

3-0 簡介

上一章是專門介紹快速傅立葉轉換之數學演算法，在這一章會將數學演算法直接對映到硬體架構，因此這一章的重心放在硬體層面。本章將使用歸納法將現行 FFT 技術分為二類：

1. 垂直投影架構 (*Row major system*)
2. 水平投影架構 (*Column major system*)

3-1 快速傅立葉轉換之架構分類

從 Cooley-Tukey algorithm 所推導出來的 FFT，其蝴蝶圖 (butterfly diagram) 如 Fig3-1.1 所示。

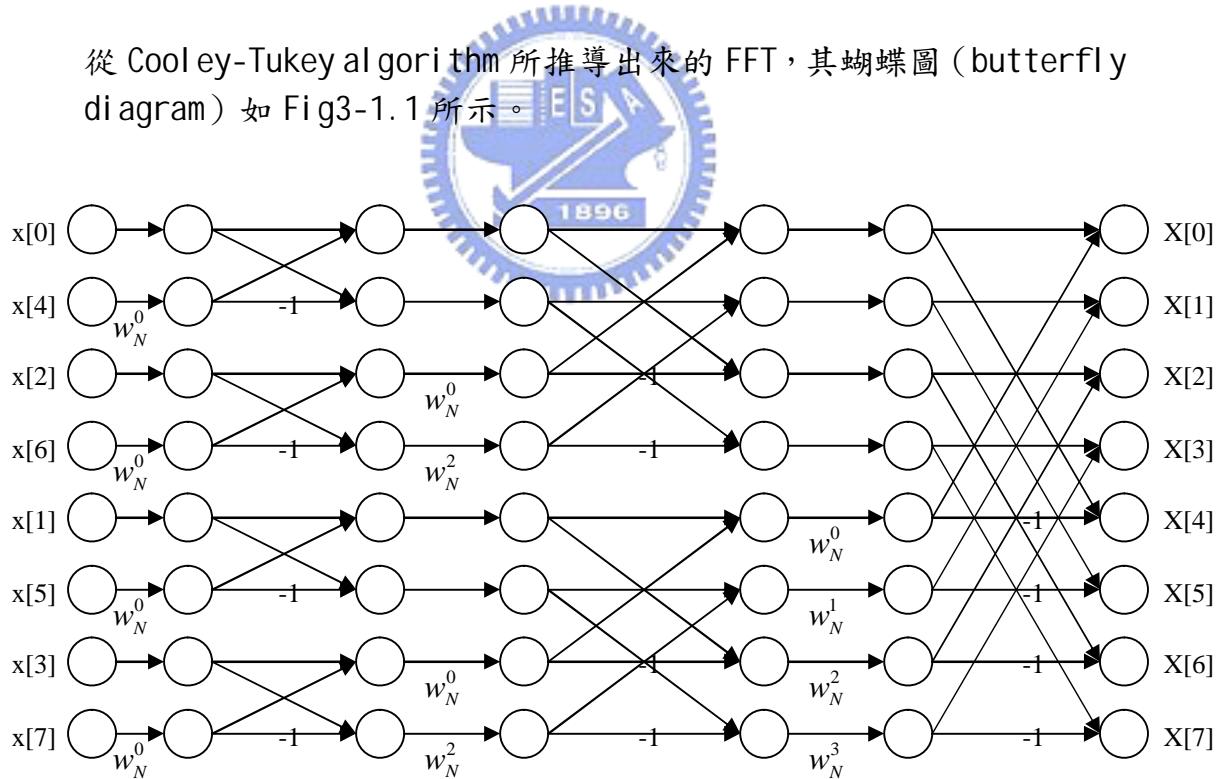


Fig 3-1.1 8-point FFT with radix-2

假設光線由上方向下垂直照射，可得到橫向之 1-D 處理器架構，這就是垂直投影的基本模式 (row major system)，如 Fig3-1.2 所示：

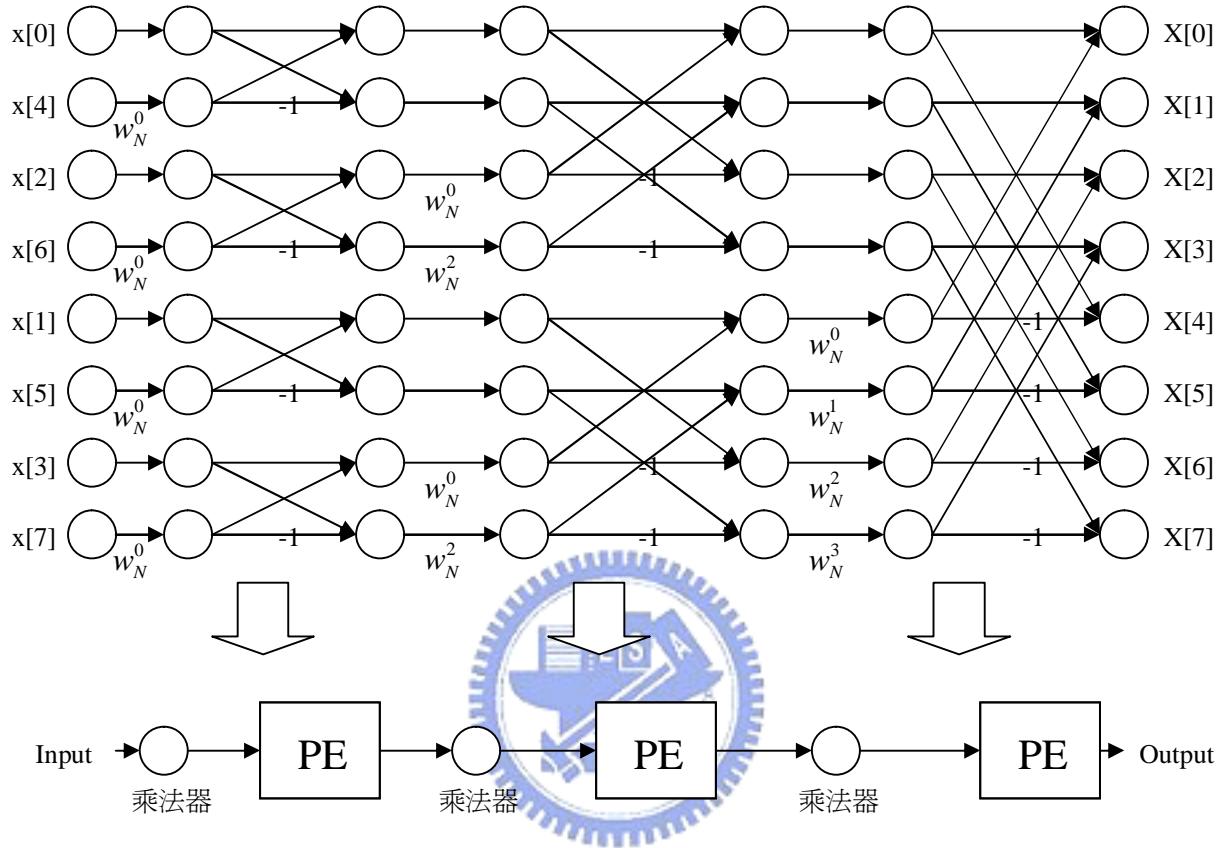


Fig3-1.2 FFT 垂直投影架構

使用相同方法，假設光線由左方向右方水平照射，可得到垂直的 1-D 處理器架構，這就是水平投影的基本模式 (column major system)，如 Fig 3-1.3 所示：

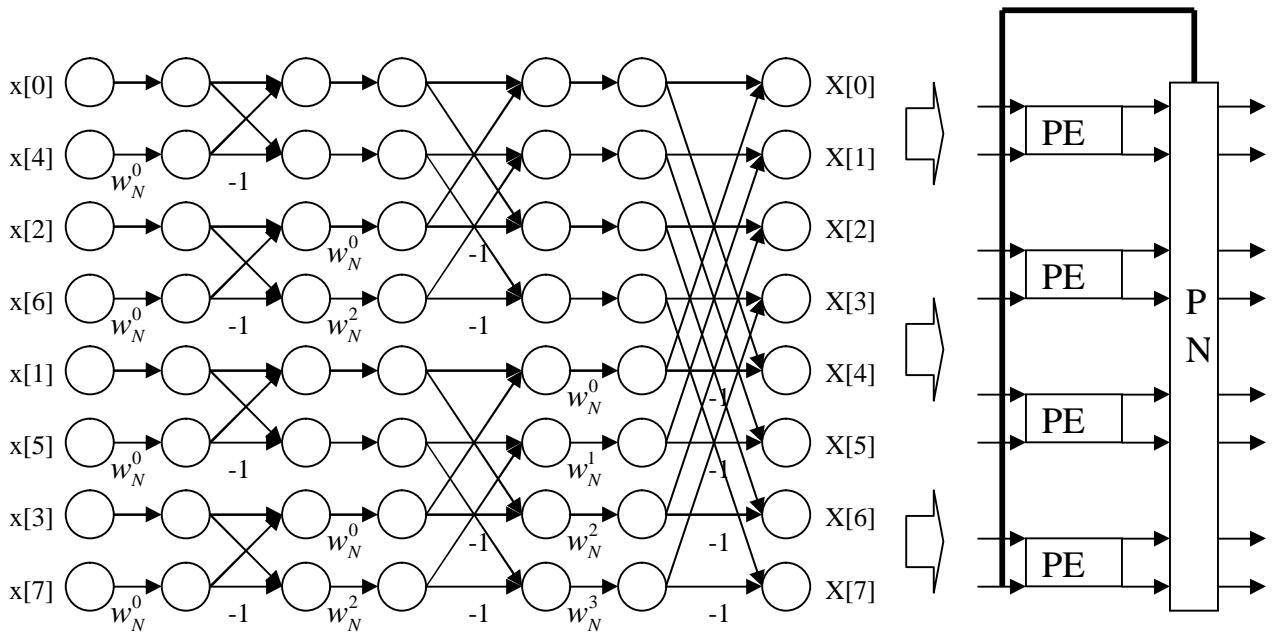


Fig 3-1.3 FFT 水平投影架構

1. 垂直投影架構 (row major system)

