

國立交通大學  
電機與控制工程研究所  
碩士論文

針對長係數有限脈衝響應濾波器硬體實現  
且符合成本效益的摺疊技巧

A Cost-Efficient Folding Technique for Long-Length  
FIR Filter Implementation

研究生：楊明峰

指導教授：董蘭榮 博士

中華民國九十四年七月

針對長係數有限脈衝響應濾波器硬體實現

且符合成本效益的摺疊技巧

A Cost-Efficient Folding Technique for Long-Length

FIR Filter Implementation

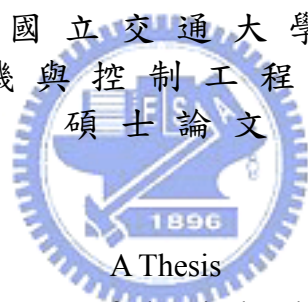
研究生：楊明峰

Student : Ming-Feng Yang

指導教授：董蘭榮

Advisor : Lan-Rong Dung

國立交通大學  
電機與控制工程學系  
碩士論文



A Thesis

Submitted to Department of Electrical and Control Engineering

College of Electrical Engineering and Computer Science

National Chiao Tung University

in partial Fulfillment of the Requirements

for the Degree of

Master

in

Electrical and Control Engineering

July 2005

Hsinchu, Taiwan, Republic of China

中華民國九十四年七月

# 針對長係數有限脈衝響應濾波器硬體實現 且符合成本效益的摺疊技巧

學生：楊明峰

指導教授：董蘭榮 博士

國立交通大學電機與控制工程研究所

## 摘 要

本篇論文針對有限資源實現長係數有限脈衝響應濾波器提出了兩種有效率摺疊硬體的演算法，其所採用的摺疊演算法是根據處理速率(Throughput)條件來做評估。在架構中使用少量的運算器，規劃其運作時序，就可分時完成工作，如此一來可以減少硬體中乘加運算器的數目，縮小硬體的面積。摺疊演算法非常適合於處理速率不需要很高的陣列硬體架構，尤其是對於陣列的運算單元個數會因規格的不同而有數目上變化的應用。利用推導成本函數來比較各種摺疊演算法在硬體實現上面積與功率消耗優劣，發現 Parallel-In 摺疊演算法在兩方面效能表現皆是最佳，並透過 WCDMA 規格實際實現各種摺疊演算法得到證明；最後，使用 UMC 0.18  $\mu m$  1P6M CMOS 製程完成晶片下線。

# A Cost-Efficient Folding Technique for Long-Length FIR Filter Implementation

Student : Ming-Feng Yang    Advisor : Dr. Lan-Rong Dung

Department of Electrical and Control Engineering  
National Chiao Tung University

## ABSTRACT

This thesis presents two hardware efficient folding techniques for limited-resource implementation of long-length FIR filtering. Under the requirement of the throughput rate, we fold the FIR with the minimal number of processing elements (PEs) while the complexity of scheduler is low. The proposed folded architecture is highly scalable as the application parameters change. Cost functions are derived and these are used to address two related issues. The first issue focuses on memory requirements in folded architectures. The second is power consumption. The result of memory requirements and power estimation show that Parallel-In folding technique can turn out less memory area and power dissipation than do other folding techniques. Finally, the chip is implemented by using the UMC 0.18  $\mu m$  1P6M CMOS technology.

## 目錄

中文摘要.....	I
英文摘要.....	II
目錄.....	III
圖目錄.....	V
表目錄.....	VII
<b>第一章 緒論</b> .....	1
1.1 研究動機.....	1
1.2 論文摘要.....	4
<b>第二章 研究背景</b> .....	6
2.1 摺疊技術.....	6
2.2 時序規劃.....	7
2.2.1 單一節點時序規劃.....	9
2.2.2 切集合時序規劃.....	10
2.3 第一篇系統化摺疊演算法[9].....	11
2.4 摺疊位元平面有限脈衝響應濾波器[6].....	13
2.4.1 位元平面濾波器架構.....	14
2.4.2 位元平面濾波器轉換公式推導.....	15
2.4.3 實現位元平面摺疊架構.....	16
<b>第三章 新摺疊演算法之推演與建立</b> .....	18
3.1 新摺疊演算法的硬體架構推導.....	19
3.2 以摺疊演算法實現硬體之方法與特色.....	25
3.3 新摺疊演算法一( Parallel-In ).....	26
3.3.1 離型轉換.....	26
3.3.2 排程矩陣.....	29
3.4 新摺疊演算法二( Serial-In ).....	34
<b>第四章 硬體實現架構</b> .....	36
4.1 新摺疊演算法一硬體實現.....	37
4.1.1 處理器陣列.....	38
4.1.2 暫存器檔案.....	39
4.1.3 控制單元.....	40
4.2 新摺疊演算法二硬體實現.....	41
4.2.1 處理器陣列.....	43
4.2.2 暫存器檔案.....	43

4.2.3	控制單元.....	45
4.3	硬體實現成本函數.....	45
4.3.1	面積成本函數.....	46
4.3.2	功率消耗成本函數.....	50
4.4	摺疊演算法實現 WCDMA 規格並比較.....	53
4.4.1	原始架構及各種摺疊架構實現.....	53
4.4.2	軟體模擬面積大小及功率消耗情形.....	56
4.4.3	結果分析.....	58
<b>第五章</b>	<b>設計流程與模擬結果</b> .....	<b>60</b>
5.1	設計流程.....	60
5.2	功能驗證.....	62
5.3	電路合成.....	65
5.3.1	測試電路.....	65
5.3.2	邏輯閘合成.....	67
5.4	晶片規格與佈局設計.....	68
5.5	DRC&LVS 驗證與佈局後模擬.....	70
<b>第六章</b>	<b>結論</b> .....	<b>72</b>
6.1	主要貢獻.....	72
6.2	未來展望.....	73
		
	<b>參考文獻</b> .....	<b>75</b>

## 圖目錄

圖 2.1	時序規劃範例.....	8
圖 2.2	單一節點時序規劃.....	9
圖 2.3	切集合時序規劃範例.....	10
圖 2.4	乘加器實現五階有限脈衝響應濾波器.....	11
圖 2.5	圖 2.4 的摺疊架構.....	12
圖 2.6	位元平面架構資料流程圖( $K=3, b=4$ ).....	14
圖 2.7	轉換資料流程圖使能應用摺疊技術.....	16
圖 2.8	位元平面濾波器摺疊架構圖.....	16
圖 2.9	位元平面摺疊架構.....	17
圖 3.1	基本運算單元轉換乘加點.....	20
圖 3.2	運算單元陣列.....	21
圖 3.3	運算單元陣列時序規劃.....	22
圖 3.4	運算單元陣列摺疊架構.....	23
圖 3.5	摺疊架構之乘加器與暫存器規劃.....	23
圖 3.6	摺疊架構暫存器定址規劃.....	24
圖 3.7	$K-1$ 階有限脈衝響應濾波器架構.....	26
圖 3.8	雛型轉換步驟—Scaling.....	27
圖 3.9	雛型轉換步驟—Retiming.....	28
圖 3.10	雛型轉換步驟—Replacing.....	28
圖 3.11	MAC 操作時序.....	31
圖 3.12	Serial-In 演算法( $K=12, r=3$ ).....	34
圖 3.13	圖 3.12 摺疊架構實現.....	35
圖 4.1	新摺疊演算法實現架構概念圖.....	37
圖 4.2	Parallel-In 處理器陣列.....	38
圖 4.3	Parallel-In 暫存器檔案.....	39
圖 4.4	Parallel-In 控制單元.....	40
圖 4.5	Serial-In 演算法.....	41
圖 4.6	Serial-In 摺疊架構實現.....	42
圖 4.7	Serial-In 的 MAC 運作內容時序.....	42
圖 4.8	Serial-In 處理器陣列.....	43

圖 4.9	Serial-In 暫存器檔案.....	44
圖 4.10	Serial-In 控制單元.....	45
圖 4.11	面積成本函數.....	49
圖 4.12	功率消耗成本函數.....	52
圖 4.13	原始架構.....	53
圖 4.14	[9]實現規格架構.....	54
圖 4.15	[6]實現規格架構.....	55
圖 4.16	Parallel-In 實現規格架構.....	55
圖 4.17	Serial-In 實現規格架構.....	56
圖 5.1	設計流程圖.....	61
圖 5.2	量化係數及其頻率響應圖.....	63
圖 5.3	Impulse Response 驗證結果圖.....	64
圖 5.4	$\sin(w_1t) + \sin(w_2t)$ 驗證結果圖.....	64
圖 5.5	正反器置換為可掃描正反器.....	65
圖 5.6	Design Compiler 合成電路圖.....	68
圖 5.7	晶片佈局圖.....	69
圖 5.8	DRC 驗證.....	70
圖 5.9	LVS 驗證.....	71
圖 5.10	佈局後驗證.....	71



## 表目錄

表 2.1	圖 2.5 時序操作表.....	13
表 3.1	Data Matrix.....	30
表 3.2	Coefficient Matrix.....	30
表 3.3	Feedback Matrix.....	30
表 3.4	Accumulator Matrix.....	30
表 3.5	暫存器內容.....	32
表 3.6	Data Matrix.....	33
表 3.7	Coefficient Matrix.....	33
表 3.8	Feedback Matrix.....	33
表 3.9	Accumulator Matrix.....	33
表 4.1	成本函數符號意義.....	46
表 4.2	WCDMA 規格.....	53
表 4.3	Design Compiler 模擬面積大小.....	57
表 4.4	PrimePower 模擬功率消耗.....	57
表 5.1	符合規格量化係數.....	62
表 5.2	測試涵蓋率評估.....	67
表 5.3	各單元面積統計.....	68
表 5.4	晶片規格列表.....	69
表 5.5	PAD 使用列表.....	69

---

---

# 第一章

---

---

## 緒論

---

---

### 1.1 研究動機




隨著科技的日新月異與物質生活水準的提昇，科技生活的來臨，更是拉近了人與人之間的距離，除了追求生活上的便利，更重視聽覺與視覺等感官上的享受，這些全都依賴具有高效能、高精確度、高可靠度與可晶片化等多項優點之數位訊號處理器的輔助，舉凡 3D 動畫、行動電話、虛擬實境、語音辨識、DVD 等語音、音效、影像的應用，都可以利用數位訊號處理器來實現。

今日的寬頻無線通訊與多媒體應用更是如此地蓬勃發展，消費性電子產品更如雨後春筍般出現在每個人的生活中，也因此造成數位訊號處理器被廣泛的採用。隨著製程技術的進步，特殊應用積體電路(ASIC)具有體積小與低功率消耗等特性，正符合可攜式元件的市場需求，而由於電池材質無法突破，所以數位訊號處理器(DSP Processor)的設計除了高速外，為了能延長產品的使用時間，低功率消耗也成為目前數位訊號處理器的發展研究重點。

而在數位訊號處理系統中，有限脈衝響應濾波器( FIR filter )是最基本的函數功能及元件，例如在通訊與多媒體系統中，有限脈衝響應濾波器可以保證其系統的穩定性且實現的架構不需要太複雜，所以有限脈衝響應濾波器成為移除訊號中不需要部份的熱門技術。在品質要求很高的( Quality-Sensitive )應用中，有限脈衝響應濾波器的 taps 數通常要很大，taps 的範圍從數十到數百之間。然而，越高階的有限脈衝響應濾波器將會導致越大的硬體面積，同時，也有著嚴重的功率消耗問題。

數十年來，建議使用許多方法來減少有限脈衝響應濾波器的硬體複雜度 [1]-[12]，可以歸類成三大方向：乘法器的減少( Multiplier Reduction )、去乘法器化的實現( Multiplierless Realization )、資源重複使用( Resource Sharing )。相關減少有限脈衝響應濾波器硬體複雜度，列出幾個常見的做法如下：

1. 線性相位有限脈衝響應濾波器[3][8]推薦來減少乘法器個數，利用線性相位有限脈衝響應濾波器係數的對稱特性，可以減少二分之一的乘法器個數，大約可減少一半的硬體面積；
2. 區塊化( Block )有限脈衝響應濾波器設計[4][11]使用[13]中的 Overlap-Add 方法來減少區塊化有限脈衝響應濾波器中乘法的複雜度，但必須做一些犧牲，就是需要增加一些額外的加法器；
3. 使用分散式運算( Distributed Arithmetic, DA)技術來完成有限脈衝響應濾波器的乘累加動作，由於分散式運算是位元串流的去乘法器化( Bit-Serial Multiplierless )的計算操作，主要是用在計算向量內積上，而分散式運算是利用查表法來縮短乘法動作所耗費時間，藉著儲存全部可能的中間計算值在查表( Look-Up-Table )記憶體中，而藉由乘數的位元值來選擇正確的中間計算值，然後做累加移位的動作來完成向量內積的動作[1][5][12]；然而，減少計算複雜度可能會導致設計本身的效能( Performance )變差，使得成本最佳化( Cost-Optimization )成為另一重要議題。

當應用所需的資料處理速度( Throughput )小於實際電路本身可操作的速度時，這時候可以使用上一種技術—摺疊( Folding )技術也是一種有效率資源共用( Resource Sharing )的技術，此技術允許以犧牲電路可操作的最大速度( Throughput )來交換減少硬體實現的面積( Silicon Area )，但同時也減少了功率的消耗，原因是假設完成一件工作的總能量不變，但執行時間延長，所以，功率消耗減少了。

K. K. Parhi 在[2][9]中推導了第一個對任意數位訊號處理演算法的對稱摺疊轉換技術，此技術可用來系統化地決定控制電路及摺疊架構。這時候發展了一管線操作技術( Pipelining )理論來確保重複( Iteration )可獨立於控制電路中的暫存器，同時並藉著此理論，發展出一種摺疊的位元平面( Folded Bit-Plane )有限脈衝響應濾波器架構[6]，他們發展的摺疊架構能夠保證可改變的摺疊因數在位元層次(Bit-Level)的可合成性。當摺疊技術使用在時間上時，可以用來減少電路的邏輯閘數( Gate Count )，亦可減少總漏電流( Leakage )的產生，當漏電流能夠大量地避免重複使用，這對降低總功率消耗是一很重要的技術，這也是未來 CMOS 技術的重要特徵之一。

為了完成上述的想法，在本篇論文中提出一個新的摺疊有限脈衝響應濾波器硬體架構，不僅可以達到程式化的目的，並可以大幅降低功率上不必要的損失。主要是根據給定的處理速率( Throughput )條件的評估，找到最佳的摺疊硬體架構，減少硬體面積及功率消耗。

本篇論文所設計的新摺疊有限脈衝響應濾波器硬體架構特色如下：

- 新摺疊硬體架構主要分為三部分：
  - 摺疊架構中處理乘累加動作( Processing Element )
  - 規劃地管理暫存器位址( Register Bank )
  - 管理暫存器的儲存及讀取( Control Unit )

其主要貢獻在於找到一套可系統化的暫存器定址方法，並對摺疊過程中所需的暫存器個數做到最佳化管理。

- 在已給定規格條件下，當確定規格的速度(Throughput)小於實際電路可操作的速度時，可以找到摺疊因數(Folding Factor)及其相對應的排程矩陣(Scheduling Matrix)來達到給定的規格，並兼有減少硬體面積及功率消耗的優點。

## 1.2 論文摘要

在此小節中先對本篇整個論文架構作個概略性的介紹。

### 第一章 緒論

提出論文主題、想要解決的問題及其主要應用所在。

### 第二章 研究背景

介紹摺疊技術出現背景，及其基本運作原理—時序規劃(Retiming)，並以簡單的例子講解時序規劃的方法。接著，介紹現今關於有限脈衝響應濾波器的摺疊演算法[6][9]。

### 第三章 摺疊演算法之推演與建立

提出一套摺疊演算法之理論，經由排程矩陣(Scheduling Matrix)上的推導，讓人擁有明確的流程可以遵循，並簡單地交代採用新摺疊演算法後的架構。

### 第四章 硬體實現

先對整個硬體架構運作與操作流程作介紹，之後再對摺疊架構中的個別方塊加以詳細解說，並推導其摺疊之硬體架構。最後，推導各種不同的摺疊演算法硬體實現的成本函數，成本函數包括了面積的推導，及功率消耗的推導，再透過WCDMA規格實現各種摺疊演算法來印證新摺疊演算法 Parallel-In 確實為低成本及低功率架構。

## 第五章 設計流程與模擬結果

主要描述電路設計流程與模擬驗證之結果，所提出的電路都使用VHDL硬體描述語言來實現，再使用SYNOPTSYS公司發展的Design Compiler軟體進行邏輯合成；然後採用Cadence公司的SOC Encounter來做下線晶片的佈局與繞線設計(Place&Route)並配合UMC 0.18  $\mu m$  1P6M CMOS製程技術來自動合成出晶片電路；最後使用Mentor Graphics公司的Calibre軟體做DRC&LVS驗證。

## 第六章 結論

本篇論文之結語與未來展望。



---

---

## 第二章


---

---

### 研究背景

---

---



在深入探討本篇論文研究之前，本章先來介紹一些摺疊技術的相關資訊及其技術重要性。接著，介紹應用摺疊技術的理論基礎—時序規劃(Retiming)，對時序規劃的方法有深入了解後，接著就可以很容易用來推導各種摺疊演算法的推演。最後，特別舉兩個摺疊演算法來介紹：一是[9]第一篇提出系統化的摺疊演算法，二是[6]位元平面的摺疊演算法，而選擇這兩篇的原因是因其摺疊演算法與我們提出的新摺疊演算法較為相似，特別提出一起討論，在第四章末節將會再次提出並依起比較彼此的成本函數。

#### 2.1 摺疊技術

現今，數位訊號處理廣泛地使用在即時(Real-Time)應用上，並且在數位革命上亦佔有非常重要的地位；然而，在這當中有限脈衝響應(FIR)濾波器是數位訊號處理中最基本的構成要素，因它有著穩定性良好且容易實現的優點，但是，在品質要求很高的(Quality-Sensitive)應用中，有限脈衝響應濾波器的 taps 數目



通常需要很多，其 taps 的範圍從數十到數百之間，然而，過多的 taps 將會導致硬體面積過度地增加。

而當應用規格所需資料處理速度( Throughput )小於實際電路可以操作的速度時，[9]-[11]推薦了許多摺疊演算法來以犧牲資料處理速度的方式，進而來達到節省硬體面積的目的。然而，有限脈衝響應濾波器本身是非常適合用來探討研究並實現摺疊演算法的實際應用，因其只是一直重複做乘累加的動作。

在同步設計中一個很重要的議題是如何去避免時脈歪曲( Clock- Skew )的問題，而在大部分的非同步架構[14][15]針對此問題的解決方法是再多做一塊額外的硬體來避免時脈歪曲的問題；然而，摺疊的有限脈衝響應濾波器架構的其中一項重要優點便是：它們一方面可以減少硬體面積(與不做摺疊的架構相比)，而同時另一方面又不會有時脈歪曲的問題，因此，可以不用多做一塊額外的非同步硬體電路來避免時脈歪曲的問題。

摺疊技術允許執行在資料流上幾乎相同的操作，可以在同一電路上利用時間上的多路傳輸( Time-Multiplexed )技巧來進行切換操作；然而，為了可以完成摺疊演算法的動作，摺疊架構內部時脈頻率就必須是未摺疊前的時脈頻率的倍數，所以，摺疊演算法是一種以提高處理速度來換取硬體面積的方法，因此，犧牲功率來換取面積；然而，在應用規格可允許的範圍內，讓摺疊演算法架構的面積盡量逼近最佳，且不希望犧牲太多功率的情況下，本篇論文在以此為目標的前提下，在下一章中提出新摺疊演算法。

## 2.2 時序規劃

時序規劃( Retiming )是一種在不改變演算法輸入輸出之時序關係的狀況下，改變演算法中記憶元件位置的一種技巧。時序規劃的主要應用範圍在於將演算法利用同步電路來實現時，利用移動演算法中記憶元件位置的方法，達到縮小



電路時脈週期，減少暫存器數量，以及降低功率消耗的目的。在本論文中，我們利用其可以移動演算法中記憶元件位置的特性，來完成我們的新摺疊演算法架構，以圖 2.1 來說明一個簡單的時序規劃例子。

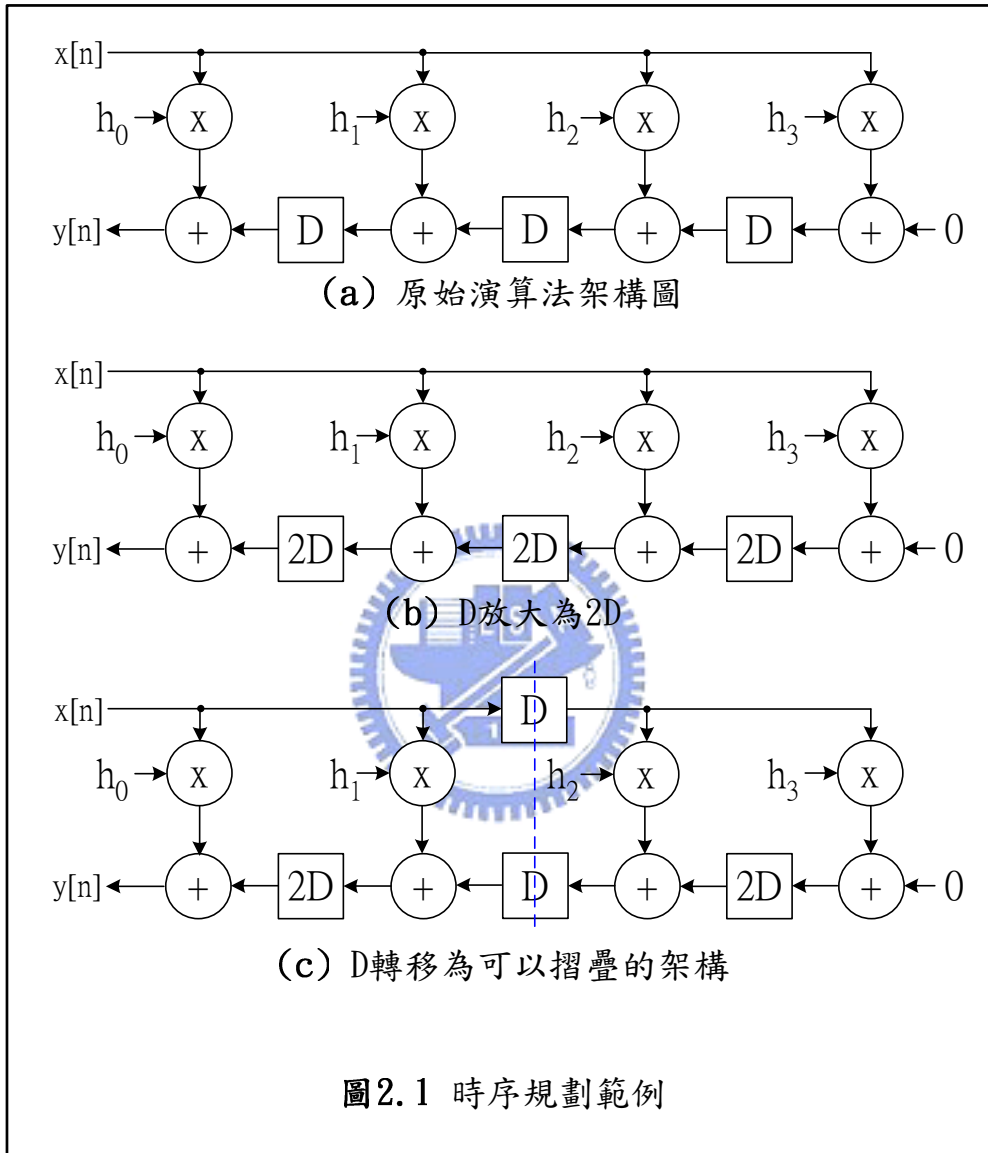


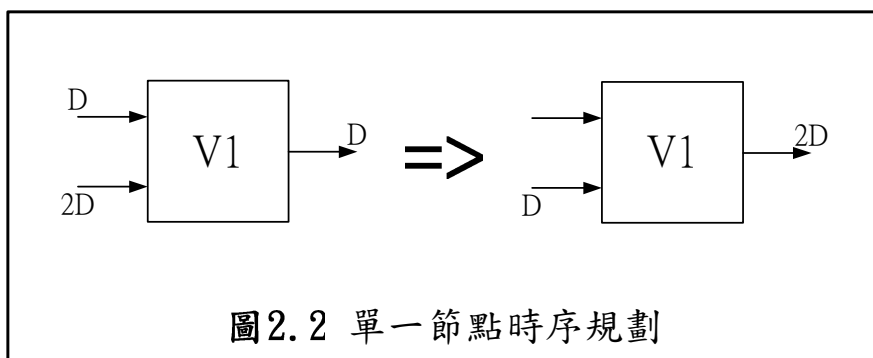
圖 2.1 為時序規劃的簡單範例，(a)為有限脈衝響應濾波器演算法原始資料流程圖 (DFG)；(b)顯示可以將資料流程圖中全部的 D 一起放大 (Scaling) 成任意倍數，其顯示為放大兩倍，推論其輸入輸出的時序關係仍維持一致；(c)移動係數  $h_1$  與  $h_2$  之間下方的  $2D$ ，並將其移動  $1D$  到係數  $h_1$  與  $h_2$  之間上方，而係數  $h_1$  與  $h_2$  之間下方只遺留下  $1D$ ，此時，便能執行摺疊演算法從  $h_1$  與  $h_2$  中間切斷成兩個部份，而其切斷的 D 便是用來實現切斷兩部分的資料傳遞橋樑，如此在實

