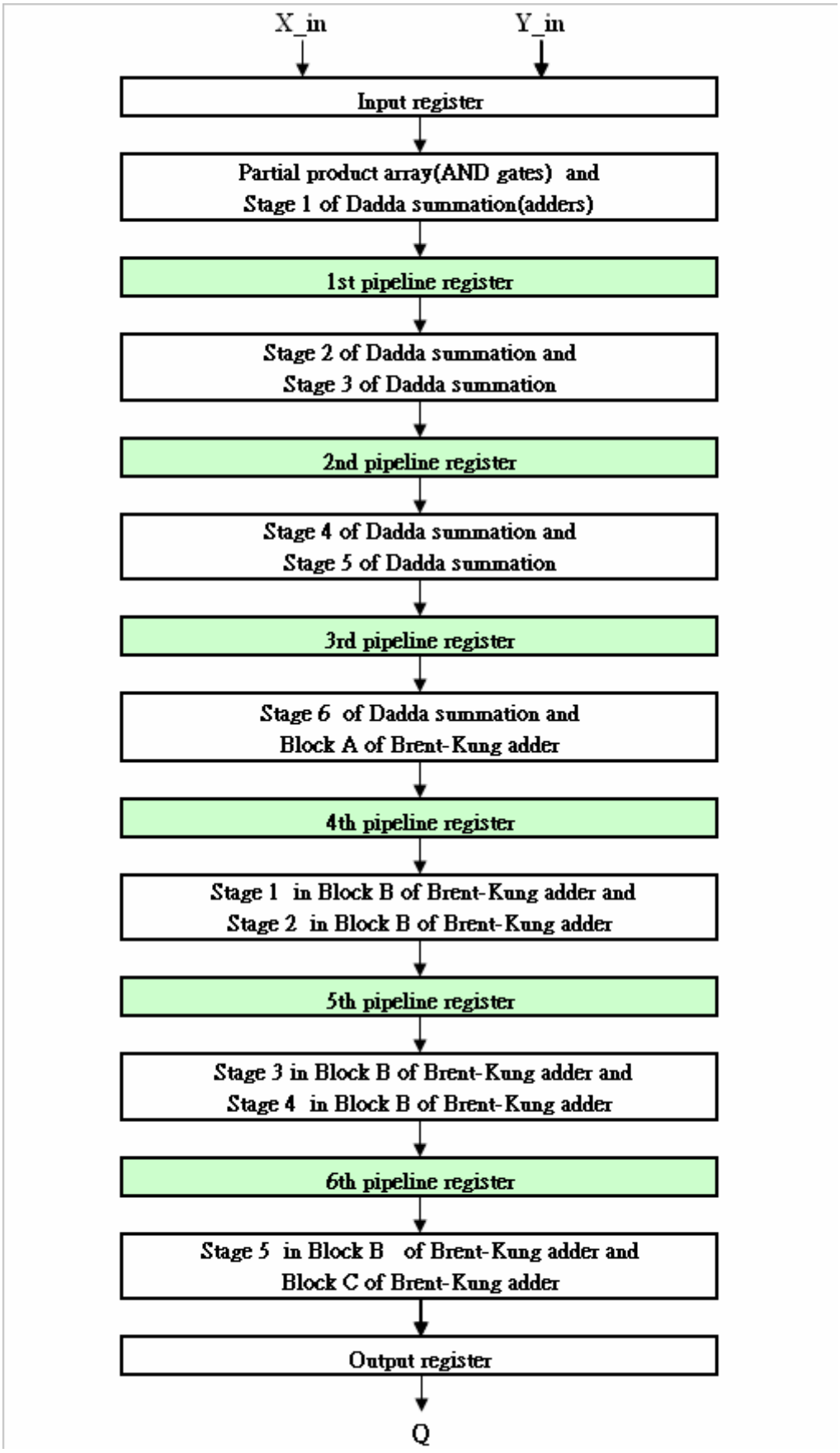

圖 34　16 × 16 bit Dadda 乘法器的管線設置