垂直投影架構是將 butterfly 作垂直投影所獲得之架構，其好處在於可用較少之處理器及管線技術 (pipeline technique) 來作 FFT 運算，因此在硬體及速度相當有效率，而且因為有可變性 (variable) 及擴充性 (scalable) 之特點，在商業上具有競爭力，常被用於電子產品之技術開發。其硬體只需 $\log N$ 個處理器 (PE) 即可 (R 代表使用的 radix, N 代表輸入資料的總點數)，因此非常節省硬體。

2. 水平投影架構 (column major system)

水平投影架構和垂直投影架構的產生方式相當類似，差別在於投影方向是水平。其所需的硬體為 N/R 個處理器 (PE)，最主要的特色是計算時，是一整排 PE 同時計算，在每一排 butterfly 計算好後，輸出資料會透過交換網路 (PN) 進行資料順序之跟越式交換處理 (stride permutation)，處理好的資料會被送到記憶體或暫存器 (register) 儲存。下一次 butterfly 會從記憶體或暫存器 (register) 讀出資料，將正確的資料順序送到 butterfly PE 進行計算，一樣的動作重複 $\log N$ 次，即可完成一次完整的 FFT 運算。這個架構和其他架構最大的不同在於交換網路 (permutation network) 的使用。

	Throughput time	Processor (area)	Area*Time
Row major system	N/R	$\log_2 N$	$N/R \log_2 N$
Column major system	$\log_2 N$	N/R	$N/R \log_2 N$

TABLE 3-1.1 垂直投影及水平投影架構的比較

- 註：1. 輸出時間（Throughput time）以多處理器架構為計算基本單位。
 2. 硬體面積（area）以垂直兼水平架構（Single processor system）為計算基本單位。

由 TABLE 3-1.1 可知，不同種類之面積時間乘數（area*time）是一樣的，因此在設計硬體架構時，可以適當地選擇速度導向，功率導向或是面積導向，再於其中作評估（trade-off）以獲得最好的使用效率。

3-2 垂直投影架構（Row major system）

垂直投影架構是將 butterfly 由上向下投影的架構，每一級（stage）由單一處理器負責運算，運算好的資料再送到下一個處理器作運算，由於資料是一級串著一級（stage by stage），因此這種架構在 VLSI 的實現，很適合加入管線技術（pipeline），速度因此可以提升。

我們大概可以將垂直投影架構細分為下列幾種：

1. 單一路徑延遲迴授系統（SDF）
2. 多路徑延遲交換線路系統（MDC）

以下各節將會詳細介紹這幾種架構。

3-2.1 單一路徑延遲迴授系統（SDF）

單一路徑延遲迴授系統（single path delay feedback system）的架構主要是由蝴蝶器（butterfly），暫存器（register），及轉動因數乘法器（twiddle factor multiplier）所構成，其系統如 Fig3-2.1：

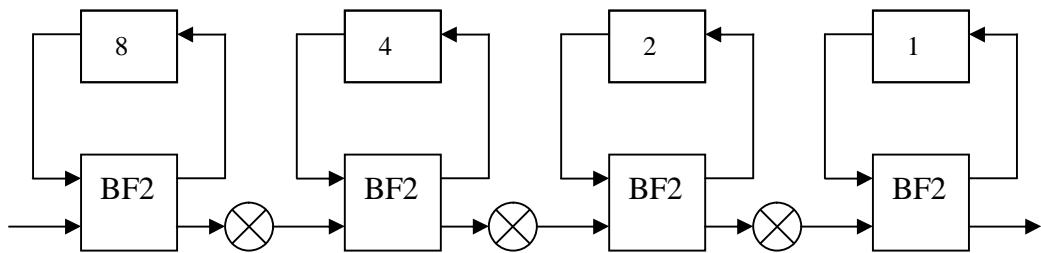


Fig3-2.1 16-point FFT with radix-2 基本 SDF 架構

當 FFT 運算進行定狀態後，每個暫存器的使用率將為 100%，因此系統運作正好是一級接著一級。每一次資料都會通過轉動因數乘法器 (twiddle factor multiplier)，因此可以使用 CORDIC 的技術，進行更深層的管線 (pipeline) 處理，速度將可進一步提昇，在本章 3-6 節，將介紹 CORDIC 架構。

單一路徑延遲迴授系統 (SDF) 只需 $N-1$ 個記憶體儲存位置，相當節省硬體空間，Fig3-2.1 和 Fig3-2.2 分別是 radix-2 及 radix-4 單一路徑延遲迴授系統之硬體構造：

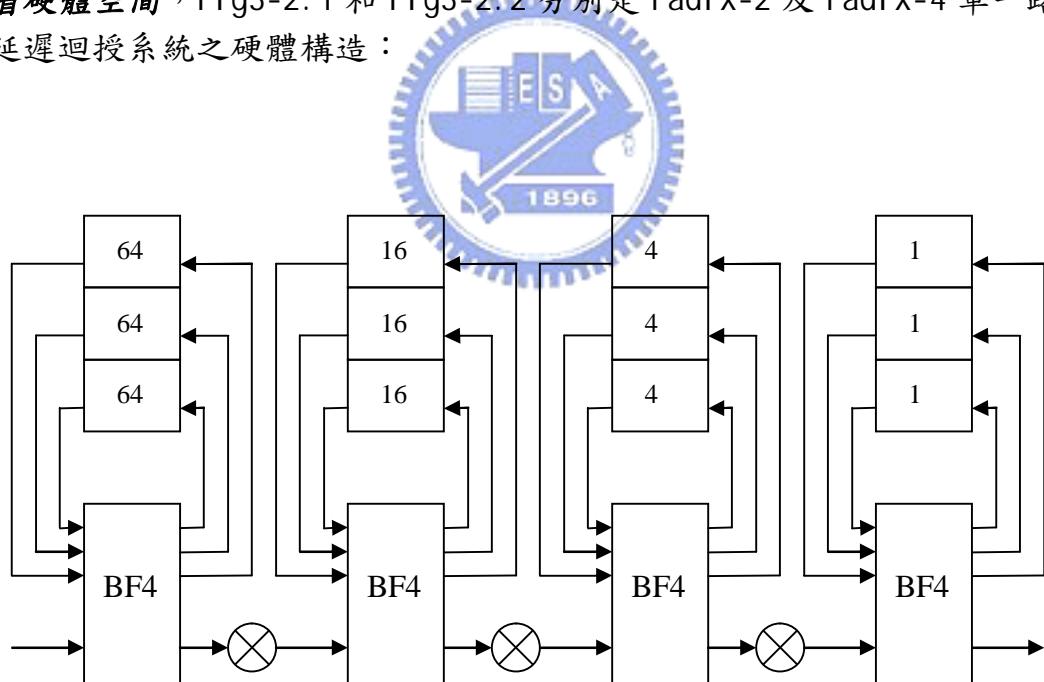


Fig3-2.2 256-point FFT with radix-4 基本 SDF 架構

Radix-4 單一路徑延遲迴授系統的優點，在於將乘法器的使用率提昇至 75%，因此可以讓 CORDIC 顯得更有功效，進一步可以提昇運算的輸出量 (throughput)，其缺點就是蝴蝶器 (butterfly) 的設計會比 radix-2 蝴蝶器來得複雜，這已是架構需要作評量 (trade-off) 的地方。radix-4 單一路徑延遲迴授系統所需要的硬體為 $\log_4(N-1)$ 個轉動因數乘法器 (twiddle factor multiplier)， $\log_4 N$ 個 radix-4 蝴蝶器，及記憶體儲存空間 ($N-1$) 個。

以下為 radix-2 與 radix-4 SDF 之比較：

	Adder	Multiplier	Memory	Control complexity	Processing rate (R Hz)
Radix-2	$4\log_4 N$	$2(\log_4 N - 1)$	$N-1$	Simple	R
Radix-4	$8\log_4 N$	$\log_4 N - 1$	$N-1$	Complex	R

TABLE3-2.1 radix-2 以及 radix-4 SDF 的比較

單一路徑延遲迴授系統目前被視為平均使用面積最精簡之系統架構，不過其 PE 需要在蝴蝶器及流通器作切換，硬體設計及控制較為複雜。以下將介紹 SDF 的相關架構：

3-2.1.1 Radix-2² SDF FFT

由第二章演算法的分析可知，radix-4 在運算複雜度會較 radix-2 簡單，因為其利用對稱性，將更多重要 (non-trivial) 乘法以不重要 (trivial) 乘法取代，即 $+j$, $-j$ 的複數運算以實部虛部交換位置的方式達成，因此計算量可以減小，也較有效率。但真正在 VLSI 實現，則需考量扇入 (fan-in), 扇出 (fan-out) 的問題，因此 radix-2 的架構反而較好，因為其扇入 (fan-in), 扇出 (fan-out) 數都較少，也因此較能適應 VLSI 的頻寬限制。整合這二項優點，即可構成 Radix-2² FFT[20]，這是將 radix-4 以二個 radix-2² 串接起來，並且適當使用轉動因數乘法，因此集合 radix-2, radix-4 之優點，形成一個新的架構，即 Radix-2² FFT architecture。Radix-4 及 Radix-2² FFT 的 SFG 如 Fig3-2.3 與 Fig3-2.4 所示：