現摺疊摺疊演算法時，便能確保不會讓傳遞的資料遺失。

實現任何摺疊演算法都是比照此模式進行資料流程圖( DFG )的轉換，因此，許多時序規劃技巧對於摺疊演算法就顯得相當重要[19][20]，以下章節就幾種常見的時序規劃技巧做說明。

### 2.2.1 單一節點時序規劃

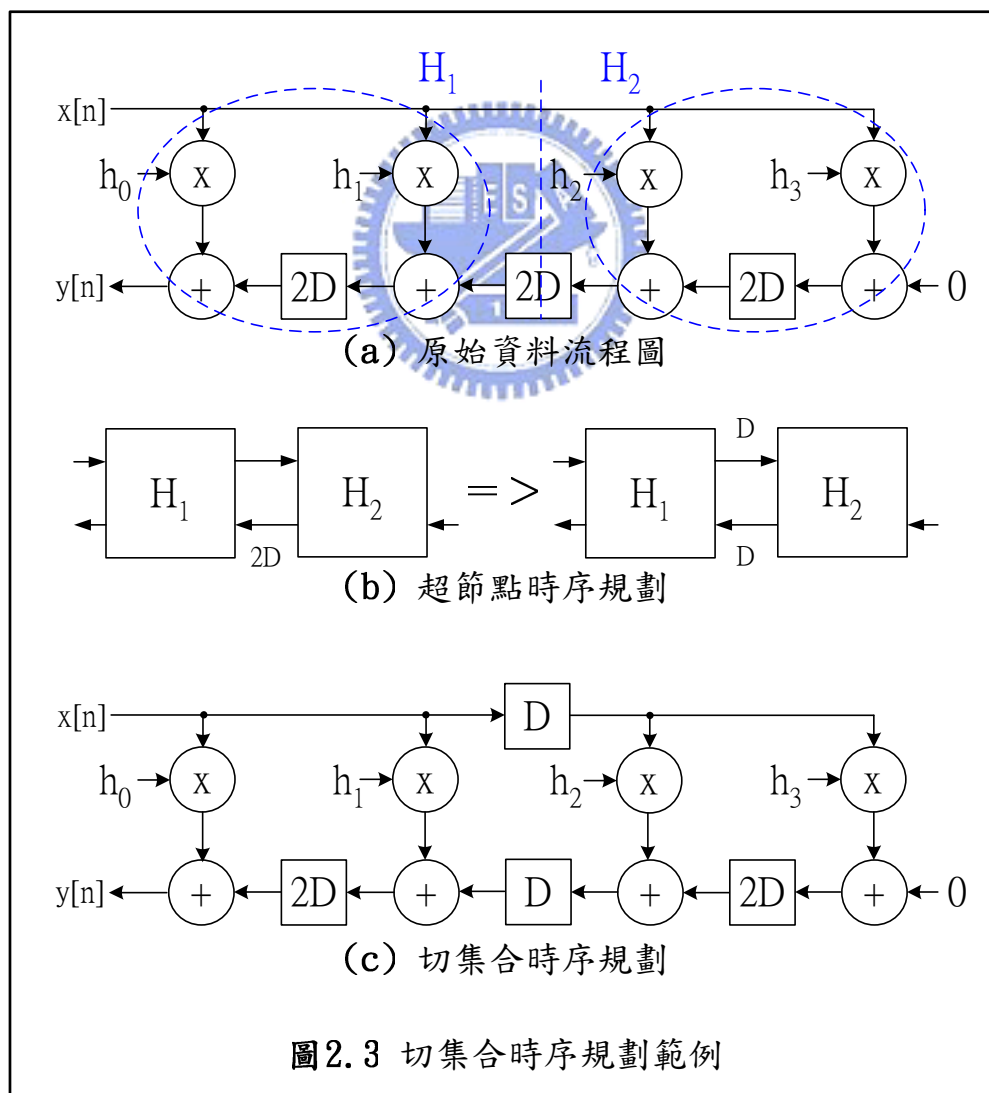
在試著對一個完整的系統做時序規劃之前，首先得了解如何對單一節點做時序規劃的工作，而節點轉換( Nodal Transfer )提供了最單純時序規劃的方法。考慮一個資料流程圖上的單一節點，則此節點輸入及輸出連線上的記憶元件，可在不改變資料進出總時間的前提下做移動，也就是若單一節點所有的輸入端上均增加或減少  $n$  個記憶元件，則其所有的輸出端上也須同時減少或增加  $n$  個記憶元件，以確保在針對單一端點做時序規劃考量時，不會影響到系統的輸入輸出關係，下圖 2.2 為一個簡單的節點轉換的例子。



如上圖所示，若我們在節點  $V1$  的兩個輸入端分別各減少一個記憶元件( $D$ )，則必須在節點  $V1$  的輸出端加回一個記憶元件，以確保系統輸入輸出時序的正確性，此即為節點轉換的時序規劃方法。

## 2.2.2 切集合時序規劃

延續上一節節點轉換的概念，當我們要替一個由多個節點所組成的系統做時序規劃時，只要可以正確的將系統切分為多個小系統，每個小系統都可以視為一個單一節點，接著利用節點轉換的概念針對單一節點(切分後的小系統)做時序規劃，即可正確的轉移系統內的記憶元件而不會改變原本系統輸入輸出的時序關係；這種將系統切分為小系統的方法並且用超節點的觀點來做時序規劃的方法稱之為切集合時序規劃(Cut Set Retiming)。圖 2.3 為切集合時序規劃的轉換說明，圖中的虛橢圓線可視為此系統的超節點，而虛直線可視為切集合。



## 2.3 第一篇系統化摺疊演算法[9]

[9]這篇提出系統的摺疊轉換演算法來對任意數位訊號處理演算法的資料流程圖進行摺疊動作，其中的摺疊集合(Folding Set)規畫了處理器(Processor,此處指摺疊架構中最基本的運算單元，通常為乘加器)在哪個摺疊集合中且在哪個時脈中該執行哪件工作，而其摺疊集合的產生是利用對演算法資料流程圖進行執行排程(Scheduling)及資源配置(Resource Allocation)來規畫的。

[9]提出了許多他們如何使用時脈規畫去針對各種類型的數位訊號處理架構進行摺疊的方法，其討論的數位訊號類型包括有單一時脈(Single Clock)與多時脈(Multiple Clock)及向前路徑(Forward Path)與含迴圈(Loop)等類型電路；以下介紹其如何對有限脈衝響應濾波器(單一時脈且向前路徑)執行摺疊架構轉換，如圖 2.4 及圖 2.5 所示—

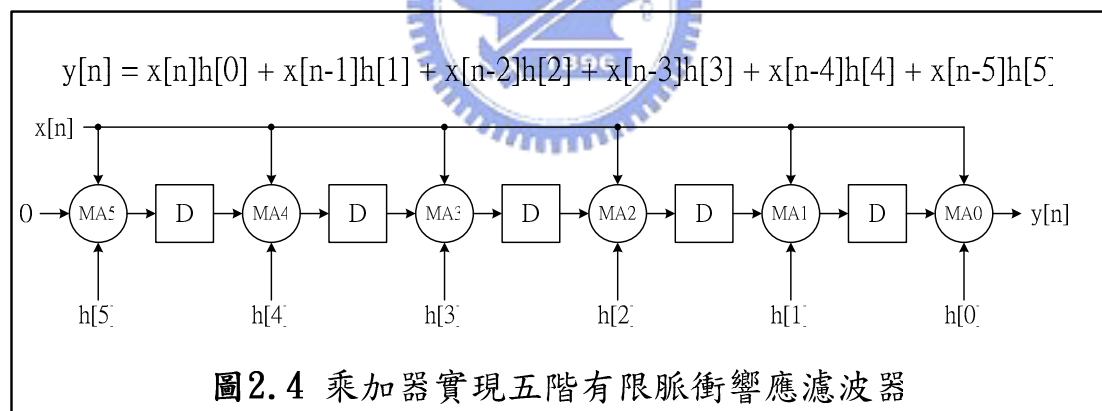
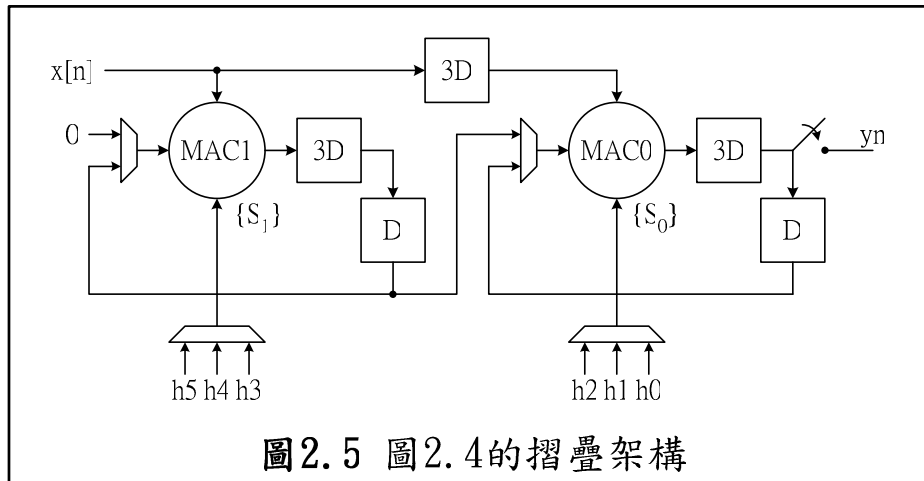


圖 2.4 顯示了以乘加(Multiply-Add)運算器來實現五階有限脈衝響應濾波器架構，圖中的 D 表示  $z^{-1}$  代表著 1 個 clock 的延滯(Delay)，而圖 2.5 為圖 2.4 使用時序規畫(Retiming)後，[9]推演出來的摺疊演算法架構圖，且圖 2.5 為其摺疊因數(摺疊因數見式子(3.1))為 3 的摺疊架構圖，圖中所示的摺疊集合  $S_1 = \{ MA5, MA4, MA3 \}$  為 MAC1 所執行的工作內容，而  $S_0 = \{ MA2, MA1, MA0 \}$  為 MAC0 所執行的工作內容。



接著簡易說明圖 2.5 其運作方式：它是將圖 2.4 的工作量等量切成兩段，MAC1 執行圖 2.4 的前半工作量，剩下的後半工作量利用多工器與暫存器配合時序並透過切換開關而傳給 MAC0 正確的值來執行累加。每個 MAC 詳細工作情形請參見表 2.1 的說明，表 2.1 中 MAC1 欄位為紀錄該時脈 MAC1 執行乘累加的運算結果，箭號為提示該值將在 4D 後透過多工器切換再被讀出而執行累加。

觀察圖 2.5 中的暫存器個數似乎有機會可以再化簡，原因是如果以直接方式實現圖 2.4 的架構(指未運用摺疊演算法)，就單純考慮中間運算暫存器的個數只需要 6 個(等於 taps 數目)，然而使用其摺疊演算法，中間運算暫存器個數卻增加到 8 個(圖中兩個 MAC 的輸出端 D 加起來有 8 個)，所以，發現有機會可以朝此方向來發展新摺疊演算法並降低中間運算暫存器個數亦即節省架構面積。

Clock	x[n]	MAC1	MAC0	y[n]
0	x0	x0h5	0	0
1	x0	x0h4	0	0
2	x0	x0h3	0	0
3	x1	x1h5	x0h2	0
4	x1	x1h4+ x0h5	x0h1	0
5	x1	x1h3+ x0h4	x0h0	y[0]
6	x2	x2h5	x1h2+ x0h3	
7	x2	x2h4+ x1h5	x1h1+ x0h2	
8	x2	x2h3+ x1h4+ x0h5	x1h0+ x0h1	y[1]
9	x3	x3h5	x2h2+ x1h3+ x0h4	
10	x3	x3h4+ x2h5	x2h1+ x1h2+ x0h3	
⋮	⋮	⋮	⋮	⋮

表2.1 圖2.5時序操作表

## 2.4 摺疊位元平面有限脈衝響應濾波器[6]

本節將仔細介紹另一現今摺疊演算法[6]，此篇將摺疊技術應用到心脈式位元平面的有限脈衝響應濾波器( Bit-Plane Systolic FIR )架構上，並詳細介紹位元平面摺疊演算法的公式推導，而提出了如何去摺疊轉換位元平面架構的資料流程圖( Data Flow Graph )，最後成功地實現其位元平面摺疊演算法架構。

## 2.4.1 位元平面濾波器架構

因為有限脈衝濾波器的輸出公式如下：

$$y[n] = \sum_{i=0}^{K-1} h[i]x[n-i] \quad (2.1)$$

式子(2.1)中的  $h[i]$  為濾波器係數， $K-1$  為濾波器的階數(Order)， $y[n]$  為濾波器輸出訊號， $x[n]$  為濾波器輸入訊號。

位元平面濾波器架構圖如圖 2.6 所示，其中， $K-1$  為濾波器階數， $b$  為係數的位元長度， $h_i^j$  為係數  $h_i$  的第  $j$  個位元(其權重為  $2^j$ )，圖 2.6 顯示為一個二階有限脈衝響應濾波器且其係數位元長度等於 4 的位元平面架構圖。

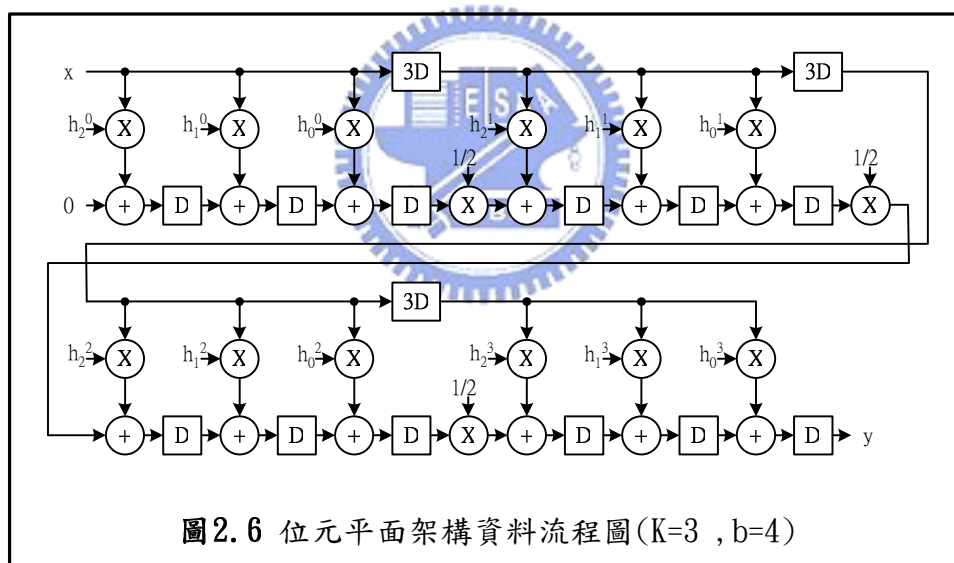


圖 2.6 是先做每一個係數的最低位元的乘加運算，然後依序做到每一係數最高位元的乘加運算；圖中的  $\otimes$  代表 1 位元的乘法器也就是 AND 邏輯閘，而  $\oplus$  代表 1 位元的全加法器(Full Adder，注意其輸出端將會有 Sum ,Carry 兩位元)；當乘上  $1/2$  時，代表資料右移一位。

## 2.4.2 位元平面濾波器轉換公式推導

根據圖 2.6 推導轉換公式(Transfer Function)如下：

$$\begin{aligned}
 G(z) &= Y(z)/X(z) \\
 &= z^{-1}(h_0^3 2^3 z^{-9} + z^{-1}(h_1^3 2^3 z^{-9} + z^{-1}(h_2^3 2^3 z^{-9} + z^{-1}(h_0^2 2^2 z^{-6} + z^{-1}(h_1^2 2^2 z^{-6} + z^{-1}(h_2^2 2^2 z^{-6} \\
 &\quad + z^{-1}(h_0^1 2^1 z^{-3} + z^{-1}(h_1^1 2^1 z^{-3} + z^{-1}(h_2^1 2^1 z^{-3} + z^{-1}(h_0^0 2^0 + z^{-1}(h_1^0 2^0 + z^{-1}(h_2^0 2^0)...)
 \end{aligned} \tag{2.2}$$

將式子(2.2)推展成濾波器為  $K-1$  階，且係數為  $b$  位元長度的通式：

$$\begin{aligned}
 G(z) &= z^{-1}(h_0^{b-1} 2^{b-1} z^{-(b-1)K} + z^{-1}(h_1^{b-1} 2^{b-1} z^{-(b-1)K} + \dots + z^{-1}(h_{K-1}^{b-1} 2^{b-1} z^{-(b-1)K} \\
 &\quad + z^{-1}(h_0^{b-2} 2^{b-2} z^{-(b-2)K} + z^{-1}(h_1^{b-2} 2^{b-2} z^{-(b-2)K} + \dots + z^{-1}(h_{K-1}^{b-2} 2^{b-2} z^{-(b-2)K} \\
 &\quad + \dots \\
 &\quad + z^{-1}(h_0^0 2^0 + z^{-1}(h_1^0 2^0 + \dots + z^{-1}(h_{K-1}^0 2^0)...)
 \end{aligned} \tag{2.3}$$

然後再將式子(2.3)去括號並重新整理得下列式子：

$$\begin{aligned}
 G(z) &= z^{-1} h_0^{b-1} 2^{b-1} z^{-(b-1)K} + z^{-2} h_1^{b-1} 2^{b-1} z^{-(b-1)K} + \dots + z^{-K} h_{K-1}^{b-1} 2^{b-1} z^{-(b-1)K} \\
 &\quad + z^{-(K+1)} h_0^{b-2} 2^{b-2} z^{-(b-2)K} + z^{-(K+2)} h_1^{b-2} 2^{b-2} z^{-(b-2)K} + \dots + z^{-2K} h_{K-1}^{b-2} 2^{b-2} z^{-(b-2)K} \\
 &\quad + \dots \\
 &\quad + z^{-[(b-1)K+1]} (h_0^0 2^0 + z^{-[(b-1)K+2]} (h_1^0 2^0 + \dots + z^{-bK} (h_{K-1}^0 2^0)...)
 \end{aligned} \tag{2.4}$$

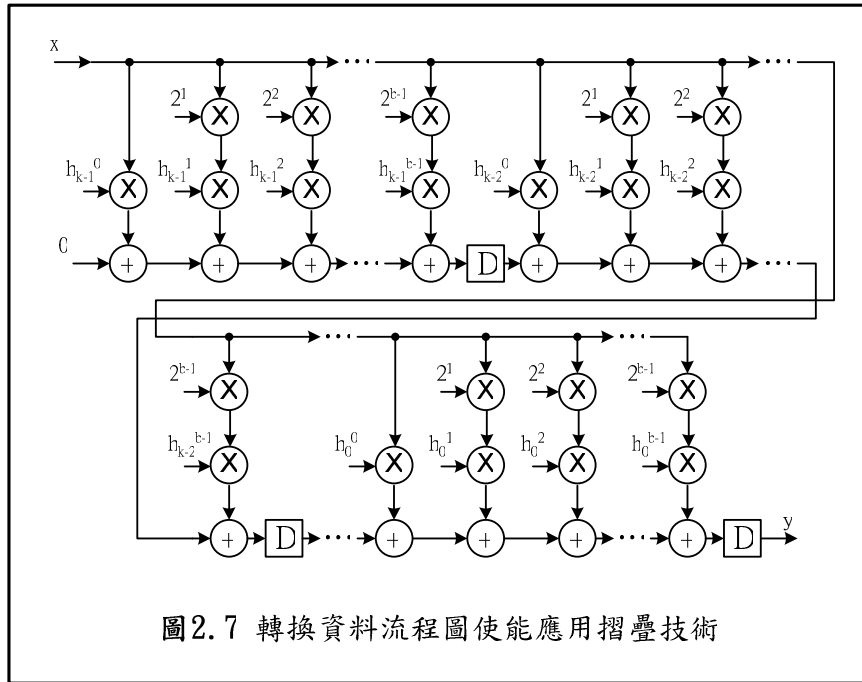
發現式子(2.4)中有共同的  $z^{-[(b-1)K+1]}$  因數，將之提出得下列式子  $G_1(z)$ ：

$$G(z) = z^{-[(b-1)K+1]} G_1(z), \text{ 所以 } G_1(z) \text{ 為一}$$

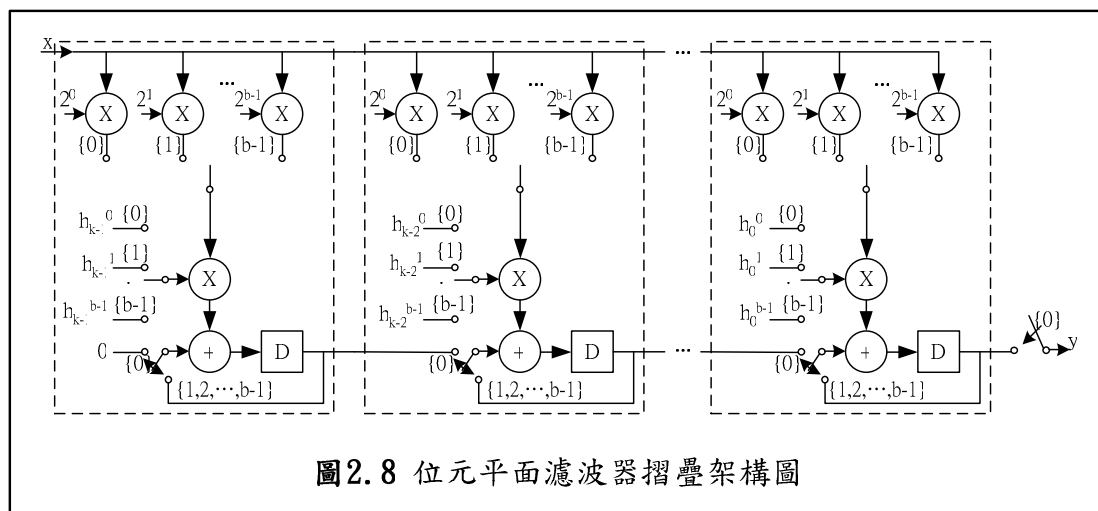
$$\begin{aligned}
 G_1(z) &= z^0(h_0^{b-1} 2^{b-1} + h_0^{b-2} 2^{b-2} + \dots + h_0^0 2^0 + z^{-1}(h_1^{b-1} 2^{b-1} + h_1^{b-2} 2^{b-2} + \dots + h_1^0 2^0 + \dots \\
 &\quad + z^{-1}(h_{K-1}^{b-1} 2^{b-1} + h_{K-1}^{b-2} 2^{b-2} + \dots + h_{K-1}^0 2^0)...)
 \end{aligned} \tag{2.5}$$

