國 立 交 通 大 學

電機資訊學院電子與光電學程

碩 士 論 文

利用混合方法進行管線化快速傅利葉轉換處理器的字元長度最佳化之研究

Hybrid Wordlength Optimization Methods of Pipelined FFT Processors

研 究 生 ： 郭 志 彬
指導教授 ： 周 景 揚 教授

中華民國 九十三 年 八 月

利用混合方法進行管線化快速傅利葉轉換處理器的字元長度最佳化之研究

Hybrid Wordlength Optimization Methods of Pipelined FFT Processors

研 究 生: 郭 志 彬　　Student : Chih-Bin Kuo
指導教授: 周 景 揚　　Advisor : Jing-Yang Jou

國 立 交 通 大 學

電機資訊學院電子與光電學程

碩 士 論 文

A Thesis

Submitted to Degree Program of Electrical Engineering Computer Science
College of Electrical Engineering and Computer Science
National Chiao Tung University
in Partial Fulfillment of the Requirements
for the Degree of
Master of Science
in
Electronics and Electro-Optical Engineering
August 2004
Hsinchu, Taiwan, Republic of China

中華民國 九十三 年 八 月

利用混合方法進行快速傅利葉轉換處理器的字元長度最佳化之研究

研究生：郭 志 彬　　指導教授：周 景 揚 教授

國立交通大學

電機資訊學院　電子與光電學程 (研究所) 碩士班

摘 要

快速傅利葉轉換處理器是許多通訊系統中的關鍵元件，利用快速傅利葉轉換處理器的設計自動化，可減輕系統設計的時程壓力。在設計管線化快速傅利葉轉換處理器時，處理級(Process Element)的字元長度是重要的參數。在本篇論文中，我們提出關於管線化快速傅利葉轉換處理器每一級的字元長度對訊號對量化雜訊比(SQNR)的統計學模型。更進一步地，提出透過混合使用統計與模擬誤差分析的管線化快速傅利葉轉換處理器字元長度的最佳化流程。在系統設計者所提供的快速傅利葉處理器點數、訊號對量化雜訊比和處理器速度限制條件下，本方法可於數秒內自動地產生一組最佳化的字元長度參數。實驗的結果顯示，此快速的最佳化流程可縮小 8192 點管線化快速傅利葉轉換處理器面積達 24%。

Hybrid Wordlength Optimization Methods of Pipelined FFT Processors

Student: Chih-Bin Kuo Advisor: Dr. Jing-Yang Jou

Degree Program of Electrical Engineering Computer Science
National Chiao Tung University

Abstract

The Fast Fourier Transform (FFT) processor is a key component in many communication systems. To reduce design time of FFT processors through design automation is to reduce the time pressure of system designers. When implementing a pipelined FFT processor the wordlength is of great importance. This thesis describes a statistical error model of pipelined FFT processors that calculates the signal to quantization noise ratio (SQNR) with wordlength of each process element (PE) stage. Furthermore, to speed up the design of specified FFT processor, a hybrid optimization method with statistical and simulation-based error analysis is presented. Under constraints of the number of FFT points, SQNR, and required processors speed, the optimized wordlength set for each PE stage can be generated within several seconds. The experimental results designate that this speedy flow can reduce 24% area of 8192-point pipelined FFT processors.

Acknowledgement

I would like to express my heartfelt gratitude to my advisor, Professor Dr. Jing-Yang Jou, for his guidance and encouragement throughout the three-year graduate course. I also deeply appreciate Assistant Professor Dr. Juinn-Dar Huang for his constructive suggestion on this thesis. Special thanks to all the EDA members for the wonderful time we share together. Finally, I display my warmest appreciation to my parents and my dear Juno and A-Kang for their love and support.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

The FFT is one of the most widely used digital signal processing algorithms. Recently, attention has been returned to real-time FFT processors in many communication systems. For example, FFT is one of the major building blocks in an Orthogonal Frequency Division Multiplexing (OFDM) based system, as shown in Figure1.1 [3], like HDTV, xDSL modems, and wide band mobile terminals.

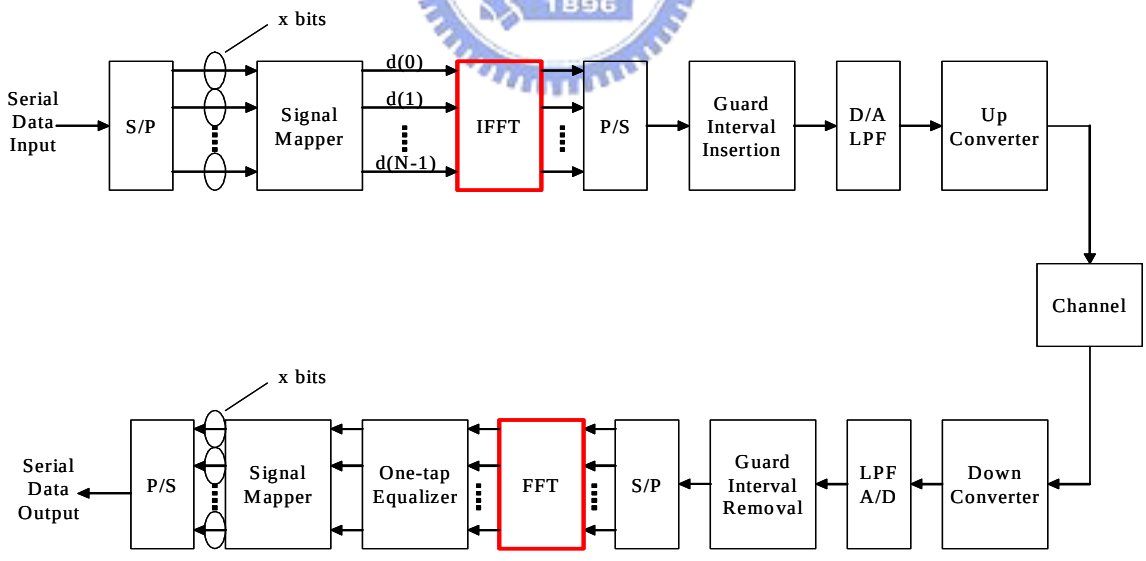


Figure 1.1 Architecture View of OFDM

There are many possible architecture choices for FFT processors. Among them, the pipelined FFT architectures that are particularly suitable for real-time applications since it

can easily be merged with the sequential nature of sampling. It is suitable for VLSI technology progresses because it is regular and its control circuit is easy to implement.

In the pipelined FFT architectures, the most research effort has been relative to the regular module implementations, which uses fixed wordlength for both data and coefficients for each stage. The possibility to use different wordlength is often ignored to achieve modulized solutions. However the fast growing use of Intellectual Property (IP) makes the non-module implementation viable, which allows us to exploit the pipeline architectures further. The wordlength may affect the precision, quantization error, and complexity of hardware. The increased wordlength will increase the precision and decrease the quantization error at the cost of area and power. On the other hand, to maintain a lower hardware cost, a shorter wordlength may be chosen at the sacrifice of the precision.

In general, a FFT can't be implemented exactly. Each multiplier and adder in pipelined FFT architectures may introduce an error caused by the rounding or truncation of the arithmetic results. Errors will accumulate successively over the FFT stages. The error introduced at the early stages may influence the performance in the later stages. Therefore, it is required to find an optimized solution of wordlength in pipelined FFT processors.

The statistical method and simulation-based method are popular for FFT error analysis between signal-to-quantization-noise ratio (SQNR) and wordlength. The SQNR can be calculated quickly by statistical model. With the advent of more powerful computers recently, SQNR of different algorithms and different architectures can be accurately simulated.

Stage level statistical error analysis method of pipelined FFT processor will be presented in this thesis. Furthermore, a hybrid wordlength optimization method of pipelined FFT processor will be introduced. The wordlength parameters of each stage are

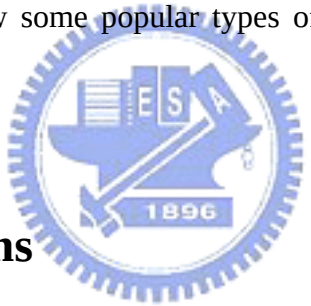
generated automatically by using the constraints of point of FFT, SQNR, and throughput of processors.

The rest of this thesis is organized as follows. In Chapter 2, a brief review of FFT algorithms and architectures is given. Error analysis methods are introduced in Chapter 3. In Chapter 4, the wordlength optimization of our approach is presented step-by-step. The experimental results are then presented in Chapter 5. Finally, the conclusions and the future works are given in Chapter 5.

Chapter 2

Review of FFT

Substantial literatures are available on algorithm and architecture of FFT. In this chapter, we will briefly review some popular types of algorithms of FFT. And we will introduce architectures of FFT.



2.1 FFT Algorithms

The Discrete Fourier Transform (DFT) plays a significant role in the region of digital signal processing (DSP) and communications. However, the computational complexity of direct evaluation of an *N*-point DFT is $O(N^2)$, which costs a long computation time and large power. Thus, there is a great requirement to develop a fast DFT algorithm. Many FFT algorithms have been derived to reduce computation complexity, such as Cooley-Turkey algorithm [1] [2], Rader algorithm [2], and Winograd algorithm [2]. Among them, the Cooley-Turkey algorithm is very popular because it can reduce the computational complexity from $O(N^2)$ to $O(N \log_2 N)$, and the regularity of the algorithm makes it suitable for VLSI implementation. It will be discussed in this section.

2.1.1 Basic Concepts of FFT Algorithms

FFT algorithms are approaches to compute DFT. The formulation of N length DFT is define as equation (2.1).

$$X(k) = \sum_{n=0}^{N-1} x(n) W_N^{nk}, \quad k = 0,1,\ldots\ldots,N-1 \tag{2.1}$$

where the coefficient W_N^{nk} is defined as equation (2.2) and is called twiddle factor, and the

$$W_N^{nk} = e^{\frac{-j2\pi nk}{N}} = \cos(\frac{2\pi nk}{N}) - j\sin(\frac{2\pi nk}{N}) \tag{2.2}$$

symmetric property is showed in Figure 2.1. The $X(k)$ is in frequency domain, and $x(n)$ is in time domain. Algorithms in which the decomposition is based on decomposing $x(n)$ term are called decimation-in-time (DIT) algorithms. On the other hand, algorithms in which the decomposition is based on decomposing $X(k)$ are called decimation-in-frequency (DIF) algorithms.

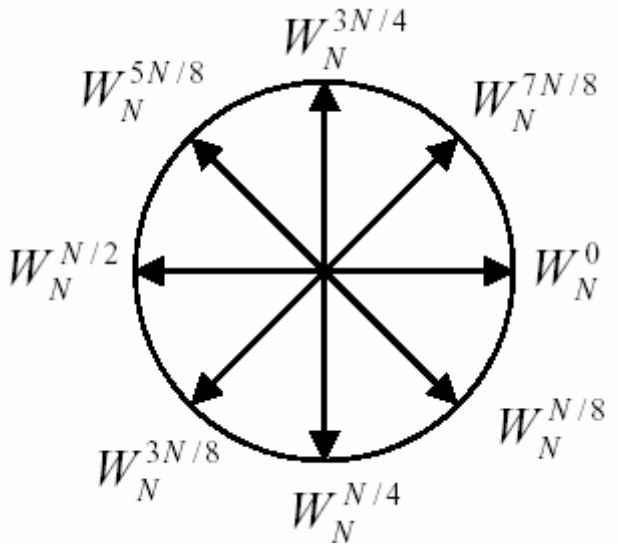


Figure 2.1 Symmetric Property of Twiddle Factor

2.1.2 Fixed-Radix FFT Algorithms

Fixed-Radix algorithms include the radix-2, radix-4/radix-2^2, radix-8/radix-2^3, etc. Among them, the radix-2 algorithm is the simplest one. The radix-4 algorithm has the smallest multiplicative complexity. And the radix-2^2 has benefits of radix-2 and radix-4. They will be reviewed in this section.

2.1.2.1 Radix-2 Algorithm

The radix-2 algorithm is using the divide-and-conquer approach with which algorithm is dividing the problem of N point FFT, where N is power-of-2, by factor of 2. With the symmetric property of equation (2.2), $W_N^{nk} = -W_N^{nk+N/2}$, the equation (2.3) will be founded.

$$A \times W_N^{nk} + B \times W_N^{nk+N/2} = (A + B \times W_N^{N/2}) \times W_N^{nk} = (A - B) \times W_N^{nk} \qquad (2.3)$$

By using the property of equation (2.3), the summation of equation (2.1) can be divided into two summations in equation (2.4), and it is the equation of radix-2 DIF.

$$X(2r) = \sum_{n=0}^{N/2-1} [x(n) + x(n+N/2)] W_{N/2}^{nr}$$

$$X(2r+1) = \sum_{n=0}^{N/2-1} [x(n) - x(n+N/2)] W_N^n W_{N/2}^{nr}, \qquad r = 0,1,\ldots,(N/2)-1 \qquad (2.4)$$

The addition and the subtraction operation of $x(n)$ and $x(n+N/2)$ in equation (2.4) are called the butterfly (BF) operation as shown in Figure 2.2. After $\log_2 N$ - times recursive decomposing, the complete radix-2 DIF algorithm can be obtained. Figure 2.3 shows the Signal Flow Graph (SFG) of N=16 radix-2 DIF algorithm FFT.