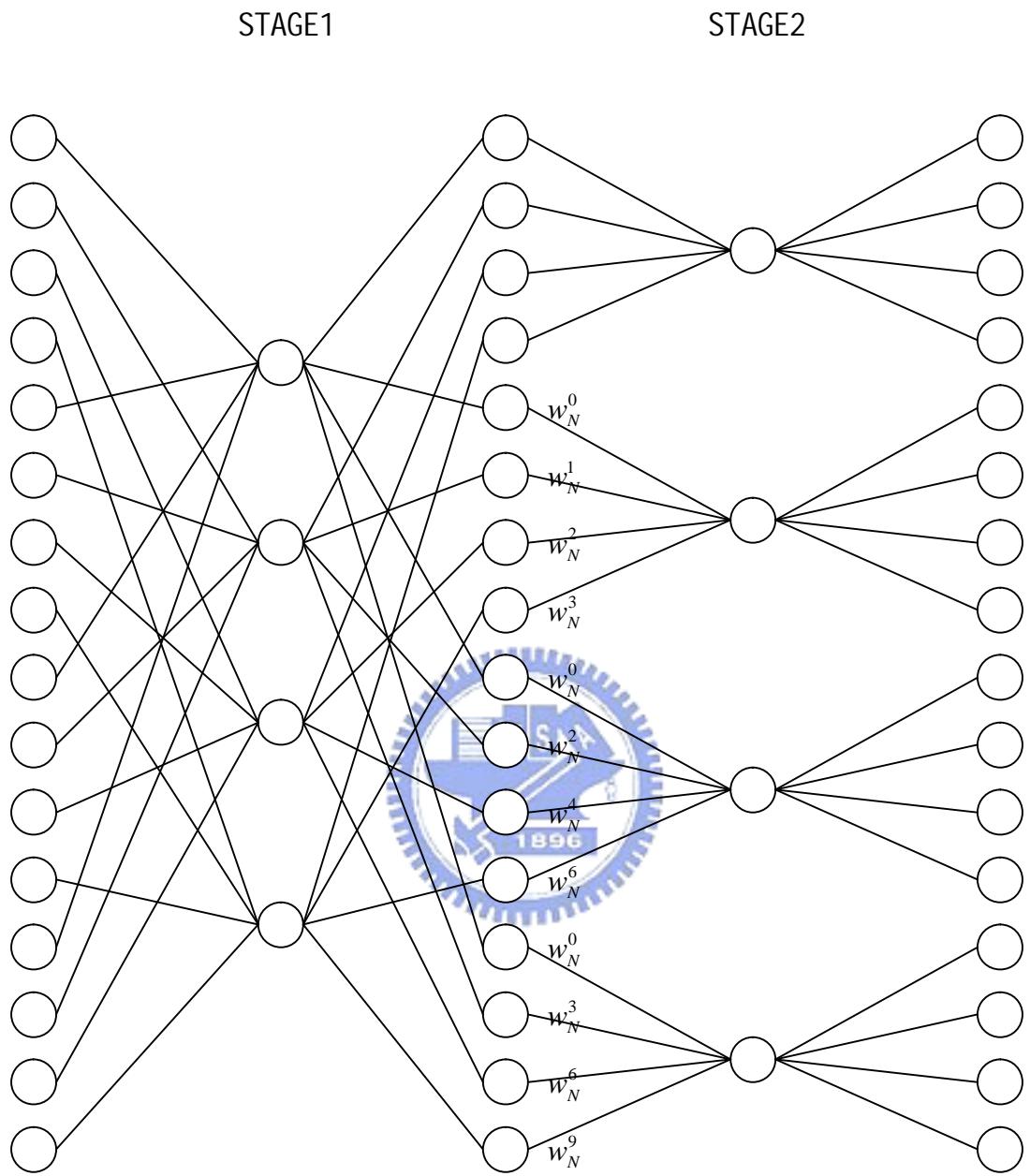


Fig3-2.3 FFT with Radix-4 簡化流程圖

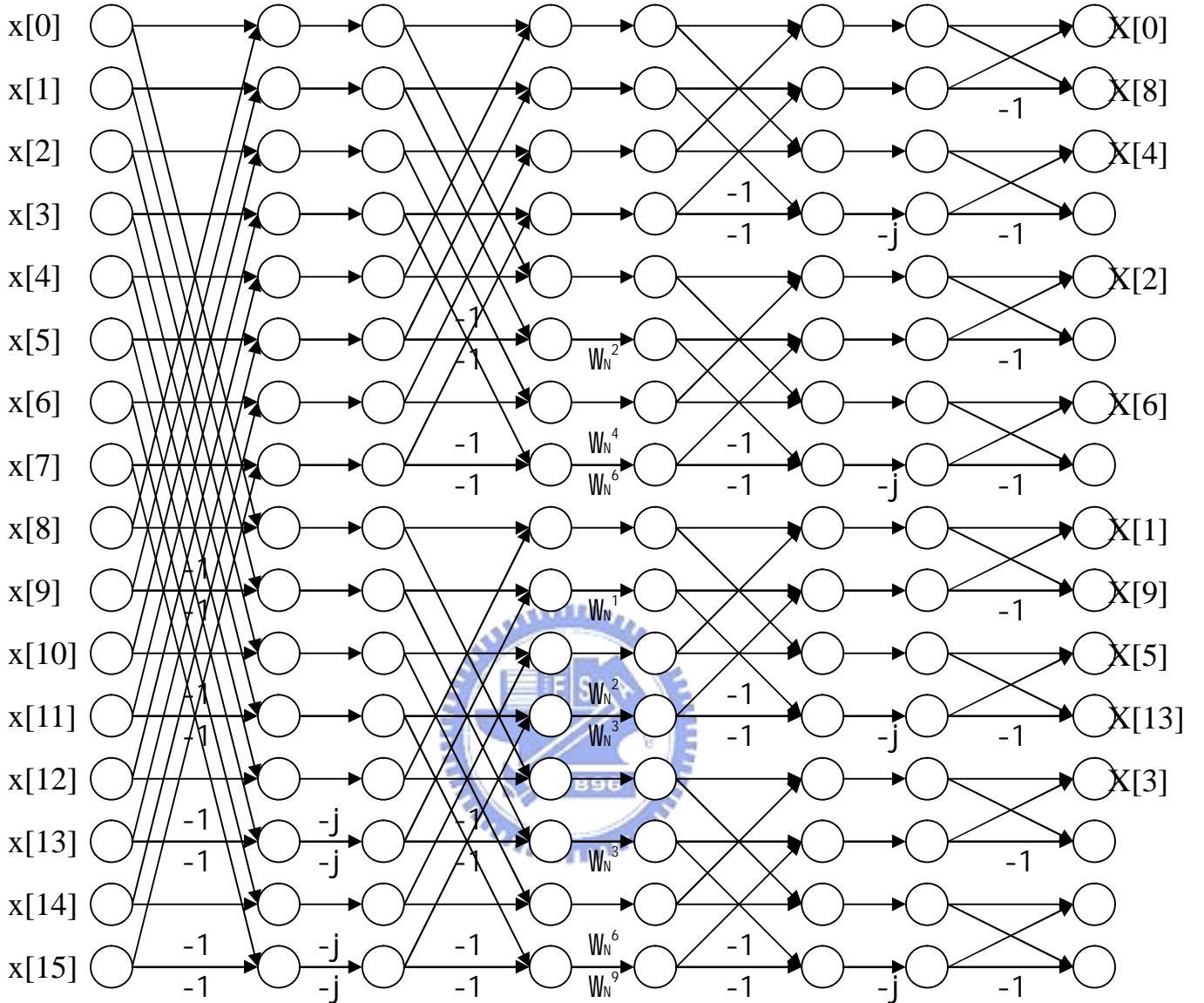


Fig3-2.4 16-point Radix-2² FFT

由 SFG 可以發現 Radix-2² FFT 的運算複雜度和 radix x-4 一樣，因此很有效率，以 N=256 作為例子，其硬體架構如 Fig3-2.5 所示：

