



國立中山大學資訊工程學系

碩士論文

Department of Computer Science Engineering

National Sun Yat-sen University

Master Thesis

適用於低功率應用的多重模式浮點乘加器

Multi-Mode Floating-Point Multiply-Add Fused Unit for

Low-Power Applications

研究生：余其坤

Kee-Khuan Yu

指導教授：鄺獻榮 博士

Dr. Shiann-Rong Kuang

中華民國 100 年 7 月

July 2011

國立中山大學研究生學位論文審定書

本校資訊工程學系碩士班

研究生余其坤（學號：M983040073）所提論文

適用於低功率應用的多重模式浮點乘加器
Multi-Mode Floating-Point Multiply-Add Fused Unit for Low-Power Applications

於中華民國 100 年 7 月 26 日經本委員會審查並舉行口試，
符合碩士學位論文標準。

學位考試委員簽章：

召集人 蕭勝夫 蕭勝夫 委員 鄭獻榮 鄭獻榮

委員 張雲南 張雲南 委員 郭可驥 郭可驥

委員 陳銘志 陳銘志 委員 _____

指導教授(鄭獻榮) 鄭獻榮 (簽名)

學年度: 99
學期: 2
校院: 國立中山大學
系所: 資訊工程學系
論文名稱(中): 適用於低功率應用的多重模式浮點乘加器
論文名稱(英): Multi-Mode Floating-Point Multiply-Add Fused Unit for Low-Power Applications
學位類別: 碩士
語文別: 中文
學號: M983040073
頁數: 61
研究生(中)姓: 余
研究生(中)名: 其坤
研究生(英)姓: Yu
研究生(英)名: Kee-Khuan
指導教授(中)姓名: 鄭獻榮
指導教授(英)姓名: Shiann-Rong Kuang
關鍵字(中): 多重模式浮點乘加器, 重複式乘法, 截斷式加法, 低功率
關鍵字(英): iterative multiplication, low power, truncated addition, multi-mode floating point multiply-add-fused



致謝

首先誠摯的感謝指導教授鄺獻榮博士，老師用心的教導是我得以一窺低功率領域的深澳，不時的討論並且指證我正確的方向，且很有耐心的給予教導，讓我在碩士班兩年獲益匪淺，老師對學問的嚴謹更是我學習的典範。

在這兩年的日子，實驗室裡共同生活的點滴，學術上的討論、言不及義的閒扯、趕作業的革命情感等等，都是我這些日子的原動力。感謝眾位學長姐、同學、學弟妹的共同陪伴，讓兩年的研究生活變得炫麗多采。

接著，我要感謝實驗室的學長俊評、坤益、勁麟、凱程、英程、家惠和威廷，在我研究過程中給予知道和協助。同學弘譯、偉強、保辰、時捷以及桓偉，能在研究室一起生活、一起學習的日子，是我未來寶貴的回憶。學弟建綱、一平、詮勝以及佳玟，謝謝你們這一段時間的協助和陪伴。

摘要

在數位訊號處理或計算圖形的系統中，浮點乘法與浮點加法為最常使用的運算，且每次浮點乘法運算後緊接著浮點加法運算的頻率很高，因此為了達成高效能與低成本的目標，通常將浮點乘法與浮點加法合併為一個單元以執行浮點乘累加，稱為浮點數乘加器(Floating-Point Multiply-Add Fused, MAF)。現今行動裝置的發展越來越蓬勃，效能以及電源持久性的要求成為主要發展趨勢，因此低功率的機制和技術越來越受重視。因此，我們提出一種多重模式浮點乘加器，以重複式乘法(Iterative Multiplication)和截斷式加法(Truncated Addition)之方式，設計出具有多種誤差運算模式的浮點乘加器。此乘加器共有七種誤差模式，其中浮點乘累加運算有三種誤差模式、單獨浮點乘或單獨浮點加運算各有二種誤差模式。在浮點乘累加運算中，提供使用者三種誤差模式，分別產生0%、0.328%和1.107%之運算誤差，而其中的0%誤差為IEEE754單精度浮點乘累加運算。另外在浮點乘法和浮點加法運算中，各有兩種誤差模式讓使用者選擇，分別對浮點乘法產生0.328%、0%之誤差以及為浮點加法引入0.781%、0%之誤差，而其中的0%誤差為IEEE754單精度浮點運算。

本論文所提出的多重模式浮點乘加器架構與 IEEE754 單精度浮點乘加器比較，電路面積減少了 5%而電路延遲增加 23%，如此即可達到多重模式的效果。在功率消耗方面，使用本論文所提出之浮點乘加器並且在允許誤差的模式狀況下，執行圖片格式轉換之 RGB 轉 YUV 應用程式時均能達到降低功率消耗的效果。

Abstract

In digital signal processing and multimedia applications, floating-point(FP) multiplication and addition are the most commonly used operations. In addition, FP multiplication operations are frequently followed by the FP addition operations. Therefore, in order to achieve high performance and low cost, multiplication and addition are usually combined into a single unit, known as the FP Multiply-Add Fused (MAF). On the other hand, the mobile devices nowadays are rapidly developing. For this kind of devices, performance and power sustainability have to become the major trend in the research area. As a result, the mechanisms to reduce energy consumption become more important. Therefore, we propose a multi-mode FP MAF based on the concept of iterative multiplication and truncated addition, to achieve different operating modes with different errors. This MAF, with a total of seven modes, includes three modes for the FP multiply-accumulate operations, two modes for single FP multiplication operation and single FP addition operation, respectively. FP multiply-accumulate operations provide three modes to user, and this three modes have 0%, 0.328% and 1.107% of error. The 0% error is the same with the standard IEEE754 single-precision FP Multiply-Add Fused operations. For FP multiplication and FP addition operations, the proposed MAF allows users to choose two kinds of error modes, which are 0%, 0.328% error for FP multiplication and 0%, 0.781% error for FP addition. The 0% error is the same with the standard IEEE754 single-precision floating-point operations.

When compared with the standard IEEE754 single-precision FP MAF, the proposed multi-mode FP MAF architecture has 4.5% less area and increase about 22% delay to achieve the effect of multi-mode.

To demonstrate the power efficiency of proposed FP MAF, it is used to perform the operations of FP MAF, FP multiplication, and FP addition in the application of RGB to YUV format conversion. Experimental results show that, the proposed multi-mode FP MAF can significantly reduce power consumption when the modes with error are adopted.

目錄

Chapter 1. 緒論.....	1
1.1 研究動機.....	1
1.2 論文大綱.....	2
Chapter 2. 研究背景與相關研究.....	3
2.1 IEEE 754 規格簡介.....	3
2.2 浮點數加法原理	4
2.3 浮點數乘法原理	6
2.4 布斯乘法器簡介	7
2.5 壓縮樹.....	11
2.6 捨進.....	14
2.7 傳統浮點乘加器架構.....	18
Chapter 3. 提出的多重模式浮點乘加器	23
3.1 簡介.....	23
3.2 多重模式浮點乘加器架構.....	24
3.3 改良重複式浮點乘法器.....	26
3.4 改良重複式浮點乘法器之誤差	31
3.5 截斷式加法	32
3.6 加法運算之誤差	36
3.7 特殊浮點乘與浮點加運算	37
3.8 多重模式浮點乘加器之誤差	39
3.9 控制電路.....	40
Chapter 4. 實驗結果及比較	42
Chapter 5. 結論與未來研究方向.....	47
5.1 結論.....	47
5.2 未來研究方向	47
參考文獻.....	48

圖目錄

圖 2-1 IEEE754 規格.....	3
圖 2-2 基本浮點加法器架構.....	5
圖 2-3 布斯編碼電路[7].....	8
圖 2-4 部份乘積產生器[23].....	9
圖 2-5 Radix-4 布斯乘法之部分乘積	9
圖 2-6 Radix-4 布斯乘法器電路架構.....	10
圖 2-7 4-2 壓縮器	11
圖 2-8 台積電(TSMC) 半加器元件.....	12
圖 2-9 台積電(TSMC)全加器元件.....	12
圖 2-10 台積電(TSMC) 4-2 壓縮元件.....	12
圖 2-11 8x8 壓縮樹	13
圖 2-12 浮點乘法運算以及浮點加法運算.....	14
圖 2-13 RP(round to positive infinity)與 RNE(round to nearest/even).....	16
圖 2-14 RI(round to infinity)與 RM(round to minus infinity)	16
圖 2-15 傳統浮點乘加器之架構.....	18
圖 2-16 假數 A 對齊(a)對齊前(b)當 $d \leq 0$ (c)當 $d \geq 0$ (d)當 $d \geq 74$	20
圖 2-17 加法運算結果之第一個“1”的位置(a) CASE I (b)CASE L (c) CASE K	22
圖 3-1 多重模式浮點乘加器的架構	25
圖 3-2 改良重複式乘法器乘執行之區域(a)無誤差模式(b)誤差模式 ...	27
圖 3-3 改良重複式乘法器之架構.....	28
圖 3-4 執行無誤差模式乘法之乘積壓縮狀況.....	30
圖 3-5 執行有誤差模式乘法之乘積壓縮狀況.....	30
圖 3-6 改良重複式乘法器之乘積誤差	31
圖 3-7 clock gating cell	32
圖 3-8 截斷加法機制	33
圖 3-9 Ladner-Fischer Parallel Prefix Adder 進位網路	34
圖 3-10 (a)Prefix adder 元件一 (b) Prefix adder 元件二	35
圖 3-11 執行浮點加法所關閉之電路區域.....	38
圖 3-12 執行浮點乘法所關閉之電路區域.....	38
圖 3-13 控制器架構圖	41

表格目錄

表格 2-1 Radix-2 布斯編碼.....	7
表格 2-2 Radix-4 布斯編碼.....	8
表格 2-3 RNE(round to nearest/even)	17
表格 3-1 多重模式浮點乘加器之模式及誤差	25
表格 4-1 傳統浮點乘法器與多重模式浮點乘法器之面積以及電路延遲 比較.....	44
表格 4-2 浮點乘加器執行浮點加運算之功率消耗比較.....	44
表格 4-3 浮點乘加器執行浮點乘運算之功率消耗比較.....	44
表格 4-4 使用傳統浮點乘加器與多重模式浮點乘加器於應用程式(RGB 轉 YUV)	44
表格 4-5 多重模式浮點乘加器執行 RGB 轉 YUV 圖片格式轉換之圖片 效果.....	45
表格 4-6 各浮點乘加器之比較	46

Chapter 1. 緒論

1.1 研究動機

多媒體盛行的時代，對於數值範圍的要求提高，而浮點數(floating-point)比定點數(fixed-point)擁有更大的動態範圍和精確度，故浮點數更具備運算上的優勢。然而，應用程式的運算過程中(如:數位訊號處理、影像處理等)，連續的浮點乘累加為基本運算方式，故浮點乘加器(Floating-Point Multiply-Add Fused, MAF)能更有效率地執行(運算式 $A+B\times C$)。許多應用程式可使用浮點乘加器來提升其方程式的執行效率，像是數位訊號和影像處理[1]，快速傅立葉轉換(Fast Fourier Transforms, FFT)[2]，有限脈衝響應(Finite-Impulse Response, FIR)等。許多嵌入式系統內，已把乘加器整合在晶片內，比如 IBM[17]，HP[18]、[19]，MIPS[20]，INTEL[21]、[22]等，當遇到浮點乘運算接著浮點加運算時，即可直接進行浮點乘累加之運算。當只需要執行浮點乘或浮點加時，把其中一個輸入設定成“0” ($0+B\times C$)或“1”($A+1\times C$)即可。故浮點乘加器不但可以單獨執行乘法或乘加法的運算，它同時可以提高乘加運算效能。

近年行動裝置系統越來越發達，如智慧手機、數位相機、PDA 等系統，其內部的運算量也越來越複雜，因此效能以及電源持久性的要求已經成為主要研發的趨勢。而降低電路消耗功率能直接影響電源的持久性，故低功率的機制和技術也越來越受重視，例如:降低電路面積、降低運算精確度、減少電路切換功率或靜態功率消耗、clock gating 等。

另一方面，許多的多媒體應用中，在運算精確度上是可以容許誤差的，例如:圖片壓縮或轉換並顯示在解析度較低的顯示器，或 3D 遊戲中使用者允許背景物件之運算存在誤差等等。因此，我們提出一種可以讓使用者選擇誤差模式並且能夠執行浮點乘累加、浮點乘和浮點加運算的電路架構，在可容忍的誤差限

制下盡可能的節省功率消耗。不同於傳統乘加器把輸入設定成“1”或“0”，本論文提出的電路架構，可以在執行浮點乘或浮點加時，關閉不必要的電路，進而達到省電的效果。

1.2 論文大綱

本論文分成四大章節。第一章介紹研究動機和論文大綱。第二章節介紹傳統浮點乘法器、布斯乘法器、傳統浮點加法器和傳統浮點乘加器。第三章將深入介紹我們所提出的多重模式浮點乘加器和相關的細節部分。第四章是實驗結果，比較傳統的浮點乘加器與多重模式浮點乘加器之間的電路面積、電路延遲和功率消耗等。第五章討論未來的研究方向以及結論。

Chapter 2. 研究背景與相關研究

2.1 IEEE 754 規格簡介

此論文所提出的浮點乘加器是以 IEEE754 標準規格為基礎所設計，因此我們先簡單說明 IEEE754 標準規格[11]。IEEE754 分別有單精度浮點數(single-precision floating point)和雙精度浮點數(double-precision floating point)的規格，其數值的表示皆為 $(-1)^S \times 2^{E-bias} \times 1.f$ 。以單精度為例，位元長度為 32 位元，從最高位元(MSB)至最低位元(LSB)，其中第 1 個位元表示正負號(sign bit) S ，接著 8 個位元表示指數(exponent bits) E ，接著 23 個位元表示它的有效位數或稱為假數(fraction or mantissa bits) f 。在假數前面隱藏一個 1，而在單精度的指數部分基底(bias)為 127，即指數 E 範圍在 0~255，而 $E-bias$ 的範圍則是 -127~128 之間。圖 2-1 為 IEEE754 浮點數之規格。

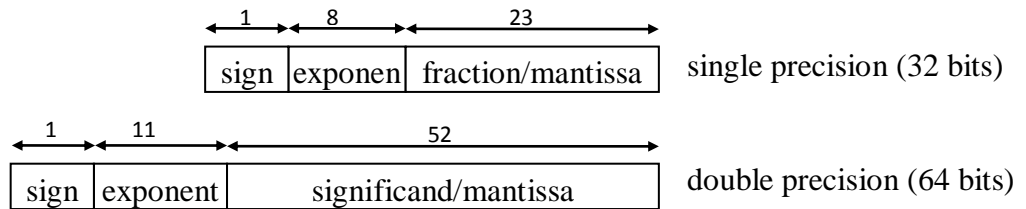


圖 2-1 IEEE754 規格

本論文使用單精度實作，以下為本論文統一使用之浮點符號：

$$A = (-1)^S \times 2^{E-bias} \times 1.f = (-1)^S \times 2^e \times 1.f$$

S :浮點數正負號(sign)，1 位元。

f :假數(mantissa)，23 位元。

E :指數(exponent)以 127 為基底，8 位元。

e :指數 $E-bias$ 。

2.2 浮點數加法原理

浮點數加法運算中，假數的部分必須先作移位之動作(移位量為兩浮點數指數之相差量 $E_a - E_b$)，將小數點對齊之後方可相加。指數和正負號的部分則以較大的浮點數為基準，當兩個浮點數之正負號相異，即代表執行減法，硬體電路實現減法方式可用 1's 補數或 2's 補數之加法電路實現，本論文使用 2's 補數加法運算電路實做。以下為單精度 IEEE754 浮點數加法運算規則：

$$A = (-1)^{S_a} \times 2^{E_a - bias} \times 1.f_a = (-1)^{S_a} \times 2^{e_a} \times 1.f_a$$

$$B = (-1)^{S_b} \times 2^{E_b - bias} \times 1.f_b = (-1)^{S_b} \times 2^{e_b} \times 1.f_b$$