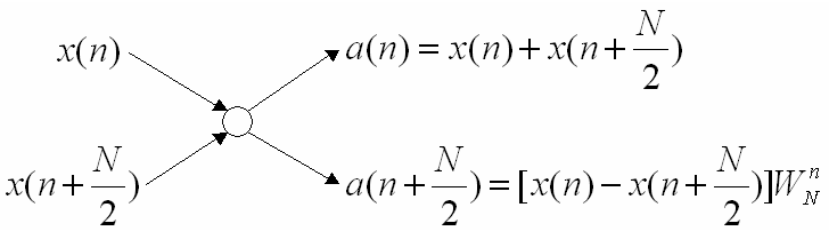


Figure 2.2 Butterfly Graph of the Radix-2 DIF FFT

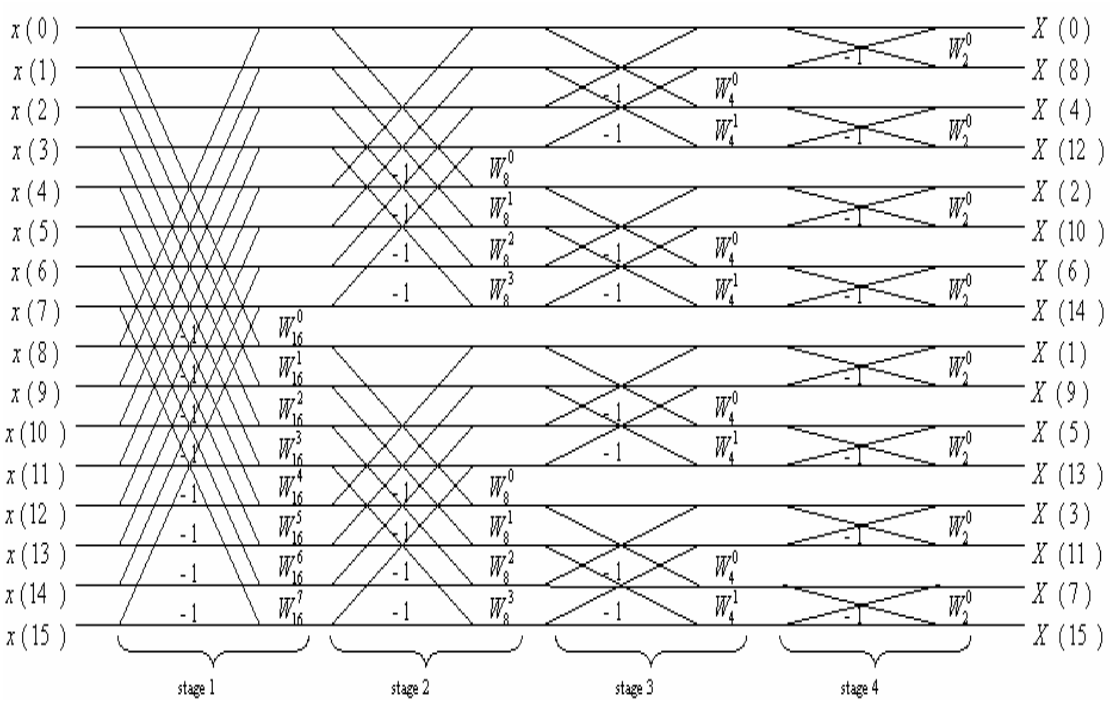


Figure 2.3 SFG of 16 Point Radix-2 DIF FFT

2.1.2.2 Radix-4 Algorithm

There is another symmetry property of equation (2.2) shown in equation (2.5).

$$W_N^{nk+N/4} = -W_N^{nk+3N/4} = -jW_N^{nk} \tag{2.5}$$

Because of the $-j$ term, we only need to exchange 2's complement of real part data and image part data instead of applying multiplication operation. The arithmetic cost can be reduced. The equation of radix-4 DIF [4] is shown in equation (2.6).

$$X(4r+l) = \sum_{n=0}^{N/4-1} [x(n) + x(n+\frac{N}{4}) \times W_4^l + x(n+\frac{N}{2}) W_4^{2l} + x(n+\frac{3N}{4}) \times W_4^{3l}] W_N^{nl} W_{N/4}^{nr},$$

$$r = 0,..., \frac{N}{4} - 1, \quad l = 0,1,2,3 \tag{2.6}$$

The mapping butterfly graph of equation (2.6) is shown in Figure 2.4.

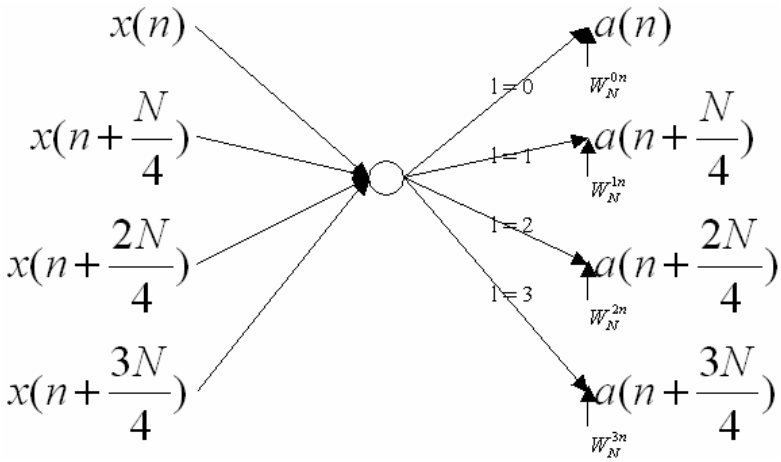


Figure 2.4 Butterfly Graph of Radix-4 DIF FFT

2.1.2.3 Radix-2^2 Algorithm

If we further divide the equation (2.6), we can get the equation (2.7) of radix-2^2 [4]. It implements the radix-4 BF by two radix-2 BFs. The mapping butterfly graph of equation (2.7) is shown in Figure 2.5.

$$X(4r+2l_2+l_1)=\sum_{n=0}^{N/4-1}[x(n)+x(n+\frac{N}{4})\times W_4^{2l_2+l_1}+x(n+\frac{N}{2})W_4^{4l_2+2l_1}+x(n+\frac{3N}{4})\times W_4^{6l_2+3l_1}]W_N^{n(2l_2+l_1)}W_{N/4}^{nr},$$

$$X(4r+2l_2+l_1)=\sum_{n=0}^{N/4-1}[x(n)+x(n+\frac{N}{4})\times W_4^{2l_2+l_1}+x(n+\frac{N}{2})W_4^{4l_2+2l_1}+x(n+\frac{3N}{4})\times W_4^{6l_2+3l_1}]W_N^{n(2l_2+l_1)}W_{N/4}^{nr}$$

$$r=0,...,\frac{N}{4}-1,\quad l_1=0,1\quad l_2=0,1 \qquad (2.7)$$

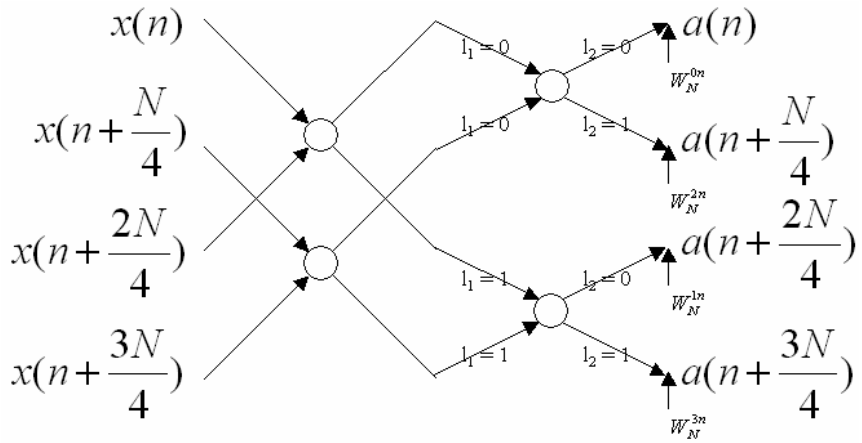


Figure 2.5 Butterfly Graph of Radix-2^2 DIF FFT

2.1.3 Split-Radix FFT Algorithms

The computation cost of the Fixed-Radix algorithm FFT can be further reduced by combining radix-2 and radix-4 or radix-2 and radix-8, called Split-Radix algorithm. It has fewer multiplications and additions. So, they have advantage on computational complexity. But, they are not regular as radix-2^r algorithms and seldom used in ASIC design. The most popular split-radix algorithms are proposed by Duhamel et al. [5].

2.2 FFT Architectures

The FFT is one of the most widely used digital signal processing algorithms. Recently, attention has been returned to real-time processors in many communication systems. There are many architecture choices for these processors. Among them, the pipelined architectures are particularly suitable for real-time applications since they are easily merged with the sequential nature of sampling. And they are popular for large FFT VLSI realization too, due to their high regularity.

In this section, we will introduce the pipeline-based architecture. The architecture that we want to discuss is used to implement DIF FFT algorithms. Similar structures can be designed for DIT FFT algorithms, too.

Several architectures for pipelined FFT processors have bean proposed. There are Radix-2 Multi-path Delay Commutator (R2MDC) [6], Radix-2 Single-path Delay Feedback (R2SDF) [7], Radix-2^2 Single-path Delay Feedback (R2^2SDF) [8][9], Radix-4 Single-path Delay Feedback (R4SDF) [6], Radix-4 Multi-path Delay Commutator (R4MDC) [6], etc. They will be introduced in this section.

- **R2MDC**

It is the most straightforward way to reorganize the data for FFT algorithms. At each stage half the data stream is delayed via the memory and processed with the second half data stream. An 16-point R2MDC is shown in Figure 2.6.

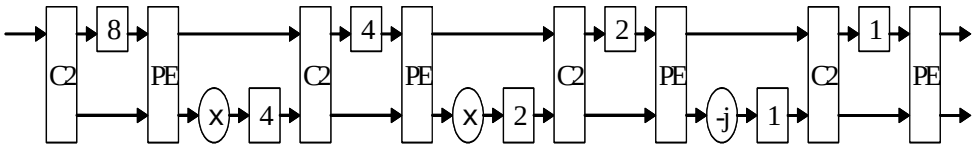


Figure 2.6 R2MDC Architecture (N=16)

- **R2SDF**

Since memory in R2MDC is idle at 50% of time, it can be reused as shown in Figure 2.7 This scheme utilizes the different arrival time of input data and processed data. The utilization of the memory is 100%.

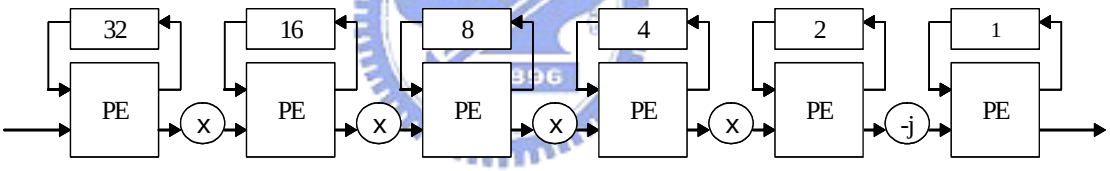


Figure 2.7 R2SDF Architecture (N=64)

- **R4MDC**

It is similar with R2MDC, but it utilizes only 25% of time for memory. A 256-point R4MDC is shown in Figure 2.8.

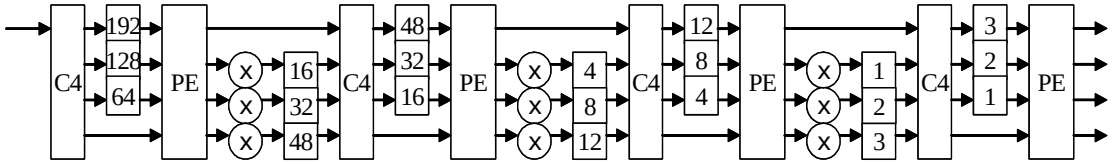


Figure 2.8 R4MDC Architecture (N=256)

- **R4SDF**

It is a radix-4 version of R2SDF. It is as efficient as R2SDF in terms of memory utilization and the utilization of multipliers increases from 50% to 75% at a cost of only 25% utilization of the BF element. A 64-point R4SDF is shown in Figure 2.9.

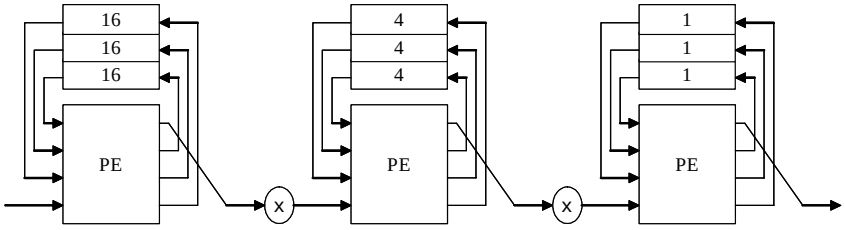


Figure 2.9 R4SDF Architecture (N=64)

- **R2^{2}SDF**

It breaks one radix-4 BF operation into two radix-2 BF operation with trivial multiplications of ± 1 and $\pm j$. With a feedback mechanism, the memory is fully utilized as R2SDF and R4SDF. A 64-point R2^{2}SDF is shown in Figure 2.10.

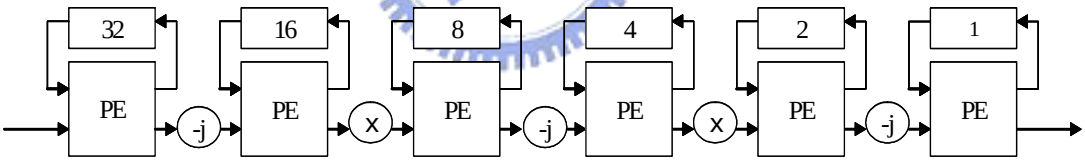


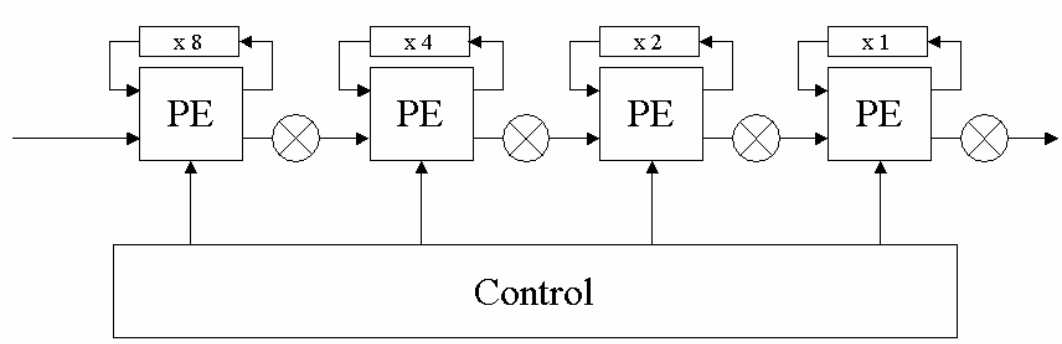
Figure 2.10 R2^2 SDF Architecture (N=64)

Architecture	Multiplier	Adder	Memory	Control
R2MDC	$2(\log_4 N-1)$	$4\log_4 N$	$\frac{3N}{2}-1$	Simple
R2SDF	$2(\log_4 N-1)$	$4\log_4 N$	$N-1$	Simple
R4SDF	$\log_4 N-1$	$8\log_4 N$	$N-1$	Medium
R4MDC	$3(\log_4 N-1)$	$8\log_4 N$	$\frac{3N}{2}-1$	Simple
R2^{2}SDF	$\log_4 N-1$	$4\log_4 N$	$N-1$	Simple

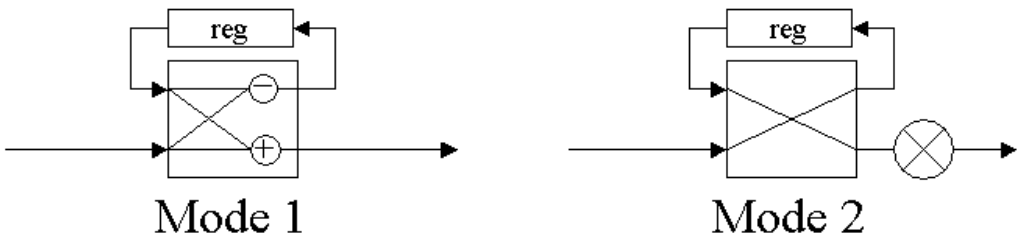
Table 2.1 Summary of N Point Pipelined FFT Architectures

Summary of these architectures are shown in Table 2.1 [5]. The delay feedback approached are always more efficient than corresponding delay commutator approaches in terms of memory requirements. The Radix-4 algorithm based single-path architectures have fewer multipliers than those of radix-2 algorithm. However, radix-2 algorithm based architectures have properties of simple and regular. And radix-2^2 algorithm is characterized with the trait that it has same multiplicative complexity as radix-4 algorithms but still retains the radix-2 butterfly structure. In this thesis we will focus on R2SDF and R2^2SDF architectures.