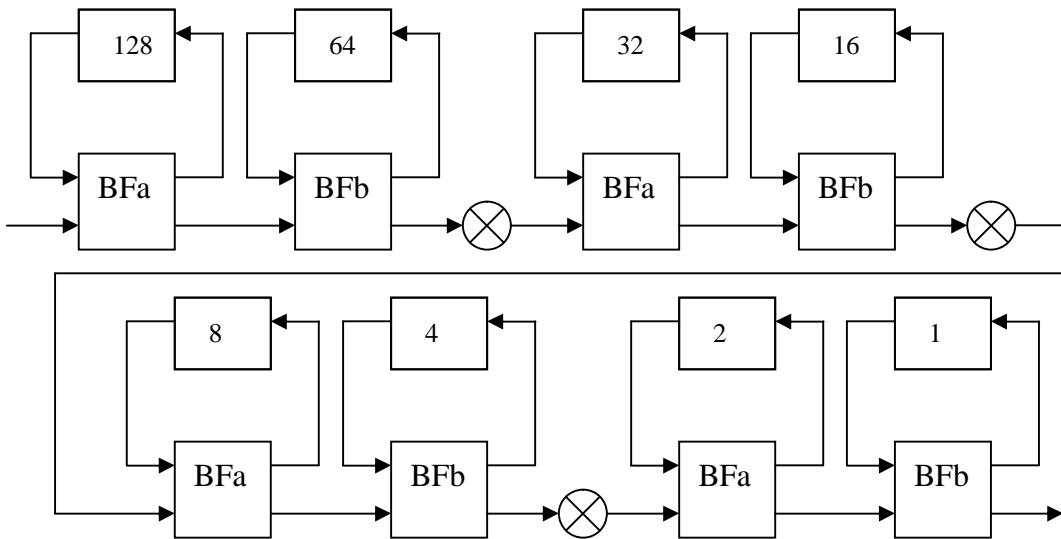


Fig3-2.5 256-point Radix- 2^2 FFT 硬體架構圖

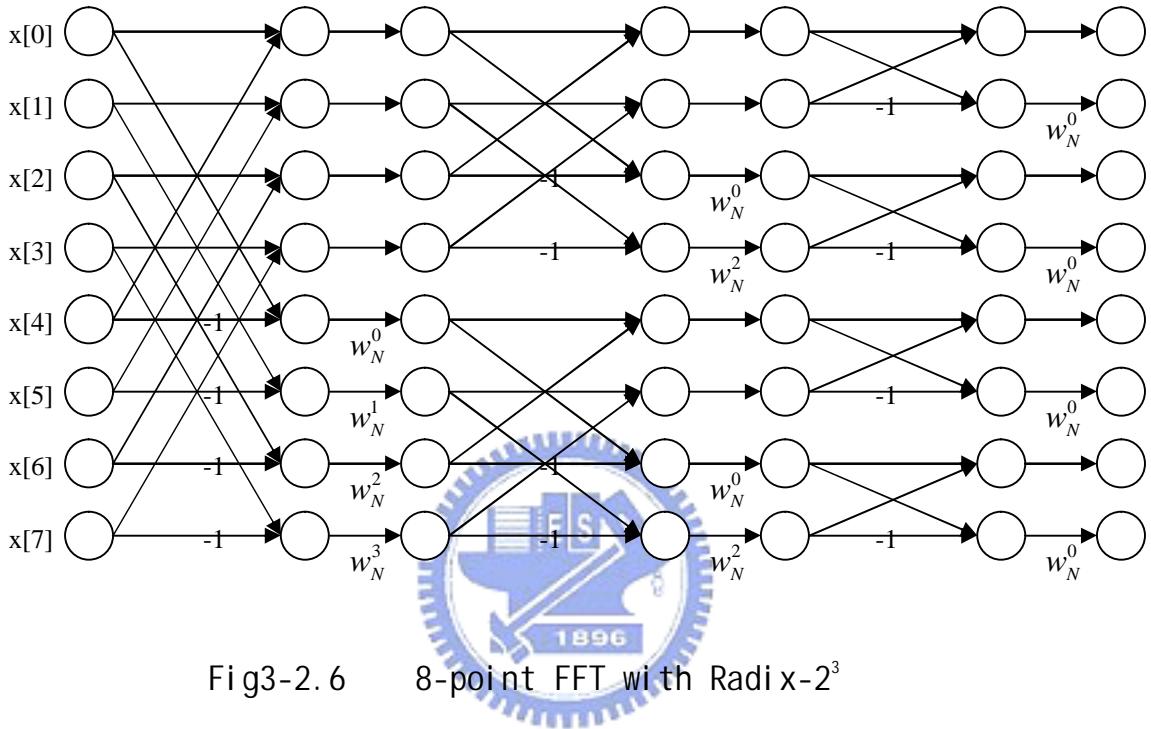
可以看到，Fig3-2.5 的架構非常簡單，使用時脈（clock）的跳動即可控制整個硬體操作，然後再運用 counter 來控制所有階段的運算行為，即可完成整個 FFT 的運算控制，用此方法之好處就是可以達到整體系統的同步。值得注意的是，這個架構包括二種不同的 butterfly，一種是原始的 radix-2 butterfly，另一種則是加入不重要轉動因數乘法器（trivial twiddle factor multiplier）邏輯處理電路的 radix-2 butterfly，由這二個 butterfly 即可達成一級的 radix- 2^2 FFT，也因此可利用 radix-4 不重要轉動因數（trivial twiddle factor）的特性，又具備 radix-2 彈性又簡易的架構，TABLE3-2.2 將 Radix- 2^2 FFT 與其它單一路徑延遲迴授系統架構的硬體複雜度作比較。由 TABLE3-2.2 可以知道 Radix- 2^2 在硬體複雜度上確實有較好的表現。

	Adder	Multiplier	Memory	Complexity
Radix-2	$4 \log_4 N$	$2(\log_4 N - 1)$	$N - 1$	Simple
Radix-4	$8 \log_4 N$	$\log_4 N - 1$	$N - 1$	Complex
Radix- 2^2	$4 \log_4 N$	$\log_4 N - 1$	$N - 1$	Simple

TABLE3-2.2 不同架構(Radix)的 FFT 硬體比較

3-2.1.2 Radi x- 2^3 SDF FFT

上一小節介紹 radix x- 2^2 FFT 架構，在這一小節將會提出更有效率之 radix x- 2^3 架構。Fig3-2.6 為 Radi x- 2^3 FFT 架構之 SFG。



由圖可知，此架構是由三級 radix x-2 FFT 所構成，並且配置乘法器於適當的位置，再加上適當的乘法器來達成。所以可視為這二種組合的硬體是一樣的。

從架構可以清楚看到，這種方式所需的乘法運算量非常少，所以在數學的計算是很有效率的，而且其由三個 radix x-2 FFT 構成，架構非常簡單，因此相當適合 VLSI 電路的實現。這些也就是這個架構優於其它架構的地方。

由第二章的演算法推導可以知道，這樣的架構很適合做管線技術處理 (pipeline)，所以可具有管線技術快速處理的優點，其架構如 Fig3-2.7 所示：

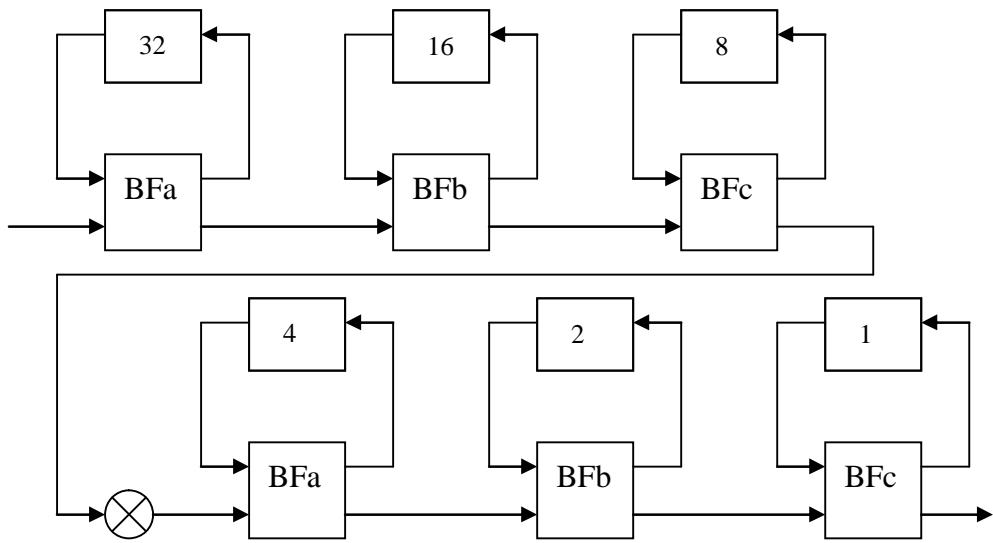


Fig3-2.7 64-point with radix $x-2^3$ 系統架構圖

其中 PE2 為包括處理 W_{4^n} 之不重要乘法運算 (trivial) 之蝴蝶器，PE3 為包括處理 W_{8^n} 之不重要乘法運算 (trivial) 之蝴蝶器，PE1 為基本的蝴蝶器。不過因為這個架構是由 3 個 radix $x-2$ 構成一級，所以每級都是 8 的基數，因此輸入的資料就需為 $N=8^n$ ，這是這個架構最大的缺點，改善方法為利用 mixed-radix 的方法，混合其它 radix 來符合所要計算的 FFT。舉個例子，假如使用這樣的架構來完成 DVB-T 的 8k mode [54]，則可以用 4 個 radix $x-2^3$ FFT，及 1 個 radix $x-2$ FFT 即可達成，如 Fig3-2.8 所示，每個 PE 代表一個 radix $x-2^3$ FFT。

