行政院國家科學委員會補助專題研究計畫成果報告

※※※※※※※※※※※※※※※※※※※※※※※※※

單晶片第四代寬頻無線通信系統設計技術之研究–子計畫三
高效能之快速傅立業轉換演算法、架構設計及其在數位通信上的應用

Design of High-Performance FFT/IFFT and Its Applications to Digital Communication

※※※※※※※※※※※※※※※※※※※※※※※※※
※

計畫類別：☐個別型計畫　　■整合型計畫

計畫編號： NSC90 -2218-E-009-038

執行期間：90 年　8 月　1　日至　91　年　7　月　31　日

計畫主持人：陳紹基，教授，國立交通大學電子研究所

共同主持人：

計畫參與人員：曲建全、張朝凱、林孟學、周俊義、洪崇斌、陳坤隆

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：交通大學電子研究所

中　華　民　國　91　年 10　月 24　日

行政院國家科學委員會專題研究計畫成果報告

單晶片第四代寬頻無線通信系統設計技術之研究-子計畫三 高效能之快速傅立業轉換演算法、架構設計及其在數位通信上的應用

Design of High-Performance FFT/IFFT and Its Applications to Digital Communication

計畫編號：NSC90 -2218-E-009-038
執行期限：90 年 8 月 1 日至 91 年 7 月 31 日
主持人：陳紹基，教授，國立交通大學電子研究所
計畫參與人員：曲建全、張朝凱、林孟學、周俊義、洪崇斌、陳坤隆，國立交通大學電子研究所

一、中文摘要

本計畫主要目標是針對正交分頻多工通信系統中基頻收發機有關快速傅立業及反快速傅立業轉換處理器的研究與設計，在本計畫中，我們完成了數項成果：(1) 各種快速傅立業轉換演算法的分析與比較，(2) 各種快速傅立業轉換架構的分析與比較，(3) 提出一可變長度、定位(in place)單一 FFT 處理器架構，(4) 提出非定位一 FFT 處理器架構，(5) 提出兩個 FFT 係數(Twiddle factors)合成器架構

關鍵詞：正交分頻多工、快速傅立業轉換、FFT 係數

Abstract

This project is “Investigation and Design of FFT Core for OFDM Communication System.” The main object is to investigate and design an efficient FFT/IFFT core for OFDM communication systems. This is the first-year project. This project accomplishes several results including: (1) analyses and comparisons of several FFT algorithms, (2) analyses and comparisons of several FFT architectures, and (3) a variable-length memory-based FFT/IFFT architecture is proposed.

Keywords: Orthogonal Frequency Division Multiplexing (OFDM), Fast Fourier Transform (FFT)，Twiddle factors

二、計畫緣由與目的

近年來由於多媒體的需求與 DSP 跟 VLSI 製程進步的關係，正交分頻多工調變技術(OFDM)再度引起廣泛的關注。在有線通訊方面，如：ADSL、VDSL，在無線通訊方面，如：DAB、DVB、802.11a 等等，均是利用此一調變方法來達到高傳輸效率的目的。但在 OFDMSystem 中，FFT Module 不論在計算上或硬體上的複雜度都非常龐大，所以 Low Power 與 Low Cost 的考量是必要的。我們的研究方向主要是針對各種快速傅立葉演算法去分析比較其算術複雜度，並對不同的架構做硬體效率、硬體成本和速度上的比較，以求盡可能的達到低功率與低成本的要求。除此之外，又因為不同的應用其所需的 FFT 點數大小不盡相同。所以去設計一個可適用於各種應用的 FFT 處理器也是我們研究的重點之一。我們亦引用了幾種可用來簡化硬體複雜度的設計技巧，如：位元長度的模擬、快速傅立業與反快速傅立業處理器的硬體共用、係數暫存器的縮小等等。

三、結果與討論

(1)FFT 演算法的分析比較

快速傅立業轉換(FFT)主要是利用遞迴分解(recursive decomposition)來降低複數乘法運算量，以求達到快速運算、低功率消耗的目的。此演算法基本上分為兩大類型，一類為分時(DIT)演算法，另一類則為分頻(DIF)演算法。由於此兩種演算法的運算複雜度一樣，所以我們只針對分頻演