(此例假設 $A \geq B$ ，若 $A < B$ 則把 A，B 互換即可進行相加)

$$\begin{aligned} A + B &= (-1)^{S_a} \times 2^{E_a} \times 1.f_a + (-1)^{S_b} \times 2^{E_b} \times 1.f_b \\ &= 2^{E_a} \times \{ (-1)^{S_a} \times 1.f_a + (-1)^{S_b} \times 1.f_b \times 2^{E_b - E_a} \} \end{aligned}$$

由上述的公式可以看出其假數部分是以無號數(Sign-Magnitude)來表示，即可視為浮點數大小和正負號是分開表示(f 表示假數， S 表示正負號)。圖 2-2 是基本的浮點加法器架構，其運算步驟如下：

1. 比較 E_a 與 E_b 的大小，將較大的指數值當成運算中的指數基準。同時 mux1 與 mux2 可進行 mantissa 的交換，浮點指數較大的 mantissa 將由 mux1 輸出，mux2 將輸出浮點指數較小的 mantissa。
2. 若兩浮點數正負號相異，此時對負號的假數作 2's 補數(反向加 1)的動作，但 bit invert 只作反相的動作，而加 1 則是由加法器的 Cin 輸入。
3. E_a 與 E_b 之間的絕對差值($|E_a - E_b|$)輸出到 Right Shifter 並對指數較小的假數作右移，此動作稱為小數對齊。

4. 2's Complement adder 以 2's 補數實作加法運算，而在第 2 步驟 bit invert 少加的 1 將由加法器的 Cin 輸入，最後計算出假數相加的結果。在加法運算的同時，由於假數相加後之和第一個 1 的位置不固定，故必須依賴 LOP(Leading One Predictor)[13]預測和的第一個 1 落在第幾個位元並且輸出移位量。
5. 由於加法運算的結果是以 2's 補數表示，而 IEEE754 浮點規格的假數部分是以無號數表示，故加法運算產生之 Cout 將協助將和由 2's 轉為無號數表示式，此時也修正最後輸出的正負號。
6. 假數正規化(Normalization)把 LOP 所給予的移位量對答案左移，並且對基準指數作增加或減少。若 LOP 預測左移量大於對齊電路之移位量，則減少指數基準；LOP 預測左移量小於對齊電路之移位量，則增加指數基準。
7. 最後 Round 作捨位的動作，以完成浮點加法之運算。

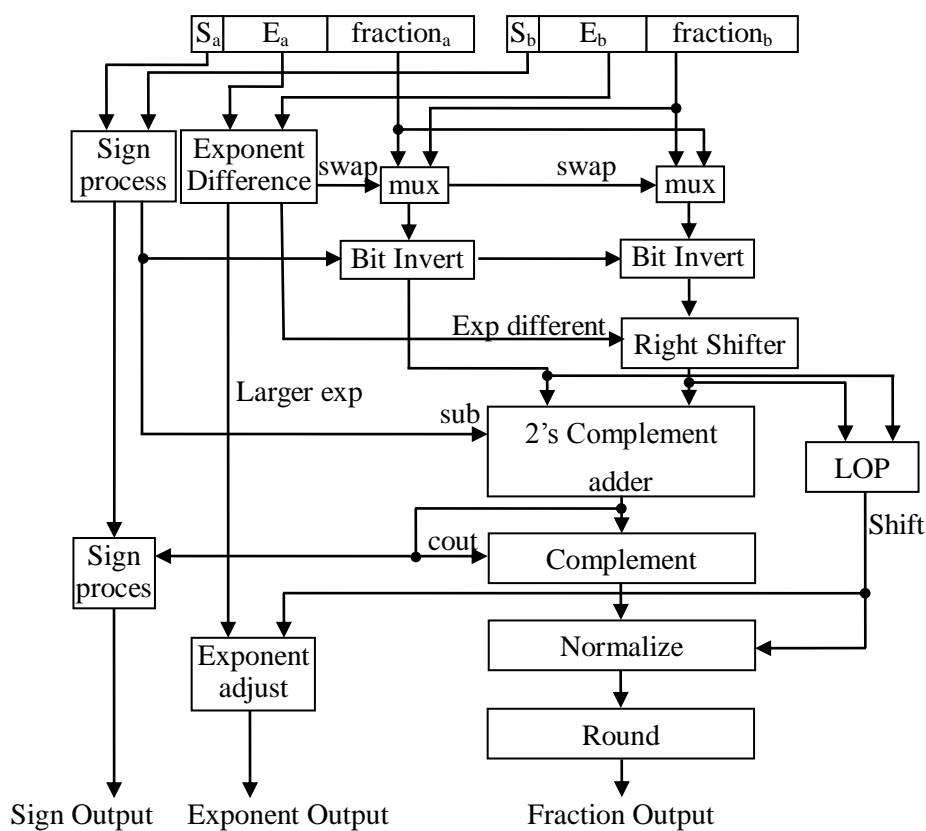


圖 2-2 基本浮點加法器架構

2.3 浮點數乘法原理

浮點數乘法較為簡單。由於浮點數之假數為無號數，故假數乘法與無號數乘法運算相同，指數部分為兩浮點指數相加，正負號部分為兩浮點數之正負號作互斥或運算，以下為單精度 IEEE754 浮點乘法運算規則：

$$A = (-1)^{S_a} \times 2^{E_a - \text{bias}} \times 1.f_a = (-1)^{S_a} \times 2^{e_a} \times 1.f_a$$

$$B = (-1)^{S_b} \times 2^{E_b - \text{bias}} \times 1.f_b = (-1)^{S_b} \times 2^{e_b} \times 1.f_b$$

$$A \times B = (-1)^{(S_a \oplus S_b)} \times 2^{(E_a + E_b - 127) - \text{bias}} \times (1.f_a \times 1.f_b)$$

目前實現乘法運算之乘法器有以下幾種：

- 串並式乘法器(serial-parallel multiplier structure)[3][4]
- 陣列式乘法器(array multiplier structure)[5]
- 樹狀式乘法器(tree multiplier structure)[7][8]
- 布斯編碼乘法器(Booth multipliers structure)[6][9]

文獻[10]比較了各種乘法器的特色及優缺點，由於布斯編碼乘法器能有夠效地減少部分乘積(Partial Product)的列數，而部分乘積的減少也使得電路的最長路徑延遲(critical path)縮短，故本論文乘法運算的電路架構都以布斯編碼乘法器實現。

2.4 布斯乘法器簡介

透過適當的編碼，使得乘法運算過程中產生的部分乘積(Partial Product)的數目減少，進而加快了乘法器的運算速度。而部分乘積數目減少也使得電路面積減小，這也表示產生最後結果所需的運算量也會減少，因此亦可達到降低功率消耗的效果。

布斯演算法就是根據表格 2-1 中的原則對乘數進行編碼。Radix-2 布斯演算法一次看兩個位元，現在的位元(亦即 Y_i)和右邊一位的位元(亦即 Y_{i-1})，用這兩個位元進行編碼。編碼完後會得到三種結果，一種是編碼 0，其表示把被乘數(Multiplicand)乘上 0 作為此列的部份乘積，並且部分乘積之 sign 為 0。另一種編碼為 +X，即把被乘數乘上 1 作為此列的部分乘積並且 sign 為 0。如果編碼出來的結果是 -X，此時就要把被乘數取 2 的補數並且其 sign 為 1。Radix-2 布斯演算法雖然無法減少部份乘積的數目，但因為其特殊的編碼方法，可以跳過連續 0 或者連續 1 的運算，減少運算時所消耗的功率。

表格 2-1 Radix-2 布斯編碼

Y_i	Y_{i-1}	OP
0	0	0
0	1	+X
1	0	-X
1	1	0

Radix-4 布斯演算法[14] 一次對三個位元進行編碼，比 Radix-2 布斯演算法多了一個位元，因此其編碼器就更加的複雜，編碼完後的結果共有 5 種，分別為 -X、+X、0、-2X、+2X。其中 +2X 的部分乘積就是把 +X 的值往左移一位，同理 -2X 即為 -X 往左移一位，其編碼的方法和結果如表格 2-2 以及以下方程式所示。

表格 2-2 Radix-4 布斯編碼

Y_{2i+1}	Y_{2i}	Y_{2i-1}	OP	neg_i	two_i	one_i
0	0	0	+0	0	0	0
0	0	1	+X	0	0	1
0	1	0	+X	0	0	1
0	1	1	+2X	0	1	0
1	0	0	-2X	1	1	0
1	0	1	-X	1	0	1
1	1	0	-X	1	0	1
1	1	1	-0	0	0	0

$$neg_i = y_{2i+1}(y_{2i+1}y_{2i-1})'$$

$$two_i = y'_{2i+1}y_{2i}y_{2i-1} + y_{2i+1}y'_{2i}y'_{2i-1}$$

$$one_i = y_{2i} \oplus y_{2i-1}$$

Radix-4 布斯編碼方法有很多種類，其硬體的架構也有很不同的設計。本論文中使用的是 NPR3b[23]的硬體架構，因為 NPR3b 把輸入(0, 0, 0)和(1, 1, 1)的 neg 、 two 、 one 都編碼成一樣(都是為 0)，這樣可以降低切換時所消耗的功率，其硬體電路架構如圖 2-3 所示。

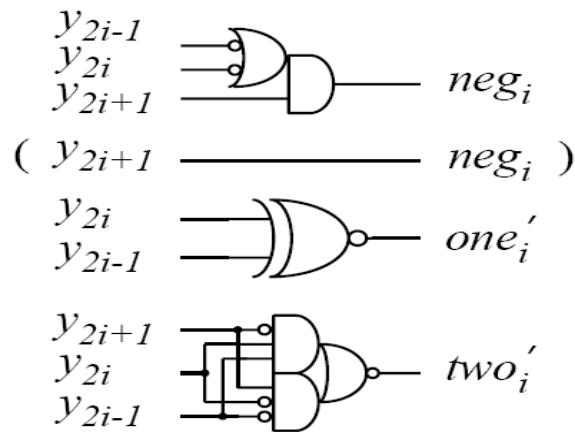


圖 2-3 布斯編碼電路[7]

經過編碼後，需要再經過一個部份乘積產生器(PP generator)，把產生的 neg、two、one 訊號加以運算來產生我們所需要的部份乘積。部分乘積產生器的硬體電路如圖 2-4 所示，其中的 nx_{j-1} 輸入腳位是由前面一組部份乘積產生器所傳過來的。

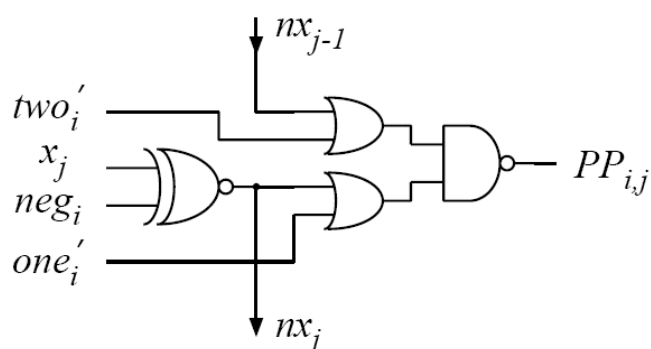


圖 2-4 部份乘積產生器[23]

經過部份乘積產生器後，就可以得到一系列的部分乘積，與這一系列部分乘積的符號位元(neg)。假設乘數有 26 bits，就會產生 13 列的部份乘積和 13 個 1 位元的符號位元。由於第 13 列永遠為正，故第 13 列之符號位元(neg)可不加入乘積內。圖 2-5 為 radix-4 布斯乘法所產生之乘積圖，圖 2-6 為 radix-4 布斯乘法器電路架構。

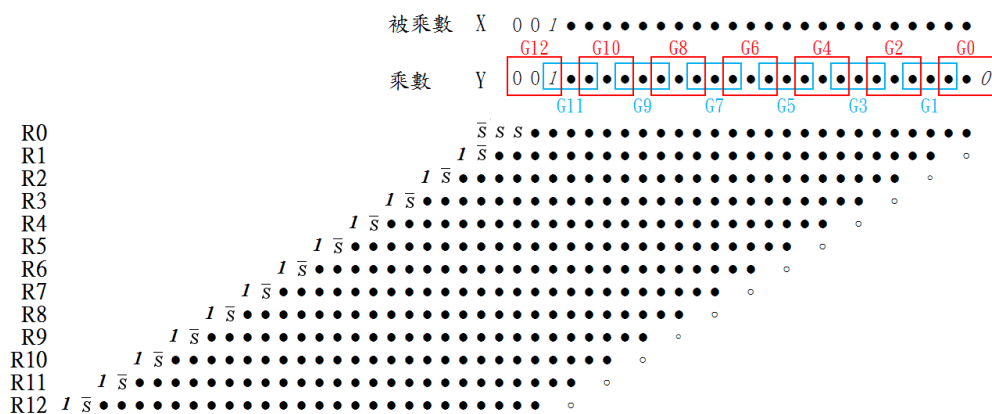


圖 2-5 Radix-4 布斯乘法之部分乘積

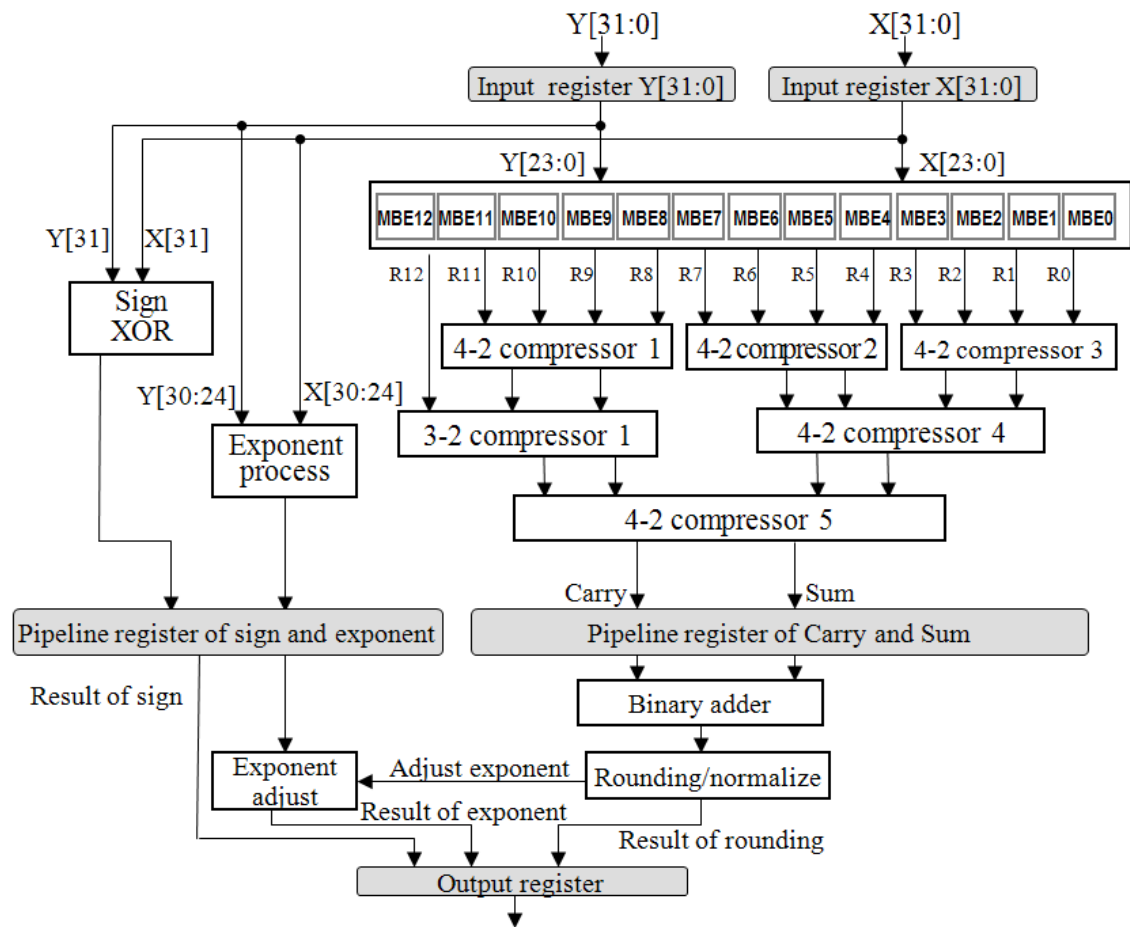


圖 2-6 Radix-4 布斯乘法器電路架構

2.5 壓縮樹

我們得到多列的部分乘積後，必須把全部的部分乘積相加即是乘法運算的答案。然而對每一列部分乘積都以加法器執行相加的動作，會造成很長的電路延遲，故使用壓縮樹的方式減少電路延遲。壓縮樹之目的在於把多列數壓縮成兩列(其兩列稱為 sum 和 carry)，而沒有加法器中連帶進位(propagate carry)的問題。而這個部分需要用到一系列的元件來降低樹的高度，其中包括半加器(half adder or 2-2 counter)、全加器(full adder or 3-2 counter)、4-2 壓縮器[24][25][26]等，圖 2-7 之 4-2 壓縮器是由 2 個全加器所組成的，此為基本的 4-2 壓縮電路。壓縮器的設計直接影響乘法器之電路延遲，故我們使用 TSMC 0.13um[32]之壓縮器，如圖 2-8、圖 2-9、圖 2-10。圖 2-11 以 8×8 Radix-4 布斯乘法為例並且使用 Wallace tree 的壓縮過程。

