行政院國家科學委員會專題研究計畫　成果報告

具可重覆使用智財之可組態可擴充特殊應用導向數位訊號處理架構

計畫類別：個別型計畫
計畫編號：NSC91-2218-E-009-019-
執行期間：91 年 12 月 01 日至 92 年 07 月 31 日
執行單位：國立交通大學電信工程學系

計畫主持人：紀翔峰

計畫參與人員：賴昭宏、 李昭宏 、李佳勳

報告類型：精簡報告

處理方式：本計畫可公開查詢

中　華　民　國 92 年 11 月 20 日

行政院國家科學委員會補助專題研究計畫成果報告

※※※※※※※※※※※※※※※※※※※※※※※※※
※　　　　　　　　　　　　　　　　　　　　　　　　※
※　可重覆使用智財之可組態可擴充特殊應用導向　※
※　　　　　　　　數位訊號處理架構　　　　　　　※
※　　　　　　　　　　　　　　　　　　　　　　　　※
※※※※※※※※※※※※※※※※※※※※※※※※※

計畫類別：■個別型計畫　　□整合型計畫

計畫編號：NSC　91－2218－E－009－019－

執行期間：91 年 12 月 01 日 至　92 年 07 月 31 日

計畫主持人：　紀翔峰助理教授

本成果報告包括以下應繳交之附件：

■赴國外出差或研習心得報告一份

□赴大陸地區出差或研習心得報告一份

□出席國際學術會議心得報告及發表之論文各一份

□國際合作研究計畫國外研究報告書一份

執行單位：

中　華　民　國　　　年　　　月　　　日

行政院國家科學委員會專題研究計畫成果報告

可重組雙模式可變大小之FFT/IFFT處理器核心及在分頻以符號為主之正交頻分調變多工存取及離散多重音調系統

Reconfigurable Dual-mode Variable-size FFT/IFFT Processor Core and the Application to Symbol-based FDD Adaptive Modulated OFDM and DMT Systems.

計畫編號：NSC 91－2218－E－009－019

執行期限：91 年 12 月 01 日至 92 年 07 月 31 日

主持人：紀翔峰助理教授　國立交通大學電信工程學系

計畫參與人員：賴昭宏 李昭宏 李佳勳　國立交通大學電信工程學系

一、中文摘要

由於正交頻分調變(OFDM)及離散多重音調(DMT)技術在解決 ISI 問題上的優越性,使得它們已廣泛地被應用在無線及有線傳輸系統上,隨著日漸增加的寬頻需求,高 OFDM (DMT) symbol 速率且大點數之快速(反)傅立葉轉換(FFT/IFFT)已是不可避免的趨勢,始得 FFT/IFFT 在晶片實現上變成相當龐大的運算,在此論文中,針對以 symbol-based FDD 方式運作的 OFDMA 及 DMT 系統(例如 IEEE 802.16a 及 VDSL),我們將提出一個有效的計算法,來縮短運算時間,也相對地可在要求的 timing 限制下達到最低硬體成本的 solution,為了在具有 adaptive modulation 機制之 OFDM(DMT)系統上發揮此演算法最佳的效能,我們需要在計算核心平台上動態地調適FFT/IFFT運算的大小,因此在本論文中,我們會先提出一個可架構雙模式可變大小之FFT/IFFT處理器架構,以此為基本的計算核心,最後提出一個適合用於FDD-based OFDMA 寬頻無線系統及DMT 寬頻有線系統之高效率低成本FFT/IFFT運算硬體架構。

關鍵詞：正交頻分調變、正交頻分調變多工存取、離散多重音調、雙模式快速(反)傅立葉轉換、可重組硬體架構、適應式調變、低成本。

Abstract

OFDM and DMT techniques have caught much attention because of the strength in solving the intersymbol interference (ISI) problem. To meet the broadband requirement, high OFDM (DMT) symbol rate and large-size FFT/IFFT are inescapable features in high bandwidth transmission systems. The short symbol period makes FFT/IFFT timing-critical calculation in OFDM (DMT) systems. Not conducting large-size FFT/IFFT upon an expensive hardware platform, we propose a smart way of computation scheme for symbol-based FDD OFDMA (e.g. IEEE 802.16a) and VDSL systems. To maximally utilize the computation kernel and minimize the processing time especially for a OFDM(DMT) system with adaptive modulation schemes, we also propose reconfigurable dual-mode variable size FFT/IFFT processor core architecture. By optimally partition FFT/IFFT upon this reconfigurable processing core, we could speed up the computation significantly without increasing the extra hardware cost.

Keywords: OFDM, OFDMA, DMT, VDSL, Reconfigurable hardware, Dual-mode FFT/IFFT, adaptive modulation, low-cost.

