國 立 交 通 大 學

電 機 與 控 制 工 程 研 究 所

碩 士 論 文

用於多輸入輸出無線通訊之快速傅立葉轉換加速器設計

Design of FFT Accelerator for MIMO Wireless Communication Standards

研究生：呂俊衛

指導教授：董蘭榮　博士

中華民國九十六年九月

Design of FFT Accelerator for MIMO Wireless Communication Standards

Advisor: Dr. Lan-Rong Dung

Graduate Student: Chun-Way Lyu

September 2007

Graduate Institute of Electrical and Control Engineering

National Chiao Tung University

Hsinchu, Taiwan, ROC

Design of FFT Accelerator for MIMO Wireless Communication Standards

Graduate Student: Chun-Way Lyu Advisor: Dr. Lan-Rong Dung

Department of Electrical and Control Engineering

National Chiao Tung University

Abstract

FFT module is an indispensable part for wireless and mobile communication, especially when broadband wireless systems require a high speed and low power hardware module for its packet-based high-speed data transfer. This has made the design of FFT processor a critical requirement for the next generation wireless systems. In general, FFT module is designed for specific system. Therefore, it is desirable to design an adaptive FFT module for different standards. This thesis adopts processor flexible characteristic and ASIC accelerated mechanism to set up a flexible FFT module which can meet IEEE 802.11n/16e standards. Besides, we propose optimized time schedule in the SISO/MIMO systems. After processor computational analysis, 64-point branch FFT of ASIC can be applied in proposed system and it computes 16-bit input data at the 85 MHz throughput rate. After that, we compare various pipeline-based FFT architectures of ASIC and analyze their characteristic. Finally, it not only verifies the 64-point branch FFT on FPFA, but also checks proposed time schedule which can satisfy IEEE 802.11n/16e specification.

用於多輸入輸出無線通訊之快速傅立葉轉換加速器設計

學生：呂俊衛　　　　指導教授：董蘭榮　博士

國立交通大學

電機與控制工程學系研究所

摘要

對於無線和行動通訊系統，傅立葉轉換模組是不可或缺的部分，特別是當寬頻無線系統需要一個高速且低功率硬體於高速封包式資料傳輸，這使得傅立葉轉換成為下一代無線系統必要的需求。一般而言，傅立葉轉換模組的設計會針對特定的系統，因此，希望能去設計一個可以適合不同標準規格的傅立葉轉換模組。在此論文中採用處理器彈性的特色和硬體具有加速的機制去建立一個傅立葉轉換模組，並且可以符合 IEEE 802.11n/16e 的規格要求。除此之外，我們提出對於單輸入輸出/多輸入輸出系統的最佳排程。在經過處理器運算量分析後，64 點分支傅立葉轉換以硬體實現於系統中，並且它已 16 位元及 85 MHz 產出率為規格。之後，我們有針對用於系統的傅立葉轉換硬體架構做比較與分析其特性。最後，不只有對 64 點分支傅立葉轉換於 FPGA 上做驗證，並且有針對所提出的排程做驗證是可以滿足 IEEE 802.11n/16e 的規格。

誌謝

本篇論文得以順利完成，首先要感謝的是我的指導教授——董蘭榮教授，於碩士班兩年內，對於我的指導，尤其是當遇到一些研究上的疑問時，老師都能不厭其煩地給予方向與意見，使我能修正錯誤觀念，才能完成此篇碩士論文。並且提供非常豐富的資源，讓我能更方便於學習研究上，使我在碩士班期間內受益良多。

特別要感謝實驗室學長——學之、盟淳，在研究中給予的意見與幫忙，以及我的同學們——峻徹、仕捷、信丞、致惟，在這兩年內，於課業與生活上的互相扶持照顧，給予我一段美好的研究時光。

最後要感謝我的家人和女友——秀娟的支持，有了你們的鼓勵，讓我可以繼續面對下一個問題，並且無後顧之憂的完成碩士學位。

謹將此論文獻給所有關心我的人，在此致上最深的謝意。

Contents

List of Tables

List of Figures

Chapter 1　Introduction

1.1　Motivation

FFT module is one of the most utilized operations in digital signal processing and communications. The FFT and its inverse transform-IFFT are key component in modern communication systems. It is desirable that FFT module can flexibly adjust FFT size to meet various standards. In general, FFT module is designed for specific standard such as Ultra-Wide Band (UWB) system which needs high throughput FFT module and Very High Data Rate DSL (VDSL) system which demands long length FFT computation. Therefore, it is difficult to design a FFT module which is suitable for any specification.

ASIC approaches have been used to achieve the high performance demands which software or general purpose DSP implementations fail to deliver but custom hardware are often less cost-effective and flexible than general processors. Hence, it adopts ASIC and processor characteristic to set up FFT module in this thesis. ASIC plays an accelerated role in the proposed system and it executes partial FFT algorithm. Processor can flexibly execute the remaining FFT computation and it only takes processor performance into consideration. Therefore, the proposed system can meet different communication systems by reconfiguring computation of processor.

First, we need to analyze computational complexity of FFT algorithm. Because FFT algorithm is composed of addition and multiplication operations, it calculates computational loading between processor and ASIC based on Million Operations Per Second (MOPS) in this thesis. Therefore, in this thesis, the given processor can be calculated that it needs to spare how much computational power for the proposed FFT module in various standards. After that, the branch FFT of ASIC can be designed as an accelerator to complete FFT algorithm.

1.2 Organization of This Thesis

In this thesis, the proposed FFT system can process IEEE 802.11n/16e standards and we not only propose optimized time schedule, but also provide users ASIC and processor computational allocation analysis which are shown in the following chapters. The summary of each chapter can be listed as below：

Chapter 2 Backgrounds

It introduces MIMO OFDM system standards and FFT algorithm. Different radix FFT architectures are compared and variable length FFT architectures are described.

Chapter 3 ASIC and Processor Co-Design Analysis

First, we calculate FFT computational complexity which can analyze processor performance. After that, the proposed time schedule and architectures are applied in the SISO/MIMO systems independently. Finally, we can analyze relationship between ASIC and processor.

Chapter 4 Implementation of the ASIC Architecture

It compares various pipeline-based FFT architectures and then introduces detailed sub-module architectures. Then, it calculates error noise.

Chapter 5 System Verification and Simulation Results

It shows verification of branch FFT which covers cell-based flow and FPGA verification. After that, we model behavior of processor to verify variable-length FFT computation and time schedule in the SISO/MIMO systems.

Chapter 6 Conclusion and Future Work

Chapter 2 Backgrounds

2.1 WLAN MIMO-OFDM System

Orthogonal Frequency Division Multiplexing (OFDM) is widely applied in high-speed Wireless Local Area Network (WLAN) such as IEEE 802.11a/g/n, Hiperlan/2 and Wireless Personal Area Network (WPAN) such as Ultra-Wide Band (UWB) system. OFDM is a special case of multicarrier transmission, where a single data stream is transmitted over a number of lower rate subcarriers. OFDM can be seen as either a modulation technique or a multiplexing technique. One of the main reasons to use OFDM is to increase the robustness against frequency selective fading or narrowband interference. To eliminate the banks of subcarriers oscillators and coherent demodulators required by frequency division multiplex, Discrete Fourier Transform (DFT) processor is essential to be implemented.

Multiple-Input Multiple-Output (MIMO) system was instituted by Marconi in 1908. Channel fading can be suppressed by multiple antennas in both transmitter and receiver, the so-called MIMO system, has received significant attention in recent years owing to their potential to increase system capacity.

The High Throughput Task Group which establishes IEEE 802.11n standard is going to draw up the next-generation WLAN proposal based on the 802.11a/g which is the current OFDM-based WLAN standards [1]. The IEEE 802.11n standard based on the MIMO OFDM system provides very high data throughput rate from the original data rate 54 Mb/s to the data rate in excess of 600 Mb/s because the technique of the MIMO can increase the data rate by extending an OFDM-based system. A block diagram of the 2x2 transceiver of IEEE 802.11n is shown in Fig. 2.1 and Fig. 2.2. Depending on the desired data rate, the modulation scheme can be Binary Phase Shift Keying (BPSK), Quaternary Phase Shift Keying (QPSK), or Quadrature Amplitude Modulation (QAM) with 1-6 bits. The encoding rates in this specification are 1/2, 2/3, 3/4, or 5/6. The number of spatial sequence is supported by 1, 2, 3, or 4. The guard interval period is 400 ns or 800 ns. The bandwidth of the transmitted signal is 20 or 40 MHz. The FFT (Fast Fourier Transform) size is 64 points or 128 points.

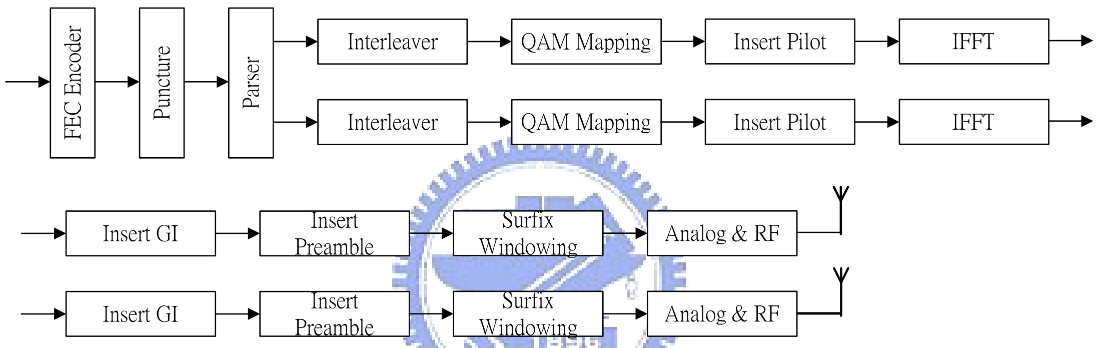


Fig. 2.1 Block diagram of IEEE 802.11n WLAN 2x2 transmitter system

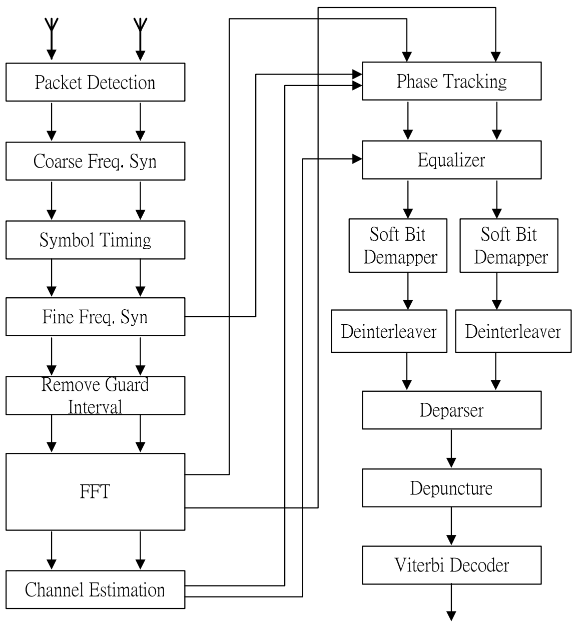


Fig. 2.2 Block diagram of IEEE 802.11n WLAN 2x2 receiver system

However, the IEEE 802.11n standard also increases the computational and hardware complexity greatly which is compared with the current WLAN standards. The FFT/IFFT processor is one of the highest computational complexity modules in the physical layer of the IEEE 802.11n standard, as shown in Table 2.1 [29]. Multiple FFT modules are added to deal with multiple data sequences in the MIMO OFDM system and therefore it causes a large increase in the hardware complexity and power consumption.

Table 2.1 The comparison of the hardware complexity of the receiver

	Multiplier	Adder	Register	Gate Count (K)
Packet Detection	4	4	50	50
AGC	1	1	1	30
Frequency Offset	4	18	96	80
Frame Detection	8	8	8	50
FFT	1	12	68	160
Channel Estimation	0	0	128	60

2.2 Flexible FFT Processor

OFDM techniques play an important role in modern wireless and wireline communication systems. The FFT processor is one of the highest computational complexity modules and FFT sizes, sampling rates are different in various standards which are shown in Table 2.2. It is desired to design a FFT processor which adapts to various FFT sizes in different communication standards. This paper proposes an FFT processor which can deal with the variable length FFT for different standards.

Table 2.2 FFT sizes and sampling rates in various communication systems

Communication System	FFT Size (Sampling Rate)
802.11a	64 (20MHz)
802.11n	64 (20MHz)、128 (40MHz)
802.16e	2048 (20MHz)、1024 (10MHz)、512 (5MHz)、128 (1.25MHz)
DAB	2048、1024、512、256 (2MHz)
DVB-T	8192、2048 (8MHz)
DVB-H	4096 (8MHz)
ADSL	512 (2.2MHz)
VDSL	8192 (34.5MHz)、4096 (17.3MHz)、2048 (8.6MHz)、1024 (4.3MHz)、512 (2.2MHz)
UWB	128 (528MHz)

2.3 Discrete Fourier Transform

The N-point Discrete Fourier Transform (DFT) X(k) of a complex data sequence x(n) is defined as

$$X(k) = \sum_{n=0}^{N-1} x(n) W_N^{nk}, \quad k \in \{0, 1, \ldots, N-1\} \tag{1.1}$$

where the twiddle factor is

$$W_N^{nk} = e^{-j(\frac{2\pi nk}{N})} \tag{1.2}$$

If the periodic and symmetric properties of the twiddle factor W_N^{nk} are exploited, the computation of $X(k)$ will be more efficient.

$$\text{Symmetric：} W_N^{m+N/2} = -W_N^m$$

$$\text{Periodic：} W_N^{m+N} = W_N^m$$

According to equation (1.1), the computational complexity is O(N^2) through directly performing the required computation. It needs N^2 complex multiplication and N(N-1) complex addition. If using the FFT algorithm, the computational complexity can be reduced to O($Nlog_rN$), where r means the radix-r FFT. The radix-r FFT can be derived from DFT by decomposing the N-point DFT into a set of recursively related r-point transform. There are two types of FFT algorithm are Decimation in Time (DIT) and Decimation in Frequency (DIF). The computational complexity is the same. The DIT algorithm decomposes $x(n)$ into radix-r module sequence

$$\begin{aligned}
X(k) &= \sum_{n=0}^{N-1} x(n) W_N^{nk} \\
&= \sum_{n\,:\,even} x(n) W_N^{kn} + \sum_{n\,:\,odd} x(n) W_N^{kn} \\
&= \sum_{r=0}^{N/2-1} x(2r) W_N^{2rk} + \sum_{r=0}^{N/2-1} x(2r+1) W_N^{(2r+1)k} \\
&= \sum_{r=0}^{N/2-1} x(2r) W_{N/2}^{rk} + W_N^k \sum_{r=0}^{N/2-1} x(2r+1) W_{N/2}^{rk}
\end{aligned}$$

and the DIF algorithm decomposes $X(k)$ in the same way [2].

$$X(k)=\sum_{n=0}^{N-1}x(n)W_N^{nk},\quad k\in\{0,1,\ldots,N-1\}$$

$$\begin{aligned}
\textit{even term}:X(2r) &= \sum_{n=0}^{N-1}x(n)W_N^{2nr},\quad r\in\{0,1,\ldots,N-1\}\\
&= \sum_{n=0}^{N/2-1}x(n)W_N^{2nr}+\sum_{n=N/2}^{N-1}x(n)W_N^{2nr}\\
&= \sum_{n=0}^{N/2-1}x(n)W_N^{2nr}+\sum_{n=0}^{N/2-1}x(n+\frac{N}{2})W_N^{2(n+\frac{N}{2})r}\\
&= \sum_{n=0}^{N/2-1}\left\{x(n)+x(n+\frac{N}{2})\right\}W_{N/2}^{nr}
\end{aligned}$$

$$\begin{aligned}
\textit{odd term}:X(2r+1) &= \sum_{n=0}^{N/2-1}x(n)W_N^{n(2r+1)}+\sum_{n=N/2}^{N-1}x(n)W_N^{n(2r+1)}\\
&= \sum_{n=0}^{N/2-1}\left\{x(n)-x(n+\frac{N}{2})\right\}W_N^{n(2r+1)}\\
&= \sum_{n=0}^{N/2-1}\left\{x(n)-x(n+\frac{N}{2})\right\}W_{N/2}^{nr}W_N^{n}
\end{aligned}$$

2.4 Complexity Comparison

From Table 2.3 [3] and Table 2.4 [4], the multiplications and additions of radix-8 have the lowest complexity compared with radix-2 and radix-4. The constant multiplication can be implemented by shifters and adders whose hardware is smaller than real multiplication. Table 2.5 [5] is the complexity equation of multiplications and additions. The radix-8 type-1 algorithm is the original radix-8 FFT algorithm. In radix-8 type-2 algorithm, we replaced multiplication of W_8^1 into p addition.

According to the hardware area and power consumption of complex number multipliers, we only focus on the number of real number multiplications. In Fig. 2.3, radix-8 type-2 has the lowest computational complexity, so we choose radix-8 type-2 as the building block to implement FFT algorithm.

