

國立交通大學

電機與控制工程學系

碩士論文

適用於多輸入多輸出正交分頻多工 Wi-MAX 系統之可變長度
快速傅立葉轉換

A Variable FFT for MIMO-OFDM Systems over Wi-MAX
Applications

研究生：葉柏賢

指導教授：蔡尚濶 教授

中華民國九十七年十一月

適用於多輸入多輸出正交分頻多工 Wi-MAX 系統之可變長度快速傅
立葉轉換
A Variable FFT for MIMO-OFDM Systems over Wi-MAX Applications

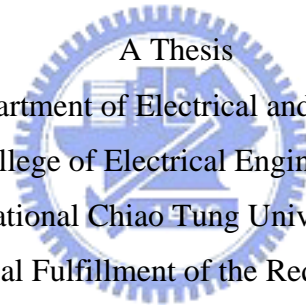
研究生：葉柏賢

Student : Bo-Xian Ye

指導教授：蔡尚濶

Advisor : Shang-Ho Tsai

國立交通大學
電機與控制工程學系
碩士論文



A Thesis
Submitted to Department of Electrical and Control Engineering
College of Electrical Engineering
National Chiao Tung University
in partial Fulfillment of the Requirements
for the Degree of
Master
in
Electrical and Control Engineering

November 2008
Hsinchu, Taiwan, Republic of China

中華民國九十七年十一月

學生：葉柏賢

指導教授：蔡尚澤

國立交通大學電機與控制工程學系（研究所）碩士班

摘 要

在這篇論文，我們介紹一個可以應用於 Wi-MAX 系統中的可變長度快速傅立葉轉換。這個可變長度快速傅立葉轉換可以提供許多快速傅立葉轉換的長度及多天線傳輸。這個 2048/1024/512/128-point 可變長度快速傅立葉轉換是以 radix-2 及 radix-2³ 快速傅立葉轉換演算法。我們也提出一個記憶體分享的方法去減少記憶體的使用。這個方法比較於 R2SDF 的方法可以減少 ROM 表格大小從 1023N/1024 到 N/4，N 為快速傅立葉轉換的長度。此外，我們使用 radix-2³ 快速傅立葉轉換演算法使得複數乘法器的數量減少並且也使用修正的複數乘法器使的所使用的邏輯閘數比較少。如此功率消耗也能更加節省。我們所提出的可變長度快速傅立葉轉換是使用台積電 0.18um CMOS 製程所製造，晶片的面積為 25mm²。當處理器操作於頻率 40MHz 時所需的功率為 181 mW。

A Variable FFT for MIMO-OFDM Systems over Wi-MAX Applications

Student : Bo-Xian Ye

Advisors : Dr. Shang-Ho Tsai

Department (Institute) of Electrical and Control Engineering
National Chiao Tung University

ABSTRACT

In this thesis, we present a variable FFT that it support multiple FFT size and multiple antennas for Wi-MAX systems. The 2048/1024/512/128-point variable FFT is based on radix-2 and radix-2³ FFT algorithm. We propose a memory sharing method to reduce the memory size. This method can reduce the ROM table size from 1023N/1024 to N/4, where N is the FFT size, compared with R2SDF. Furthermore, we use the radix-2³ FFT algorithm to reduce the number of complex multipliers, and the modified complex multiplier leads to a smaller gate count. Thus, the power consumption can be to reduced as well. The proposed variable FFT is fabricated using a TSMC 0.18um CMOS technology with chip area 25 mm². The average dynamic power consumption is 181 mW at 40 MHz operating frequency.

誌 謝

兩年來的研究生活終於要告一個段落了，此篇論文能夠順利的完成首先要感謝的是我的指導教授蔡尚澤教授。在兩年的研究生活中，老師不辭辛苦的一步一步的帶領我們走進通訊晶片設計的領域，也很佩服老師的研究精神及超人的體力，讓我在學習上也有更明確的目標。也希望老師在忙碌之餘能多愛惜自己的身體。也感謝我的口試委員：林源倍教授、簡鳳村教授、董蘭榮教授的經驗提供使得我的論文更加的完整。


另外，感謝 535 實驗室的學長及同學，因為有你們在課業上的幫忙及意見的提供，讓我在修課上的疑惑能夠有很大的幫助。另外，還需感謝實驗室一起打拼的同學，讓我在作研究中可以有更多的思考方式去解決作研究時所遇到的種種困難。也感謝學弟妹們的加入，因為有你們的加入使我的研究生活更加有樂趣。

最後，我要感謝的是我偉大的母親，感謝她一直在背後為了我默默的付出，一直在背後支持著我，因為有妳的支持及鼓勵使得我在作任何事情都能更有力量更有信心。

將此篇論文獻給所有關心我幫助我的人，感謝你們。

A Variable FFT for MIMO-OFDM Systems over Wi-MAX Applications

Bo-Xian Ye



Advisor: Dr. Shang-Ho Tsai
Department of Electrical and Control Engineering
National Chiao Tung University

November 28, 2008

Abstract

In this thesis, we present a variable FFT that it support multiple FFT size and multiple antennas for Wi-MAX systems. The 2048/1024/512/128-point variable FFT is based on radix-2 and radix- 2^3 FFT algorithm. We propose a memory sharing method to reduce the memory size. This method can reduce the memory size from $\frac{1023N}{1024}$ to $\frac{N}{4}$, where N is the FFT size, compared with R2SDF. Furthermore, we use the radix- 2^3 FFT algorithm to reduce the number of complex multipliers, and the modified complex multiplier leads to a smaller gate count. Thus, the power consumption can be to reduced as well. The proposed variable FFT is fabricated using a TSMC 0.18um CMOS technology with chip area 25mm^2 . The average dynamic power consumption is 181mW at 40MHz operating frequency.

Contents

1	Introduction	1
1.1	Motivation and goal	1
1.2	Contributions and Features	3
2	Background	5
2.1	FFT for MIMO systems	5
2.1.1	Algorithm	6
2.1.2	Architecture	9
2.2	Variable FFT	19
2.2.1	Pipeline FFT processor architecture	20
2.2.2	Variable FFT processor architecture	21
3	The proposed variable FFT for MIMO systems	24
3.1	Algorithm	24
3.2	Architecture	27
3.2.1	Module 1 (data reordering)	28
3.2.2	Module 2 to 6 (radix-2 FFT algorithm)	29
3.2.3	Module 7 (radix-2 ³ FFT algorithm)	33
3.2.4	Module 8 (radix-2 ³ FFT algorithm)	33
3.3	Complexity comparison	34
3.4	Simulation	35
4	Chip implementation and verification	37
4.1	Cell-based design flow	39

4.2 Chip summary	43
5 Conclusions	45



List of Figures

2.1	The SFG of 128-point mixed-radix FFT.	8
2.2	Block diagram FFT/IFFT.	9
2.3	Block diagram of the 128/64-point FFT/IFFT processor.	10
2.4	(a) Order of input; (b) Order of output.	11
2.5	Block diagram of Module 1.	11
2.6	Relation between Module 1 input and Module 1 output.	12
2.7	Block diagram of Module 2.	13
2.8	(a) Architecture of multiplexer; (b) Operation mode of multiplexer.	14
2.9	Architecture of four antenna R2SDF FFT 128-point at stage one.	15
2.10	Save data in memory.	15
2.11	Operation of radix-2.	16
2.12	Module 2 memory bank.	16
2.13	Block diagram of Module 3.	17
2.14	Two operation mode.	17
2.15	Eight region of twiddle factor.	18
2.16	Block diagram of Module 4.	19
2.17	Architecture of R2MDC.	21
2.18	Architecture of R2SDF.	21
2.19	Block diagrams of a variable FFT processor.	22
2.20	The SFG of 64-point mixed-radix FFT.	23
3.1	The SFG of stage 1 to stage 5 (radix-2).	27
3.2	The SFG of stage 6 to stage 7 (radix-2 ³).	27
3.3	Block diagram of the variable FFT processor.	28

3.4	The input and output relationship of FFT.	28
3.5	(a) Read and write with column; (b) Read and write with row. . .	29
3.6	Relation between Module 1 input and Module 1 output.	29
3.7	Block diagram of Module 2.	30
3.8	Memory sharing from Module 2 to Module 6.	31
3.9	(a) ROM table at clock cycle 1024 to 2047; (b) ROM table at clock cycle 2048 to 3071.	32
3.10	(a) ROM table at clock cycle 1024 to 2047; (b) ROM table at clock cycle 2048 to 3701.	32
3.11	Analysis for critical path.	33
3.12	The FFT critical path.	33
3.13	System model of SQNR.	35
3.14	SQNR v.s. SNR.	36
4.1	A Design flow.	38
4.2	Simulation environment for variable FFT.	39
4.3	Bit-width in all stage.	40
4.4	BIST circuit.	41
4.5	From Flip-Flop to scan Flip-Flop.	41
4.6	Layout view of the proposed FFT processor.	43

List of Tables

2.1	Mapping table of twiddle factors in different regions.	18
2.2	FFT size in several OFDM systems.	20
3.1	Comparison of hardware requirement.	34
4.1	Expected chip performance of the proposed FFT processor.	44
4.2	Comparison of chip performance.	44



Chapter 1

Introduction

1.1 Motivation and goal

Wi-MAX (Worldwide Interoperability for Microwave Access) is a technique aimed to provide applications in wireless metropolitan area networking (WMAN). This technique was developed by the IEEE 802.16 groups and was adopted by both the IEEE and the ETSI HIPERMAN groups. The IEEE 802.16 group was formed in 1998 to develop an air-interface standard for wireless broadband. At begin, the Wi-MAX solutions targeted on fixed applications, e.g. IEEE 802.16-2004 which is also called as fixed Wi-MAX. In December 2005, the IEEE group completed and approved IEEE 802.16e-2005 standard, which was an amendment to IEEE 802.16e-2004 standard that added mobility support. The IEEE 802.16e-2005 is often called as mobile Wi-MAX. Wi-MAX offers a rich set of features with a lot of flexibility including deployment options and potential service offerings. Some important features of Wi-MAX in physical layer are as follows:

- **OFDM techniques:**

The Wi-MAX physical layer is based on orthogonal frequency division multiplexing (OFDM), which is robust to multipath effect. Thus, it can be used in Non-Line-of-Sight (NLOS) environments. In OFDM-based systems, FFT (Fast Fourier Transform) is key component and hence it is widely studied there years.

- **Variable bandwidth:**

Wi-MAX can support variable bandwidth in physical layer. That is, it can adjust the transmission rate via changing the bandwidth. As a result, we need FFT with various sizes. For example, the bandwidth for Wi-MAX systems can be 1.25MHz, 5MHz, 10MHz, 20MHz corresponding to the 128-, 512-, 1024- and 2048-point FFT. This dynamic adjustment of and bandwidth real location possible FFT size enables user roaming in different network. Thus, the variable FFT is a key component for OFDM systems with various bandwidth.

- **MIMO techniques:**

The Wi-MAX solution uses multiple antenna techniques, such as beamforming, space-time coding, and spatial multiplexing to enhance system performance, including the overall system capacity and spectral efficiency. Therefore, an FFT architecture that can be efficiently used in MIMO-OFDM systems is also important.

The traditional FFT algorithm can be roughly classified into three types. The first type is fixed-radix FFT algorithm. The fixed-radix algorithm can be further to divided into the radix-2, radix-4/radix-2² and radix-8/radix-2³ algorithm [1] - [3]. The second type is split-radix FFT algorithm. The split-radix algorithm can be to divided into the radix-2/4 , radix-2/8 and radix-2/4/8 algorithm [4] - [5]. Third, we can used the method of common-factor algorithm (CFA) or prime-factor algorithm (PFA) to preform the mixed-radix algorithm [6].

The traditional pipeline FFT architecture can be roughly classified into two types. The first type is the single-path delay feedback (SDF). The single-path delay feedback can be divided into the R2SDF, R4SDF/R2²SDF and R8SDF/R2³SDF. The second type is the multi-path delay commutator (MDC). The multi-path delay commutator can be further divided into the R2MDC, R4MDC/R2²MDC and R8MDC/R2³MDC. Others architecture rather than the pipeline include Memory-based FFT [6], Cordic-base FFT [7] and systolic FFT [8] - [16]. The above architectures are not suitable to be used in Wi-MAX systems without modification since both MIMO and variable FFT are not needed in Wi-MAX systems.

Recently, some architectures for MIMO FFT or variable FFT were proposed. For example, the combination of MIMO and OFDM such as mixed-radix multi-path delay feedback (MRMDF) use proposed by Lin [19]. The authors use the characteristic of mixed-radix, multi-path and feedback plan for FFT and apply is in standard 802.11n. As for the variable FFT the authors in [8] used radix- 2^3 and multiplexors to implement the low power FFT architecture. Also, the others variable FFT architecture as refer to [9] - [11]. In general, we needed to increase the number of FFT processing units to our best knowledge in order to increase the throughput rate in MIMO-OFDM systems. Thus, the hardware complexity and power consumption increase as well. When MIMO-OFDM need variable FFT size, the hardware complexity increases dramatically. However, few researches has been conduct about the architecture of combining MIMO-OFDM and variable FFT which is used in Wi-MAX systems. A good processor not only need to support high throughput rate and variable FFT size, but also they need to be more efficient for hardware implementation. It is challenging to combine the advantages of MIMO-OFDM and variable FFT size.