一、Introduction

隨著通訊及訊號處理技術的快速進步，以 OFDM 或 DMT 為調變技術的傳輸系統越來越普遍，這些多載頻通訊系統皆需要用 FFT/IFFT 來實現，所以如何設計一個高效率低成本的 FFT/IFFT 處理器一直是受注意的問題，由於迫切的傳輸寬頻需求，先進的 FDD-OFDMA 寬頻無線系統(如 IEEE 802.16a[1])及 DMT-based 寬頻有線系統 VDSL[2]皆採取多點數之 FFT/IFFT，且上傳 /下傳模式採用 symbol-based FDD 方式運作，對於單邊傳輸端而言，並非所有的頻率柱(frequency bin)都需要，我們只需計算出所要接收及傳送所需的的寬頻柱即可。一般的做法通常其接收/ 傳送機前端的 FFT/IFFT 都採取固定點數且相當大的 FFT/IFFT，因此而浪費了許多計算效能以及記憶體面積。本論文將提出一個可架構雙模式可變大小之 FFT/IFFT 處理器架構， 利用此可架構式可變大小處理器為基本核心，我們將提出一個可動態調整 FFT/IFFT 運算大小之演算法，此方法對具有調整所用 Frequency bin 數之 adaptive modulation 的機制，更能顯出它的長處，此外，我們將進一步發展一個適合用於 FDD-based OFDMA 寬頻無線系統及 DMT 寬頻有線系統之高效率低成本 FFT/IFFT 運算硬體架構。

本論文首先將在第二節提出以可架構雙模式可變大小之 FFT/IFFT 處理器，然後在第三節提出一個高效率低成本 FFT/IFFT 運算架構，在第四節將作一個簡單的結論。

二、Reconfigurable Dual-mode Variable-Size FFT/IFFT

快速傅立葉轉換在 OFDM-based 無線寬頻通訊及 DMT-based 有線寬頻通訊是相當重要的運算，為了滿足寬頻的需要，在傳輸標準中 symbol 傳送率往往是相當高的， FFT 及 IFFT 必須要在幾個 microseconds 內就完成。除此之外，為了因應多模通訊系統，硬體需要支援可變點數之 FFT 及 IFFT，更進一步地，為了 提供足夠的彈性給後述之 FDD-based 調變系統，雙模式之 FFT/IFFT 必需在同一硬體上實現， 來達到有效地運用硬體資源並減少其面積大小。為滿足所需的高運算率，一般都採用 pipeline FFT [3, 4] 架構，然而此架構需要大量的記憶體模組來完成 permutation，浪費晶片成本，此外 pipeline FFT 架構無法提供足夠的可重組態(reconfigurability)及彈性來達到可變點數 FFT/IFFT， FFT 與 IFFT 架構通常都是個別做成不同的平台，面積的大小與硬體的浪費是非常顯著的，在 FFT/IFFT 執行時間及硬體成本間找到平衡點。