由式子(2.5)推論出位元平面的濾波器可以摺疊的架構圖如圖 2.7 所示：





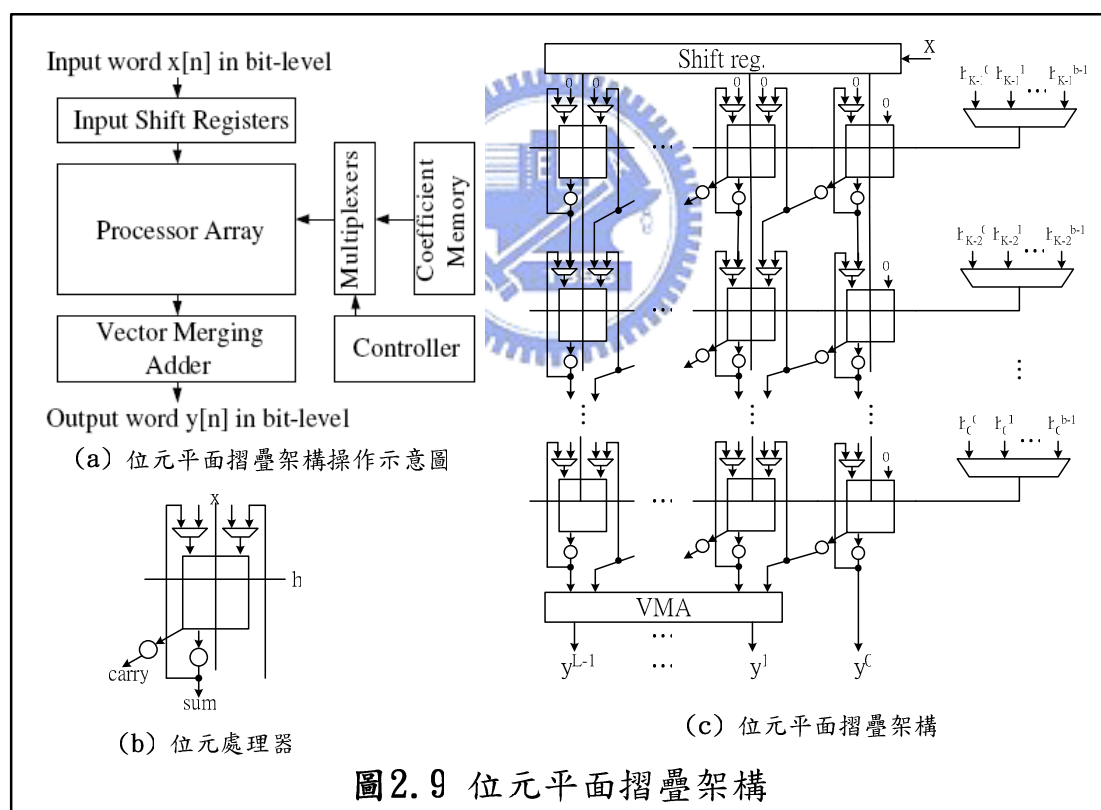
### 2.4.3 實現位元平面摺疊架構



實現摺疊位元平面濾波器架構如圖 2.8 所示，假設為  $K-1$  階有限脈衝濾波器，且其係數長度為  $b$  位元，所以，圖中虛線矩形總共會有  $K$  個，且其摺疊因

數為  $b$ ，原因是它分成  $b$  回合來完成一次  $K-1$  階有限脈衝濾波器運算； $\{ \}$  中的數字代表  $b$  回合中的第幾回合。觀察圖 2.8 並初步估計其暫存器個數應該為  $K$  個，但又因為最基本的運算器是一個全加器 (Full Adder)，推論要有兩倍的記憶儲存元件 (Sum, Carry)，所以，中間運算的暫存器個數顯然亦未達到最佳化，仍然有機會可朝向減少中間運算暫存器的方向來發展一套新摺疊演算法。

資料輸入、係數輸入與資料輸出實際操作流程圖，請參考圖 2.9，將會有更清楚的概念，(a) 中的 Processor Array 那塊即是執行圖 2.8 的摺疊架構；(b) 顯示為位元平面摺疊演算法中基本構成元件一位元處理器，由兩個多工器、兩個記憶元件 (正反器) 及全加器所組成，其連線方式如圖所示；(c) 以 (b) 圖為基本元件搭配多工器、移位暫存器及 VMA 完成 (a) 架構圖。



---

---

## 第三章

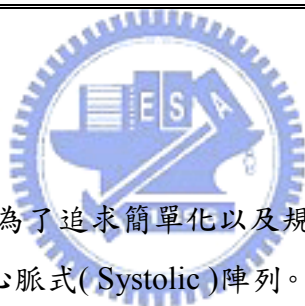
---

---

### 新摺疊演算法之推演與建立

---

---



現今有許多的硬體設計為了追求簡單化以及規律性，大多採用運算單元陣列的形式，其中最著名的就是心脈式(Systolic)陣列。心脈式陣列是由 Kung [16] 在 1980 年提出來的一種硬體架構，這種架構是由一些簡單的基本運作單元(PE)所連接組成，所以其架構擁有規則且簡單的特性，適合用在超大型積體電路的設計上。此外，心脈式陣列用到了大量的管線化(Pipelining)以及多重運作(Multi-Processing)，因此，它可以達到高速的運作效能以及維持很高的資料處理速率(Throughput)。

當應用所需資料處理速度(Throughput)小於實際電路可以操作的速度時，[9]-[11]推薦了一些摺疊演算法來減少硬體實現面積。然而，有限脈衝響應濾波器是非常適合用來討論及實現摺疊技術的應用，因為其基本上只是一直在做重複乘累加的動作。

從另一個角度來看，倘若在整個系統中，有限脈衝響應濾波器方塊的下一級若不需要很高的處理速率(Throughput)，那麼此高處理速率的優點，勢必可以拿

來交換一些其他的好處。摺疊硬體就是一個以降低處理速率而獲得硬體面積減少的一個方法，將一連串基本運作單元的陣列透過摺疊演算法，進而使用少量的運算單元，而使用分時來完成原本的工作，雖然完成工作的時脈加長，但運算單元卻減少，對於不必要很高處理速率的系統，不失為一個好方法，而其最大的好處除了以時間上的優勢取得空間上的改善外，摺疊後的架構也有可重覆使用性，增加了硬體的彈性。

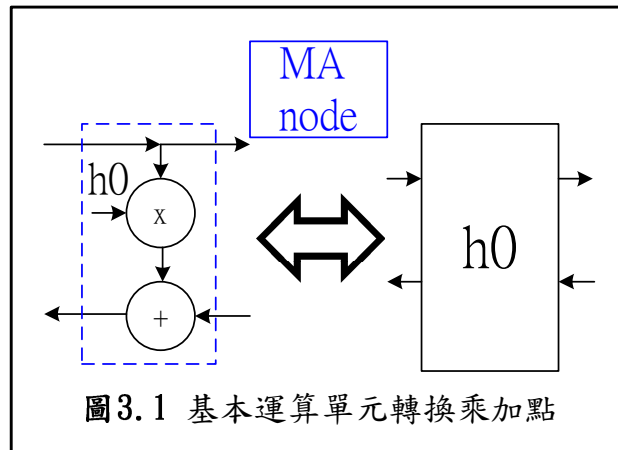
### 3.1 新摺疊演算法的硬體架構推導

對於硬體摺疊的好處，是讓人顯而易見的，但是對於摺疊方式的推導，卻沒有一個非常明確的流程讓人遵循，所以在這一個章節中，首要的目的就是介紹硬體摺疊的步驟，並將其稱為新摺疊演算法，只要依照步驟實行，整個過程是相當有規劃地進行，且摺疊後的架構在時脈上也不至於出錯。

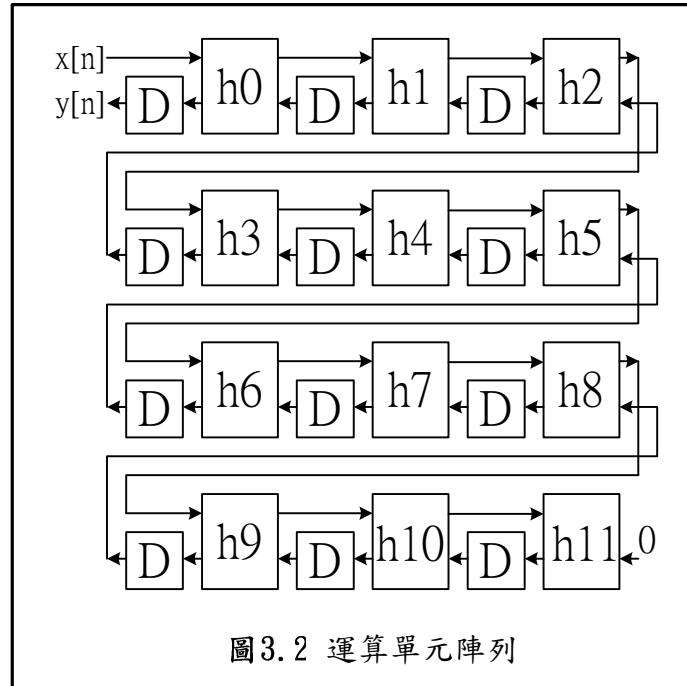
一般傳統運算單元陣列，對於訊號傳遞的路徑，可分為兩類：向前路徑 (Forward Path) 以及回授路徑 (Feedback Path)。向前路徑上的訊號，在兩兩相鄰的運算單元之間互有關係；而回授路徑上的訊號，則只有在本體運算單元中有影響。

對於陣列中的每個基本運作單元，主要區分成乘法器與加法器兩部分；而陣列中儲存資料的暫存器，則將之抽離開資料路徑 (Data Path)，另外再規劃一個區塊稱為暫存器檔案 (Register File)，專門負責暫存器資料儲存與更新，並有技巧性地對其暫存器定址，成立此一區塊的目的是：希望讓本篇推導的新摺疊演算法架構與其他現今相關摺疊演算法架構相比，有較佳成本效益即希望能有較少的硬體面積與較低的功率消耗。

圖 3.1 所示即為一個基本運算處理單元—乘加器( MAC )，為了讓新摺疊演算法推導表示上方便，本篇將乘加器視為一個乘加點( MA Node )，往後在運算單元的表示上，也都以乘加點的方式呈現。



現在就以一個串接 12 個基本運作單元的有限脈衝響應濾波器為例子，說明新摺疊演算法的推導。一開始在摺疊之前，首先考慮所需要使用的運算單元個數，假設採用 3 個運算單元，於是將 12 個基本運作單元分成 4 列，每一列的運算單元必須完整的排列對齊，這樣的目的是打算將原本一個回合內完成的資料計算，分成 4 個的時間點來運作，每一列各分配到一個時間點進行。圖 3.2 所示就是運算單元的排列方式。



接著，對整體的架構開始作一些簡單的推導轉換。將每個暫存器(為圖中所示的D)放大成(Scaling)4倍，接著在列與列的每個向前路徑上加入資料銜接的暫存器(位於每列最後的深色D)，這麼一來，在每回合的第一個時間點時， $x[n]$ 送入資料，僅第一列的三個乘加器( $h_0, h_1, h_2$ )開始動作，且將其乘加運算後的結果存入該列各暫存器的第一個位置，同時，向前路徑上第一列最後一個暫存器的值，也會傳遞至第一列與第二列的銜接暫存器中，而第二列、第三列及第四列的運算單元則沒有動作；到了第二個時間點，第二列的第一個乘加器就讀入銜接暫存器的值，第二列的運算單元開始運作，並且將結果存入該列各暫存器的第二個位置，其餘三列的運算單元均不動作，當然向前路徑上第二列最後一個暫存器的值也傳至下一個銜接暫存器；到了第三個時間點，第三列的運算單元會開始計算並將結果存到各暫存器的第三個位置，第一、二、四列則無動作；到了第四個時間點，第四列的運算單元會開始計算並將結果存到各暫存器的第四個位置，而第一、二、三列則無動作；之後進入下一個新的回合，如此一直反覆循環。圖3.3說明了上述的動作。

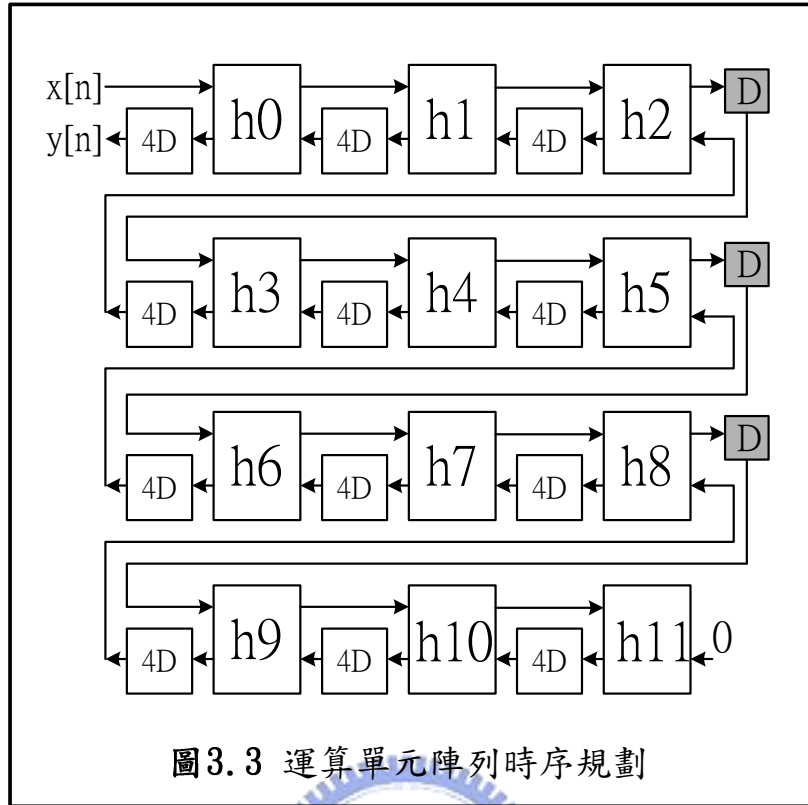
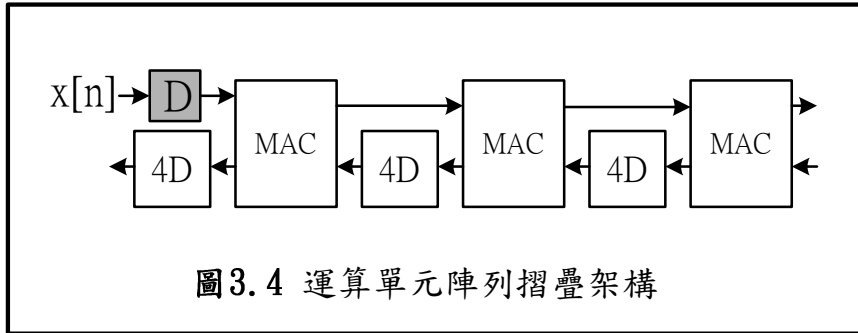
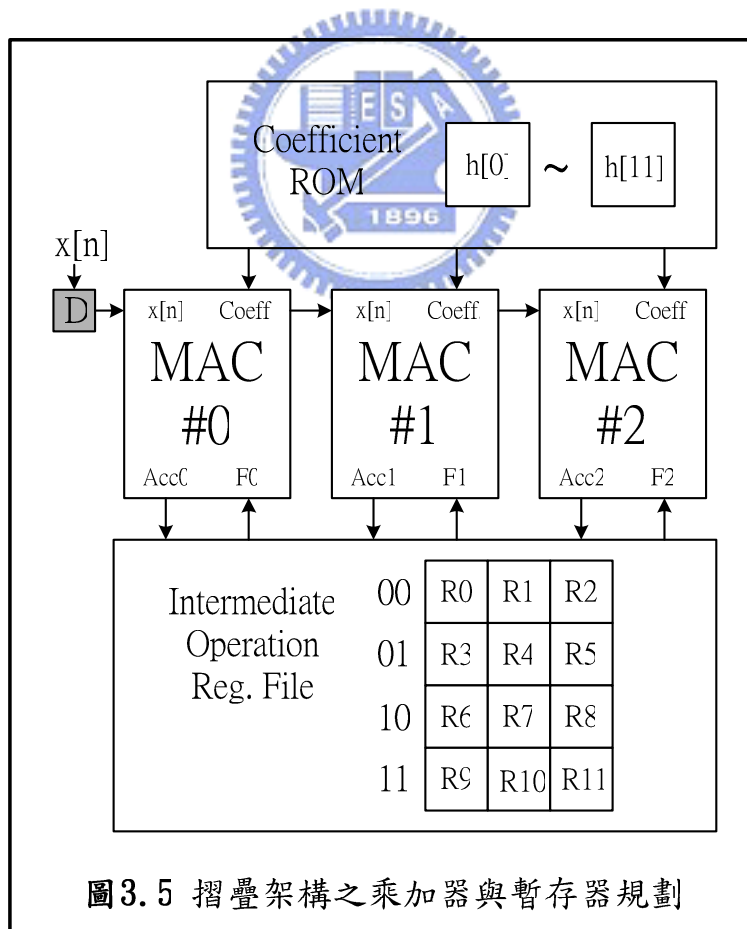


圖3.3 運算單元陣列時序規劃

由此可知，這四列中的第一個乘加器，在每回合的四個動作時間點中是不重疊的，也就是每列的第一個乘加器不會在同一個時間點同時動作，而每列的第二、三個乘加器亦然。由於在每列中相對位置的乘加器的工作時間點是兩兩互斥的，也就是工作的時機互相不衝突，所以可以將四列的運算單元壓縮成一列，成為圖 3.4 所展示的。於是，只要在一個回合中，送入一個  $x[n]$  資料，並且在這個回合裡，根據不同時間點，調整每個乘加器各自所需輸入的來源暫存器，以及將每個乘加器輸出所需儲存的計算結果存入正確的暫存器位置，就可得到正確的運算。這個方法把整體的運算時間拉長了，但卻減少四分之三的乘加器數目，這就是目的所在。



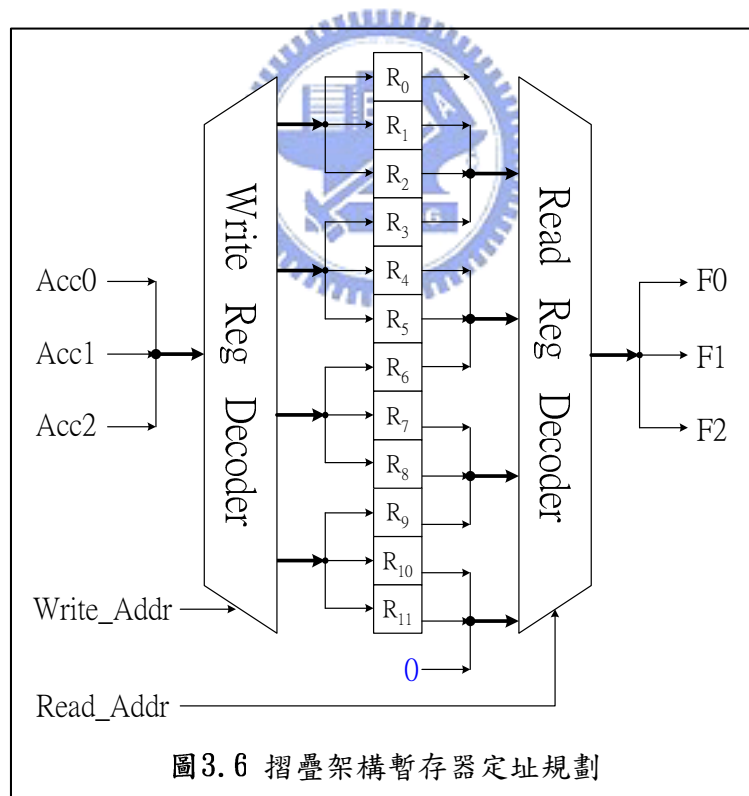
原先 12 個運算單元的陣列，經過摺疊演算法調整後，可以明確的分出乘加器群 (MAC Group) 及暫存器檔案 (Register File)，要使整個運算流程正確的動作，只需要在每個時刻點調整暫存器檔案所需輸出及儲存的資料，並由外部  $x[n]$  輸入至銜接暫存器，然後都是讀取銜接暫存器，就可以完成工作，至於乘加器群只要不停的運算就可以了。圖 3.5 是經過摺疊演算法所產生的架構圖。





值得一提的是，在暫存器檔案中每一行( Column )共有四個暫存器，就是先前推導過程中四倍暫存器(4D)大小所代表的意思。另外，暫存器檔案每一列( Row )中也有三個暫存器，同一列的三個暫存器是必須在同一時刻送到乘加器群運算，因此將每一列的三個暫存器一起作捆綁，給予同一個定址名稱，往後只需要正確控制暫存器位址，架構就能夠正常地運作。圖 3.6 即為暫存器捆綁及定址的示意圖，注意讀取與更新暫存器兩邊的解碼器對於暫存器位址的捆綁定義並不相同，而其捆綁的定義會錯開一個用來表示資料有向前傳遞的意思，原因在之後的排程矩陣( Scheduling Matrix )章節會有更詳細地介紹。

摺疊演算法最主要的精神就是在於時脈上的規劃，將每個回合的動作分時完成，而依此演算法所推演出的硬體架構，擁有可重複使用的特性。在下一節中主要說明套用摺疊演算法時運算單元個數的選擇，以及摺疊硬體架構的好處。



## 3.2 以摺疊演算法實現硬體之方法與特色

在使用摺疊演算法之前，首要的工作就是決定要使用的乘加器個數。假設現在系統中有一個方塊，其硬體架構由  $K$  個相同運算單元(例如：硬體架構為  $K-1$  階有限脈衝響應濾波器)所組成，且其實際硬體電路可以實現到的最大處理速度 (Throughput) 為  $xput_{max}$ ，而這個方塊的規格所需的處理速率為  $xput_{spec}$ ，於是若使用  $r$  個乘加器來實現摺疊架構，則摺疊因數 (Folding Factor)  $f$  為：

$$f = \left\lceil \frac{xput_{max}}{xput_{spec}} \right\rceil \geq \frac{K}{r} \quad (3.1)$$