1.2 Contributions and Features

The contributions of this research include:

- We proposed the method of reduce the memory size to 25% in Wi-MAX compared with that using R2SDF: Since the maximum supported FFT size is 2048-point, and the major part of the FFT Module is radix-2 algorithm, the memory occupancies much gate count. Thus, reducing memory leads to reduction of die area and power consumption.
- Multiplier sharing: In each radix-2 stage, the number of complex multipliers can be reduced from 4 to 2. Thus, the utilization rate of the multipliers increases from 50% to 100%.
- High radix to reduce the complexity: Because higher-radix algorithm can reduce number of multiplier and power consumption, we employ radix- 2^3

algorithm to implement the last two stages.



Chapter 2

Background

2.1 FFT for MIMO systems

The combination of the multiple-input multiple-output (MIMO) signal processing with orthogonal frequency-division multiplexing (OFDM) is considered as a promising solution for enhancing the data rates of the next generation wireless communication systems operated in frequency-selective fading environments. Because the technique of the MIMO can increase the data rate by extending an OFDM system, in the IEEE802.11n standard that uses a 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. However, the IEEE802.11n standard also increases the computational and hardware complexities, compared with the current SISO standards. It is a challenge to realize the physical layer of the MIMO-OFDM system with small hardware complexity and power consumption in very large scale integration (VLSI) implementation. Because the employing traditional approach to solve the simultaneous multiple data sequence, several FFT/IFFT processors are needed in the physical layer of a MIMO-OFDM system, we present the fast Fourier transform (FFT)/inverse FFT (IFFT) architecture was proposed by Lin for applications in a MIMO-OFDM systems [19]. The mixed-radix multi-path delay feedback (MRMDF) FFT architecture can provide higher throughput rate with small hardware cost, and can support 1-4 data sequence transmitted. The MRMDF architecture utilizes the advantages of the

following two FFT architectures: one is the single-path delay feedback and the other is the multi-path delay commutator [2].

2.1.1 Algorithm

A basic N -point discrete Fourier transform (DFT) is defined as

$$X(k) = \sum_{n=0}^{N-1} x(n)W_N^{kn}, \quad k = 0, 1, \dots, 127 \quad (2.1)$$

where $x(n)$ and $X(k)$ are complex number. The twiddle factor is

$$W_N^{nk} = e^{-j\frac{2\pi nk}{N}} = \cos \frac{2\pi nk}{N} - j \sin \frac{2\pi nk}{N}. \quad (2.2)$$

From the equation (2.1) we know that computational complexity is $O(N^2)$ through directly performing the required computation. By using the FFT algorithm, the computational complexity can be reduced to $O(N \log_r N)$, where r means the radix- r FFT algorithm. Although higher radix FFT algorithm has smaller process element (PE) iteration counts, generally require higher PE complexities in implementation. One well-known approach to solving this problem is the method introduced by He and Torkelson [3]. The solution for the problem is used the radix- 2^3 algorithm replace radix-8 algorithm, and then PE complexity can be reduce to radix-2 FFT algorithm. Because the 128-point FFT is not a power of 8, the mixed-radix algorithm is needed. The mixed-radix include the radix-2 and radix-8 FFT algorithm. we shall be derived in detail below.

First let

$$\begin{aligned} N &= 128 \\ n &= 64n_1 + n_2, & \begin{cases} n_1 = 0, 1 \\ n_2 = 0, 1, \dots, 63 \end{cases} \\ k &= k_1 + 2k_2, & \begin{cases} k_1 = 0, 1 \\ k_2 = 0, 1, \dots, 63 \end{cases} \end{aligned}$$

the equation (2.1) can be rewritten as

$$X(2k_2 + k_1) = \sum_{n_2=0}^{63} \sum_{n_1=0}^1 x(64n_1 + n_2)W_{128}^{(64n_1+n_2)(2k_2+k_1)}$$

$$= \sum_{n_2=0}^{63} \left\{ \underbrace{\sum_{n_1=0}^1 x(64n_1 + n_2)W_2^{n_1k_1}}_{\text{2-point}} \underbrace{W_{128}^{n_2k_2}}_{\text{twiddle factor}} \right\} W_{64}^{n_2k_2} \quad (2.3)$$

64-point

$$= \sum_{n_2=0}^{63} BU_2(k_1, n_2)W_{64}^{n_2k_2}. \quad (2.4)$$

Equation (2.3) can be regarded as a two-dimensional DFT, one is 64-point DFT and the other is 2-point DFT. Thus, we can complete the 128-point mixed-radix FFT operation. Furthermore, we can decompose 64-point DFT into 8-point DFT recursively 2 times and replace by radix-2³ FFT algorithm. Because the radix-2³ FFT algorithm is more efficient for VLSI design, we further reduce the PE complexity by using radix-2 process element. By a four-dimensional linear index map, we can rewrite n_2 and k_2 as

$$\begin{aligned} n_2 &= 32\alpha_1 + 16\alpha_2 + 8\alpha_3 + \alpha_4 & \alpha_1, \alpha_2, \alpha_3 &= 0, 1; \alpha_4 = 0, 1, \dots, 7 \\ k_2 &= \beta_1 + 2\beta_2 + 4\beta_3 + 8\beta_4 & \beta_1, \beta_2, \beta_3 &= 0, 1; \beta_4 = 0, 1, \dots, 7 \end{aligned} \quad (2.5)$$

By means of equation (2.5), equation (2.4) takes the form of

$$\begin{aligned} &X(2(\beta_1 + 2\beta_2 + 4\beta_3 + 8\beta_4) + k_1) \\ &= \sum_{\alpha_4=0}^7 \sum_{\alpha_3=0}^1 \sum_{\alpha_2=0}^1 \sum_{\alpha_1=0}^1 BU_2(k_1, 32\alpha_1 + 16\alpha_2 + 8\alpha_3 + \alpha_4) \\ &\quad \times W_{64}^{(32\alpha_1+16\alpha_2+8\alpha_3+\alpha_4)(\beta_1+2\beta_2+4\beta_3+8\beta_4)} \\ &= \sum_{\alpha_4=0}^7 BU_8(k_1, \beta_1, \beta_2, \beta_3, \alpha_4)W_8^{\alpha_4\beta_4}, \end{aligned} \quad (2.6)$$

where $BU_8(k_1, \beta_1, \beta_2, \beta_3, \alpha_4)$ is shown in equation (2.7).

$$\begin{aligned} &BU_8(k_1, \beta_1, \beta_2, \beta_3, \alpha_4) \\ &= \sum_{\alpha_3=0}^1 \sum_{\alpha_2=0}^1 \sum_{\alpha_1=0}^1 BU_2 \times W_4^{\alpha_2\beta_1} W_2^{\alpha_2\beta_2} W_8^{\alpha_3(\beta_1+2\beta_2)} W_2^{-\alpha_3\beta_3} W_{64}^{\alpha_4(\beta_1+2\beta_2+4\beta_3)} \end{aligned} \quad (2.7)$$

where $BU_2 = BU_2(k_1, 32\alpha_1 + 16\alpha_2 + 8\alpha_3 + \alpha_4) \times W_2^{\alpha_1\beta_1}$. The 128-point mixed-radix FFT algorithm signal flow graph is show in Fig. 2.1. The radix-2 FFT algorithm is used in the first stage, and the radix-8 FFT algorithm is applied in the second and third stage. The black point between the stage is twiddle factor.

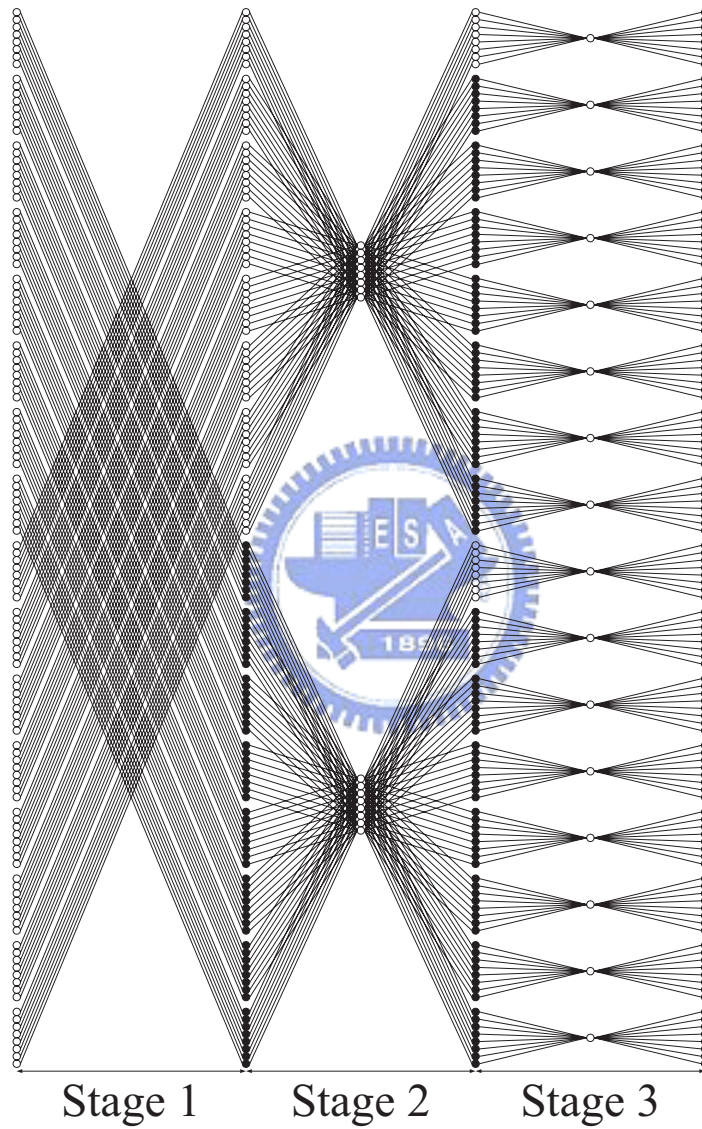


Figure 2.1: The SFG of 128-point mixed-radix FFT.

The IFFT of an N -point sequence $x(n)$, $k = 0, 1, \dots, N - 1$ is defined as

$$x(n) = \frac{1}{N} \sum_{k=0}^{N-1} X(k)W_N^{-nk}. \quad (2.8)$$

In order to implement the IFFT algorithm more efficiently, equation (2.8) can be rewritten as

$$x(n) = \frac{1}{N} \left\{ \sum_{k=0}^{N-1} X^*(k)W_N^{nk} \right\}^*. \quad (2.9)$$

According to equation (2.9), the IFFT can be performed by taking the complex conjugate of input data and then taking the complex conjugate of output data without change in any coefficient in the original FFT architecture. Thus, the hardware implementation can be more efficient. The block diagram of FFT/IFFT is show in Fig. 2.2. It was utilized multiplexer to change in the operation mode that operation of FFT or IFFT.

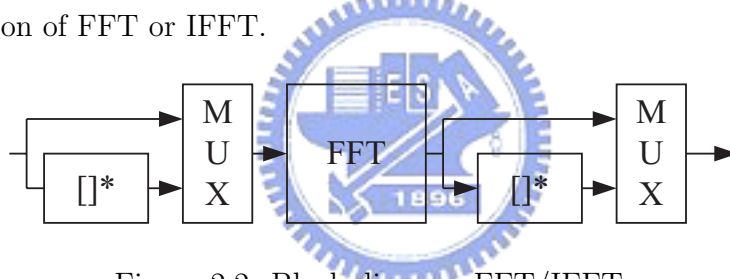


Figure 2.2: Block diagram FFT/IFFT.

2.1.2 Architecture

The FFT of MRMDF is provide 128/64-point FFT/IFFT operation, and then can support 1-4 data sequence transmitted for MIMO-OFDM system. From the Fig. 2.3 show that system architecture contains of Module 1 (data reorder), Module 2 (radix-2), Module 3 (radix- 2^3), Module 4 (radix- 2^3), conjugate block, division block and multiplexer. The characteristic of the MRMDF FFT architecture with size 128/64 are the following:

- The 128/64 point FFT with 1-4 simultaneous data sequence can be operated in this design.

- The FFT architecture can provide 1-4 throughput rates to achieve the requirements of IEEE802.11n standard.
- Small memory is needed by using the delay feed back scheme.
- High throughput rate can achieve by using the multi-path scheme.
- Higher radix FFT algorithm can be implemented to save power consumption.
- Modify complex multiplier can be implemented by constant multiplier to save power consumption.

Because the MRMDF architecture based on a radix-2 butterfly, the order of the output sequence is the bit reversal of the order of the input sequence, as shown in Fig. 2.4. The operation of the FFT and IFFT is controlled by the control signal, FFT/IFFT signal is show in Fig. 2.3. The details of this FFT architecture will be described in the next subsection.