● Radix-2 Variable-Size FFT/IFFT

一個長度為 N 的序列 $x[n]$，其離散傅利葉轉換可以表示為：

$$X[k]=\sum_{n=0}^{N-1}x[n]W_N^{nk} \quad ,k=0,1,2,\cdots,N-1 \qquad (1)$$

其中 $W_N=e^{-j(2\pi/N)}$

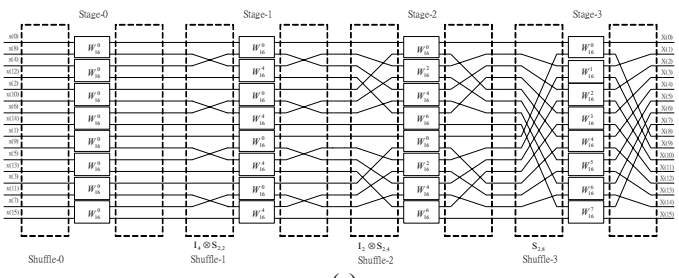
N 點 $X(k)$之反離散傅利葉轉換為

$$x[n]=\sum_{k=0}^{N-1}X[k]W_N^{-nk} \quad ,n=0,1,2,\cdots,N-1 \qquad (2)$$

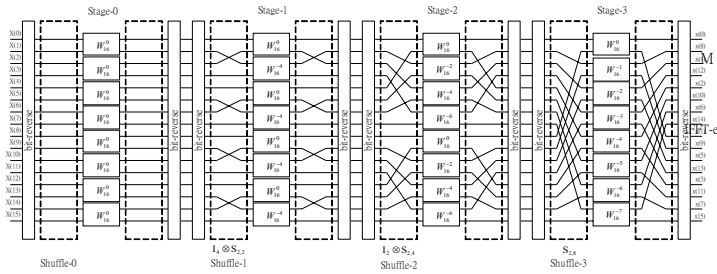
採用 Decimation-in-Time(DIT)方式， 我們可得到快速離散傅利葉轉換，若是用 radix-2 作法，其最基本運算單元是 radix-2 DIT butterfly。圖 1-(a) 為 16 點 radix-2 DIT FFT 運算架構， 其輸入序列的儲存順序為位元反向順序，輸出序列則為正常順序，且每個 butterfly 計算完後，會把結果存回原本的記憶體位址，也就是所謂的 in-place 計算。我們將用單一個處理單元 (processing unit)序列式地(serially)計算所有需要的 radix-2 DIT butterfly。整個 16-點分時(time-sharing)快速傅立葉轉換流程圖共有 4 (=$\log_2$16) Stages 與 4 個 Shuffles，且不同的 Stage 有不同的 Shuffle。

至於 IFFT，一般的作法都是直接用 FFT 的運算模式來完成，差別只是要另外將輸入及輸出取共軛運算。

若欲運算實數 FFT 及 Hermitian symmetric IFFT，我們可利用 half-size 之複數 FFTIFFT 並配合適當的 post-processing 及 pre-processing 來完成，然而此有效率之運算需要 linear-order 之 FFT 輸出及 linear-order 之 IFFT 輸入，只用圖-1(a)當作 IFFT 運算方式是不恰當的，故我們另外設計具 linear-order 輸入之 IFFT，圖-1(b) 為 16 點 radix-2 DIT IFFT 運算架構， 其輸入序列的儲存順序為正常順序，輸出序列則為位元反向順序，且同樣的每個 butterfly 計算完後，會把結果存回原本抓取的記憶體位址。我們把上圖反向分時快速傅立葉轉換(IFFT)與分時快速傅立葉轉換(FFT)做個比較之後，發覺 IFFT 與 FFT 在相同 Stage 內所做的 Shuffling 都是一樣的。只是 IFFT 須比 FFT 多做一個 bit-reverse 的步驟，且 butterfly 所需的 twiddle factor 需取共軛。故我們只須稍微修改 FFT 架構就可得我們所要的 IFFT 架構圖。



(a)



(b)

Figure 1: (a) 16-point radix-2 DIT FFT; (b) 16-point radix-2 DIT IFFT

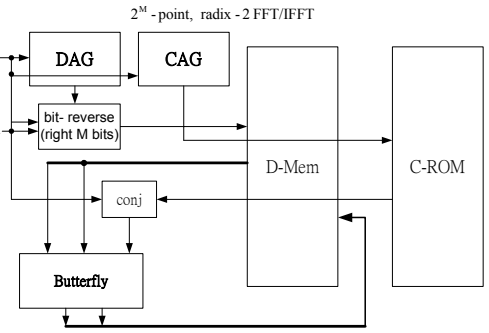


Figure 2: Dual-mode variable-size FFT/IFFT Architecture

- **Dual-mode Re-configurable FFT/IFFT Processor Architecture**

圖-2 為雙模可變點數 FFT/IFFT 處理器架構圖。整個架構只採用一個 DIT butterfly 作為運算單元，且每個 Stage 的 butterfly 都是由上而下依序算完後才算下一個 Stage。其次，FFT 的點數是可變動的，可以藉由控制訊號 M 來選擇做 N 點的分時快速傅立葉轉換，其中 $M=\log_2 N$。

其中 DAG 與 CAG 分別為資料位址產生器(Data Address Generator)與係數位址產生器(Coefficient Address Generator)用來產生 Butterfly 運算元的記憶體位址。D-Mem 與 C-ROM 分別為儲取輸入資料與 twiddle factors 的儲存元件。Butterfly 有三個輸入運算元，這三個運算元分別依照 DAG 和 CAG 所產生出來之位址去記憶體裡抓取，經由運算後會產生兩個輸出結果，輸出結果儲存的位址為原本抓取的輸入資料記憶體位址。

Radix-2 分時快速傅立葉轉換的 butterfly 運算單元中複數乘法需要由四個實數乘法和兩個實數加法來完成。但因為我們所用的記憶體區塊是 single-port，無法在同時執行讀寫動作，讀寫各需花一個 clock cycle。故 butterfly 有兩個 clock cycles 可供其作運算。所以若要完成一個複數乘法，平均一個 clock cycle 只需作兩個實數乘法，即可完成。故所需之 butterfly 運算因有兩個 clock cycles 可使用，它的複數乘法器只需兩個實數乘法器和兩個實數加法器即可達成複數乘法。

我們可以藉由控制訊號 IFFT-en 來把 FFT 與 IFFT 整合在同個平台裡。當控制訊號 IFFT-en 為 high 時，bit-reverse 與 conj(共軛)運作，做 IFFT 運算。反之，IFFT-en 為 low 時，bit-reverse 與 conj(共軛)停止運作，做 FFT 運算。