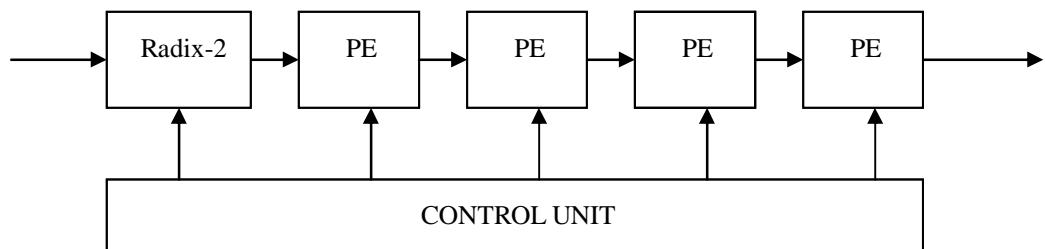


Fig3-2.8 DVB-T 8k mode 圖

綜合言之，Radix-2³的優點為VLSI硬體實作所佔的總面積小，低功率消耗，及架構簡單，所以相當適合系統設計的使用。其算複雜度已在第二章做過介紹，在這邊再稍作整理，資料列於TABLE3-2.3：

N	Radix-2	Radix-4
8	2	*
64	98	76
512	1538	*
4096	18434	13996

TABLE3-2.3 不同點數以及不同架構所用到的大約面積

備註：“*”代表此點不適用此演算法

3-2.1.3 Radix-8 SDF FFT



Radix-4 是利用對稱性，將更多的+j，-j 複數乘法以簡單的實部，虛部交換取代，來增加運算效率，同樣地，radix-8 也有相似的方法[19]，假設輸入訊號為 (a+jb)，轉動因數 (twiddle factor) 為 W 則：

$$(a+jb) \times W^{N/8} = (a+jb) \times \left(\frac{\sqrt{2}}{2} - j \frac{\sqrt{2}}{2} \right) = \frac{\sqrt{2}}{2} [(a+b) + j(b-a)] \quad (3.1)$$

$$(a+jb) \times W^{3N/8} = (a+jb) \times \left(-\frac{\sqrt{2}}{2} - j \frac{\sqrt{2}}{2} \right) = \frac{\sqrt{2}}{2} [(b-a) + j(b+a)] \quad (3.2)$$

$$(a+jb) \times W^{5N/8} = -(a+jb) \times \left(\frac{\sqrt{2}}{2} - j \frac{\sqrt{2}}{2} \right) = -\frac{\sqrt{2}}{2} [(a+b) + j(b-a)] \quad (3.3)$$

$$(a+jb) \times W^{7N/8} = -(a+jb) \times \left(-\frac{\sqrt{2}}{2} - j \frac{\sqrt{2}}{2} \right) = -\frac{\sqrt{2}}{2} [(b-a) + j(b+a)] \quad (3.4)$$

因此很明顯地可以知道，一個複數乘法對映 4 個實數乘法與 2 個實數加法可轉變為只用 2 個實數乘法與 2 個實數加法(control block)即可，所以

實作方法變得較簡單，運算量也會減少，由以下即為其實現的架構。

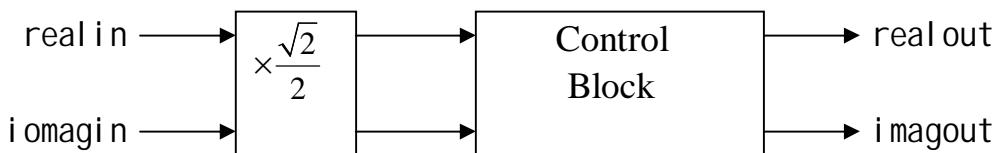


Fig3-2.9 W8 乘法部分的簡化架構

更進一步，可以將乘法的部份再次簡化，已知

$$\frac{\sqrt{2}}{2} = 0.70710678 = 2^{-1} + 2^{-2} + 2^{-4} + 2^{-6} + 2^{-8} + 2^{-9} \quad (3.5)$$

2^{-n} 代表將二位元資料向右移位 (shift) n 位，因此乘法部份可由 6 個移位器 (shifter) 及 10 個加法器來完成，其架構圖如 Fig3-2.10 所示：

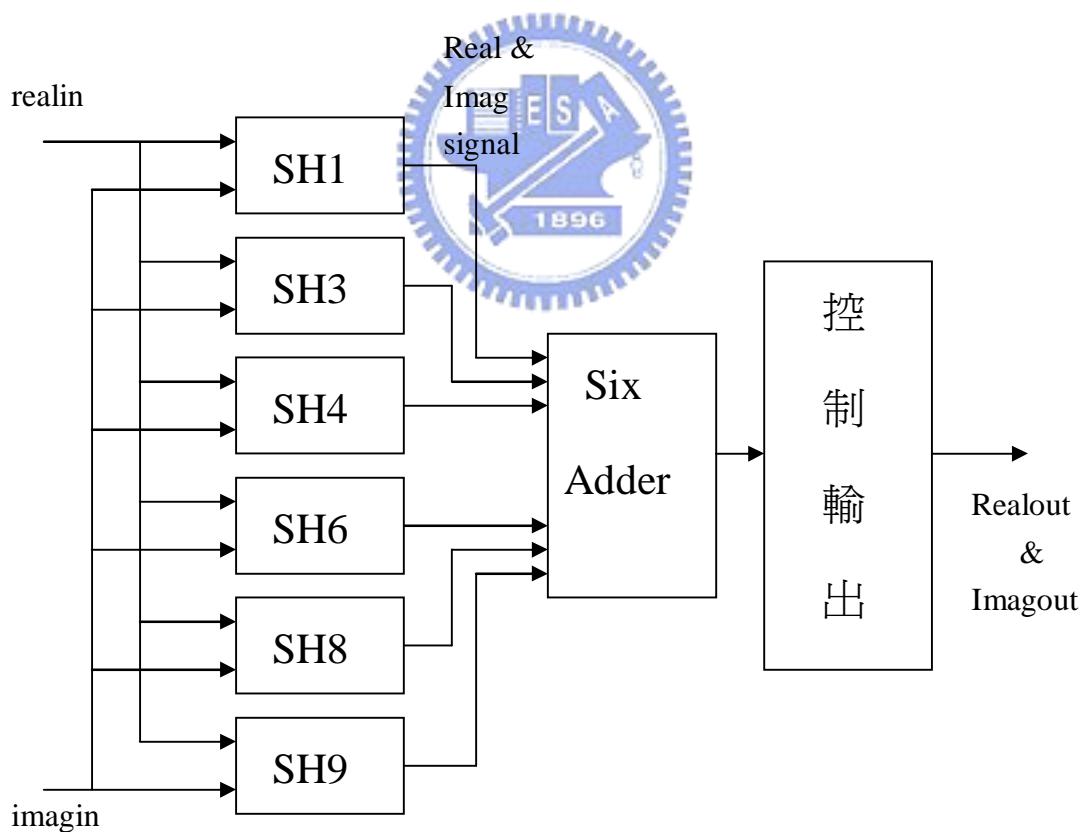


Fig3-2.10 W8 block 硬體架構

整體架構變得較為簡單，最後整個基礎體只需 12 個移位加法器 - (shift-adder) 即可完成，不需要使用任何乘法器，所以 $W^{N/8}$, $W^{3N/8}$, $W^{5N/8}$, $W^{7N/8}$ 的複數運算可以被視為不重要 (trivial) 運算，也因此大大地減少快速傅立葉轉換的運算量，此架構我們稱做 W8 block，而只包括位移加法器的部分稱做 W8 diagram，也就是說 W8 diagram 加上其 control 就等於 W8 block。完整的 radix-8 FFT 如 FIG3-2.11。

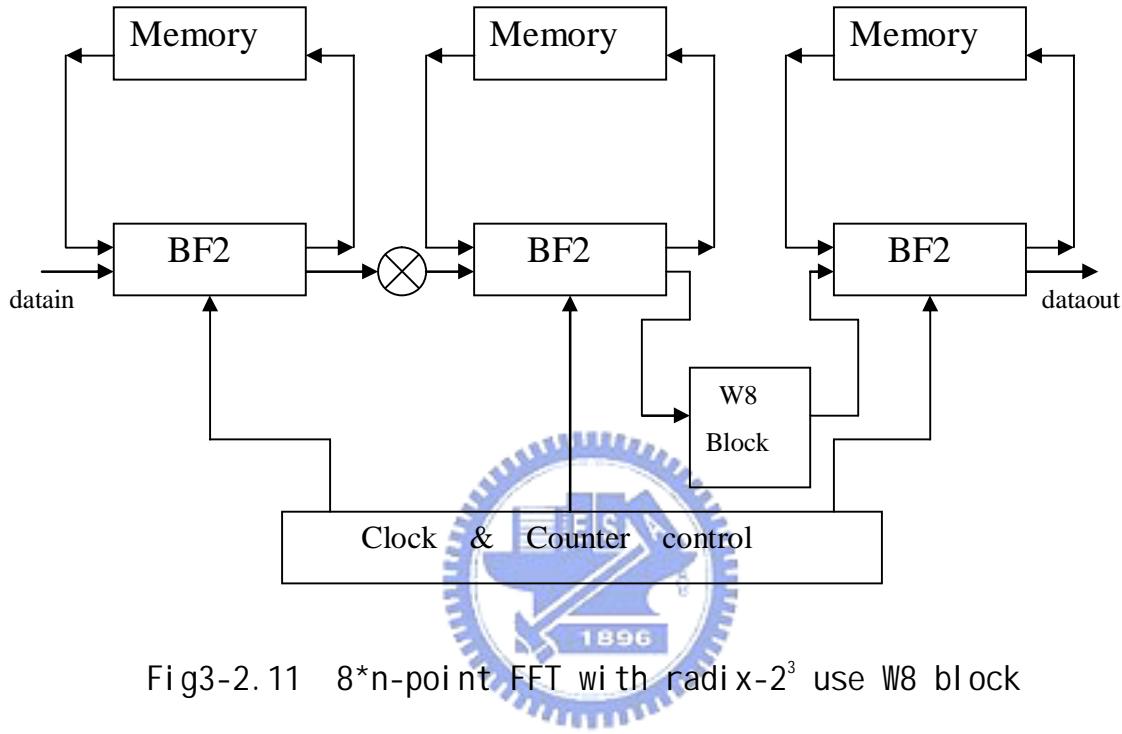


Fig3-2.11 8*n-point FFT with radix-2³ use W8 block

這個架構主要由三個 radix-2 FFT 蝴蝶器 (butterfly) 串接 (cascade) 所構成，在轉動因數 (twiddle factor) 複數乘法器的部份，則是由上述 12 個移位加法器 (shift-adder) 所構成，由這些移位加法器就可以算出實部虛部的計算結果。整體架構並不需要複數運算，而且因為具有規則性，因此可以切成多級管線 (pipeline)，使用具有很好的效能，這也就是許多高速運算的電子產品會採取這種方式的原因，這種架構已經成為將 radix-8 複數乘法視為不重要乘法 (trivial multiplication) 計算的基本方式之一。