算法作詳細的分析與比較。表一為幾種不同基數(radices)的 FFT 演算法之乘法、加法運算複雜度的比較。

表一：計算複雜度的比較

Complexity / Algorithm		No. of real multiplications	No. of real additions
Radix-2		$\frac{3N}{2}\log_2 N-\frac{7}{2}N+8$	$\frac{5N}{2}\log_2 N-\frac{7}{2}N+8$
Radix-4		$\frac{9N}{8}\log_2 N-3N+3$	$\frac{25N}{8}\log_2 N-3N+3$
Radix-8	Case1*	$\frac{25N}{24}(\log_2 N-3)+4$	$\frac{73N}{24}\log_2 N-\frac{25N}{8}+4$
	Case2*	$\frac{21N}{24}\log_2 N-\frac{25N}{8}+4$	$\frac{8t+73N}{24}\log_2 N-\frac{25N}{8}+4$
Split-radix-2/4		$N\log_2 N-3N+4$	$3N\log_2 N-3N+4$

由表一可知 split-radix 演算法有最少的計算量 但因為它的 twiddle factors 出現較不規則 所以較不利於 ASIC 設計; 相對地 high-radix 演算法也有低的計算量 但不適用於所有 2^n 點的 FFT 所以 low-radix 演算法仍然適用。

(2)FFT 架構的分析比較

用來實現FFT的架構基本上也是分為兩大類型，一類為管線化(pipeline based)架構，另一類則為記憶體(memory based)架構。

(a) 管線化架構

就管線化的架構來說，其特色是架構規則、控制訊號簡單，所以非常適合 VLSI 的實現，但相對於記憶體架構來說，其所需的硬體面積也較大(因為需要 $\log_r N$ 個 PE)。表二、三分別列出幾種不同基數的管線化架構之所需的硬體量與其使用率。

表二：硬體需求的比較

Architecture	No. of Complex Multipliers	No. of Complex Adders	Complex Memory Size	Multiplicative Complexity
R2SDF	$\log_2 N-2$	$2\log_2 N$	N-1	Radix-2
R4SDF	$\log_4 N-1$	$3\log_4 N$	N-1	Radix-4
R2^{2}SDF	$\log_4 N-1$	$4\log_4 N$	N-1	Radix-4
R8SDF	$\log_8 N-1$	$(24+2T)\log_8 N$	N-1	Radix-8
R2^{3}SDF	$\log_8 N-1$	$(6+T)\log_8 N$	N-1	Radix-8
SRSDF	$\log_4 N-1$	$4\log_4 N$	N-1	Split-Radix
R2MDC	$\log_2 N-2$	$2\log_2 N$	1.5N-2	Radix-2
R4MDC	$3\log_4 N-3$	$4\log_2 N$	2.5N-4	Radix-4
R2^{2}MDC	$\log_2 N-2$	$2\log_2 N$	1.5N-2	Radix-4
R8MDC	$7\log_8 N-7$	$(24+2T)\log_8 N$	4.5N-8	Radix-8
R2^{3}MDC	$2\log_8 N-2$	$(6+T)\log_8 N$	1.5N-2	Radix-8
SRMDC	$\log_4 N-1$	$4\log_4 N$	1.5N-2	Split-Radix

表三：硬體使用率的比較

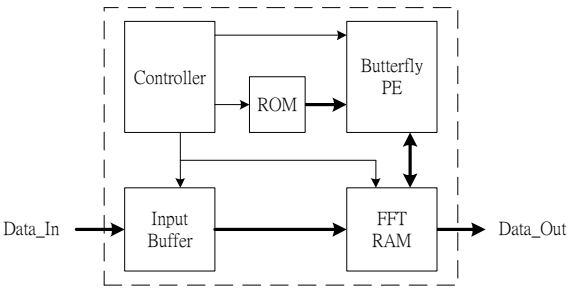
Utilization rate / Architecture	Multipliers	Adders	Registers
R2SDF	50%	50%	100%
R4SDF	75%	25%	100%
R8SDF	87.5%	12.5%	100%
R2^{2}SDF	75%	50%	100%
R2^{3}SDF	87.5%	50%	100%
SRSDF	≤ 75%	50%	100%
R2MDC	50%	50%	50%
R4MDC	25%	25%	25%
R8MDC	12.5%	12.5%	12.5%
R2^{2}MDC	37.5%	50%	50%
R2^{3}MDC	43.5%	50%	50%
SRMDC	≤ 37.5%	50%	50%