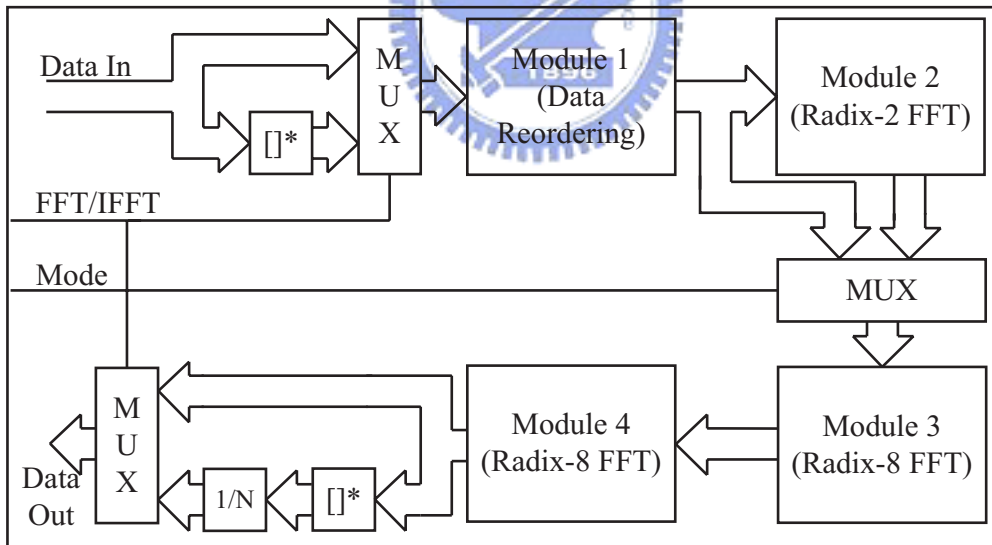


Figure 2.3: Block diagram of the 128/64-point FFT/IFFT processor.

a) Module 1: Module 1 contains several different size delay elements and switch block, as shown in Fig. 2.5. The function of Module 1 is to reorder the

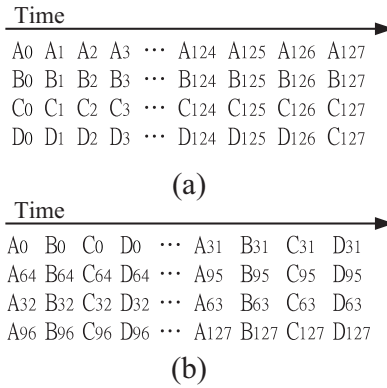


Figure 2.4: (a) Order of input; (b) Order of output.

input data sequence to achieve two goals. First, let Module 2, Module 3 and Module 4 implement the operation the FFT/IFFT more efficient. Second, avoid the data sequences in Module 3 to be multiplied by the same twiddle factor in each data path simultaneously. Thus, the modify complex multiplier can be used in Module 3 to reduce the hardware complexity by using the shift-and-add method [20]. The operation of the Module 1 is show in Fig. 2.6.

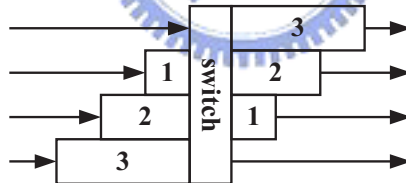


Figure 2.5: Block diagram of Module 1.

First, the four adjoining sequence with across difference delay unit, and then four adjoining data will be reordered by the appropriate operation of the switch. Finally, the adjoining data will simultaneous by difference delay unit. The re-ordered data will be separated into 32 groups or 16 groups for 128 or 64 point FFT calculation. If four data sequence will be transmitted, each group contains four data sequence, A, B, C and D. And in each group has the same sub-index, as shown in Fig. 2.6. As seen in Fig. 2.6, if there is only three data sequence will

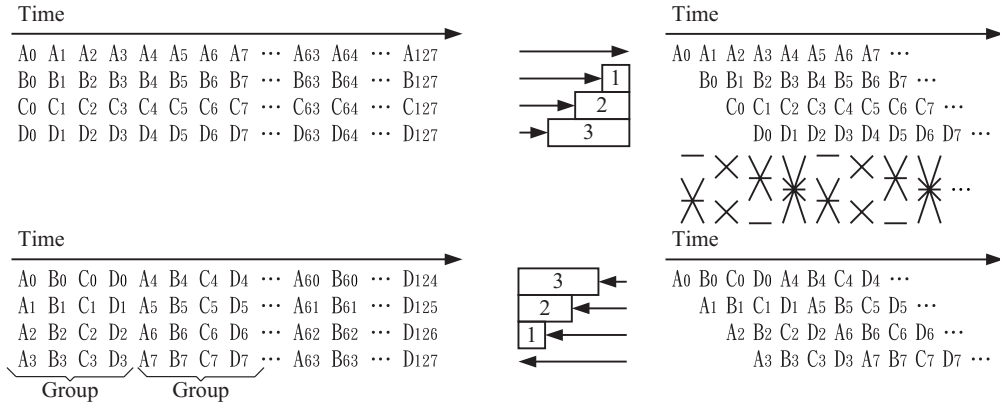


Figure 2.6: Relation between Module 1 input and Module 1 output.

be transmitted, the number of operation is three in the each group and so on. The operation of FFT/IFFT will more efficient through the reordering module.

b) Module 2: The Module 2 contains memory, two complex multipliers, four butterfly units, two ROM tables and some multiplexors as shown in Fig. 2.7. There are two kind of radix-2 butterfly unit in the FFT/IFFT processor. One of radix-2 butterflies is in Module 2, denoted by BF1. The function of BF1 is $X(i) = x(i) - y(i)$ and $Y(i) = x(i) + y(i)$. The dot-line rectangular in Fig. 2.7 is redrawn more detailed in Fig. 2.8(a). The control signal of the multiplexer is to determine one of the two operation modes of data change, as shown in Fig. 2.8(b).

When a 64-point FFT/IFFT is used in this architecture, the input data will skip Module 2 and directly go to Module 3. Four memory units are needed to save the result of butterfly operation. Only 1/8 cycle of cosine and sine values are needed to be stored in ROM table, and the other values can be reconstructed by these stored values. Thus, the ROM table size can be to reduce. In general, four complex multipliers are needed to implement FFT/IFFT with four-parallel data sequences by traditional radix-2 SDF architecture, but we only needed two multipliers to implement four-parallel data sequences in this module. We can first multiply the twiddle factors of two data sequences, and then multiply the other two data sequences. We call this method as time sharing. The time sharing is explained below.

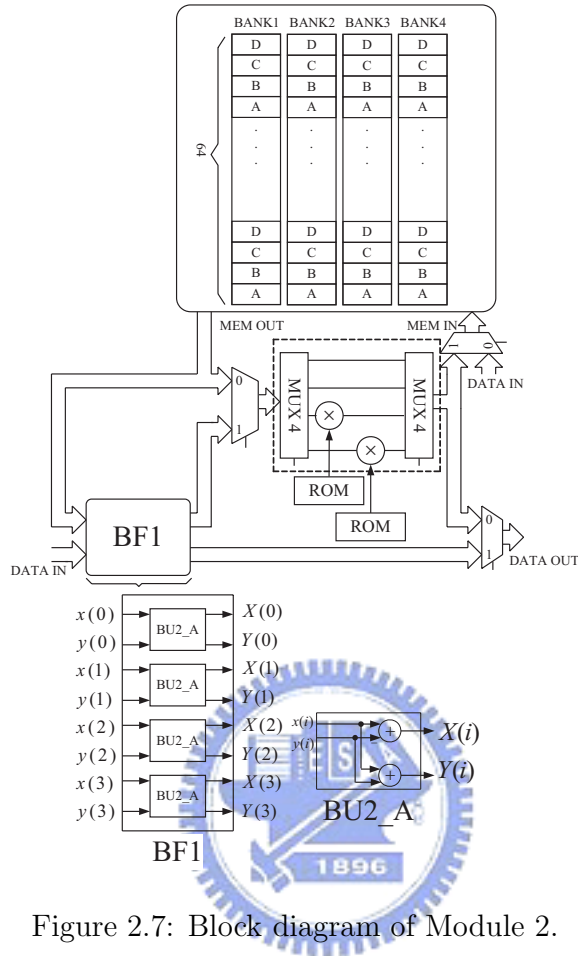


Figure 2.7: Block diagram of Module 2.

Time sharing: Consider the traditional R2SDF FFT 128-point architecture with four data sequences at the first stages, as shown in Fig. 2.9. When data sequence form $x(0)$ to $x(63)$ arrive, they are stored in memory (at clock cycle 0 to 63), as shown in Fig. 2.10. When data from $x(64)$ to $x(127)$ arrive, radix-2 butterfly starts to work (at clock cycle 64 to 127). Then, added results are fed to next stage, and the subtract result are sent back and saved in memory, as shown in Fig. 2.11, where $\hat{A}_0 = A_0 - A_{63}$, $\hat{A}_1 = A_1 - A_{64}$, $\hat{A}_2 = A_2 - A_{65}, \dots, \hat{A}_{63} = A_{63} - A_{127}$. Finally the data are read from memory, multiplied appropriate twiddle factors and then passed to next stage (at clock cycle 128 to 191). We know that there is no need to operate addition and subtraction since the operation of adder or subtract was completed before clock cycle 128. Consequently, we can utilize

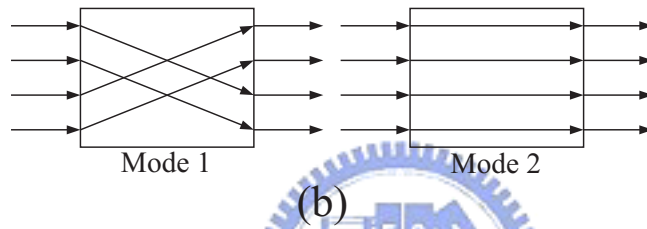
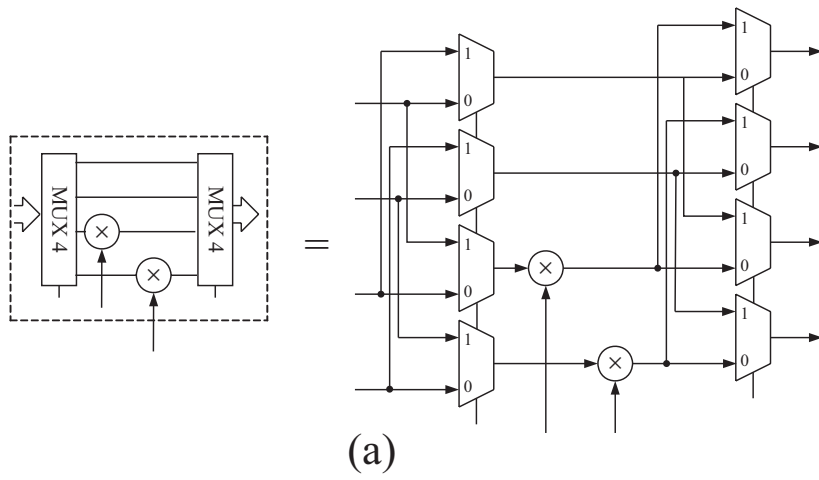


Figure 2.8: (a) Architecture of multiplier; (b) Operation mode of multiplier.

the two or periods of clock cycle 64~127 and clock cycle 128~191 to multiply twiddle factors for four data sequences. Thus, we can use two multipliers to complete the multiplication of twiddle factors for four data sequence.

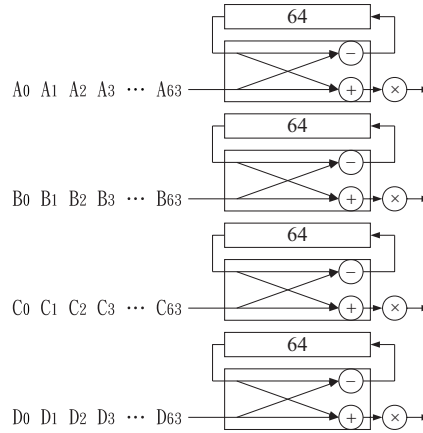


Figure 2.9: Architecture of four antenna R2SDF FFT 128-point at stage one.

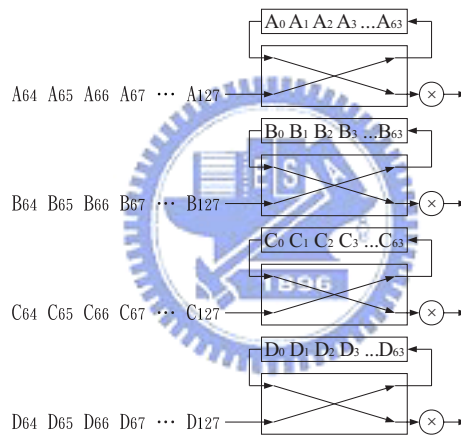


Figure 2.10: Save data in memory.

When a 128-point FFT/IFFT is used in this architecture, at first the data sequences from $x(0)$ to $x(63)$ arrive, they are stored in memory and operation mode of multiplexer are mode 2 (at clock cycle 0 to 63), as shown in Fig. 2.12. When data sequences from $x(64)$ to $x(127)$ arrive, radix-2 PE starts to work (at clock cycle 64 to 127). Then, the added results are fed to next stage, and the subtracted results are multiply appropriate twiddle factors with two sequences and save in memory. Moreover, the operation mode of multiplexer are mode 1. Finally, these data stored in the memory are read, multiplied appropriate twiddle

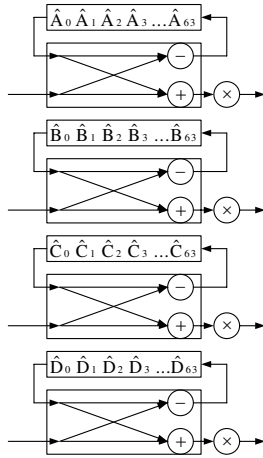


Figure 2.11: Operation of radix-2.

factors with other two sequences and then passed to next stage (at clock cycle 128 to 191). In traditional single path delay feedback architecture the utilization rate of the complex multiplier is only 50%. By the timing sharing, only two complex multipliers are needed and the utilization of the complex multipliers can achieve 100% in this scheme.

	BANK1	BANK2	BANK3	BANK4
64	D ₆₀	D ₆₁	D ₆₂	D ₆₃
	C ₆₀	C ₆₁	C ₆₂	C ₆₃
	B ₆₀	B ₆₁	B ₆₂	B ₆₃
	A ₆₀	A ₆₁	A ₆₂	A ₆₃

	D ₀	D ₁	D ₂	D ₃
	C ₀	C ₁	C ₂	C ₃
	B ₀	B ₁	B ₂	B ₃
	A ₀	A ₁	A ₂	A ₃

Figure 2.12: Module 2 memory bank.

c) Module 3: The Module 3 contains three radix-2 PEs and one modified complex multiplier, as shown in Fig. 2.13. The BU2_B include control signal,

which controls the operation modes of radix-2, as show in Fig. 2.14.