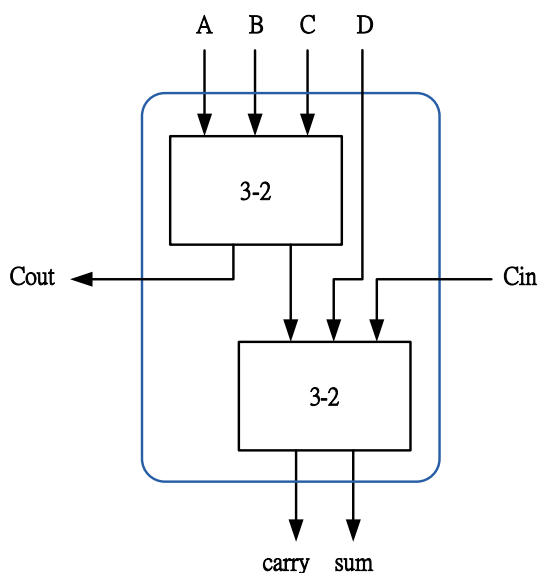


圖 2-7 4-2 壓縮器

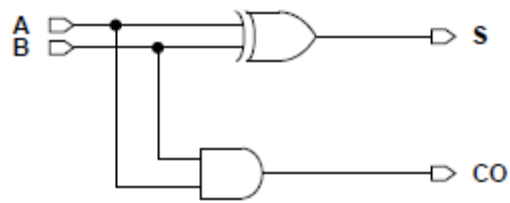


圖 2-8 台積電(TSMC) 半加器元件

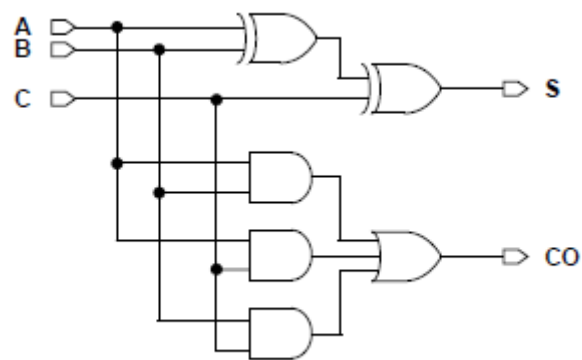


圖 2-9 台積電(TSMC)全加器元件

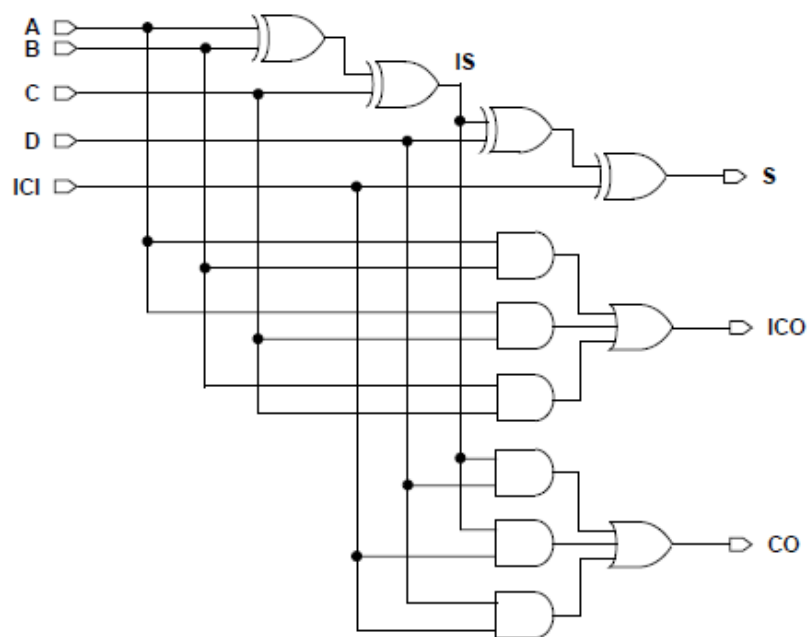


圖 2-10 台積電(TSMC) 4-2 壓縮元件

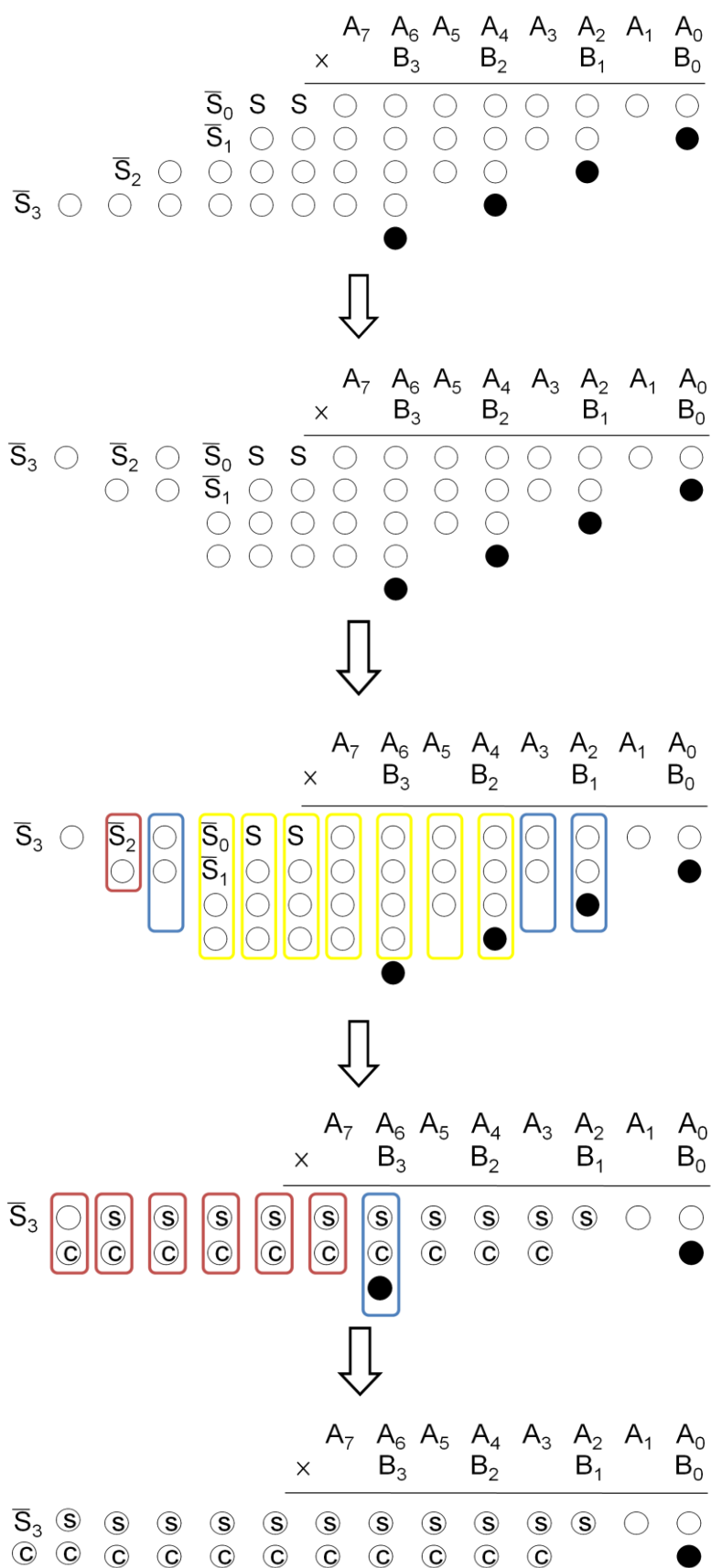


圖 2-11 8x8 壓縮樹

2.6 捨進

浮點乘法假數 $N\text{-bit} \times N\text{-bit}$ 會產生 $2N\text{-bit}$ 之乘積，而浮點加法之假數運算其結果也會產生多於 $N\text{-bit}$ 的結果(當兩個浮點數之指數相異時)，但是最後輸出的答案只能留下 $N\text{-bit}$ (圖 2-12)，所以必須捨去低於 $N\text{-bit}$ 權重較輕的位元，此動作稱為捨位。為了提高精確度，我們可以使用捨進(Round)的電路來產生最後的結果，捨進的方式有四種：RP(round to positive infinity)(圖 2-13)、RNE(round to nearest/even)(圖 2-13)、RI(round to infinity)(圖 2-14)、RM(round to minus infinity)(圖 2-14)，無論使用任何的捨進方法，其最多在留下的乘積之 LSB (least significant bit) 的地方加 1。

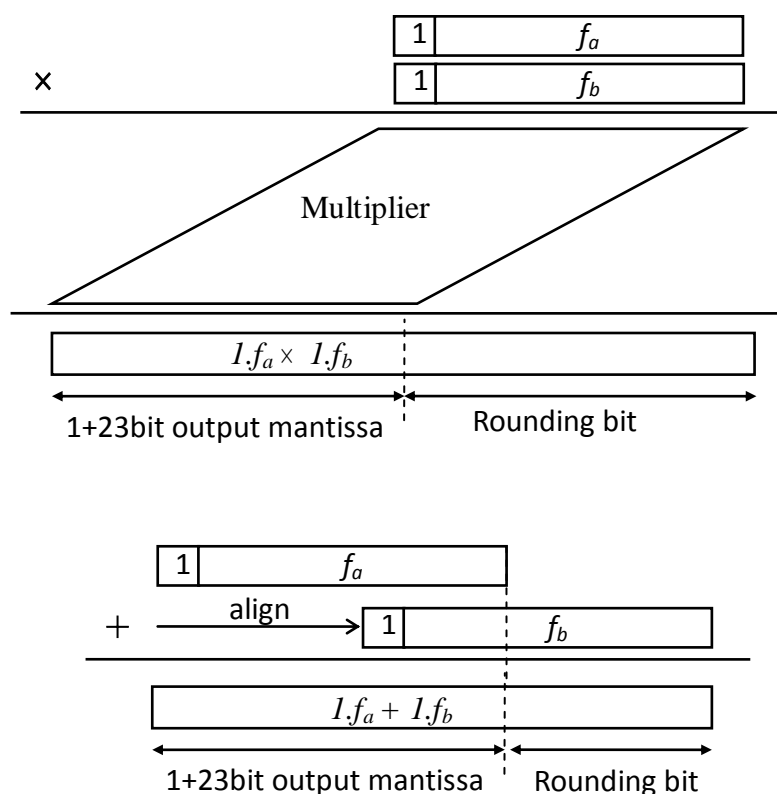


圖 2-12 浮點乘法運算以及浮點加法運算

通常在進行捨進之前的數值會介於兩個數之間。令捨進之前的數值為 x (其 x 介於 N_1 與 N_2 之間)，假設較大的數值為 N_1 ，小的數值為 N_2 ，設完成捨進後的數值為 X 。對 RM 來說，無論 x 是正數或者是負數，捨進後 $X = N_2$ 。對 RI 來說，就分成正、負兩種情況，若 x 為正數，則捨進後 $X = N_1$ ，若 x 為負數，則捨進後 $X = N_2$ 。

對 RP 來說其捨進無論正負都是取較大的數，也就是說 $X = N_1$ 。最後是 RNE，若 N_1 為奇數， N_2 為偶數，且 $x \leq 0.5$ 的話，則 $X = N_2$ ；但若 $x > 0.5$ 的話，則 $X = N_1$ 。如果 N_1 為偶數， N_2 為奇數， $x < 0.5$ ，則 $X = N_2$ ，反之若 $x \geq 0.5$ ，則 $X = N_1$ 。RNE 可以用下列示子來表示

$$\text{Rnear}(X) = \begin{cases} N1 & , \text{if } |x - N1| < |x - N2| \\ N2 & , \text{if } |x - N1| > |x - N2| \\ \text{even}(N1, N2) & , \text{if } |x - N1| = |x - N2| \end{cases}$$

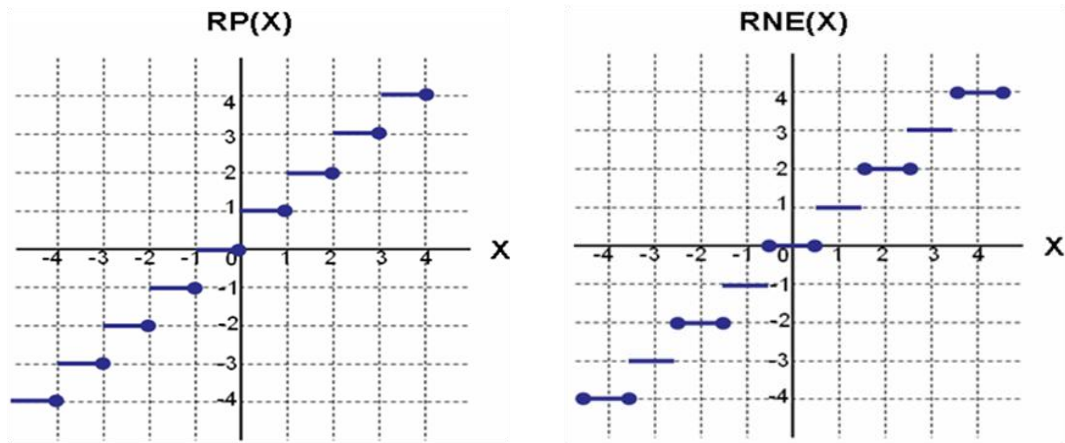


圖 2-13 RP (round to positive infinity)與 RNE (round to nearest/even)