由表二、三可知 架構(SDF)不論在硬體的需求上或使用率上均優於多重路徑延遲回授架構(MDC)。就現有的應用系統來說 因為 throughput rate 的關係 SDF 架構仍然比 MDC 架構適用。

(b) 記憶體架構

一般的記憶體架構如圖一所示 利用較少(或單一個)的處理單元(PE)來執行 FFT 的運算 所以其所需的硬體較管線化架構還少。但也因為這樣

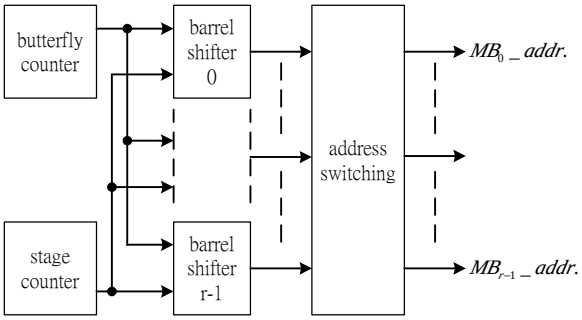
。



圖一：一般的記憶體架構圖

此一缺點可以利用多重記憶體單元(multi-bank memory units)加上適當的位址

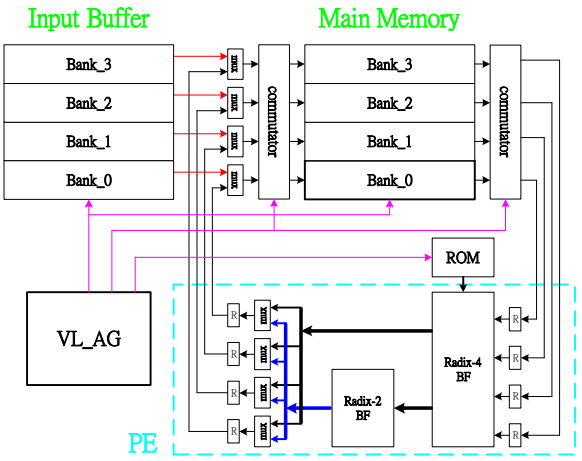
產生器(AG)的方式來提高資料量的存取。關於如何去設計一個簡單的位址產生器(如圖二所示)，可參考文獻[6]。



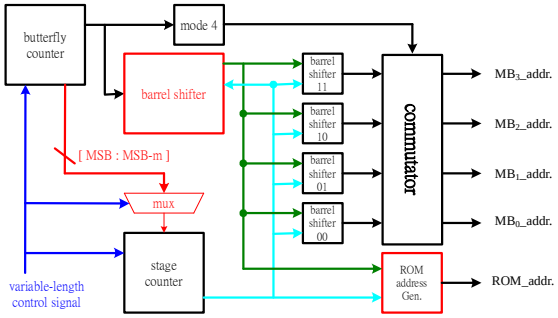
圖二：位址產生器之架構圖[6]

(3)可變長度單一處理器架構

因為不同的應用(系統)其所需的 FFT 點數大小不盡相同。所以去設計一個可適用於各種應用的FFT處理器就顯的相當有價值。在經過評估之後，發現記憶體架構是一個不錯的選擇，因為它具有低成本的好處，而且只要經由簡單的修正原本的位址產生器(圖二)，便可讓此種架構具備處理不同長度的 FFT 運算。我們所提出的可變長度單一處理器架構如圖三所示，而圖四則為此一架構中用來產生適當位址的可變長度位址產生器(VL_AG)。



圖三：可變長度單一處理器架構



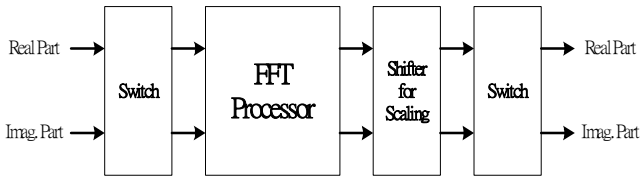
圖四：可變長度位址產生器架構(VL_AG)