Table 2.3 Multiplication comparison [3]

N-point	Radix-2	Radix-4	Radix-8	
Multiplier	Multiplier	Multiplier	Multiplier	Constant Multiplier
8	2	3	0	2
16	10	8	6	4
32	34	31	20	8
64	98	76	48	32
128	258	215	152	64
256	642	492	376	128
512	1538	1239	824	384
1024	3586	2732	2104	768
2048	8194	6487	4792	1536
4096	18434	13996	10168	4096
8192	40962	32087	23992	8192

Table 2.4 Multiplications and additions comparison [4]

	Real Multiplications			Real Additions		
N-point	Radix-2	Radix-4	Radix-8	Radix-2	Radix-4	Radix-8
16	24	20		152	148	
32	88			408		
64	264	208	204	1032	976	972
128	720			2054		
256	1800	1392		5896	5488	
512	4360		3204	13566		12420
1024	10248	7856		30728	28336	

Table 2.5　Equation of multiplications and additions comparison [5]

Algorithm	Real Multiplication	Real Addition
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 Type-1	$\frac{25N}{24}(\log_2 N - 3) + 4$	$\frac{73N}{24}\log_2 N - \frac{25}{8}N + 4$
Radix-8 Type-2	$\frac{21N}{24}\log_2 N - \frac{25}{8}N + 4$	$\frac{8p + 73N}{24}\log_2 N - \frac{25}{8}N + 4$

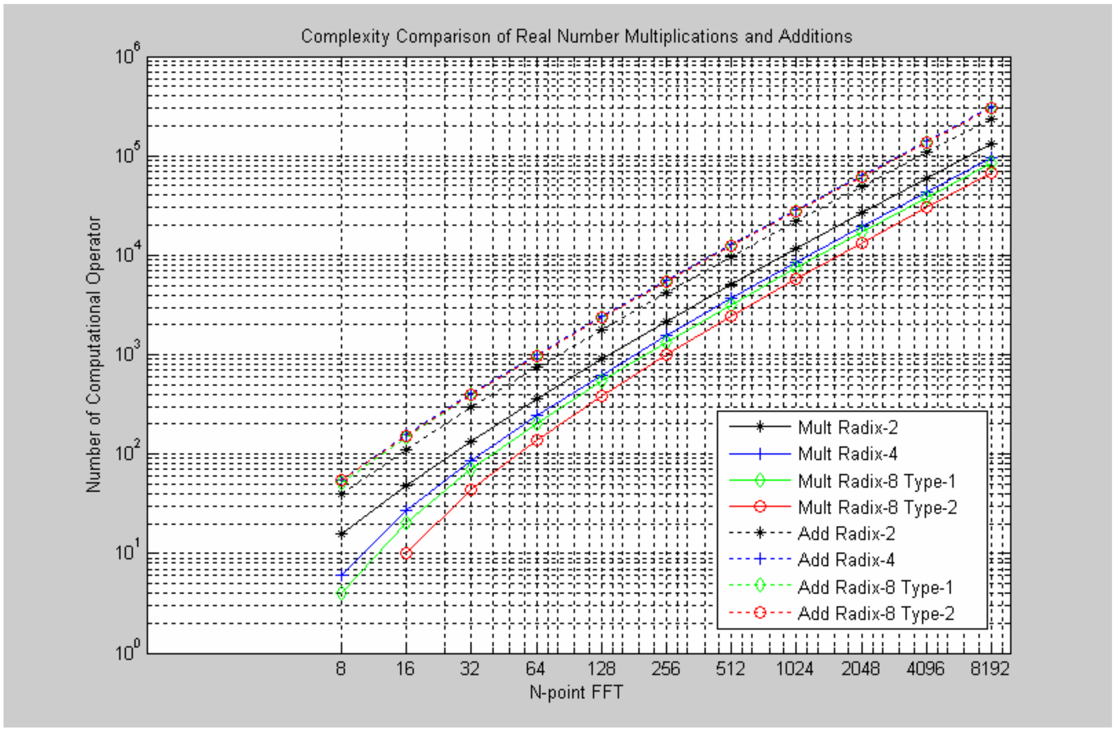


Fig. 2.3　Complexity comparison of Table 2.5

Higher radix algorithm can reduce number multiplications, but it will let controller and butterfly unit more difficult. Based on this consideration, radix-8 is the highest radix algorithm mainly.

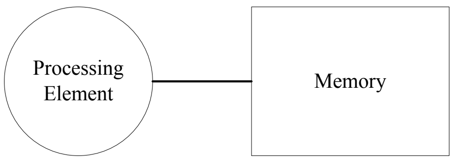
2.5 Variable Length of FFT Architectures

FFT algorithm decomposes the fundamental calculation of the DFT of a sequence of length N into continuously smaller subsequences (branch FFT). In section 2.3, the FFT algorithm is applied not only in DSP, image processing and digital data transmission systems, but also in biomedical electronic engineering and home networking. To meet various standards, designer must implement FFT module with variable length. In general, we can classify two main architectures of variable length FFT as following subsections.

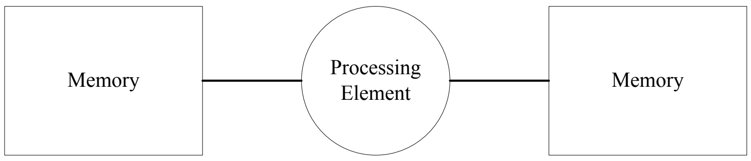
2.5.1 Memory Based Architectures

The processing element in Fig. 2.4 and Fig. 2.5 performs butterfly operation. Fig. 2.4(a) shows the single-memory architecture. It has one processing element and one memory element. Butterfly outputs are stored in the same memory location used by butterfly inputs [6]. Fig. 2.4(b) shows the dual-memory architecture. One memory is used to store butterfly inputs and the other is used for butterfly outputs. These two architectures require small areas. However, they have low throughput and require high clock frequency.

Each stage requires reading and writing to N data words, and memory access is considered to be one of the bottlenecks under the recursive architecture. Therefore, variable length FFT researches proposed different memory address controller [7][8][9][10][11].



(a) Single-memory architecture



(b) Dual-memory architecture

Fig. 2.4 Memory based architecture block diagram

2.5.2 Pipeline Architectures

For high throughput applications, pipeline architecture has been developed in some literatures. Pipeline architecture is characterized by non-stopping processing on a clock frequency of the input data sampling. However, the computational resource costs are increased because of the requirements of $\log_r N$ branch FFT and $\log_r N+1$ buffer memory, as shown in Fig. 2.5. Based on pipeline architecture, variable length FFT literatures can cascade branch FFT to achieve different sizes computation [12]. Therefore, in the design of FFT processors for different systems, we should not only enhance the speed by introducing more parallelism and pipelines, but also reduce the hardware resource consumption as possible as we can.

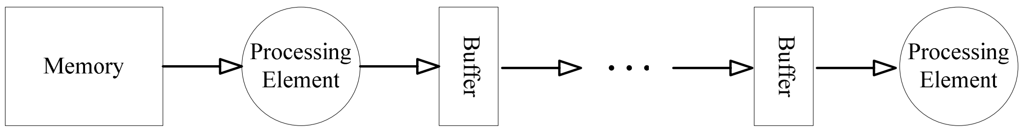


Fig. 2.5 Pipeline based of block diagram

Chapter 3 ASIC and Processor Co-Design Analysis

3.1 Introduction

In recent years, the number and variety of products that include some form of digital signal processing (DSP) has grown dramatically. They are often more cost-effective (and less risky) than custom hardware, particularly for low-volume applications, where the development cost of custom ICs may be prohibitive.

In the MIMO OFDM system, multiple antennas need multiple FFT/IFFT modules in transmitter and receiver, as shown in Fig. 2.1 and Fig. 2.2. Therefore, it causes a large increase in the hardware complexity and power consumption. Besides, based on various standards, designers need to re-design different sizes and throughput of FFT modules shown in Table 2.2. Because processor application is popular in recent years, we make use of processor advantages to propose a method that processor and ASIC co-design which can enhance flexibility and utilize time schedule efficiently to reduce ASIC cost. We can provide designers two deciding messages. How much processor performance is required in different environments? How many branch FFT need to be implemented by hardware in various processors?

3.2 FFT Computational Complexity Analysis

In equations (3.1) and (3.2) are composed of two (N/2)-point DFTs. It is well known that one can combine these two equations as one basic butterfly (BF) module as shown in Fig. 3.1, where x(n) and x(n+N/2) are the input data.

$$X(k)=\sum_{n=0}^{N-1} x(n)W_N^{nk}, \quad k\in\{0,1,\ldots,N-1\}$$

$$\textit{even term}:X(2r) = \sum_{n=0}^{N/2-1}\left\{x(n)+x(n+\frac{N}{2})\right\}W_{N/2}^{nr} \qquad (3.1)$$

$$\textit{odd term}:X(2r+1)=\sum_{n=0}^{N/2-1}\left\{x(n)-x(n+\frac{N}{2})\right\}W_{N/2}^{nr}W_N^{n} \qquad (3.2)$$

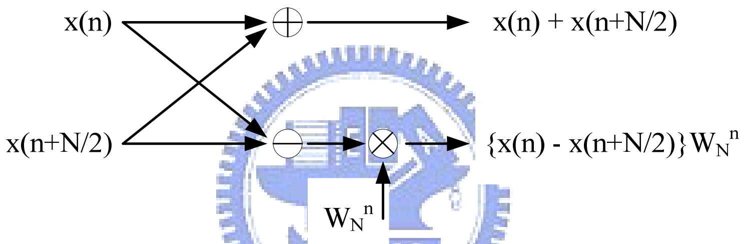


Fig. 3.1 The butterfly signal flow graph of radix-2 DIF FFT

By recursive decompositions, we can further partition small DFTs into even smaller DFTs, and so on. For example, a 16-point radix-2 DIF FFT, in signal flow graph, is shown in Fig. 3.2.

Long-length FFT can be decomposed into several branch FFT by different radix algorithm. In section 2.4 simulation results, radix-8 FFT reduces the complexity more than other radix algorithms. But FFT length is restricted to power of eight. In any event, FFT architecture is composed of many butterfly units, and additions and multiplications form butterfly units. Thus, we can analyze FFT computational complexity by calculating number of additions and multiplications.

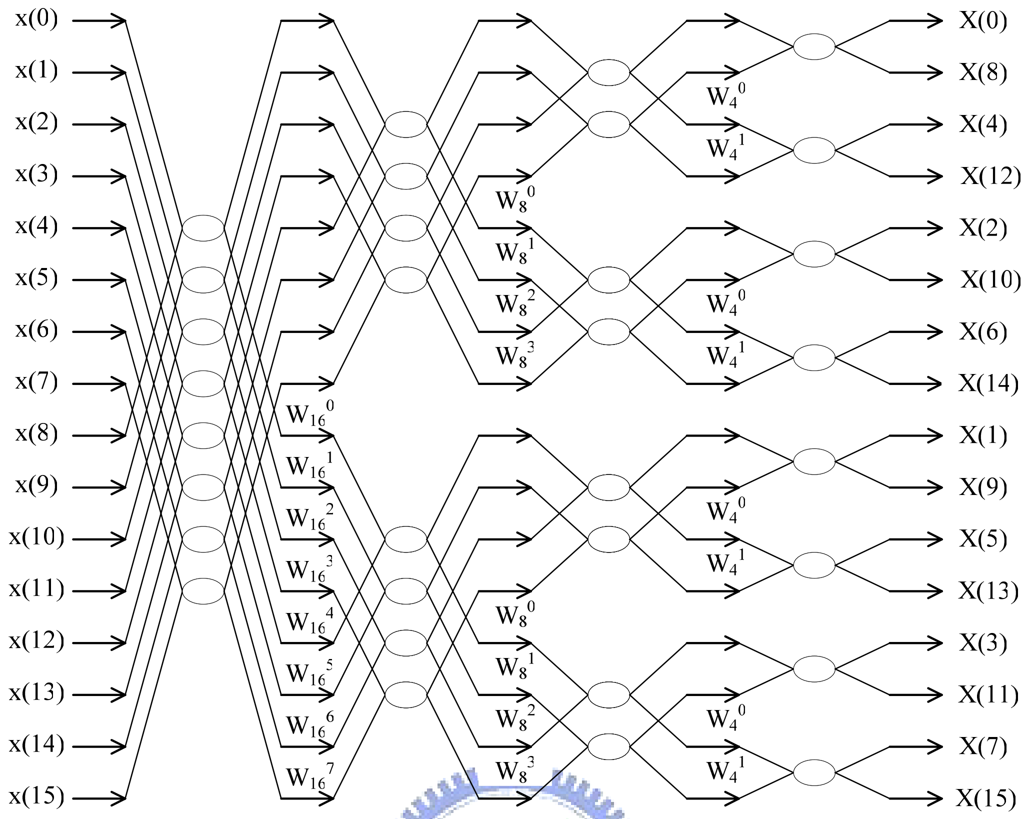


Fig. 3.2 Radix-2 DIF FFT signal flow graph of a 16-point FFT

Complex addition can be decomposed two real additions, and complex multiplication can be decomposed two real additions and four real multiplications as shown equation (3.3).

$$
\begin{aligned}
&(\mathrm{Re} + \mathrm{Im}\, j) \times (\cos(\frac{2\pi nk}{N}) + \sin(\frac{2\pi nk}{N})\, j) \\
&= (\mathrm{Re} \times \cos(\frac{2\pi nk}{N}) - \mathrm{Im} \times \sin(\frac{2\pi nk}{N})) + j(\mathrm{Re} \times \sin(\frac{2\pi nk}{N}) + \mathrm{Im} \times \cos(\frac{2\pi nk}{N})) \quad (3.3)
\end{aligned}
$$

Therefore, we try to evaluate computational complexity of different length FFT algorithm which is a little difference to section 2.4. Because we calculate any computation in terms of processor operations, it doesn’t include any hardware reduced computation, just like W_8^1 can be implemented by shifters and adders. In this thesis, we take IEEE 802.11n/16e standards into consideration as shown in Table 2.2. FFT length covers from 64-points to 2048-points. We regard real addition or real multiplication as an operation in the analysis. In IEEE 802.11n/16e standards, 128-point/2048-point is the critical case separately, because of long-length FFT increase operations dramatically and symbol durations are the same shown in Table 3.1. Therefore, we analyze these two cases and assume partial branch FFT which is implemented by hardware and then processor computational loading can be shown in Table 3.2 and Table 3.3.

Table 3.1 Comparison operations of FFT size in IEEE 802.11n/16e standards

802.11n FFT Size (Sampling Rate)	Operations = Real additions + Real multiplications
128 (40 MHz)	3142
64 (20 MHz)	1254
802.16e FFT Size (Sampling Rate)	Operations = Real additions + Real multiplications
2048 (20 MHz)	83462
1024 (10 MHz)	38150
512 (5 MHz)	16518
128 (1.25 MHz)	3142

$$\text{Processor operations} = \underbrace{2\times N\times S}_{\substack{\textit{addition}\\ \text{operations}}} + \underbrace{(\frac{N}{R_1}-1)\times(R_1-1)\times(2+4)}_{\text{group1 multipliers}} + \underbrace{R_1(\frac{N}{R_1R_2}-1)\times(R_2-1)\times(2+4)}_{\text{group2 multipliers}} + \underbrace{R_1R_2(\frac{N}{R_1R_2R_3}-1)\times(R_3-1)\times(2+4)}_{\text{group3 multipliers}} + \ldots + \frac{N}{8}\times K\times 2\times(2+4) \quad (3.4)$$

$$\text{where}\begin{cases}\text{N : FFT size , S : Number of remaining stages , K : Number of radix - 8 groups}\\ \text{R}_\text{n}\text{: radix - R}_\text{n}\text{ , n} \in \{1,2,3...\}\end{cases}$$

In equation (3.4), processor computational operations can be divided into three parts. Addition and multiplication operations, besides constant multiplication of radix-8 must be taken into account. This analysis can be applied to others FFT sizes.

Table 3.2 Comparison of different length ASIC operations of a 128-point FFT

ASIC length of 128-point FFT	Processor Operations = Real additions + Real multiplications
128	0
64	2*128+63*(2+4) = 634
32	2*128*2+31*3*(2+4) = 1070
16	2*128*3+15*7*(2+4)+1*16*2*(2+4) = 1590
8	2*128*4+63*(2+4)+2*7*7*(2+4)+1*16*2*(2+4) = 2182
4	2*128*5+31*3*(2+4)+4*3*7*(2+4)+1*16*2*(2+4) = 2534
2	2*128*6+15*7*(2+4)+8*7*(2+4)+2*16*2*(2+4) = 2886
0	2*128*7+63*(2+4)+2*7*7*(2+4)+2*16*2*(2+4) = 3142

Table 3.3 Comparison of different length ASIC operations of a 2048-point FFT

ASIC length of 2048-point FFT	Processor Operations = Real additions + Real multiplications
2048	0
1024	2*2048+1023*(2+4) = 10234
512	2*2048*2+511*3*(2+4) = 17390
256	2*2048*3+255*7*(2+4)+256*2*(2+4) = 26070
128	2*2048*4+1023*(2+4)+2*127*7*(2+4)+256*2*(2+4) = 36262
64	74246-2*2048*6-32*7*7*(2+4)+1*256*2*(2+4) = 43334

32	2*2048*6+255*7*(2+4)+8*31*7*(2+4)+2*256*2*(2+4) = 51846
16	74246-2*2048*4+32*15*3*(2+4)-32*7*7*(2+4)+1*256*2*(2+4) = 60166
8	74246-2*2048*3+2*256*2*(2+4) = 68102
4	2*2048*9+255*7*(2+4)+8*31*7*(2+4)+8*8*3*7*(2+4)+ 3*256*2*(2+4) = 75270
2	74246-2*2048+256*1*3*(2+4)+2*256*2*(2+4) = 80902
0	2*2048*2+2*511*3+4*511*3+4*(10882+3332)+3*256*2*(2+4) = 83462

3.3 ASIC and Processor Timing Schedule

In this thesis, because we need to consider multiple antennas in our system, it let time schedules needed to be deliberated independently. It tries to lower the length of branch FFT and enhances processor and ASIC utilization.