藉由上列的式子計算，可以估算出新摺疊演算法架構採用的乘加器數目  $r$ ，且將原本一回合中的計算量，分為  $f$  個時間點來達成，而摺疊後的架構不僅能夠符合規格要求的資料處理速度，而在乘加器個數方面也比較原先的少，讓硬體實現面積可以減少。

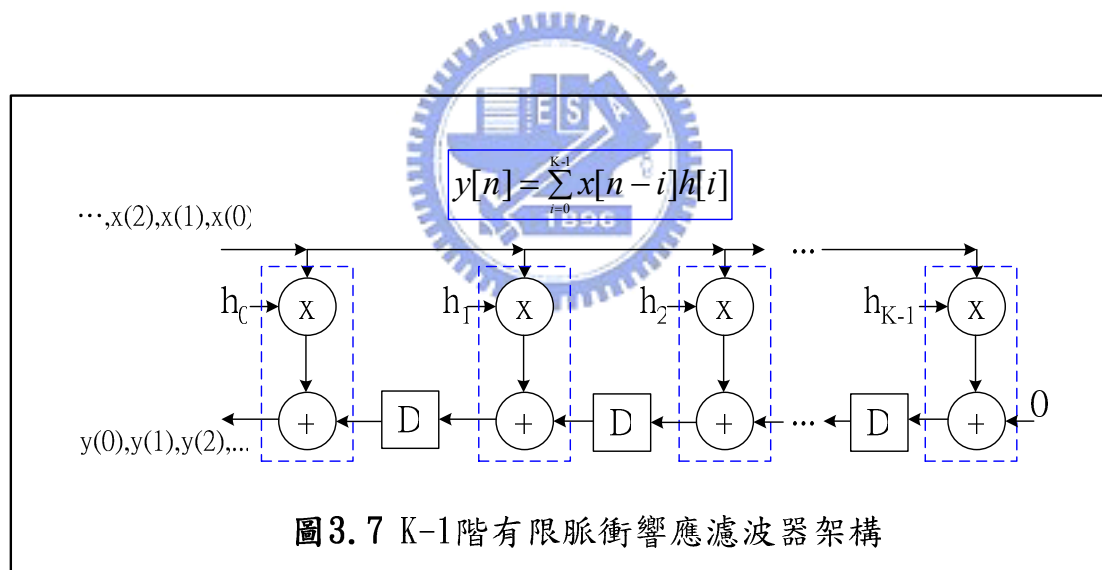
摺疊演算法非常適合於處理速率不需要很高的陣列硬體，尤其是對於陣列的運算單元個數會因規格不同而有數目上變化的架構。以有限脈衝響應濾波器架構為例，整個濾波器主要是由乘加器與暫存器所構成，非常適合來做摺疊演算法的應用。當對於這種硬體架構來實行摺疊時，則以最差狀況的處理速率及陣列長度來估計所採用的運算單元個數，摺疊之後的架構可視有無面積上的限制而來調整需幾個乘加器來實現為恰當，舉例說明，若在符合規格的條件下(指摺疊後的處理速度夠快)， $r=3$  比  $r=6$  所需較少的硬體面積，但處理速度卻需要再快 2 倍，相對地，功率消耗可能亦會隨之上升，必須視實際規格的需求來做變化應用，但前提是摺疊後的處理速度必須符合規格情形下，這樣的硬體架構便能展現了可重複使用的特性，亦可彈性地調整摺疊架構。

### 3.3 新摺疊演算法一( Parallel-In )

在介紹新摺疊演算法之前，讓我們先從推導有限脈衝濾波器的轉移函數 ( Transfer Function ) 開始， $K$  為有限脈衝濾波器的 taps 數， $r$  為摺疊演算法架構中乘加器的個數：

$$G(z) = \frac{Y(z)}{X(z)} = \sum_{i=0}^{K-1} h_i z^{-i} = \sum_{j=0}^{f-1} (z^{-jr} (h_{jr} + h_{jr+1} z^{-1} + \dots + h_{jr+(r-1)} z^{-(r-1)})) \quad (3.2)$$

再來介紹有限脈衝響應濾波器架構，如圖 3.7 所示為一調換形式( Transposed Form )的  $K-1$  階有限脈衝響應濾波器，觀察可以區分出  $K$  個相同操作單元(指乘加器)，且每個操作單元之間都有暫存器儲存乘加運算結果，發現其非常適合做為摺疊演算法的應用場景。

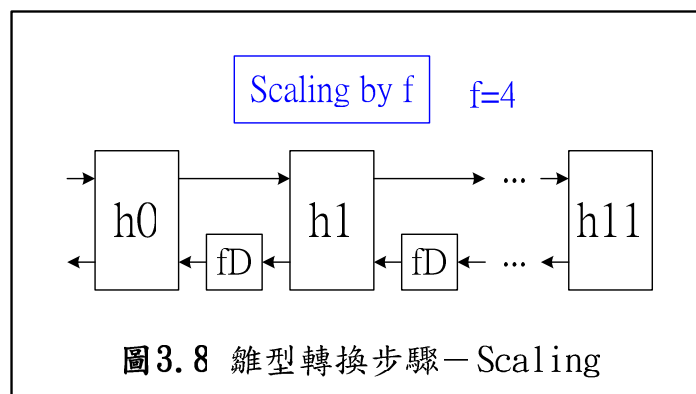


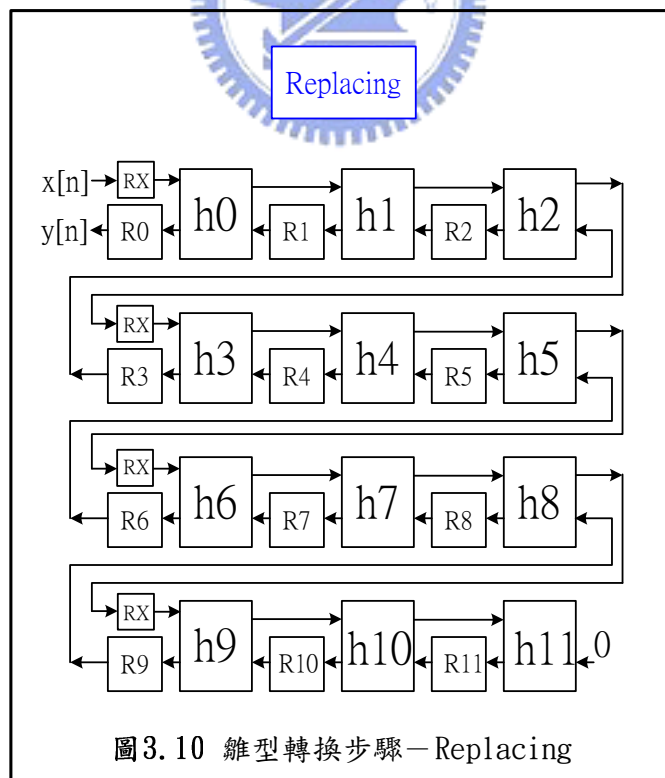
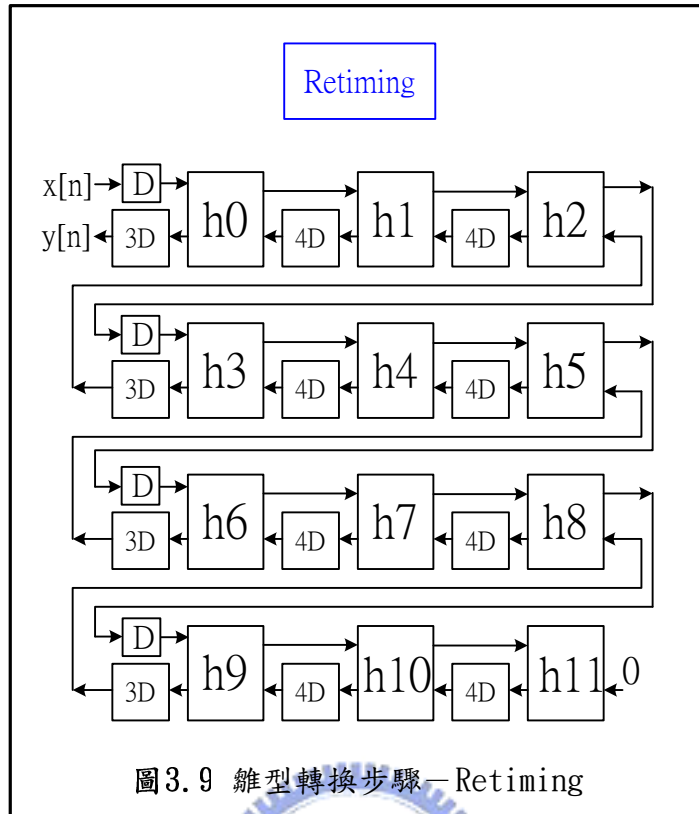
#### 3.3.1 雛型轉換

新摺疊演算法一( Parallel-in )即是以  $r$  MACs 平行處理一 Short-Length ( $r$  taps) FIR 重複執行  $f$  次運算來實現一 Long-Length ( $K$  taps) FIR 的濾波效果；以  $K =$

12，而  $r=3$ (以 3 個乘加器來實現摺疊演算法)為例說明如何做離型轉換，而新摺疊演算法一的離型轉換步驟如下—

- 1.將有限脈衝響應濾波器中間的每一乘加動作轉換為基本運算操作單位 MA 乘加點如圖 3.1 所示；
- 2.再來將圖 3.7 中的 D 放大(Scaling)成摺疊架構所需倍數(倍數依摺疊因數  $f$  而定，若  $K=12$ ， $r=3$ ，則  $f=4$ )如圖 3.8 所示；
- 3.有了足夠倍數的 D，便能使用時序規劃(Retiming)技術來做移轉摺疊架構所需的暫存器規劃如圖 3.9 所示；
- 4.最後再將圖 3.9 所示的 D 位置置換(Replacing)為實際使用的暫存器，而暫存器必須與下一小節新摺疊演算法的排程矩陣(Scheduling Matrix)搭配使用，則便能夠使得新摺疊演算法架構在記憶體管理方面達到最佳化如圖 3.10 所示，而圖 3.10 中的 RX 將會共用同一個暫存器來實現它，暫存器 RX 的目的為確保摺疊架構輸入輸出端有暫存器保護。





### 3.3.2 排程矩陣

將有限脈衝響應濾波器架構轉換為我們的新摺疊演算法應用雛型後，再來就是要將  $r$  個最小基本運算操作單元 MAC 的每一輸入、輸出端做好管理，就能完成有限脈衝響應濾波器的運算功能，以下介紹 4 個排程矩陣( Scheduling Matrix ) 如何應用其來完成排程動作，並且達到記憶體管理的最佳化(即暫存器個數減到最少)：

1. 資料輸入矩陣( Data Matrix )—輸入資料  $x[n]$  訊號到 MAC 的乘法器輸入端；
  2. 係數輸入矩陣( Coefficient Matrix )—輸入係數  $h[n]$  到 MAC 的另一乘法器輸入端；
  3. 讀取前一級運算結果矩陣( Feedback Matrix )—讀取前一級運算結果到 MAC 的加法器輸入端；
  4. 更新運算結果矩陣( Accumulator Matrix )—讀取 MAC 輸出端的運算結果數值並更新到所選取的目的暫存器。
- ( 操作原理：  $\text{Accumulator} = \text{Data} * \text{Coefficient} + \text{Feedback}$  ； )

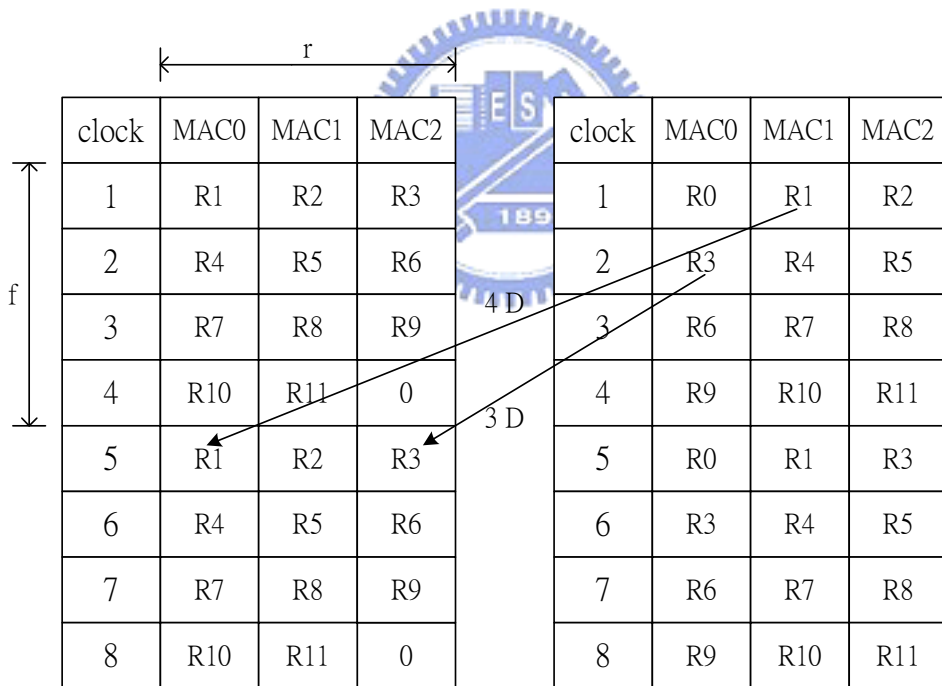


clock	MAC0	MAC1	MAC2
1	x0	x0	x0
2	x0	x0	x0
3	x0	x0	x0
4	x0	x0	x0
5	x1	x1	x1
6	x1	x1	x1
7	x1	x1	x1
8	x1	x1	x1

表3.1 Data Matrix

clock	MAC0	MAC1	MAC2
1	h0	h1	h2
2	h3	h4	h5
3	h6	h7	h8
4	h9	h10	h11
5	h0	h1	h3
6	h3	h4	h5
7	h6	h7	h8
8	h9	h10	h11

表3.2 Coefficient Matrix



clock	MAC0	MAC1	MAC2
1	R1	R2	R3
2	R4	R5	R6
3	R7	R8	R9
4	R10	R11	0
5	R1	R2	R3
6	R4	R5	R6
7	R7	R8	R9
8	R10	R11	0

表3.3 Feedback Matrix

clock	MAC0	MAC1	MAC2
1	R0	R1	R2
2	R3	R4	R5
3	R6	R7	R8
4	R9	R10	R11
5	R0	R1	R3
6	R3	R4	R5
7	R6	R7	R8
8	R9	R10	R11

表3.4 Accumulator Matrix

解讀排程矩陣：

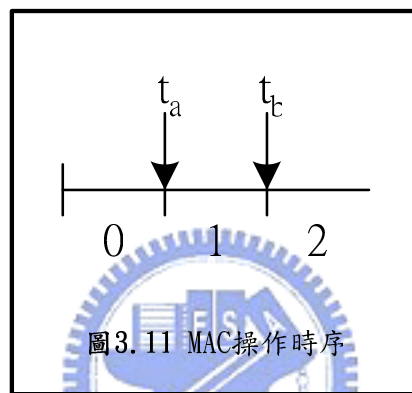
1.當  $\text{clock}=1$  時，3 個 MAC 皆同時動作，操作時序參考圖 3.11，而  $R1(t_a)$  為  $R1$  在  $t_a$  時刻的儲存值，

MAC0 在  $t_a$  時執行  $x_0 \cdot h_0 + R1(t_a)$  的運算，到  $t_b$  時才更新  $R0(t_b)$  為  $x_0 \cdot h_0 + R1(t_a)$ ，

MAC1 在  $t_a$  時執行  $x_0 \cdot h_1 + R2(t_a)$  的運算，到  $t_b$  時才更新  $R1(t_b)$  為  $x_0 \cdot h_1 + R2(t_a)$ ，

MAC2 在  $t_a$  時執行  $x_0 \cdot h_2 + R3(t_a)$  的運算，到  $t_b$  時才更新  $R2(t_b)$  為  $x_0 \cdot h_2 + R3(t_a)$ ；

所以，在操作時序的規劃上是可執行的，並不會有無限迴圈的情況出現。



2.當  $\text{clock}=2, 3, 4$  時，與  $\text{clock}=1$  同理，注意當  $\text{clock}=4$  時，MAC2 的 feedback 值為 0，原因是此時 MAC2 為代表有限脈衝響應濾波器的最後一級乘加運算，所以其加法器的輸入端給 0。

3.當  $\text{clock}=5\sim 8$  時，重複  $\text{clock}=1\sim 4$  的動作，只是 Data Matrix 的  $x_0$  換成  $x_1$ ，表示輸入下一筆資料。


4.當  $\text{clock}=1$  時，寫入新的值到  $R1$  中，而要到  $\text{clock}=5$  時，才會讀取儲存在  $R1$  中的值，此時的  $R1$  實現 4D 的效果(指暫存器每隔 4 clocks 才做一次讀取更新值的動作，所以 1D 指的是每隔 1 clock 做一次讀取更新值動作的暫存器)，所以， $R2$  也是實現 4D 效果的暫存器；而  $R3$  是在  $\text{clock}=2$  時，寫入新值，而到  $\text{clock}=5$  時，才做讀取值的動作，所以， $R3$  是實現 3D 效果的暫存器。



5.就實現角度而言，R0~R11 都是可以用 1D 的暫存器去實現，每個暫存器透過類似圖 3.6 捆綁的定址方式，就能以 1D 去直現上述 4.的效果。

6.根據 Feedback Matrix 得知，在 clock=1 時，只會讀取 R1,R2,R3，而 clock=2 時，只讀取 R4,R5,R6，而 clock=3 讀取 R7,R8,R9，而 clock=4 讀取 R10,R11,0，每隔 4 clocks 重複一樣動作；而根據 Accumulator Matrix 得知，在 clock=1 時，只需更新 R0,R1,R2，而 clock=2 時，只更新 R3,R4,R5，而 clock=3 更新 R6,R7,R8，而 clock=4 只更新 R9,R10,R11，也是每隔 4 clocks 重複一樣動作，所以，只要小心把特定的 3 個暫存器位址做捆綁的話，可以簡化更新與讀取暫存器內容的解碼器的控制訊號。

7.根據排程矩陣，推演出每個時刻中暫存器內容的變化情形，如表 3.5 所示，注意 R0 每次更新的值即為  $y[n]$ ，所以， $y[n]$  只需每隔  $f=4$  clocks 讀取一次 R0 值。



clock	MAC0	MAC1	MAC2
1	(R0) x0h0	(R1) x0h1	(R2) x0h2
2	(R3) x0h3	(R4) x0h4	(R5) x0h5
3	(R6) x0h6	(R7) x0h7	(R8) x0h8
4	(R9) x0h9	(R10) x0h10	(R11) x0h11
5	(R0) x1h0+ x0h1	(R1) x1h1+ x0h2	(R2) x1h2+ x0h3
6	(R3) x1h3+ x0h4	(R4) x1h4+ x0h5	(R5) x1h5+ x0h6
7	(R6) x1h6+ x0h7	(R7) x1h7+ x0h8	(R8) x1h8+ x0h9
8	(R9) x1h9+ x0h10	(R10) x1h10+ x0h11	(R11) x1h11

表 3.5 暫存器內容

8. 以上 1~7. 介紹整除案例( $K = rf$ )，以下介紹不整除案例， $K = 12$ ， $r = 5$ ，則  $f = 3$ ，其排程矩陣如下所示。

clock	MAC0	MAC1	MAC2	MAC3	MAC4
1	x0	x0	x0	x0	x0
2	x0	x0	x0	x0	x0
3	x0	x0	x0	x0	x0
4	x1	x1	x1	x1	x1
5	x1	x1	x1	x1	x1
6	x1	x1	x1	x1	x1

表3.6 Data Matrix

clock	MAC0	MAC1	MAC2	MAC3	MAC4
1	h0	h1	h2	h3	h4
2	h5	h6	h7	h8	h9
3	h10	h11	C	C	C
4	h0	h1	h2	h3	h4
5	h5	h6	h7	h8	h9
6	h10	h11	C	C	C

表3.7 Coefficient Matrix

		r=5				
		MAC0	MAC1	MAC2	MAC3	MAC4
f=3	1	R1	R2	R3	R4	R5
	2	R6	R7	R8	R9	R10
	3	R11	0	0	0	0
	4	R1	R2	R3	R4	R5
	5	R6	R7	R8	R9	R10
	6	R11	0	0	0	0

表3.8 Feedback Matrix

clock	MAC0	MAC1	MAC2	MAC3	MAC4
1	R0	R1	R2	R3	R4
2	R5	R6	R7	R8	R9
3	R10	R11			
4	R0	R1	R2	R3	R4
5	R5	R6	R7	R8	R9
6	R10	R11			

表3.9 Accumulator Matrix

9. 不整除案例的排程矩陣運作方式請參考上述 1~7. 的運作方式，需注意的是表 3.9 中空白的框框代表空(Null)動作，表示不寫到任何暫存器中，主要是為了平衡摺疊週期每個 MAC 的工作量。

### 3.4 新摺疊演算法二( Serial-In )

在介紹新摺疊演算法二之前，讓我們先從推導有限脈衝濾波器的轉移函數 ( Transfer Function ) 開始介紹，改寫式子(3.2)為(3.3)式：

$$G(z) = \frac{Y(z)}{X(z)} = \sum_{i=0}^{K-1} h_i z^{-i} = \sum_{j=0}^{r-1} (z^{-jf} (h_{jf} + h_{jf+1} z^{-1} + \dots + h_{jf+(f-1)} z^{-(f-1)})) \quad (3.3)$$

根據式子(3.3)推導出新摺疊演算法二—Serial-In，最主要與 Parallel-In 不同的地方是係數輸入方式，以 3 MACs 來摺疊 12taps 的有限脈衝響應濾波器(即  $K = 12, r = 3, f = 4$ ) 為例來說明，Parallel-In 方法的 MAC0 將執行工作內容為 {h0, h3, h6, h9} 見表 3.2，而 Serial-In 方法的 MAC0 執行工作內容為 {h0, h1, h2, h3}，其實現方式見下圖 3.12 描述可以有更清楚的概念。