(4)簡化硬體複雜度

幾個在此架構中簡化硬體的設計方法如下：

(a) FFT 與 IFFT 硬體的共用

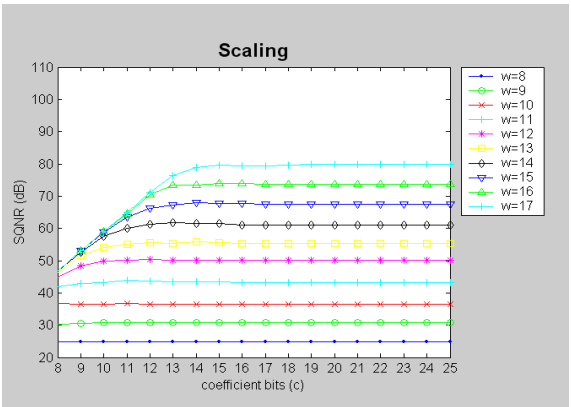
由於 OFDM 系統的收發機(transceiver)是分別利用 IFFT 與 FFT 來調變與解調變其所要傳送與接收的資料，為了降低成本，因此 FFT 與 IFFT 硬體的共用是必要的。其做法如下圖所示，只需兩個額外的交換器(switch)和一個位移器即可。



圖五：FFT/IFFT 硬體共用架構圖

(b) 位元長度的決定

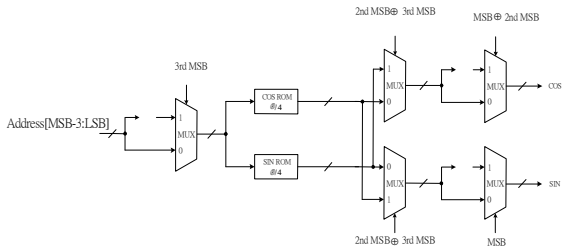
雖然用來表示資料的位元長度越多則準確性越高，但相對的其所需的硬體也隨之大增，所以適當的選擇資料位元長度在可允許的精確度之內是必要的。圖六為精確度對位元長度在 scaling(防止溢位發生)做法下的相對圖。



圖六：精確度 vs.位元長度

(c) 係數暫存器(ROM)的縮小

因為 FFT 運算所需要的係數為正弦或餘弦，所以我們可以利用它的對稱性質再加上一些小電路(如圖七所示)來減少係數必須存放在係數暫存器的數量。這個做法在 FFT 的點數越大時效果越顯著。



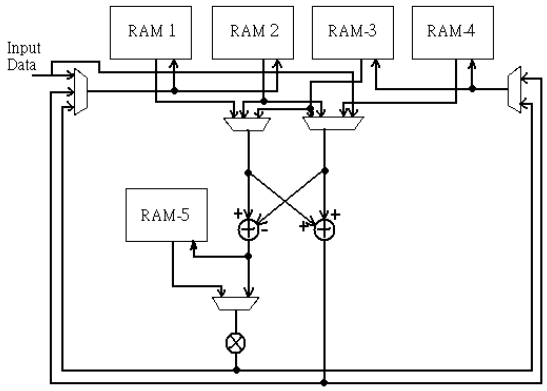
圖七：八分之一週期正弦/餘弦的儲存技術

(d) 複數乘法器的設計

兩個複數相乘最直接的做法需要四個乘法器、兩個加法器，但經由化簡之後用三個乘法器、四個加法器即可。

(5)一個使用少量記憶體之非定位(non in place)之 FFT 處理器架構

由於非定位之 FFT 架構有控制簡單及可重疊資料讀寫之優點，因此相較於前述定位(in place)架構，雖然記憶體需求較大但仍有其訴求點。[11]所提出之架構雖有不錯之效能但記憶體量稍多，如下我們提出之架構可維持同樣之效能但記憶體減少甚多



相對的效能比較如下表

	Memory size	Average throughput rate	Multiplier Utilization
Improved	1.25N	$1/(n\log n+n/2)$	100%
[11]	2.5N	$1/(n\log n+n)$	100%

此架構之運算步驟如下：

Step 1. Load $x[0], x[1],\ldots,x[n/4-1]$ to RAM-1;
Then, load $x[n/4], x[n/4+1],\ldots,x[n/2-1]$ to RAM-2.