The detail architecture with control unit of R2SDF and R2^2SDF is shown in Figure 2.11(a). The butterfly process element (PE) has two kinds of operation modes. Mode 1 is used to store the data in the shift register, wait several cycles to compute and multiply with twiddle factors, while mode 2 is responsible for butterfly computation, showed in Figure 2.11(b).



(a)



(b)

Figure 2.11 Units of R2SDF and R2^2SDF (N=16)

Chapter 3

Error Analysis

Fixed-point arithmetic is popular for FFT hardware implementation for its simplicity. Because of the finite wordlength in the computation, we have to truncate or round the answers when overflow occurs after addition or multiplication; thus, errors are produced. The statistical error analysis and simulation-based error analysis are the two most popular methods for FFT error analysis. Many papers about statistical and simulation-based error analysis of fixed-point FFT have been published [10-14]. The previous statistical error analysis is not sufficient for our purpose of choosing the required wordlength stage by stage. We derive a simplified statistical error model to meet the requirement.

In this chapter, we will briefly review the quantization error analysis first. Second, we will introduce the statistical error models in which wordlength can be freely chosen stage by stage. Third, the simulation environment will be briefly reviewed. Then accuracy of our error models will be evaluated by comparing it with that of the simulation-based error analysis.

3.1 Error Analysis of Quantization

The basic formula for the quantization error analysis is shown below. Let X be a

finite-length sequence $\{x(n)\}$; $n = 0,1,2,..., N-1$. The expected value of X is shown in equation (3.1). It is zero-mean random sequence at the quantizer input. The variance of X is denoted by σ_x^2 and is shown in equation (3.2).

$$\mu_x = E[X] = \frac{1}{N}\sum_{n=0}^{N-1} x(n) = 0 \tag{3.1}$$

$$\sigma_x^2 = E[X^2] = \frac{1}{N}\sum_{n=0}^{N-1} [x(n) - \mu_x]^2 \tag{3.2}$$

where the $E[\cdot]$ in equation (3.1) and (3.2) is the expected value operator.

A quantizer $Q(\cdot)$ maps X into the discrete-valued Y. Thus, the quantization error $Q = X - Y$. Denote the boundaries by $\{b_k\}_{k=0}^{M}$ and the reconstruction levels by $\{y_k\}_{k=1}^{M}$, then the output of this quantizer is shown in equation (3.3) and the quantization error variance, denoted by σ_q^2, is then given by equation (3.4).

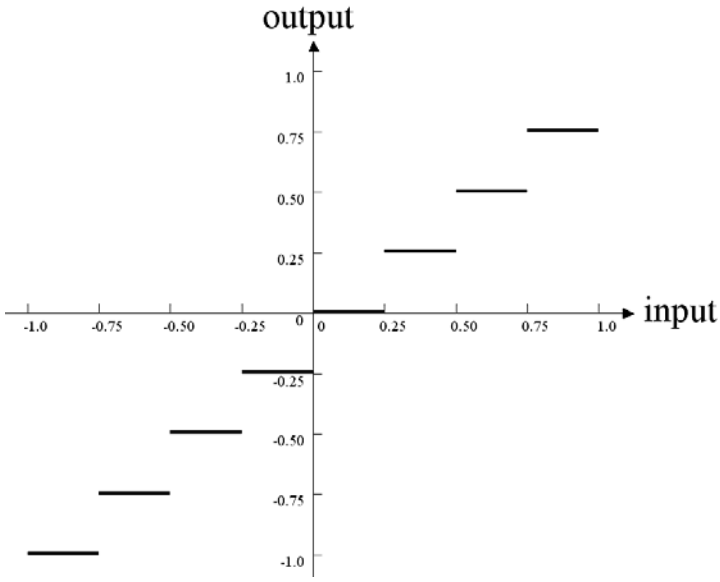
$$Y = Q(x) = y_k \quad \text{iff} \quad b_{k-1} < x \le b_k \tag{3.3}$$

$$\sigma_q^2 = E[Q^2] = \frac{1}{N}\sum_{k=1}^{M} \int_{b_{k-1}}^{b_k} [x - y_k]^2 dx \tag{3.4}$$

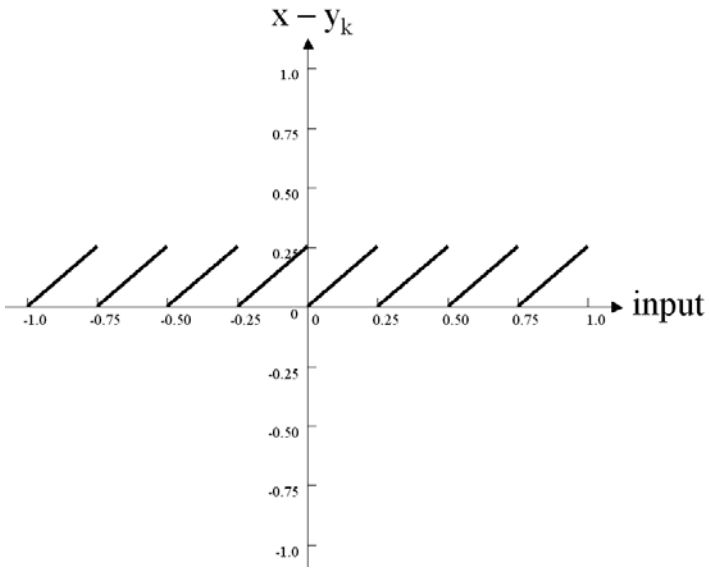
Finally, the equation of SQNR is shown in equation (3.5)

$$SQNR = \frac{\sigma_x^2}{\sigma_q^2} \tag{3.5}$$

For example, if the input is uniformly distributed in the interval (-1, 1) and the output is 2 + 1 bits sign-fractional discrete-valued data. The input-output mapping is shown in Figure 3.1(a). It is shown that, if the input data are in the interval $[0,0.25)$ then the output data of them are all have the same value as 0. If the input data are in the interval $[0.25,0.5)$ then the output data of them are 0.25, and so on. The related quantization error mapping is shown in Figure 3.1(b).



(a)



(b)

Figure 3.1 Information of 2+1 Bits Quantizer

3.2 Statistical Error Models of FFT

The previous FFT error analysis and model of DIF radix-2 algorithm have been presented by Sundaramurthy et al. [12] . They assume that all the wordlength of all PE stages is the same. This is insufficient for applications that allow the different wordlength

between PE stages.

Due to the finite wordlength in the computation, we have to truncate or round the answers after calculation. And the FFT computation is an iterative process and the value increases in magnitude. The problem of overflowing should be concerned.

In order to prevent overflow and to ensure output accuracy, data need to be scaled. There are two scaling methods to prevent FFT from overflow. One is overall scaling and the other is stage-by-stage scaling [2]. The input constraint of FFT with overall scaling is $|x(n)| < \frac{1}{N}$, and there is no need to divide the input of each butterfly by two. The input constraint of FFT with stage-by-stage scaling is $|x(n)| < 1$, and the input data should be divided by 2 for each butterfly. Due to the noise consideration [14] the stage-by-stage scaling will be used in this thesis.

In this section we aim on delivering statistical FFT error models for DIF radix-2 and radix-4 algorithms with stage-by-stage scaling scheme. These models are useable for case having the different wordlength stage by stage.

3.2.1 Definitions and Constraints

In these analyses, we assume fixed-point arithmetic with $(b_k + 1)$ bit wordlength and signed fraction, where *k* is the stage number of PE stage. The input of *N*-point FFT, denoted by $x(m)$ where $m = 0,1,2,...,N-1$, is a sequence of finite valued complex numbers. Numbers are consisted by 2*N* real random variable and they are uncorrelated. And they are distributed uniformly in $(\frac{-1}{\sqrt{2}}, \frac{1}{\sqrt{2}})$. Note the range of $(\frac{-1}{\sqrt{2}}, \frac{1}{\sqrt{2}})$ is consistent with the condition that $|x(m)| < 1$ for all *m*. The effect of the inaccuracy in the twiddle factor, W^p, is not treated here. The truncation operations are all modeled as mutually uncorrelated.

3.2.2 Expected Noise Sources

Figure 3.2 shows the error model of PE stage with stage-by-stage scaling by 2. There are several noise sources having been considered. They are the quantization error of wordlength difference between PE stages, denote by $\sigma_{q_k}^2$, the quantization error of scaling, denoted by $\sigma_{s_k}^2$, the quantization error of complex multiplication of twiddle factor, denoted by $\sigma_{m_k}^2$, and the insufficient output wordlength error, denoted by $\sigma_{Q_o}^2$.

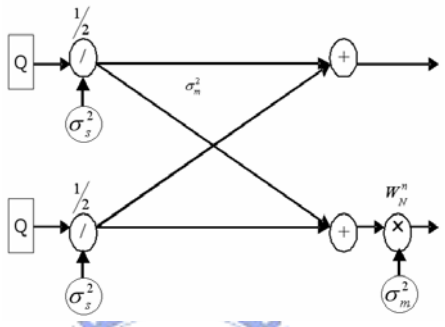


Figure 3.2 Error Model of PE Stage

- **$\sigma_{q_k}^2$ and $\sigma_{s_k}^2$**

The $\sigma_{q_k}^2$ is produced when the wordlength of stage *k*-1, denoted by $b_{k\text{-}1}$, is greater then that of stage *k*, denoted by b_k. $\sigma_{q_k}^2$ is the variance of truncated bits from $b_{k\text{-}1}$ to b_k. The scaling error is produced when $b_k < b_{k\text{-}1} + 1$. A complex scaling consists of two real scaling, i.e., the real and imaginary parts of the number are scaled separately. Scaling by a factor $\frac{1}{2}$ involves a 1-bit right shift and truncation of the last bit. $\sigma_{s_k}^2$ is the variance of this bit.

The sum of errors $\sigma_{q_k}^2$ and $\sigma_{s_k}^2$ can be replaced as the error of directly scaling the data of stage *k*-1 then truncate to b_k. This new error is denoted by $\sigma_{s_k}^2$ to replace the combination error of old $\sigma_{s_k}^2$ and $\sigma_{q_k}^2$. It is shown in equation (3.6).

$$\sigma_{s_k}^2 = \begin{cases} 0 & ; b_{k-1}+1 \le b_k \\ 2\left(\frac{1}{M}\cdot 2^{-2(b_{k-1}+1)}\cdot \sum_{v=0}^{M-1} v^2\right) & , M = 2^{(b_{k-1}+1)-b_k} \quad ; b_{k-1}+1 > b_k \end{cases} \tag{3.6}$$

- $\sigma_{m_k}^2$

It is assumed that a complex multiplication is implemented by four real multiplications and each real multiplication is truncated separately. The complex multiplication error variance, denoted by $\sigma_{m_k}^2$, is equal to the variance of truncated bits of the result of multiplication. It is shown in equation (3.7).

$$\sigma_{m_k}^2 = \begin{cases} 4\left(\frac{1}{M}\cdot 2^{-2\cdot(b_{k-1}+1+b_k)}\cdot \sum_{v=0}^{M-1} v^2\right) & , M = 2^{(b_{k-1}+1+b_k)-b_k} \quad ; b_{k-1}+1 \le b_k \\ 4\left(\frac{1}{M}\cdot 2^{-2\cdot 2b_k}\cdot \sum_{v=0}^{M-1} v^2\right) & , M = 2^{2b_k-b_k} \quad ; b_{k-1}+1 > b_k \end{cases} \tag{3.7}$$

- $\sigma_{q_o}^2$

If the output wordlength is small then the output wordlength of the last PE stage the quantization error will be produced. The variance $\sigma_{Q_o}^2$ is shown in equation (3.8), where the b_L is the wordlength of the last PE stage and the b_o is the FFT output wordlength.

$$\sigma_{q_o}^2 = \begin{cases} 0 & ; b_L \le b_o \\ 2\left(\frac{1}{M}\cdot 2^{-2b_L}\cdot \sum_{v=0}^{M-1} v^2\right) & , M = 2^{b_L-b_o} \quad ; b_L > b_o \end{cases} \tag{3.8}$$

3.2.3 Output Signal to Quantization Noise Ratio (SQNR)

Since all the noise sources are assumed to be uncorrelated, the variance of the noise at output node of the SFG of Figure 2.5 is the sum of contributions from all the individual noise sources that propagate to that output node. Some of noise variance of output nodes

that is contributed by $\sigma_{s_k}^2$ is denoted by σ_S^2, and the contribution of $\sigma_{m_k}^2$ is denoted by σ_M^2.

From Figure 3.3, the propagation of $\sigma_{s_k}^2$ in 8-point DIF Radix-2 it can be found. The number of error source $\sigma_{s_k}^2$ propagating to any output node from the first, second, and third stage are 8, 4, and 2, respectively. And the equation of σ_S^2 is shown below, equation (3.9), where the total stage number *n* is equal to $\log_2 N$, and the factor of $(\frac{1}{4})^{n-k}$ is the effect of scaling on the error propagating at stage *k*.