- **DAG(資料位址產生器)**

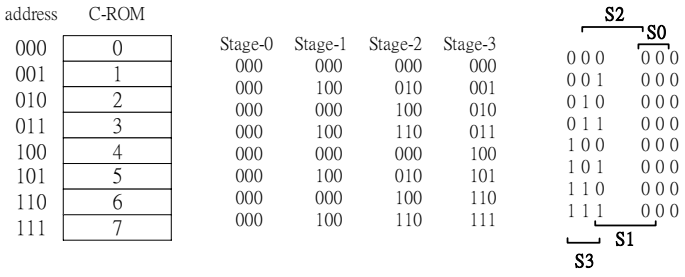
一個 $N=2^M$ 點的 radix-2 分時快速傅立葉轉換，其輸入序列為位元反向順序。共有 M 個 Stages，且每個 Stage 對應到不同形式的 Shuffling。第 i 個 Stage 的 Shuffling 為：

$$\mathrm{I}_{2^{M-1-i}} \otimes \mathrm{S}_{2^i,2} \qquad i=0,1,\cdots,M-1 \qquad (3)$$

我們可以很容易地知道其 DAG 的邏輯架構。其中 i-bit left-shift 動作無須用佔空間的 barrel shifter， 只需要一個簡單的解碼器就可完成.

- **CAG(係數位址產生器)**

一個 $N=2^M$ 點的 radix-2 分時快速傅立葉轉換，其輸入序列為位元反向順序。共有 M 個 Stages，且每個 Stage 之 twiddle factor 位址的抓法都是不同的。圖-3 為一 16-點 radix-2 分時快速傅立葉轉換的例子。



address	C-ROM
000	0
001	1
010	2
011	3
100	4
101	5
110	6
111	7

Stage-0	Stage-1	Stage-2	Stage-3
000	000	000	000
000	100	010	001
000	000	100	010
000	100	110	011
000	000	000	100
000	100	010	101
000	000	100	110
000	100	110	111

Then, we get that the i-th stage must shift left M-i-1 bits.

Figure 3: Coefficient address generation scheme

C-ROM 裡存的為 twiddle factor 值，0 表示 W_N^0 、1 表示 W_N^1 依此類推。由上所示的 16-點 FFT 中 Stage-0 所要抓取的位址可由計數器所產生之值左移 3 位元後的 S0 得之，Stage-1 所抓取的位址可由計數器所產生之值左移 2 位元後的 S1 得之，同理 Stage-2 則需由左移 1 位元後的 S2 得之，最後的 Stage-3 則是左移 0 位元後的 S3。經由上面的分析，我們可以很簡單的找到其規律性。一個 $N=2^M$ 點的 FFT 在第 i 個 Stage 只需把計數器產生之值左移 M-i-1 個位元就可得到所需的 twiddle factor 位址。下圖為其 CAG 邏輯架構圖。

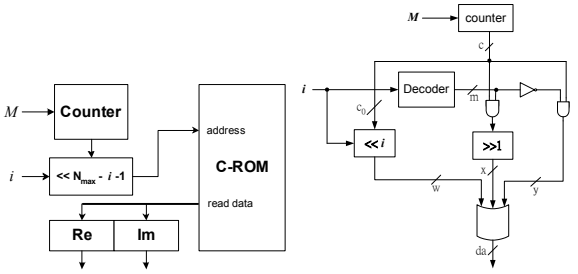


Figure 3: Block diagram of CAG

- **Conflict Free Memory Addressing**

為了避免butterfly在同一記憶體區塊同時讀寫兩筆之資料以上的情況發生，我們必須對FFT的輸入序列做適當的記憶體區塊分配，使得每次butterfly的運算元都個別來自不同的記憶體區塊。若 butterfly 為 radix-r，則記憶體需被分為 r 塊，使得 radix-r 的 butterfly 可以同時讀寫 r 個值。由簡單的方式可將區塊配置如下述[5]。

根據上述提出之 radix-2 FFT/IFFT 處理器架構，我們想把原本儲存在一個記憶體內的輸入序列，重新分配至兩個記憶體區塊，記憶體區塊分別為 Bank0 與 Bank1。根據下面兩個式子，可清楚的知道輸入序列要配置到哪個區塊和區塊中的哪個位址。

$$\mathrm{M}\left[\mathrm{F}_{r^c}(n_0,\cdots,n_{R-1-c},k_{c-1},\cdots,k_0)\right]=n_0\oplus\cdots\oplus n_{R-1-c}\oplus k_{c-1}\oplus\cdots\oplus k_0 \quad (4)$$

$$a_i\left[\mathrm{F}_{r^c}(n_0,\cdots,n_{R-1-c},k_{c-1},\cdots,k_0)\right]=\begin{cases}k_i & ,0\le i\le c-1\\ n_{R-1-i} & ,c\le i\le R-2\end{cases} \quad (5)$$