Step 2. Read $x[k]$ from address k of RAM-1, and $x[n/2+k]$ from external input buffer and send them to butterfly PE, and perform
$a[k]=x[k]+x[n/2+k]$,
$b[k]=(x[k]-x[n/2+k]) \times W_N^k$,
$k=0,1,2,\ldots,n/4-1$.
Store $a[r]$ in address r of RAM-1, and
$b[r]$ in address r of RAM-3,
$r=0,1,2,\ldots,n/4-1$.
Then, read $x[k]$ from address $k-n/4$ of RAM-2, and $x[n/2+k]$ from external input buffer and send them to butterfly PE, and perform
$a[k]=x[k]+x[n/2+k]$,
$b[k]=(x[k]-x[n/2+k]) \times W_N^k$.
$k=n/4,n/4+1,n/4+2,\ldots,n/2-1$.
Store $a[r]$ in address $r-n/4$ of RAM-2, and
$b[r]$ in address $r-n/4$ of RAM-4,
$r=n/4,n/4+1,\ldots,n/2-1$..

Step 3. Read $a'[k]$ from address k of RAM-1 and $b'[k]$ from address k of RAM-2 and send them to butterfly PE, and perform $a[k]=a'[k]+b'[k]$, $b[k]=a'[k]-b'[k]$, $k=0,1,2,\ldots,n/4-1$.
Store $a'[r]$ in address r of RAM-1, $r=0,1,2,\ldots,n/8-1$.
Store $a'[r+n/8]$ in address r of RAM-2, $r=0,1,2,\ldots,n/8-1$.
Store $b'[r]$ in address r of RAM-5, $r=0,1,2,\ldots,n/4-1$.

Step 4. $c[k]=b[k] \times W_{N/2}^k$, $k=0,1,2,\ldots,n/4-1$.
Store $c[r]$ in address r of RAM-1, $r=0,1,2,\ldots,n/8-1$.
Store $c[r+n/8]$ in address r of RAM-2, $r=0,1,2,\ldots,n/8-1$.
Read $a'[k]$ from address k of RAM-3 and $b'[k]$ from address k of RAM-4 and send them to butterfly PE, and perform $a[k]=a'[k]+b'[k]$, $b[k]=a'[k]-b'[k]$, $k=0,1,2,\ldots,n/4-1$.
Store $a[r]$ in address r of RAM-3, $r=0,1,2,\ldots,n/8-1$.
Store $a[r+n/8]$ in address r of RAM-4, $r=0,1,2,\ldots,n/8-1$.
Store $b[r]$ in address r of RAM-5, $r=0,1,2,\ldots,n/4-1$.

Step 5. $c[k]=b[k] \times W_{N/2^{t-1}}^k$, $k=0,1,2,\ldots,n/4-1$, $t=3,4,\ldots,n$.
Store $c[r]$ in address r of RAM-3, $r=\{y|[y/(n/2^t)]\bmod(2)=0\}$, $t=3,4,\ldots,n$.
Store $c[r]$ in address $r-n/2^t$ of RAM-4, $r=\{y|[y/(n/2^t)]\bmod(2)=1\}$, $t=3,4,\ldots,n$.

Read $a'[k]$ from address k of RAM-1 and $b'[k]$ from address k of RAM-2 to butterfly PE, and perform $a[k]=a'[k]+b'[k]$, $b[k]=a'[k]-b'[k]$, k=0,1,2,…,n/4-1.

Store $a[r]$ in address r of RAM-1, r={y|[y/(n/2^t)]mod(2)=0}, t=4,5,…,n.

Store $a[r]$ in address r-n/2^t of RAM-2, r={y|[y/(n/2^t)]mod(2)=1}, t=4,5,…,n.

Store $b[r]$ in address r of RAM-5, r=0,1,2,…,n/4-1.

Step 6. $c[k]=b[k] \times W_{N/2^{t-1}}^{k}$, k=0,1,2,…,n/4-1, t=4,5,…,n.

Store $c[r]$ in address r of RAM-1, r={y|[y/(n/2^t)]mod(2)=0}, t=4,5,…,n.

Store $c[r]$ in address r-n/2^t of RAM-2, r={y|[y/(n/2^t)]mod(2)=1}, t=4,5,…,n.

Read $a'[k]$ from address k of RAM-3 and $b'[k]$ from address k of RAM-4 to butterfly PE, and perform $a[k]=a'[k]+b'[k]$, $b[k]=a'[k]-b'[k]$, k=0,1,2,…,n/4-1.