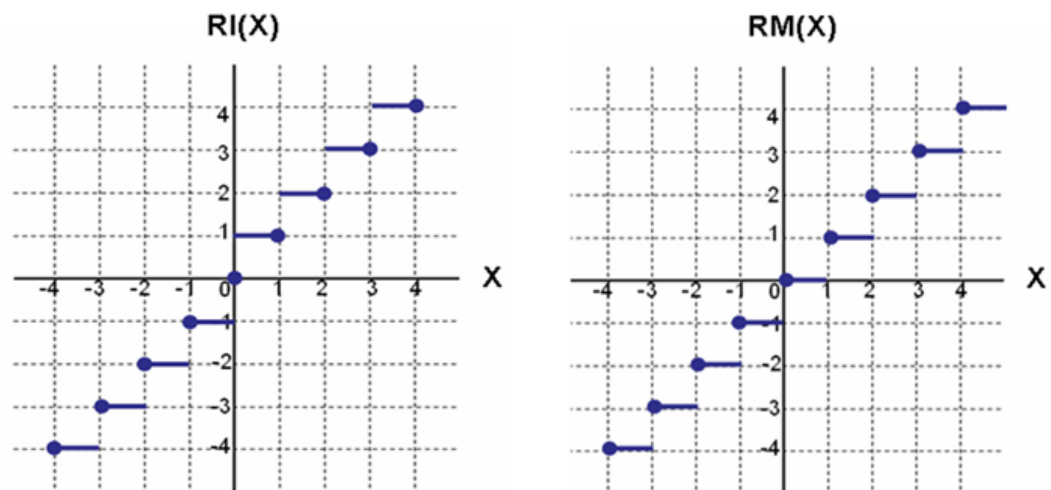


圖 2-14 RI (round to infinity)與 RM (round to minus infinity)

本論文以 RNE(round to nearest/even)實作，故在 rounding bit 中(圖 2-12 中，低於 N-bit 權重較輕的位元)，必須產生 round bit 以及 sticky bit，此 2 位元可以決定答案之假數的 LSB 是否需要進位。round bit 為 rounding bit 中的 MSB，剩下的 rounding bit 中若有至少一個位元為 1，則 sticky 為 1；若剩下的 rounding bit 全為 0，則 sticky bit 為 0。表格 2-3 為 RNE 判斷是否進位之真值表。

表格 2-3 RNE(round to nearest/even)

Before LSB	Rounding		Add to Mantissa	After Rounding LSB
	round	sticky		
X	0	0	Don't care	X
X	0	1	Don't care	X
0	1	0	0	0
1	1	0	1	0
X	1	1	1	\bar{X}

2.7 傳統浮點乘加器架構

2.2 和 2.3 節介紹的浮點乘法和浮點加法運算，是許多系統的應用程式中最常使用的運算，且多數應用程式有連續性乘累加的運算。因此為了提高運算效能，可以利用單一電路一次完成浮點乘累加的運算($Y=A+B\times C$)，此電路稱為浮點乘加器(Multiplication-Add Fused unit, MAF)[12]。

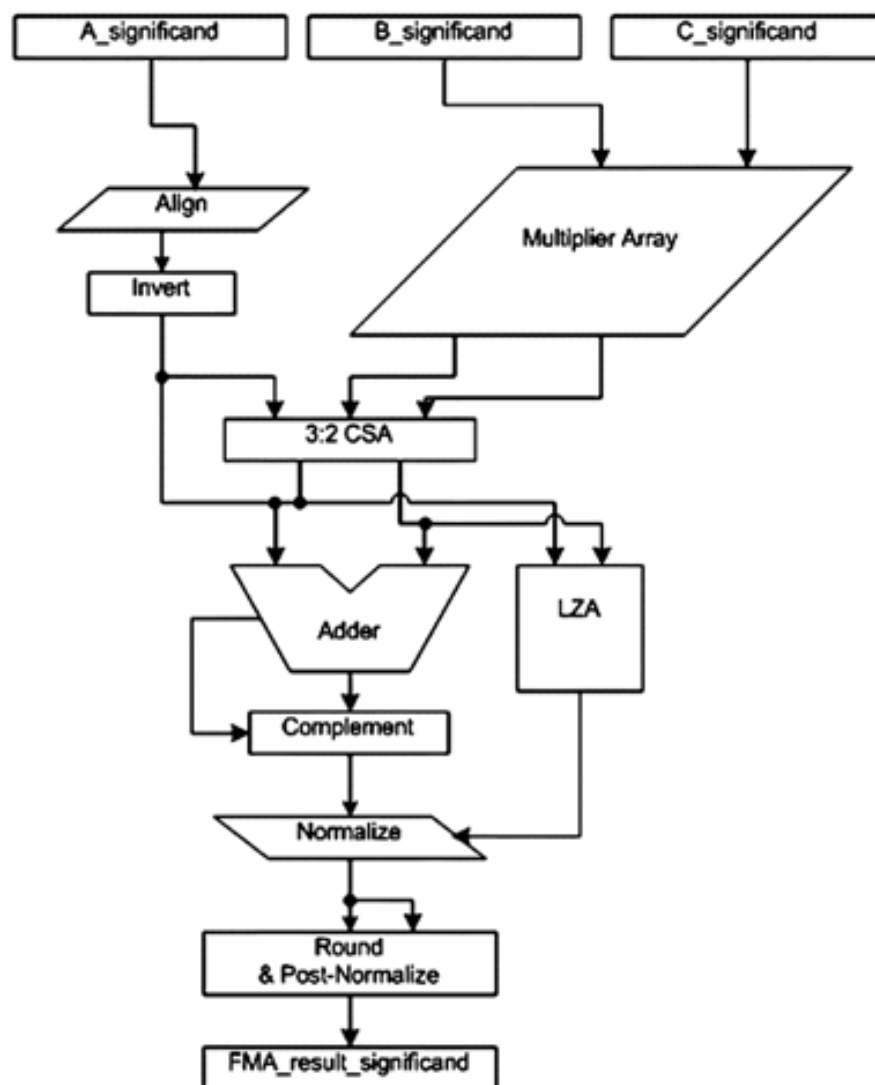


圖 2-15 傳統浮點乘加器之架構

圖 2-15 為傳統浮點乘加器的架構，其輸入主要為 A、B、C 三個浮點數，其輸出為 $A+B \times C$ 浮點數之運算結果。傳統浮點乘加器的特點在於進行乘法運算的同時進行加法運算的移位動作，且浮點乘加器只作一次的捨位動作，有效的提升電路共用性、電路效能及精準度。傳統浮點乘加器大致分成三個部分：

第一部分執行假數 $B \times$ 假數 C 的乘法運算，而假數 $B \times$ 假數 C 之部分乘積加總以 carry-save form 表示。在進行假數 $B \times$ 假數 C 乘法運算的同時，計算指數 A 與指數 B 加指數 C 之指數相差 $(E_a - (E_b + E_c))$ 並對假數 A 作移位(此動作稱為對齊 align)以對齊小數點。圖 2-16 為假數 A 對齊假數 $B \times$ 假數 C 的方式，一律針對假數 A 作右移，而且以 $27-d$ 為右移量(其中 $d = E_a - (E_b + E_c)$)，其範圍為 0 到 74 之間，其中假數 A 移位後的前後位元(虛線框內包含之位元)依乘加器是執行加或減填入 0 或 1(當浮點數 A 與浮點數 $B \times C$ 正負號異號時填入 1，同號時填入 0)，最後 Invert M_a 之位元數為 74 位元。對假數 A 右移超過 74 之位元，將視為捨進電路所需之 sticky bit(圖 2-16 之 sticky bit)。

第二部分為加法運算，主要把對齊後之 Invert M_a 與假數 $B \times C$ 作相加。由於假數 $B \times C$ 為兩列(carry-save form)，而 Invert M_a 為一系列，故需使用全加器(或 3:2 CSA)壓縮成兩列，再輸入至 2's 補數加法運算單元算出最後的和。浮點數之假數為 1.f 格式，故必須知道加法運算結果的第一個“1”所在的位置。

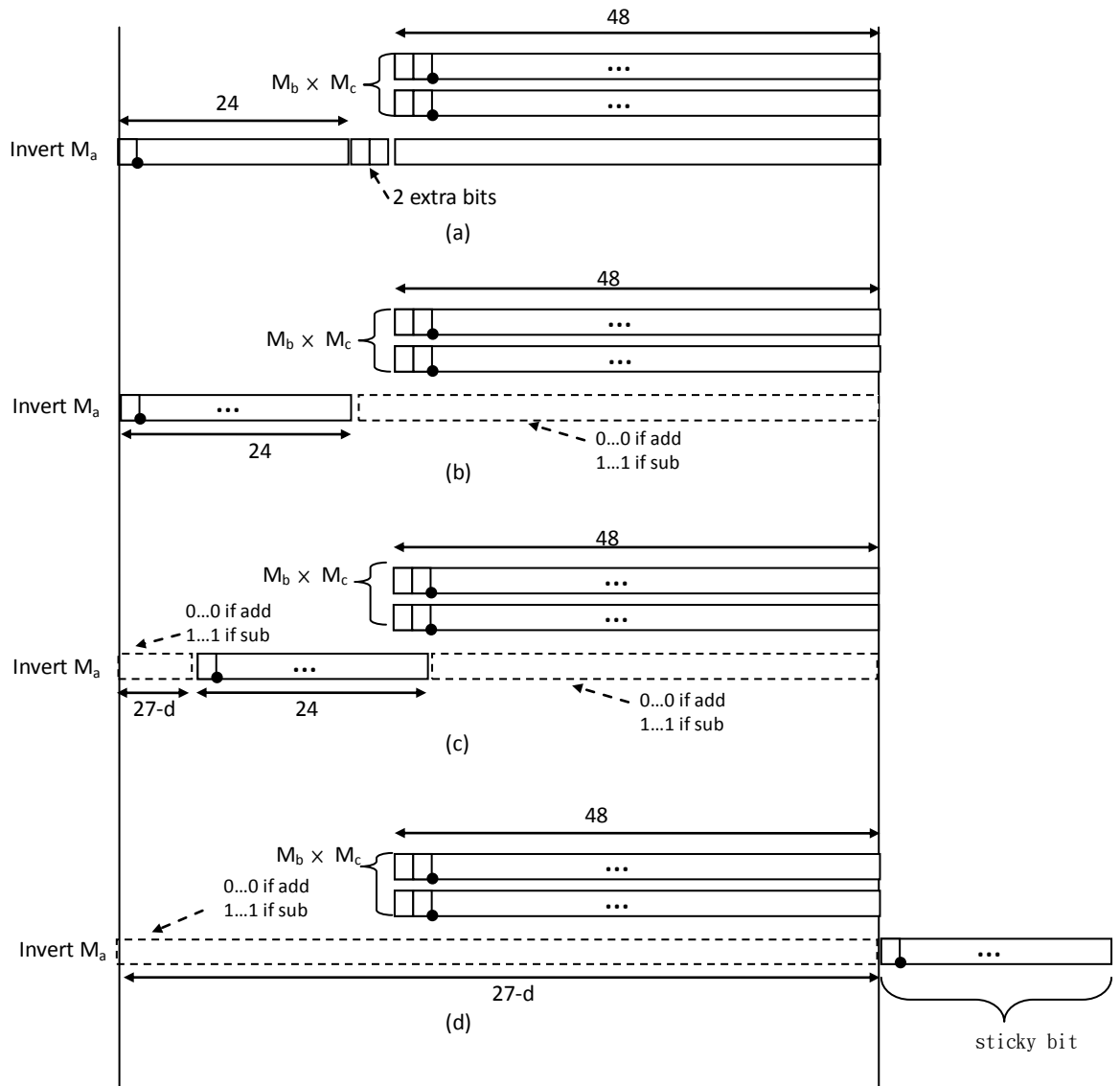


圖 2-16 假數 A 對齊(a)對齊前(b)當 $d \leq 0$ (c)當 $d \geq 0$ (d)當 $d \geq 74$

圖 2-17 將所有加法運算結果的第一個“1”可能的位置歸類為三種情況。圖 2-17(a)的 CASE I 為浮點數 A 之指數大於浮點數 BxC 之指數 27(含 27)以上，此時可以預知其加法運算結果(Result of adder)的第一個“1”會落在 MSB(第一位元)或是 MSB 的右邊位元(第二位元)。在圖 2-17(b) 的 CASE L 中，當 Exp_{diff} ($\text{Exp}_{\text{diff}} = |E_a - (E_b + E_c)|$) 大於 2 的狀況下，其加法運算結果的第一個“1”會落在第 $n-1$, n , $n+1$, 27, 28 或 29 位元，其中 n 為 A 假數對齊之移位量。圖 2-17(c)

的 CASE C 為浮點乘加器中最為複雜之狀況，在 $\text{Exp}_{\text{diff}} \leq 2$ 時，其加法運算結果的第一個“1”將無法根據浮點指數相差來預知，此時必須使用額外的電路來計算第一個“1”所在的位置，此電路稱為零位預測電路(Leading-Zero Anticipator, LZA)[13]。而另一種方式是直接在加法運算結果使用零位偵測電路(Leading-Zero Detection, LZD)[27]，如此同樣可以尋找到第一個“1”之位置，但此方式之電路最長路徑較長。由於電路延遲的考量，本論文使用零位預測電路(LZA)來完成。圖 2-17(a)和圖 2-17(b)之加法運算結果(Result of adder)之“1”有落在小數點右邊位置，為執行減法(當浮點數 A 與浮點數 BxC 異號)時，有可能形成借位所產生。Concordia MAF[27]把圖 2-17 所歸類的三種狀況中的 CASE L 再細分為 CASE J 和 CASE L，而形成四種狀況。另外，我們使用 Ladner-Fischer Prefix Adder[15] 架構來實作 2's 補數加法運算單元，若 2's 執行加法運結果產生進位則代表其結果為負，此時必須經過 Complement 把此數轉成無號數。

第三部分為正規化和進位，根據 LZA 偵測(CASE I 和 CASE L 可直接指定移位量)出第一個“1”所在位置之移位量進行移位。由於 CASE I 和 CASE L 所預測的第一個“1”位置有 3 bits 誤差，故在移位之後仍然需要經過多工器把第一個“1”移位成 1.f 的格式。在第一部分乘法和第二部分加法運算過程中，為了保持精準度，並沒有捨棄任何位元，但這樣將造成乘加運算的假數大於 23 bits，故必須進行四捨五入的機制(Round)。本論文一律使用 rounding to nearest even 來實做，而四捨五入後的結果即為浮點數 $A + (\text{浮點數 } B \times \text{浮點數 } C)$ 之結果。

浮點乘加器中，浮點加法運算的對齊動作與乘加器中的浮點乘運算同時進行，如此可以在浮點乘運算後直接進行加法運算，進而提高乘累加的效能。另一方面，乘加器只作一次的捨位(四捨五入)動作(若使用浮點乘法器與浮點加法器來達成 $A+B \times C$ 則會經過兩次的捨位機制)，進而提高了乘累加運算的準確度。