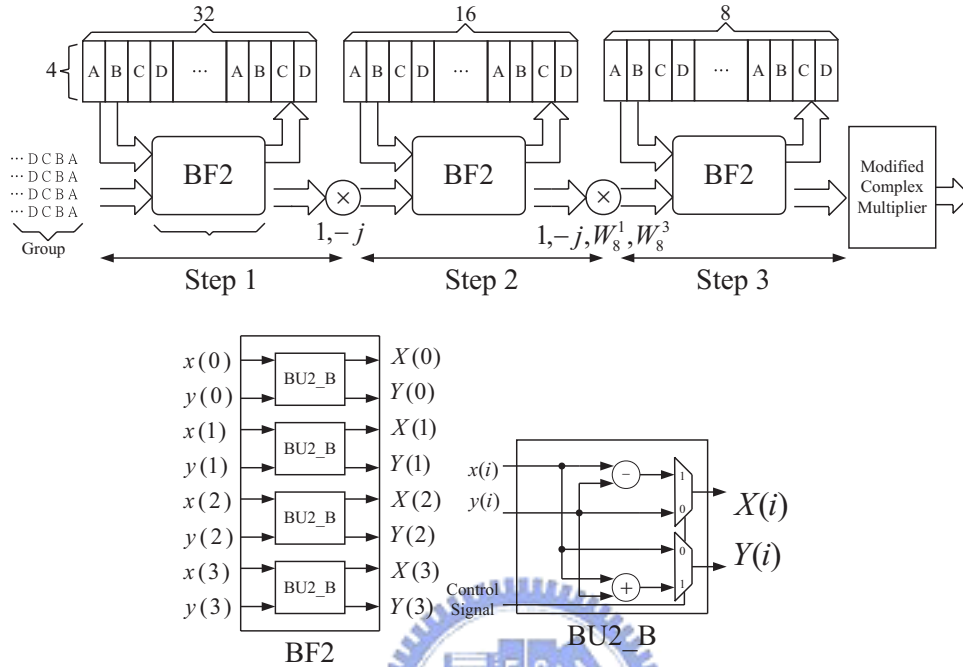


Figure 2.13: Block diagram of Module 3.

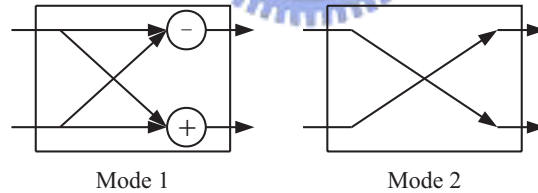


Figure 2.14: Two operation mode.

Module 3 is the radix- 2^3 FFT algorithm proposed by He and Torkelson [3], whose SFG is shown in the second stage of Fig. 2.1. The function of Module 3 is to preform the second stage butterfly of Fig. 2.1, where each stage is multiplied by the twiddle factor $1, -j, W_8^3$ and W_8^1 . From Fig. 2.9, it is inefficient to have four complex multipliers to multiply different twiddle factors. Here we can utilize an approach proposed by Maharatna [20] to reduce the complexity of the complex

multipliers. The twiddle factor in Module 3 is $W_{64}^p = e^{-\frac{j2\pi p}{64}} = X_p + jY_p = \cos\left(\frac{2\pi p}{64}\right) - j \times \sin\left(\frac{2\pi p}{64}\right)$, where p is from 0 to 49, as shown in Fig. 2.15. Due to the symmetric or anti-symmetric property of sine and cosine function, only nine sets of twiddle factors is needed to construct. That is, the X_p and Y_p with $p=0\sim 8$ in region A are needed, because the twiddle factors in the other seven regions can be obtained by changing their sign as shown in Table 2.1. Thus, these complex values can be realized more efficiently by using shift-and-add method [20]. The gate count of this method can save about 38% compared to the approach four complex multipliers. In addition, using the performance of this method is equivalent to that using four complex multipliers.

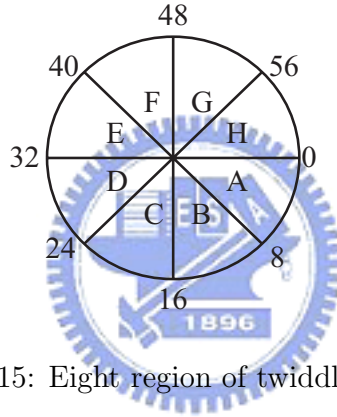


Figure 2.15: Eight region of twiddle factor.

Region	Real	Image
A	X_p	Y_p
B	$-Y_p$	$-X_p$
C	Y_p	$-X_p$
D	$-X_p$	Y_p
E	$-X_p$	$-Y_p$
F	Y_p	X_p
G	$-Y_p$	X_p
H	X_p	$-Y_p$

Table 2.1: Mapping table of twiddle factors in different regions.

d) Module 4: The block diagram of the Module 4 is show in Fig. 2.16. The function of Module 4 is the radix- 2^3 FFT algorithm, which is directly mapped to

the third stage of Fig. 2.1. Although Module 3 and Module 4 are both radix- 2^3 FFT algorithm, the architecture of Module 4 is different from that of Module 3, due to the scheme makes the circuit utilization more efficiently, and reduce the processing unit. Referral Fig. 2.16, due to the four data sequences are proposed simultaneously in one clock cycle, thus the data are ready for step 2 and step 3. Hence, the data sequences do not need to be stored in memory at step 2 and step 3.

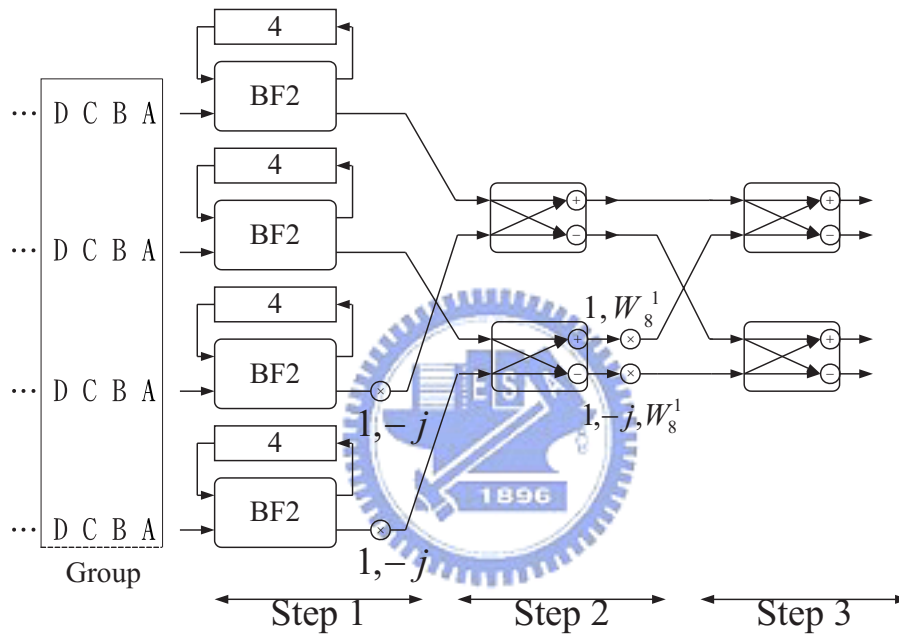


Figure 2.16: Block diagram of Module 4.

2.2 Variable FFT

The standards of DAB, DVB-T, VDSL and Wi-MAX need various FFT sizes, as shown in Table 2.2. Hence, the design of a variable FFT for different purposes become more important. In this section we present a variable FFT that can support 64, 32, 16 and 8-point operation. It is very easy to modify our design by adding 128-point or others 2^k -point FFT operation size to create any required length of FFT, where k is integer. Hence, this modification of circuit is convenient

and simple to be used for DAB, DVB-T, Wi-MAX and VDSL systems. For the others variable FFT, please refer to the [8].

Communication system	FFT size
Wi-MAX	128,512,1024,2048
VDSL [15]	8192,4096,2048,1024,512
DAB [12]	2048,1024,512,256
DVB-T [13]	8192,2048

Table 2.2: FFT size in several OFDM systems.

2.2.1 Pipeline FFT processor architecture

The traditional radix-2 pipeline FFT architectures can be roughly classified multi path delay commutator and single path delay feedback [3]. A radix-2 multi path delay commutator (R2MDC) architecture with $N=8$ is shown in Fig. 2.17. The data sequence is divided into two data paths by commutator, and then properly scheduled for two data paths. The processor element (PE) is implemented by radix-2 algorithm. The numbers of multipliers, PE unit and delay elements are with order $(\log_2 N - 2)$, $\log_2 N$ and $(\frac{3N}{2}) - 2$ respectively. A radix-2 single path delay feedback (R2SDF) architecture with $N=8$ is shown in Fig. 2.18. The utilization of delay elements in R2SDF is more efficient than R2MDC by sharing the memory. The numbers of multipliers, PE units and delay elements for R2MDC are $(\log_2 N - 1)$, $N - 1$ and $\log_2 N$. The SDF FFT and MDC FFT are decied as follows.

- a) SDF FFT: Because the SDF FFT uses feedback to reuse memory, the SDF FFT can reduce the memory usage. Its drawback is that the through put rate is low.
- b) MDC FFT: Because the MDC FFT uses multi path to increase data path, the MDC FFT can increase the through put rate. Its drawback is that the memory size is so large.

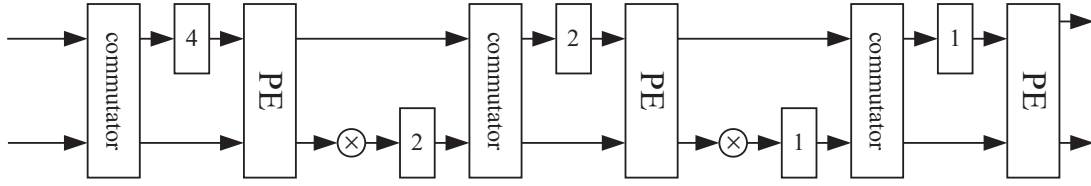


Figure 2.17: Architecture of R2MDC.

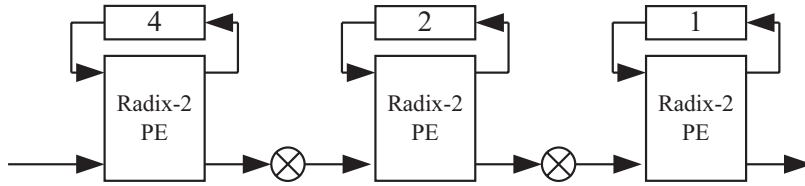


Figure 2.18: Architecture of R2SDF.

2.2.2 Variable FFT processor architecture

Let us see a variable FFT that can achieve 64-point, 32-point, 16-point and 8-point operation as shown in Fig. 2.19. The radix-2/ 2^3 64-point mixed-radix SFG is shown in Fig. 2.20. Where stage 1 to stage 3 are radix-2 algorithm and stage 4 is radix- 2^3 algorithm. It uses the above architectures and multiplexors to perform the variable FFT. This FFT can deal with 4 types of transformation. Moreover, it is easy to be modified to any transformation length. For the 64-point FFT, all stages are active. For the 32-point FFT, the input data will skip the first stage and go to the second stage directly. For the 16-point FFT, the input data will skip the first stage and second stage and go to the third stage. Multiplexors are used to switch to different FFT size operation.

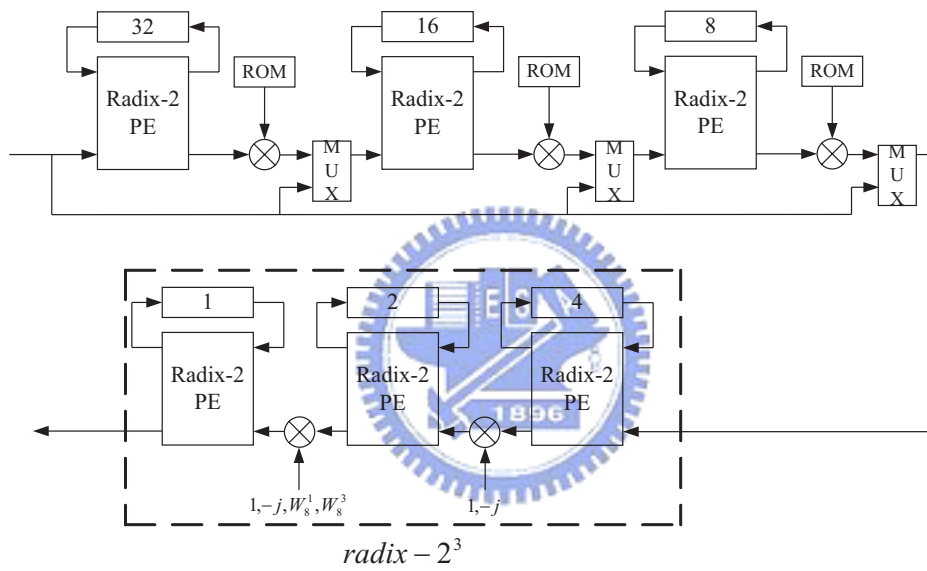


Figure 2.19: Block diagrams of a variable FFT processor.