Store $a[r]$ in address r of RAM-3, r={y|[y/(n/2^t)]mod(2)=0}, t=4,5,…,n.

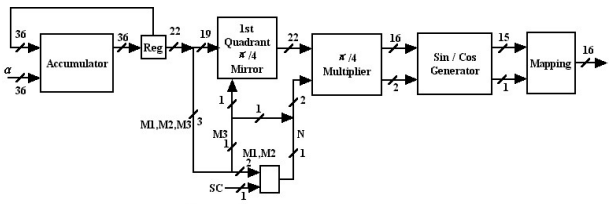
Store $a[r]$ in address r-n/2^t of RAM-4, r={y|[y/(n/2^t)]mod(2)=1}, t=4,5,…,n.

Store $b[r]$ to address r of RAM-5, r=0,1,2,…,n/4-1.

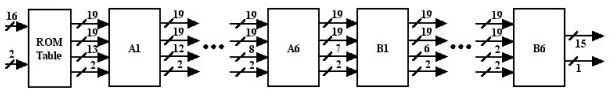
(6)兩個高效能 FFT 係數產生器

(a) 管線化之架構

整體架構：



其中 Sin/cos 產生器架構如下圖：



運算步驟：

Stage1: The Accumulator block is for the incremental generation (at a step of F_{clk}~/2^M) of the input binary $B=f\cdot t$ values to the DDFS , where M=36 is assumed here, which controls the resolution of the output frequency, F_{clk} is the frequency of the system clock, "~" is the frequency control word, and B is composed of 3 integer bits and 33 fractional bits..

Stage2: The 1st Quadrant Mirror block performs symmetry reduction of the input phase to $\pi/4$, for reducing table size. This stage strips off the integer part of the input phase. In Fig. 2, only 22 MSB's of the accumulator's output is retained for lower complexity consideration.

Stage3: The $\pi/4$ Multiplier block maps its fractional input to the phase range of $[0,\pi/4]$, using the equation $\bar{}=B\cdot\pi/4=0.a_1a_2\cdots a_N$.

Stage4: The Sin/Cos Generator block realizes the 2nd-order interpolation algorithm detailed in the previous section. It consists of a lookup ROM table, Ai and Bi blocks, as will be detailed next.

Stage5: The Mapping block is for the final sign correction of the results.

ROM Table: This block stores all the eight (i.e., $2^{N/4-1}$) cosine values addressed by the three fractional MSB bits $0.a_1a_2a_3$ of $\bar{}$, for the retrieval of $\cos(0.a_1a_2a_3+0.001)$ and $\cos(0.a_1a_2a_3)$.

A1 stage: The A1 block solves $\cos(0.a_1a_2a_31)$, using

$$[1+\frac{1}{2}(\Delta\bar{}_4)^2][\cos(0.a_1a_2a_3)+\cos(0.a_1a_2a_3+0.001)]/2$$
$$=(1+2^{-9})[\cos(0.a_1a_2a_3)+\cos(0.a_1a_2a_3+0.001)]/2$$

Note that $\Delta\bar{}_4=2^{-4}$ (i.e., $\Delta\bar{}_{N/4}=2^{-N/4}$). This stage requires two additions.

A2 stage: The A2 block checks bit a_4 to decide if $\bar{}$ lies in the phase ranges of $[0.a_1a_2a_30,0.a_1a_2a_31)$ or $[0.a_1a_2a_31,0.a_1a_2a_3+0.001)$. Then, similar interpolation operation as in A1 stage is done to find $\cos(0.a_1a_2a_3a_41)$, where $\Delta\bar{}_5=2^{-5}$. So do the pipeline stages A3 to $A_{N/4}$=A_4.

B1 to B9 stages (i.e., 1 to N/2+1=9):

These pipeline stages have similar operations to that of Ai stages, except that here $(\Delta\bar{}_i)^2/2\le 2^{-(N+1)}=2^{-17}$ can be omitted. Hence, the interpolation reduces to a single addition. The B9 stage is the output

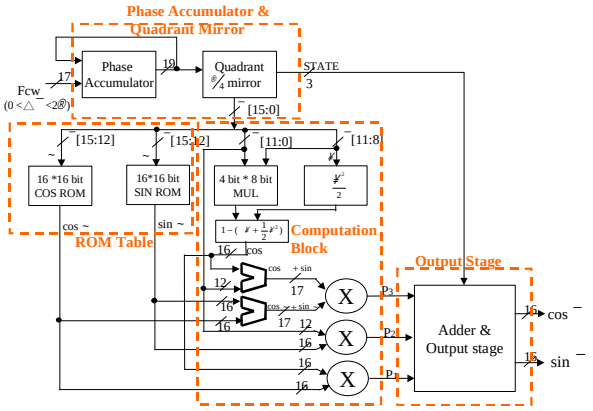
stage.