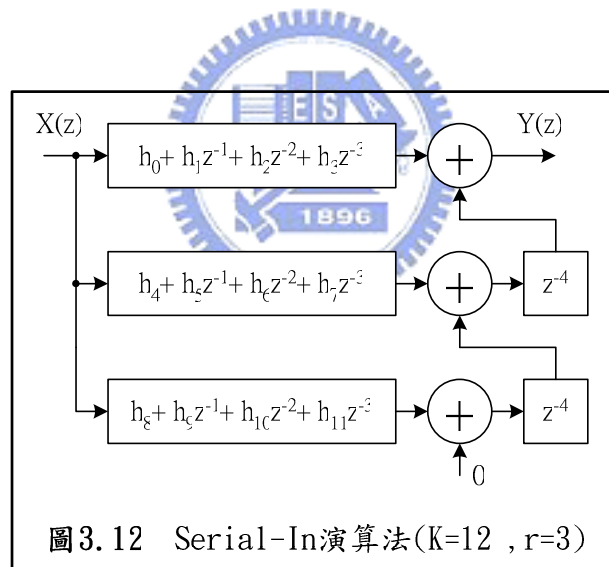
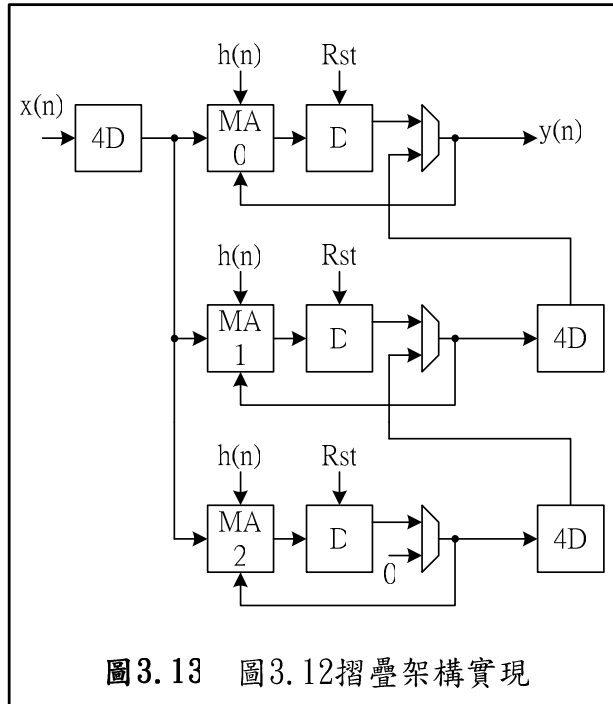


圖 3.12 中每一長方形框可視為一 short-length (4-tap) FIR，且共有 3 條(代表有 3 個 MACs)，此為 Serial-In 演算法的精隨；接著推導圖 3.12 演算法的硬體架構圖如圖 3.13 所示，圖 3.12 中長方形框實現方式為 1 MAC 加上 1 暫存器，而圖中的⊕以一多工器再配合上拉線來實現見圖 3.13，而圖 3.13 中 x 輸入端的 4D 暫存器實現方式為圓形定址(Circular Addressing)，暫存每一 short-length 所需 4 個 x 值。



---

---

## 第四章

---

---

### 硬體實現架構

---

---

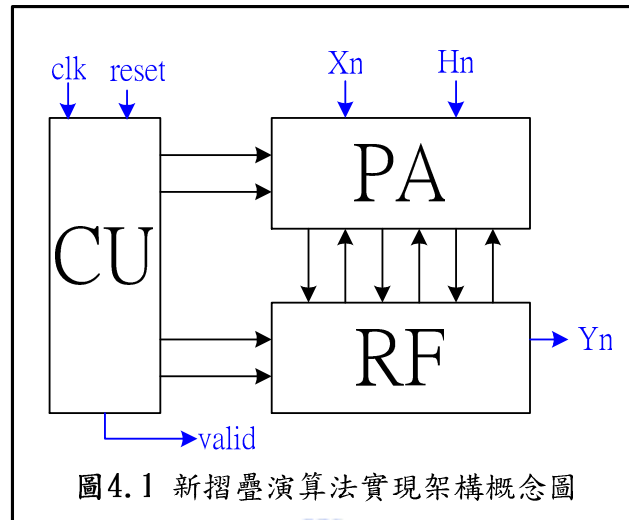


在介紹摺疊理論推導之後，再來就是考慮新摺疊演算法硬體實現的部分。有些在演算法上看似簡單的問題，在硬體實現上不見得相對單純，在接下來的內容之中，會作較為詳盡的探討；但主要焦點會集中在比較各種摺疊演算法在硬體實現部分的成本(指面積與功率消耗)分析討論。

首先，會先概括地介紹整個新摺疊演算法的完整電路架構，然後再個別介紹摺疊硬體實現的每一部分結構與操作，而實現摺疊演算法架構都是由三塊模組所構成見圖 4.1：一是處理器陣列(Processor Array, PA)主要由 MAC 所組成，負責有限脈衝響應濾波器的乘累加運算；二是暫存器檔案(Register File, RF)主要是由暫存器所組成，負責有限脈衝響應濾波器中間運算轉移資料的儲存與更新；三是控制單元(Control Unit, CU)主要是由計數器所組成，負責摺疊架構中處理器陣列與暫存器檔案的溝通，後面的章節都會一一作仔細的介紹。

接著討論各種摺疊演算法硬體實現架構的成本函數(Cost Function)，其一成本函數是討論硬體面積，另一是討論功率消耗；最後再以 IS-95 WCDMA Pulse Shaping 33 taps FIR Filter 見表 4.2 所列電路規格來實現各種摺疊演算法以及原始

架構，並且以 SYNOPSIS 公司的 Design Compiler 與 PrimePower 軟體來分析討論各種摺疊演算法以及原始架構的面積與功率消耗情形是否符合成本函數預期估計結果。



## 4.1 新摺疊演算法—硬體實現

實現的摺疊架構如圖 4.1 所示由三個模組所構成：處理器陣列、暫存器檔案、控制單元，以下三小節有更詳細的描述功能與實現方式。處理器陣列(Processor Array, PA)主要由 MAC 所組成，負責有限脈衝響應濾波器的乘加運算，觀察 PA 有外部輸入  $X_n$  與  $H_n$ ，並與 RF 之間有資料互傳，且接受 CU 的控制；暫存器檔案(Register File, RF)主要是由暫存器所組成，負責有限脈衝響應濾波器中間轉移資料的儲存與更新，觀察 RF 與 PA 有互傳資料的活動，並接受 CU 的控制，在預定的時序吐出  $Y_n$  的輸出；控制單元(Control Unit, CU)由計數器和一些數位邏輯閘所組成，負責摺疊架構中處理器陣列與暫存器檔案的溝通，觀察 CU 有  $clk$  與  $reset$  輸入訊號控制著 CU 的狀態(State)，並送出控制訊號到 PA 與 RF，及一輸出訊號  $valid$  為宣告此時  $Y_n$  為有效值。

### 4.1.1 處理器陣列

處理器陣列的實現如圖 4.2 所示，每個處理器元素( Processor Element ,PE ) 都是由一組乘法器與加法器構成亦稱為 MAC 共  $r$  個，圖 4.2 中符號  $X_n$  為一個  $m$  位元的輸入訊號， $H_0, H_1, \dots, H_{r-1}$  為  $b$  位元的係數輸入訊號，所以，MAC 中的  $\otimes$  為  $m+b$  位元的乘法器。

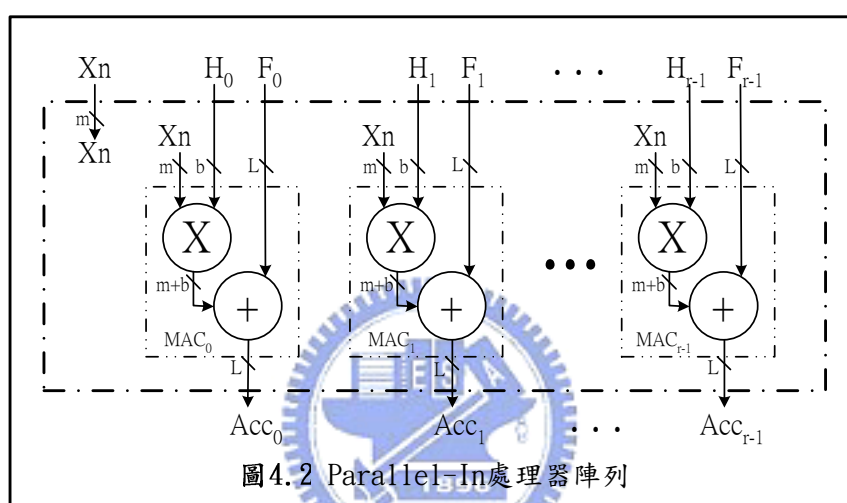
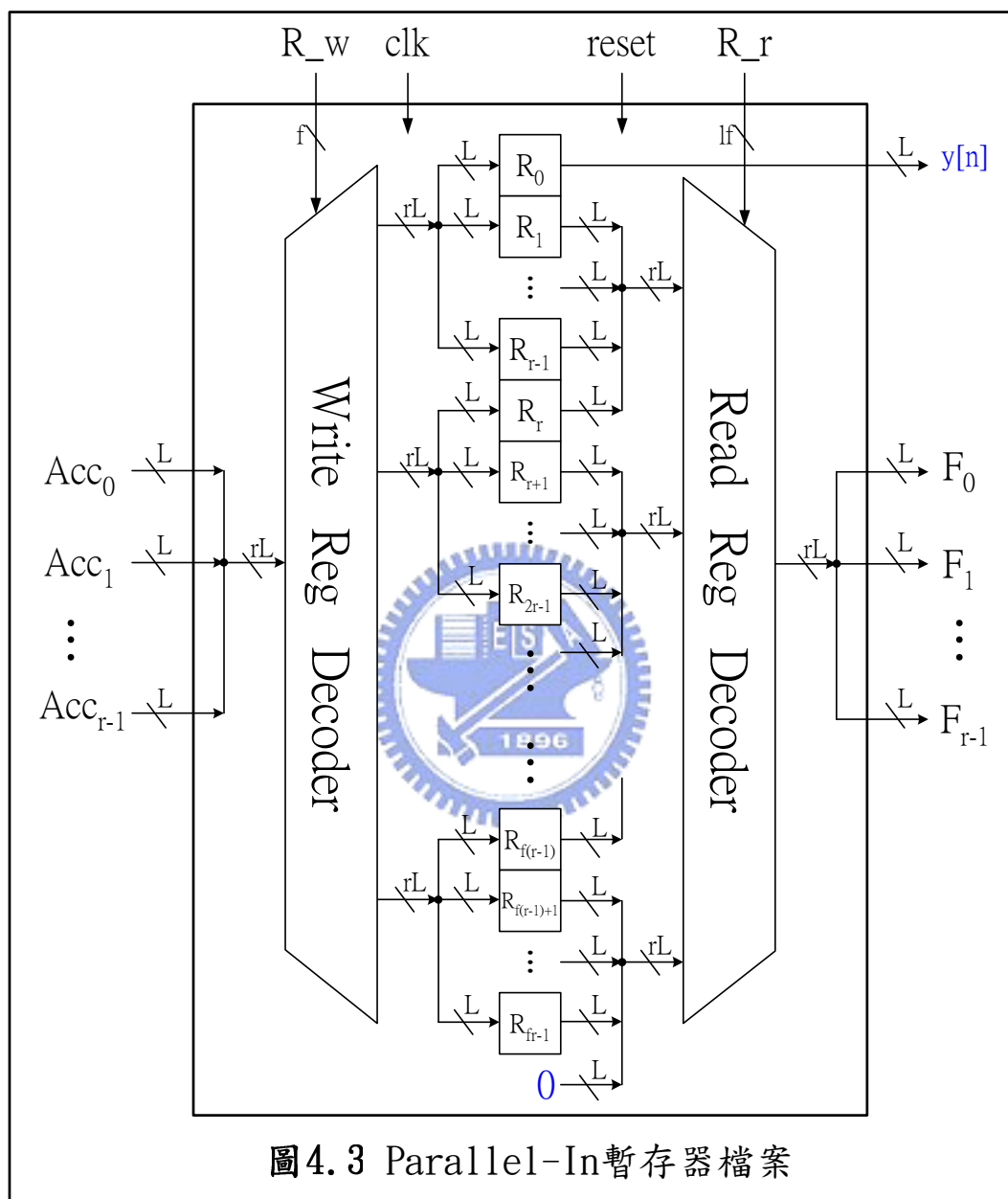


圖 4.2 中的  $L$  是代表設計規格所規定的精確度來決定暫存器的位元長度，其中， $F_0 \sim F_{r-1}$  與  $Acc_0 \sim Acc_{r-1}$  都是  $L$  位元長度的訊號，代表著與中間移轉暫存器進行傳遞資料動作，亦即是代表處理器陣列與暫存器檔案間互相溝通的訊號； $F_0 \sim F_{r-1}$  為由暫存器檔案所輸入的訊號(為前一刻暫存的運算結果)，並與  $MAC_0 \sim MAC_{r-1}$  一起做加法運算，而  $Acc_0 \sim Acc_{r-1}$  為這一刻  $MAC_0 \sim MAC_{r-1}$  的運算結果儲存回暫存器檔案中；圖 4.2 中虛線小方塊  $MAC_0$  簡易敘述其操作原理為： $Acc_0 = X_n * H_0 + F_0$ ；而其他 MAC 的操作同理。

## 4.1.2 暫存器檔案



$Acc_0 \sim Acc_r$  為由處理器陣列輸入到暫存器檔案的資料訊號線(亦即乘加器的輸出中間運算結果需被暫存，將其送到暫存器檔案中特定的暫存器儲存)，用來寫入如圖 4.3 所示左半邊的  $f$  ( $f$  為摺疊因數)群暫存器其中一群(分群的方法請參考第三章所敘述的排程矩陣原理)，而寫入暫存器的解碼器由控制單元輸入的控制訊號線  $R_w$  來選擇欲更新到哪一群( $r$  個)暫存器中。

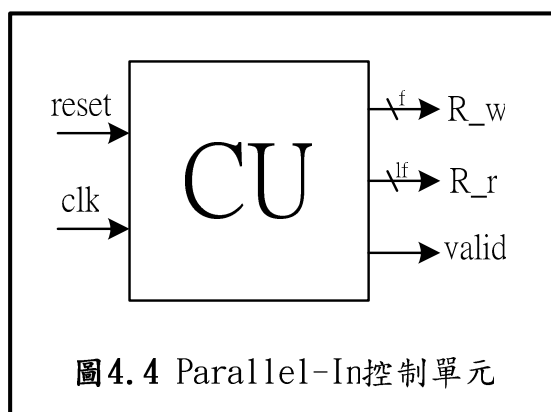


$F_0 \sim F_r$  為輸出到處理器陣列的資料訊號線( 用來做為 MAC 中的加法輸入端，即代表為加上前一刻的中間運算結果 )，用來讀取圖 4.3 中所示右半邊的  $f$  群暫存器其中一群( 分群的方法請參考第三章所敘述的排程矩陣原理 )的儲存值，而讀取暫存器的解碼器由控制單元輸入的控制訊號線  $R_r$  來選擇讀取哪一群暫存器。

$R_w$  與  $R_r$  訊號線並不等長，因為寫讀暫存器解碼器的實現方式並不一樣，寫暫存器解碼器的輸入與輸出關係是一對多，必須以每群為單位增加 enable 訊號線，所以， $f$  群必須有  $f$  位元長度的訊號線，即  $R_w$  為  $f$  位元的訊號線；而讀暫存器解碼器可以使用多工器來實現(原因是輸入與輸出的關係是多對一)，所以， $R_r$  長度為  $\lceil \log_2 f \rceil$  位元。

圖 4.3 中 reset 與 clk，為由外部輸入的訊號，reset 負責工作為控制每一暫存器何時該清為 0 並且暫存器同時亦開始動作，而 clk 負責工作是暫存器每隔多久更新一次內容；然而其連線方式僅為示意，由於 reset 與 clk 都必須連接到每一暫存器上，為了避免讓圖 4.3 看起來太過於複雜，所以，僅以簡單的兩個箭號表示。因此，暫存器檔案只需一組暫存器的讀取、寫入的控制訊號(  $R_r$ 、 $R_w$  )，便能完成在該時脈中  $r$  個乘加器該讀取哪  $r$  個暫存器以及將運算結果更新到哪  $r$  個暫存器上；最後，圖中  $y[n]$  為摺疊架構的輸出訊號。

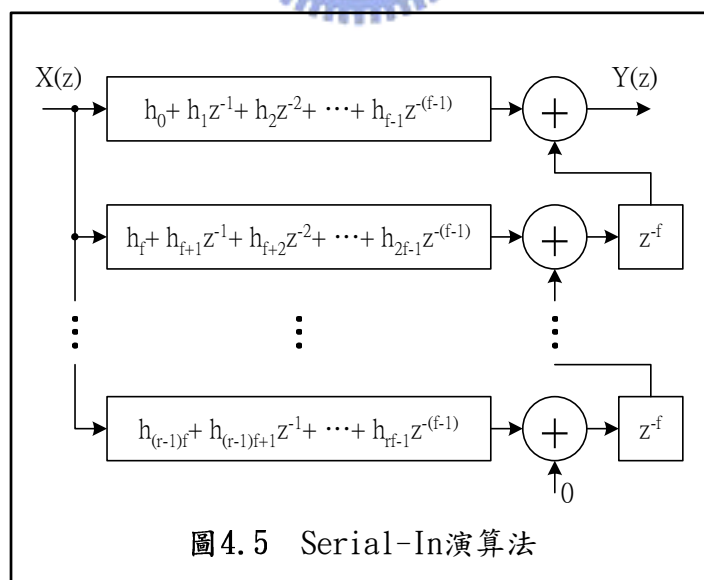
### 4.1.3 控制單元



clk、reset 為由外部輸入的時脈訊號與重置訊號，兩者負責整個控制單元目前處於哪一個狀態( State ); valid 為輸出訊號告知此刻暫存器檔案的輸出訊號  $y[n]$  值為有效輸出；而 R\_w、R\_r 為輸出到暫存器檔案的訊號，用來控制寫入或讀取哪一個中間運算暫存器。因為暫存器檔案中每一個暫存器的位置已經過精心安排而定址過，所以，只需依照排程矩陣的安排，在每一時序送出一組所需控制訊號 (R\_w、R\_r) 管理哪個暫存器該做讀取、哪個該做更新。

控制單元實現上應該是由一個  $\text{ceil}(\log_2 f)$  位元長度的計數器， $f$  為摺疊因數，並加上由邏輯合成軟體(如：Design Compiler)對每個狀態送出的控制訊號所合出的數位邏輯(此一部分不是本篇研究範圍，未加以詳細描述)所構成，且其控制單元整體佔整個折疊演算法架構比例甚小。

## 4.2 新摺疊演算法二硬體實現



新摺疊演算法二如圖 4.5 所示由  $r$  個長方形框所構成，而每一長方形框執行一小段連續係數乘加，其硬體實現見圖 4.6 所示，其每一 MAC 運作時序見圖 4.7

所示；然後再將圖 4.6 的硬體實現架構圖以圖 4.1 的概念再區分出三個工作區塊：處理器陣列、暫存器檔案、控制單元，以下三小節有更詳細的描述功能與實現方式。

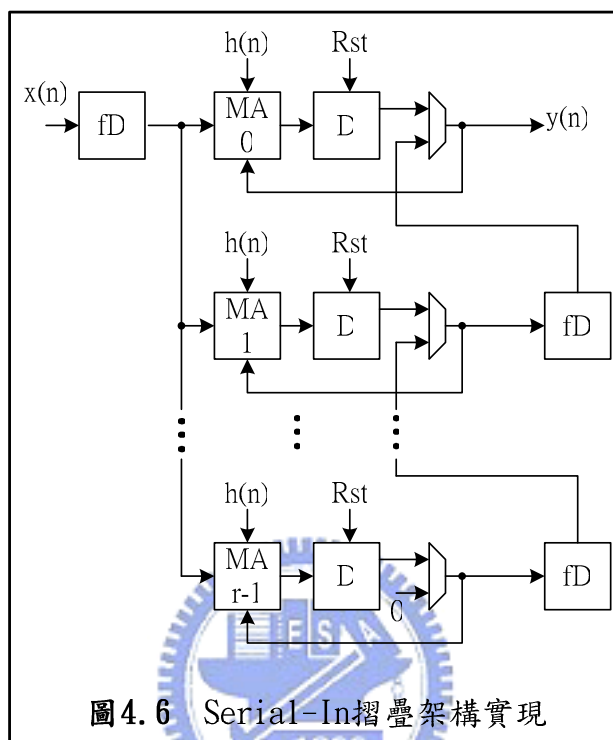


圖 4.6 Serial-In 摺疊架構實現

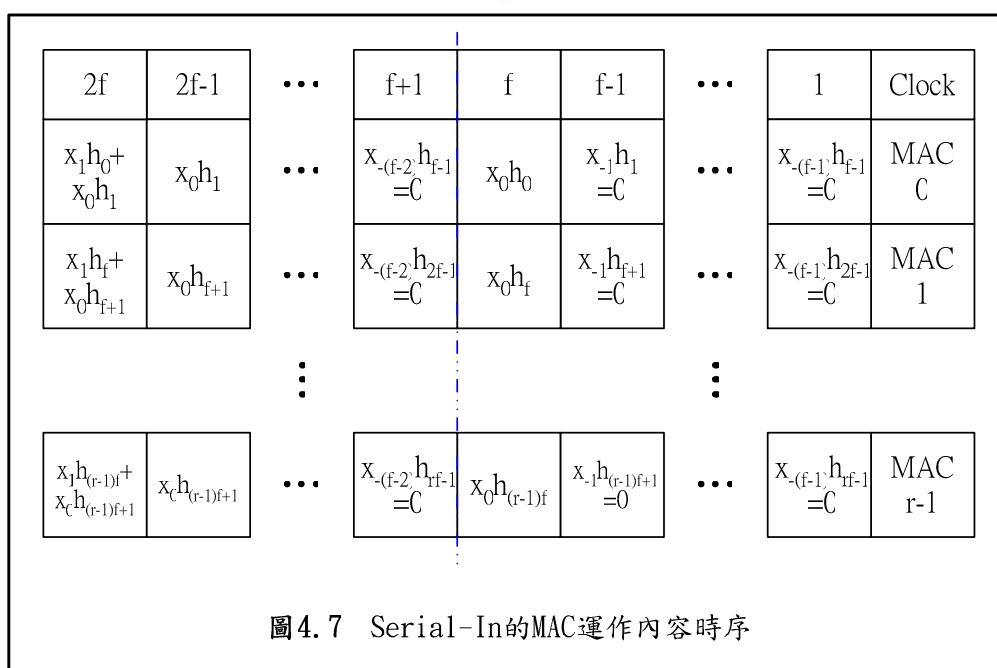
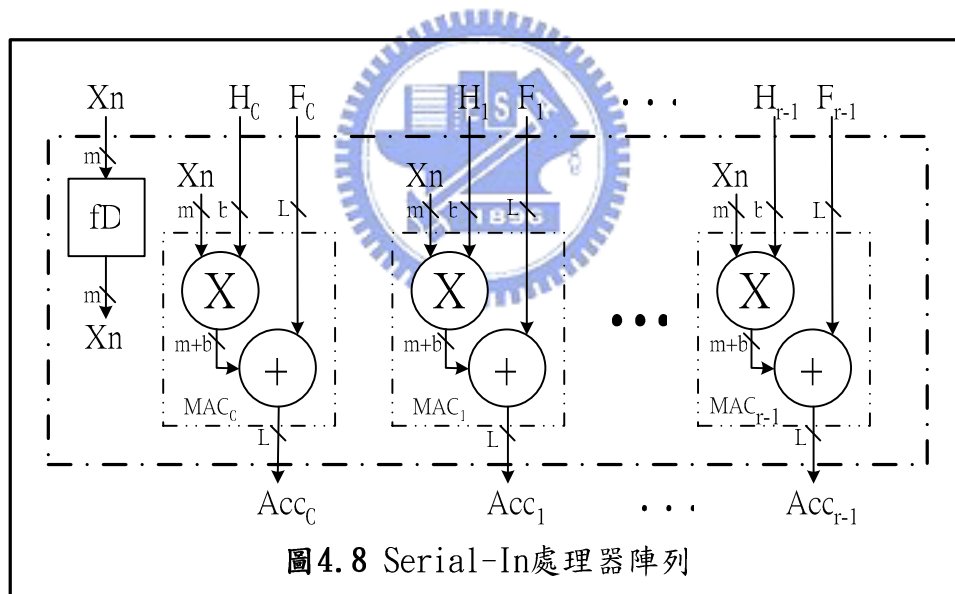