The σ_S^2 of DIF Radix-2^2 algorithm is the same as DIF Radix-2.

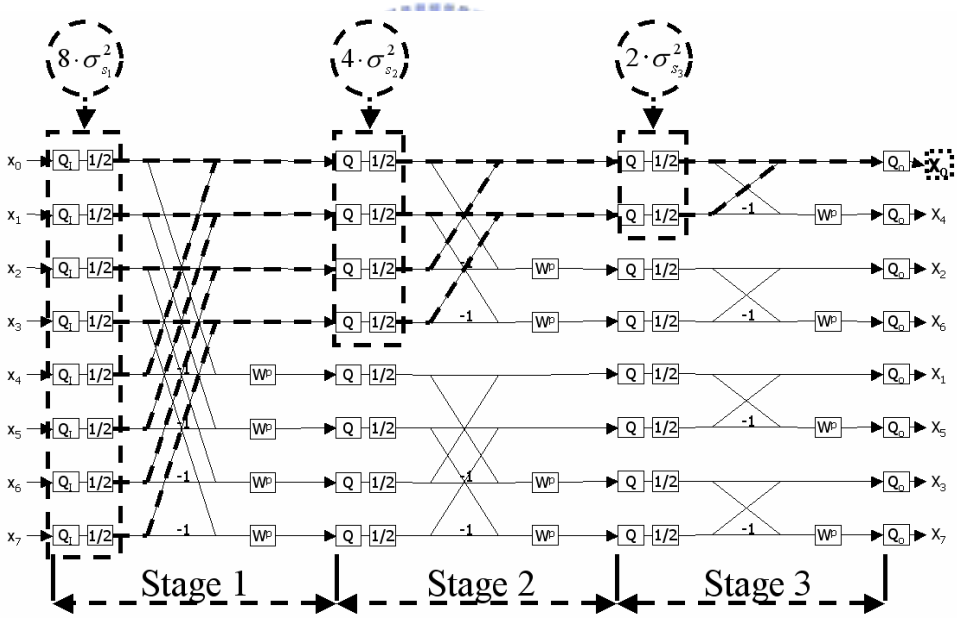


Figure 3.3 Propagating Flow of Quantization and Scaling Errors

$$\sigma_S^2 \approx N(\frac{1}{4})^{n-1}\cdot\sigma_{s_1}^2 + \frac{N}{2}(\frac{1}{4})^{n-2}\cdot\sigma_{s_2}^2 + \cdots + \frac{N}{2^{n-1}}(\frac{1}{4})^{n-n}\cdot\sigma_{s_n}^2 \qquad (3.9)$$

It can be assumed that all the complex multiplications are noisy for convenience of derivation. Figure 3.4 show that the propagation of $\sigma_{m_k}^2$ in 8-point DIF Radix-2 algorithm

SFG. In general, there are four, half of 8, $\sigma_{m_k}^2$ in each stage, and each $\sigma_{m_k}^2$ from the first (k=1), second (k=2), and third (k=3) stage propagates to 4, 2, and 1 output nodes. Hence it is easy to show σ_M^2 in equation (3.10).

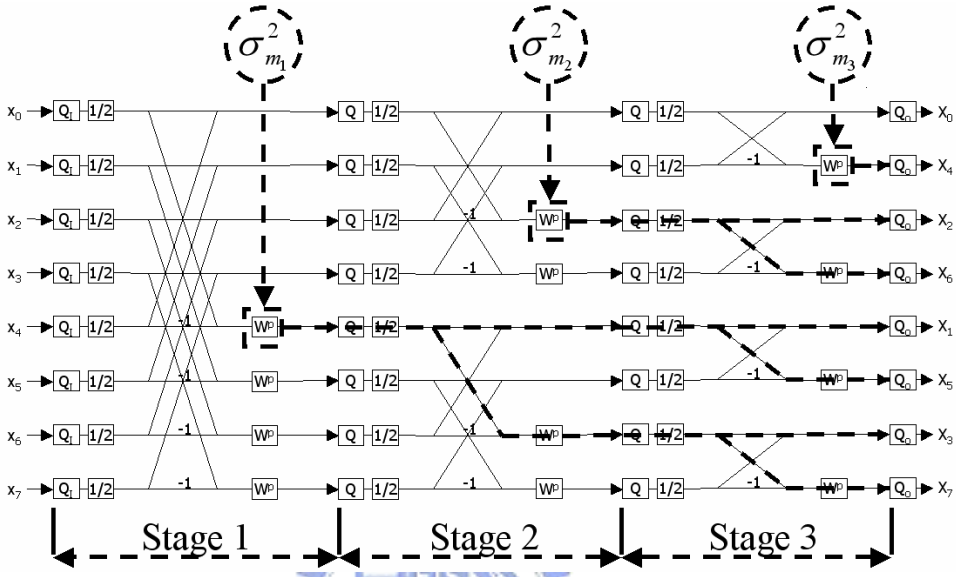


Figure 3.4 Propagating Flow of Mutiplication Errors

$$\sigma_{\mathrm{M}}^2 \approx \frac{1}{N}\frac{N}{2}[\frac{N}{2}(\frac{1}{4})^{n-1}\cdot\sigma_{\mathrm{m}_1}^2+\frac{N}{2^2}(\frac{1}{4})^{n-2}\cdot\sigma_{\mathrm{m}_2}^2+\cdots+\frac{N}{2^n}(\frac{1}{4})^{n-n}\cdot\sigma_{\mathrm{m}_n}^2] \tag{3.10}$$

The corresponding expression of σ_M^2 of Radix-2^2 algorithm is shown in equation (3.11)

$$\sigma_{\mathrm{M}}^2 \approx \frac{1}{N}\frac{3N}{4}[\frac{N}{4}(\frac{1}{16})^{n-1}\cdot\sigma_{\mathrm{m}_1}^2+\frac{N}{4^2}(\frac{1}{16})^{n-2}\cdot\sigma_{\mathrm{m}_2}^2+\cdots+\frac{N}{4^n}(\frac{1}{16})^{n-n}\cdot\sigma_{\mathrm{m}_n}^2] \tag{3.11}$$

In obtaining equation (3.10) and (3.11), it is assumed that all complex multiplications are noisy. But multiplications associated with twiddle factor $W^p=\pm 1$ or $W^p=\pm j$ introduce no errors. Figure 3.5 shows the position of noiseless twiddle factors of 8-point Radix-2 algorithm. The propagation of these noise sources is identical to that in the σ_M^2.

Thus, denoting the noise variance contribution of these multiplications by σ_C^2, and the expression of σ_C^2 is shown in equation (3.12). The corresponding expression of Radix-2^2 algorithm is shown in equation (3.13).

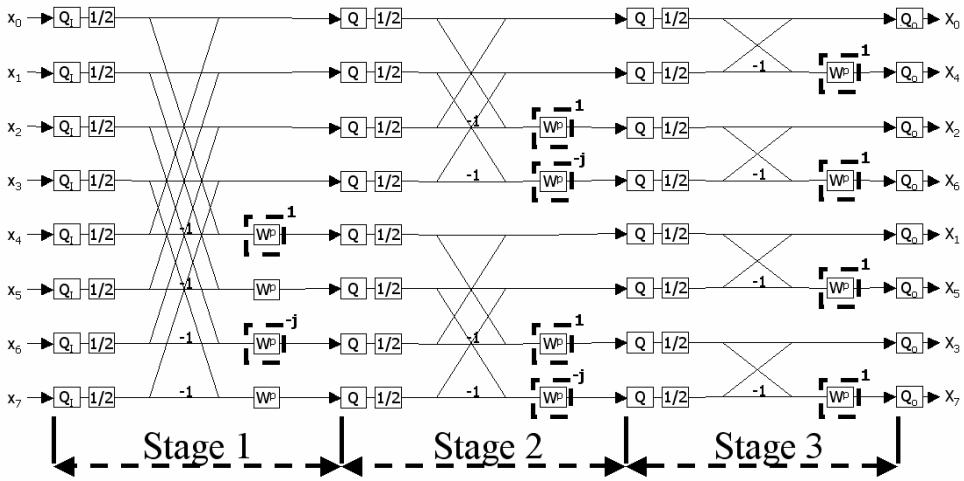


Figure 3.5 Propagating Flow of Noiseless Mutiplication

$$\sigma_C^2 \approx \frac{1}{N}[2\frac{N}{2}(\frac{1}{4})^{n-1}\cdot\sigma_{m_1}^2 + 2^2\frac{N}{2^2}(\frac{1}{4})^{n-2}\cdot\sigma_{m_2}^2 + \cdots + 2^{n-1}\frac{N}{2^{n-1}}(\frac{1}{4})\cdot\sigma_{m_{n-1}}^2] + \frac{1}{N}[\frac{N}{2}\frac{N}{2^n}\cdot\sigma_{m_n}^2] \tag{3.12}$$

$$\begin{aligned}\sigma_C^2 \approx \frac{1}{N}\cdot\frac{3N}{4}[&(\frac{1}{4^{n-1}})\cdot\frac{N}{4}\cdot(\frac{1}{16})^{n-1}\cdot\sigma_{m_1}^2 + (\frac{1}{4^{n-2}})\cdot\frac{N}{4^2}\cdot(\frac{1}{16})^{n-2}\cdot\sigma_{m_2}^2 + \cdots \\ &+ (\frac{1}{4^2})\cdot\frac{N}{4^{n-2}}\cdot(\frac{1}{16})^2\cdot\sigma_{m_{n-2}}^2 + (\frac{1}{4})\cdot\frac{N}{4^{n-1}}\cdot\frac{1}{16}\cdot\sigma_{m_{n-1}}^2 + \frac{N}{4^n}\cdot\sigma_{m_n}^2]\end{aligned} \tag{3.13}$$

The average output signal variance is in equation (3.14) [2].

$$\sigma_x^2 = \frac{1}{3N} \tag{3.14}$$

Finally, the SQNR expression is shown in equation (3.15).

$$SQNR = 10\log_{10}\frac{\sigma_x^2}{[\sigma_S^2 + (\sigma_M^2 - \sigma_C^2) + \sigma_{Qo}^2]} \tag{3.15}$$

3.3 Simulation-Based Error Analysis of FFT

There are many papers about simulation-based error analysis being published. Johansson et al. published a paper on simulation-based error analysis [17] in 1999. The C model is used to perform the simulation. User can get the proper result under their constraints. The wordlength of each stage, rounding or truncation for each stage, number of stages to do scaling, and the number of bits are parameters which can be chosen by users. Figure 3.6 shows the simulation environment of SQNR. It compares the outputs of fixed-point FFT and floating-point FFT to calculate the SQNR. The SQNR calculation expression is shown in equation (3.16).

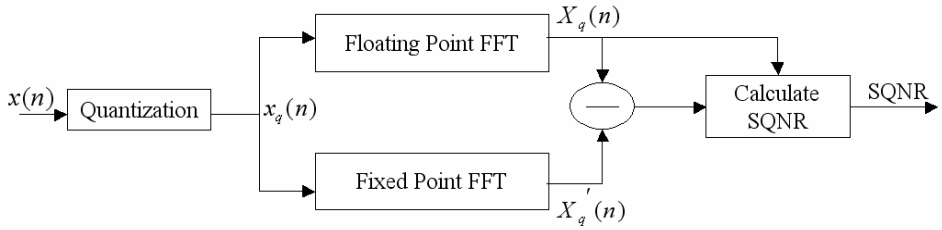


Figure 3.6 Simulation Environment of SQNR

$$SQNR = 10\log_{10}\frac{\sum_{n=0}^{N-1} X_q(n)^2}{\sum_{n=0}^{N-1}(X_q(n) - X_q'(n))^2} \quad (3.16)$$

3.4 Verifications

Since the SQNR can be calculated by simulation-based error analysis the simulation setup can be used to verify our new error models too.

The wordlength 8 to 32 bits is the popular selection to implement fixed-point FFT architectures. In this section, we will calculate the SQNR by statistical and simulation-based methods for 8, 16, …, 8192 points DIF Radix-2 FFT and 16, 64,…, 4096 points DIF Radix-2^2 FFT with the freely chosen wordlength from 8 to 32 bits for each PE

stage. Then, we will compare the results to verify statistical error models.

First, we choose wordlength, 8 to 32 bits, for each PE stage randomly. Second, we will compare all wordlength set in a special range.

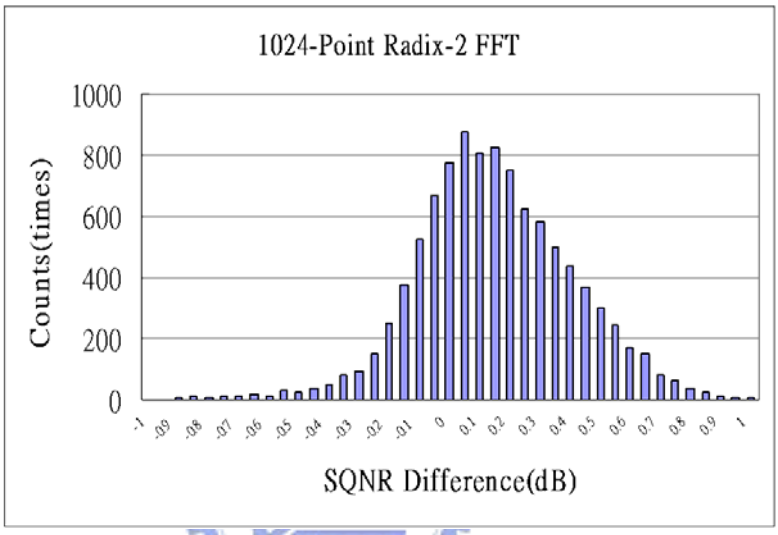
3.4.1 Random Verification

For example, we randomly generate 20 wordlength sets of 1024 points DIF Radix-2 FFT. The input worldlength is equal to that of the first PE stage, and the output wordlength is equal to that of the last PE stage. Then, calculate the SQNR by statistical and simulation-based methods, respectively. Then, the SQNR difference between them can be calculated. Table 3.1 shows the results. The first column shows the number of wordlength sets, next column shows the wordlength of each PE stage, column 3 shows the result SQNR of simulation-based error analysis, column 4 shows the SQNR result of statistical error analysis, and the last column is the difference of SQNR.