與現有最佳設計之效能比較：

	ROM size	No. of adders
New desgin	$2^{N/4-1}$	N
[12]	$2^{N/3}$	$3N/2$

(b) 非管線化之架構

整體架構：



與現有設計之效能比較：

Method	New design	Conventional Taylor series Approximation	[12]
ROM size	$2^{N/4+1}$	$2^{N/3+1}$	$2^{N/3}$
Adders	3	4	$3N/2$
Multiplier	4	4	0
	(N+1) (N+1)*	All (N/2)*N	
	N*N		
	$3N/4 * N$		
	$3N/4 * 3N/4$		
Equivalent Adder No.	3N/2+3	$N+4$	$3N/2$

四、計畫成果自評

研究結果大部完成預定目標，但仍待應用之設計如考慮到 DAV/DVB 之設計參數。未來的目標如下：(1) 針對 FFT 來設計一個更有效率的處理單元(PE) 如：乘法器;(2) 利用資料壓縮或內建一個係數產生器來進一步的降低 ROM table size;(3) 利用資料壓縮或數值表示法轉換的概念降低運算時所需要的功率;(4) 結合系統之 IC 實現。

五、參考文獻

[1] Yeong-Terng Lin, Design and Implementation of a Variable-Length FFT Processor for OFDM Systems, NTU, Master Thesis, June 2001.

[2] A. V. Oppenheim R. W. Schafer, Discrete-Time Signal Processing, Prentice-Hall Inc., 1999.

[3] Pierre Duhamel, "Implementation of Split-Radix FFT Algorithms for Complex, Real, and Real-Symmetric Data," IEEE Transactions on Acoustics, Speech and Signal Processing, Vol. ASSP-34 No. 2, April 1986.

[4] Jun-Jie Fang, Design of FFT Processor, NCTU, Master Thesis, August 2001.

[5] Shousheng He and Mats Torkelson, "Designing Pipeline FFT Processor for OFDM (de) Modulation," URSI International Symposium on Signals, Systems and Electronics, pp. 257-262, 1998.

[6] Hsin-Fu Lo, Ming-Der Shieh, and Chien-Ming Wu, "Design of an Efficient FFT Processor for DAB System," IEEE International Symposium on Circuits and Systems, Vol. 4, pp. 654 –657, 2001.

[7] L. G. Johnson, "Conflict Free Memory Addressing for Dedicated FFT Hardware," IEEE Transactions on Circuit and System-II: Analog and Digital Signal Processing, Vol. 39 No.5, pp.312-316, May 1992.

[8] Shousheng He and Mats Torkelson, "Design and Implementation of a 1024-point FFT Processor" in Proc. IEEE Custom Integrated Circuit Conference, pp. 131-134, 1998.

[9] Wen-Chang Yeh, Arithmetic Module Design and its Application to FFT, NCTU, Doctor Thesis, August 2001.

[10] Lihong Jia, Yonghong Gao and Hannu Tenhunen, " A Pipelined Shared-Memory Architecture for FFT Processors," 42nd Midwest Symposium on Circuits and Systems, Vol. 2 pp. 804 –807, 2000.

[11] Chin-Liang Wang and Ching-Hsien Chang, "A DHT-based FFT-IFFT processor for VDSL transceivers," Acoustics, Speech, and Signal Processing, 2001. Proceedings. 2001 IEEE international Conference on, Volume:2, Pages:1213-1216, 2001.

[12] A. Madisetti, A. Y. Kwentus, and A. N. Willson, Jr., "A 100-MHz, 16-b, Direct Digital Frequency Synthesizer with a 100-dBc Spurious-Free Dynamic Range," *IEEE Journal of Solid-State Circuits*, vol. 34, no. 8, pp. 1034-1043, Aug. 1999.