圖 4.7 Serial-In 的 MAC 運作內容時序

依據圖 4.7 MAC 運作內容 Clock=1 ~ f 時，x 輸入端的 fD 圓形定址暫存器存放內容為 $(x_{-(f-1)}, \dots, x_{-1}, x_0)$ ，而 Clock=f+1 ~ 2f 時，x 輸入端存放內容為 $(x_{-(f-2)}, \dots, x_0, x_1)$ ，每個 MAC 負責 1 short-length FIR 運算。

## 4.2.1 處理器陣列

處理器陣列的實現如圖 4.8 所示，每個處理器元素(Processor Element, PE)都是由一組乘法器與加法器構成亦稱為 MAC 共 r 個，其操作原理及描述請參考 4.1.1 小節，而 x 輸入端增加一 fD 暫存器為圓形定址模式。

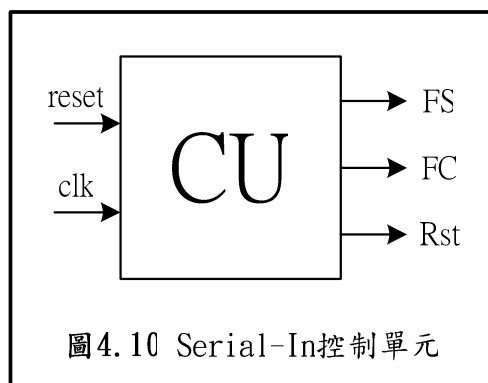


## 4.2.2 暫存器檔案

暫存器檔案架構如圖 4.9 所示，圖中的  $Acc_0 \sim Acc_{r-1}$  為由處理器陣列輸入到暫存器檔案的資料訊號線(亦即乘加器的輸出中間運算結果需被暫存，將其送到



### 4.2.3 控制單元



clk、reset 為由外部輸入的時脈訊號與重置訊號，兩者負責整個控制單元目前處於哪一個狀態( State )；而 FS、FC、Rst 為輸出到暫存器檔案的訊號，用來控制 F0 ~ Fr 以及 y(n)的輸出是否為正確值。因為暫存器檔案中在做寫入的動作不像 Parallel-In 那樣複雜，只會寫入固定的移位暫存器中，所以，只需做好 FS 與 FC 控制訊號的管理，便能正確地操作 Serial-In 摺疊演算法。

而控制單元實現上應該是由一個  $\text{ceil}(\log_2 f)$  位元長度的計數器，f 為摺疊因數，並加上由邏輯合成軟體(如：Design Compiler)對每個狀態送出的控制訊號所合出的數位邏輯(此一部分不是本篇研究範圍，未加以詳細描述)所構成，且其控制單元整體佔整個折疊演算法架構比例甚小。

## 4.3 硬體實現成本函數

在介紹完現今的摺疊演算法—第一篇系統化摺疊演算法[9]與位元平面摺疊演算法[6]，以及新摺疊演算法—Parallel-In 與 Serial-In 摺疊演算法後，再來便是討論分析各種摺疊演算法架構在硬體實現方面孰能較佔優勢，亦即何者演算法在

面積、功率消耗兩方面可以有較佳的表現；然而在成本函數中沒有考慮到執行時間的因素，原因是假設討論比較的前提是彼此都是使用相同的製程，並且彼此的資料處理速度( Throughput )皆以相同速度實現，所以，執行時間變得大家都一樣，因此，本篇的成本函數純粹就架構方面來討論功率消耗與面積大小的問題，展示各種摺疊演算法的面積與功率消耗的成長趨勢，並希望藉此凸顯何者演算法可以有最佳的表現；在以下的章節將藉由推論出的面積及功率消耗成本函數來分析新摺疊演算法與現今理論演算法[6][9]的成長趨勢，同時並假設彼此的電路是以單一時脈來實現的；表 4.1 列出我們在推導各種摺疊演算法實現架構的成本函數過程中會使用到的各種參數符號說明。

$i, j = \text{integers}$ $m = \text{bit-length of input sample}$ $b = \text{bit-length of coefficient}$ $K = \text{number of filter taps}$ $r = \text{number of MACs}$ $f = \text{folding factor}$ $L = \text{bit-length of transferred Reg}$ $x_{\text{put}}_{\text{max}} = \text{maximum throughput}$ $x_{\text{put}}_{\text{spec}} = \text{specified throughput}$
--

表4.1 成本函數符號意義

### 4.3.1 面積成本函數

在推論成本函數之前，我們先假設大家使用的乘法器與加法器是使用相同的技術( Technology )來實現的並且摺疊個數一樣，所以，並不將之列入成本函數中考慮；因此，簡化問題為討論彼此摺疊演算法硬體實現架構會使用到的記憶體來當作面積成本函數的評估，評估方式是以正反器( Flip-Flop )為記憶體面積的最小單位，而來估計其演算法所含正反器個數多寡。

### (a) 第一篇系統化摺疊演算法[9]

摺疊架構圖請參考圖 2.4 與圖 2.5，並使用表 4.1 的參數，來推理其整個摺疊演算法架構會使用到的記憶體基本元件正反器個數，即其面積成本函數推導應如下列公式所示：

$$r(f+1)L + (K-f)m + \text{ceil}(\log_2 f) + Kb \quad (4.1)$$

其中，若  $i = \text{ceil}(j)$ ,  $i \in \text{integer}$ ,  $j \in \text{real number}$ ，則  $i$  為最小整數大於或等於  $j$ 。

式子(4.1)中的  $L$  為乘加動作會使用到的中間運算暫存器的長度，而  $L$  需視規格規定的精確度來決定，且  $L$  亦會隨著  $m$ 、 $b$ 、 $K$  成長，在 Matlab 模擬中以全精確度來模擬  $L = m + b + \text{ceil}(\log_2 K)$ ，觀察圖 2.5 ( $r = 2, f = 3$ ) 中發現有 8 個中間運算暫存器，所以，估計它的暫存器檔案(RF)總共需要  $r(f+1)$  個。

而式子(4.1)中的  $(K-f)m$  為代表圖 2.5 上方的  $x$  輸入訊號多工取樣 (Input Sample Multiplexing) 行為會使用到的正反器個數， $(K-f)$  是因為只有第一級的  $x$  輸入訊號不需要卡  $f$  個輸入訊號暫存器，其他級在實現演算法上皆需卡輸入訊號暫存器； $\text{ceil}(\log_2 f)$  為表示控制單元會合成出的正反器個數；式子中最後的  $Kb$  為估計係數多工會使用到的正反器個數， $b$  為係數位元長度， $K$  為 FIR 的 taps 數。

### (b) 位元平面摺疊演算法[6]

摺疊架構圖請參考圖 2.8 及圖 2.9，並使用表 4.1 的參數，來推理其整個摺疊演算法架構會使用到的記憶體基本元件正反器個數，即其面積成本函數推導應如下列公式所示：

$$(2K+1)L + \text{ceil}(\log_2 b) + Kb \quad (4.2)$$



式子(4.2)中的  $L$  為表示中間運算暫存器的長度，而  $(2K+1)$  為估計其摺疊架構的暫存器檔案會使用到的  $L$  位元長度的暫存器個數， $2K+1$  的原因是每個位元處理器中估計要有  $\text{sum}$  及  $\text{carry}$  這兩個位元，而全部共有  $K$  個位元處理器，加上 1 個  $L$  位元長度移位暫存器來保存  $x$  端的輸入值； $\text{ceil}(\log_2 b)$  為估計控制單元所需正反器個數，因其摺疊因數為  $b$ ；式子最後的  $Kb$  為估計係數多工會使用到的正反器個數。

### (c) Parallel-In 摺疊演算法

摺疊架構圖請參考圖 4.1 ~ 圖 4.4，並使用表 4.1 的參數，來推理其整個摺疊演算法架構會使用到的記憶體基本元件正反器個數，即其面積成本函數推導應如下列公式所示：

$$KL + \text{ceil}(\log_2 f) + rb \quad (4.3)$$

其中， $L$  為每一個中間運算暫存器的位元長度， $K$  代表著 Parallel-In 摺疊演算法架構中的暫存器檔案所包含的暫存器個數，其詳情請參考圖 4.3，利用本篇論文在第三章提出的排程矩陣原理，讓中間暫存器個數達到最少的實現方式； $\text{ceil}(\log_2 f)$  代表 Parallel-In 摺疊演算法架構中的控制單元會合出的正反器個數；式子最後的  $rb$  為 Parallel-In 摺疊演算法架構係數多工會使用到的正反器個數。

### (d) Serial-In 摺疊演算法

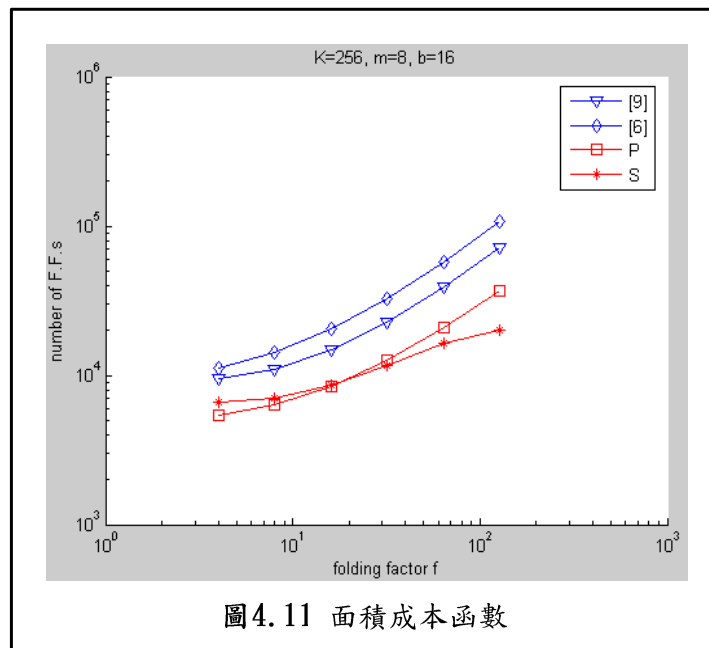
摺疊架構圖請參考圖 4.8 ~ 圖 4.10，並使用表 4.1 的參數，來推理其整個摺疊演算法架構會使用到的記憶體基本元件正反器個數，即其面積成本函數推導應如下列公式所示：

$$(r + (r - 1)f)L + \text{ceil}(\log_2 f) + rb + fm \quad (4.4)$$

其中， $L$  為每一個中間運算暫存器的位元長度， $(r + (r - 1)f)$  代表著 Serial-In 摺疊演算法架構中的暫存器檔案所包含的暫存器個數詳情見圖 4.9， $r$  為 MAC 中暫存器個數，而  $(r-1)f$  為 MAC 之間溝通暫存器個數； $\text{ceil}(\log_2 f)$  代表 Serial-In 摺疊演算法架構中的控制單元會合出的正反器個數； $rb$  為 Serial-In 摺疊演算法架構係數多工會使用到的正反器個數；式子最後的  $fm$  為  $x$  輸入端所需正反器個數。

### (e) 模擬面積成本函數成長趨勢

當我們固定  $K=256$ ， $m=8$ ， $b=16$  然後，使用 Matlab 來模擬面積成本函數(4.1) ~ (4.4)，進而來探討當摺疊因數  $f$  越大(即表示相對實現的基本運算單元(MAC)越少的情況下)，各種演算法其正反器個數的成長趨勢為何，模擬結果如圖 4.11 所示。觀察各種摺疊演算法的面積成本函數的成長趨勢，發現兩種新摺疊演算法在正反器個數的成長趨勢方面較現今摺疊演算法有明顯的優勢，而當摺疊因數  $f$  越大，Serial-In 演算法架構在面積節省上更加明顯。



### 4.3.2 功率消耗成本函數

最後，採用[18]這篇估計功率消耗的方法—其評估的方式是計算每次執行重複(Iteration)時使用到的暫存器總個數來做功率消耗的評估；還有功率消耗成本函數不做多工器的功率消耗評估，原因是它的功率消耗遠小於暫存器的功率消耗；另外，功率成本函數亦不將乘加器的功率消耗列入考慮，原因是假設彼此的摺疊演算法都是用同樣的技術來實現乘法器與加法器，所以其功率消耗應相同。

#### (a) 第一篇系統化摺疊演算法[9]

摺疊架構圖請參考圖 2.4 與圖 2.5，並使用表 4.1 的參數，來推理其整個摺疊架構在每次執行重複(Iteration)時使用到暫存器的總個數公式，即其功率消耗成本函數推導應如下列公式所示：

$$f(r(f+1)L + (K-f)m) + Kb \quad (4.5)$$

式子(4.5)中  $L$  為估算中間運算暫存器的位元長度，參考圖 2.5 發現一回合中總共需執行  $r(f+1)$  個中間運算暫存器；而  $(K-f)m$  為輸入訊號的多工取樣(Input Sample Multiplexing)行為會使用到的正反器個數次數；然而，想要完成一次重複(Iteration)需將上述的動作執行  $f$  回合；而式子最後的  $Kb$  為代表圖 2.5 的 MAC 下方的係數多工(Coefficient Multiplexing)行為會使用到的正反器個數， $b$  為係數長度， $K$  為 FIR 的 taps 數，而在  $f$  回合中總共執行  $K$  次。

#### (b) 位元平面摺疊方法[6]

摺疊架構圖請參考圖 2.8，並參考表 4.1 的參數，來推理其整個摺疊架構在每次執行重複(Iteration)時使用到暫存器的總個數公式，即其功率消耗成本函數應如下：

$$b( (2K+1)L + K ) \quad (4.6)$$

式子(4.6)中的(2K+1)為估計其摺疊架構的暫存器檔案會使用到的暫存器個數，而一個中間運算的暫存器長度是 L 位元；因為 b 為位元平面摺疊演算法的摺疊因數，所以執行一次完整重複( Iteration )，需將其摺疊架構全部的暫存器使用 b 次；而除摺疊架構外，會使用到記憶體的地方，還有每個位元處理器的長度為 1 位元的係數暫存器，而同時共有 K 個位元處理器一起運作，且完成一次完整重複( Iteration )總共需要執行 b 回合。

### (c) Parallel-In 摺疊演算法

摺疊架構圖請參考圖 4.1~圖 4.3，並使用表 4.1 的參數，來推理其整個摺疊架構在每次執行重複( Iteration )時使用到暫存器的總個數公式，即其功率消耗成本函數應如下：

$$f( 2rL + rb ) \quad (4.7)$$

其中，L 為中間運算暫存器的位元長度，且一回合中只有讀取特定 r 個暫存器內容以及寫入特定 r 個暫存器，所以將會執行中間運算暫存器 2r 次；b 為係數暫存器長度，而在一回中只會讀取 r 個係數暫存器，完成一次重複需執行 f 回合。

### (d) Serial-In 摺疊演算法

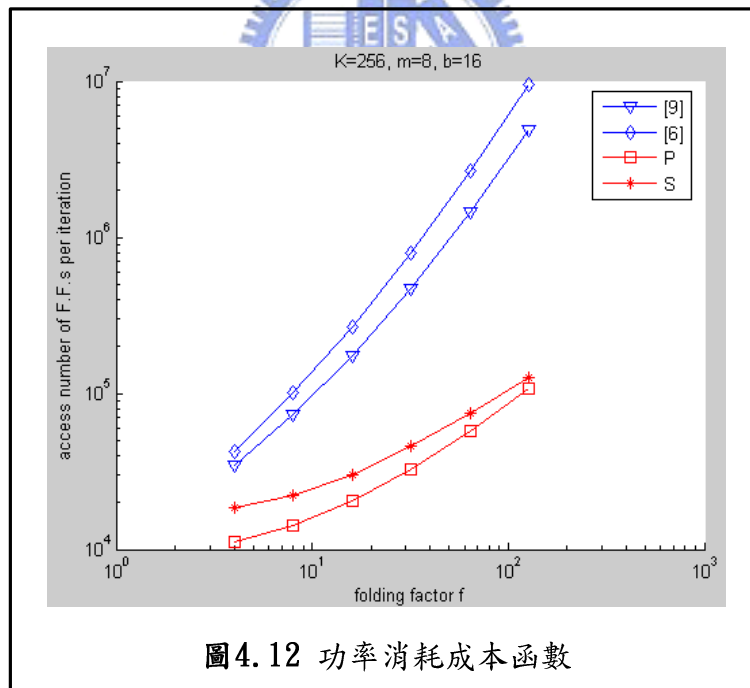
摺疊架構圖請參考圖 4.8 ~ 圖 4.10，並使用表 4.1 的參數，來推理其整個摺疊架構在每次執行重複( Iteration )時使用到暫存器的總個數公式，即其功率消耗成本函數應如下：

$$f(2rL + rb + (r-1)L + rm) \quad (4.8)$$

其中， $2rL+rb$  解釋見上一節(c)的說明； $(r-1)L$  為估計每一回合寫入 MA 之間溝通的 fD 暫存器次數；式子最後的  $rm$  為圖 4.9 中 x 輸入端 fD 讀取次數。

### (e) 模擬功率消耗成本函數成長趨勢

當我們固定  $K=256$ ， $m=8$ ， $b=16$  然後，使用 Matlab 來模擬成本函數 (4.5)~(4.8)，而來探討當摺疊因數  $f$  越大時(即  $r$  越小，使用越少基本運算處理器去實現摺疊架構)，哪個摺疊演算法的功率消耗成長趨勢較緩，模擬結果如圖 4.12 所示，觀察各種摺疊演算法的功率消耗成本函數的成長趨勢，發現當摺疊因數  $f$  越大時，Parallel-In 摺疊演算法在功率消耗成長趨勢方面有明顯優勢，而因為 Serial-In 摺疊演算法需使用到移位暫存器讓功率消耗大大地增加。



## 4.4 摺疊演算法實現 WCDMA 規格並比較

本章節將以實際應用規格 IS-95 WCDMA Pulse Shaping 33 taps FIR Filter 見表 4.2 來實現各種摺疊演算法架構及原始架構，並透過 SYNOPSIS 公司發展的 Design Compiler 與 PrimePower 軟體來實際模擬各種摺疊演算法架構及原始架構其面積大小與功率消耗情形來證明 4.3 節推導成本函數成長趨勢的可信度。

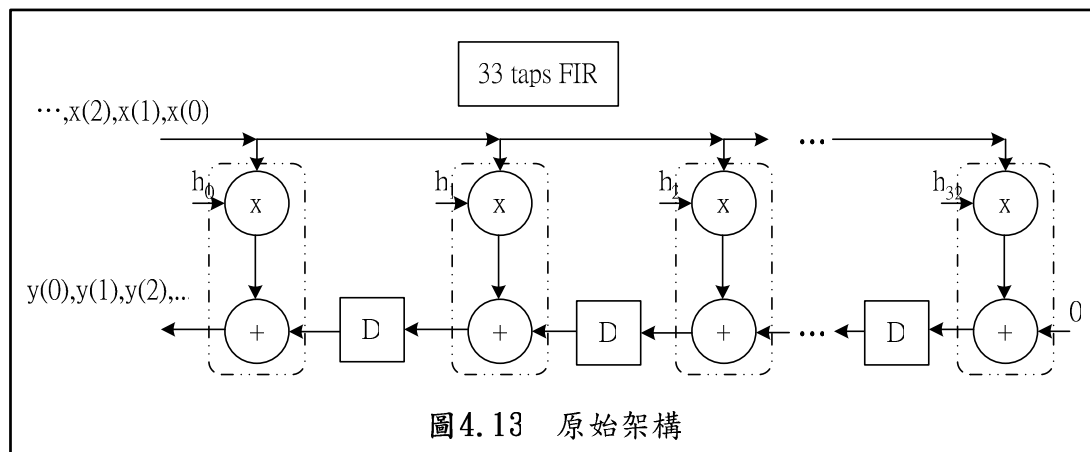
Filter Length :	33	taps
Passband Edge :	0.1	$\pi$
Stopband Edge :	0.28	$\pi$
Passband Ripple:	1.5	dB
Stopband Ripple:	40	dB
Input :	8	bits
Output :	18	bits
Throughput :	15.36	MSPS

表 4.2 WCDMA 規格



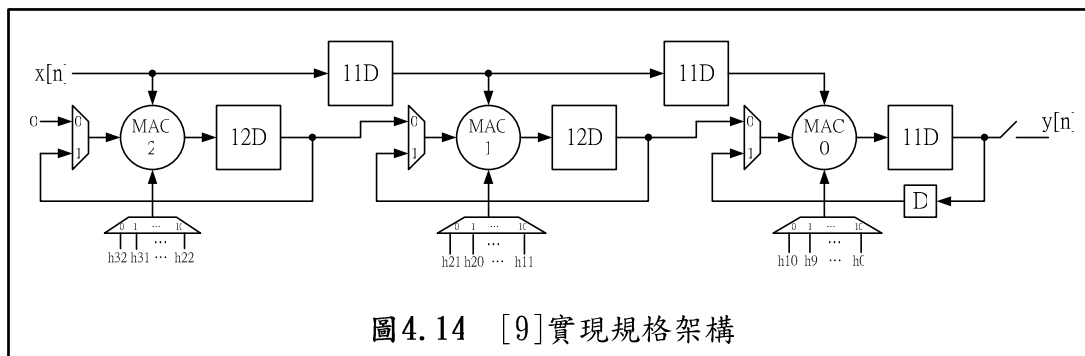
### 4.4.1 原始架構及各種摺疊架構實現

#### (a) 原始架構



以直接方式去實現表 4.2 所列 FIR 規格，見圖 4.13 原始架構由 33 個 MAC(將圖中⊗與⊕視為最小操作單元)並加上 33 個中間運算暫存器所組成。

(b) 第一篇系統化摺疊演算法[9]

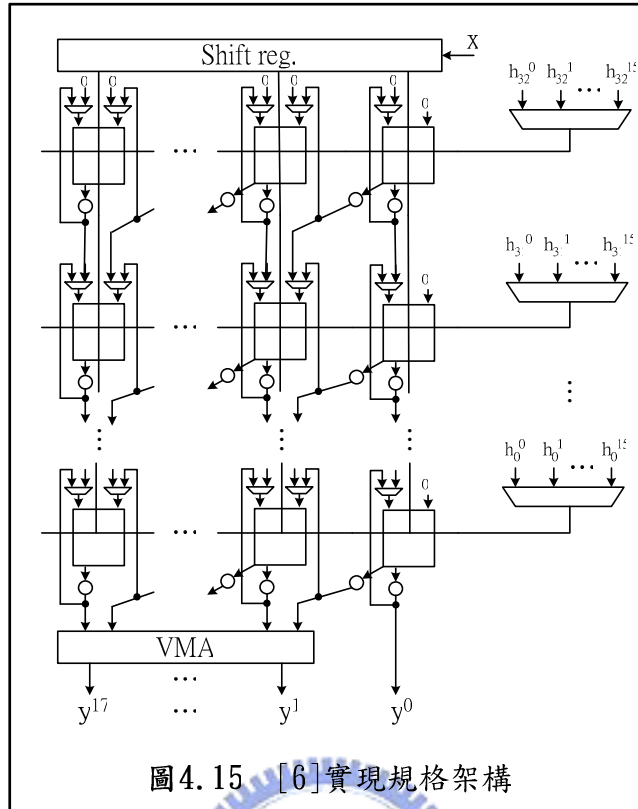


推導[9]的摺疊演算法應用在 IS-95 WCDMA Pulse Shaping 33 taps FIR Filter 的規格，摺疊架構應如上圖 4.14 所示，以 3 MACs 來實現 33 taps 的摺疊架構，操作簡介：MAC2 做係數  $h_{32}, h_{31}, \dots, h_{22}$  的乘加運算，將運算結果存在 MAC2 的輸出 12D 移位暫存器中，依時序切換多工器將運算結果加到下一級 MAC1；而圖中的 MAC1, MAC0 操作同理。