3.3.1 SISO System Timing Schedule

We propose two schedules in SISO system. In schedule Ⅰ, input sequences are written into memory first which can receive continuous data and reorder data sequences. Afterward it processes data sequences sequentially within one symbol duration, and therefore processor and ASIC utilization are not 100% as shown in Fig. 3.3. Besides, processor has little time to operate because of ASIC also occupies partial symbol duration. Therefore, processor needs better operation performance.

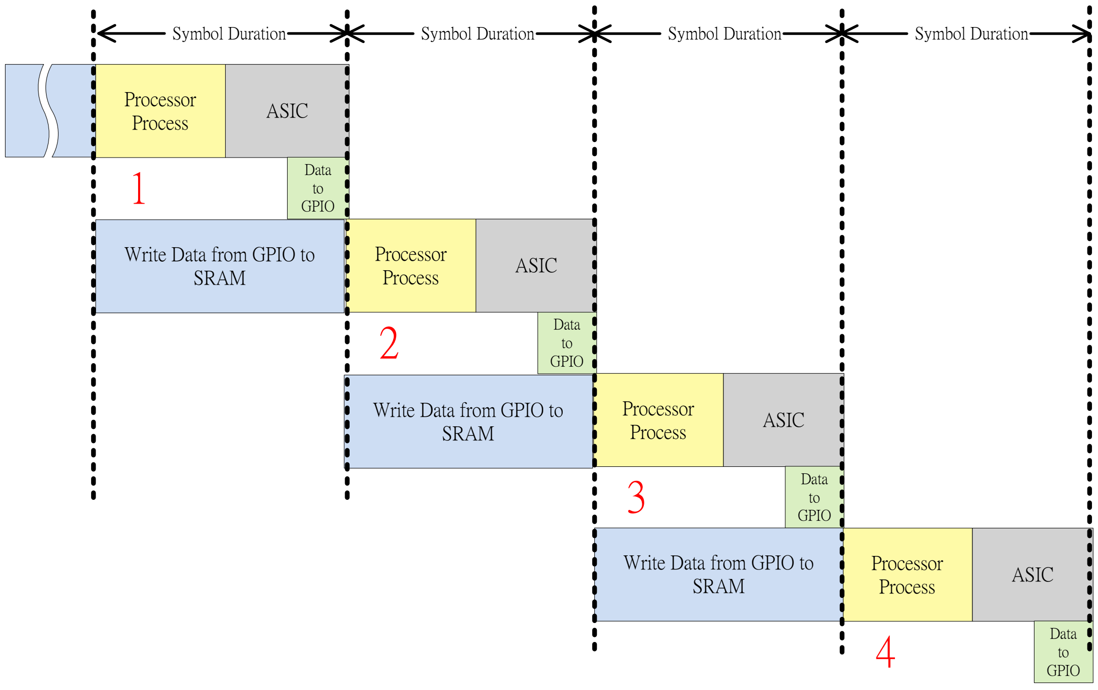


Fig. 3.3 Time schedule Ⅰ of the SISO system

In Fig. 3.4, it shows system block diagram based on time schedule Ⅰ. The wrapper module is used to communicate On-Chip Peripheral Bus (OPB) handshaking signals [13]. ASIC is responsible for branch FFT algorithm. Register module is stored control signals which govern entire data flow.

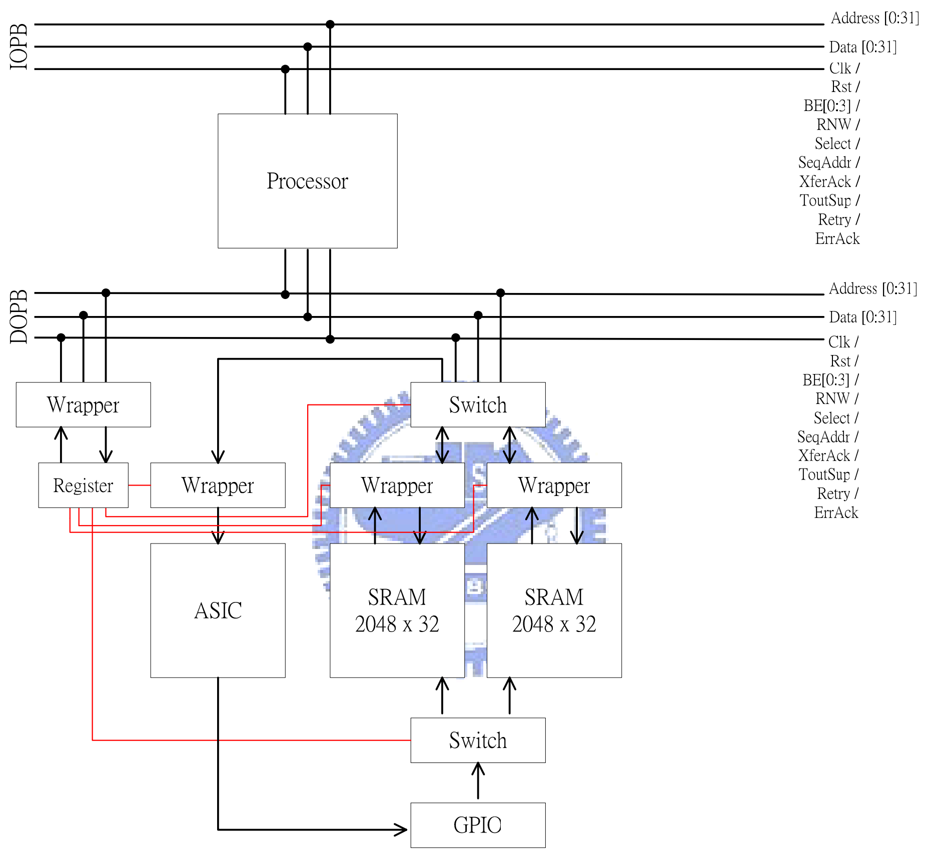

IOPB
Address [0:31]
Data [0:31]
Clk / Rst / BE[0:3] / RNW / Select / SeqAddr / XferAck / ToutSup / Retry / ErrAck
Processor
DOPB
Address [0:31]
Data [0:31]
Clk / Rst / BE[0:3] / RNW / Select / SeqAddr / XferAck / ToutSup / Retry / ErrAck
Wrapper
Switch
Register
Wrapper
Wrapper
Wrapper
ASIC
SRAM 2048 x 32
SRAM 2048 x 32
Switch
GPIO

Fig. 3.4 SISO system block diagram of the time schedule Ⅰ

In schedule Ⅱ, it makes efforts to raise processor and ASIC utilization as shown in Fig. 3.5. It not only decreases processor operations per second, but also lowers power consumption because of decreasing clock frequency. Additional buffer is used to increase processor and ASIC processing time up to one symbol duration, but it causes more hardware cost shown in Fig. 3.6.

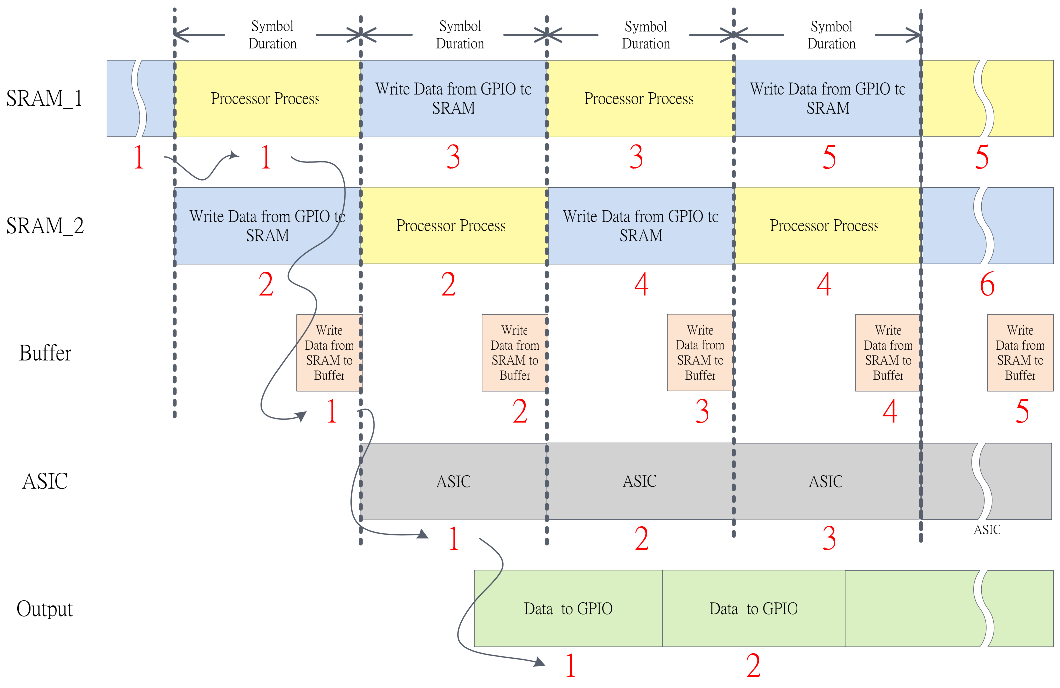


Fig. 3.5 Time schedule II of the SISO system

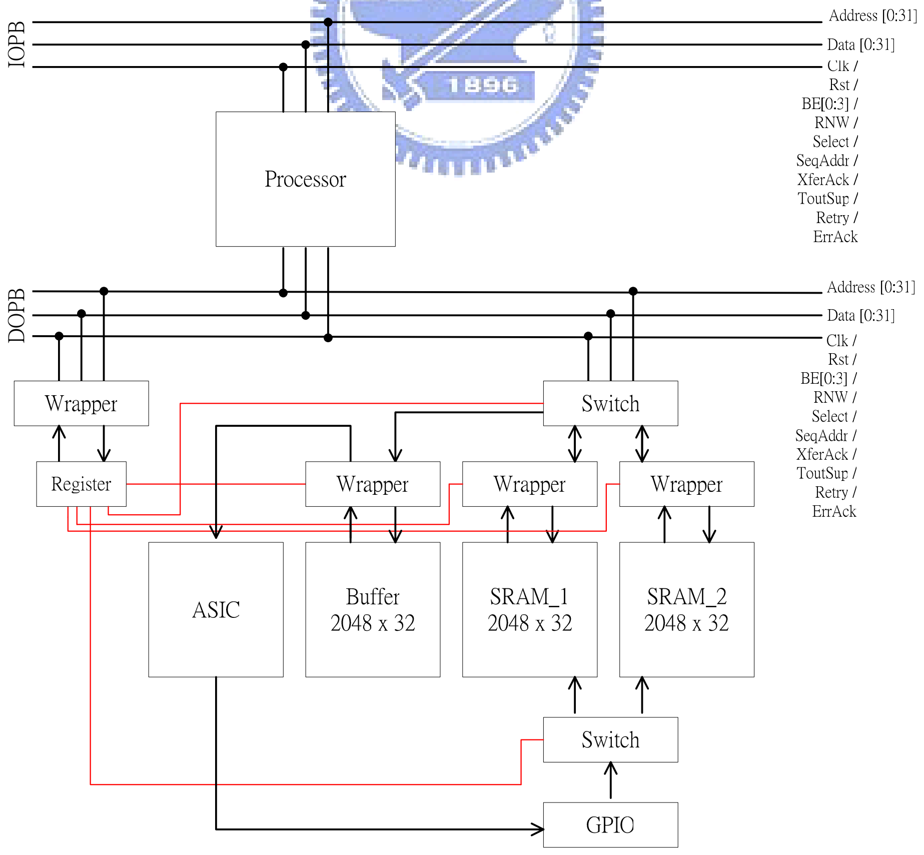


Fig. 3.6 SISO system block diagram of the time schedule II

3.3.2 MIMO System Timing Schedule

In general, channel fading can be suppressed by multiple antennas in both transmitter and receiver in the MIMO system, but it also increases hardware area dramatically. Therefore, time schedule in the MIMO system, it tries to minimize hardware area and enhances processor and ASIC utilization simultaneously. We find that time schedule Ⅰ in the SISO system which have many bubbles can be utilized to process others computation. Based on this concept, we propose a suitable method for the MIMO system which can eliminate bubbles by processing sequences of another antenna which exchange processor and ASIC processing order as shown in Fig. 3.7

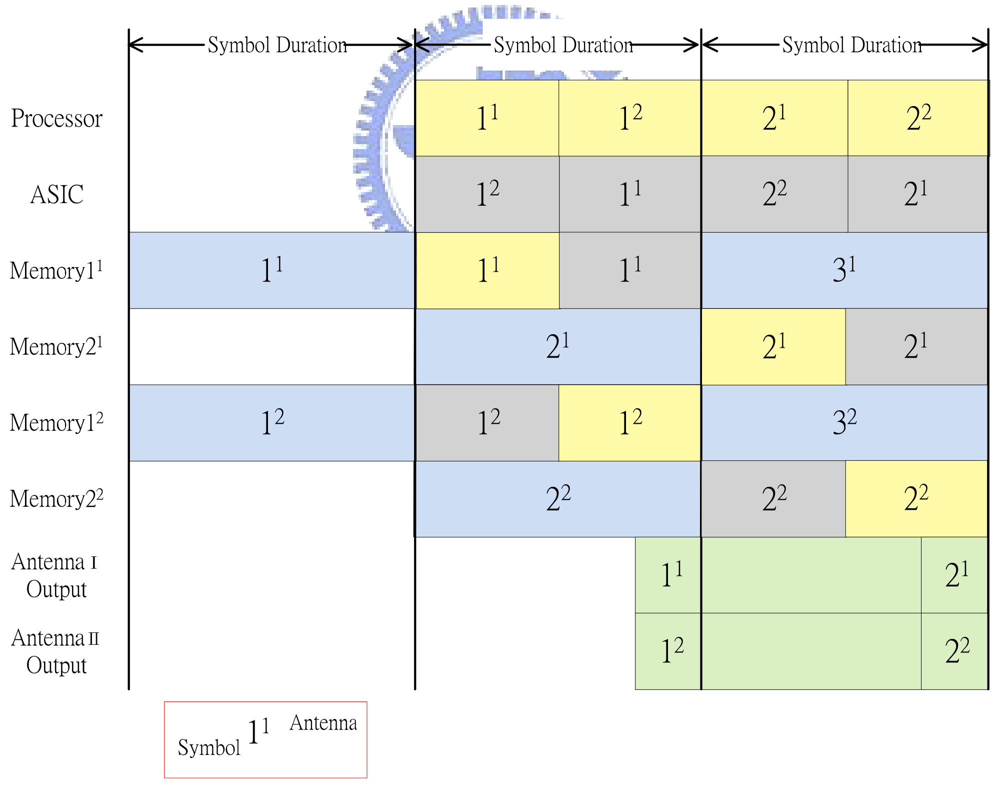


Fig. 3.7 Time schedule of the MIMO system

ASIC and processor compute sequences of different antennas by turns within half symbol duration. Therefore, comparison with schedule Ⅱ in the SISO system, processor needs two times operation performance per second in the MIMO system. It can process sequences of two antennas simultaneously, and doesn't need additional hardware of branch FFT shown in Fig. 3.8.

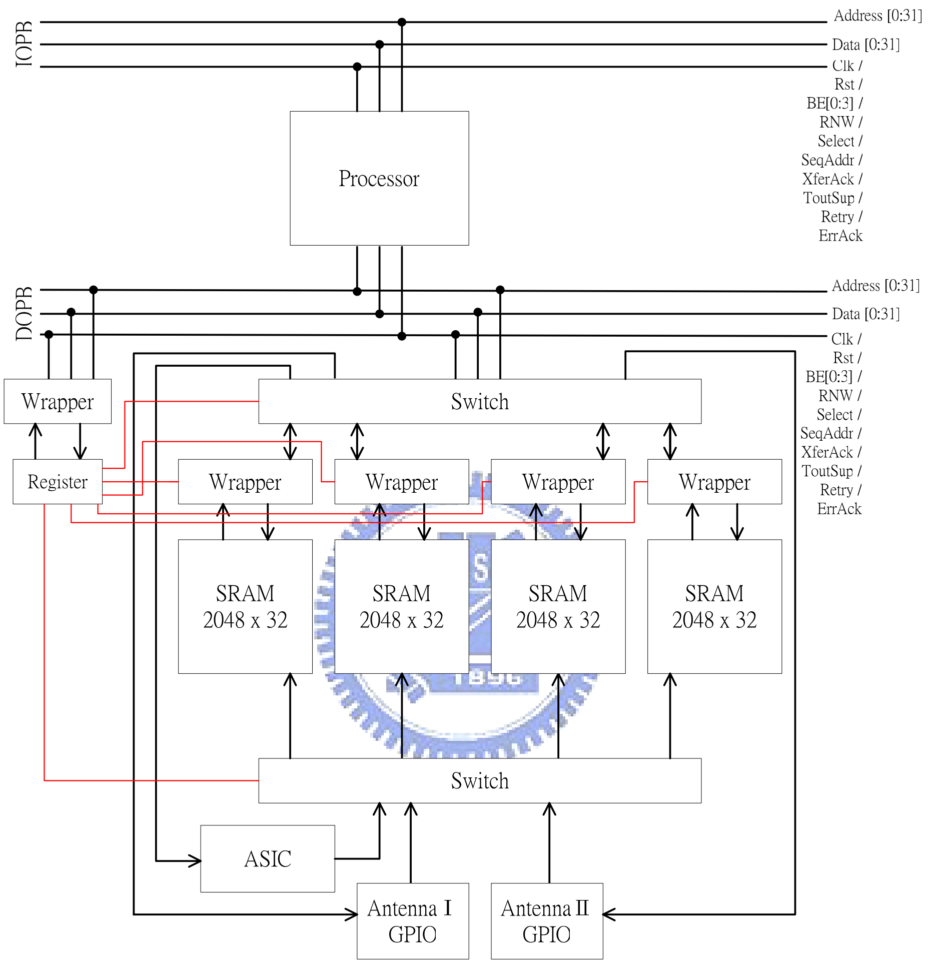


Fig. 3.8 MIMO system block diagram

3.4 ASIC and Processor Performance Simulation

Since SISO/MIMO schedules are proposed, an evaluation model is developed to verify specification requirements. Base on IEEE 802.11n/16e standards, we can introduce symbol period to calculate processor performance when different length of branch FFT is implemented by hardware. ASIC plays an accelerative role in the system. Increasing the length branch FFT of ASIC can release burden of processor, and vice versa. Because 128-point/2048-point is the critical case in IEEE 802.11n/16e separately, we just only calculate these two cases.