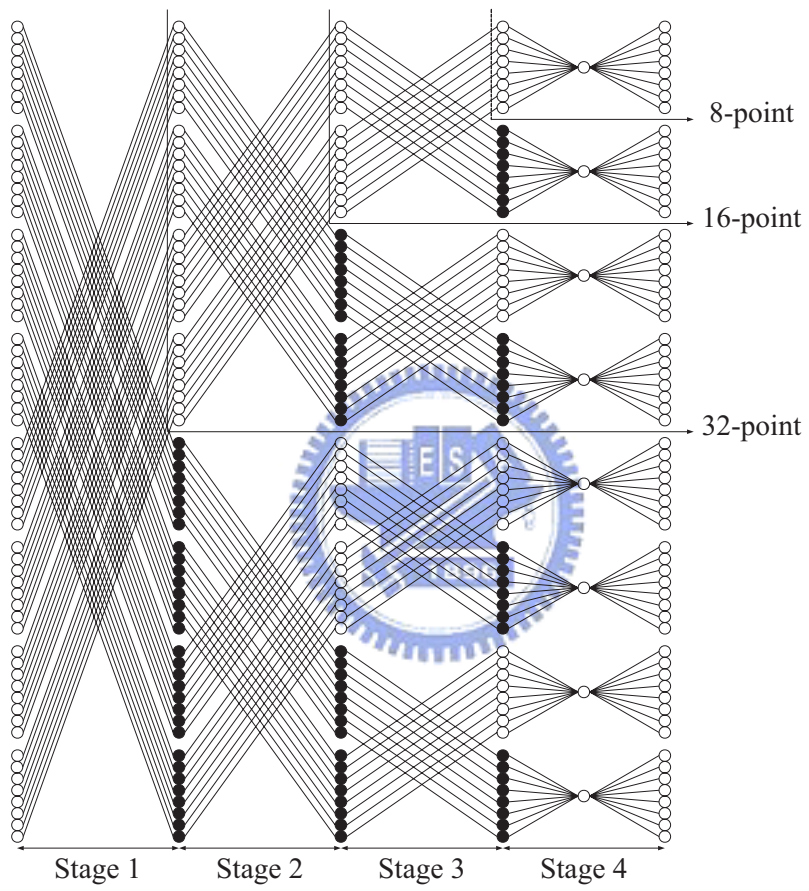


Figure 2.20: The SFG of 64-point mixed-radix FFT.

Chapter 3

The proposed variable FFT for MIMO systems

In this chapter, we detail the FFT in IEEE 802.16e which is for MIMO-OFDM application. The variable FFT can support multiple antenna and 2048/1024/512/128-point FFT size. In Sec. 3.1 we shall derive the FFT algorithm, and to show the SFG. In Sec. 3.2 we shall detail the FFT architecture, it is introduced in each Module. In Sec. 3.3 we compare the hardware requirement with several classes FFT and proposed approach in case 2048-point FFT. Finally, we show the SQNR simulation in Sec. 3.4, and to explain about how to determine bit width.

3.1 Algorithm

From (2.1), the N -point DFT operation can be decomposed to $N_1 \times N_2 \times \dots \times N_k$ point DFT operation. We use the radix-2³ as many as possible reduce the multipliers. The mathematical representation is shown in equation (3.1).

$$N = 2048 = \underbrace{8 \times 8 \times 2}_{128\text{-point}} \times 2 \times 2 \times 2 \times 2. \quad (3.1)$$
$$\underbrace{\hspace{10em}}_{512\text{-point}}$$
$$\underbrace{\hspace{10em}}_{1024\text{-point}}$$
$$\underbrace{\hspace{10em}}_{2048\text{-point}}$$

From (3.1), $N = 2048 = N_1 \times N_2 = 2 \times 1024$. Define the following indices

$$n = N_2 n_1 + n_2 = 1024 n_1 + n_2, \quad \begin{cases} n_1 = 0, 1 \\ n_2 = 0, 1, \dots, 1023 \end{cases}$$

and

$$k = k_1 + N_1 k_2 = k_1 + 2k_2, \quad \begin{cases} k_1 = 0, 1 \\ k_2 = 0, 1, \dots, 1023 \end{cases}$$

(2.1) can be rewritten as

$$\begin{aligned} & X(k_1 + 2k_2) \\ &= \sum_{n_2=0}^{1023} \sum_{n_1=0}^1 x(1024n_1 + n_2) W_{2048}^{(1024n_1+n_2)(k_1+2k_2)} \\ &= \sum_{n_2=0}^{1023} \left\{ \underbrace{\sum_{n_1=0}^1 x(1024n_1 + n_2) W_2^{n_1 k_1}}_{\text{2-point}} \underbrace{W_{2048}^{n_2 k_1}}_{\text{twiddle factor}} \right\} W_{1024}^{n_2 k_2} \end{aligned} \quad (3.2)$$

$$= \sum_{n_2=0}^{1024} B_2^{(1)}(n_2) W_{1024}^{n_2 k_2}, \quad (3.3)$$

where $B_r^{(k)}$ denotes the radix- r algorithm at stage k . Now $N_2 = 1024 = N_2 \times N_3 = 2 \times 512$. Define the indices n_2 and k_2 as

$$n_2 = N_3 n_2 + n_3 = 512 n_2 + n_3, \quad \begin{cases} n_2 = 0, 1 \\ n_3 = 0, 1, \dots, 511 \end{cases}$$

and

$$k_2 = k_2 + N_2 k_3 = k_2 + 2k_3, \quad \begin{cases} k_2 = 0, 1 \\ k_3 = 0, 1, \dots, 511 \end{cases}$$

We have:

$$\begin{aligned} & X(k_1 + 2k_2 + 4k_3) \\ &= \sum_{n_3=0}^{511} \sum_{n_2=0}^1 B_2^{(1)}(512n_2 + n_3) W_{1024}^{(512n_2+n_3)(k_2+2k_3)} \\ &= \sum_{n_3=0}^{511} \left\{ \underbrace{\sum_{n_2=0}^1 B_2^{(1)}(512n_2 + n_3) W_2^{n_2 k_2}}_{\text{4-point}} \underbrace{W_{1024}^{n_3 k_2}}_{\text{twiddle factor}} \right\} W_{512}^{n_3 k_3} \end{aligned}$$

512-point

$$= \sum_{n_3=0}^{511} B_2^{(2)}(n_3) W_{512}^{n_3 k_3}. \quad (3.4)$$

In this way, we can obtain the result is given by

$$\begin{aligned} & X(k_1 + 2k_2 + 4k_3 + 8k_4 + 16k_5 + 32k_6) \\ &= \sum_{n_6=0}^{63} B_2^{(5)}(n_6) W_{64}^{n_6 k_6}, \end{aligned} \quad (3.5)$$

where $k_6 = 0, 1, \dots, 63$. By decomposing the 64-point DFT into the 8-point DFT, we can achieve the 2048-point mixed-radix FFT algorithm.

$$\begin{aligned} & X(k_1 + 2k_2 + 4k_3 + 8k_4 + 16k_5 + 32k_6 + 256k_7) \\ &= \sum_{n_7=0}^7 \left\{ \sum_{n_6=0}^7 B_2^{(5)}(8n_6 + n_7) W_8^{n_6 k_6} W_{64}^{n_7 k_6} \right\} W_8^{n_7 k_7}. \end{aligned} \quad (3.6)$$

Because the radix-8 butterfly unit is inefficient in the use of adders and multipliers. we use the radix- 2^3 FFT algorithm [3] to replace the radix-8 FFT algorithm. In this case, we can further reduce the complexity of the butterfly by using the radix-2 butterfly three times. The SFG of the 2048-point mixed-radix FFT algorithm, is as shown in Fig. 3.1 and Fig. 3.2.

Let $n_6 = 4\alpha_1 + 2\alpha_2 + \alpha_3$ and $k_6 = \beta_1 + 2\beta_2 + 4\beta_3$, we can obtain the form with radix- 2^3 in equation (3.9).

$$\begin{aligned} & X(k_1 + 2k_2 + 4k_3 + 8k_4 + 16k_5 + 32(\beta_1 + 2\beta_2 + 4\beta_3) + 256k_7) \\ &= \sum_{n_7=0}^7 B_8^{(6)}(n_7, k_7) W_8^{n_7 k_7}, \end{aligned} \quad (3.7)$$

where

$$\begin{aligned} B_8^{(6)}(n_7, k_7) &= \sum_{\alpha_3=0}^1 \sum_{\alpha_2=0}^1 \sum_{\alpha_1=0}^1 B_2^{(5)}(8(4\alpha_1 + 2\alpha_2 + \alpha_3) + n_7) W_2^{\alpha_1 \beta_1} \\ & W_4^{\alpha_2 \beta_1} W_2^{\alpha_2 \beta_2} W_8^{\alpha_3(\beta_1 + 2\beta_2)} W_2^{\alpha_3 \beta_3} W_{64}^{n_7(\beta_1 + 2\beta_2 + 4\beta_3)}. \end{aligned} \quad (3.8)$$

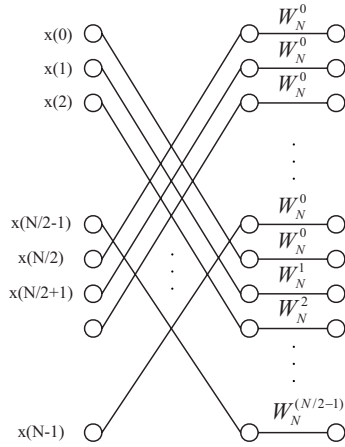


Figure 3.1: The SFG of stage 1 to stage 5 (radix-2).

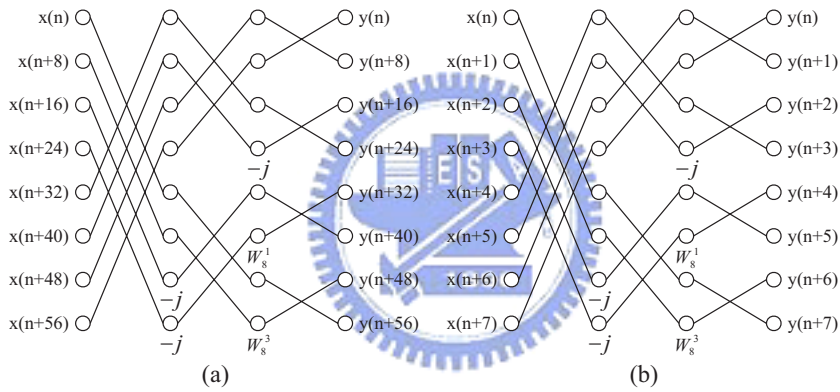


Figure 3.2: The SFG of stage 6 to stage 7 (radix-2³).

3.2 Architecture

The variable FFT for MIMO-OFDM system is provide 2048/1024/512/256-point FFT/IFFT operations and can support number T of data streams from $T = 1$ to $T = 4$. From Fig. 3.3, the system is contains of Module 1 (data reordering), Module 2 to Module 6 (radix-2), Module 7 (radix-2³) and Module 8 (radix-2³), conjugate blocks, some divide blocks and multiplexors. Because the FFT is based on a radix-2 butterfly, the order of the output sequences is bit reversal of input, as shown in Fig. 3.4.

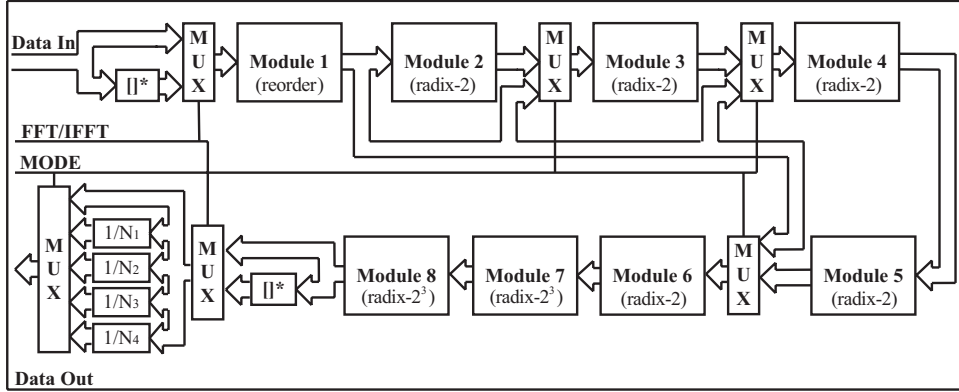


Figure 3.3: Block diagram of the variable FFT processor.

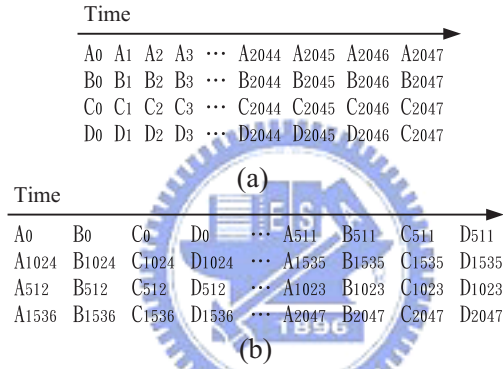


Figure 3.4: The input and output relationship of FFT.

3.2.1 Module 1 (data reordering)

The Module 1 is implemented by registers with size 4×4 , and we use clock gating to save the power consumption. The time schedule of Module 1 is shown in Fig. 3.5. The input and output relationship of Module 1 is shown in Fig. 3.6. In Module 1, the input data sequences are re-permuted so that it leads to efficient operation for radix-2 module and Module 8 implementation as we will mention later. From Fig. 3.6, where $n = 128, 512, 1024$ and 2048 , $N = 32, 128, 256$ and 512 for various FFT sizes. For example, when the number of transmitted sequences is four and the FFT size is 128, each group is contains four data sequences and there are 32 groups.