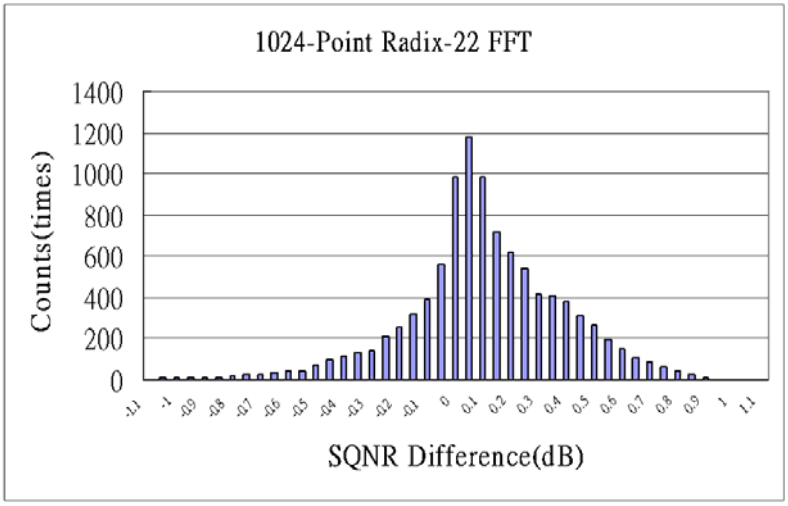
Wordlength Scheme	PE Stage										SQNR (dB)		
	1	2	3	4	5	6	7	8	9	10	Simlation	Statistical	Difference
1	29	32	22	32	30	21	10	10	20	18	21.57151	21.225869	-0.345643
2	17	26	21	32	23	12	31	21	26	23	40.64704	40.568971	-0.078069
3	8	28	14	16	13	32	29	32	10	11	20.73324	20.836938	0.103696
4	9	12	11	11	12	12	16	16	10	22	20.55134	20.906414	0.355071
5	19	30	27	15	22	19	16	21	31	18	58.45886	59.02069	0.561833
6	29	15	23	25	13	32	17	14	11	8	5.981925	5.987078	0.005153
7	17	22	16	26	30	23	15	31	18	30	55.56245	55.547552	-0.014897
8	29	23	23	30	10	22	16	31	18	29	31.52884	31.443187	-0.085655
9	23	16	15	31	28	28	24	8	20	25	11.22512	11.082254	-0.142871
10	29	20	21	23	32	17	14	8	21	20	11.20543	11.081993	-0.123438
11	11	21	22	22	12	15	25	16	13	21	36.91202	37.570486	0.658464
12	28	13	13	24	12	27	12	10	30	14	22.67044	22.880391	0.209947
13	20	17	14	22	15	8	11	15	11	21	16.35159	16.105621	-0.24597
14	20	21	21	16	12	11	29	32	17	32	33.87208	34.053531	0.181455
15	27	20	21	10	19	13	29	25	18	9	12.01716	12.014605	-0.002551
16	32	15	25	24	8	8	9	8	21	11	9.187271	9.280194	0.092923
17	26	23	12	11	22	29	13	30	26	12	29.20325	29.517037	0.313784
18	20	29	16	28	31	13	25	20	22	14	40.35284	40.808381	0.455539
19	22	18	9	17	23	20	30	25	8	16	8.961997	9.005713	0.043716
20	11	25	27	19	24	14	8	29	31	9	9.633909	9.774141	0.140233

Table 3.1 Examples of Random Verification (N=1024)

We had compared 10000 wordlength sets for 1024-point FFT of Radix-2 and Radix-2^2 algorithm. The maximum difference of Radix-2 is almost within ±1dB. The maximum difference of Radix-2^2 for each FFT is almost within ±1.1dB. Fig. 3.7(a) shows the distribution of difference of 1024-point Radix-2 FFT, Fig. 3.7(b) shows the 1024-point Radix-2^2 FFT.



(a)

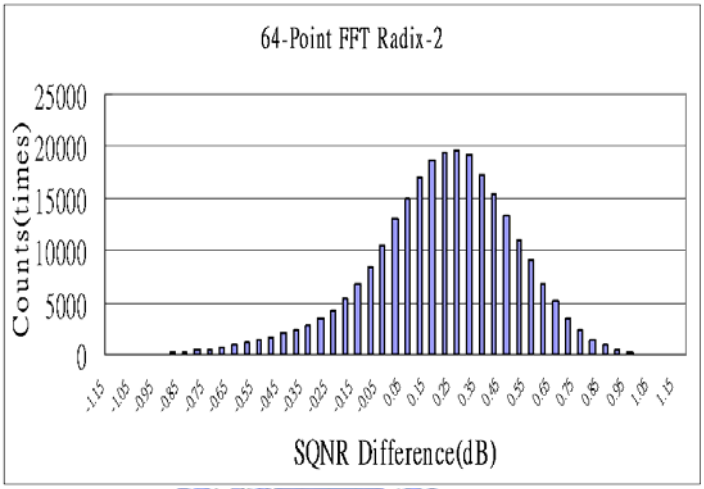


(b)

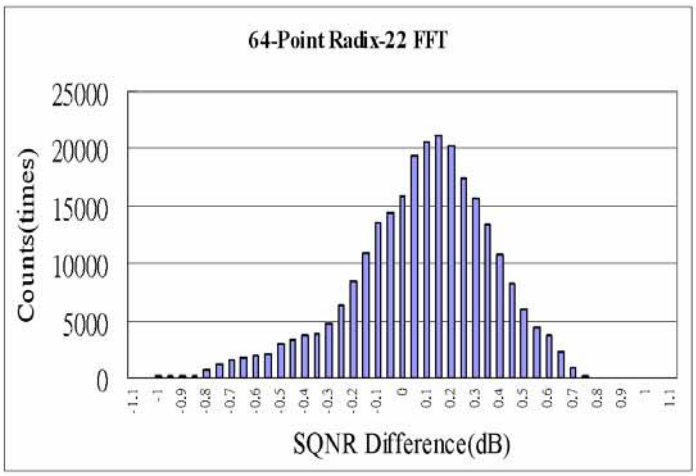
Figure 3.7 Results of Random Verification of Radix-2 and Radix2^2

3.4,2 Partial Exhaustive Verification

To exhaustively compare all wordlength sets of 8 to 32 bits is not practical because the simulation time is not endurable. However we can do exhaustive comparison in some special range, maybe some of the solution space, to verify. We had chosen the wordlength 11 to 18 bits to do partially exhaustive comparison of 64 points DIF Radix-2 and Radix-2^2 FFT. They spent about 130 hours comparison time, and the results are shown in Figure 3.8. The difference is within ± 1.1 dB.



(a)



(b)

Figure 3.8 Results of Partial Exhaustive Verification of Radix-2 and Radix2^2

Section 3.4.1 and 3.4.2 clearly show that the result obtained from the statistical error model can be very close to that obtained from the simulation-based approach.

Chapter 4

Wordlength Optimization

The wordlength is an important design parameter. It will affect both the performance and complexity. Longer wordlength is preferred for good precision. But, increase wordlength will increase the complexity. It will increase the size of memory and computational units and thereby increase power consumption and decrease performance. Hence, the wordlength requires careful optimization.

In this chapter, we will briefly review the design flow of FFT processor first. Then, we will describe our approach, hybrid wordlength optimization method. Finally, two examples are shown.

4.1 FFT Processor Design Flow

There are many factors have to be considered o design the FFT processor. Figure 4.1 shows the over all design flow of FFT processor. First, system requirements need to be specified. They are points of FFT, SQNR, throughput, area, power, …, etc. Then, the proper FFT algorithm and FFT architecture need to be chosen. Finally, the wordlength of architecture need to be analyzed.

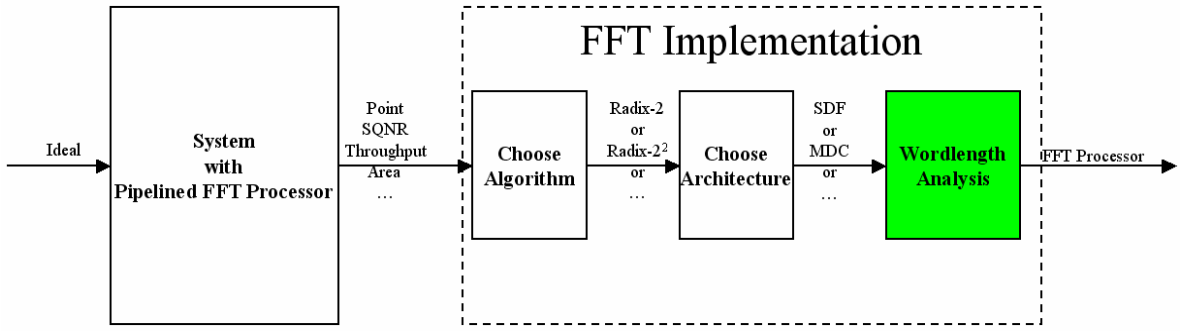


Figure 4.1 Design Flow of FFT Processors

When the FFT is implemented as a fully custom ASIC, the wordlength of each stage can be freely chosen except input and output wordlengths of FFT processor, which are system specified. Internal wordlengths of FFT processor can be chosen to decide the precision and complexity. In general, longer wordlength is preferred for better precision of numbers. On the other hand, increase the wordlength will increase the complexity, it will increase the hardware cost, power consumption, and decrease the speed. Thereby, the optimization is a trade-off between precision and complexity.

To reduce the time of over all system design, the automatic wordlength optimization solution is preferred. A simulation-based method on pipelined FFT had presented by Lin [3]. We will present a faster hybrid method in this thesis. Figure 4.2 outlines the automation flow. There are four steps in sequence, i.e., upper bound wordlength evaluation, lower bound wordlength evaluation, optimized wordlength candidate searching, and optimized wordlength selection. Additionally, there are some tables and libraries built offline to speed up this flow.

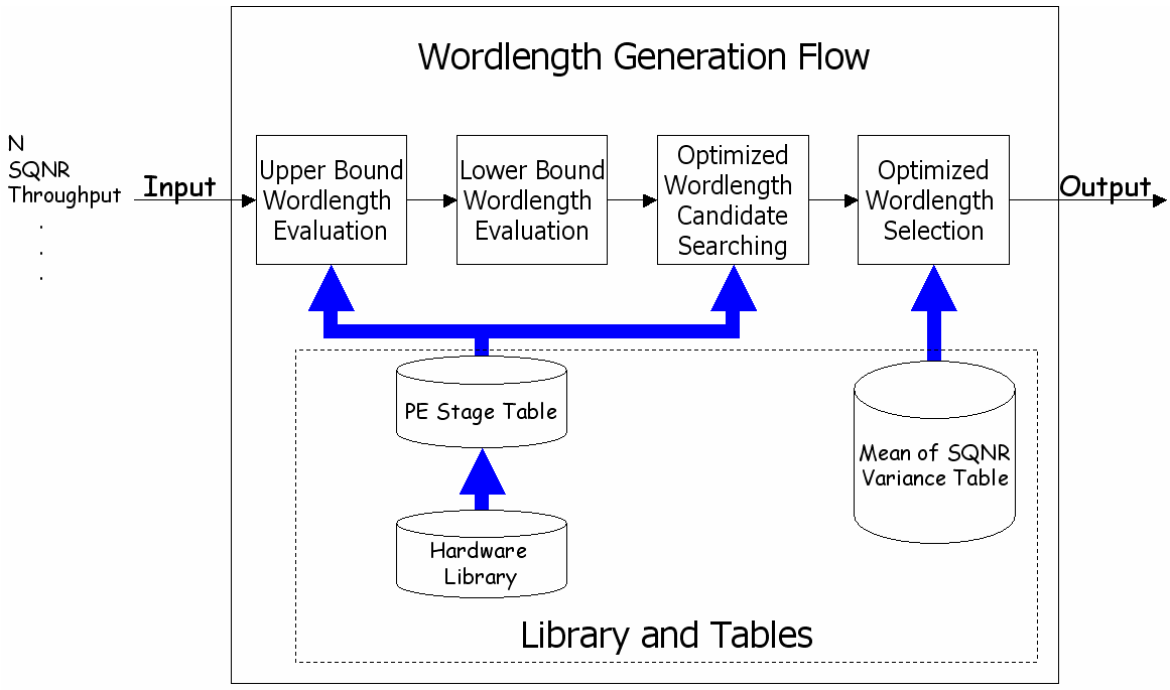


Figure 4.2 Over All Flow of Wordlength Optimization

4.2 Wordlength Generation

Items in Fig. 4.2 will be introduced in this section. This flow is to optimize the area under input constraints. Input constraints include points of FFT, SQNR, throughput, FFT input and output wordlength, SQNR simulation confidence interval, and SQNR simulation error. The output data are wordlengths of each PE stage.

4.2.1 Library and Table

Since we optimize hardware cost, the relative hardware library needs to be chosen. Adder, multiplier, multiplexer, read only memory (ROM), and shift register are five basic elements of FFT. Hardware library decides the area and critical path to wordlength table for these components [3].

PE stages are hardware blocks in the wordlength generation flow, which is built by

the basic components. We need a table that stores the information of area and critical path for each PE stage to speed up the automation flow, PE stage table [3].

In Figure 4.2, the mean of SQNR variance table is used to calculate the simulation times of different confidents of simulation [3].

4.2.2 Upper Bound Wordlength Evaluation

Throughput is one of the input constraints. Satisfy the throughput constraint implies that the critical path must be short enough to meet equation (4.1). In other words, it means that some stages violate the timing of pipeline if there are critical paths greater then $\frac{1}{throughput}$.

$$critical \quad path < \frac{1}{throughput} \tag{4.1}$$

The upper bound wordlength(UBW) is defined as the largest possible wordlength such that the critical path of the corresponding PE stage satisfies equation (4.1). And, the upper bound wordlength set (**UBW**) is defined as a set which includes all wordlength of PE stages and each wordlength is UBW. Note that we use bold print to denote a set and light print to denote the element in a set. For example, if the **UBW** of 1024-point FFT (10 PE stages) is {14 15 15 16 17 18 18 18 19 20} then the UBW of stage 1 (UBW_1) is 14, UBW_2 is 15, UBW_3 is 15, and so on.