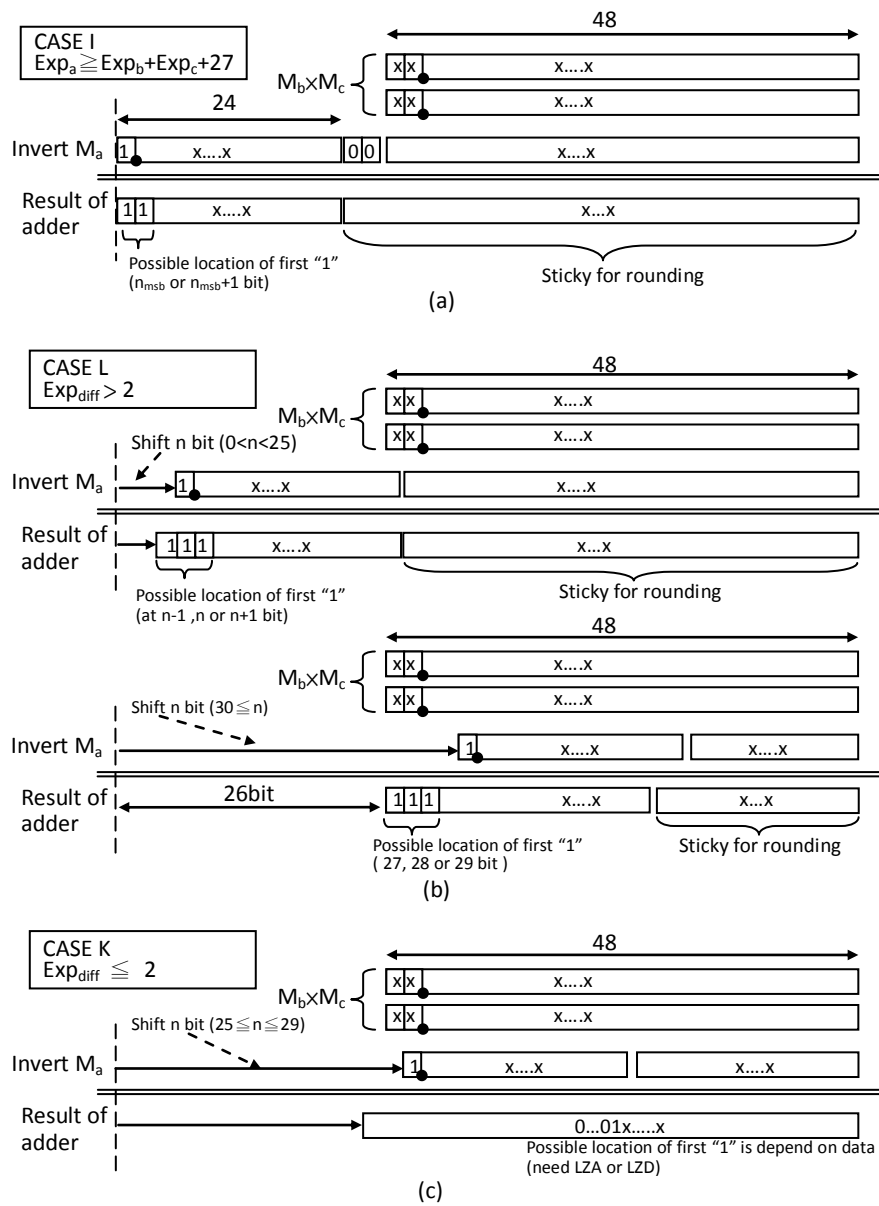


圖 2-17 加法運算結果之第一個“1”的位置(a) CASE I (b)CASE L (c) CASE K

Chapter 3.

提出的多重模式浮點乘加器

3.1 簡介

在很多的應用當中，並不是所有的運算都需要達到單精度浮點數之精準度，亦即許多的情況允許運算較大的誤差。比如在多媒體應用中，並不是每個畫面都需要最高精確度來呈現，畫面的失真是被允許的，而這些失真的大小可以從使用者所需要的影像品質去調整，最後達到使用者所能接受的失真範圍。在誤差被允許的情況下，我們提出一個可讓使用者選擇運算誤差的多重模式浮點乘加器，此浮點乘加器為乘累加的運算提供三種不同誤差模式，第一種模式為 0% 誤差(與標準 IEEE 754 浮點乘加運算結果比較)，第二種模式為 0.328% 誤差，第三種模式為 1.017% 誤差，而且本論文所提出之多重模式浮點乘加器在進行每筆運時算都可以任意指定誤差模式執行。

傳統浮點乘加器在執行浮點乘($B \times C$)或浮點加($A+B$)時，是設定其中一個參數為 0($0+B \times C$)或 1($A+B \times 1$)來達成，但是實際上電路仍然是執行乘累加的運算，故與乘累加運算有相同的功率消耗，這樣明顯地形成了功率的浪費。而我們所提出的浮點多重模式乘加器在單獨執行浮點乘運算時，能夠關閉浮點加法的對齊電路；若單獨執行浮點加運算時，能關閉浮點乘法電路，以達到節省功率消耗的效果。在浮點乘和浮點加運算中，各有兩種誤差模式可供使用者選擇，分別為浮點乘法 0%、0.328% 誤差和浮點加法 0%、0.781% 誤差，其中的誤差是以標準 IEEE754 單精度浮點運算結果為比較標準，所提出之多重模式浮點乘加器在進行每筆運算時都可以任意指定誤差模式執行。

傳統浮點乘加器產生的誤差與標準 IEEE754 浮點數運算的誤差相同，而我

們所提出的浮點乘加器不同的地方在於它可以利用較低的精確度模式來運算，在可以接受的情況下，達到降低功率消耗的效果，同時保持 IEEE754 浮點數運算之格式。

3.2 多重模式浮點乘加器架構

圖 3-1 是我們所提出的多重模式浮點乘加器架構，它主要由基本的浮點乘法器和浮點加法器架構組合而成，能提供浮點乘、浮點加和浮點乘累加三種基本運算。在浮點乘(圖 3-1 中的 Booth iterative multiplier)部分，運用了重複式浮點乘法器[16]的概念，並且改良其架構以配合多重模式浮點乘加器的特性，其優點是可以減少電路面積及電路延遲。由於重複式浮點乘電路延遲的減少，乘法電路能在一個時脈週期內運算兩次，故使用拴鎖器(Latch)以達成反饋(feedback)的效果。圖 3-1 之多工器(Mux)根據不同模式的運算，選擇加法運算中被加數為假數 D 或假數 B×C，以達到執行浮點乘累加或浮點加法的運算功能。在圖 3-1 中的截斷式加法(Truncated addition)可以在適當的模式下，運算出具有誤差的加法運算。當執行有誤差的模式，截斷式加法電路大部分的電路維持不切換狀態，使電路能夠節省功率消耗。

提出的多重模式浮點乘加器的多重模式來自於重複式浮點乘法器執行一次或執行兩次(透過回饋重複執行乘法)而形成不同的誤差。另外，在截取式加法電路中也有兩個模式，其中為不作截斷模式和截斷模式，這些模式的組合一共有如表格 3-1 中所示的 7 種模式提供給使用者選擇。

表格 3-1 中，OP 代表使用者欲執行加法(Add)、乘法(Mult)或乘累加(MAF)的運算，{add,mult}為位元模式選擇，Mode(模式)為電路實際執行之模式，error(accuracy)為電路所產生之誤差和精準度。

表格 3-1 多重模式浮點乘加器之模式及誤差

OP	{add,mult}	Mode	error(accuracy)
Add	1100	H-FADD	0%
Add	1000	L-FADD	0.781%(99.219%)
Mult	0011	H-FMUL	0%
Mult	0010	L-FMUL	0.328%(99.671%)
MAF	1111	H-MAF	0%
MAF	1110	M-MAF	0.328%(99.671%)
MAF	1010	L-MAF	1.107%(98.893%)

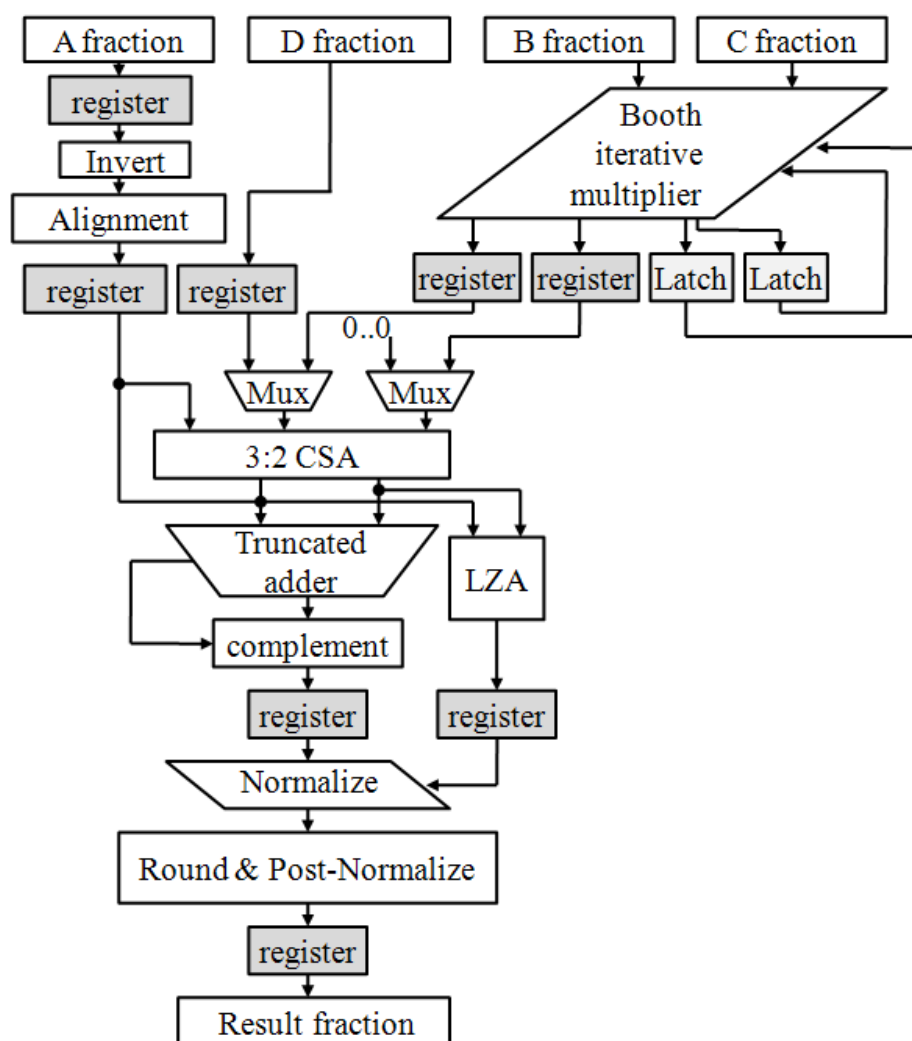
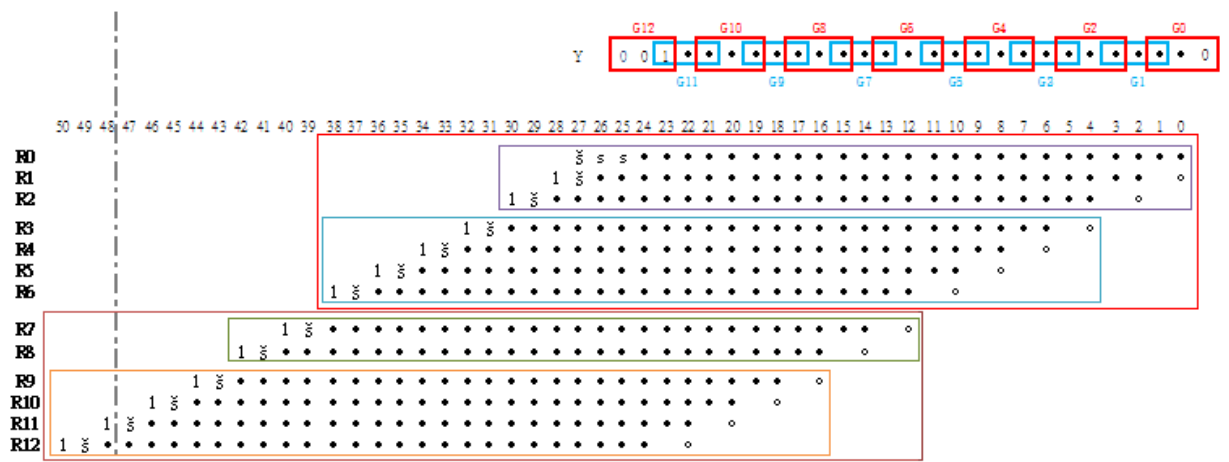


圖 3-1 多重模式浮點乘加器的架構

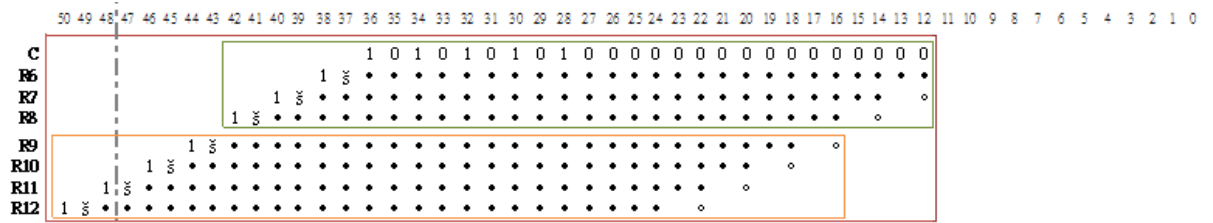
3.3 改良重複式浮點乘法器

本論文中使用 Radix-4 布斯乘法器來實現浮點乘加器中的假數 $B \times$ 假數 C ，乘法器中的 B 稱為乘數， C 稱為被乘數，其部分乘積如圖 3-2(a)所示一共有 13 列(R0~R12)。圖 2-5 為傳統的 Radix-4 布斯乘法器電路，透過 13 個布斯編碼器 MBE(Modify Booth Encode)產生 13 列部分乘積，再把 13 列的部分乘積運用壓縮方式，得到 13 列加總後的 carry-save form 表示式之乘積。

本論文提出的改良重複式乘法器，其電路設計只運用了 7 個 MBE 產生 7 列的部分乘積以及可以將 8 列部分乘積壓縮為 2 列的壓縮樹元件，若要計算 13 列乘積就必須運算兩次(透過反饋機制)，而運算一次就只能計算 7 列。此乘法電路架構比傳統的乘法電路面積小且電路延遲也較短。只執行一次運算時，會有誤差的產生，但電路功率消耗比傳統的乘法電路小，我們希望藉由這樣的重複式乘法器電路來達到低功率的效果。由於電路延遲較短，故使用拴鎖器(Latch)來達成一個週期運算兩次的機制，此機制必須使用電路的 2 倍頻率之時脈($\frac{1}{2}$ clk)訊號來控制部分電路。一個時脈週期中，在 $\text{clk}=1$ 和 $\text{clk}=0$ 分別控制多工器的選擇訊號，且必須滿足乘法器的電路延遲為 $\frac{1}{2}$ clk(在合成時乘法電路延遲的 Timing Constraint 必須滿足 $\frac{1}{2}$ clk)，這樣就能確保電路能在一個時脈週期運算兩次。以下將介紹圖 3-3 乘法電路在兩種不同模式運算時，電路之運作狀況。在執行有誤差的乘法模式時，為了降低誤差，因此加入了補償常數，此補償方式會在 3.4 節中詳細討論。



(a)



(b)

圖 3-2 改良重複式乘法器乘執行之區域(a)無誤差模式(b)誤差模式

carry-save form 位元減少，以減少 Pipeline register RS。無誤差模式乘法運算壓縮過程，如圖 3-4。

具有誤差的乘法運算模式中，只要把 R6~R12 和 C(補償常數)加總成 carry-save form，如圖 3-2(b)和圖 3-5。圖 3-3 中，當 $\text{clk}=1$ 時($\frac{1}{2}$ clk 的第一個工作時脈)，M1 選擇左邊的 B[25:13]給 MB0~MB6 作布斯編碼並且產生 R6~R12 的部分乘積，M2 和 M3 則選擇左邊的輸入(C 模組電路根據 mode 輸出 {1010101010,15'd0} 補償常數)。在產生 R6~R12 的部分乘積和 C 的補償常數後，透過 3 組(2 層)的 4-2 compressor 電路壓縮成 carry-save form。此 carry-save form 為 R0~R6 和 C 補償常數的加總，將寫入拴鎖器(Latch)中。當 $\text{clk}=0$ 時($\frac{1}{2}$ clk 的第二個工作時脈)，保持 $\text{clk}=1$ 的控制狀態，電路不再進行切換，拴鎖器之輸出並不會被 M2 和 M3 選擇，8-bit adder 輸入由 8-bit AND 強制設定成 0，故不影響其計算結果，最後所得到的 carry-save form 為 R6~R12 和 C 的加總。具誤差模式乘法運算壓縮過程如圖 3-5 所示。