3-2.2 多路徑延遲交換線路系統 (MDC)

多路徑延遲交換線路系統 (Multi-path delay commutator) 包括 radix-2 MDC 及 radix-4 MDC 二種架構，其中 radix-2 MDC 是 radix-2 FFT pipeline 最典型的架構，其將輸入資料區分為二個團體 (group)，每個團體透過交換器的切換，行經不同的路徑。MDC 架構最主要的元件為交換器 (commutator)，暫存器 (register)，蝴蝶器 (butterfly)，及乘法器 (multiplier) 如 Fig3-2.12 所示：

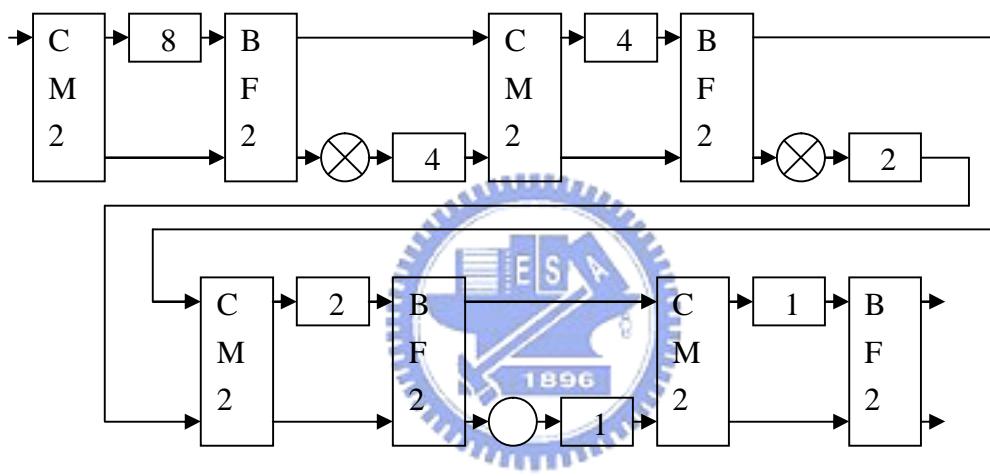


Fig3-2.12 Radix-2 MDC

Fig3-2.13 是 radix-2 MDC 的架構器，由其中可以算出其構造為 $(\log_2 N) - 2$ 個乘法器， $\log_2 N$ 個 radix-2 蝴蝶器及 $(1.5N - 2)$ 個單位延遲暫存器 (delay element register)。其蝴蝶器和乘法器的使用率為 50%，因為這個架構相當簡易，因此常用來做基本硬體。Radix-4 MDC 的架構和 radix-2 MDC 的架構相似，其構造如 Fig3-2.13 所示：

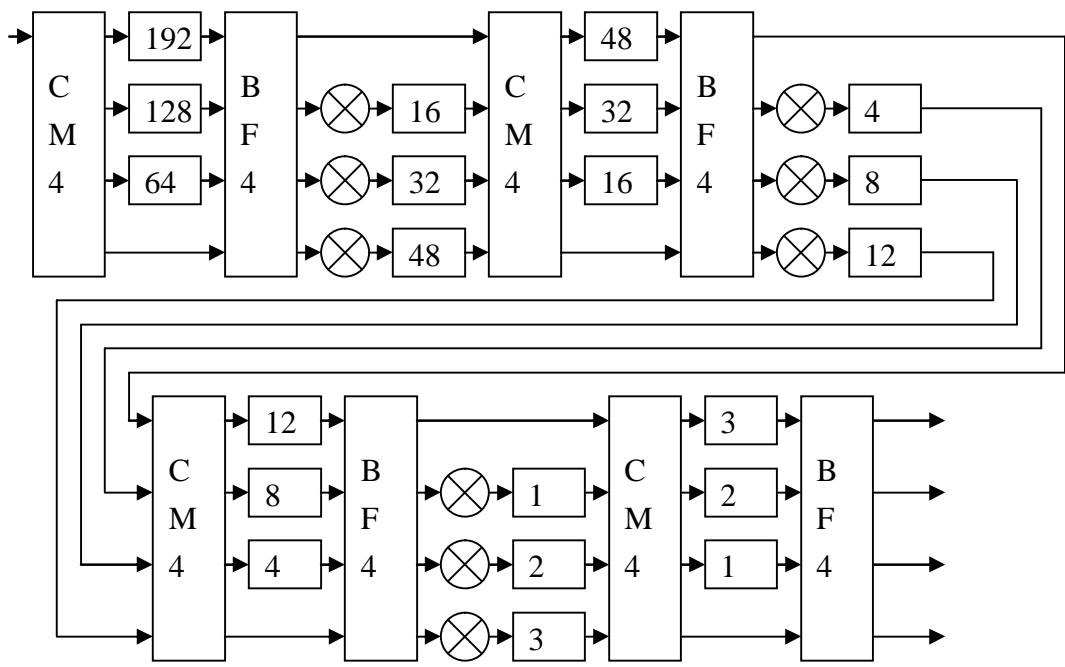


Fig 3-2.13 Radix-4 MDC

這個架構已經有 Pipeline FFT processor 的 VLSI 硬體實現。

 Radix-4 MDC 的乘法器使用率只有 25%，較其它架構效率低，因此如果要提昇其輸出量 (throughput)，則可用多筆 FFT 資料於同一架構進行同時處理 (parallel)，以增加效率，例如一次可以有四筆 FFT 資料同時進行計算，如此硬體使用率可提升至 100%。其所需之硬體為 $3(\log_2 N)$

-3 個乘法器， $\log_2 N$ 與 radix-4 蝴蝶器， $(2.5N-4)$ 個暫存器，TABLE3-2.4 為 radix-2 與 radix-4 MDC 之比較。在之後將會介紹一些多路徑延遲交換線路系統 (multi-path delay commutator) 的相關架構。

	Adder	Multiplier	Memory	Complexity
Radix-2	$4\log_2 N$	$2(\log_2 N - 1)$	$1.5N-2$	Simple
Radix-4	$8\log_2 N$	$3(\log_2 N - 1)$	$2.5N-4$	Simple

TABLE3-2.4 MDC 運用不同架構(radix)的硬體消耗比較

3-2.2.1 Radix- 2^3 MDC FFT

在 3-2.1 單一路徑延遲迴授系統介紹過 radix- 2^3 FFT 在這一節將使用不同架構來實現相同的演算法，其架構稱為 radix- 2^3 多路徑延遲交換線路系統 (multi-path delay commutator)。為 SDF 架構最大的不同，在於交換器 (commutator) 的使用。Radix- 2^3 FFT MDC 硬體的架構圖如 Fig3-2.14 所示：