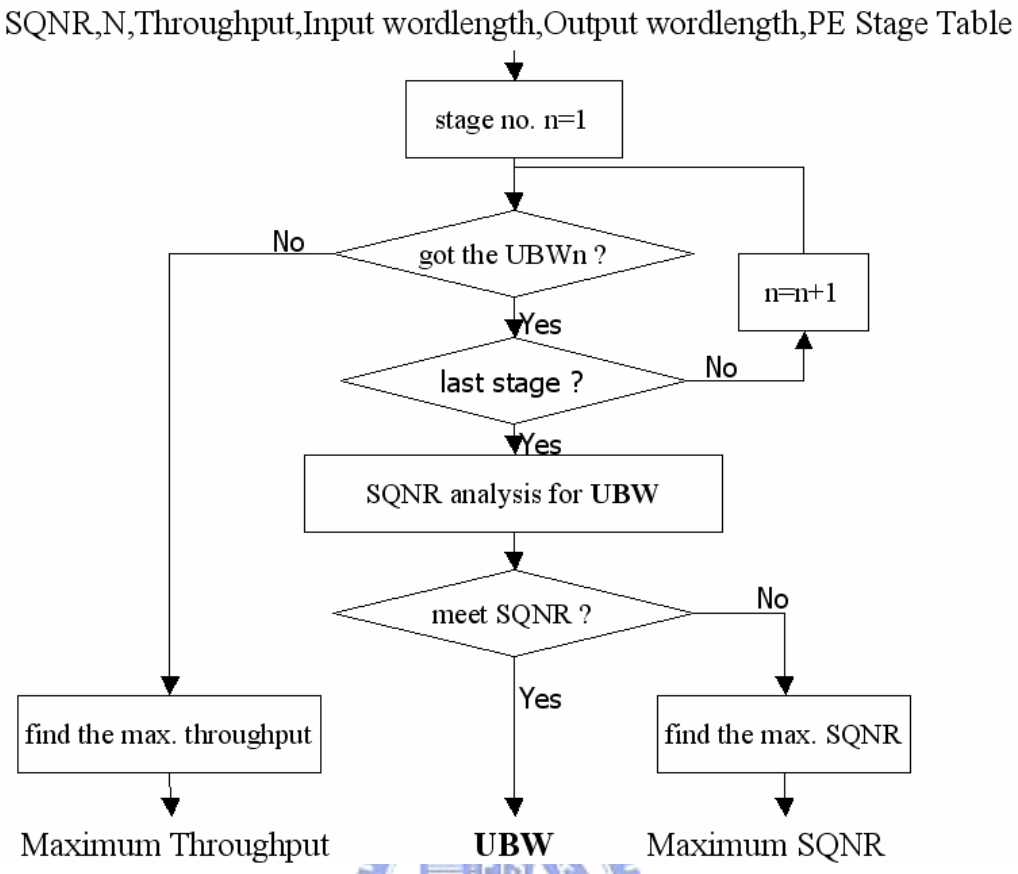


Figure 4.3 Flow of Upper Bound Wordlength Evaluation

Fig. 4.3 shows the flow of **UBW** evaluation. There are three conditions to stop the evaluation. Condition 1, the **UBW** is founded if SQNR and throughput constraints are both met. Condition 2, the optimization is failed if the SQNR constraint can't be met. The maximum possible SQNR will be reported before stop. Condition 3, the optimization is failed if throughput constraint can't be met. The maximum possible throughput will be proposed before stop.

4.2.3 Lower Bound Wordlength Evaluation

The lower bound wordlength (LBW) is defined such that if any wordlength of PE stage is equal to LBW, the SQNR of new set is just small than the SQNR of input constraint. The lower bound wordlength set **(LBW)** is defined as $\{LBW_x \mid x \in N, 1 \le x \le n\}$,

x means the *x*th PE stage. Based on the definition of **LBW,** it is easy to see that SQNR of **LBW** is small then the SQNR of input constraint.

Fig. 4.4 shows the flow of **LBW** evaluation. The input are N (point of FFT), SQNR, input and output wordlength, and **UBW**. Then, the output is **LBW**

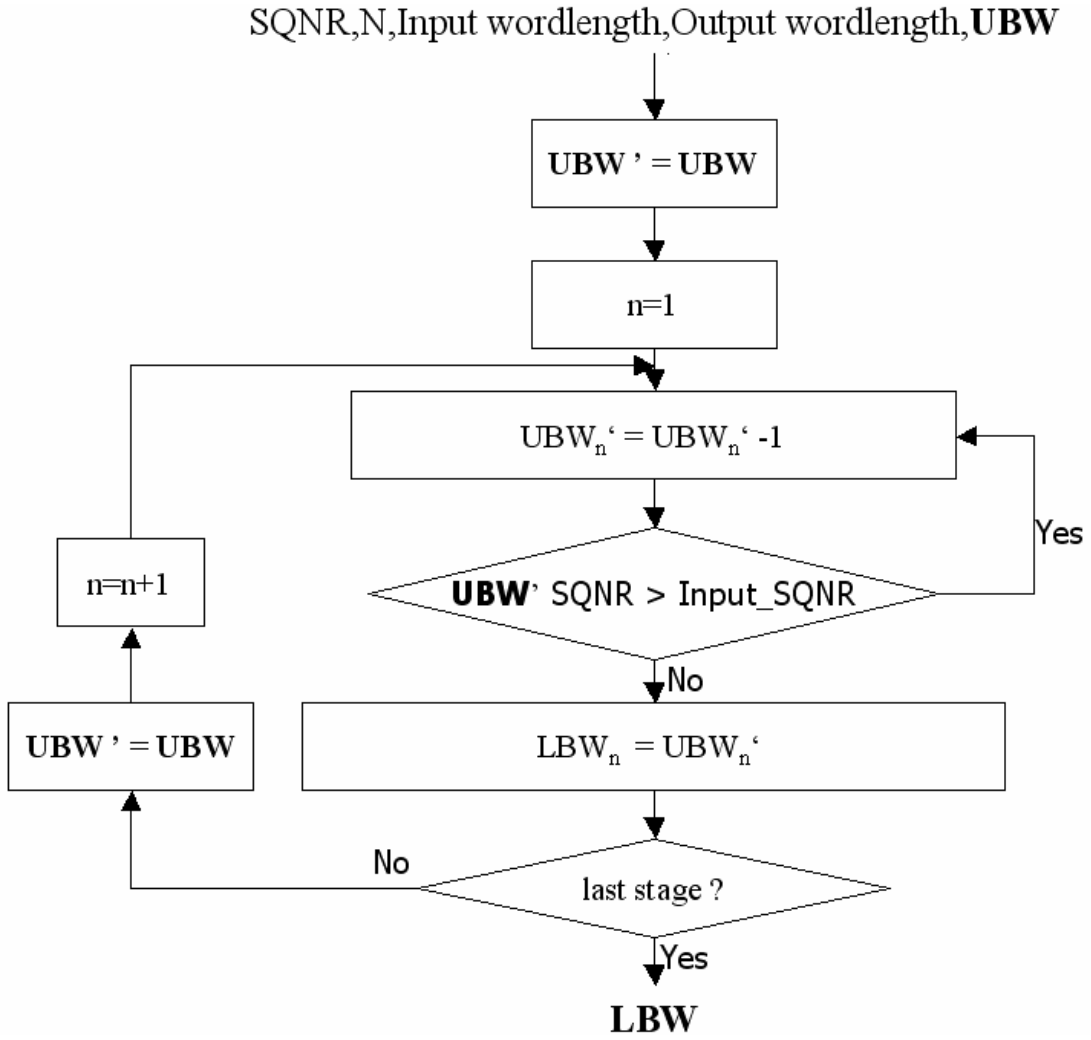


Figure 4.4 Flow of Lower Bound Wordlength Evaluation

Fig. 4.5 shows an example of **LBW** evaluation. Where the iSQNR is the input SQNR constraint. The step of Fig. 4.5 is top to bottom and left to right. The arrow shows the detail steps. And the more than,“>”, and small than, “<”, mean the comparison results between SQNR of statistical error analysis and SQNR of input constraint.

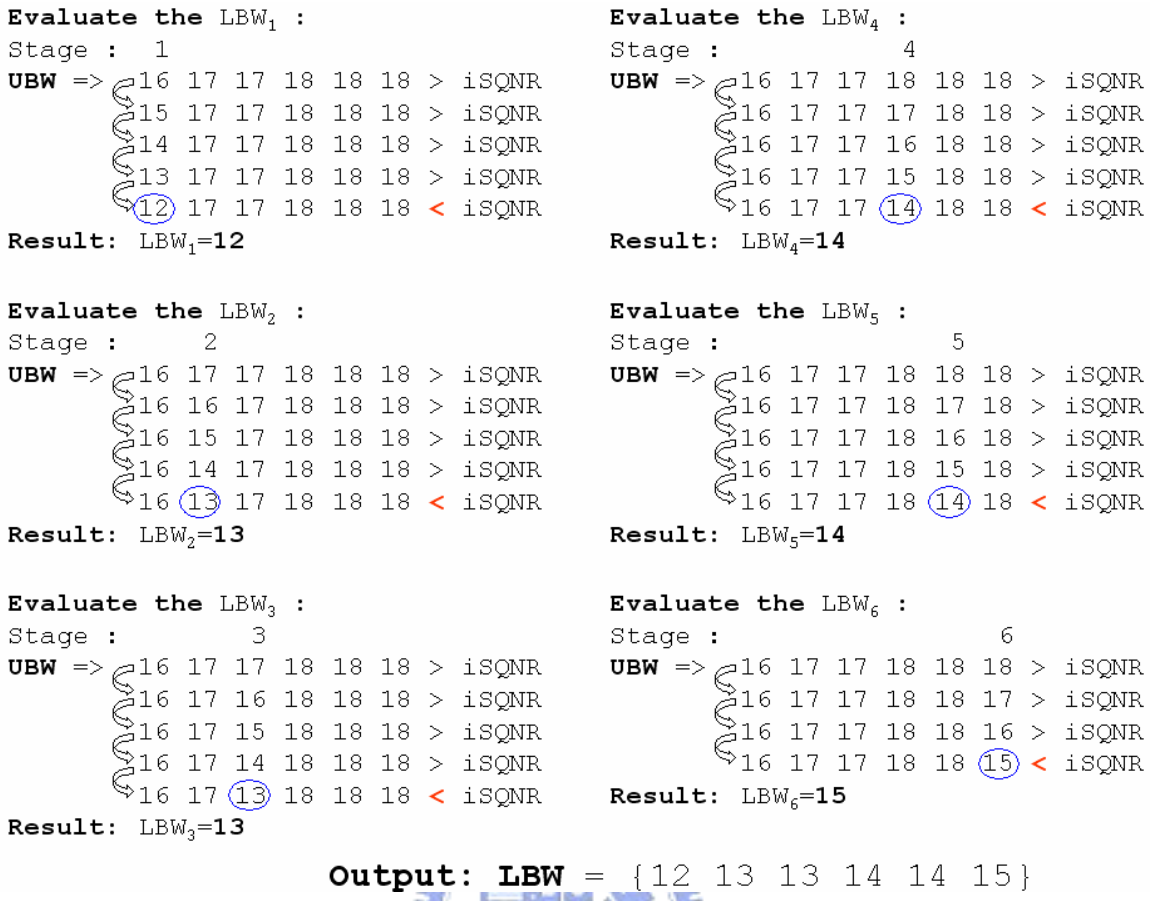


Figure 4.5 Example of Lower Bound Wordlength Evaluation (N=64)

4.2.4 Optimized Wordlength Candidate (OWC) Searching

4.2.4.1 Optimization Format

Since the FFT processor uses large memories especially in the early stages. Figure 4.6 shows the area increment of each PE stage when the wordlength of each stage was added by 1 bit. Therefore, to keep the wordlength short in the early stages is a good choice for area optimization.

The property of output SQNR of pipeline FFT processor is shown in equation (4.2).

$$SQNR \approx 10\log_{10}\frac{a_k}{(a_1 2^{-2b_1+1} + a_2 2^{-2b_2+2} + \cdots + a_n 2^{-2b_n+n})} \quad (4.2)$$

where a_n is constant of PE stage n, b_n are wordlength of PE stage n. It is easy to see that if there exists one $x, x \in N, 1 \le x \le n$ such that $a_x 2^{b_x+x} >> a_m 2^{b_m+m}$ $m \in N, 1 \le m \le n, m \ne x$, then, the PE stage x will be the bottleneck of output SQNR. In the other word, the value of $(a_1 2^{-2b_1+1} + a_2 2^{-2b_2+2} + \cdots + a_n 2^{-2b_n+n})$ will be dominated by $a_x 2^{b_x+x}$. So, the wordlength of each stage is efficient when they are close.

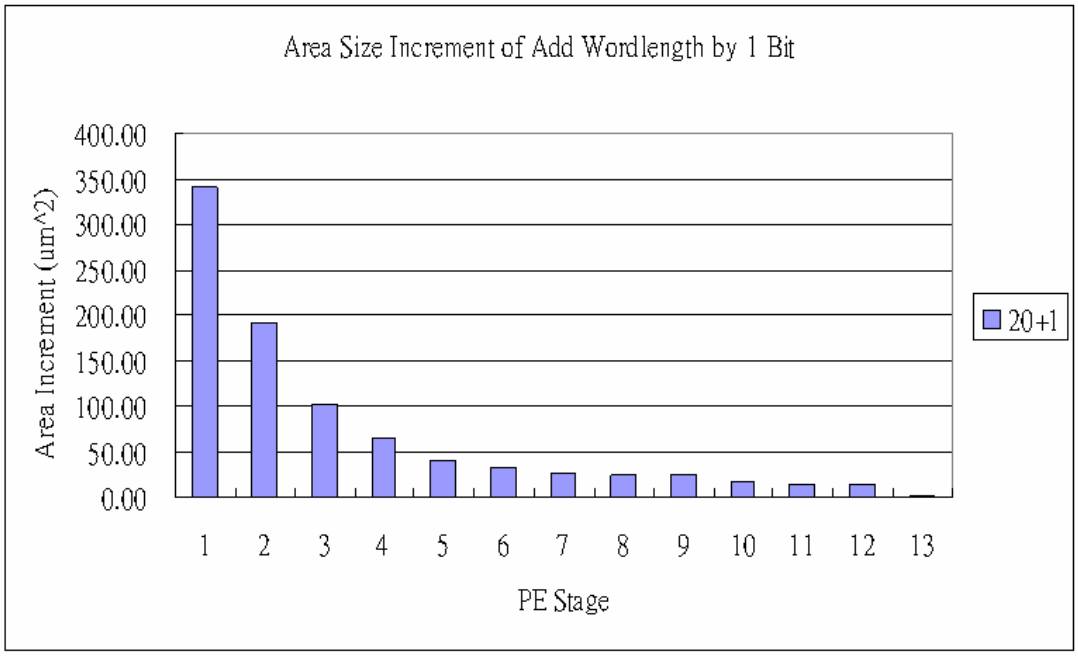


Figure 4.6 Area Increment of Add Wordlength 1 Bit of Each Stage (N=8192)

Due to upon properties the expected optimization wordlength set will be sorted in ascending order from stage 1 to stage n, and the wordlength is closed stage by stage. {11 11 12 13 13 14} and {14 14 14 14 15 16} for examples. We refer these schemes of wordlegth set as *optimization format* for simplicity in the remaining section.