Schedule Ⅰ in the SISO system, ASIC occupies partial symbol duration as shown in Fig. 3.3. Therefore, we need to calculate ASIC latency cycles approximately shown in Table 3.4 [14] and assume clock frequency is 100 MHz for simulation, as shown in Fig. 3.9. When length branch FFT of ASIC is too short, it cannot gain any benefits to the processor. The length branch FFT of ASIC affects processor operations directly. More length branch FFT is implemented by hardware will lower processor computational loading, but it increases area cost.

Table 3.4 Approximately calculation of latency cycles

FFT Length	Latency	FFT Length	Latency
0	0	64	103
2	2	128	208
4	4	256	336
8	8	512	592
16	26	1024	1616
32	44	2048	2640

In Fig. 3.9, we can analyze relationship between processor operations and branch FFT of ASIC. MOPS imply that processor computational loading divided by redundant symbol duration. When our system processes FFT algorithm only by processor, it shows that IEEE 802.11n needs more processor operations per second. Therefore, we can calculate performance of processor probably by MOPS. For instance, if processor can provide 700 MOPS for our system, we need to implement the 16-point branch FFT of ASIC in order to match IEEE 802.11n/16e specifications.

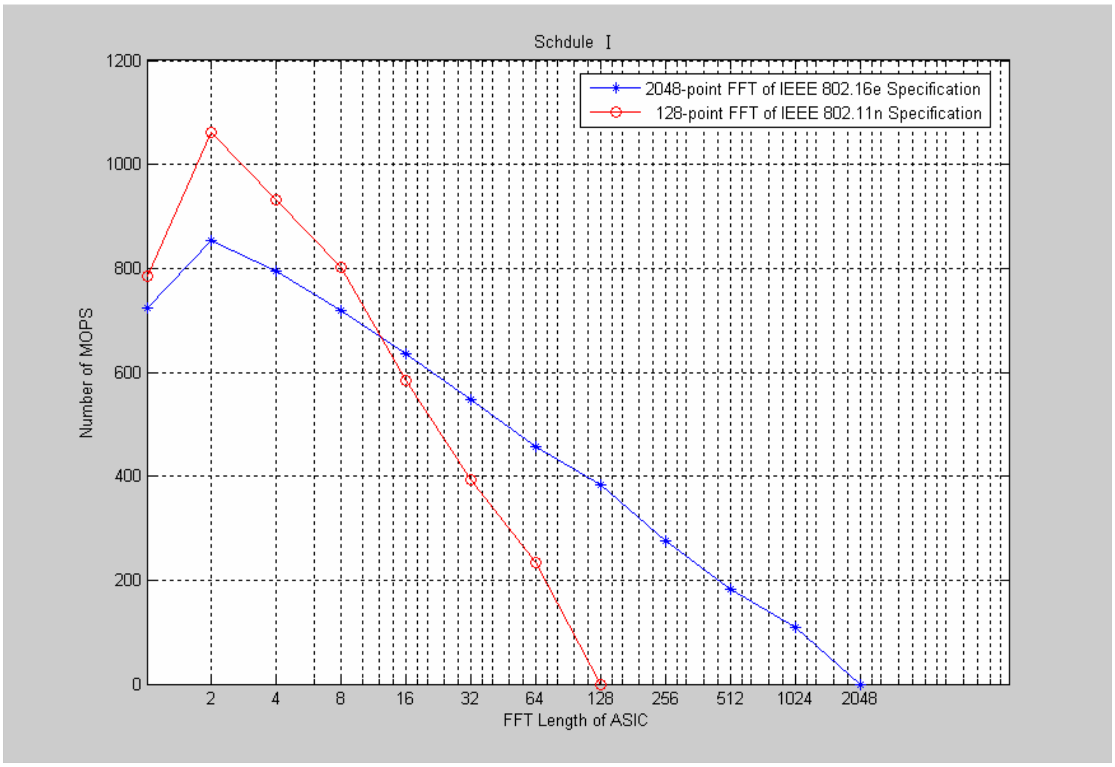


Fig. 3.9 Processor operations analysis of schedule Ⅰ in the SISO system

Schedule Ⅱ in the SISO system, processor has a symbol duration to process operations. Therefore, using the same processor as above, designer can implement the 4-point branch FFT of ASIC to reach specifications as shown in Fig. 3.10.

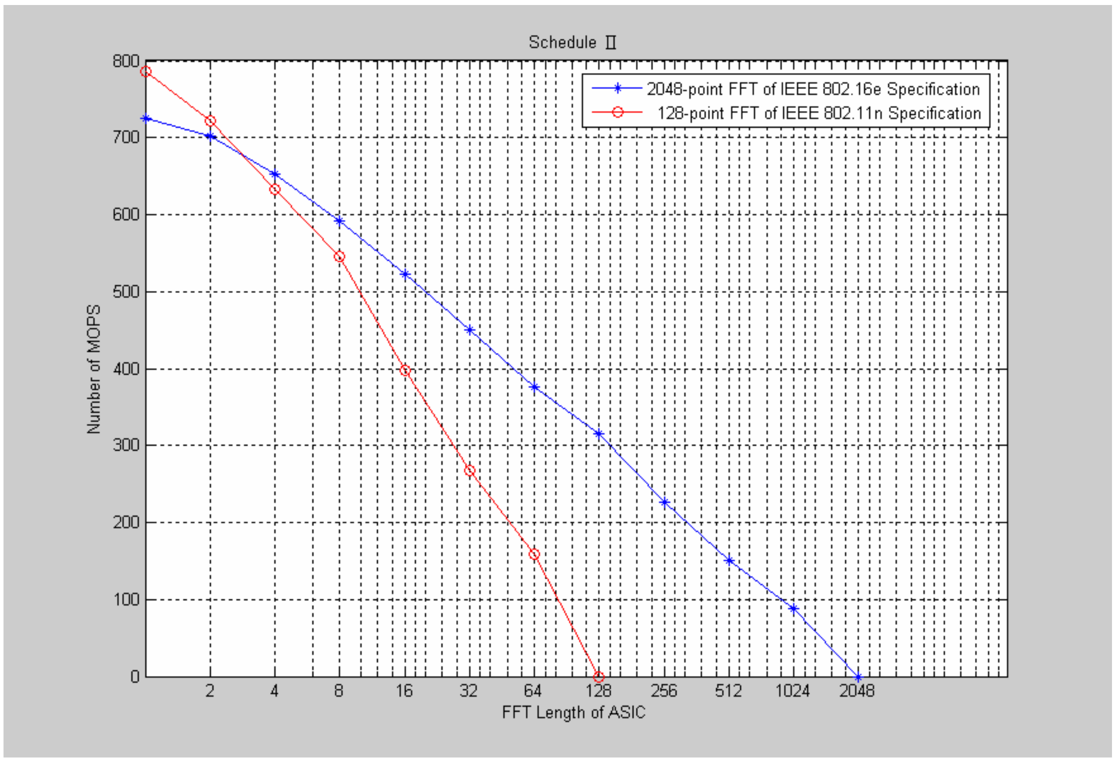


Fig. 3.10 Processor operations analysis of schedule Ⅱ in the SISO system

Therefore, in the MIMO system, processor and ASIC own half a symbol duration to complete operations. It can be expected that processor operations per second will double that of schedule Ⅱ in the SISO system. Using the same processor as above, system requires ASIC which is 128-point branch FFT as shown in Fig.3.11.

In this section, we introduce time schedule Ⅰ 、 Ⅱ in the SISO system, and time schedule in the MIMO system. Based on system block diagram, hardware cost in time schedule Ⅰ is less than time schedule Ⅱ, but utilization of time schedule Ⅰ is less than time schedule Ⅱ. In the MIMO system, bad utilization can be improved by changing ASIC and processor order. In the 2x2 MIMO systems, it only needs a processor and a branch FFT of ASIC. This time schedule not only lowers hardware cost, but also increases utilization.

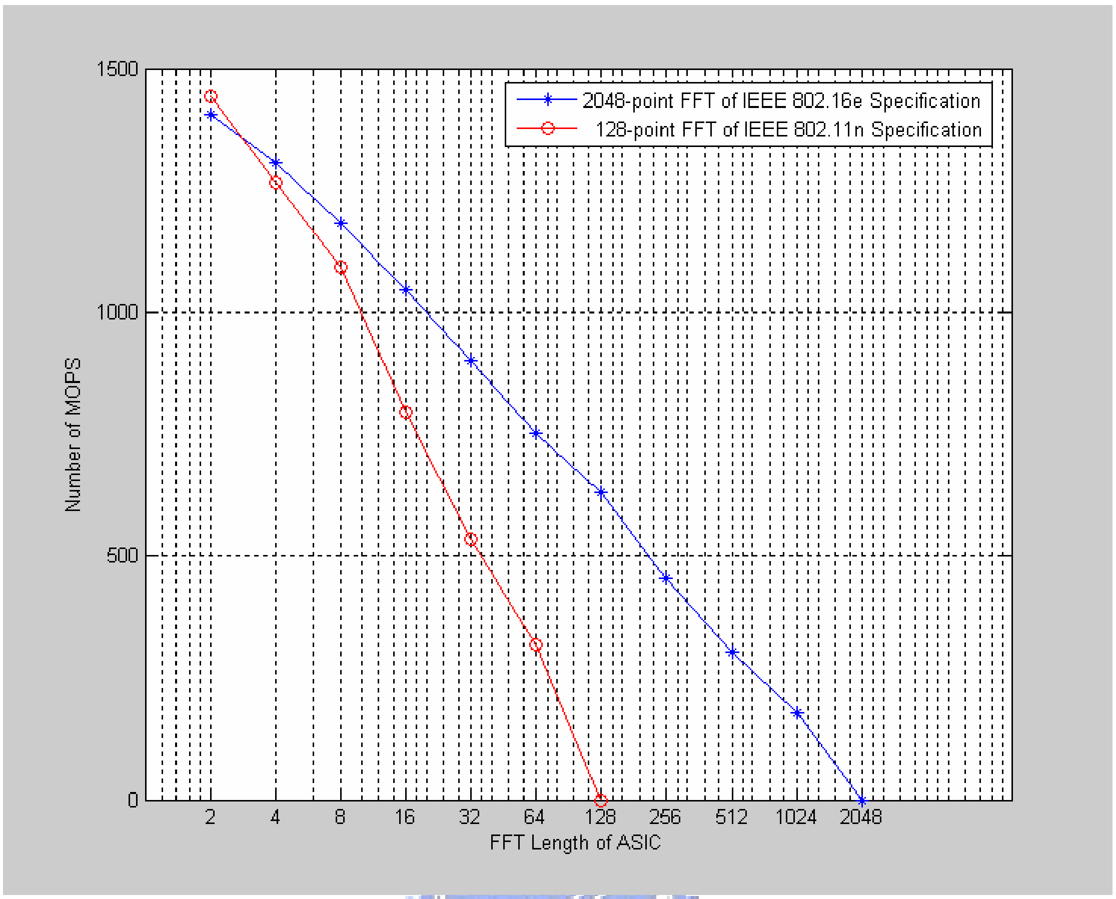


Fig. 3.11 Processor operations analysis in the MIMO system

In this article, we hope that proposed system can handle IEEE 802.11n/16e specifications flexibly. We assume processor can provide proposed FFT system with 800 MOPS. Therefore, 64-point branch FFT of ASIC is used to increase operating capacity as shown in Fig.3.11. In next chapter, we concentrate on how to implement the 64-point branch FFT of ASIC.

In Fig.3.12, it calculates computing loading ratio between processor and the 64-point branch FFT of ASIC. The FFT size includes IEEE 802.11n/16e standards. It shows that ASIC executes above half operations except 2048-point FFT algorithm. Therefore, we can regard the 64-point branch FFT of ASIC as an accelerator in the proposed FFT system.

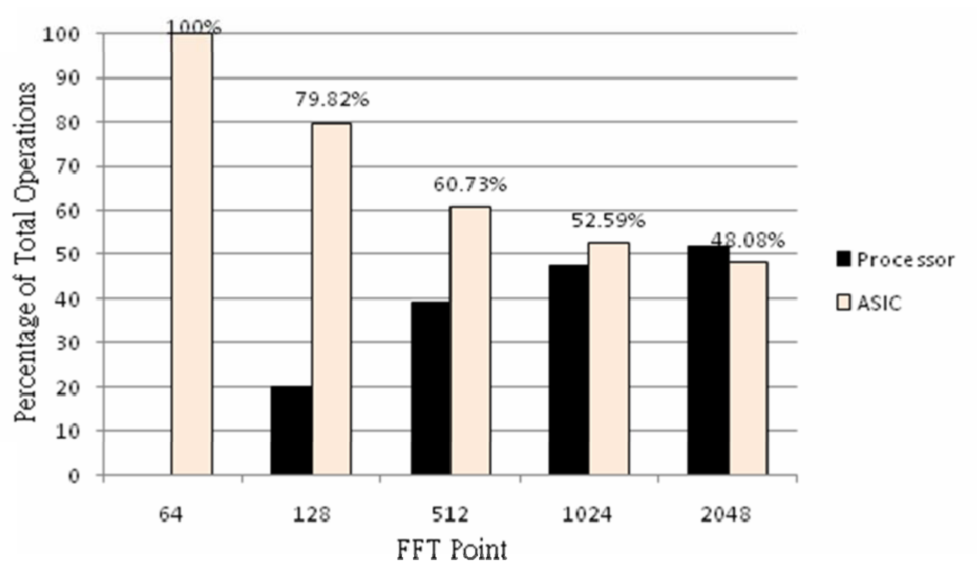


Fig. 3.12 Percentage of total operations in different FFT size

Chapter 4 Implementation of the ASIC Architecture

4.1 Classification Pipeline-Based FFT Architectures

In the domain of implementation of FFT processor, two architectures are commonly used. One is pipelined-based FFT, the other is memory based FFT. The pipelined architecture consumes a relatively large chip area compared with memory based architecture, because the pipelined architecture may need more complex value multipliers and complex value adders. But pipelined architecture lets clock rate is comparatively low as the same frequency of the sampling rate to meet real-time FFT processing. Because ASIC can be regarded as an accelerator and the length branch FFT of ASIC is only 64-point FFT computation, we adopt pipelined branch FFT of ASIC to promote throughput. In the following subsections, it introduces and compares various pipelined-based FFT architectures.

4.1.1 Radix-2 Multipath Delay Commutator (R2MDC)

The block diagram of N-point radix-2 DIF MDC pipelined FFT architecture is shown in Fig.4.1 [15]. The elements between PEs consist of shift registers and a commutator switch which are used to form a proper set of data for the next PE. First, input subcarriers written into upper shift registers until (N/2+1)th subcarrier inserted, and then input subcarriers go into butterfly directly by multiplexer.

Therefore, 1st and (N+1)th subcarriers go into butterfly simultaneously, and then the adder processing results select upper path. The subtraction processing results multiplied by twiddle factor which are written into lower N/4 shift registers and the adder processing results written into upper N/4 shift registers at the same time. The first butterfly adder results select lower path by switch. Therefore, the first butterfly 1st and (N/4+1)th results go into second butterfly and the first butterfly subtraction results written into upper N/4 shift registers simultaneously. By the same way, process entire subcarriers. The R2MDC architecture contains $\log_2 N$-2 multipliers, $2\log_2 N$ adders and 1.5N-2 registers. Butterfly and multipliers only have 50% utilization rate.

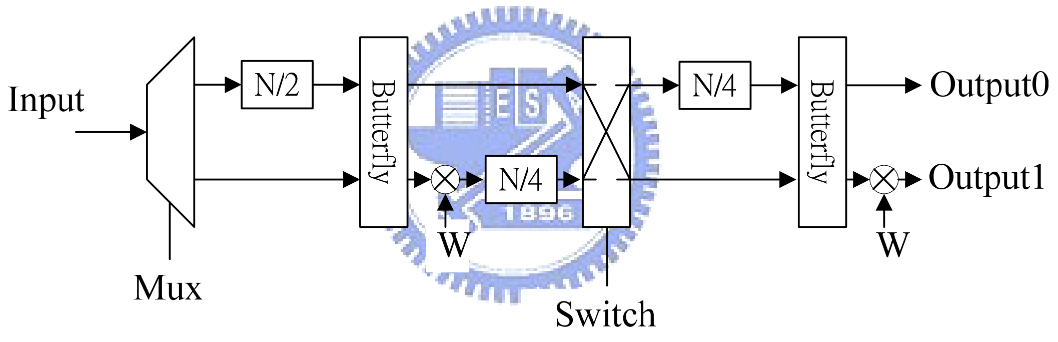


Fig. 4.1 R2MDC architecture

4.1.2 Radix-4 Multipath Delay Commutator (R4MDC)

The block diagram of N-point radix-4 DIF MDC pipelined FFT processor is shown in Fig.4.2 [15]. It is similar to R2MDC architecture, but butterfly implement radix-4 algorithm. Because it has four paths, butterfly and multipliers have 25% utilization rate. It contains $3\log_4 N$-3 multipliers, $8\log_4 N$ adders and 2.5N-4 registers.