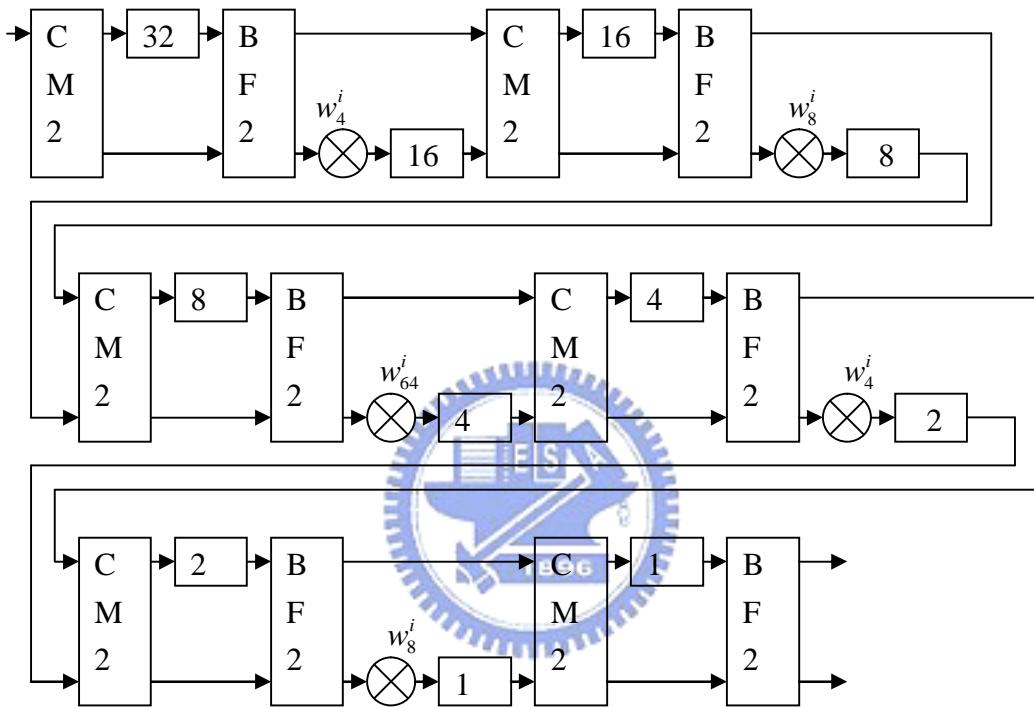


Fig3-2.14 radix- 2^3 MDC

由圖中可以看到其最主要的元件為蝴蝶器 (butterfly) 交換器 (switch commutator)，暫存器 (register)，及轉動因數乘法器。這種架構的好處和 radix- 2^3 SDF 一樣，可將更多的複數乘法轉變為不重要的乘法 (trivial multiplication)，例如 $+j$, $-j$ 的乘法只需作實部，虛部交換再於其中一項變號即可，不需要 4 個實數乘法器，2 個實數加法器來實現複數乘法。還有可將轉動因數 $W^{N/8}$, $W^{3N/8}$, $W^{5N/8}$, $W^{7N/8}$ 之複數運算簡化，如 3-2.1.3 節之說明。

radix- 2^3 的優點在於可以減少重要性乘法 (non-trivial

multiplication) 的計算量，因此在減少複數乘法器的使用情況下，硬體的面積可以變得更小，功率消耗也會降低(即 low-area and low-power)，不過因為其為 radix-2 之架構，所以其最高之同步處理率 (processing rate) 只有 2R，較 radix-4 之最高同步處理率 4R 來的小。還有整個系統架構因為加入許多緩衝暫存器，因此當輸入資料點數變多，例如 8192 點，則整個計算延遲 (latency) 將會變得很大，此為其最大之缺點。不過優點就是架構設計簡單。因此系統規畫時，要在硬體簡單化，時間延遲，功率消耗與速度作評估 (trade-off)。

值得注意的是，每三級最多只有一個重要性乘法運算 (non-trivial multiplication)，因此整體運算量變得非常少，比起其它架構，例如 radix-4 MDC 需要 6 個重要性乘法運算，radix-4/2 MDC 需要 4 個重要性乘法運算等，radix-2³ 功率較好，因此常被使用。TABLE3-2.5 為不同架構最高處理率 (processing rate) 與重要性乘法運算 (non-trivial multiplication) 之比較：

Algorithm	Processing rate	Non-trivial multiplication
Radix-2 ³ MDC	2R	1
Radix-4 MDC	4R	6

TABLE3-2.5 Radix-2³ MDC 以及 Radix-4 MDC 的比較

3-3 基於座標旋轉數位電腦之快速傅力葉轉換架構(CORDIC)

Cordic，其全名為 Coordinate Rotation Digital Computer，其原理是利用座標平面代換的方法，將比較複雜的複數運算(乘法)用比較簡單的加法和一些位移(shift)來完成。既是將 X-Y 座標平面換到極座標來做運算，也就是說將原向量旋轉一個角度之後，振幅再乘上一個實數，最後其向量位置就是複數乘法計算後的值，由於在旋轉的過程是將角度作

加減法來逼近數值，所以做完逼近後的數值會放大一個數量，以 cordic system 來看其放大倍率約為 1.64676，但是因為只是簡單的角度加減法之逼近，所以其數學運算實現到硬體上會變的比複數乘法器減少許多運算量，而且用到的加法器(adder)和位移器(shift)也比複數乘法器來的少許多，因此 cordic 非常適合用在 FFT 硬體架構上替代原本的複數乘法器，以下介紹 cordic 的數學演算式和原理。

假如 $A=a+jb$ (假設在極座標角度為 q_A)為原始 data，現在假設要乘上一複數為 $B=c+jd$ (假設在極座標角度為 q_B)，得到的結果假設為 $Z=x+jy$ (假設在極座標角度為 q_C)，也就是 $Z=x+jy=(a+jb)(c+jd)$ (3.6)

$$\text{其中 } x=ac-bd \quad y=ad+bc \quad (3.7)$$

以上是利用複數乘法器時的原理，若將此運算換到極座標上計算，則運算就如下圖所示：

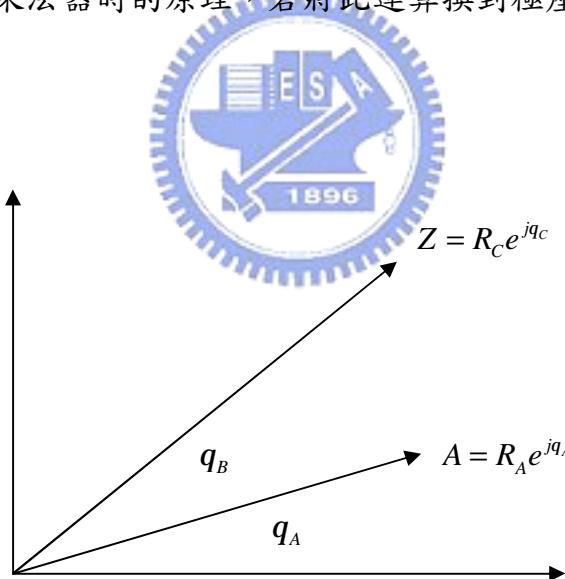


Fig. 3-3.1 複數乘法的極座標表示法

$$\text{其中 } A=a+jb=R_A e^{j q_A} \quad B=c+jd=R_B e^{j q_B} \quad Z=x+jy=R_C e^{j q_C} \quad (3.8)$$

$$R_Z = R_A \times R_B \quad q_Z = q_A + q_B \quad (3.9)$$

所以我們可以發現，cordic 主要在振幅上和角度上做運算，此方法比原本的複數乘法器就來的要簡化許多。所以若將 cordic 運算用在 FFT 上：

$$\text{已知 } X[K] = \sum_{n=0}^{N-1} x[n]w_N^{kn} , \quad K=0, 1, \dots, N-1 \quad (3.10)$$

其中 $w_N = \exp(-j2\pi/N)$ ， $X[K], x[n]$ 為複數，因此

$$(X_r + jX_i) = x[n]\exp\left(\frac{-j2knp}{N}\right) = (x_r + jx_i)(\cos\frac{2knp}{N} - j\sin\frac{2knp}{N}) \quad (3.11)$$

$$\begin{bmatrix} X_r \\ X_i \end{bmatrix} = \begin{bmatrix} \cos\frac{2knp}{N} & \sin\frac{2knp}{N} \\ -\sin\frac{2knp}{N} & \cos\frac{2knp}{N} \end{bmatrix} \begin{bmatrix} x_r \\ x_i \end{bmatrix} \quad (3.12)$$

$$\text{令 } q = \frac{2knp}{N} = \sum_k q_k , \quad q_k = \tan 2^{-k}$$

$$\begin{bmatrix} a_{k+1} \\ b_{k+1} \end{bmatrix} = \begin{bmatrix} \frac{1}{\sqrt{1+\tan^2 q_k}} & \frac{\tan q_k}{\sqrt{1+\tan^2 q_k}} \\ -\frac{\tan q_k}{\sqrt{1+\tan^2 q_k}} & \frac{1}{\sqrt{1+\tan^2 q_k}} \end{bmatrix} \begin{bmatrix} a_k \\ b_k \end{bmatrix} = \frac{1}{\sqrt{1+2^{-2k}}} \begin{bmatrix} 1 & 2^{-k} \\ -2^{-k} & 1 \end{bmatrix} \begin{bmatrix} a_k \\ b_k \end{bmatrix} \quad (3.13)$$

由電腦二位元的特性，此運算指只需要將 a 和 b 移位(shift)k 個位元(bits)後再相加，再乘上正規因數， $(\frac{1}{\sqrt{1+2^{-2k}}})$ 即可完成複數運算。因為 q_k 是經過電腦系統數位化處裡($q_k = \tan 2^{-k}$)，每個所旋轉的角度是不連續

的，因此要完成 q ，需要做多次 q_k 的移位後相加動作(也就是先前說的逼近法)。因為每一次動作都相似所以只需要做多次之旋轉再乘上正規值(此正規值即為 $1/\sqrt{1.64676}$)即可達到複數乘法的目的。因此

$$\begin{bmatrix} X_r \\ X_i \end{bmatrix} = \begin{bmatrix} a_t \\ b_t \end{bmatrix} = \frac{1}{\sqrt{1+2^{-2(t-1)}}} \cdot \frac{1}{\sqrt{1+2^{-2(t-2)}}} \cdots \frac{1}{\sqrt{1+2^{-2(t-t+1)}}} \cdot \begin{bmatrix} 1 & 2^{-(t-1)} \\ -2^{-(t-1)} & 1 \end{bmatrix} \cdot \begin{bmatrix} 1 & 2^{-(t-2)} \\ -2^{-(t-2)} & 1 \end{bmatrix} \cdots \begin{bmatrix} 1 & 2^{-1} \\ -2^{-1} & 1 \end{bmatrix} \cdot \begin{bmatrix} x_r \\ x_i \end{bmatrix} \quad (3.14)$$