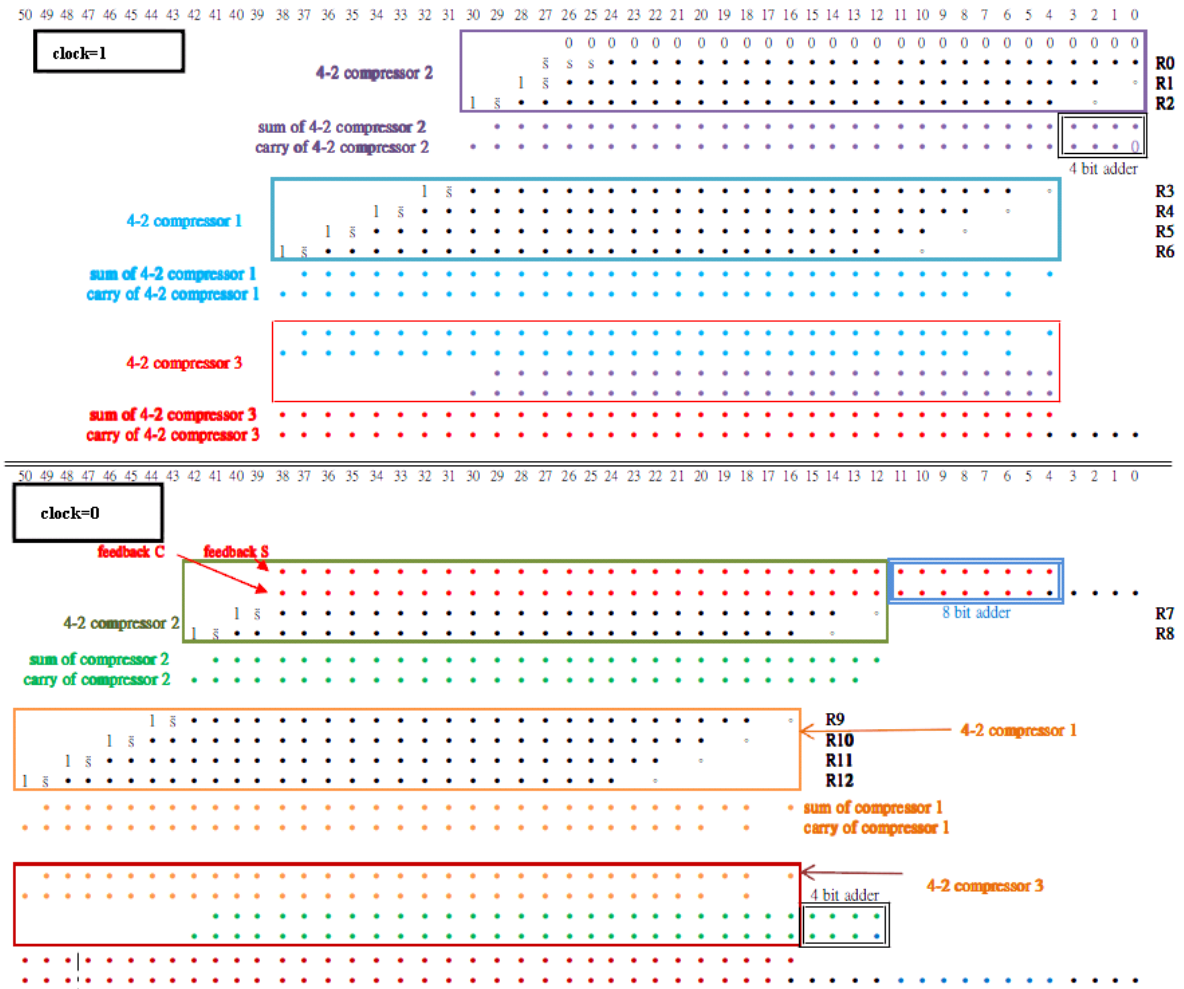


圖 3-4 執行無誤差模式乘法之乘積壓縮狀況

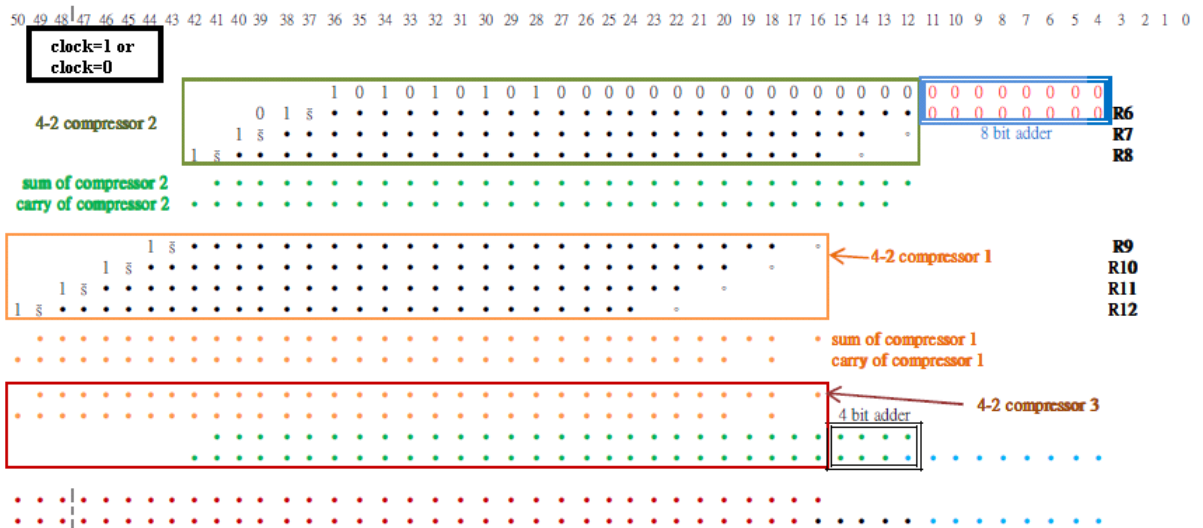


圖 3-5 執行有誤差模式乘法之乘積壓縮狀況

3.4 改良重複式浮點乘法器之誤差

在 3.3 節中介紹之重複式浮點乘法器存在兩種模式，其中有誤差的乘法模式之誤差來自於沒有加入 R0~R5 的值，故我們以最大誤差方式來評估誤差值，亦即把 R0~R5 全部的值(·和š)都以 1 來取代，以計算最大誤差值。其中 R0~R5 每列的 MSB 固定為 1，而與乘數和被乘數無關，故設計了補償常數 C，將這些 1 補入電路中。如圖 3-6 所示，將 R0~R5 加總(排除每列 MSB 之 1)，其值為 3.28e-03。由於乘法器中，最小假數乘積值為 1×1=1，故改良重複式浮點乘法器執行誤差模式時，最大誤差及準確度百分比為：

$$\text{FMUL}_{\text{error}} = \frac{3.28 \times 10^{-3}}{1} \times 100\% = 0.328\%$$

$$\text{FMUL}_{\text{accuracy}} = \frac{1 - 3.28 \times 10^{-3}}{1} \times 100\% = 99.672\%$$

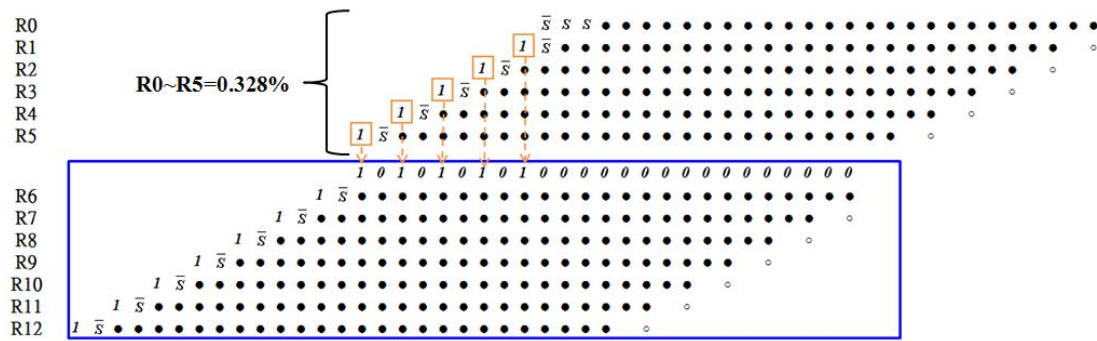


圖 3-6 改良重複式乘法器之乘積誤差

3.5 截斷式加法

在乘法運算後($B \times C$)得到的 carry-save form 將與對齊後之假數 A (Align M_a) 進行 3-2 壓縮，以得到 $A + (B \times C)$ 的 carry-save form，之後必須再交由加法電路把此兩列數值加成一列。我們提出一個截斷式加法電路(Truncated Adder)，只需要增加一些 clock gating cell 和 AND 閘，便可以達成具有兩種誤差模式的加法運算。在截斷模式下，能夠降低電路切換動作次數進而達到低功率效果，同時其誤差亦在可容忍之範圍內。電路增加 clock gating 的特性為借由攔截 CLK 訊號，進而阻止暫存器寫入數值，同時使用一個 clock gating 元件能代替多個暫存器輸入端之多工器，降低暫存器的功率消耗與減少電路面積的好處，圖 3-7 為暫存器加入 clock gating cell 之情況。

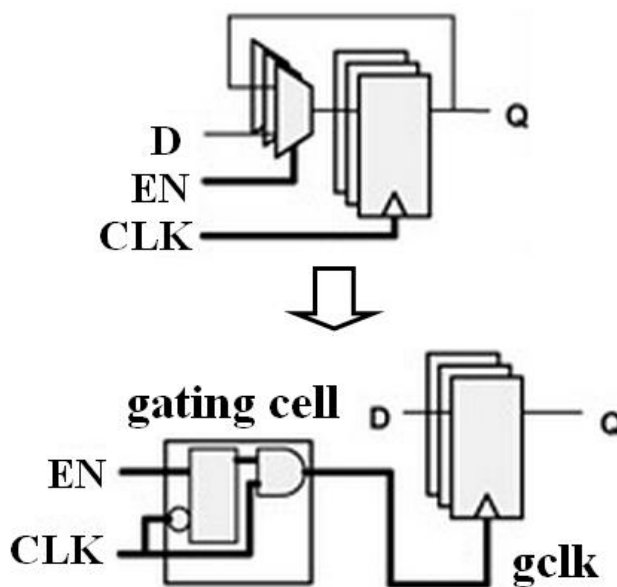


圖 3-7 clock gating cell

圖 3-8 中標示了欲達到截斷加法所需要攔截(gating)的訊號位置，亦即在乘法電路輸出之暫存器低位元和對齊後之 Align Ma 加上 clock gating 的機制，使這些訊號都維持上一個工作時脈的訊號，進而減少了 Mux、3:2 CSA、Truncate Adder 和 Complement 電路的切換動作。

由於 Truncate Adder 電路 LSB 38 位元都是前一個運算的訊號，此時必須防止進位影響第 38 位元的結果，故可以在 Truncate Adder 兩列輸入的第 37 位元各串入一個 AND 閘，就能有效防止進位的影響，如圖 3-9 所示。

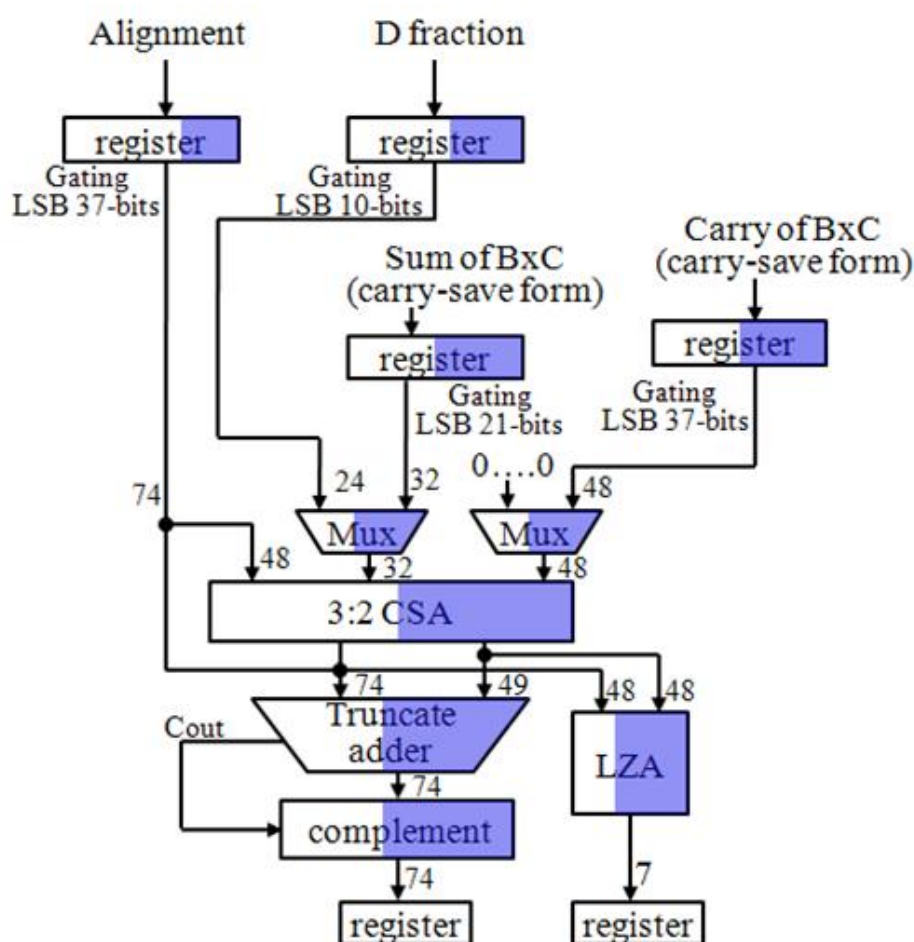


圖 3-8 截斷加法機制

本論文以 Ladner-Fischer Prefix Adder[15]架構來實現加法電路，它比傳統的漣波進位加法器(Ripple Carry Adder)快許多，圖 3-9 與圖 3-10 是一個 48 位的 Ladner-Fischer Parallel Prefix Adder。Prefix adder 中有兩種不同的元件，圖 3-10 即表示它們的功能。

假設有兩個數 $A = a_{n-1}a_{n-2}...a_0$ 和 $B = b_{n-1}b_{n-2}...b_0$ 要相加，prefix adder 的運作過程可以看成兩個階層。首先，在 prefix adder 的輸入 A 和 B 進入以後有一個方形的圖示(圖 3-10(a))，這裡面的元件分別用來產生 $g_i = a_i \wedge b_i$ 和 $p_i = a_i \vee b_i$ 的訊號，而 \wedge 與 \vee 分別為 AND 邏輯運算以及 OR 邏輯運算。經過第一層產生 g_i 和 p_i 之後，另外一層則利用這些訊號產生 c_i 。在圓形的元件二(圖 3-10(b))裡面主要是接收所有產生的 g_i 與 p_i 訊號，之後再利用兩個 AND 閘和一個 OR 閘產生 $g = g_i \vee (g_{i-1} \wedge p_i)$ 和 $p = p_i \wedge p_{i-1}$ ，使用這兩個訊號我們可以快速的產生 carry bit。