其中 M[ ]代表分至的記憶體區塊，a_i[ ]代表分至區塊裡的位址。

- **Architecture with Conflict Free Memory Banks**

圖-4 為將輸入序列分至兩個記憶體區塊的整個架構概況圖。控制訊號 IFFT-en 用來決定作何運算，IFFT-en 為 high 時，為 IFFT 運算，反之，IFFT-en 為 low 時，做 FFT 運算。

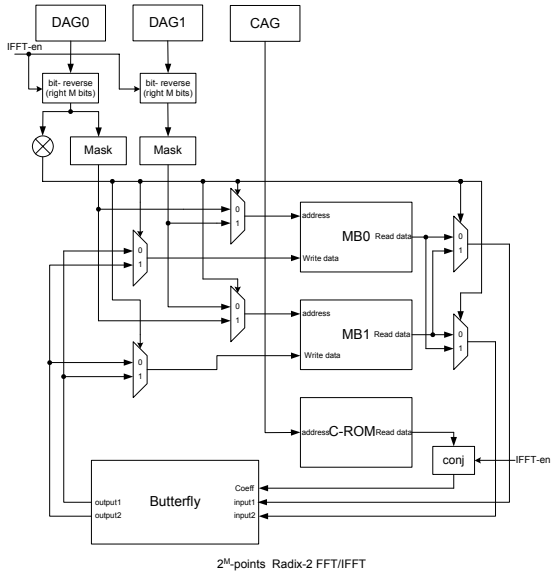


Figure 4: Dual-mode variable-size FFT/IFFT Architecture with conflict-free memory access mechanism

三、Symbol-based FDD-MCM FFT/IFFT 架構

隨著運用 MCM(Multi-Carrier Modulation)為方式之寬頻傳輸系統興起， 多點數之 FFT/IFFT 運算是基本的要求，若是直接將原始的 FFT/IFFT 運算法實踐於硬體， 則必定耗掉相當大的晶片空間成本及耗電量，然而幸運地，若是 MCM 以 symbol-based FDD(IEEE 802.16a 之 OFDMA 模式 及 DMT-based VDSL 皆為此類系統)為上下傳方式，我們將發展更有效率的演算法來節省運算量，並依此提出一個高效率 FFT/IFFT 硬體架構，此架構將運用於 OFDMA 及 DMT-based VDSL 實體層(PHY)接收器晶片上。下面將會逐步地描述此演算法及硬體架構。

- **FFT for Few Frequency Bins**

若 $x[n]$為一 discrete time sequence，假設有 N 點的資料，其 Discrete Fourier Transform (DFT)如 Eqn(1) 所示，若所需之輸出 FFT 點數為 P 點且 P 遠小於 N， 不同於 Sorensen[6, 7]在 1988 所提出的方式，我們提出另一修訂版本之演算，用此方法我們可以更有效率地且動態調整切斷方式來算出所需要的 FFT bins，詳細如下述。

將 N points FFT 分割為數個 L points 的 FFT，定義 $Q=N/L$， 則 P 點 DFT 可由下面方式算出

$$X[k]=\sum_{n_2=0}^{Q-1}X_{n_2}[<k>_L]\cdot W_N^{n_2k} \quad (6)$$

其中$<k>_p$的符號表示對 k 作 modulo-P 的運算。而我們可以看出內層中括號的運算式實際上就是一個 L 點的 FFT。

$$X_{n_2}[l]=\sum_{l=0}^{L-1}x[mQ+n_2]\cdot W_L^{ml} \qquad (7)$$

以上算式所代表的意義為: 我們可以將一較大點數的 FFT 分割成許多較小之 L 點數的 FFT 來作，而後再作一些後續的處理及可得到 P 的 FFT 輸出點。其好處在於可以不用把所有輸出點計算出來，只需算出所需要的點即可。圖-5 為此演算法之圖解。

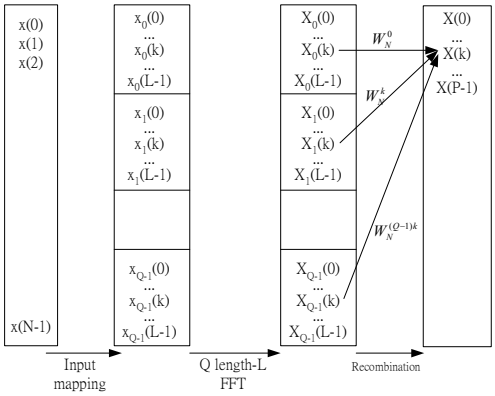


Figure 5: Computation of segmental FFT for few Frequency Bins

由圖-5 可知，若要完成此演算法，需完成以下步驟:

● IFFT for Few Non-Zero Frequency Bins

至於 IFFT，我們將提出下面的演算法來運算，若在 frequency 上只指定了 P (bin)個值在 frequency bin J ~ (J+P-1)，其餘為零，我們可以將原本要做 N 點 IFFT 的點分割成 L 等份，每份有 Q 點後，對下式變數變換。

$$n=n_1Q+n_2,\quad 0\le n_1\le L\text{-}1,\quad 0\le n_2\le Q\text{-}1 \qquad (8)$$

我們由 Eqn(1)可得

$$x[n]=\sum_{k=0}^{P-1}V_{J,n_2}[k]W_L^{-kn_1} \qquad (9)$$

其中 $V_{J,n_2}[k]=X[k+J]\cdot W^{-(Jn+kn_2)}$

假設 $P=L\times M$，我們可將 Eqn(9) 分解為 M 個長度 -L 的 IFFT:

$$x[n]=\sum_{m=0}^{M-1}\left(\sum_{k=0}^{L-1}V_{J,n_2}[mL+k]W^{-kn_1}\right) \qquad (10)$$

也就是說，欲算出 $x[n]$，可先將每個 $X[k+J]$乘上一個 $W_N^{-(Jb+kn_2)}$ ，再作 Q 個 L 點的 IFFT。

● 運算時間分析

對於架構設計方面，可以觀察到如果將一點數較多的 FFT 分割成許多點數較少的 FFT，則所需的 cycle 數目將會減少，亦即處理速度將會加快。但還需再對點數較少的 FFT 輸出之資料在進行後續的處理，如此方可得到正確的輸出。其大略架構如圖-6。

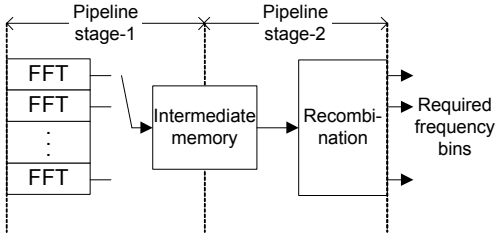


Figure 6: Pipeline processing of proposed FFT computation

這部分點數較小的FFT 的運算核心將使用前面提出之 variable-size FFT/IFFT 作為基本運算單元。對於一個 2^K 點的 FFT 而言，若採用前述之雙乘法器之 FFT/IFFT 運算核心，則需要 $K\times 2^K$ 個 clock cycles 來算出結果。因此對一個點數固定的 FFT 而言，例如 8192 點的 FFT，如果不分割成較小的 FFT，整各運算 FFT 的過程共需要 13×2^{13} cycles 才可算完。但若是把一個大點數的 FFT 分割成許多小的 FFT，例如分割成 32 塊 256 點的 FFT 來作，則只需要 8×2^8(cycles/block) $\times 2^5$(number of blocks) cycles 即可完成。而做完這些點數較小的FFT 後的資料必須再作一些處理方可得到正確的資料，在之後所提出的硬體架構上，我們暫時只用一個複數乘法器來實現 Eqn(10)所述之 recombination 的運算。我們並且把作 FFT 的部分和後續處理部份以兩級 pipeline 方式處理。

對於該如何分割點數較多的FFT 成為許多點數較少的FFT。我們針對 8192 點 FFT 做了以下的運算時間分析。

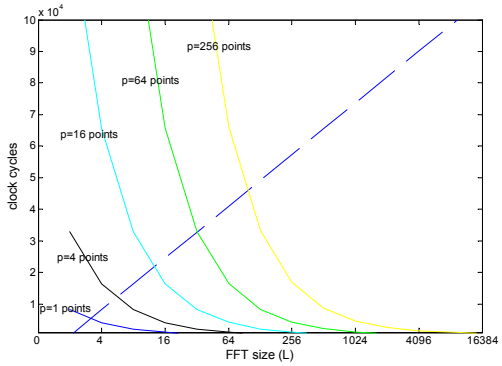
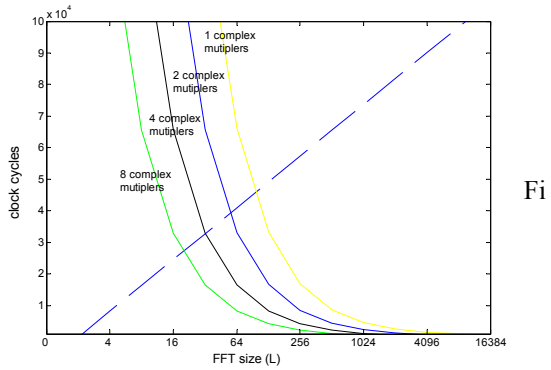