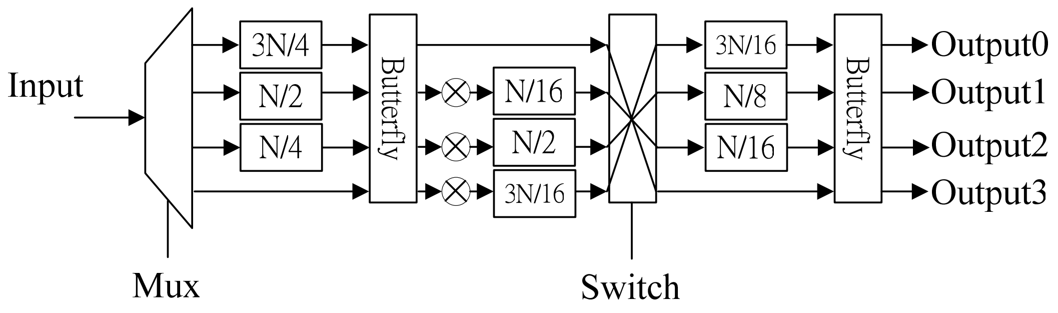


Fig. 4.2 R4MDC architecture

4.1.3 Radix-2^2 Multipath Delay Commutator (R2^2MDC)

In order to implement radix-4 algorithm, it uses cascade of radix-2 architectures as shown in Fig.4.3 [16]. We can see that this architecture contains 2$\log_4$N-2 multipliers, 4$\log_4$N adders and 1.5N-2 registers. It is better than R4MDC.

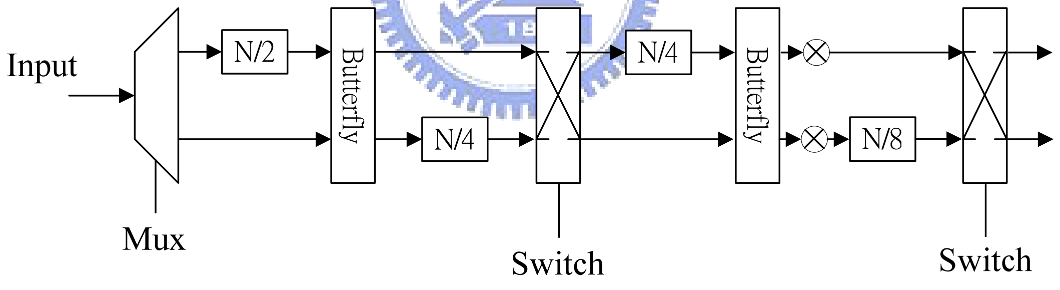


Fig. 4.3 R2^2MDC architecture

4.1.4 Radix-8 Multipath Delay Commutator (R8MDC)

The block diagram of N-point radix-8 DIF MDC pipelined FFT processor is shown in Fig.4.4. It is similar to R2MDC architecture, but butterfly implement radix-8 algorithm. It contains 7$\log_8$N-7 multipliers, (24+2T)$\log_8$N adders and 4.5N-8 registers.

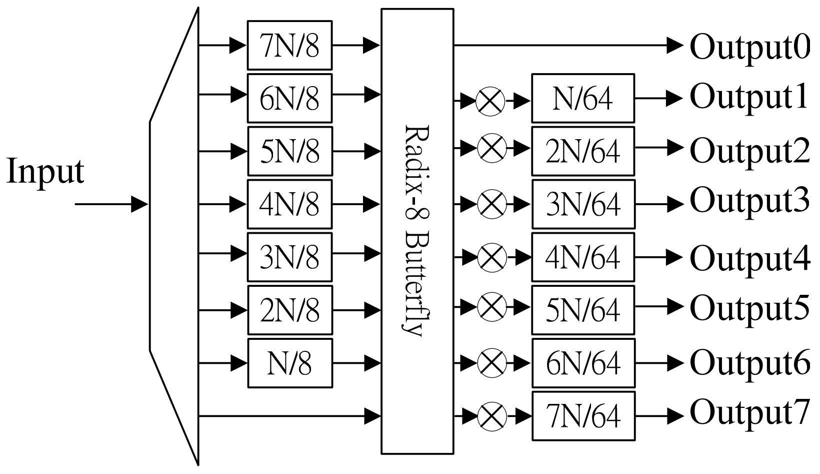


Fig. 4.4 R8MDC architecture

4.1.5 Radix-2^3 Multipath Delay Commutator (R2^3MDC)

In order to implement radix-8 algorithm, it uses cascade of radix-2 architectures as shown in Fig.4.5 [16]. We can see that this architecture contains $\log_2 N/3-1$ multipliers and 1.5N-2 registers.

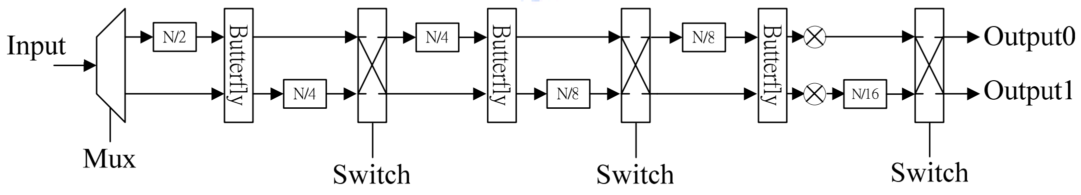


Fig. 4.5 R2^3MDC architecture

4.1.6 Radix-2 Single-Path Delay Feedback (R2SDF)

The butterfly unit in Fig.4.6 shows two kinds of operation modes. In operation mode1, PE pushes input data into the last location of shift register and pops the data from the first location to output port. In operation mode2, the output data of addition part of butterfly unit is directly passed to the next stage and the output data from the subtraction part of butterfly unit is written back to shift register. The block diagram of 16-point radix-2 DIF SDF pipelined FFT processor is shown in Fig.4.7 [17]. First, the input subcarriers are stored into 1st shift registers until the 9th input subcarrier which is sent to butterfly simultaneously. The adder processing results are sent into next stage shift registers and the subtraction processing results are written into the same stage shift registers.

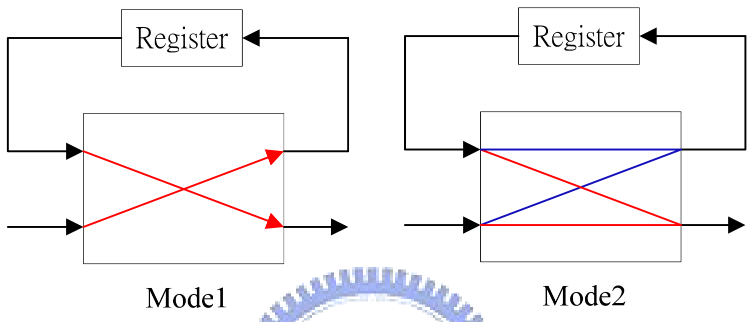


Fig. 4.6 Operation mode1 and mode2

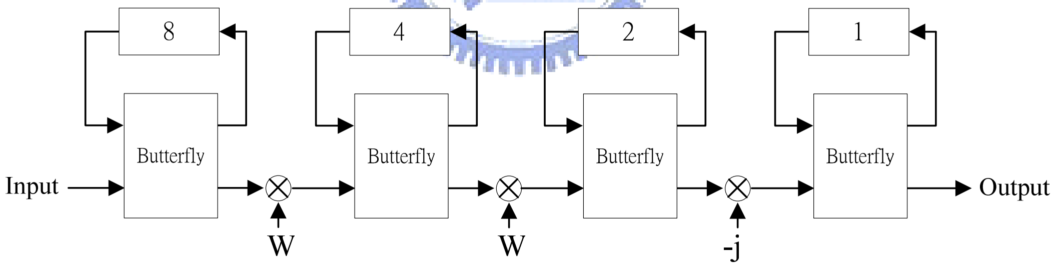


Fig. 4.7 16-point radix-2 DIF SDF architecture

Therefore, the butterfly unit provides a complete feedback loop. We can see that this architecture contains $\log_2 N-2$ multipliers, $2\log_2 N$ adders and N-1 registers.

4.1.7 Radix-4 Single-Path Delay Feedback (R4SDF)

R4SDF architecture is similar to R2SDF architecture is shown in Fig.4.8

[18][19]. It can be seen that radix-4 algorithm needs less multipliers than radix-2 algorithm, but the butterfly unit and multipliers only have 25% utilization rate. It contains $\log_4 N$-1 multipliers, $8\log_4 N$ adders and N-1 registers.

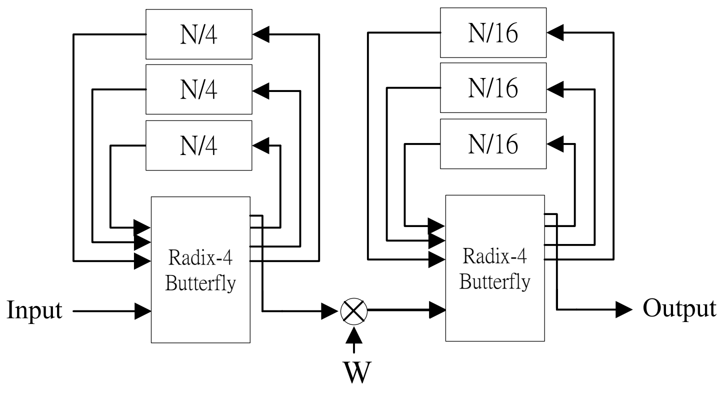


Fig. 4.8 R4SDF architecture

4.1.8 Radix-2^2 Single-Path Delay Feedback (R2^2SDF)

It uses cascade of two radix-2 butterflies to form radix-4 algorithm. Its' shift registers and multipliers are the same with R4SDF architecture as shown in Fig.4.9 [16].

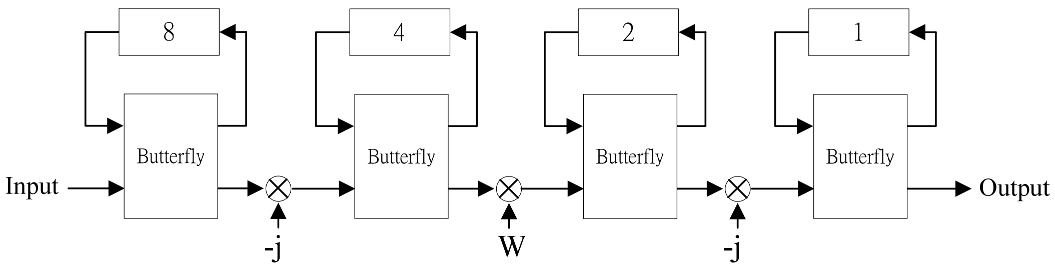


Fig. 4.9 16-point R2^2SDF architecture

4.1.9 Radix-8 Single-Path Delay Feedback (R8SDF)

It uses cascade of three radix-2 butterflies to form radix-8 algorithm as shown in Fig.4.10. It contains $\log_8 N-1$ multipliers, $(24+2T)\log_8 N$ adders and N-1 registers.

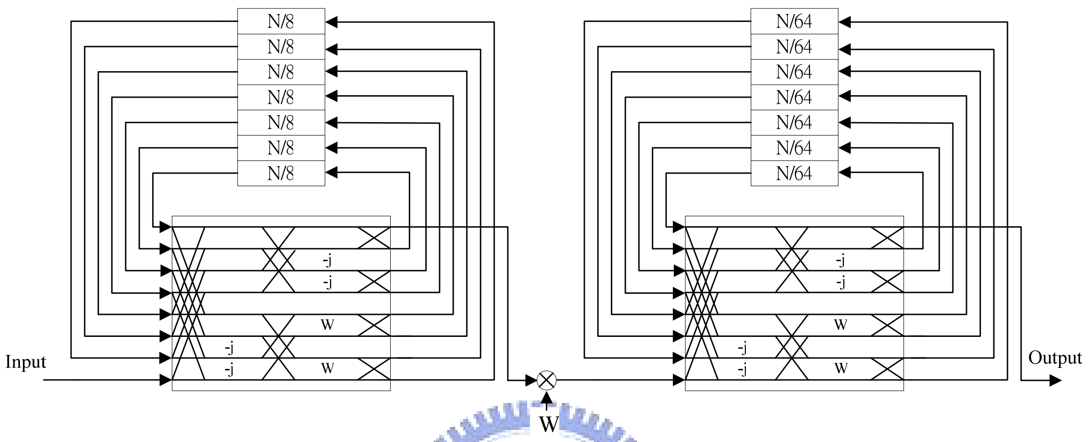


Fig. 4.10 R8SDF architecture

4.1.10 Radix-2^3 Single-Path Delay Feedback ($R2^3$SDF)

It uses cascade of three radix-2 butterflies to form radix-8 algorithm as shown in Fig.4.11 [16]. It contains $\log_2(N)/3-1$ multipliers and N-1 registers.

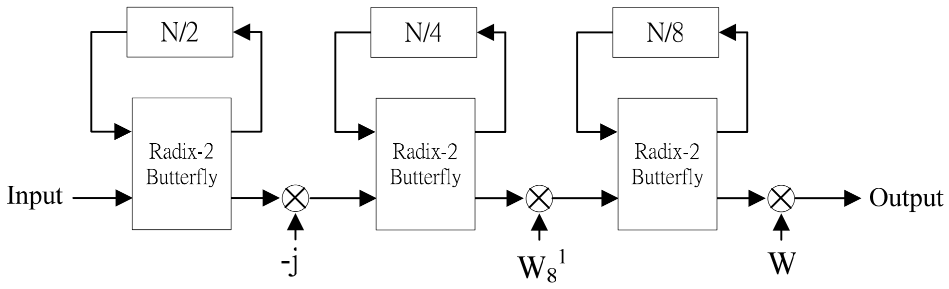


Fig. 4.11 $R2^3$SDF architecture

4.2 Comparison of Pipeline-Based FFT Architectures

According to the previous subsections, we can compare several pipelined architectures for FFT as shown in Table 4.1 and Table 4.2 [20]. One can find that SDF architectures have less area cost and higher utilization rate, but longer FFT computation time, compared with MDC architectures under the same operating clock rate. Besides, the number of switch shows in equation (4.1), where N is number of points, r is radix-r, and stage is total stage. The parameter p is the number of switch in different radix. Afterwards, we can adopt their characteristic to analyze the 64-point branch FFT of ASIC in Table 4.3 and Table 4.4.

$$Number\ of\ \ switch = N \times \left\{ \left(1 - \frac{1}{r}\right) \times (stage - 1) + p \right\} \quad (4.1)$$

Table 4.1 Hardware requirement comparisons of several pipeline structures

Architecture	Complex Multipliers	Complex Adders	Memory Size	Multiplicative Complexity
R2MDC	$\log_2 N-2$	$2\log_2 N$	1.5N-2	Radix-2
R4MDC	$3\log_4 N-3$	$8\log_4 N$	2.5N-4	Radix-4
$R2^2MDC$	$2\log_4 N-2$	$4\log_4 N$	1.5N-2	Radix-2^2
R8MDC	$7\log_8 N-7$	$(24+2T)\log_8 N$	4.5N-8	Radix-8
$R2^3MDC$	$\log_2 N/3-1$	$2\log_2 N$	1.5N-2	Radix-2^3
R2SDF	$\log_2 N-2$	$2\log_2 N$	N-1	Radix-2
R4SDF	$\log_4 N-1$	$8\log_4 N$	N-1	Radix-4
$R2^2SDF$	$\log_4 N-1$	$4\log_4 N$	N-1	Radix-2^2
R8SDF	$\log_8 N-1$	$(24+2T)\log_8 N$	N-1	Radix-8
$R2^3SDF$	$\log_2(N)/3-1$	$(6+2T)\log_8 N$	N-1	Radix-2^3

Table 4.2 Hardware utilization comparisons of several pipeline structures

Architecture	Utilization of Multipliers	Utilization of Adders	Utilization of Registers
R2MDC	50%	50%	50%
R4MDC	25%	25%	25%
$R2^2$MDC	37.5%	50%	50%
R8MDC	12.5%	12.5%	12.5%
R2SDF	50%	50%	100%
R4SDF	75%	25%	100%
$R2^2$SDF	75%	50%	100%
R8SDF	87.5%	12.5%	100%

Table 4.3 Comparison of switch number in different radix algorithm

N=64		
Radix-r	Stage	Number of switch
2	6*(radix-2)	64*{(1-1/2)*5}=192
4	3*(radix-4)	64*{(1-1/4)*2}=96
8	2*(radix-8)	64*{(1-1/8)*1}=56

Table 4.4 Hardware requirement comparisons of 64-point FFT architectures

N=64		
Architecture	Registers	Multipliers
R2MDC	94	5
R4MDC	154	6
$R2^2$MDC	94	4

R8MDC	280	7
R2^{3}MDC	94	2
R2SDF	63	5
R4SDF	63	2
R2^{2}SDF	63	2
R8SDF	63	1
R2^{3}SDF	63	1

In Fig.4.12, it lets multipliers, adders and registers with the 16-bits precision to calculate hardware gate count [28]. It is synthesized at 150 MHz for TSMC 0.18 μm single-poly six-metal CMOS technology using Synopsys Design Compiler.