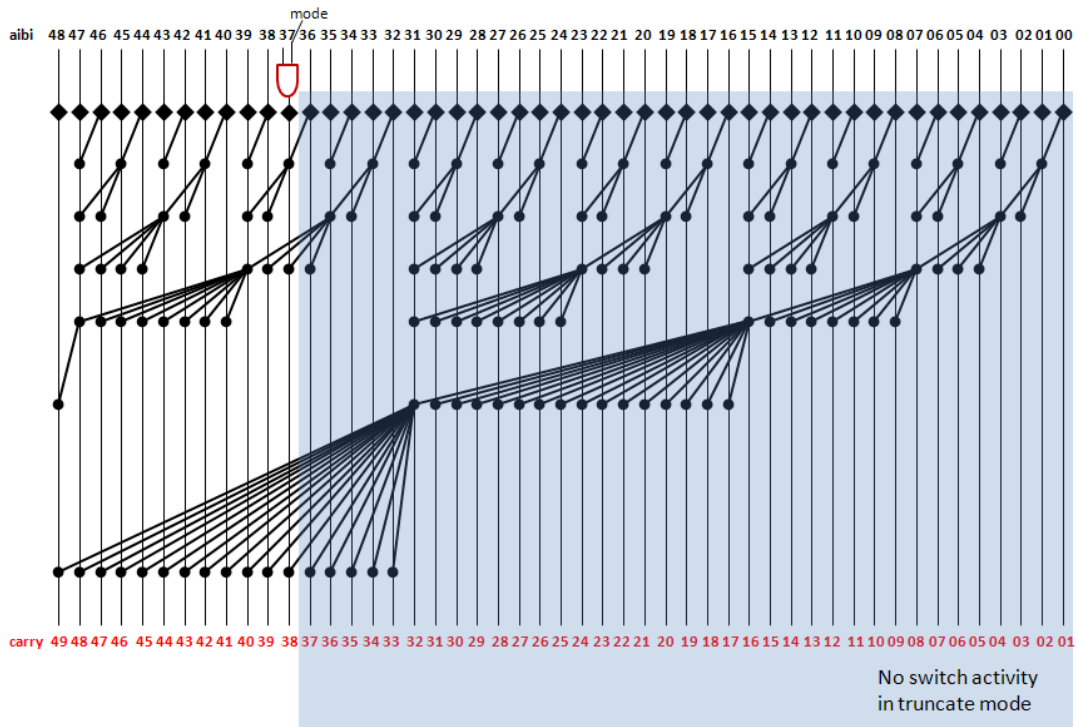
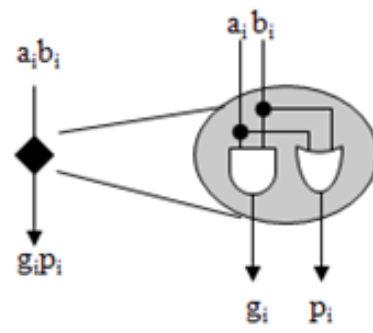
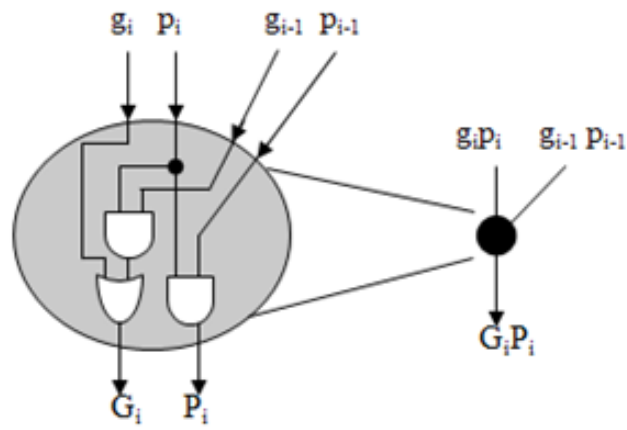


圖 3-9 Ladner-Fischer Parallel Prefix Adder 進位網路



(a)



(b)

圖 3-10 (a)Prefix adder 元件一 (b) Prefix adder 元件二

3.6 加法運算之誤差

加法運算所存在的誤差是因為 Truncate adder 的兩列輸入 a[37:0]和 b[37:0]不計算所引起的誤差，如圖 3-9 之陰影部分。在誤差模式下，仍然會產生上一個運算訊號之 a[37:0]與 b[37:0]之和，但 stage2 暫存器(圖 3-8 Complement 電路之後的暫存器)之 LSB [37:0]會被重置成零，故其誤差為仍然為兩列輸入 a[37:0]和 b[37:0]之最大值。其 a[37]之權重為 2^{-7} ，a[37:0]權重加總接近於 2^{-8} ，故兩列不計算之最大誤差百分比誤差為：

$$\text{FADD}_{\text{error}} = \frac{(2 \times 2^{-8})}{1} \times 100\% = 0.781\%$$

$$\text{FADD}_{\text{accuracy}} = \frac{1 - (2 \times 2^{-8})}{1} \times 100\% = 99.219\%$$

3.7 特殊浮點乘與浮點加運算

傳統浮點乘加器執行浮點乘時，是將浮點數 A 設定為 0，亦即 $Y=0+B \times C$ ；若執行浮點加則把浮點數 C 設定為 1，即 $Y=A+B \times 1$ 。此設定方式下的浮點乘加器之所有電路仍然有切換動作，而無關閉任何電路。一些特別的乘加器[28]、[31]為了在執行浮點乘或浮點加時，關閉不需要運算的電路，加入了額外的電路來達成。我們所提出的多重模式浮點乘加器，只需要在適當的位置增加一些邏輯電路和控制訊號，便可有效地減少電路的切換，以達到關閉電路的效果。

圖 3-1 中 Mux 電路將選擇浮點數 D 或浮點數 $B \times C$ 進行加法運算。單執行浮點加法時，Mux 選擇左邊，即為 $Y=A+D$ ，此時浮點數 B 和浮點數 C 不需要被送進 Booth Iterative Multiplier 進行運算，故可在浮點數 B 和浮點數 C 之輸入暫存器加入 clock gating 的機制，而 Booth Iterative Multiplier 之 Pipeline register RS 同樣可被 clock gating，此方式可有效地減少 Booth Iterative Multiplier 電路切換的功率消耗，圖 3-11 為執行單純浮點乘法時，所關閉之電路區域。

圖 3-1 執行單純浮點乘法時，只需要浮點數 B 和浮點數 C 的訊號，而浮點數 A 及浮點數 D 不需要被寫入暫存器，此時亦可用 clock gating 機制達成。由於假數 A 對齊電路輸出會影響 $B \times C$ 的結果，故可在 Ma 暫存器加上重置訊號(register 的 rst 訊號)，強制讓 Ma 暫存器輸出一整列“00.....00”的訊號，即可得到 $Y=B \times C$ 的正確答案，此方式能有效減少對齊電路的切換功率消耗，圖 3-12 為單執行浮點乘法時，所關閉電路之區域。

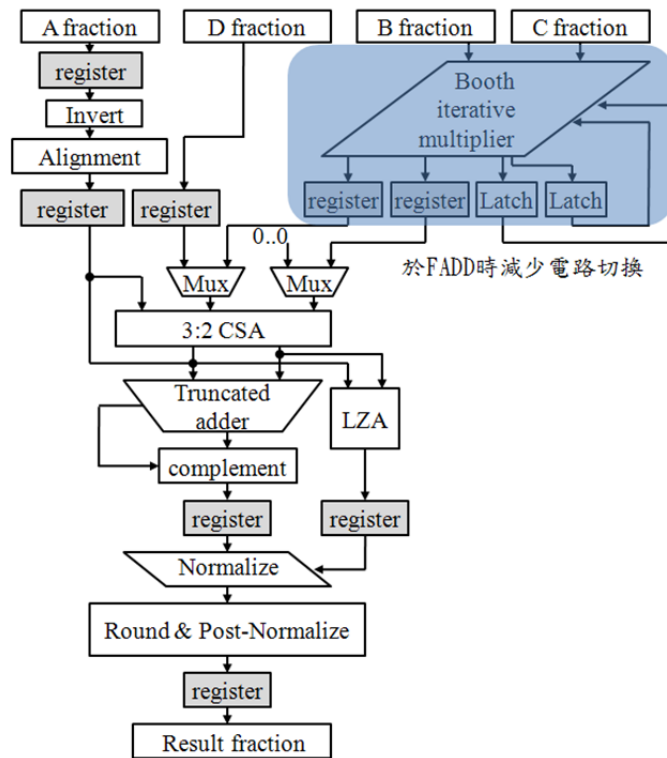


圖 3-11 執行浮點加法所關閉之電路區域

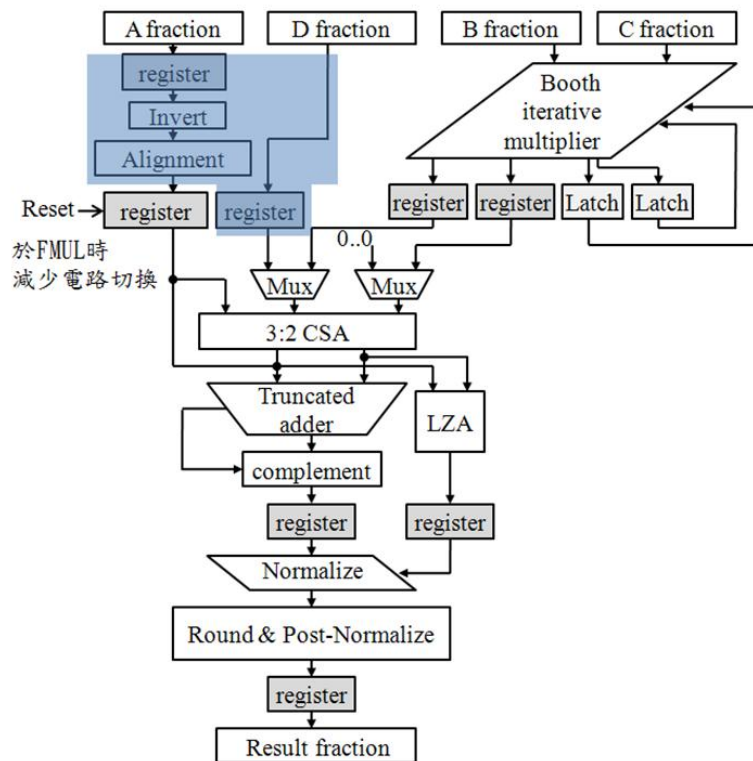


圖 3-12 執行浮點乘法所關閉之電路區域

3.8 多重模式浮點乘加器之誤差

3.4 節和 3.6 節分別說明了浮點乘和浮點加的誤差，而浮點乘加器是由兩個主要電路串接而成，故其誤差可由浮點乘精準度(FMUL_{accuracy})和浮點加精準度(FADD_{accuracy})計算出 H-MAF、M-MAF 和 L-MAF 三種模式的精準度和誤差百分比：

$$\text{H-MAF}_{\text{accuracy}} = 100\%$$

$$\text{H-MAF}_{\text{error}} = 0\%$$

$$\begin{aligned}\text{M-MAF}_{\text{accuracy}} &= A + ((B \times C) \times \text{FMUL}_{\text{accuracy}}) \\ &\approx (A + (B \times C)) \times \text{FMUL}_{\text{accuracy}} = 99.672\%\end{aligned}$$

$$\text{M-MAF}_{\text{error}} = 0.328\%$$

$$\begin{aligned}\text{L-MAF}_{\text{accuracy}} &= (A + ((B \times C) \times \text{FMUL}_{\text{accuracy}})) \times \text{FADD}_{\text{accuracy}} \\ &= (A \times \text{FADD}_{\text{accuracy}}) + \\ &\quad ((B \times C) \text{FMUL}_{\text{accuracy}} \times \text{FADD}_{\text{accuracy}}) \\ &\approx (A + (B \times C)) \times \\ &\quad \text{FMUL}_{\text{accuracy}} \times \text{FADD}_{\text{accuracy}} = 98.893\%\end{aligned}$$

$$\text{L-MAF}_{\text{error}} = 1.107\%$$

3.9 控制電路

在浮點乘加器的實作上，我們必須有一個控制器來控制 Datapath 電路的運作。我們設計的控制電路由三個元件所組成，分別為 Output Logic、Next State Logic 以及 State Logic，其架構由圖 3-13 表示。此控制電路輸出主要控制乘加器的 datapath 在不同模式運作下所需要的電路訊號，主要包括多功器的選擇訊號(Mux signal)、暫存器的寫入訊號(enable of register)、攔截 clock 的 gating 訊號等。

電路的開始執行透過一個 start 訊號來控制啟動運算，之後每一筆資料則是透過結束訊號 done 來決定是否要輸入下一筆資料以及乘加器電路已經運算完筆。Output logic 的輸入有兩個，一個由 state logic 用來控制 state 轉換的訊號，另一個是使用者輸入的 Mode 訊號，其用來控制要執行的各浮點運算的誤差模式，。Mode[3:0]訊號區別 7 種模式，浮點乘累加運算有三種模式，分別為 4'b1111、4'b1110、4'b1010，浮點乘法以及浮點加法分別為 4'b0011、4'b0010、4'b1100、4'b1000 四種模式。Mode[3]以及 mode[1]分別代表是否執行乘加器內浮點加與浮點乘，故在乘累加的模式中 Mode[3]與 Mode[1]都必定位 1。由於我們所提出的浮點乘加器必須符合每個時脈週期能夠切換模式執行，故必須記錄乘加器 datapath 內運算的狀態，此部分由 Next State Logic 和 State Logic 來判斷。

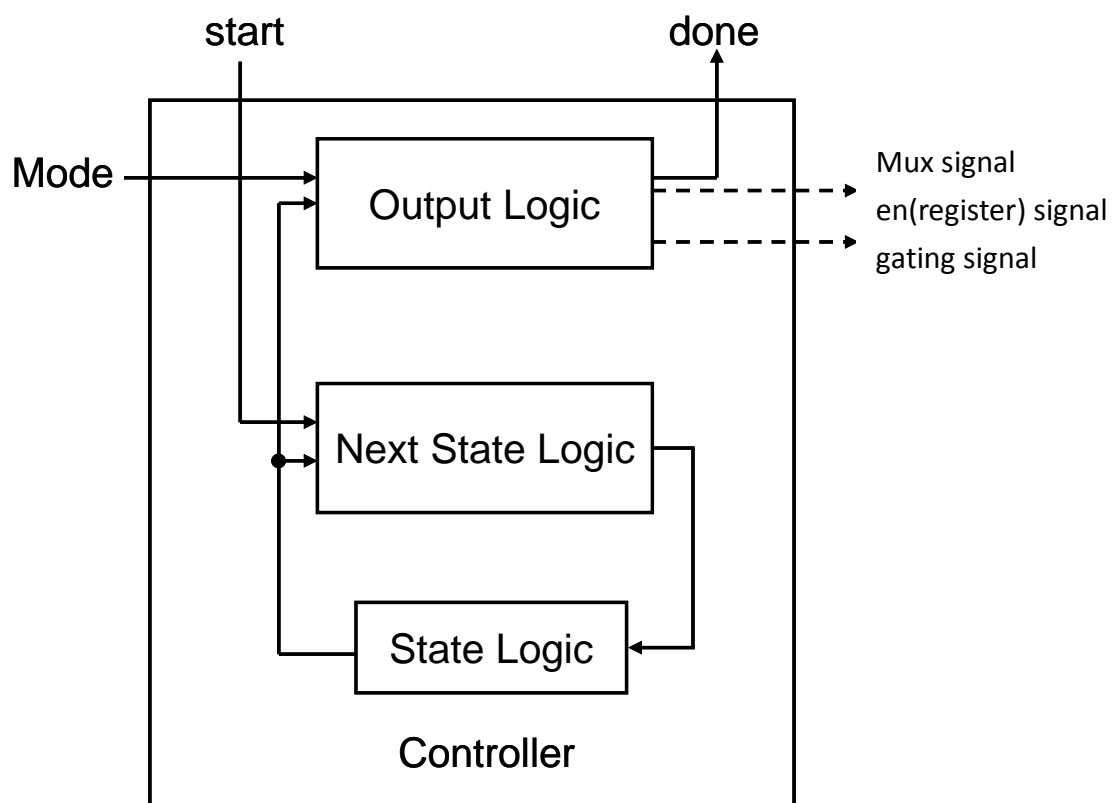


圖 3-13 控制器架構圖

Chapter 4. 實驗結果及比較

我們使用 Verilog HDL 硬體描述語言撰寫浮點乘家器的 RTL code，邏輯合成則採用 Synopsys 之 Design Compiler 合成軟體，並使用 TSMC 0.13 μ m CMOS 製程 Artisan 標準元件庫，將 RTL code 合成 Gate Level netlist。本論文提出的多重模式浮點乘加器主要的比較對象為[12]所提出的浮點乘法器，我們同樣對[12]浮點乘加器以相同的 Synopsys 合成軟體和元件庫加以合成，以確保電路能在相同的設計環境下作比較。表格 4-1 為電路合成後的數據，我們所提出的多重模式浮點乘加器之電路延遲比傳統浮點乘加器多 23%，電路面積比傳統浮點乘加器少 4.5%。這是因為我們所提出之乘加器中的重複式乘法電路必須以一半的工作時脈進行運算，導致電路延遲以及工作時脈較長。表格 4-2 到表格 4-4 為各浮點乘加器之功率消耗測試，其中是以各浮點乘加器之 Gate Level net-list 執行模擬，再使用 Prime Power 量測功率消耗。