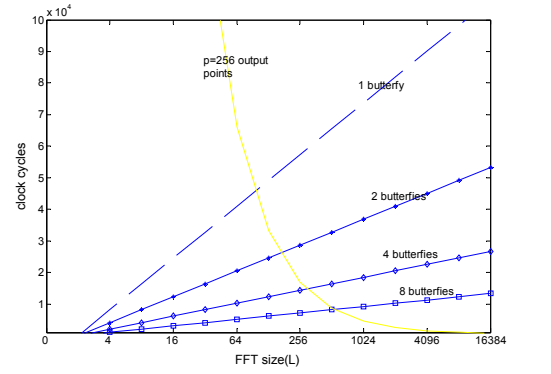
Figure 7: Number of clock cycles for execution vs partitioned FFT sizes when different numbers of frequency bins are required.

在圖-7 中橫軸代表所切割點數(L=2^K)以 2 為底的冪次方，而縱軸代表所需的 clock cycle 數目。虛線代表管線前級的所有的小點數FFT 隨著切割點數大小所需要的總 clock cycle 數目，此數量等於 $K\times 2^K$(cycles/number)$\times 2^{13-K}$(number of blocks)= $K\times 2^{13}$，呈線性變化。而標有 R=0，2，4，6，8，10 的曲線，代表管線後級所需不同輸出點數之 clock cycles 變化的曲線，其中 R 代表所需點數以 2 為底的冪次方， 例如 R=4 表示所需點數為 2^4=16 點。由此圖中可得知，若是只需要 256 點的輸出，則對應到 R=8 的曲線，為了使兩級之管線運算時間(以 clock cycle 數為單位)達到最小化，我們須依照曲線和斜線交點決定之 FFT 大小做切割，因此管線兩級會花費同樣的 clock cycle 數。

由此圖中我們可以看到，以現行的 VDSL 標準為例，如果不做較佳化之少點數 FFT 的分割，最有可能使用 4096 點的 FFT，那麼所需的 clock cycles 為 49152 個 clock cycle。但若是只需要 1024 的 FFT 輸出 bins，或者更少，則可根據此圖對點數較大的 FFT 的做分割，而使得在較少的 clock cycles 之內完成運算。此外，由圖 8 可知，若我們用較多的乘法器來供 FFT/IFFT 運算核心或 recombination 運算核心使用，所需的運算時間將可以更進一步地減少。在圖-8(a)中，不同曲線表示的意義為：在負責做 recombination 動作的後級管線，其中所包含的複數乘法器分別為一個、二個、四個、八個複數乘法器。由圖中可知，若後級管線所包含的複數乘法器越多，在固定輸出點數的條件下，其運算所需的 clock cycle 數越少；如圖-8(b)所示，若後級管線所包含的複數乘法器越多，則和斜線的交點越低，則運算所需的 clock cycle 數就越低。同樣地，若前級管線所負責的 FFT/IFFT 運算核心具較多個 butterfly 處理單元，我們將需更少的 cycle 數，如圖?所示，曲線代表一固定的輸出點數(2^8=256 點)，而斜線表示的意義為 FFT 計算核心中所包含的 butterfly 數目。如圖所示，虛線表示 FFT 計算核心中包含一個 butterfly，而由菱形所構成的斜線，其 FFT 計算核心中包含四個 butterfly。由本圖可知，若 FFT 計算核心中所包含的 butterfly 數目越多，則負責計算點數較少 FFT 的前級管線所需的的 clock cycle 數就會減少。如前所述，若把管線 clock cycle 數依照斜線與曲線的交點而分配，則整體所需之 clock cycle 數目可以降的更低。



(a)



(b)

Figure 8: Number of clocks cycles of two pipeline stages with different segmentation and computation resource arrangements (a) different numbers of multipliers for recombiniation in Stage-2 pipeline; (b) different numbers of butterfly processing units for Stage-1 pipeline.

- **Dynamic Block Size Adjustment**

由上述分析，我們可由 FFT/IFFT 大小、所用到的 Frequency bin 數量及 available 乘法器數量，來找到一個最佳的 block size (L)，若所應用的傳輸系統具有 adaptive modulation 的機制來調整 Frequency bin 數量，我們則可以動態地調整 block size，使得所需的 clock cycles 數最少，更方面達到 real-time 運算，為滿足此彈性運算，上一節所提出的大小可調之 FFT/IFFT 處理核心是必須的。

- **硬體架構設計**

在硬體電路架構中，若採用 Goertzel 演算法[8]來完成後級管線 recombination 運算，將可以僅用實數乘法而因此減少約一半的乘法運算量，但此方式卻需要大空間的記憶體來存中間過程的資料，所以我們仍採用原本使用複數乘法的方式，為減少因複數乘法器所造成的硬體成本，我們用 CORDIC 實現複數乘法[9]，CORDIC 有許多加快運算速度的方法，有可能使運算所需之 clock cycle 數更為減低。圖-9 為高效率 Symbol-based FDD Dual-mode Variable-Size/Variable-Frequency-Bin FFT/IFFT 架構圖。