4.2.4.2 OWC Searching Flow

The optimized wordlength set candidates (**OWC**) have three properties. (1) It is between **LPW** and **UBW**. (2) It is in optimization format. (3) The SQNR of FFT processor meets the input SQNR constraint when the wordlength scheme is the same as that of any

OWC.

To search the **OWC**, we scan the wordlength set from **LBW** to **UBW** and compare SQNR of each set with the input SQNR constraint. Figure 4.7 shows the flow of **OWC** searching. The output of this flow is the **OWC Array**. It contains all the information of **OWC** and is sorted by area size.

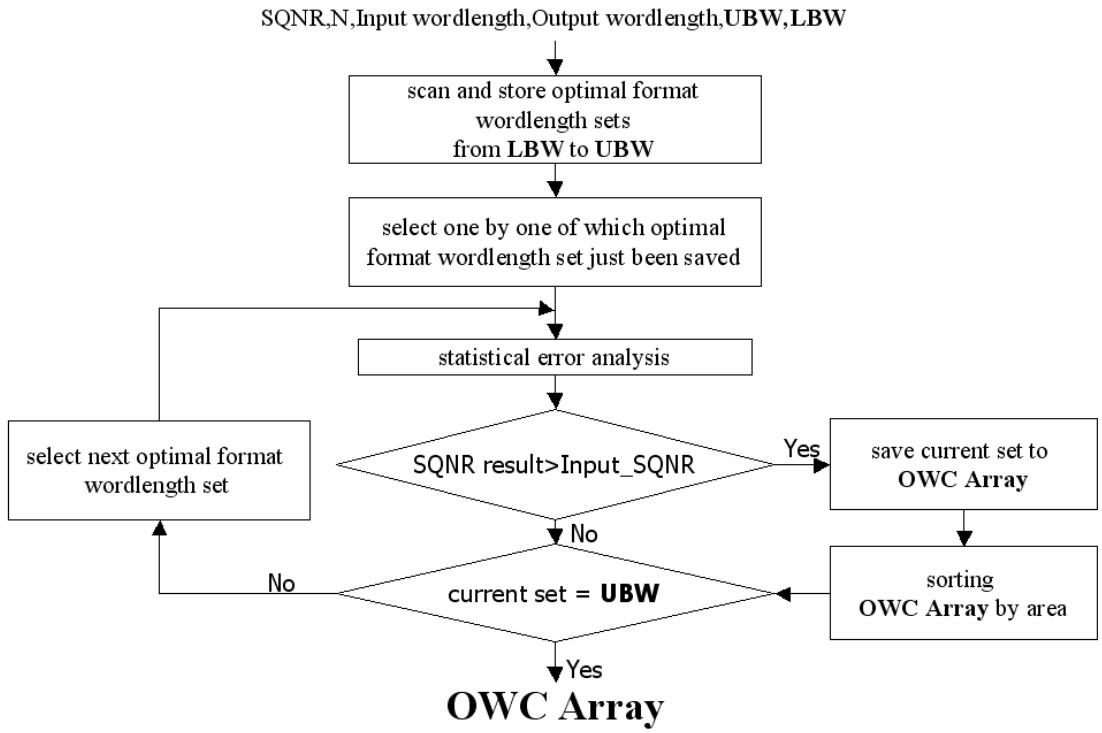


Figure 4.7 Flow of Optimized Wordlength Candidate Searching

4.2.5 Optimized Wordlength (OW) Selection

The **OW** is an **OWC** which has the smallest area size and good SQNR. There are two methods to get the optimized wordlength in **OWC Array**. Method 1, the optimized wordlength set will be found by simulation-based method if user's SQNR error constraint is under ± 1 dB. Method 2, the optimized wordlength set will be found by statistical method if users SQNR error constraint is more than ± 1 dB.

Figure 4.8 shows the flow of **OW** selection. In Method 1, we simulate all **OWC** of **OWC Array** one by one from the one with the smallest area size until the SQNR of simulation meets the SQNR of the input constraint. In Method 2, we judge all the OWC in OWC Array by a benefit function to get the OWC with the best benefit. The benefit function is shown in equation (4.3).

$$Benefit = \frac{SQNR\ increament}{area\ size\ increament} \tag{4.3}$$

where the increment is the difference between the SQNR or area size of **LBW** and those of **OWC**.

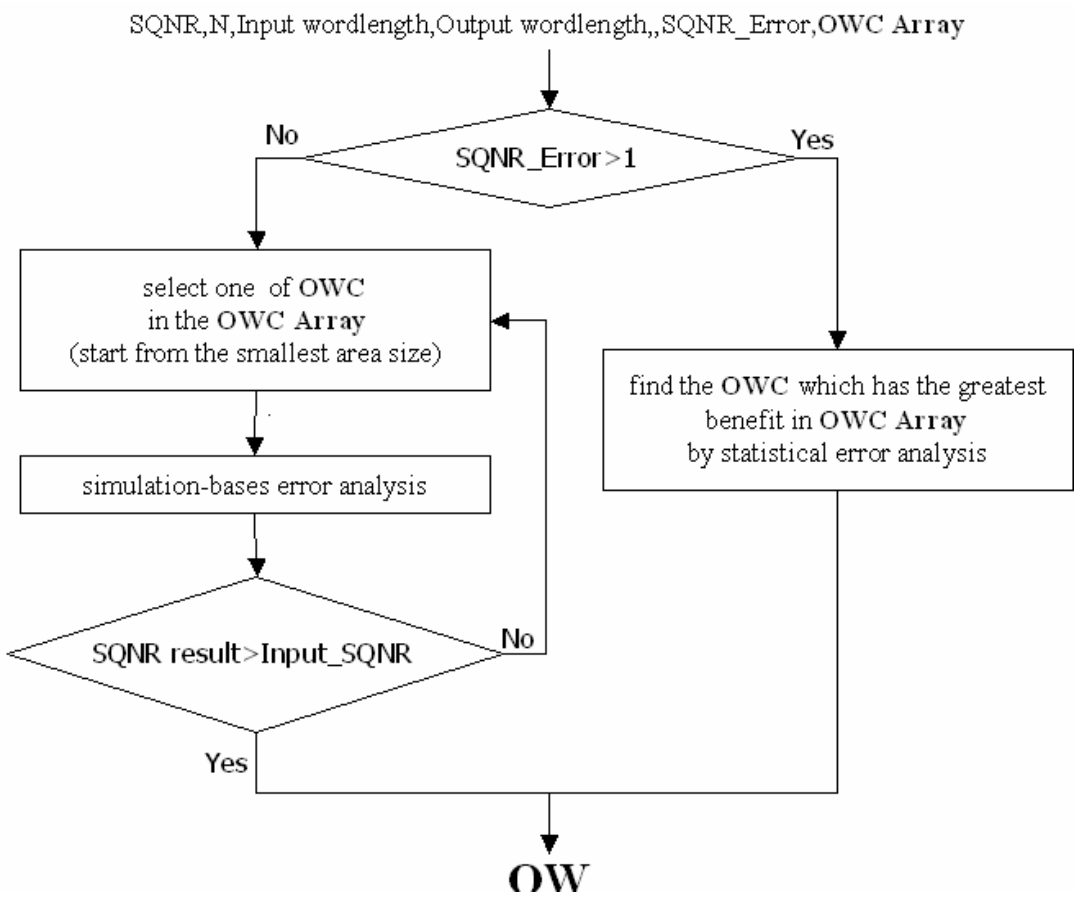


Figure 4.8 Flow of Optimized Wordlength Selection

4.3 Examples of Wordlength Optimization

4.3.1 Hybrid Method

Input constraints of this example are {N=1024(n=10), SQNR=45 dB, input_wordlength=output_wordlength=18, throughput=50MHz, and SQNR_error=0.1 dB}. Since the SQNR_error constraint is smaller than ±1 dB, the hybrid method will be used. Figure 4.9 shows the steps of this example. The "sim_SQNR" means the result of simulation and the "iSQNR" means the SQNR of input constraint.

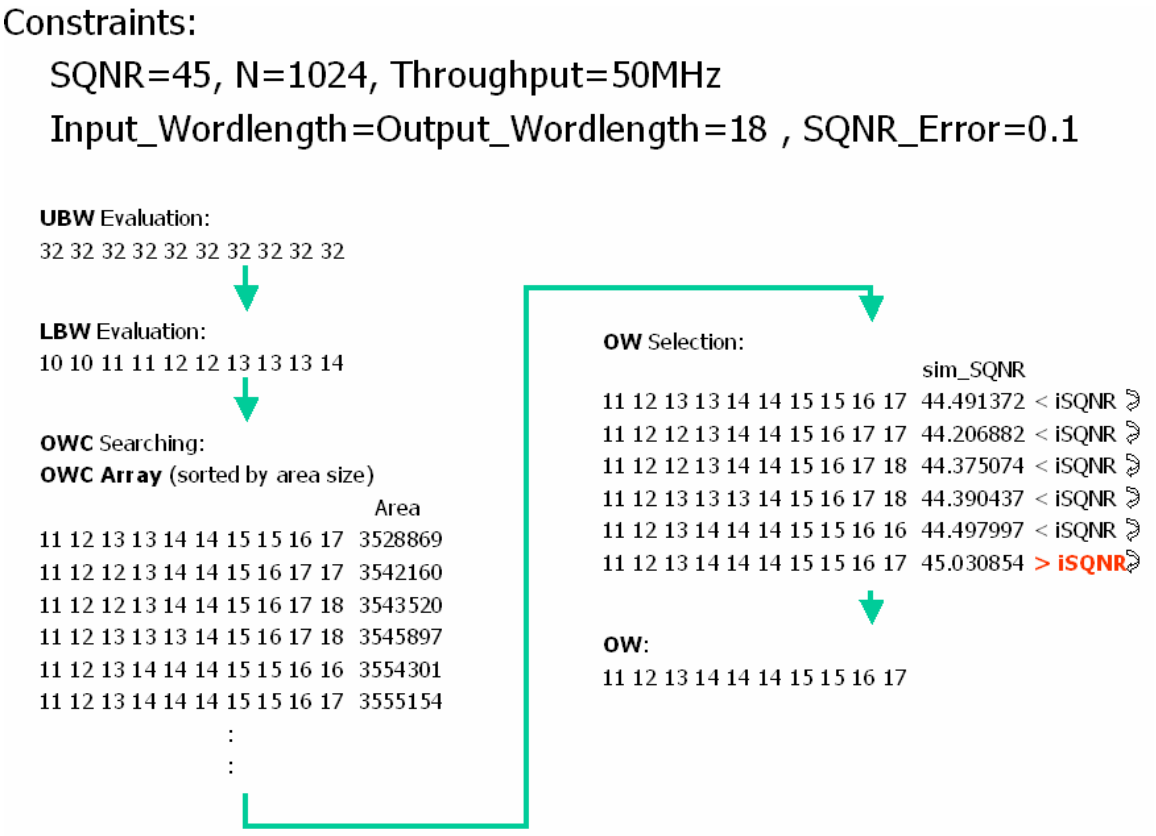


Figure 4.9 Example of Hybrid Wordlength Optimization Method

4.3.2 Pure Statistical Method

Input constraints of this example are {N=1024 (n=10), SQNR=45 dB, input_wordlength=output_wordlength=18, throughput=50MHz , and SQNR_error=1.1 dB}. Since the SQNR_error constraint is more than ±1 dB the pure statistical method will be used. Figure 4.10 shows the steps of this example.

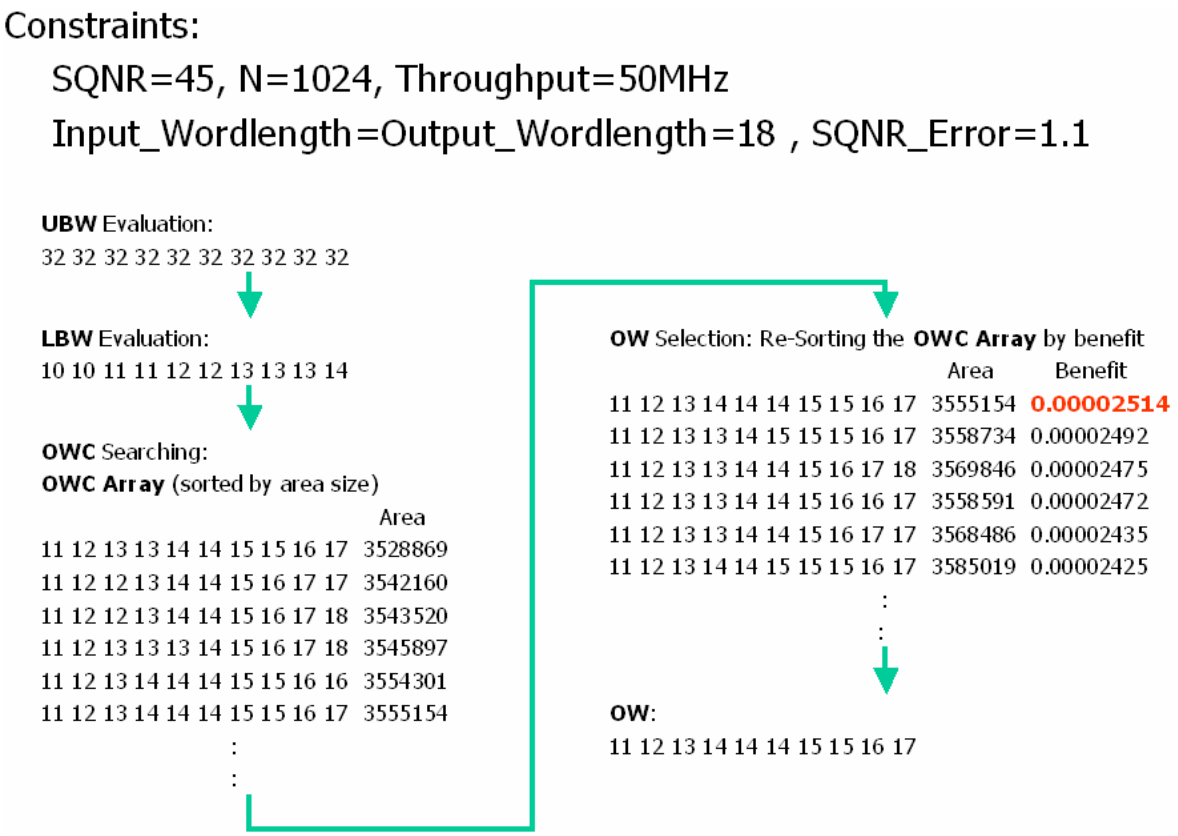


Figure 4.10 Example of Pure Statistical Method

Chapter 5

Experimental Results

5.1 Introduction

We implement two FFT architectures, including DIF R2SDF and DIF $R2^2$ SDF. The range of N can be adjusted from 8 to 8192 points, and wordlength from 8 to 32 bits in each stage. We pipe each PE stage of FFT architectures and apply stage-by-stage scaling.