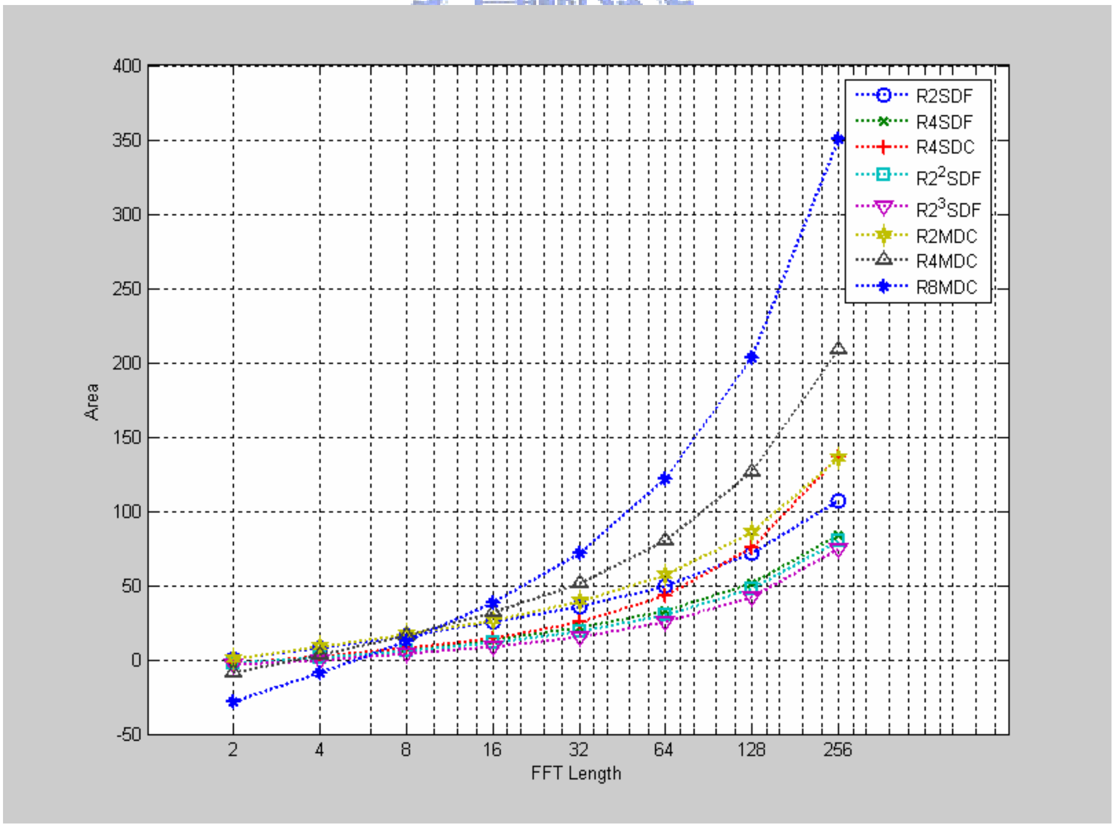


Fig.4.12 Hardware gate count comparison of several pipelined structures

In our proposed system, the 64-point branch FFT of ASIC plays an accelerated role. Therefore, the previous subsections we only analyze pipeline-based FFT architectures which can process data sequence continuously. In Table 4.3, we can find that high radix algorithm not only reduce multipliers, but also lower the number of switch. In Table 4.4, it show that R2^3SDF architecture require less registers and multipliers. Based on area issue, the 64-point branch FFT of ASIC is based on R2^3SDF architecture as shown in Fig. 4.12.

4.3 ASIC Throughput Analysis

If we decide the length branch FFT of ASIC based on section 3.4 analysis, we need to know ASIC throughput requirement. After that, we can design appropriate architecture. Because ASIC occupies partial symbol duration in the time schedule Ⅰ, we only take time schedule Ⅱ in the SISO system into simulation as shown in Fig.4.13. It shows that IEEE 802.11n handles 128-point FFT algorithm requires highest throughput rate, no matter in the SISO/MIMO system. It implies that the 64-point branch FFT of ASIC requires 85 MHz throughput rate at least.

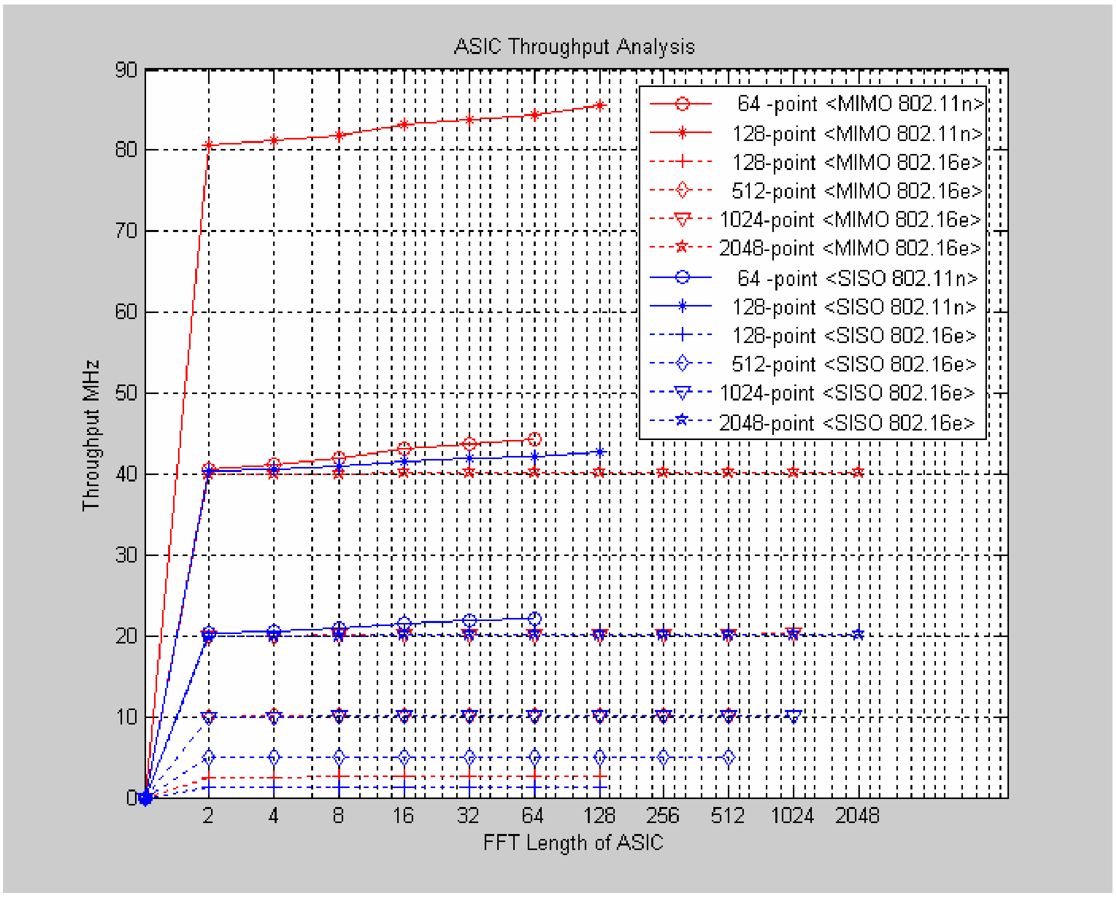


Fig. 4.13 ASIC throughput analysis in different environments

4.4 Implement the 64-point Branch FFT of ASIC

In this section, we will introduce the R2^{3}SDF architecture of each sub-module in great detail and calculate error analysis because of the fixed-point computation.

4.4.1 Complex Multiplier

The 64-point FFT is based on two stage radix-2^3 butterfly and it needs 49 times complex multiplications which excludes from W_8^1 , W_8^3 and W_8^0 . We use the radix-2 index map to divide the 8-point DFT into three steps. Fig.4.14 shows the butterfly of the three-step DIT radix-8 FFT. The twiddle factors, W_8^1 and W_8^3 at the second step are trivial complex multiplications, because they can be written as $\sqrt{2}/2(1-j)$ and $\sqrt{2}/2(-1-j)$. Thus, a complex multiplication with one of the two coefficients and a real multiplication, whose hardware can be realized by shifters and adders in Fig.4.15 .

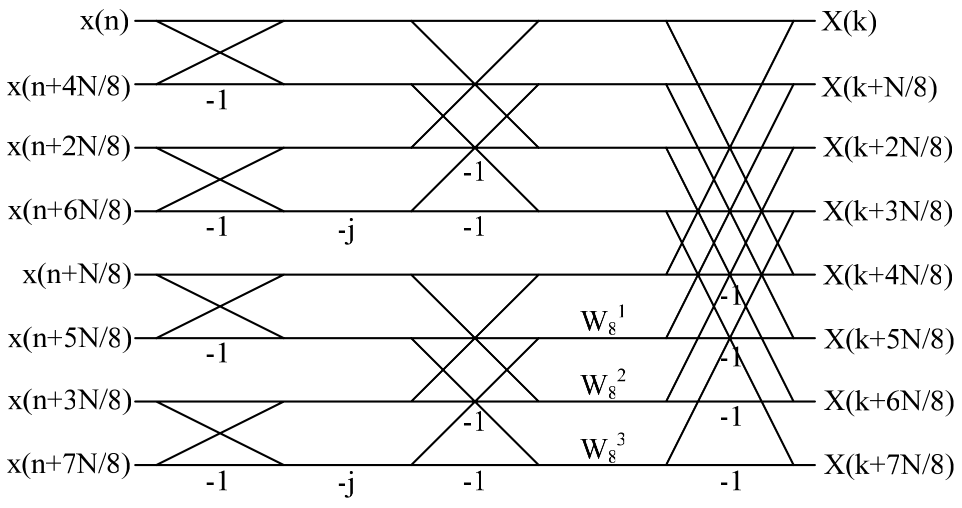


Fig. 4.14 Radix-2^3 FFT butterfly unit

$$\sqrt{2}/2 = 0.70710678 = 2^{-1} + 2^{-3} + 2^{-4} + 2^{-6} + 2^{-8} + 2^{-9}$$

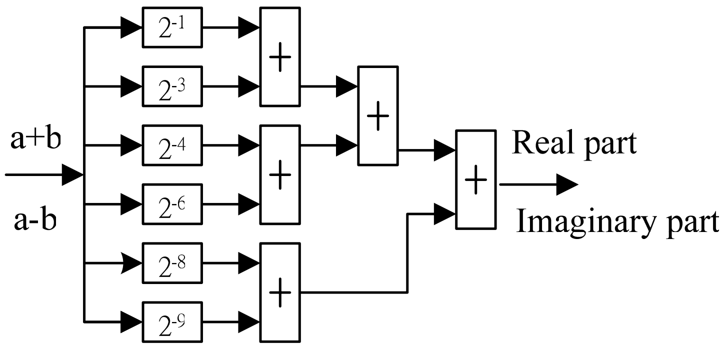


Fig. 4.15 Implementation hardware of multiplication with $\sqrt{2}/2$

The multiplication by –j can be realized with no extra hardware cost by simply interchanging the real and imaginary part of the product as shown in equation (4.2).

$$(a+bj)\times(-j) = b - aj \quad (4.2)$$

One complex multiplier can be realized by four real multiplications and two real additions as shown in Fig. 4.16. Its mathematical form can be expressed as equation (4.3).

$$(a+bj)\times(c+dj) = (ac-bd) + j(bc+ad) \quad (4.3)$$

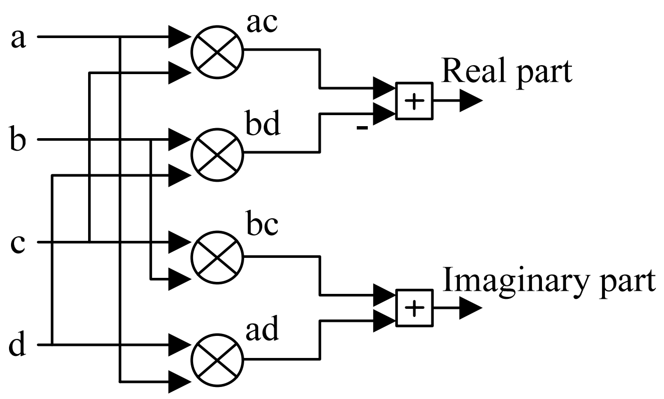


Fig. 4.16 Complex multiplier with four real multiplications and two real additions

4.4.2 Twiddle Factor Coefficient of ROM Table

N-point radix-8 FFT implementations can require seven complex twiddle factor coefficients $W_N^n, W_N^{2n}, \ldots, W_N^{7n}$. Such implementations can require a twiddle factor ROM table to store the real and imaginary parts of these values which have phase angles in the range (0, 2π) in the complex plane. If we store all required coefficient values in a ROM table, we must use a large chip area. Thus, this subsection presents a method to reduce the size of the twiddle factor ROM table.

It is only necessary to store the twiddle factor coefficients between the interval 0~N/8 [21]. We denote the interval 0~N/8 as region 0. The remaining interval regions are listed in Fig.4.17 and Table 4.5. The storage coefficients in region 0 are only in (0, $\pi/4$) to save hardware cost because it can represent all the angles in (0, 2π) by exploiting the symmetry of the sine and cosine functions. This means that the sine of elements in (0, $\pi/4$) are equal to the cosine of elements in ($\pi/4$, $\pi/2$) and vice versa. Thus, if the values in the region 0 are known (stored in a reduced size ROM), the values from all the regions can be computed [30].

Table 4.5 Interval regions of twiddle factor design

No	Region	Interval	Boundary

(a)	0	$0 \le m \le N/8$	Boundary0=0
			Boundary1=N/8

(b)	1	$N/8+1 \le m \le N/4-1$	Boundary2=(N/4)-1
(c)	2	$N/4 \le m \le 3N/8$	Boundary3=3N/8
(d)	3	$3N/8+1 \le m \le N/2-1$	Boundary4=(N/2)-1
(e)	4	$N/2 \le m \le 5N/8$	Boundary5=5N/8
(f)	5	$5N/8+1 \le m \le 3N/4-1$	Boundary6=3N/4-1
(g)	6	$3N/4 \le m \le 7N/8$	Boundary7=7N/8
(h)	7	$7N/8+1 \le m \le N-1$	

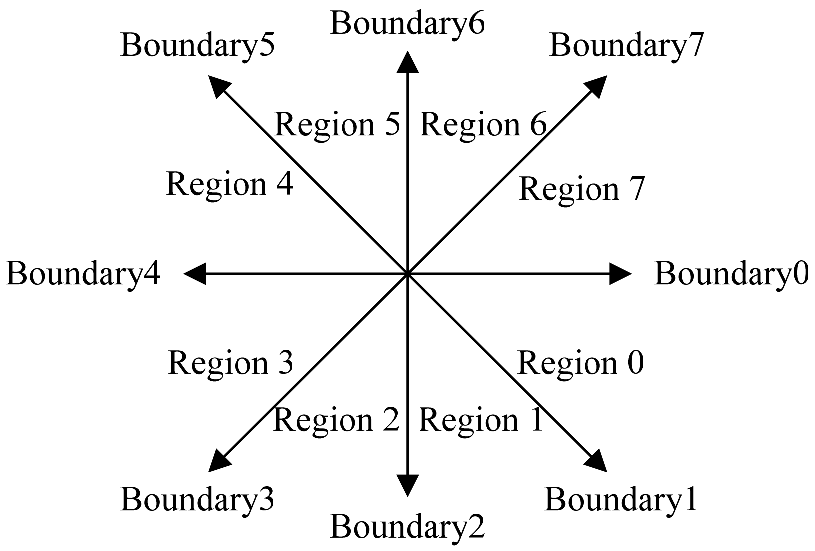


Fig. 4.17 Twiddle factor boundary diagram

When the twiddle factor address generator calculates the address in each region, the corresponding address of region 0 are given by equation (4.4) through equation (4.11). Table 4.6 lists the parameters used in equation (4.4) through (4.11). After that, we can get the twiddle factor coefficients of remaining region from region 0 as shown in Table 4.7.