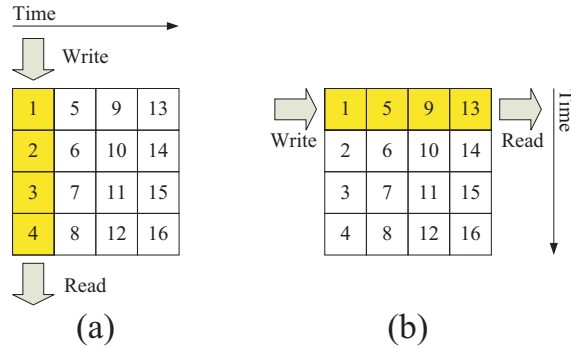


Figure 3.5: (a) Read and write with column; (b) Read and write with row.

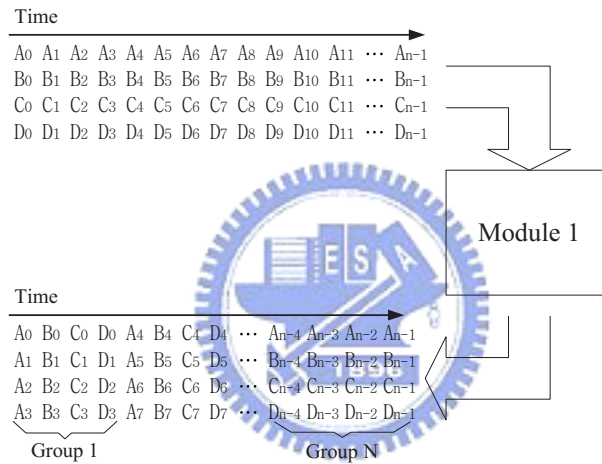


Figure 3.6: Relation between Module 1 input and Module 1 output.

3.2.2 Module 2 to 6 (radix-2 FFT algorithm)

Module 2 contains four butterfly units, two multipliers, four memory banks, ROM table, rotation factor and some multiplexors, as shown in Fig. 3.7. Module 2 is similar to previous architecture as mentioned in Sec. 2.1.2. The difference is that Module 2 includes the rotation factor W_N^1 to reduce the memory size. The architecture of Module 2 to Module 6 are actually the same except their memory sizes are different. The memory sizes from Module 2 to Module 6 are 1024, 512, 256, 128 and 64, respectively. Module 2 to Module 6 realize the radix-2 FFT algorithm and the SFG is shown in Fig. 3.1. From Fig. 3.1, the value N for

Module 2 to Module 6 are 2048, 1024, 512, 256 and 128, respectively. In addition to the advantage of time sharing as mentioned in Sec. 2.1.2, we also have the advantage of memory sharing in Module 2. The memory sharing is explained below.

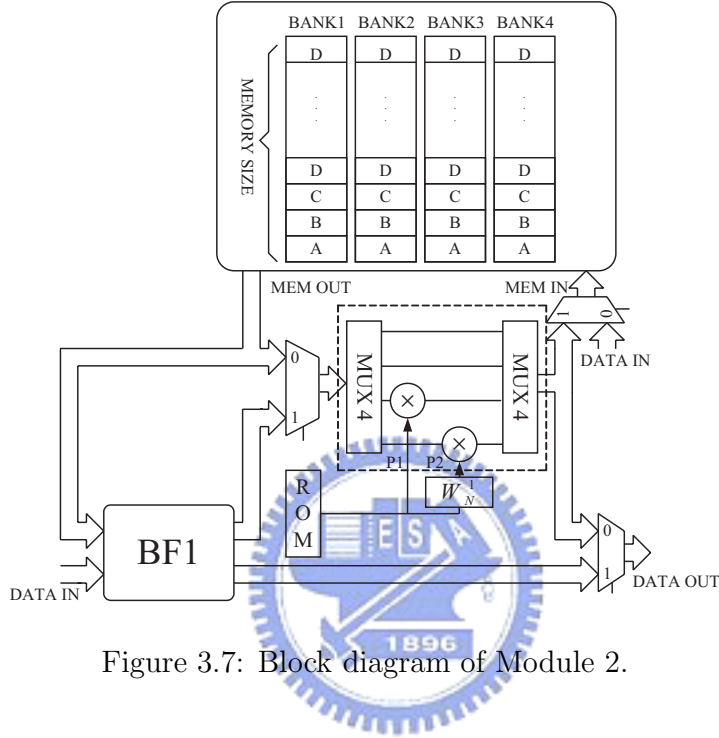


Figure 3.7: Block diagram of Module 2.

Memory sharing: The twiddle factors for $T = 4$ at the first stage butterfly is shown in Fig. 3.9, where A, B, C and D represent the four data sequences. This figure also shows the used twiddle factors in each time instance. Originally, we need to store $\frac{N}{2}$ twiddle factors for first stage butterfly. Also, since we need ROM tables to store the twiddle factors for the radix-2 butterfly in modules 3, 4, 5 and 6. Thus, the total twiddle factors are $\frac{N}{2} + \frac{N}{4} + \frac{N}{8} + \frac{N}{16} + \frac{N}{32} = \frac{62N}{64}$, where $N = 2048$ here. Hence, when N is large, the memory size also become large. Here, we will propose a memory sharing method that can reduce the memory size from $\frac{62N}{64}$ to $\frac{31N}{64}$ and we can sharing the memory from Module 2 to Module 6 as shown in Fig. 3.8, thus the memory size can be reduced to $\frac{N}{4}$. This memory sharing method is described as follows:

When a 2048-point FFT/IFFT is used in this architecture, at first the data

sequences from $x(0)$ to $x(1023)$ arrive, they are stored in memory of Module 2 (at clock cycle 0 to 1023). When the data sequences from $x(1024)$ to $x(2047)$ arrive, radix-2 PEs start to work (at clock cycle 1024 to 2047). Then, the added results are fed to the next stage, and two of the four subtracted data sequences are multiplied by appropriate twiddle factors and all of the four data sequences are saved in memory. Finally, all the four data sequences stored in the memory are read, and two of them (those who have not multiplied the twiddle factors) are multiplied by appropriate twiddle factors and then pass to the next stage (at clock cycle 2048 to 3071). Hence, we need a ROM table of size $\frac{N}{2}$ to store the twiddle factor for the first stage butterfly.

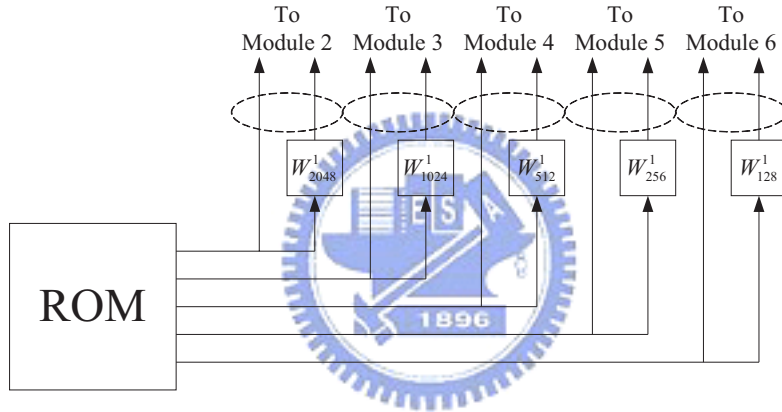


Figure 3.8: Memory sharing from Module 2 to Module 6.

However, from Fig. 3.9, the twiddle factors in $P1$ and $P2$ has a ratio of $W_{\frac{N}{2}}^1$, where $P1$ and $P2$ are also shown in Fig. 3.7. Since the rotation factor is a shift-and-add, from Fig. 3.10 we know the rotation factor does not increase the critical path. The critical path is shown with red color in Fig. 3.12, which is 13ns.

	Time →												
P1	W_N^0	W_N^0	W_N^0	W_N^0	W_N^4	W_N^4	W_N^4	W_N^4	...	$W_N^{(\frac{N}{2}-4)}$	$W_N^{(\frac{N}{2}-4)}$	$W_N^{(\frac{N}{2}-4)}$	$W_N^{(\frac{N}{2}-4)}$
P2	W_N^1	W_N^1	W_N^1	W_N^1	W_N^5	W_N^5	W_N^5	W_N^5	...	$W_N^{(\frac{N}{2}-3)}$	$W_N^{(\frac{N}{2}-3)}$	$W_N^{(\frac{N}{2}-3)}$	$W_N^{(\frac{N}{2}-3)}$
	A	B	C	D	A	B	C	D		A	B	C	D

(a)

	Time →												
P1	W_N^2	W_N^2	W_N^2	W_N^2	W_N^6	W_N^6	W_N^6	W_N^6	...	$W_N^{(\frac{N}{2}-2)}$	$W_N^{(\frac{N}{2}-2)}$	$W_N^{(\frac{N}{2}-2)}$	$W_N^{(\frac{N}{2}-2)}$
P2	W_N^3	W_N^3	W_N^3	W_N^3	W_N^7	W_N^7	W_N^7	W_N^7	...	$W_N^{(\frac{N}{2}-1)}$	$W_N^{(\frac{N}{2}-1)}$	$W_N^{(\frac{N}{2}-1)}$	$W_N^{(\frac{N}{2}-1)}$
	A	B	C	D	A	B	C	D		A	B	C	D

(b)

Figure 3.9: (a) ROM table at clock cycle 1024 to 2047; (b) ROM table at clock cycle 2048 to 3071.



	Time →												
P1	W_N^0	W_N^0	W_N^0	W_N^0	W_N^4	W_N^4	W_N^4	W_N^4	...	$W_N^{(\frac{N}{2}-4)}$	$W_N^{(\frac{N}{2}-4)}$	$W_N^{(\frac{N}{2}-4)}$	$W_N^{(\frac{N}{2}-4)}$
	A	B	C	D	A	B	C	D		A	B	C	D

(a)

	Time →												
P1	W_N^2	W_N^2	W_N^2	W_N^2	W_N^6	W_N^6	W_N^6	W_N^6	...	$W_N^{(\frac{N}{2}-2)}$	$W_N^{(\frac{N}{2}-2)}$	$W_N^{(\frac{N}{2}-2)}$	$W_N^{(\frac{N}{2}-2)}$
	A	B	C	D	A	B	C	D		A	B	C	D

(b)

Figure 3.10: (a) ROM table at clock cycle 1024 to 2047; (b) ROM table at clock cycle 2048 to 3701.

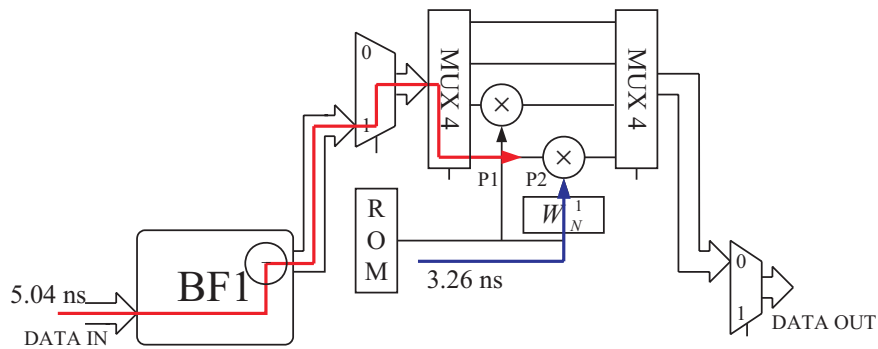


Figure 3.11: Analysis for critical path.

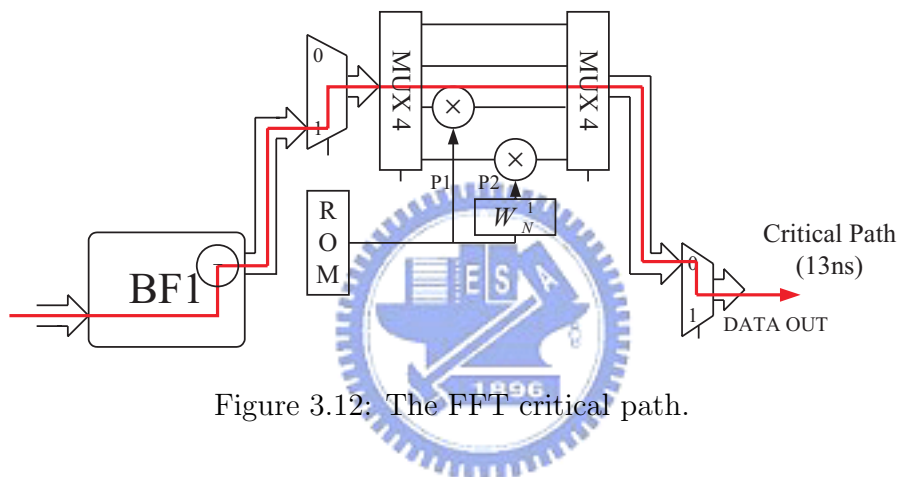


Figure 3.12: The FFT critical path.

3.2.3 Module 7 (radix-2³ FFT algorithm)

Module 7 is the same as Module 3 in Sec. 2.1.2. The function of Module 7 is to perform the 6th stage butterfly of Fig. 3.2.

3.2.4 Module 8 (radix-2³ FFT algorithm)

Module 8 is the same as Module 4 in Sec. 2.1.2. The function of Module 8 is to perform the 7th stage butterfly of Fig. 3.2.

3.3 Complexity comparison

Let $T = 4$, let us compare the hardware complexity of the proposed architecture and the others FFT architectures which is shown in Table 3.1. For four data sequences the proposed approach can save 69% complex multiplier and 75% ROM tables compare with R2SDF. Note that for other architectures in Table 3.1, they may not be able to support the variable FFT sizes required by Wi-MAX standards. For instance, although using the R2³SDF can save 70% complex multiplier, its FFT size is limited to power of eight.