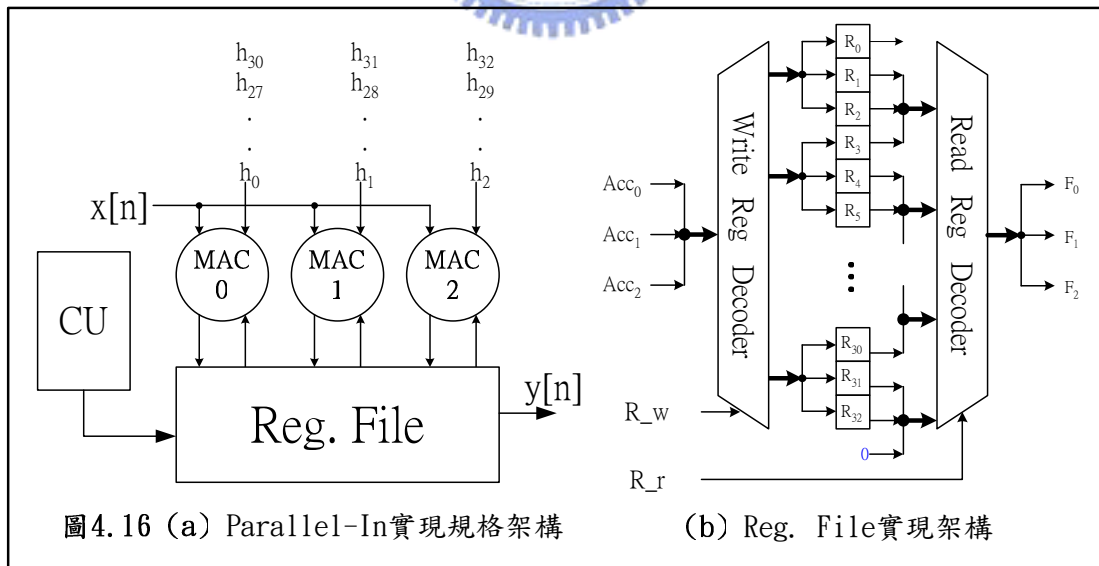
(c) 位元平面摺疊方法[6]

推導[6]的摺疊演算法應用在 IS-95 WCDMA Pulse Shaping 33 taps FIR Filter 的規格，摺疊示意架構見圖 2.8 所示，以 33 個位元處理器(圖 2.8 中虛線矩形為一位元處理器)來實現 33 taps 的摺疊架構，而每個位元處理器將重複執行  $b = 16$  ( $b$  為係數位元長度)回合後才做下一次重複(Iteration)；下圖 4.15 為[6]實現架構圖，圖中有 33 排位元處理器，33 個 16 位元係數暫存器及多工器，1 個  $x$  輸入移位暫存器，一個 VMA( Vector Merging Adder)，操作簡介：每一排位元處理器重複執行 16 回合完成一重複(Iteration)。





(d) Parallel-In 摺疊演算法

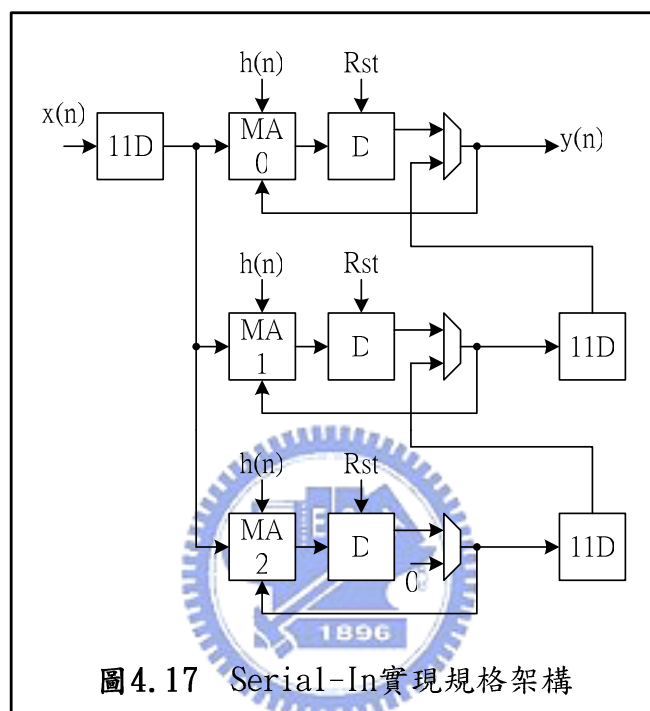


本篇提出符合規格的 Parallel-In 摺疊架構示意圖如上圖 4.16(a)所示，分成三區塊：一是處理器陣列(由 3 個 MACs 所組成)，二是暫存器檔案(33 reg.與 2 個位址解碼器組成)，三是控制單元；其暫存器檔案詳細架構如上圖 4.16(b)所示，由



33 reg.與 2 個位址解碼器(一個是寫入解碼器，另一是讀出解碼器)組成，因為處理器陣列為 3 MACs，所以寫入解碼器的輸入及讀出解碼器輸出端各有 3 條訊號線。

### (e) Serial-In 摺疊演算法




本篇提出符合規格的 Serial-In 摺疊架構示意圖如上圖 4.17 所示，由 3 MACs 加上 3 個 1D 暫存器與 2 個 11D 暫存器(作用為保留相鄰 MAC 的資料溝通)及 3 個多工器(作用為切換正確資料流)並在 x 輸入端加上 11D 的圓形定址暫存器，操作原理請參考 3.4 節。

## 4.4.2 軟體模擬面積大小及功率消耗情形

使用 SYNOPSIS 公司所發展的 Design Compiler 與 PrimePower 軟體來模擬 4.4.1 節實現 WCDMA 規格的原始架構與摺疊架構，其面積大小與功率消耗情形見表 4.3 與表 4.4。

	33 MAs	[9]	[6]	Parallel-In	Serial-In
Clock	15.36 M	15.36*11 =168.96 M	15.36*16 =245.76 M	15.36*11 =168.96 M	15.36*11 =168.96 M
Total	2,484,605	1,071,729	2,438,555	828,228	685,479
Net	1,884,177	802,064	1,906,297	639,150	524,116
MAC	438,207	63,318	148,500	63,318	63,318
Temp_Reg	105,699	115,308	229,680	105,699	80,075
coeff_Reg	56,529	56,529	56,529	5,139	5,139
CU	0	648	788	1,230	661
Mux (Temp_Reg)	0	2,574	59,400	12,741	2,694
X_Reg	0	18,952	3,902	0	9,476
Mux (coeff_Reg)	0	10,986	12,408	0	0
VMA	0	0	2,165	0	0

表4.3 Design Compiler模擬面積大小



	33 MAs	[9]	[6]	Parallel-In	Serial-In
Clock	15.36 M	15.36*11 =168.96 M	15.36*16 =245.76 M	15.36*11 =168.96 M	15.36*11 =168.96 M
Total(mW)	6.81	41.81	59.59	18.27	22.912
MAC	3.07	10.681	13.17	7.318	7.325
Temp_Reg	3.04	18.588	35.02	6.30	12.103
coeff_Reg	0.7	8.135	11.97	1.265	1.711
CU	0	0.221	0.160	0.074	0.301
Mux (Temp_Reg)	0	0.26	5.29e-02	3.313	0.16
X_Reg	0	2.735	1.258	0	1.312
Mux (coeff_Reg)	0	1.19	4.8e-02	0	0
VMA	0	0	6.2e-04	0	0

表4.4 PrimePower模擬功率消耗

### 4.4.3 結果分析

觀察表 4.3 發現在總面積(Total)比較上，Parallel-in 演算法與 Serial-in 演算法總面積最小，而[9]演算法總面積較大一些，[6]位元平面摺疊演算法總面積暴增幾乎與原始架構一樣大，在總面積的比較上與圖 4.11 面積成本函數預估的趨勢相當一致，不過，每種演算法總面積大小與面積成本函數是相對關係，因為繞線面積是軟體自動合成無法估測；而[6]的演算法由於是實現在位元平面上的摺疊，當 taps 與處理精確度(即位元長度)增加，便會使得位元處理器大大地增加，將會使得其繞線變得更加複雜，讓整體面積大量暴增，甚至可能高於原始架構。表 4.3 中 Temp\_Reg 為 FIR 中間運算的暫存器，Parallel-In 摺疊演算法讓其達到最佳(33 個)，並且利用規則化地定址大大地降低繞線面積(Net)。雖然高速(168.96MHz)處理的單一 MAC 面積比低速(15.36MHz)處理的 MAC 面積來的大，但其實大的量遠低於摺疊所省下來的面積(以 3 個高速 MAC 來節省 30 個低速 MAC 面積)。

觀察表 4.4 發現在總功率消耗(Total)的比較上，除原始架構外，Parallel-In 摺疊演算法的總功率消耗為最少，Serial-in 摺疊演算法第二少，[9]演算法次多，[6]最多，與圖 4.12 功率消耗成本函數預估的趨勢相當一致，不過，每種演算法總功率消耗與成本函數是相對關係。原始架構的功率消耗為最少，其原因為摺疊演算法本身是以高處理速度來換取硬體面積，所以在功率消耗方面會比原始架構要大，然而，就只比較摺疊演算法而言，Parallel-In 摺疊演算法的功率消耗又低的多，原因是 Parallel-In 摺疊演算法在讀取中間運算暫存器利用暫存器檔案中的解碼器讓每一回合中只讀寫 3 個暫存器，且同時其他 30 個暫存器並不動作；而[9]與 Serial-In 摺疊演算法的中間運算暫存器是使用移位暫存器來實現，相當地耗功率；而[6]摺疊架構由於其繞線太過於複雜導致其功率消耗大增。

摺疊演算法是一種以高處理速度來換取硬體面積的方法，所以犧牲了功率來換取面積，然而，在規格可允許的範圍內，而可以讓摺疊架構的面積盡量逼近最佳，且又不希望犧牲太多功率的情況下，提出了新摺疊演算法；最後，分析[6]

的摺疊演算法其作法主要是討論位元平面上的摺疊，並未考慮實際應用到電路上時會發生嚴重的繞線問題，所以其面積大小與功率消耗在各種摺疊演算法中皆是最差的。



---

---

## 第五章

---

---

### 設計流程與模擬結果

---

---



在這個章節中，將完整描述我們晶片設計的過程，從一開始的構思，到使用 MATLAB 軟體與 ModelSim 軟體來驗證構思，接著操作所需 CAD 軟體，執行功能驗證、邏輯閘合成、加入掃描鏈(Scan Chain，目的是為了可以做日後驗證設計中的正反器是否功能正常)、佈局繞線、估測功率、DRC 及 LVS 驗證，其流程圖如圖 5.1 所示，最後，附上模擬結果驗證第四章所提實現的架構為可工作的並符合預期結果。

#### 5.1 設計流程

第一步先熟悉 MATLAB 軟體所提供的濾波器設計工具箱，對有限脈衝濾波器有概念性的瞭解後，再利用 MATLAB 程式語言，完成 IS-95 WCDMA Pulse Shaping 33 taps FIR Filter 的功能驗證與並找到符合規格( Specification )的濾波器係數，規格列在表 4.2。

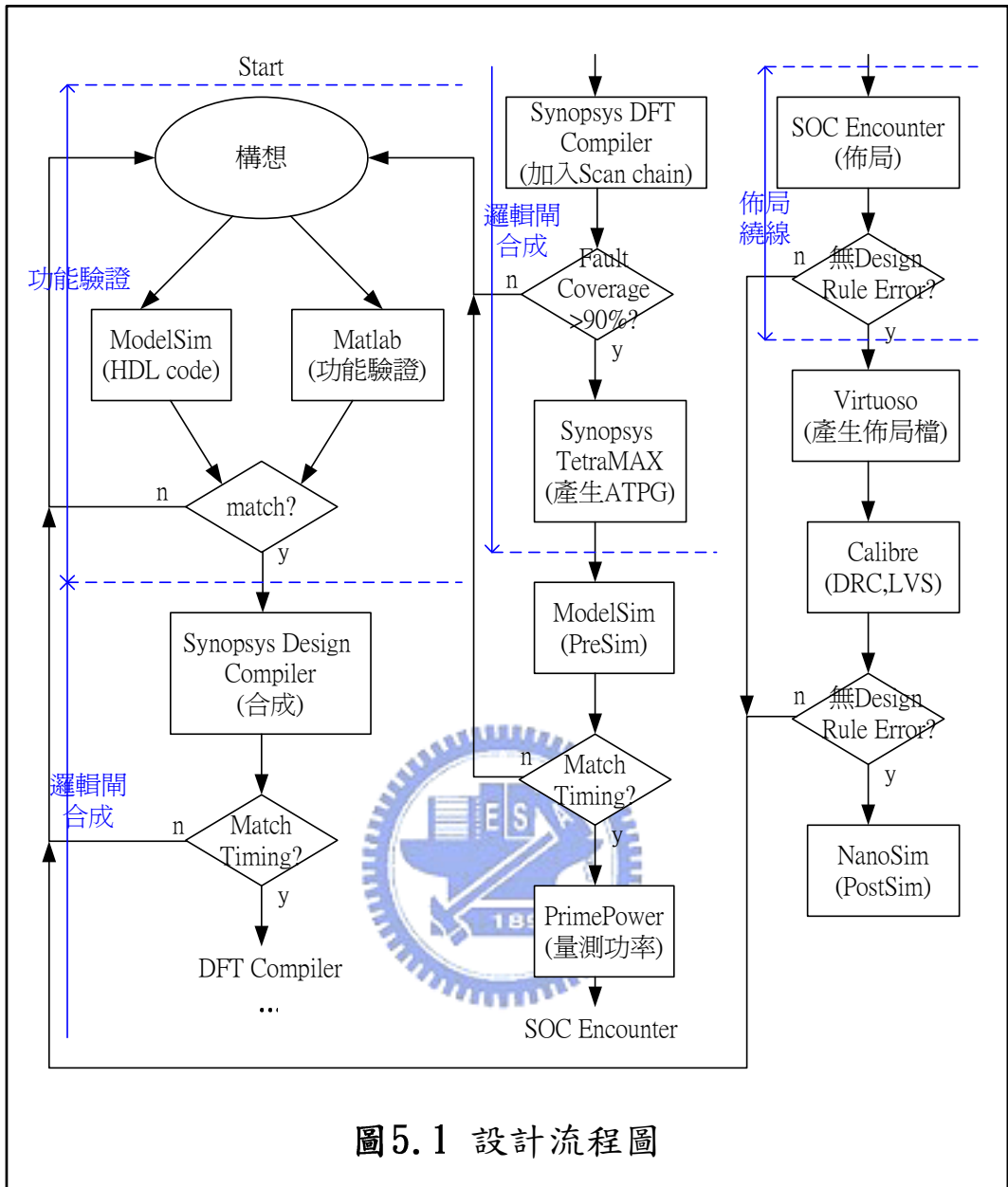


圖5.1 設計流程圖

## 5.2 功能驗證

首先，使用MATLAB找到符合IS-95 WCDMA Pulse Shaping 33 taps FIR Filter 規格的係數如下表5.1所示：

	Infinite Precision	Quantizing		Infinite Precision	Quantizing
n	h[n]	hq[n]	n	h[n]	hq[n]
0	-0.0062209	-62	17	0.1986	1986
1	-0.0032034	-32	18	0.1526	1526
2	0.00015872	2	19	0.09113	911
3	0.0062388	62	20	0.03101	310
4	0.012423	124	21	-0.013536	-135
5	0.014652	147	22	-0.035337	-353
6	0.0093662	94	23	-0.035566	-355
7	-0.0041004	-41	24	-0.02182	-218
8	-0.02182	-218	25	-0.0041004	-41
9	-0.035566	-356	26	0.0093662	94
10	-0.035337	-353	27	0.014652	147
11	-0.013536	-135	28	0.012423	124
12	0.03101	310	29	0.0062388	62
13	0.09113	911	30	0.00015872	2
14	0.1526	1526	31	-0.0032034	-32
15	0.1986	1986	32	-0.0062209	-62
16	0.21564	2156			

表5.1 符合規格量化係數

而該  $hq[n]$  的頻率響應圖如圖 5.2 所示，亦符合表 4.2 所列的規格。

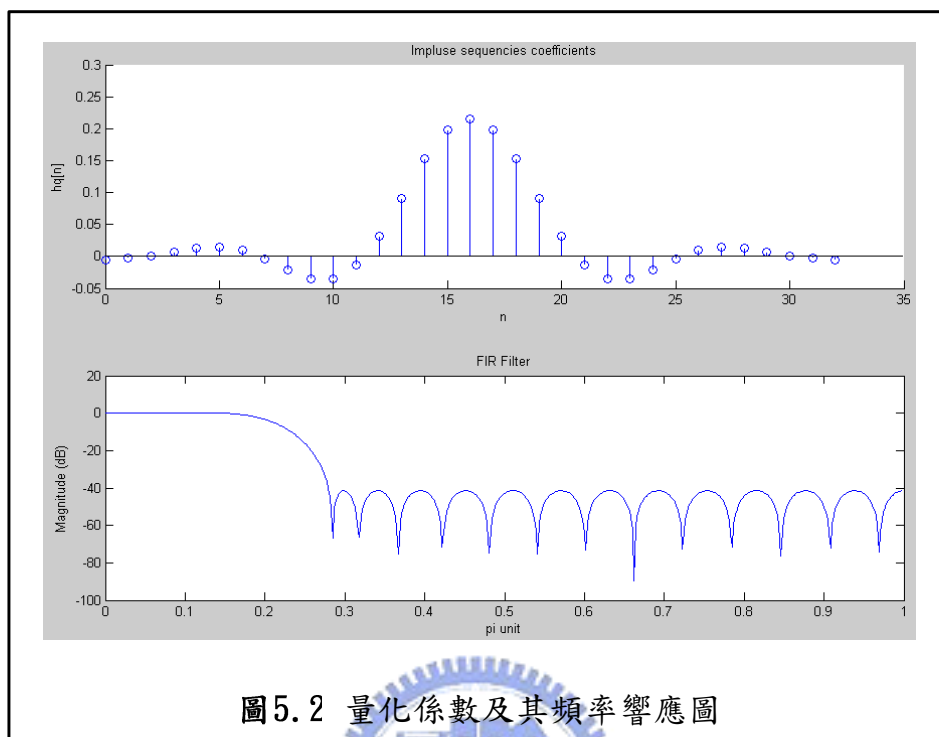


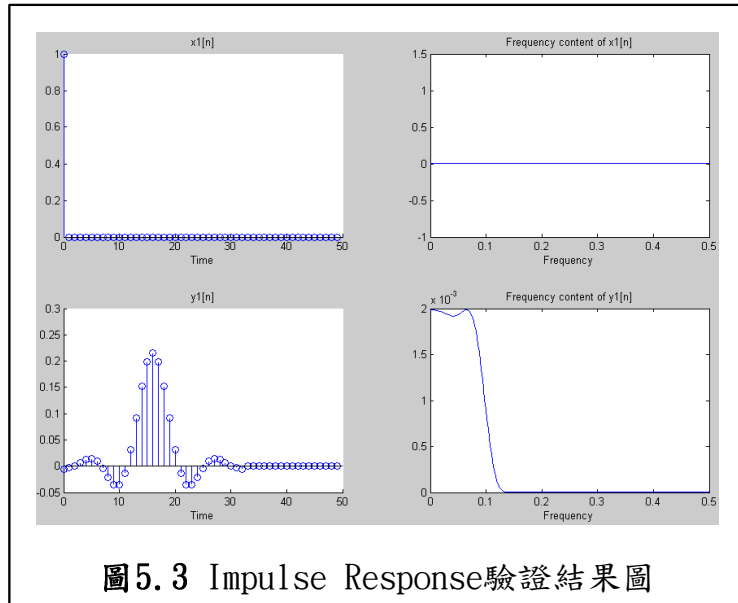
圖 5.2 量化係數及其頻率響應圖

濾波器功能驗證無誤後，根據第四章硬體架構的設計，使用VHDL硬體描述語言，在RTL層級描述此硬體架構，依照圖5.1的流程，使用ModelSim來做RTL層級的模擬輸出並與MATLAB計算結果互相比對驗證是否一樣，若不同則回去修改硬體描述語言，直到相符為止，以下展示驗證濾波器功能的實例—

#### ● 驗證實例一—脈衝響應(Impulse Response)

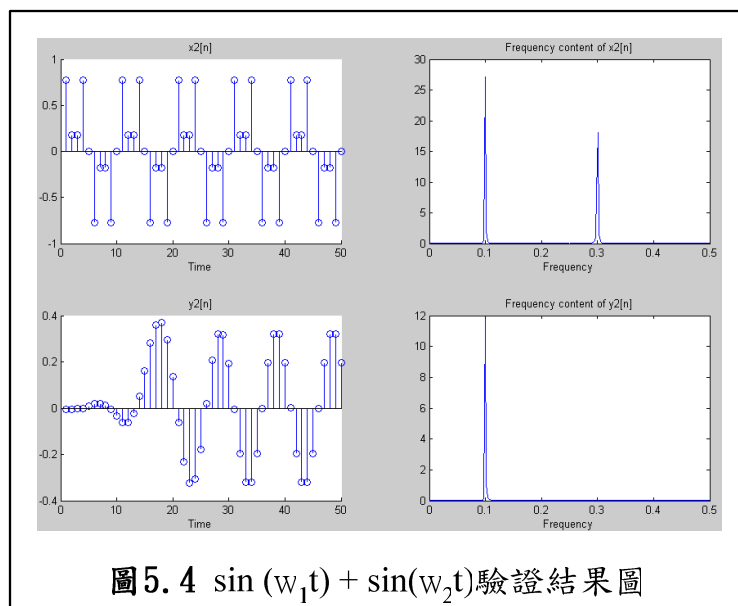
當輸入訊號 $x1[n]$ 為一脈衝訊號，通過該有限脈衝響應濾波器後，輸出訊號 $y1[n]$ 應等於 $hq[n]$ ，所以，ModelSim模擬RTL輸出結果應符合上述預期結果，截取ModelSim模擬結果畫在圖5.3中，發現ModelSim模擬RTL輸出結果符合預測結果與 $hq[n]$ 係數相等。





● **驗證實例二—弦波相加(  $\sin(\omega_1 t) + \sin(\omega_2 t)$  )**

當輸入訊號 $x_2[n]$ 為一不同頻率弦波相加的訊號(選擇 $\omega_1 < 0.28\pi$ ，而 $\omega_2 > 0.28\pi$ )，通過該有限脈衝響應濾波器後，分析輸出訊號 $y_2[n]$ 的頻譜應剩下 $\omega_1$ 而濾掉 $\omega_2$ ，所以，ModelSim模擬RTL輸出結果應符合上述預期結果，截取ModelSim模擬結果畫在圖5.4中，發現ModelSim模擬RTL輸出結果符合預測結果。