Table 4.6 Description of twiddle factor parameters

Parameter	Description
$W_N^{nk} = e^{-j\frac{2\pi nk}{N}}$	Twiddle factor coefficient
N	FFT length size
m	The actual address
t	$\log_2 N/2$
Rregion0_addr	The real data value of the region0 address
Iregion0_addr	The image data value of the region0 address
'~'	A complement operation

$$No.(a)\quad region0\_addr = actual\_addr[t-2:0] \quad 0 \le m \le \frac{N}{8}$$

$$actual\_data = [R_{region0\_addr}] + j[I_{region0\_addr}] \qquad (4.4)$$

$$No.(b)\quad region0\_addr = \sim actual\_addr[t-2:0]+1 \quad \frac{N}{8}+1 \le m \le \frac{N}{4}-1$$

$$actual\_data = [\sim I_{region0\_addr}] + j[\sim R_{region0\_addr}] \qquad (4.5)$$

$$No.(c)\quad region0\_addr = actual\_addr[t-2:0] \quad \frac{N}{4} \le m \le \frac{3N}{8}$$

$$actual\_data = [I_{region0\_addr}] + j[\sim R_{region0\_addr}] \qquad (4.6)$$

$$No.(d)\quad region0\_addr = \sim actual\_addr[t-2:0]+1 \quad \frac{3N}{8}+1 \le m \le \frac{N}{2}-1$$

$$actual\_data = [\sim R_{region0\_addr}] + j[I_{region0\_addr}] \qquad (4.7)$$

$$No.(e)\quad region0\_addr = actual\_addr[t-2:0] \quad \frac{N}{2} \le m \le \frac{5N}{8}$$

$$actual\_data = [\sim R_{region0\_addr}] + j[\sim I_{region0\_addr}] \qquad (4.8)$$

$$No.(f)\quad region0\_addr = \sim actual\_addr[t-2:0]+1 \quad \frac{5N}{8}+1 \le m \le \frac{3N}{4}-1$$

$$actual\_data = [I_{region0\_addr}] + j[R_{region0\_addr}] \qquad (4.9)$$

$$No.(g)\quad region0\_addr = actual\_addr[t-2:0] \quad \frac{3N}{4} \le m \le \frac{7N}{8}$$

$$actual\_data = [-I_{region0\_addr}] + j[R_{region0\_addr}] \qquad (4.10)$$

$$No.(h)\quad region0\_addr = \sim actual\_addr[t-2:0]+1 \quad \frac{7N}{8}+1 \le m \le N-1$$

$$actual\_data = [R_{region0\_addr}] + j[\sim I_{region0\_addr}] \qquad (4.11)$$

Table 4.7 Twiddle factor coefficient values in the region 0

Address (m)	Coefficient		16-bits quantized coefficient	
	Real	Image	Real	Image
0	1.000000	0.000000	0100000000000000	0000000000000000
1	0.995185	-0.098017	0011111110110001	1111100110111010
2	0.980785	-0.195090	0011111011000101	1111001110000100
3	0.956940	-0.290285	0011110100111111	1110110101101100
4	0.923880	-0.382683	0011101100100001	1110011110000010
5	0.881921	-0.471397	0011100001110001	1110000111010101
6	0.831470	-0.555570	0011010100110111	1101110001110010
7	0.773010	-0.634393	0011000101111001	1101011101100110
8	0.707107	-0.707107	0010110101000001	1101001010111111

Block diagram of the 64-point branch FFT is shown in Fig.4.18. BF2 I implies that it has two modes as shown in Fig.4.6 and BF2 II needs to process (-j) multiplication additionally. In this architecture, it only needs 63 registers and 1 multiplier.

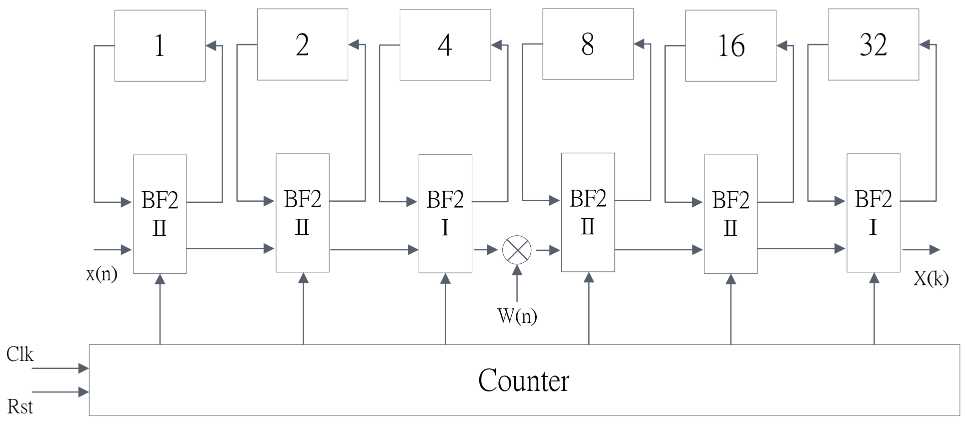


Fig.4.18 R2^{3}SDF Pipelined FFT Architecture for N= 64

4.5 Error Analysis

In the case of FFT hardware implementation, the finite bit-width must be considered because of the fixed-point computation. Many statistical error analysis papers on FFT implementations are proposed [22][23][24]. Assume the input sequence of FFT x(n) is a sequence of finite-valued and white complex numbers. The variance of x(n) can be expressed as

$$\sigma_x^2 = \frac{1}{N}\sum_{n=0}^{N-1}(x(n)-\mu_x)^2 = \frac{1}{N}\sum_{n=0}^{N-1}(x(n))^2 \tag{4.12}$$

where μ_x is the mean of x(n) and μ_x=0. The SQNR (Signal to Quantization Noise Ratio) is defined as

$$SQNR = \frac{\sigma_x^2}{\sigma_q^2} \tag{4.13}$$

Where σ_x^2 is the variance of output and σ_q^2 is the variance of the quantization error. For an N-point FFT module with input of which real and imaginary parts are uniformly distributed in $(-\frac{1}{\sqrt{2}}N, \frac{1}{\sqrt{2}}N)$, the variance [23] of the output is

$$\sigma_x^2 = \frac{1}{3N} \tag{4.14}$$

From (4.13) and (4.14), the SQNR [24] of the conventional FFT implementation can be carried out：

$$SQNR_{FFT} = \frac{2^{2B}}{5N-4m-3} \tag{4.15}$$

Where B is the bit-width of the input sequence and m=log$_2$N.

In Fig.4.19, it shows equation (4.15) with IEEE 802.11n/16e standards which include five FFT sizes. The more rounding stages, the more noise will be produced. Because long-length FFT will decrease SQNR, it needs to increase bit-width. It will cause more power consumption and area cost.

In this chapter, we introduce various pipeline-based FFT architectures and then compare their characteristic to evaluate our proposed system based on throughput rate and hardware cost analysis. After that, it shows detailed sub-module architectures and analyzes noise issue finally.

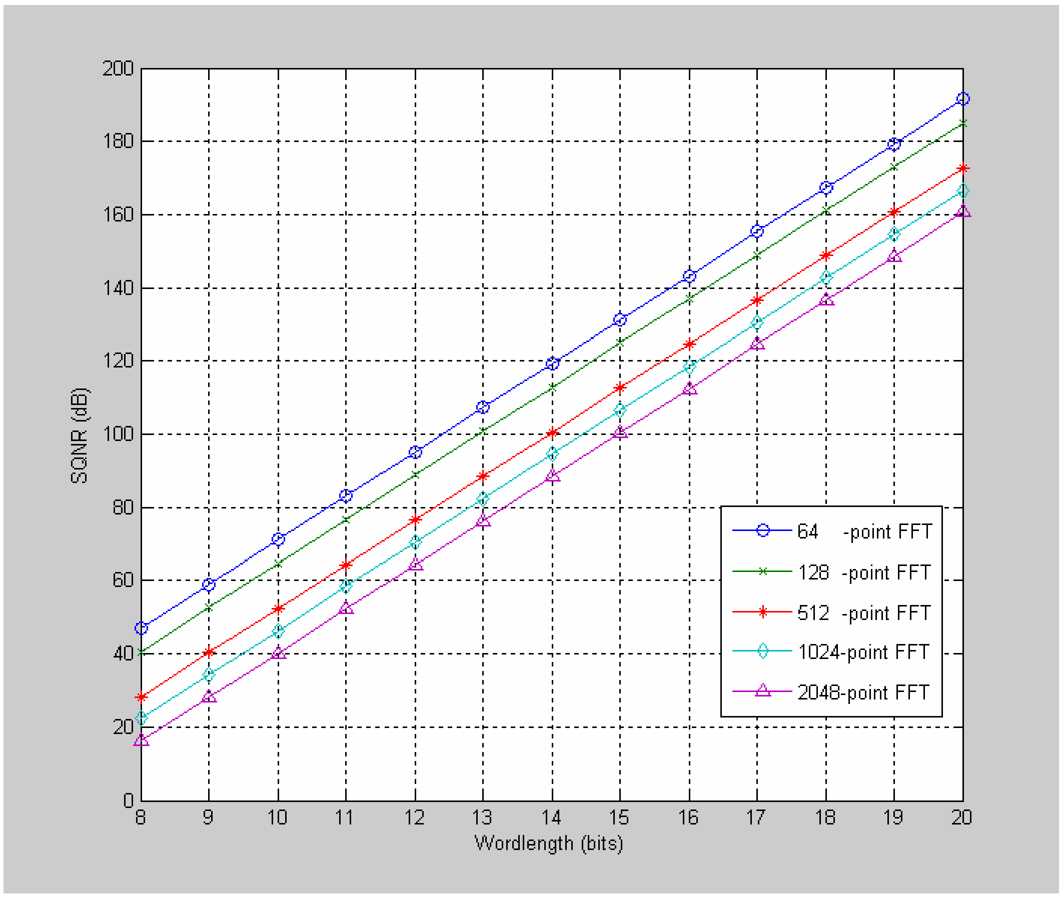


Fig. 4.19 Noise analysis with different FFT length

Chapter 5 System Verification and Simulation Results

5.1 System Environments

In the proposed system, the FFT module handles IEEE 802.11n/16e standards. The FFT sizes include 64-point、128-point、512-point、1024-point and 2048-point FFT computation. We hope that the proposed system can change FFT size flexibly and meet specification requirements simultaneously. Therefore, system verification will cover that the 64-point branch FFT of ASIC verification, variable-length FFT module functional simulation and time schedule simulation.

5.2 The 64-point Branch FFT Verification

After functional validation, the branch FFT is synthesized for TSMC 0.18μm single-poly six-metal CMOS technology using Synopsys Design Compiler [25][26]. After synthesis, floor planning, P&R, and layout are carried out using Cadence SOC Encounter. Finally, the post-simulation power analysis on the netlists exported from SOC Encounter is carried out using Synopsys PrimePower. Fig.5.1 gives the design flow chart and CAD tools used in the branch FFT of ASIC.

In Fig.5.2, the die size of the 64-point branch FFT is 2270 x 2270 μm^2. It synthesizes with 46236 gate counts which include testing circuits and the minimum clock period is 6.31 ns which is reported by Synopsys Design Compiler.

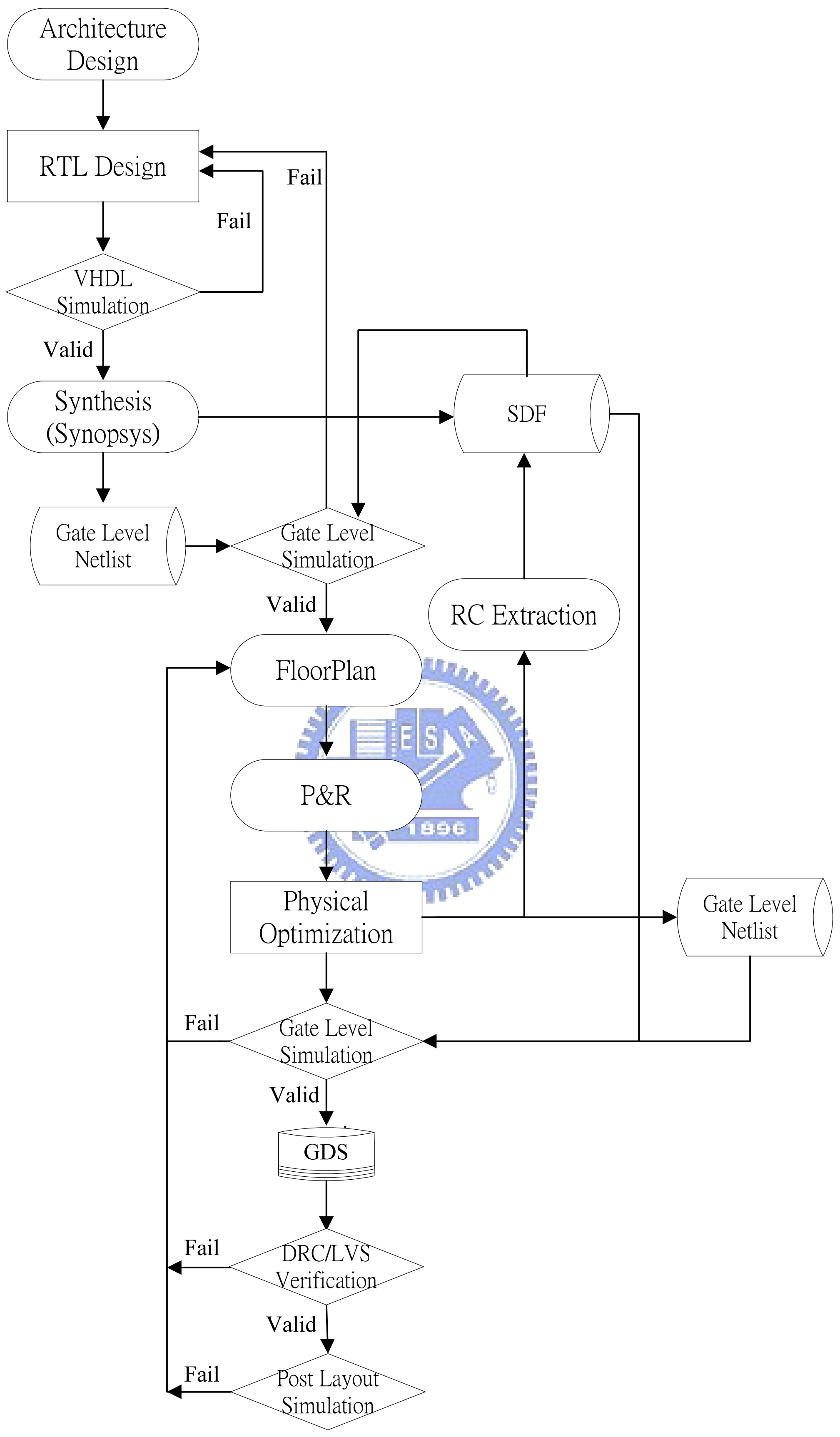


Fig. 5.1 Flow chart of the branch FFT chip design

We adopt full scan test for the chip. The test circuits are inserted during compilation using Synopsys DFT Compiler. The existing flip-flops inside the chips were replaced with scan flip-flops. Test vectors are generated by Synopsys TetraMax. The fault coverage of the R2^{3}SDF FFT is up to 99.40%.

Based on throughput rate analysis, the 64-point branch FFT should work upon 85 MHz. Therefore, the chip summary can be listed in Table 5.1.

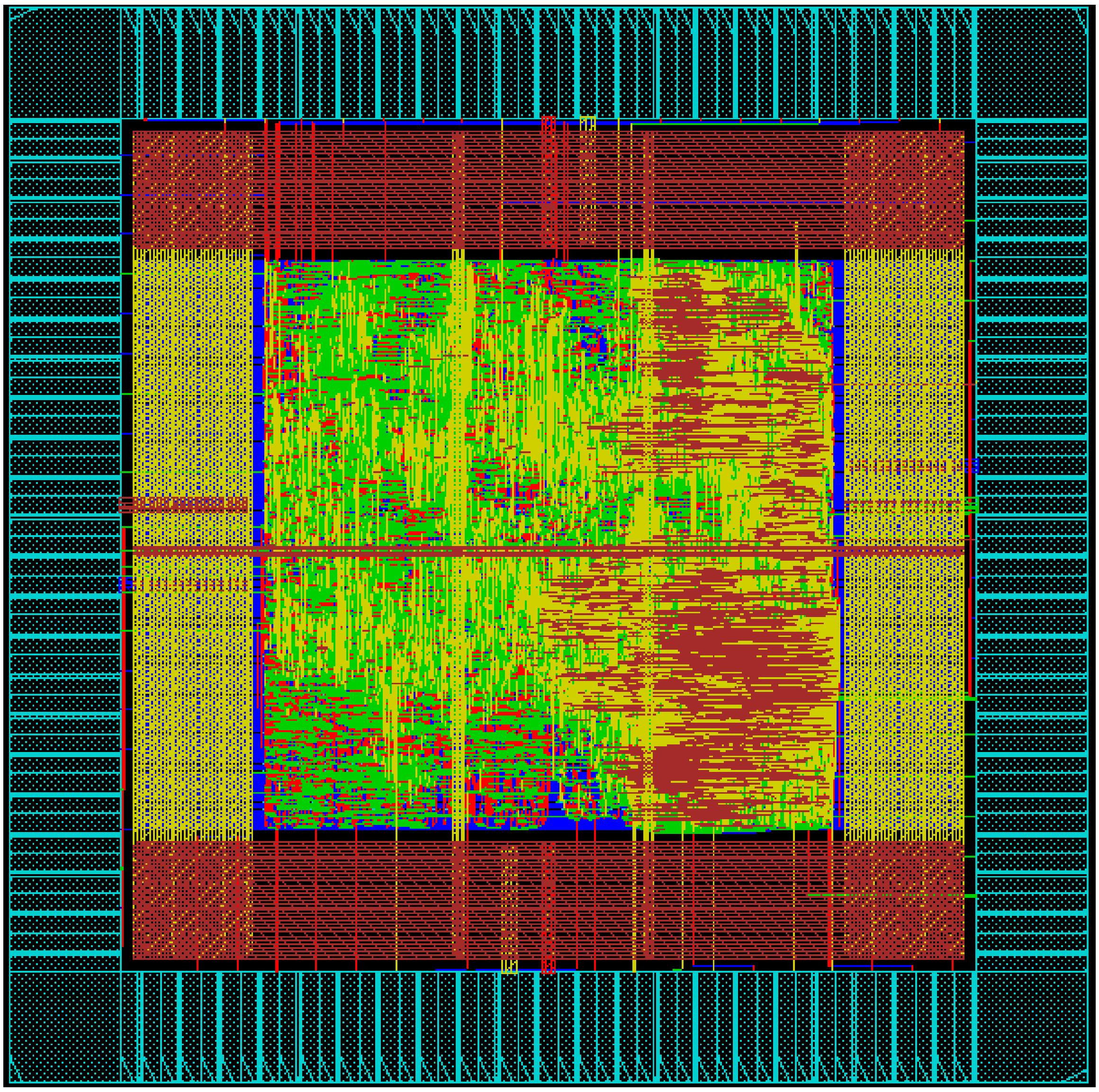


Fig. 5.2 Layout view of the 64-point R2^{3}SDF FFT structure

Table 5.1 Chip summary of 64-point R2^3SDF FFT

Design	R2^3SDF
Clock rate	87 MHz
Datapath width	16 bits
Latency	71 cycles
Synthesized gate count	46236 (with testing circuits)
Core size	1195 x 1195 μm^2
Die size	2270 x 2270 μm^2
Core power	112.6mW @ 87 MHz
Die power	158.1mW @ 87 MHz

In Fig.5.3, it shows test bench of the 64-point branch FFT. There are input control and output compare. We can read input signal from in.txt which generated by MATLAB, and output can be compared the 64-point branch FFT with MATLAB behavior model output. This self-check test bench can verify a lot of test patterns and we can check the output signal are within error thresh or not efficiently [27].