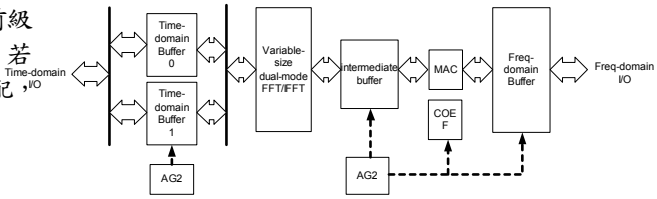


Figure 9: Efficient Symbol-based FDD Dual-mode Variable-Size/Variable-Frequency-Bin FFT/IFFT Architecture

其運作行為概述如下:

FFT:

由於FFT之輸入為bit reversal，故需要一個address generator (AG1)來抓取 input buffer 內的輸入值至適當的內部記憶體位址。此內部記憶體的大小等於點數較小的 FFT 的點數大小。接著由 FFT 算出結果後，其輸出已呈現 in order 的排列。在由 AG1 產生適當位址存入記憶體中；然後再第二級管線之中，由另一個 address generator(AG2) 來抓取記憶體中的資料，正確的係數，以及在 output buffer 中的暫存輸出；AG2 抓取資料的順序是根據 Eqn(10)決定的。

當前一級的小點數 FFT 輸出計算出來後，將每個輸出點乘上適當的 $W_N^{n_2 k}$ ，而後存於 output buffer，之後小點數的 FFT 不斷的被送入記憶體；此即管線也不斷的重複這個動作，直至所有的小點數 FFT 被計算出來後，此時 output buffer 也就是所想要的 FFT 輸出點了。以下為 IFFT 的架構圖。

IFFT:

其運作行為基本上為FFT架構圖的反向運作，由右邊輸入左邊輸出。在輸入所碰到的第一層管線，由AG2產生正確的係數位址以及輸出所擺放在記憶體的位址，而下一級的IFFT藉由AG1抓取記憶體中的值擺放至IFFT中內部記憶體內，同樣的，IFFT 的輸入也是 bit reversal 的放置方法。當作完點數較小的IFFT後，由AG1產生正確的位址放入output buffer。

四、結論:

本論文針對以 symbol-based FDD OFDM 及 DMT 傳輸系統所需之FFT及IFFT,提出有效的演算法及硬體架構,我們首先設計了可架構雙模式可變大小之FFT/IFFT處理器，並由此為核心，更進一步提出應用於上述系統FDD 之高效率低成本 FFT/IFFT 運算架構，比較起直接採用大點數 FFT 及 IFFT 的方式，所提出之架構在所需之運算時間及硬體成本上，可大幅度的減少，應用於如IEEE 802.16a (OFDMA)及VDSL系統晶片上。

[1] IEEE standard 802.18 - *Part*- 16: *Air interface for Fixed Broadband Wireless Access Systems*, 2002

[2]Committee T1-telecommunications, *Very-high-bit-rate Digital Subscriber Line (VDSL) Metallic Interface – Part 1: Functional Requirements and Common Specifications*, T1E1.4/2002-031R2, February 2002.

[3] E. H. Wold and A. M. Despain, "Pipeline and parallel-pipeline FFT processors for VLSI implementation," *IEEE Trans. Computer*, C-33(5), pp. 414-426, May 1984.

[4] S. He and M. Torkelson, "Design and Implementation of a 1024-point Pipeline FFT Processor," *IEEE Proc. Custom Integrated Circuit Conference*, 1998.

[5] L. G. Johnson, "Conflict Free memory Addressing for Dedicated FFT Hardware," *IEEE Trans. Circuits and Systems-II: Analog and Digital Signal Processing*, vol. 39, pp. 312-316, May 1992.

[6] H. V. Sorensen and C. S. Burrus, " A new efficient algorithm for computing a few DFT points," IEEE Proc. Int. Symp. Circuits and Systems, vol. 2, pp.1915-1918, 1988.

[7] H. V. Sorensen and C. S. Burrus, "Efficient computation of the DFT with only a subset of input or output points," *IEEE Trans. Signal Processing*, vol. 41, pp. 1194-1200, March 1993.

[8] A. V. Oppenheim and R. W. Schafer, *Discrete Time Signal Processing*, Englewood Cliffs, NJ: Prentice-Hall, 1992.

[9] A. M. Despain, "Fourier Transform Computers using CORDIC Iterations," *IEEE Trans. Computer*, vol. C-23, pp.993-1001, Oct. 1974.