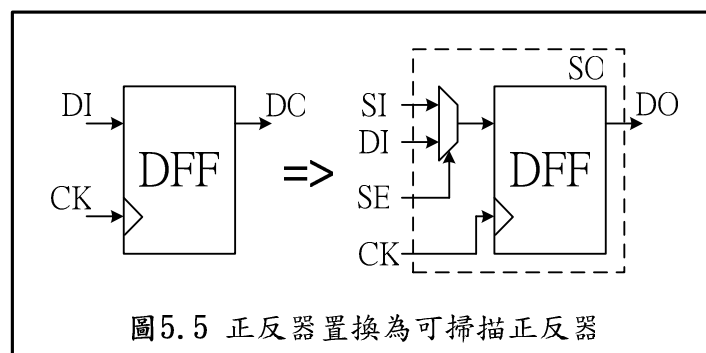
## 5.3 電路合成

本節將介紹設計流程圖5.1中邏輯閘合成部份，除了合成出第四章實現的硬體電路外，還使用SYNOPTYS公司所發展的軟體DFT Compiler軟體加入測試電路到設計中，將在以下小節中介紹加入測試電路的重要性，以及合成電路的相關資訊。

### 5.3.1 測試電路

隨著VLSI 設計之高密度化，一個晶片含有上百萬個電晶體是十分常見的情形。為了使晶片在生產後能夠具有容易測試的特性，所以在設計晶片時，需加入額外的測試電路於設計中，以幫助簡化晶片測試的困難度。一般而言，測試電路的加入技術可分成以下三種方法：Scan Path [21]、BIST [22]與Boundary Scan。

其中，Scan Path 就是在電路中額外加上SI ( Scan-in )、SE ( Scan-test )及SO ( Scan-out )腳，另外會在各個正反器( Flip-Flop )的輸入端插入適當的多工器如圖5.5所示；當處在測試模式時，令每個正反器連接成為移位暫存器的構造，將事先設計好的Test Patterns 從SI 輸入端依序位移到每一個正反器，之後切回正常模式並等待正反器、鎖住( Latch )組合電路的輸出結果，然後，再切回到測試模式將結果由SO 端輸出，並同時送入下一組Test Pattern，重複此動作測試完所有的Test Patterns。



BIST 則是“Built-In Self-Test”之縮寫，意即“內建自我測試”，也就是在晶片裡面內建樣本產生的電路和檢查結果的電路，使晶片本身具有自我測試的功能。最後，Boundary Scan 則是將多個VLSI晶片組裝在一片基板上同時測試的技術，目前已成為IEEE 的標準化規格。

由 SYNOPSIS 公司所發展的軟體 DFT Compiler 軟體，是一套整合在 Design Analyzer 中的一個功能(Function)，主要是協助設計者在電路中自動加入測試電路，包含有掃描鏈(Scan-Chain)的合成。設計者只要下指令就能產生如圖 5.5 的具有掃描(Scan)功能的正反器，並自動將全部置換後的正反器連結成移位暫存器結構，並有 Autofix 工具可自動修改違反測試規則的部份，最後並提供錯誤涵蓋率(Fault Coverage)的報告，若使用者滿意此時的錯誤涵蓋率後，便可以將設計交由 TetraMAX 軟體自動產生 Test Patterns。

由 SYNOPSIS 公司所發展的軟體 TetraMAX，是一套可產生 Test Patterns 及錯誤模擬(Fault Simulation)的軟體。使用者在操作此套軟體之前，需先準備好已加入掃描鏈的硬體描述語言程式碼檔，及 SPF(STIL Protocol File)檔，輸入到軟體中，即可對使用者電路的掃描鏈自動產生 Test Patterns。亦或可以把已存在的 Functional Pattern 輸入，用以求得該 Pattern 的錯誤涵蓋率。

表5.2是採用本篇論文第四章的硬體實現架構透過Design Analysis合成後，再由DFT Compiler自動產生加入測試電路於設計中，最後使用TetraMAX所做的測試涵蓋率評估。下列介紹表5.2中Fault Coverage與Test Coverage的定義，其中posdet\_credit內定值為50%，au\_credit內定值為0%。

$$\text{Fault Coverage} = \frac{\text{number of detected faults}}{\text{total number of faults}} \leq 1$$

$$\text{Test Coverage} = \frac{\text{DT} + (\text{PT} * \text{posdet\_credit})}{\text{all faults} - (\text{UD} + (\text{AU} * \text{au\_credit}))}$$

Uncollapsed Stuck Fault Summary Report		
fault class	code	#faults
Detected	DT	37807
Possibly detected	PT	310
Undetectable	UD	108
ATPG untestable	AU	518
Not detected	ND	1563
total faults		40306
test coverage		94.44%
fault coverage		94.18%
Pattern Summary Report		
#internal patterns		92
#basic_scan patterns		92
CPU Usage Summary Report		
Total CPU time		27.86

表5.2 測試涵蓋率評估

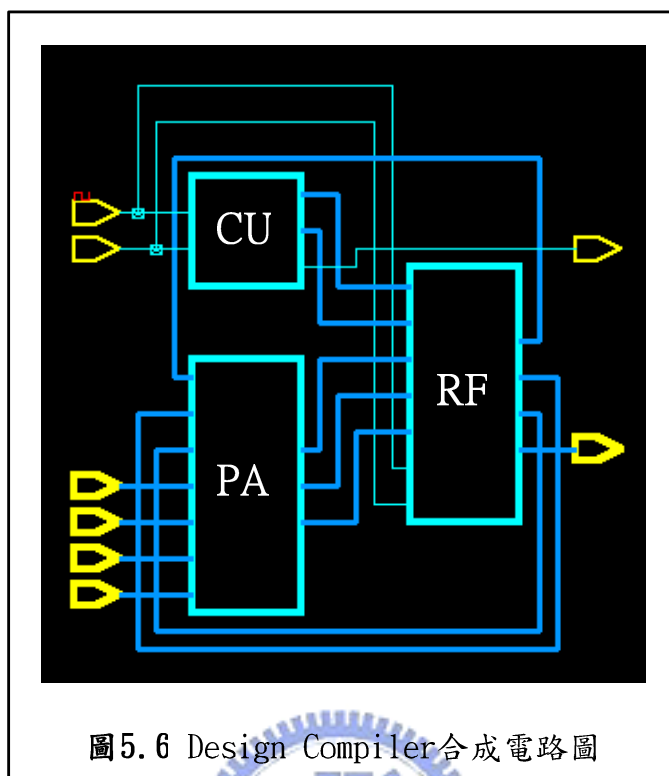


### 5.3.2 邏輯閘合成

由 SYNOPSIS 公司所發展的 Design Compiler 軟體，主要是用來執行邏輯合成(Logic Synthesis)工作，在 Cell-Based Flow 各個軟體工具中扮演著非常重要的角色，透過 Design Compiler 可以將所寫好的 RTL Verilog 或是 VHDL 程式碼轉換成閘層級電路描述檔(Gate-Level Netlist)，此外還可以搭配現有的 DesignWave Library 來完成設計，以及設定限制( Constraints )來達成效能最佳化(Performance Optimization)。

利用 Design Compiler 軟體合成電路，藉由改變設計的條件與限制，讓合成軟體編譯出最佳的閘層級電路描述檔，接著做考慮閘延遲時間的閘層級電路模擬，重複以上模擬比對步驟至符合設計規格，到這裡完成前段晶片設計，軟體合成電路見圖5.6。而表5.3列出使用 Design Compiler 軟體並採用 UMC 0.18um 製程且操作

在clock = 168.96 MHz所合成出的各單元細部電路面積統計表。



Unit	Area(SYNOPSIS)	Percentage of Chip
RF	Reg (33)	105,699
	Decoder	18,949
	Total	124,648
PA	MAC (3)	63,318
	CU	1,230
	Net	639,150
Total	828,228	100%

表5.3 各單元面積統計

## 5.4 晶片規格與佈局設計

下線晶片的完整規格列表，如表 5.4 所示，我們是採用 UMC 0.18 $\mu$ m 1P6M 製程，並使用圖 5.1 的設計流程圖來完成晶片設計與驗證，而自動化佈局設計是使用 Cadence 公司最新釋出的 RTL-to-GDSII 產品—SOC Encounter，圖 5.7 為使用 SOC Encounter 所產生的晶片佈局圖，表 5.5 顯示晶片 pad 使用狀況。

Parameters	Value	Parameters	Value
Technology	UMC 0.18 um	Package	68-pin LCC
Voltage	1.8 V	Pin Count	67
x[n] Precision	8 bits	Die Size	1411 x 1411 um <sup>2</sup>
h[n] Precision	16 bits	Core Size	861 x 861 um <sup>2</sup>
y[n] Precision	18 bits	Throughput	15.36 MSPS
Test Scan	5 bits	Power	18.27 mW

表5.4 晶片規格列表

PAD Type	Usage	PAD Count
Input PAD	Clock ,Reset ,Xn ,Hn , Scan_in ,Scan_enable	31
Output PAD	Yn ,Valid ,Scan_out	20
Core Power	CoreVDD ,CoreGND	8
IO Power	IOVDD ,IOGND	8

表5.5 PAD使用列表

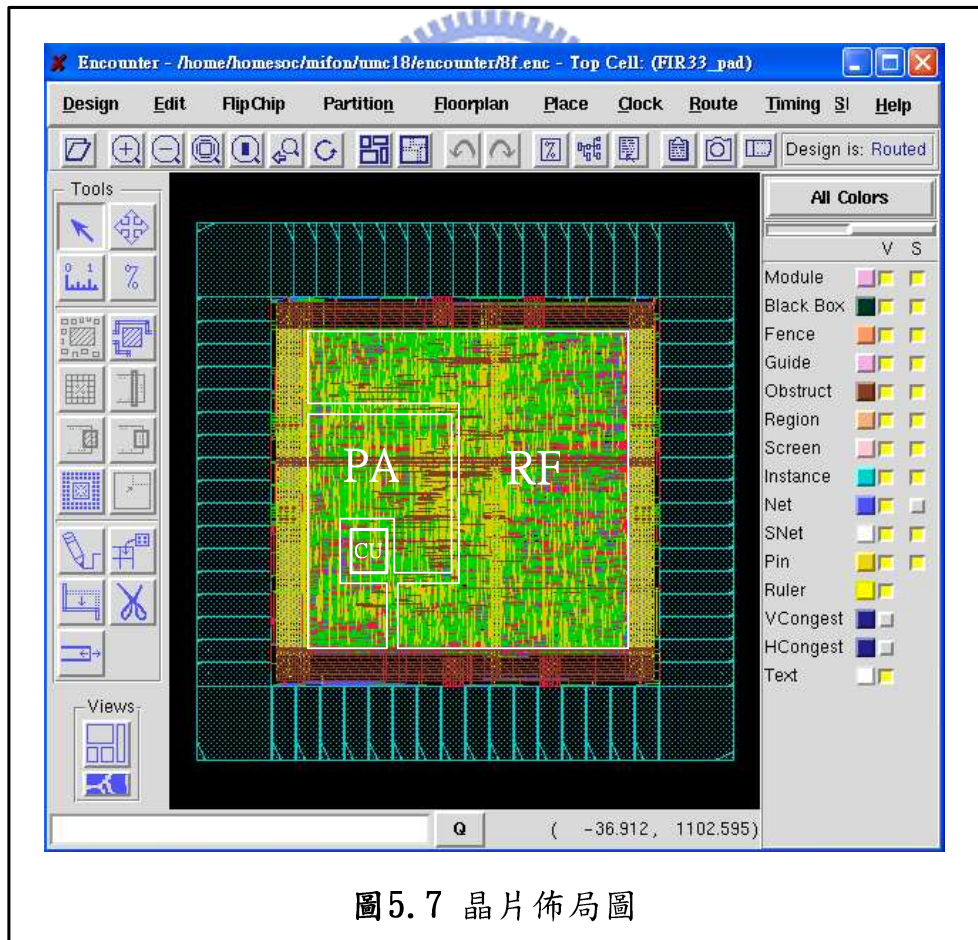


圖5.7 晶片佈局圖



## 5.5 DRC&LVS 驗證與佈局後模擬

Calibre 為 Mentor Graphics 公司所有之電路佈局驗證軟體，其功能包含電路佈局之 DRC、LVS 驗證，能幫助設計者在下線之前，快速的檢查所設計的電路佈局是否完全符合晶圓廠的要求，並可驗證數位、類比積體電路佈局的設計。在所支援的電路佈局軟體方面，包含 Cadence 公司的 Virtuoso、SpringSoft 公司的 Laker 以及 Mentor Graphics 公司的 Calibre DESIGNrev 等，以提供設計者完整修改電路佈局的環境。

我們在後段佈局設計與驗證階段，是使用 Mentor Graphics 公司所有之電路佈局驗證軟體－Calibre tool，來完成晶片自動化佈局與繞線設計後的驗證工作。當使用 SOC Encounter 軟體完成 5.4 節晶片自動化佈局與繞線設計之後，再透過使用 Cadence 公司的 Virtuoso 軟體產生最後的佈局檔 (GDS-II)，最後經過 Calibre 軟體做 DRC 與 LVS 驗證正確見圖 5.8 與圖 5.9，佈局前後驗證見圖 5.10，發現佈局後 yn 輸出只有受閘延滯 (Gate Delay) 影響，不影響 yn 值的正確性。

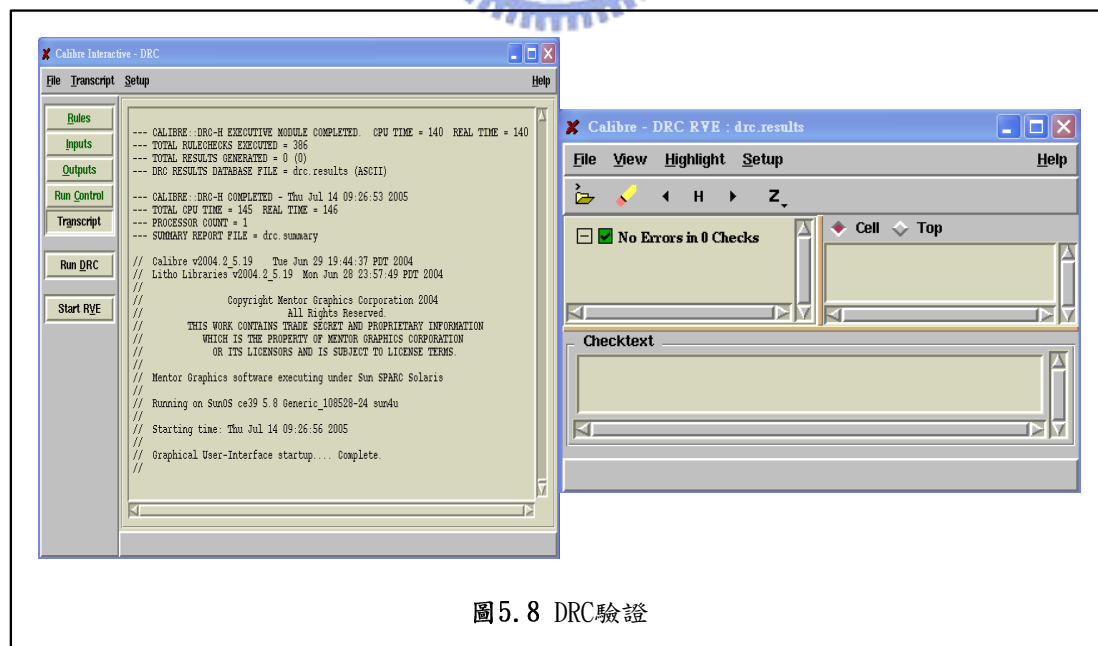


圖 5.8 DRC 驗證





---

---

# 第六章

---

---

## 結論

---

---



### 6.1 主要貢獻

本篇論文提出一套硬體架構之摺疊演算法，此演算法可應用於各種採用基本運作單元(PE)所構成之陣列架構，而其具備可重複使用的特性，特別適合用在陣列的基本運作單元個數會因規格而變動的應用上。將此摺疊演算法套用在有限脈衝響應濾波器設計上，使其成為一套可重複規劃之摺疊濾波器架構，在設計規格可以允許的範圍內可以加快資料處理速度，而且也減少了運算單元使用個數與整體架構的面積，使硬體具有較高的使用率。

本論文設計了有限脈衝響應濾波器摺疊架構與晶片實現，首先，在濾波器摺疊架構的研究中，我們提出了Parallel-In與Serial-In摺疊演算法架構設計，利用適當的時序排程和暫存器配置的方式，將電路中的暫存器數目進行最佳化縮減，進

而使能夠有效地降低硬體面積；此外，我們也比較其他摺疊演算法的架構，我們的新摺疊演算法架構是經過有系統化且又規則化地定址暫存器，讓整體摺疊架構複雜度大大降低，進而達到大量地降低繞線面積的目的。另一優點是，這個演算法架構具有相當的規則性與擴充性，可以容易地更改架構，以應用在不同濾波器長度與更高階層數的處理。

## 6.2 未來展望

### ● 降低功率消耗



本篇設計的摺疊架構中，在每回合的最後一個時間點難免會有一些運算單元是不必運算的，但架構中沒有關閉運算器計算的機制，導致功率的消耗，故可以在硬體設計中加入運算器自動調整計算開啟或關閉的切換開關，以節省電力的浪費。

### ● 非整除摺疊架構

在 3.3.2 節中有簡單地介紹非整除摺疊架構的排程矩陣的排法，這只是其中一種作法，雖然可以很簡單地去實現非整除摺疊架構，但在最後一回合中會出現幾個乘加器是不做工的，未能加以利用造成浪費，此時，醞釀另一非整除摺疊排程矩陣的構想，將摺疊工作回合數(即  $f$  回合)擴展成  $K$  taps 數與  $r$  MACs 數的最小公倍數，便能解決上述問題，未來能繼續朝此方向來改善並研究更適合的排程方法。

## ● 發展具備使用者介面之 IP 合成器

本篇論文提出的新摺疊演算法架構具有相當的規則性，可以發展一套參數輸入介面即(taps 數, MACs 數,  $x$  輸入位元長度,  $h$  係數位元長度, 規格要求精確度) =  $(K, r, m, b, L)$ ，讓軟體自動產生 HDL code 來實現符合規格要求硬體，增加摺疊演算法面積大小與功率消耗選擇的彈性，以期在規格允許範圍內及最短時間內能有最佳效能表現。



## 參考文獻

- [1] C. S. Burrus , “Digital filter structures described by Distributed Arithmetic ,”  
IEEE Trans. on Circuits Syst., pp. 674–680, Dec.1977.
- [2] T. C. Denk and K. K. Parhi, “Synthesis of folded pipelined architectures for  
multirate DSP algorithms, ” IEEE Trans. On VLSI , vol. 6, no. 4, pp. 595–607,  
Dec. 1998.
- [3] S.-F. Lin, S.-C. Huang, F.-S. Yang, C.-W. Ku, and L.-G. Chen, “Power-efficient  
FIR filter architecture design for wireless embedded System,” IEEE Trans. on  
Circuits Syst. II, vol. 51, no. 1, pp. 21–25, Jan. 2004.
- [4] E. Lueder, “Generation of equivalent block parallel digital filters and algorithms  
by a Linear Transformation,” in IEEE Int. Symp. on Circuits and Systems, May  
1993, pp. 495–498.
- [5] M. Mehendale and S. D. Sherlekar, VLSI synthesis of DSP Kernels-algorithmic  
and architectural transformations. KLUWER ACADEMIC PUBLISHERS, 2001.
- [6] I. Milentijevic, V. Ciric, T. Tokic, and O. Vojinovic, “Folded bit-plane FIR filter  
architecture with changeable folding factor ,”in IEEE Euromicro Sym. on Digital  
System Design, Sept. 2002, pp. 45–52.
- [7] Z. J. Mou and P. Duhamel, “Short-length FIR filters and their use in fast  
nonrecursive filtering,” IEEE Trans. on Signal Processing, vol. 39, no. 6, pp.  
1322–1332, June 1991.
- [8] A. V. Oppenheim and R. W. Schaffer, Discrete-time signal processing, 2nd ed.  
PRENTICE HALL, 1999.

- [9] K. K. Parhi, C.-Y. Wang, and A. P. Brown, "Synthesis of control circuits in folded pipelined DSP architecture," *IEEE Journal of Solid-State Circuits*, vol. 27, no. 1, pp. 29–43, Jan. 1992.
- [10] D. A. Parker and K. K. Parhi, "Low-area/power parallel FIR digital filter implementations," *J. VLSI Signal Processing Syst.*, vol. 17, no. 1, pp. 75–92, 1997.
- [11] D. N. Pearson and K. K. Parhi, "Low-power FIR digital filter architectures," in *IEEE Int. Symp. on Circuits and Systems*, Apr. 1995, pp. 231–234.
- [12] S. White, "Applications of distributed arithmetic to digital signal processing: A tutorial review," *IEEE ASSP Magazine*, pp.4–19, July 1989.
- [13] A.V. Oppenheim and R.W. Schaffer, *Discrete-Time Signal Processing*, Prentice Hall, Inc., 1989.
- [14] S. Hauck, "Asynchronous design methodologies: an overview," *Proc. IEEE*, vol. 83, no. 1, pp. 69–93, Jan. 1995.
- [15] W. Kuang, J. S. Yuan, R. F. DeMara, D. Ferguson, and M. Hagedorn, "A delay-insensitive FIR Filter for DSP applications," in *Proc. 9th Annu. NASA Symp. VLSI Design*, Albuquerque, NM, Nov. 2000, pp. 135–165.
- [16] Khaled M. Elleithy and Magdy A. Bayoumi, "A systolic architecture for modulo multiplication," *IEEE Transactions on Circuits and Systems-II: Analog and Digital Signal Processing*, Vol. 42, No. 11, pp. 725-729, November 1995.
- [17] Chin-Liang Wang and Jung-Lung Lin, "Systolic array implementation of multipliers for finite field  $GF(2^m)$ ," *IEEE Transactions on Circuits and Systems*,

Vol. 38, No. 7, pp. 796-800, July 1991.

- [18]R. Kleihorst and A. V. D. Avoird, "Power analysis of a general convolution algorithm mapped on a linear processor array," *Journal of VLSI Signal Processing*, vol. 37, pp. 5–19, May 2004.
- [19]Keshab K. Parhi, "VLSI digital signal processing systems design and implementation," John Wiley & Sons, Inc. published, 1995
- [20]Vijay K. Madiseti, "VLSI digital signal processors." Butterworth-Heinemann published, 1995
- [21]K. D. Wanger, "Robust scan-based logic test in VDSM technologies," *IEEE Computer*, pp. 66-74, Nov. 1999.
- [22]H. T. Nagle, S. C. Roy, C. F. Hawkins, M. G. Mcnamer, and R. R. Fritzemeier, "Design for testability and build-in self test: a review," *IEEE Trans. Industrial Electronics*, vol. 36, no. 2, pp. 129-140, 1989.
- [23]Lan-Rong Dung, Yen-Lin Lee, and Chun-Ming Wu, 2001, July/October, "A Reconfigurable Architecture for DSP System-on-Chip," *Canadian Journal of Electrical and Computer Engineering*, pp.109-113 (NSC 89-2215-E-009-119-)
- [24]Shiuh-Rong Huang and Lan-Rong Dung, 2002, "VLSI Implementation for MAC-Level DWT Architecture," *ISVLSI 2002* (NSC 90-2215-E-009-083-)
- [25]Shiuh-Rong Huang and Lan-Rong Dung, 2002, "A MAC-Level Synthesis of Resource-Constrained DWT SIP," the 13th VLSI Design/CAD Symposium (NSC 90-2215-E-009-083-)