表格 4-2 與表格 4-3 是執行浮點加法或浮點乘法運算時，所提出的浮點乘加器與傳統浮點乘加器之功率消耗比較，其執行頻率為 167Mhz，執行大約 20 萬個運算。由此可以看出我們所提出的浮點多重模式浮點乘加器的功率消耗比傳統浮點乘加器節省 5%到 55%之間。表格 4-2 與表格 4-3 為每個時脈週期執行相同的單一浮點運算(浮點乘法或浮點加法)，雖然傳統浮點乘法器沒針對浮點乘法與浮點加法作特別設計，但是其功率消耗仍然比執行浮點乘累加時小許多。在浮點乘法中，傳統浮點乘加器設定浮點數 A 之輸入為 0，而連續的浮點數 0，造成對齊電路無作移位之動作；另外，傳統浮點加法設定浮點 B(乘數)為浮點數 1，此時 Radix-4 Booth Multiplier 中之部分乘積，R0~R11 永遠被編成+0，只有 R12 會有編碼的不同，造成大部分的部分乘積都是相同，進而減少了電路切換之功率消耗。在實際應用程式執行中，大部分運算為每個時脈週期都切換不同的浮點運算，故實際應用程式運算之功率消耗會比表格 4-2 與表格 4-3 較為耗電。我們

所提出之多重模式浮點乘加器，其特殊設計與輸入浮點數無關，無論在連續相同的浮點運算或每時脈週期都切換不同的浮點運算，能夠關閉電路切換之功率消耗，故我們提出的浮點乘加器於表格 4-2 與表格 4-3 為實際執行應用程式之功率消耗。

另外，我們使用 RGB 轉 YUV 圖片格式轉換的多媒體應用程式當作輸入來測試乘加器的功率效能。當應用程式執行乘累加、乘、加的運算時，我們萃取出其中的輸入浮點數，作為浮點乘加器的運算值。在傳統浮點乘加器中，乘累加、乘、加運算都是以同一模式進行，而所提出之多重模式浮點乘加器則選擇不同的誤差模式下測量功率消耗，其執行頻率為 167Mhz，執行大約 20 萬個運算，測得的功率消耗如表格 4-4 所示。以傳統浮點乘加器為基準，我們所提出的架構在 H-MAF(無誤差 MAF)誤差模式下功率消耗比傳統浮點乘加器多出 9.8%，但是在 M-MAF和 L-MAF的誤差模式下功率消耗只佔了傳統浮點乘加器的 67%和 57%。表格 4-6 為多重模式浮點乘加器執行 RGB 轉 YUV 圖片格式轉換應用程式時，各誤差模式之運算對圖片轉換後的失真比較，由此可以看到儘管可看出儘管在 L-MAF 模式下，其 PSNR 值只下降了 17.3%，而在功率消耗上則節省 42.7%。

表格 4-1 傳統浮點乘法器與多重模式浮點乘法器之面積以及電路延遲比較

Design	Delay(ns)	Area(μm^2)	Delay(μs) \times Area(μm^2)
Conventional MAF[12]	3.10(100%)	79474(100%)	246.37(100%)
Proposed MAF	3.80(122.5%)	75918(95.5%)	288.49(117%)

表格 4-2 浮點乘加器執行浮點加運算之功率消耗比較

Design	mode	Power (Watt)
Conventional MAF[12]	MAF(FADD)	10.7e-03(100%)
Proposed MAF	H-FADD	7.27e-03(68%)
	L-FADD	4.82e-03(45%)




表格 4-3 浮點乘加器執行浮點乘運算之功率消耗比較

Design	mode	Power(Watt)
Conventional MAF[12]	MAF(FMUL)	10.8e-03(100%)
Proposed MAF	H-FMUL	10.3e-03(95%)
	L-FMUL	7.05e-03(65%)

表格 4-4 使用傳統浮點乘加器與多重模式浮點乘加器於應用程式(RGB 轉 YUV)

Design	mode	Power(Watt)
Conventional MAF[12]	MAF	13.3e-03 (100%)
Proposed MAF	H-MAF	14.6e-03(109.8%)
	M-MAF	8.94e-03(67.2%)
	L-MAF	7.62e-03(57.3%)

表格 4-5 多重模式浮點乘加器執行 RGB 轉 YUV 圖片格式轉換之圖片效果

Application of RGB to YUV format convert		
		
H-MAF (0% error) PSNR 45.63	M-MAF (0.328% error) PSNR 42.91	L-MAF (1.107% error) PSNR 37.74

表格 4-6 為各種浮點乘加器之電路延遲、電路面積和功率消耗之比較。對於其他論文或期刊內不同製程之據數，我們進行數據正規化後進行比較，比較結果顯示我們所提出的多重模式浮點乘加器，在電路面積和功率消耗上有明顯的改善。

表格 4-6 各浮點乘加器之比較

Design	Delay (ns)	Area (μm^2)	Gate count	Power (167mHz)	Architecture
Conventional MAF[12] (3-stage)	3.1 (100%)	79.4k (100%)	404k (100%)	13.3mW (100%)	Single precision
Proposed MAF (3-stage)	3.8 (122.5%)	75.9k (95.6%)	377k (93%)	14.6mW(109.8%) 8.94mW(67.2%) 7.62mW(57.3%)	Single precision (multi-mode)
Concordia MAF[27] (3-stage)	5.26 (169%)	--	1205k (290%)	10.94mW(82%)	Single precision (low power design)
Huang MAF[29] (1 stage)	7.20	88.5k (114.6%)		--	SIMD (1 single or 2 half precision)
Three-Path MAF[28] (1 stage)	4.32	259k (326.2%)		--	Single precision (with FMUL)
Qi MAF[30] (1 stage)	8.24	131.5k (165.6%)		--	SIMD (1single or 2 half precision)
Bridge MAF[31] (1 stage)	5.82	283.65k (357.2%)		--	Single precision (parallel FADD/FMUL)

Chapter 5. 結論與未來研究方向

5.1 結論

本論文所提出的多重模式浮點乘加器的功率消耗表現方面，在無誤差模式下比傳統浮點乘加器多 9.8% 功率消耗，但是在具有誤差的模式下則可節省至少 52% 的功率消耗。在圖片格式轉換 RGB 轉 YUV 的應用程式中，使用多重模式浮點乘加器最大誤差模式(運算誤差 1.107%)時，仍能保持圖片格式轉換後之 PSNR 值只下降 17.3%。

傳統浮點乘加器在執行單純的浮點加運算和浮點乘運算時，並沒有設計特別的關閉電路來減少成架器的電路切換，而本論文所提出的浮點乘加器，只增加一些電路面積，就能有效地降低電路的功率消耗(如表格 4-2 和表格 4-3)。

5.2 未來研究方向

本論文所提出的多重模式乘加器，其誤差評估方式是以部分乘積圖中不計算得每個位元皆以“1”之最大誤差來估計。而在實際的運算中，由於布斯編碼的關係，不可能出現全為“1”的狀況，故實際誤差比最大誤差小許多。我們希望找出較為實際的誤差計算方式，如此將可提高每個模式的精準度，進而提高 L-MAF 模式在實際執行應用程式時被用到的機率。另一方面，我們所提出的浮點乘加器在浮點乘加運算中只有三種模式，用實際的應用中可能因為沒有適當的模式可供選擇而損失節省功率消耗的機會，未來我們希望在不增加電路面積和電路延遲的條件下，發展出具有更多模式的浮點乘加器供使用者選用。

參考文獻

- [1] Y. Voronenko and M. Pushel, "Automatic Generation of Implementations for DSP Transforms on Fused Multiply-Add Architectures," *International Conf. on Acoustics, Speech and Signal Processing*, pp. V-101-V-104, 2004.
- [2] E. N. Linzer, "Implementation of Efficient FFT Algorithms on Fused Multiply-Add Architectures," *IEEE Transactions on Signal Processing*, Vol. 41, pp. 93-107, 1993.
- [3] H. H. Yao, E. E. Swartzlander, Jr., "Serial-parallel multiplier," *IEEE conf. on Signals, Systems and Computers*, vol.1, pp. 359-363, 1993.
- [4] H. I. Saleh, A. H. Khalil, M. A. Ashour, A. E. Salama, "Novel serial-parallel multiplier," *IEEE Journal on Circuits, Devices and Systems*, Vol. 148, pp. 183-189, 2001.
- [5] Lan-Da Van, Shuenn-Shyang Wang, Tenqchen Shing, Wu-Shiung Feng, and Bor-Shenn Jeng, "Design of a lower-error fixed-width multiplier for speech processing application," *IEEE Conf. on Circuits and Systems*, Vol.3, pp. 130-133, 1999.
- [6] Booth, A. -D., "A Signed Binary Multiplication Technique," *Quarterly J. Mechanical Application in Math.*, vol. 4, part2, pp. 236-240, 1951.
- [7] C S. Wallace, "A suggestion for a fast multiplier," *IEEE Transactions On Electronic Computers*, Vol. 13, pp. 14-17, 1964.
- [8] L. Dadda, "Some Schemes for Parallel Multipliers," *Alta Frequenza*, Vol. 34, pp. 349-356, 1965.
- [9] B. S. Cherkauer, E. G. Friedman, "A hybrid radix-4/radix-8 low power signed multiplier design," *IEEE Transactions on Computers*, Vol. 54, no. 3, March 2005.
- [10] 李朝民, "左到右陣列乘法器之分析與比較", 南台科技大學電子工程研究所碩士論文, 民 95 年.
- [11] ANSI/IEEE standard 754-1985, *IEEE Standard for Binary Floating-Point Arithmetic*, 1985.
- [12] E. Hokenek, R. Montoye, and P. W. Cook, "Second-Generation RISC Floating Point with Multiply-Add Fused," *IEEE Journal of Solid-State Circuits*, Vol. 25, pp. 1207-1213, Oct. 1990.
- [13] Martin S. Schmookler, Kevin J. Nowka, "Leading Zero Anticipation and

- Detection A Comparison of Methods,” *ARITH-15*, pp. 7, June 11-13, 2001.
- [14] P. BONATTO, V. G. OKLOBDZIJA, “Evaluation of Booth's Algorithm for Implementation In Parallel Multipliers,” *IEEE ASILOMAR-29*, pp. 608-610, 1996.
 - [15] R. E. Ladner and M. J. Fischer, “Parallel prefix computation,” *J. Ass. Comput. Mach.*, Vol. 27, pp. 831-838, Oct. 1980.
 - [16] 郭倉源, “適用於多媒體應用的低功率多重精確度重複式浮點乘法器”, 國立中山大學資訊工程學系碩士論文, 民 99 年.
 - [17] R. K. Montoye, E. Hokenek and S. L. Runyon, “Design of the IBM RISC System/6000 floating-point execution unit,” *IBM Journal of Research & Development*, Vol. 34, pp. 59-70, 1990.
 - [18] A. Kumar, “The HP PA-8000 RISC CPU,” *IEEE Micro Magazine*, Vol. 17, Issue 2, pp. 27-32, April, 1997.
 - [19] D. Hunt, “Advanced Performance Features of the 64-bit PA-8000,” *Proceedings of Compcon*, pp. 123-128, 1995.
 - [20] K. C. Yeager, “The MIPS R10000 superscalar microprocessor,” *IEEE Micro Magazine*, Vol. 16, no. 2, pp. 28-40, March, 1996.
 - [21] B. Greer, J. Harrison, G. Henry, W. Li and P. Tang, “Scientific Computing on the Itanium Processor,” *ACM/IEEE Conf. on Supercomputing*, pp. 1-8, 2001.
 - [22] H. Sharangpani and K. Arora. “Itanium Processor Microarchitecture,” *IEEE Micro Magazine*, Vol. 20, no. 5, pp. 24-43, Sept-Oct, 2000.
 - [23] Zhijun Huang, “High-Level Optimization Techniques for Low-Power Multiplier Design,” *PhD dissertation, Univ. of California, Los Angeles*, June, 2003.
 - [24] D. Radhakrishman and A. P. Preethy, “Low-power CMOS pass logic 4-2 compressor for high-speed multiplication,” *43rd IEEE Midwest Symp. on Circuits & Systems*, Vol. 3, pp. 1296-1298, 2000.
 - [25] P. Mokrian, G. Howard, G. Jullien, and M. ahmadi, “On the use of 4:2 compressor for partial product reduction,” *IEEE Canadian Conf. on Electrical and Computer Engineering*, Vol. 1, pp. 121-124, May, 2003.
 - [26] D. Villeger and V. Oklobdzija, “Analysis of Booth Encoding Efficiency in Parallel Multipliers Using Compressors for Reduction of Partial Products,” *27th Ann. Asilomar Conf. on Signals, Systems, and Computers*, Vol. 1, pp. 781-784, 1993.

- [27] R.V.K. Pillai, S.Y.A. Shah, A.J. Al-Khalili, and D. Al-Khalili, “Low Power Floating-Point MAFs – A Comparative Study”, *sixth International Symp. on Signal Processing and its Applications*, Vol. 1, pp. 284-287, August, 2001.
- [28] Eric Quinnell, Earl E. Swartzlander, Jr., Carl Lemonds, “Floating-Point Fused Multiply-Add Architectural”, *Forty-First Asilomar Conf. on Signals, Systems and Computers*, 2007.
- [29] Libo Huang, Li Shen, Kui Dai, Zhiying Wang, “A New Architecture For Multiple-Precision Floating-Point Multiply-Add Fused Unit Design”, *IEEE symp. on Computer Arithmetic*, June 2007.
- [30] Zichu Qi, Qi Guo, Ge Zhang, Xiangku Li, Weiwu Hu, “Design of Low-Cost High-performance Floating-point Fused Multiply-Add with Reduced Power”, *23rd International Conference on VLSI Design*, Jan. 2010.
- [31] Eric Quinnell, Earl E. Swartzlander Jr., Carl Lemonds, “Bridge Floating-Point Fused Multiply-Add Design”, *IEEE Transactions on VLSI Systems*, Dec. 2008.
- [32] “TSMC 0.13 μ m (CL013G) process 1.2-Volt SAGE-XTM Standard Cell Library Databook,” Jan. 2004.