Architecture	Four data sequence				
	Complex multiplier	Complex adder	ROM table	Memory size	Throughput rate
Proposed (Variable 2048-pt)	$10+4 * 0.61$ (31.1%)	80 (90.9%)	512 (25%)	8188 (100%)	4R
R2SDF (2048-pt)	$10 * 4=40$ (100%)	88 (100%)	2046 (100%)	8188 (100%)	4R
R2 ² SDF (2048-pt)	$4 * 4=16$ (40%)	88 (100%)	2040 (99.7%)	8188 (100%)	4R
R2 ³ SDF (2048-pt)	$3 * 4=12$ (30%)	88 (100%)	2044 (99.9%)	8188 (100%)	4R

Table 3.1: Comparison of hardware requirement.

3.4 Simulation

Determining appropriate bit width in the FFT processor is important. Since the bit width affects the hardware cost directly. Bit width can be determine by fixed-point simulation. We use the SQNR (Signal to Quantization Noise Ratio) system model to determine the FFT bit width. The system model of SQNR is shown below.

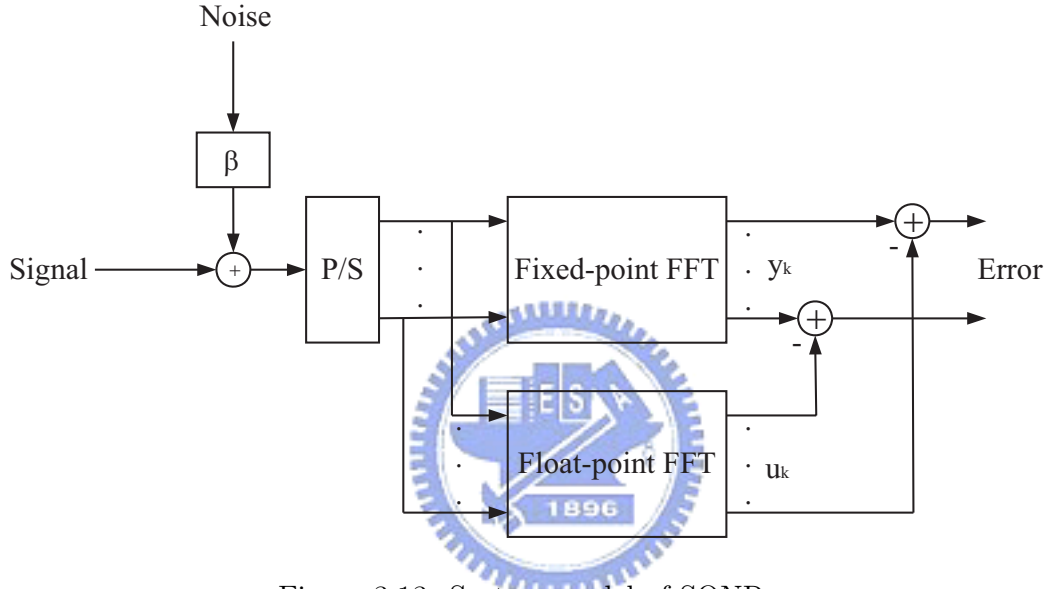


Figure 3.13: System model of SQNR.

The SQNR is defined as

$$SQNR = \frac{1}{N} \sum_{k=0}^{N-1} \frac{\sigma_x^2}{|y_k - u_k| + \beta^2 \sigma_e^2}. \quad (3.9)$$

Consider the SNR (Signal to Noise Ratio) is one ($\frac{\sigma_x^2}{\sigma_e^2} = 1$), we can rewrite the equation as

$$SQNR = \frac{1}{N} \sum_{k=0}^{N-1} \frac{1}{|y_k - u_k| + \beta^2}, \quad (3.10)$$

where $\beta = 10^{\frac{-SNR}{20}}$. The relationship between SQNR (with final bit-width) and SNR is shown in Fig. 3.12.

When SQNR is large, it means that quantization error is small. From the figure, the final bit-width can support SQNR up to 35 dB.

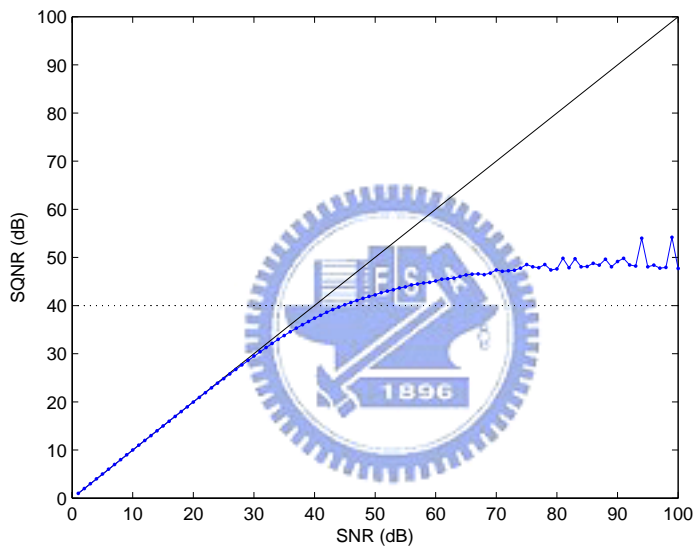


Figure 3.14: SQNR v.s. SNR.

Chapter 4

Chip implementation and verification

In this chapter we state how to build the MATLAB platform of the MIMO variable FFT and how to verify the design. The MATLAB platform can be used to verify the function of the RTL platform. The design flow is illustrate in Fig. 4.1. The design flow is suggested by CIC (Chip Implementation Center) for cell-based design. This design flow includes several important steps:

- Analysis and verification for architecture (using Matlab tool to simulation).
- Architecture design (using NCverilog, Modelsim and debussy tool to simulation).
- Design for testability (using DFT compiler and TetraMAX tool to synthesis circuit and testability estimate).
- Gate level simulation and timing verification.
- Power estimation analysis (using Encounter tool to estimate).
- Post-layout gate level simulation.
- Layout verification DRC/LVS (using calibre tool for verification).
- Static timing analysis.

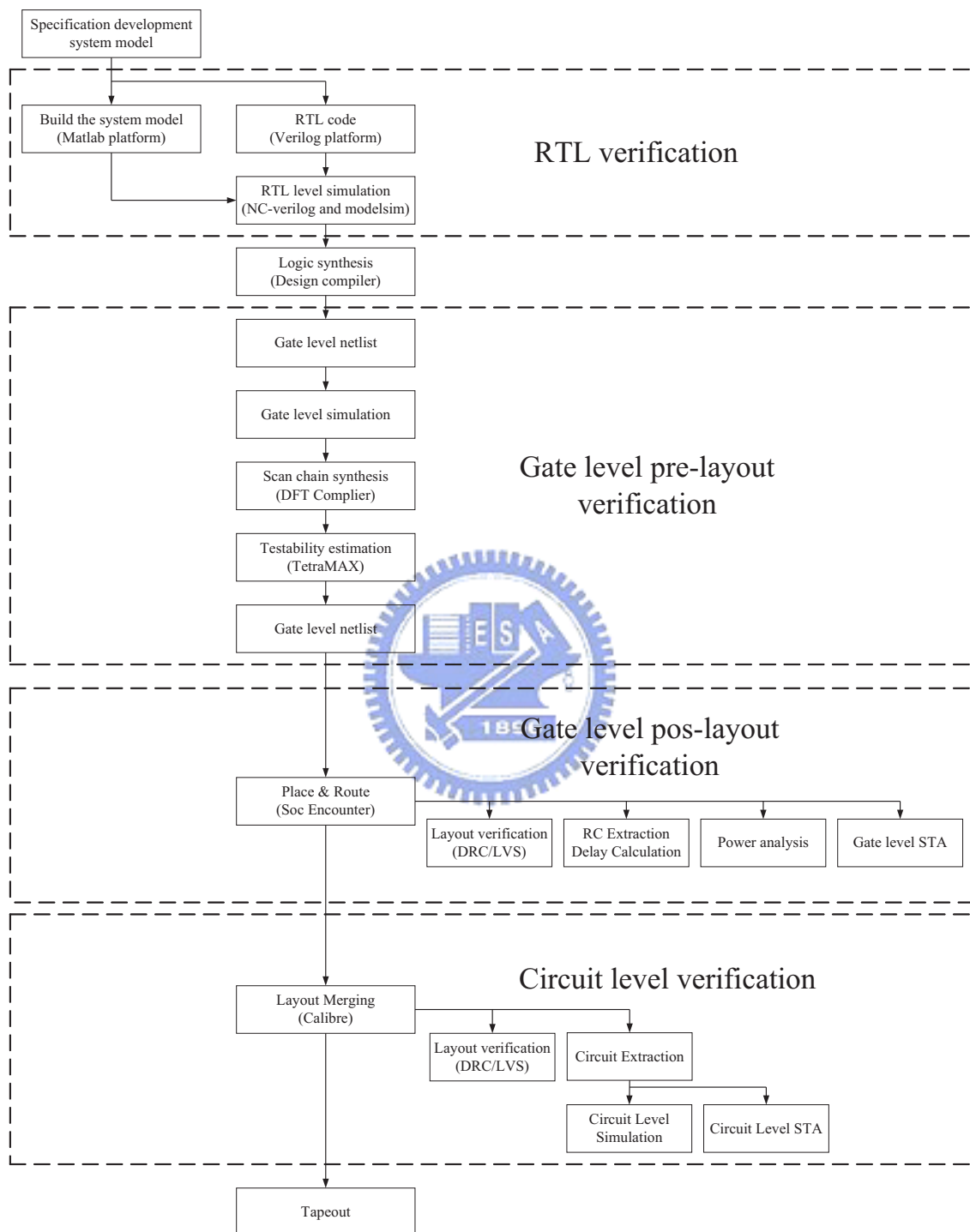


Figure 4.1: A Design flow.

4.1 Cell-based design flow

- Build the MATLAB platform and verify the RTL platform

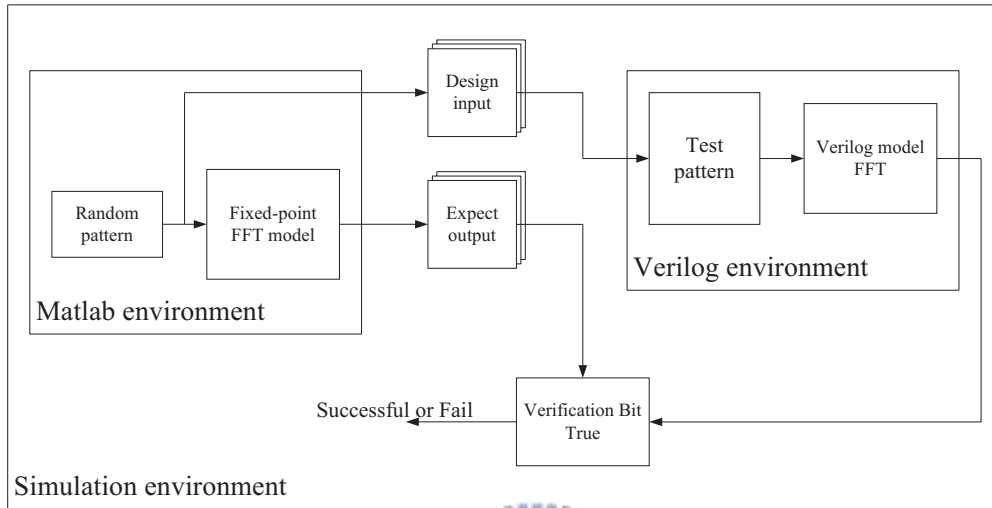
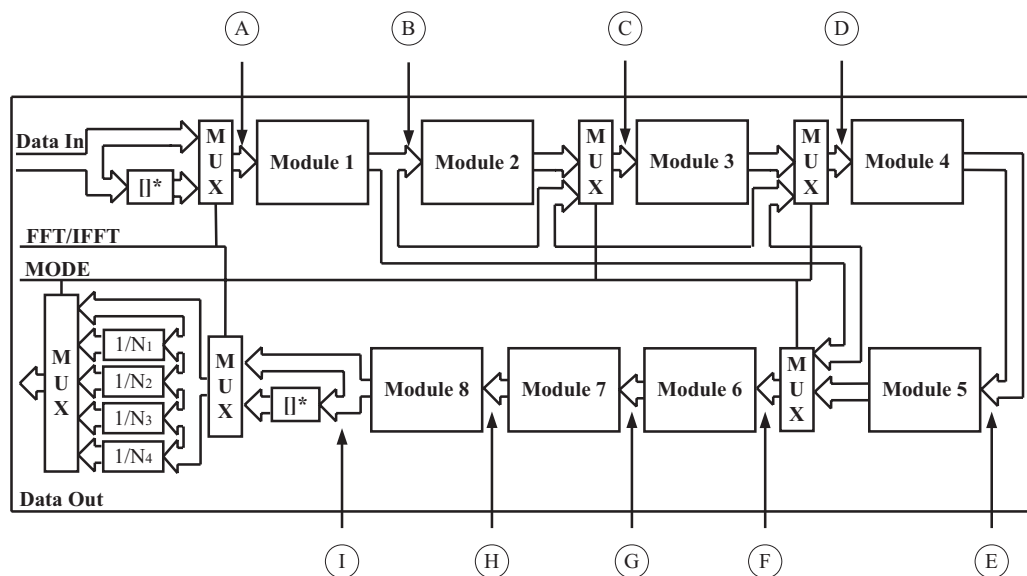


Figure 4.2: Simulation environment for variable FFT.