其中 X_r, X_i 各代表 X 的實部和虛部

由數學演算法可知，使用 *cordic* 的好處是可以將複數乘法運算變為單純的加減法，因此運算量減少。基本上在做位移加法動作的時候，要看我們所需要的精準度如何，基本上最少要用到八級到十二級數的逼近，所以可以將此多級相同之位移加法硬體架構，用來實現 pipeline，因此用在 FFT system 上相當有效果。只要 FFT system 有複數運算時，就可以使用 *cordic* 方法，尤其是 FFT system 愈大的時候，乘法器愈多的時候，此時用 *cordic* 的總提升效率會愈高。比如說，先前介紹的 radix-4 單一路徑延遲迴授系統架構(SDF)，如 Fig. 3-3.2:

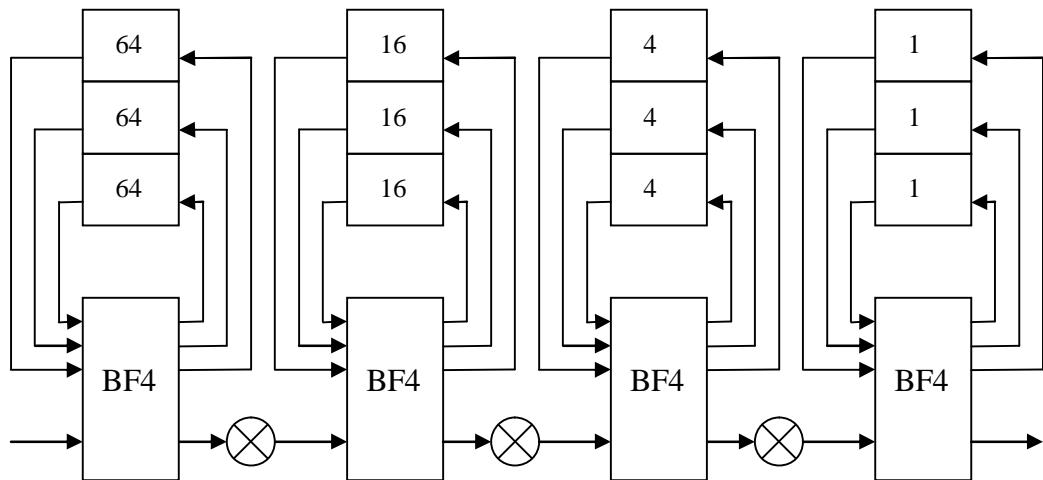


Fig. 3-3.2 256-point FFT with radix-4 基本 SDF 架構

這個架構的乘法器使用率高達 75%，若使用 cordic 方法，則此 FFT 架構會比原本的架構來的省 power 以及面積，但是其缺點就是其延遲 (*latency*) 會變大，若是 cordic 本身逼近的級數為 N 次，以 radix-4 單一路徑延遲迴授系統架構(SDF)來看，其總延遲會是 $3 \times N$ 個 *clock cycles*，比用複數乘法器的 3 個 *clock cycles* 整整變成了 N 倍，若是其 FFT system 架構很大，而複數乘法器全部都以 cordic 來替代，則最後整個架構的延遲會變的非常龐大，所以要利用 cordic 架構還是要觀察整個 FFT 架構的乘法內容。



3-4 各架構之比較

本章介紹多種系統架構，在這一節將會作綜合分析，由前面幾種不同架構，可以歸納出幾個特性：

1. 處理器數量與輸出率 (*throughput*) 及面積 (*area*) 有關，處理器越多，運算速度越快，*throughput rate* 越高，但所佔面積越大，消耗功率越多，VLSI 實作成本也越大，因此在硬體，速度，及功率需要作詳細的考量 (*trade-off*)。
2. 具有管線 (*pipeline*) 技術之架構，其速度會加快，因此如果硬體為速度導向，應選擇具有 *pipeline* 之系統架構。
3. *Radix* 層級越高，運算量越少，平均功率消耗也越小。例如 *radix-2³* 架構之運算量比 *radix-2*, *radix-4* 架構少。
4. 交換網路 (*permutation network*) 會隨著處理器數量的增加而變得更為複雜。
5. 乘法器之使用率越高，則使用 *CORDIC* 技術來作複數運算之效率提升越高。

瞭解以上特性，高可以適當選擇所設計之硬體架構，以符合需求，例如若要開發攜帶式 (portable) 電子系統，因為電池容量是固定的，因此對功率消耗相當敏感，如果系統所要求的速度標準不會很高，則可以選擇垂直兼水平投影架構，因為其只需要單一處理器 (PE) 即可完成運算，單位時間之能量消耗很低，可以節省功率。

如果設計的架構對功率條件要求不是很高，但對面積及速度要求高，例如 ADSL modem，則可以使用垂直投影架構。因為 ADSL 的運算點數是固定的，所以可以選擇變動性不大之架構，例如單一路徑延遲回授系統 (SDF)，多路徑延遲交換線路系統 (MDC)，目前已友多數廠商是利用這些系統架構做出 ADSL modem。

如果系統切換成不同點數模式 (*mode*) 之頻率很高，例如 DAB，DVB 系統，則可以使用水平投影之架構，或垂直兼水平架構，因為水平投影架構的完全行向系統及部份行向系統具有交換網路 (*permutation network*) 的架構，可以在不同模式下，採用不同之資料交換 (*data permutation*)，而不用變更原處理器 (PE)，因此相當適合，而且如果在速度要求允許下，還可以減慢速度，以部份行向系統取代完全行向系統，來減少處理器的數

量，進而縮小硬體面積，節省功率消耗。如果使用垂直兼水平架構，因為資料全部儲存於記憶體中，因此只要指定讀取資料的位置及一次 FFT 所要處理的點數，即可達到切換模式之目的。不過因為垂直兼水平架構只使用一個處理器，所以速度受到限制，只有在某些模式才可以滿足速度的要求。

如果設計的架構對功率，面積，和處理器數量比較沒有限制，但要求速度要快，例如基地台之 RAKE 接收機，則可使用垂直投影架構或多處理器架構。在垂直投影架構中，因為可以使用管線 (pipeline) 及 CORDIC 技術，所以運算很快，這些系統的優點就是速度快，構造簡單，控制容易，因此成為對速度有要求之設計者最喜歡使用之架構。尤其是系統業者常使用此種架構於其推出之商品上。另外如果在垂直投影系統上要多做改進，可以選擇同系列之不同架構，例如單一路徑延遲回授系統 體使用效能大於多路徑延遲交換線路系統。如果要向更高層次挑戰，可以根據第二章所介紹之演算法做適合的運用，即可獲得特別的效果。

如果所用的系統，要求速度，也對資料處理延遲有很嚴格的要求，例如軍方之雷達偵測系統，機場塔台雷達系統，其對飛行物體之偵測非常重視，因為一個飛彈或一架飛機的速度相當快，如果運算之速度不夠快，或是資料偵測到處理完畢的延遲時間過久，將會影響國家安全，因此需要速度快，延遲小之系統，多處理器系統是最佳的選擇，因為其具有多個處理器分工同時計算之優點，因此速度快，反應時間很短，相當適合。不過因為設計複雜度很高，因此系統價格相當昂貴，只適合重點機關使用，並不適合一般商品上之競爭。

至於 CORDIC 架構方面，只要是有使用到複數乘法運算之 FFT，皆可使用 CORDIC 技術來加速，因為其加速原理與管線技術 (pipeline) 有關，複數乘法器之使用率越高，代換成 CORDIC 系統，所提升之效率也越高，

因此設計者在作系統分析時，應該先對各硬體的運算百分率做仔細計算，才可獲知使用 CORDIC 技術是否對速度有正面的幫助。

本章在最後以 TABLE 的方式，將各系統及其子架構作比較。

	Speed	Area	Scalable	Power consuming	Modularity	Complexity
SDF	High	Small	High	Low	High	Medium
MDC	High	Large	High	High	High	Simple

TABLE3-7.1 SDF 以及 MDC 架構的比較

註：SDF：單一路徑延遲回授系統 (Single path delay feedback system)

MDC：多路徑延遲交換線路系統 (Multi-path delay commutator)

	Speed	Area	Scalable	Power consuming	Modularity	Complexity
R	Medium	Medium	High	Medium	High	Simple
C	Medium	Medium	High	Medium	High	Medium

	Latency	Price	Throughput	Application
R	Medium	Medium	Medium	Wide
C	Medium	Medium	Medium	Narrow

Table3-7.2 水平架構以及垂直架構的各種比較

註：R：垂直投影架構 (Row major system)

C：水平投影架構 (Column major system)

我們由上面的資料可以得知，垂直投影架構彼此水平投影架構的複雜度來的簡單，所以比較受大部分人的接受，這也是為什麼垂直投影架構的彈性以及可變度比較高，而我們再拿 SDF 和 MDF 來做比較，MDF 其實不論是在 power 或是面積以及 memory size 上，都比 SDF 來的多，除了在複雜度上，MDF 比 SDF 簡單一些，但是實際上根本差不到哪裡去。所以基於以上幾點原因，我們將在下一章全部以 SDF 的 FFT 架構來實現在硬體上用以方便分析數據並且了解其影響。