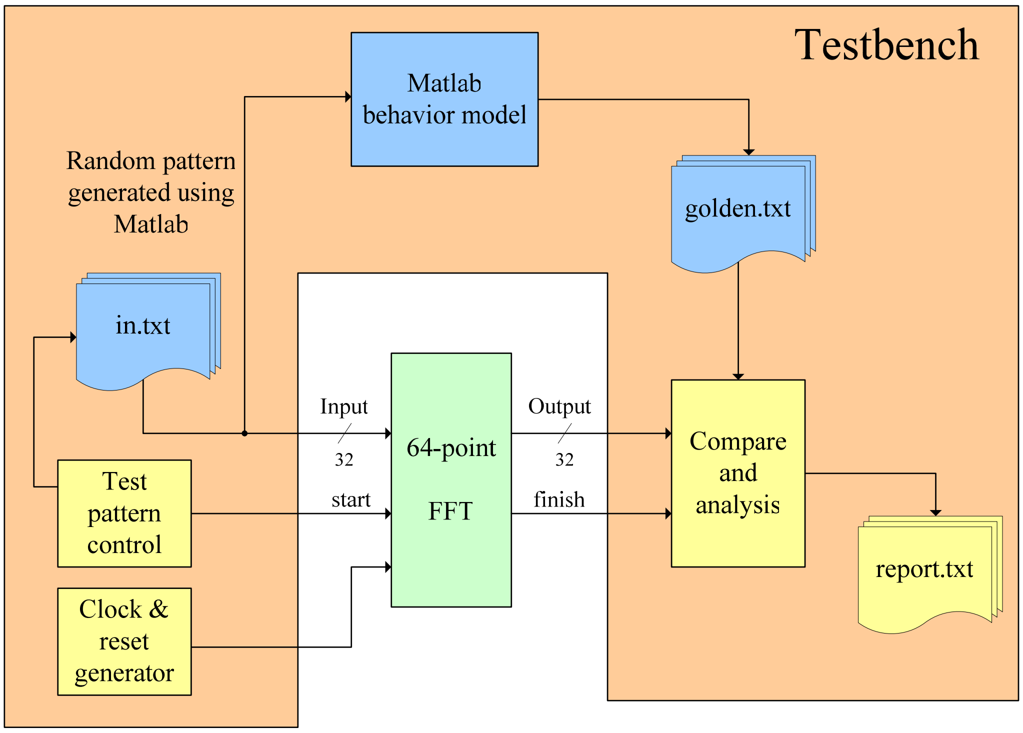


Fig. 5.3 Test bench for the 64-point branch FFT

We use FPGA to implement design and the synthesis report shown in Table 5.2, the report of the FPGA timing is very different from the report of the ASIC timing, which is only for reference. Because the characteristic of FPGA is for verification, the timing is not very important by FPGA prototyping.

Table 5.2 Xilinx FPGA synthesis report

Target Device	3s400ft256-5
Slices	1554 (43%)
Slices Flip Flops	667 (9%)
4-Input LUTs	2611(36%)
Post-Map Timing	17.120ns (14.820ns logic, 2.300ns route)
Post-P&R Timing	30.268ns (12.899ns logic, 17.369ns route)

Block diagram of FPGA verification is shown in Fig.5.4. It shows that we introduce an extra controller which generates memory address signals and rs-232 handshaking signals. It generates test patterns by MATLAB and checks the branch FFT output sequences which are within error thresh or not.

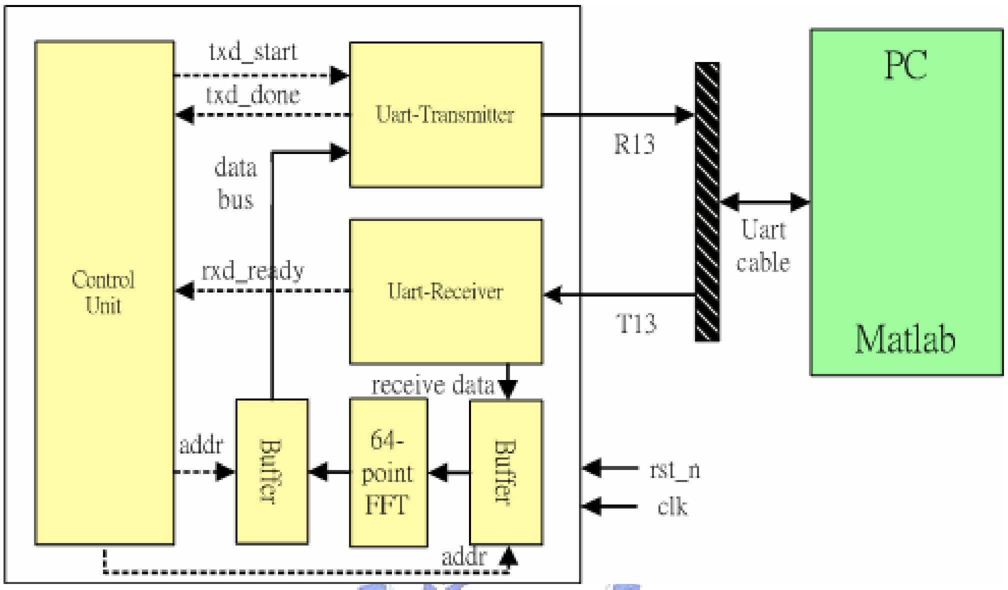


Fig. 5.4 FPGA measurement plan

5.3 MATLAB Simulation and Analysis

In the previous subsections, we only verify the 64-point branch FFT accelerator. Therefore, we will model processor behavior to verify the proposed system as shown in Fig.5.5. First, user needs to declare that FFT size and symbols number and then MATLAB models processor behavior to generate decomposed sequences. The part of algorithm check, it can verify variable length FFT computation which covers FFT sizes of IEEE 802.11n/16e standards. We can compare the output signal with golden.txt which are within error thresh or not.

The part of schedule check part, we verify time schedule in the SISO/MIMO systems as shown in Fig.3.3, Fig.3.5 and Fig.3.7. Schedule control module and test pattern module control data flow. Therefore, processor and ASIC can process data samples by turns based on the proposed time schedule.

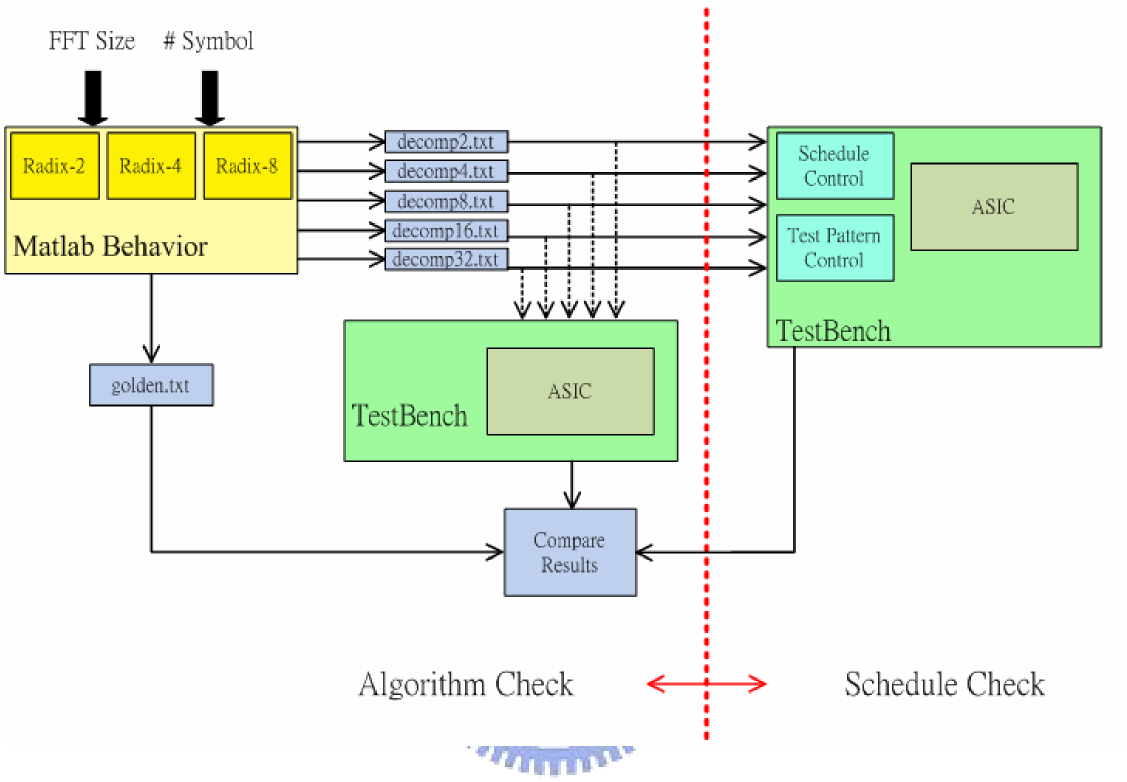


Fig. 5.5 Test bench for proposed system

In this chapter, it shows verification of 64-point branch FFT which covers cell-based ASIC flow and FPGA measurement. After that, we model processor behavior to verify variable length FFT computation and time schedule in the SISO/MIMO systems. Therefore, proposed system can complete variable length FFT algorithm and process multiple data streams to meet IEEE 802.11n/16e standards.

Chapter 6 Conclusion and Future Work

6.1 Conclusion

Because processor is popular in recent years, we intend that the FFT module can combine processor with ASIC to form the flexible system. ASIC plays an accelerated role in the proposed system. Based on FFT computational complexity analysis, it shows different length branch FFT of ASIC which affects processor performance. Therefore, it can provide user two anticipation as below：

1) Processor needs to spare how much computational performance at least for proposed FFT system.
2) In terms of processor computational performance, we can decide the branch FFT length of ASIC.

Because we adopt processor can contribute 800 MOPS in this thesis, the accelerated branch FFT is 64-point FFT algorithm. After that, we analyze various pipeline-based FFT architectures. Based on area cost issue, the 64-point branch FFT is implemented by R2^3SDF architecture. After throughput analysis, the accelerated branch FFT must work upon 85 MHz to meet IEEE 802.11n/16e standards. The branch FFT of ASIC has 16-bits wordlength and synthesized using Synopsys TSMC 0.18 μm process. After that, physical design is carried out using Cadence SOC Encounter and the chip summary is depicted in Table 5.1.

Finally, we not only verify the 64-point branch FFT on FPGA but also check the proposed time schedule which covers 64-point、128-point、512-point、1024-point and 2048-point FFT algorithm in the SISO/MIMO systems.

6.2 Future Work

Because the processor is virtual in this thesis, we provide one method to setup up proposed system. In Xilinx Spartn-3 FPGA, it has an embedded processor. Therefore, the processor can be entirely built by writing C- language and the 64-point branch FFT can be loaded to FPGA as an accelerator.

In this thesis, the processor performance analysis is based on radix-2/4/8 algorithms. Because processor computational loading is based on operations, we can try to use higher radix algorithm to reduce multiplications. The branch FFT is implemented with a high specification of 16-bits wordlength while the output is also 16-bits and 2^6 scaled. The datapath can be designed more carefully if a precise error analysis was done. Hence, the resource cost will be reduced while keeping specification requirements. The shift registers is another issue. For a bigger N, the shift registers will cause more power consumption and area cost than using memory access. Therefore, how to improve the efficiency and simplify the memory access scheme in the long length branch FFT module is left for future work.

Reference

[1] Mujtaba *et al.*, TGn Sync Proposal Tech. Specification for IEEE 802.11 Task Group 2005, IEEE 802.11-04/0889r3.

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

[3] Chih-Wei Liu, "*Introduce to FFT Processors,*" VLSI Signal Processing Lab Department of Electronics Engineering, National Chiao-Tung University.

[4] S. He and M. Torkelson, "*Designing Pipeline FFT Processor for OFDM (De)Modulation,*" in Proc. URSI Int. Symp. Signals, Systems, and Electronics, vol. 29, Oct. 1998, pp. 257-262.

[5] Ray Andraka, Andraka Consulting Group, Inc., 16 Arcadia Drive, North Kingstown, RI "*A Survey of CORDIC Algorithm for FPGA Based Computers,*" ACM Press, 1998 New York, NY, USA.

[6] "*Supplement to IEEE Standard for Information Technology Telecommunications and Information Exchange between Systems-Specific Requirements. Part 11：Wireless LAN Medium Access Control and Physical Layer,*" *IEEE 802.11a*, 1999.

[7] 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.

[8] Yutai Ma, "*An Effective Memory Addressing Scheme for FFT Processors,*" IEEE Transactions on Signal Processing, Vol. 47 Issue：3, pp. 907-911, Mar. 1999.

[9] Yutai Ma and Lars Wanhammar, "*A Hardware Efficient Control of Memory*

Addressing for High-Performance FFT Processors,” IEEE Transactions on Signal Processing, Vol. 48 Issue：3, pp. 917-921, Mar. 2000.

[10] C. L. Wang and C. H. Chang, “*A New Memory-Based FFT Processor for VDSL Transceivers,*” IEEE International Symposium on Circuits and Systems, Vol. 4, pp. 670-673, 2001.

[11] Kun-Lung Chen and Sau-Gee Chen, “*FFT Processor Design for OFDM Systems,*” Department of Electronics Engineering & Institute of Electronics College of Electrical Engineering and Computer Science, National Chiao Tung University, June 2004.

[12] Li-Yun Lin and Tsern-Huei Lee, “*Implementation of A Variable Length FFT Processor for VDSL System,*” Department of Communication Engineering, National Chiao Tung University, June 2004.

[13] “*On-Chip Peripheral Bus Architectures Specifications,*” a complete description of the bus by IBM, its inventor.

[14] S. Sukhsawas and K. Benkrid, “*A High-Level Implementation of A High Performance Pipeline FFT on Virtex-E FPGAs,*” School of Computer Science, Queen’s University Belfast, United Kingdom, 2004.

[15] L. R. Rabiner and B. Gold, *Theory and Application of Digital Signal Processing*. Englewood Cliffs, NJ：Prentice-Hall, 1975.

[16] S. He and M. Torkelson, “*Designing Pipeline FFT Processor for OFDM (De)Mddulation,*” in Proc. ISSSE, pp.257-262, 1998.

[17] E. H. World and A. M. Design, “*Pipeline and Parallel Pipeline FFT Processors for VLSI Implementation,*” IEEE Trans. Comput., vol. C-33, pp. 414-426, May 1984.

[18] A. M. Despain, “*Fourier transform computer using CORDIC iterations,*” IEEE Trans. Comput., vol. C-23, pp.993-1001, Oct. 1974.

[19] G. BI and E. V. Jones, "*A Pipelined FFT Processor for Word-Sequential Data,*" IEEE Trans. Acoust., Speech, Signal Processing, vol. 37, pp. 1982-1985, Dec. 1989.

[20] Shousheng He and Mats Torkelson, "*A New Approach to Pipeline FFT Processor,*" IEEE Parallel Processing Symposium, pp. 766-770, April 1996.

[21] M. Hasan, T. Arslan, "*FFT Coefficient Memory Reduction Technique for OFDM Applications,*" IEEE International Conference on Acoustics, Speed, and Signal Processing, vol. 1, pp. I-1085 – I-1088, May 2002.

[22] P.D. Welch, "*A Fixed-Point Fast Fourier Transform Error Analysis,*" IEEE trans. on audio and electroacoustics, vol. AU-17, no. 2, pp. 151-157, Jun. 1969.

[23] A. V. Oppenheim and C. J. Weinstein, "*Effects of Finite Register Length in Digital Filtering and the Fast Fourier Transform,*" pro. of the IEEE, vol. 60, no.8, pp. 957-976, Aug. 1972.

[24] M. Sundaramurthy and V. U. Reddy, "*Some Results in Fixed-Point Fast Fourier Transform Error Analysis,*" IEEE trans. on computers, pp. 305-308, Mar. 1997.

[25] Synopsys Design Compiler User Guide, Version W-2004.12, Dec. 2004.

[26] Synopsys Design Compiler Reference Manual: Constraints and Timing, Version W-2004.12, Dec. 2004.

[27] Chi-Wei Wu, "*Study on Short-Length FFT Design for Recursive Long-Length FFT Architecture,*" Department of Electrical and Control Engineering, National Chiao Tung University, July 2006.

[28] T. Sansaloni, "*Efficient Pipeline FFT Processors for WLAN MIMO-OFDM Systems,*" IEE electronics letters 15th September 2005 vol. 41 no. 19.

[29] Bing-Juo Chuang, "*Design and Implementation of IEEE 802.11n Based*

Receiver," Department of Communication Engineering, National Chiao Tung University, July 2005.

[30] Li-Yun Lin, "*Implementation of A Variable Length FFT Processor for VDSL system,*" Department of Communication Engineering, National Chiao Tung University, June 2004.