We need to build the MATLAB platform and verilog platform. Fig. 4.2 shows the simulation environment for bit-true verification. At first we need to develop floating-point FFT model in MATLAB environment. Since the FFT function in MATLAB tool is a floating-point function. We need to build the MATLAB quantizable FFT function using the proposed architecture mentioned earlier. When the quantizable FFT function ready, we can determine the bit-width for in dividual stages using the procedure mentioned in Sec. 3.2.6. Fig. 4.3 shows the quantized result in all stages. Note that total number of bits in all stage is 16. When the fixed-point FFT model is ready, we can generate random patterns from this model. That is, the design input and expected output can be generated from this MATLAB platform. We can save the input and the expect of output in a Text file for bit-true verification. In the verilog platform, we read input from the save text file generated by MATLAB platform. We compare the output from the verilog platform and the MATLAB platform to verify the result.



Point	A	B	C	D	E	F	G	H	I
Integer bit	2	2	3	3	4	4	4	6	7

Figure 4.3: Bit-width in all stage.

- Synthesis

We use the Design Compiler to synthesize the RTL code. At synthesis phase, we need to constrain the following conditions including timing, area and other rules to meet specification. The coding style is important since it affect the synthesis results a lot.

- Gate-level simulation

After synthesis, the gate level circuit will include timing information. Thus, we need to check the function correctness again. The nWave tool can help us to check the function with timing information.

- Memory BIST (Built-In Self-Test)

We use the memory generator to describe the specification and create the memory. We also use the memory specification to generate memory BIST

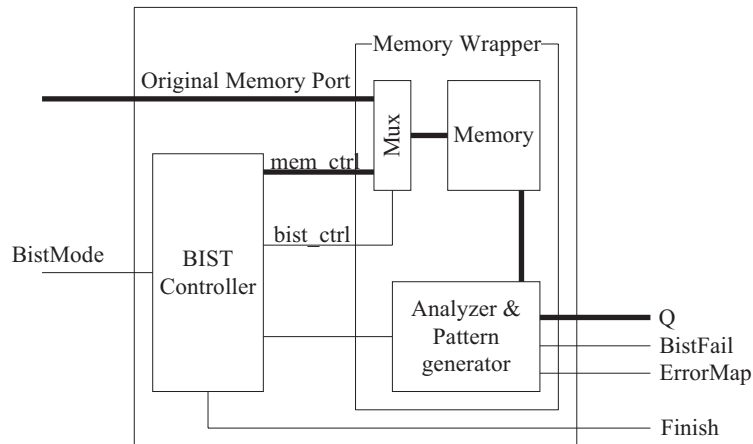


Figure 4.4: BIST circuit.

circuit, as shown in Fig. 4.4. From Fig. 4.4, the BistMode control the function mode and test mode. The BIST circuit contains memory wrapper and BIST controller. When the memory size is large, the number of pins for BistFail, ErrorMap and Finish is increase. We use the OR gate to connect each pin of BistFail or ErrorMap to reduce the core pad.

- Scan chain insertion

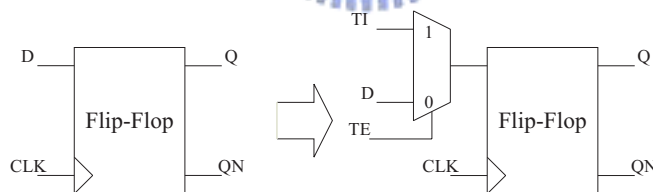


Figure 4.5: From Flip-Flop to scan Flip-Flop.

For testability the scan chain synthesis is needed and this can be done by DFT compiler. The Fig. 4.5 shows flip-flop after scan chain insertion.

- ATPG (Auto Test Pattern Generator)

We use the TetraMAX tool to generate patterns for testing. The function of testing is to test the fault of stuck-at 0 and stuck-at 1. After ATPG we

can get the information with fault coverage. The fault coverage reveal the probability that a chip is in good condition.

- Scan gate level simulation

After scan chain synthesis, we need to do scanned gate level simulation. From Fig. 4.5, due to the added multiplexors the critical path increases. Thus we need to adjust timing for function correctness.

- APR (Automatic Place and Route)

We use the SOC encounter to perform the placing and routing. After APR, we can check some parameters such as timing, power and design rule violation.

- DRC/LVS verification

We need to verify the DRC (Design Rule checking) and LVS (Layout V.S. Schematic) using the calibre tool.



4.2 Chip summary

- Chip layout

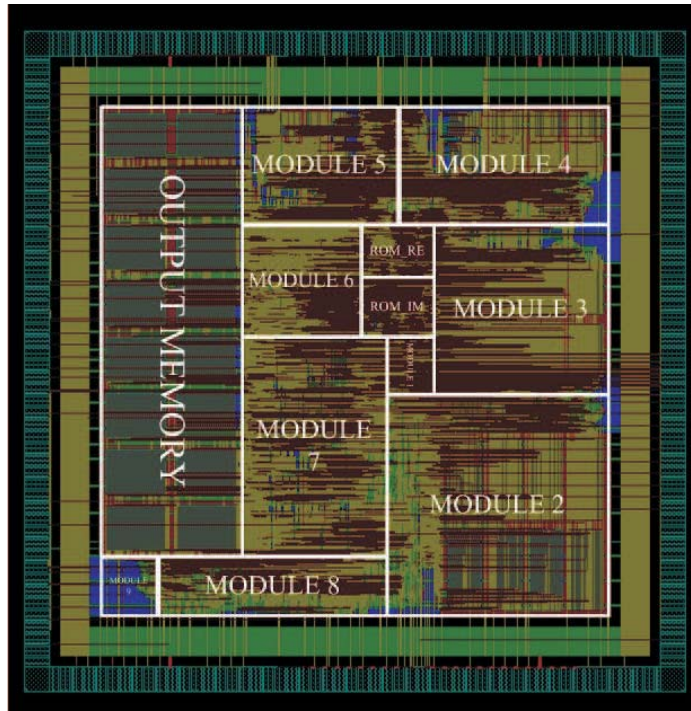


Figure 4.6: Layout view of the proposed FFT processor.

This proposed variable MIMO FFT processor is fabricated in TSMC 0.18 μm 1P6M CMOS technology. The layout is as shown in Fig. 4.6. Table 4.1 lists the expected chip performance and the performance satisfies the requirement for IEEE 802.16e standard. The 208-pin package will be used for the chip, where 175 pins are I/O pins and others are power pins. The core size is 25mm² and including total 31.9375K-byte SRAM that used in feedback memory, as shown in Fig. 4.6. The total power consumption and total area is 181mW and 41.8mm², respectively.

- Performance evaluation

Items	Specification
Technology	TSMC 18um
Package	CQFP208
Core size	25 mm ²
Die size	41.8 mm ²
Gate count	1350 K
Memory	31.9375 KB
Max Frequency	40 MHz
Power consumption	181 mW

Table 4.1: Expected chip performance of the proposed FFT processor.

- Performance comparison

Table 4.2 shows that performance comparison with others FFT architectures. The proposed architecture has advantages in throughput and area.

	This work	[16]	[19]	[20]	[21]	[22]
Technology	0.18um	0.13um	0.35um	0.18um	0.5um	0.6um
FFT size	2048-pt	128-pt	2048-pt	1024-pt	1024-pt	256-pt
Core area	25 mm ²	1.4 mm ²	N.A.	4.6 mm ²	N.A.	36 mm ²
Die area	41.8 mm ²	2.69 mm ²	12.25 mm ²	7.6 mm ²	40 mm ²	N.A.
Throughput	4R	4R	R	R	R	R
Work frequency	40 MHz	40 MHz	60 MHz	52 MHz	30 MHz	50 MHz
Power	181 mW	5.2 mW	574 mW	32 mW	N.A.	N.A.

Table 4.2: Comparison of chip performance.

Chapter 5

Conclusions

In this thesis, we proposed a variable FFT for MIMO-OFDM over Wi-MAX application. In chapter 2 we discuss several FFT architectures which can be applied in MIMO systems with various FFT size. In chapter 3 we use the advantages of the architectures in chapter 2 to implement our design. We also proposed a memory sharing method to reduce the memory size to 25% compare with R2SDF. In chapter 4, we discuss how to build the system platform in MATLAB environment and the chip implementation flow. Finally, we follow CIC design flow to implement the proposed FFT processor in a TSMC 0.18um technology. The total area and power consumption are 41.8mm² and 181mW, respectively.

Bibliography

- [1] A. V. Oppenheim, R. W. Schaffer, “Discrete-Time Signal Processing,” Prentice-Hall Inc., 1999.
- [2] S. He and M. Torkelson, “A new approach to pipeline FFT processor,” in Proc. of Int. Parallel Processing Symposium, pp. 766-770, Apr. 1996.
- [3] S. He and M. Torkelson, “Designing Pipeline FFT Processor for OFDM (de) Modulation,” URSI International Symposium on Signals, Systems and Electronics, pp. 257-262, 1998.
- [4] L. Jia, Y. Gao, J. Isoaho and H. Tenhunen,, “A New VLSI-Oriented FFT Algorithm and Implementation,” IEEE ASIC Conference, pp. 337-341, Sep. 1998.
- [5] W. C. Yeh, C. W. Jen, “High-speed and low-power split-radix FFT,” IEEE Trans. Acoust, Speech, Signal Processing, vol. 51, pp. 864-874, Mar. 2003.
- [6] B. M. Baas, “An approach to low power, high performance, fast Fourier transform processor design,” PhD Thesis, Stanford University, 1999.
- [7] C. P. Hsu, “Design of Fast Fourier Transform Processor in DVB-T Inner Receiver,” MS Thesis, Central University, 2005.
- [8] Y. T. Lin, P. Y. Tsai, and T.D.Chiueh, “Low-power Variable-length Fast Fourier Transform Processor,” IEEE Proc. Computer. Digit. Tech., vol. 152, no. 4, pp. 499-506, Jul. 2005.
- [9] J. G. Nash, “A High Performance Scalable FFT,” IEEE Wireless Communications and Networking Conference, pp. 2367-2372, Mar. 2007.

- [10] Y. Zhao, A.T. Erdogan, T. Arslan, "A low-power and domain-specific reconfigurable FFT fabric for system-on-chip applications," 19th IEEE Int., Parallel and Distributed Processing Symposium, pp. 4, Apr. 2005.
- [11] K. Manolopoulos, K. Nakos, D. Reisis, N. Vlassopoulos, V.A. Chouliaras, "High Performance 16K, 64K, 256K complex points VLSI Systolic FFT Architectures," 14th IEEE International Conference on Electronics, Circuits and Systems, pp. 146-149, Dec. 2007.
- [12] ETSI EN 300 401 (v1.3.2): "Radio broadcasting systems; digital audio broadcasting (DAB) to mobile, portable and fixed receivers," Sep. 2000.
- [13] ETSI EN 300 744 (v1.2.1): "Digital video broadcasting (DVB); framing structure, channel coding and modulation for digital terrestrial television," Jul. 1999.
- [14] T1E1.4/98-007R4: "Standards project for interfaces relating to carrier to customer connection of asymmetrical digital subscriber line (ADSL) equipment," Jun. 1998.
- [15] ETSI TS 101 270-2 (v1.1.1): "Transmission and multiplexing (TM); access transmission systems on metallic access cables; very high speed digital subscriber line (VDSL); Part 2: Transceiver specification," Feb. 2001.
- [16] S. F. Hsiao, W. R. Shiue, "Design of low-cost and high-throughput linear arrays for DFT computations: algorithms, architectures, and implementation," IEEE Trans. on Circ. and Syst. II. vol 47. pp. 1188-1203, 2000.
- [17] V. Boriakoff, "FFT computation with systolic arrays, a new architecture," IEEE Trans. on Circ. and Syst. II. vol 41. pp. 278-284, 1994.
- [18] Y. W. Lin, H. Y. Liu, and C. Y. Lee, "A 1-GS/s FFT/IFFT processor for UWB applications," IEEE Journal of Solid-State Circuits, vol. 40, issue 8, pp. 17226-1735, Aug. 2005.
- [19] Y. W. Lin and C. Y. Lee, "Design of an FFT/IFFT Processor for MIMO-OFDM Systems," IEEE Trans. Circuits Syst. I, Reg. Papers, vol.54, no.4, pp.807-815, Apr. 2007.

- [20] K. Maharatna, E. Grass, and U. Jagdhold, "A 64-Point Fourier Transform Chip For High-Speed Wireless LAN Application Using OFDM," *IEEE Journal of Solid-State Circuits*, vol. 39, no. 3, pp. 484-493, Mar. 2004.
- [21] J. C. Kuo, C. H. Wen, A. Y. Wu, "Implementation of a Programmable 64~2048-Point FFT/IFFT Processor for OFDM-based Communication Systems," in *Proc. IEEE ISCAS*, vol. 2, pp. 121-124, May 2003.
- [22] H. Zou and B. Daneshard, "VLSI implementation for a low power mobile OFDM receiver ASIC ," in *Proc. IEEE Wireless Communications and Networking Conference*, vol.4, pp. 2120-2124, Mar. 2004.
- [23] S. He and M. Torkelson, "Design and Implementation of a 1024-point pipeline FFT processor," in *Proc. IEEE Custom Integrated Circuits. Conf.*, pp. 131-134, May 1998.
- [24] L. Fanucci, M. Forliti, F. Gronchi, "Single-Chip Mixed-Radix FFT Processor for Real-Time On-Board SAR Processing," in *Proc. IEEE Int. Conf. on Electronics, Circuits and Systems*, vol. 2, pp. 1135-1138, Sep. 1999.