In order to compare the performance with previous work [3], the same hardware libraries are used here.

Logic gate model includes adder, multiplier, and multiplexer. We conduct synthesis without any constraints by Synopsys Design Analyzer [19] and the TSMC 0.25um cell library and Synopsys DesignWare [18] are used. The fast carry look-ahead synthesis model for adder, Booth-encoded Wallace tree synthesis model for multiplier, and universal multiplexer synthesis model for multiplexer are adopted and area and timing reports of Synopsys Design Analyzer are used for these models. Memory model includes shift register and ROM also use TSMC 0.25um cell library.

The SQNR range between 40 to 60 dB had been used in most system. It is for our experimentations too. Two common FFT design specifications that are typically used in OFDM systems [22] had been summarized in Table 5.1.

	Size	Operating freq.	I/O
Short Length	16-256	50MHz	Complex, word-sequential
Long Length	256-8192	20MHz	Complex, word-sequential

Table 5.1 Specification of Common FFT for OFDM

To implement the proposed flow, the C++ language with SystemC library is used. The SystemC library is used for fixed-point type to model the behavior of fixed-point hardware. The quantization mode is always truncation (SC_TRN) and the overflow mode is saturation (SC_SAT) in our experimentations.

Finally, the platform is built in a PC with Intel 2.4GHz CPU and 768M Memory. The operation system is Microsoft Windows 2000. The Visual C++ 6.0 is used for compiler.

5.2 Results

The experimental results of R2SDF and R2^{2}SDF wordlength optimization will be showed in this section.

5.2.1 Optimization of Different Constraint

Results of experiments with different constraints will be introduced in this sub-section.

5.2.1.1 FFT Point Constraint

Experimental result of area optimization for point from 8 points to 8192 points is presented in Table 5.2. Table 5.2(a) is for DIF R2SDF and Table 5.2(b) is for DIF R2^{2}SDF. Constraints include: SQNR is 45(dB), SQNR error is 0.1(dB), SQNR simulation confidence interval is at the level of 95%, the throughput is 50MHz, and the input and output wordlengths are 18 (bits). Since the constraint of maximum allowable SQNR error is small then 1 dB, the hybrid method will be used. In these tables, the first column “Point”

presents the point of FFT processor. The column of “Pre-Post” represents that parameters in the row with “Pre” belong to traditional design, without optimization, or parameters in the row with “Post” are optimized.

R2SDF Constraints: SQNR=45 dB, SQNR_error=0.1 dB, **IO=18 bits**, Throughput=50 MHz

Point	Pre-Post	PE Stage WordLength	Area (μm^2)	Area Reduction	Time (sec)
		1 2 3 4 5 6 7 8 9 10 11 12 13			
8	Pre	12 12 12	258831	16%	8
	Post	11 11 12	216184		
16	Pre	12 12 12 12	402693	11%	11
	Post	11 11 12 13	359378		
32	Pre	13 13 13 13 13	733434	9%	18
	Post	11 12 12 13 13	664339		
64	Pre	14 14 14 14 14 14	1152506	14%	20
	Post	11 12 13 13 13 14	993360		
128	Pre	14 14 14 14 14 14 14	1550048	10%	41
	Post	11 12 13 13 14 14 15	1388215		
256	Pre	15 15 15 15 15 15 15 15	2249617	16%	46
	Post	11 12 13 13 14 14 15 15	1882859		
512	Pre	15 15 15 15 15 15 15 15 15	2988883	14%	67
	Post	11 12 13 13 14 15 15 15 16	2569228		
1024	Pre	16 16 16 16 16 16 16 16 16 16	4474445	20%	139
	Post	11 12 13 13 14 14 15 16 17 17	3568487		
2048	Pre	16 16 16 16 16 16 16 16 16 16 16	6396581	19%	310
	Post	11 12 13 14 14 15 15 15 16 17 18	5184678		
4096	Pre	17 17 17 17 17 17 17 17 17 17 17 17	10255423	23%	616
	Post	11 12 13 13 14 15 15 16 17 17 17 18	7858489		
8192	Pre	17 17 17 17 17 17 17 17 17 17 17 17 17	16668224	24%	971
	Post	11 12 13 13 14 15 15 16 17 17 18 18 19	12676846		

(a)

R2^{2}SDF Constraints: SQNR=45 dB, SQNR_error=0.1 dB, **IO=18 bits**, Throughput=50 MHz

Point	Pre-Post	PE Stage WordLength	Area (μm^2)	Area Reduction	Time (sec)
		1 2 3 4 5 6 7 8 9 10 11 12			
16	Pre	12 12 12 12	214156	11%	10
	Post	11 11 12 13	189706		
64	Pre	13 13 13 13 13 13	760033	6%	20
	Post	11 12 12 13 13 14	717914		
256	Pre	15 15 15 15 15 15 15 15	1732185	16%	31
	Post	11 12 12 13 13 14 15 16	1461204		
1024	Pre	16 16 16 16 16 16 16 16 16 16	3695209	20%	65
	Post	11 12 13 13 14 14 15 15 16 17	2973710		
4096	Pre	17 17 17 17 17 17 17 17 17 17 17 17	9271212	23%	317
	Post	11 12 12 13 14 14 15 16 17 18 18 18	7120428		

(b)

Table 5.2 Area Optimization of Different FFT Point (IO Wordlength=18)

The column of “Area Reduction” presents the reduction rate of area, calculated by $\frac{pre\_area - post\_area}{pre\_area} \times 100\%$. The last column “Time” shows the computer time of optimization. It can be see that the greater N with the greater area reduction rate, generally. The maximum and minimum area reduction rates for DIF R2SDF are 24% and 9% and those are 23% and 6% for DIF R2^2SDF.

5.2.1.2 Input Wordlength and Output Wordlength

Table 5.3 introduces the experimental results with different input and output wordlength constraints to those of Table 5.2. The input wordlength is 14 bits and the output wordlength is 14 bits. The area reduction rate is still the same when point range in 8 to 1024. There is no solution when the point number is greater than 1024.

R2SDF Constraints: SQNR=45 dB, SQNR_error=0.1 dB, **IO=14 bits**, Throughput=50 MHz

Point	Pre-Post	PE Stage WordLength (1 2 3 4 5 6 7 8 9 10 11 12 13)	Area (μm^2)	Area Reduction	Time (sec)
8	Pre	12 12 12	258831	16%	8
	Post	11 11 12	216184		
16	Pre	12 12 12 12	402693	11%	11
	Post	11 11 12 13	359378		
32	Pre	13 13 13 13 13	733434	9%	22
	Post	11 12 12 13 13	664339		
64	Pre	14 14 14 14 14 14	1152506	14%	20
	Post	11 12 13 13 13 14	993360		
128	Pre	14 14 14 14 14 14 14	1550048	10%	35
	Post	11 12 13 13 14 14 15	1388215		
256	Pre	15 15 15 15 15 15 15 15	2249617	15%	46
	Post	11 12 13 13 14 15 15 16	1905692		
512	Pre	16 16 16 16 16 16 16 16 16	3301662	19%	68
	Post	12 12 13 14 14 15 15 16 17	2675873		
1024	Pre	19 19 19 19 19 19 19 19 19 19	5314581	20%	136
	Post	14 15 15 15 16 17 17 17 18 19	4250590		
2048	Pre	No solution			
	Post	No solution			
4096	Pre	No solution			
	Post	No solution			
8192	Pre	No solution			
	Post	No solution			

(a)

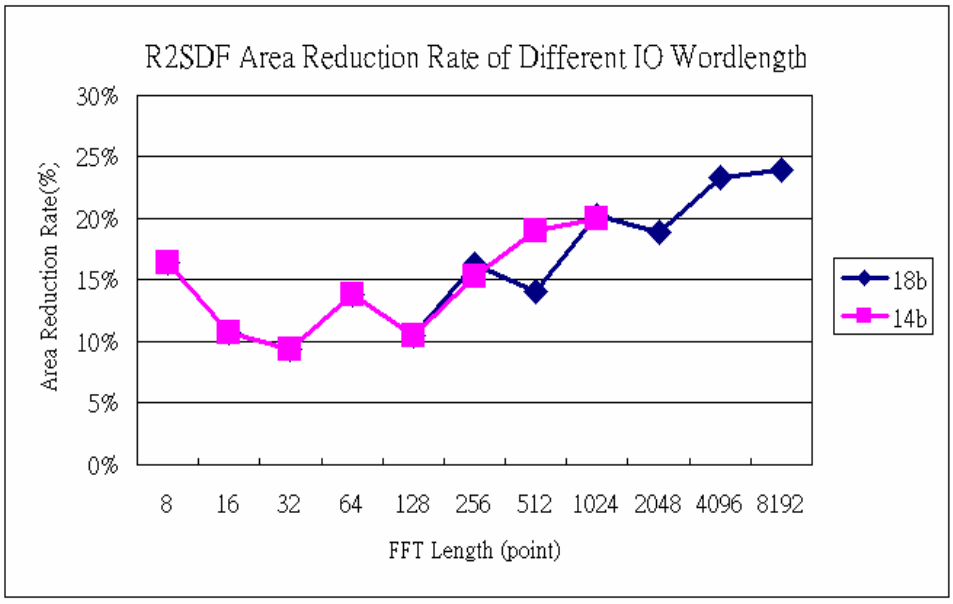
R2²SDF Constraints: SQNR=45 dB, SQNR_error=0.1 dB, **IO=14 bits**, Throughput=50 MHz

Point	Pre-Post	PE Stage WordLength												Area (μm^2)	Area Reduction	Time (sec)
		1	2	3	4	5	6	7	8	9	10	11	12			
16	Pre	12	12	12	12									214156	11%	10
	Post	11	11	12	13									189706		
64	Pre	13	13	13	13	13	13							760033	6%	20
	Post	11	12	12	13	13	14							717914		
256	Pre	15	15	15	15	15	15	15	15					1732185	15%	37
	Post	11	12	13	13	14	14	15	16					1474260		
1024	Pre	18	18	18	18	18	18	18	18	18	18			4078142	15%	28
	Post	13	14	15	15	16	17	17	18	19	19			3448117		
4096	Pre	No Solution														
	Post	No Solution														

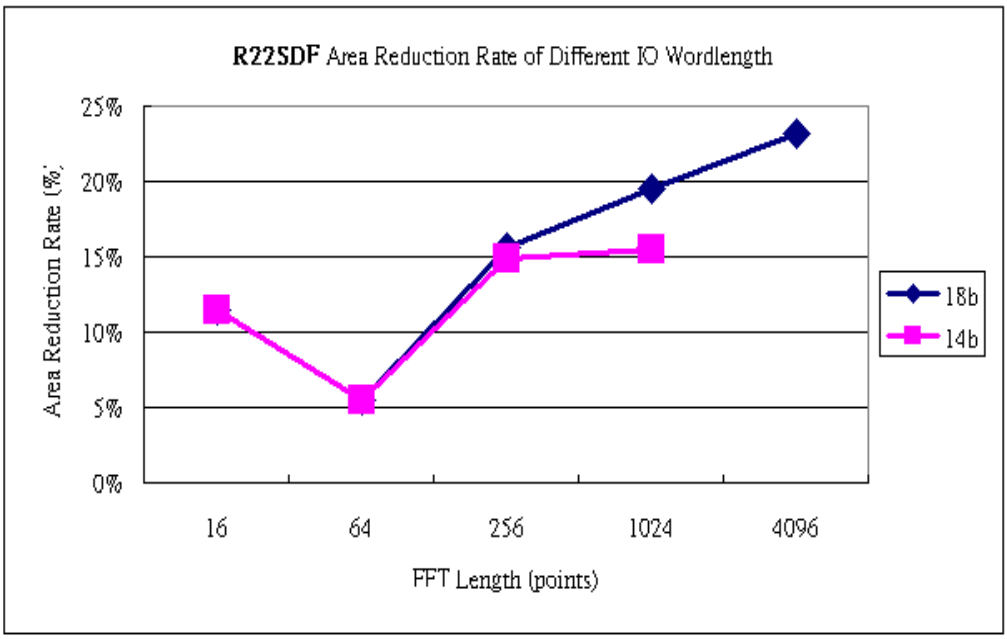
(b)

Table 5.3 Area Optimization of Different FFT Point (IO Wordlength=14)

Figure 5.1 shows the difference of area reduction rate between these two input and output wordlengths.



(a)



(b)

Figure 5.1 Area Reduction Rate of IO Wordlength=18 and 14 Bits

5.2.1.3 SQNR

Figure 5.3 presents the area reduction rate for different SQNR constraint of DIF R2SDF and DIF R2^{2}SDF. Constraint of SQNR error is 0.1(dB), SQNR simulation confidence interval is at the level of 95%, the throughput is 50MHz, and the input and output wordlengths are 18.The SQNR of traditional design increases 6 dB if all wordlength increases 1 bit. It can be found that 6 dB is a cycle of area reduction rate for different SQNR constraint, too. The range of area reduction rate is from 12% to 20%.

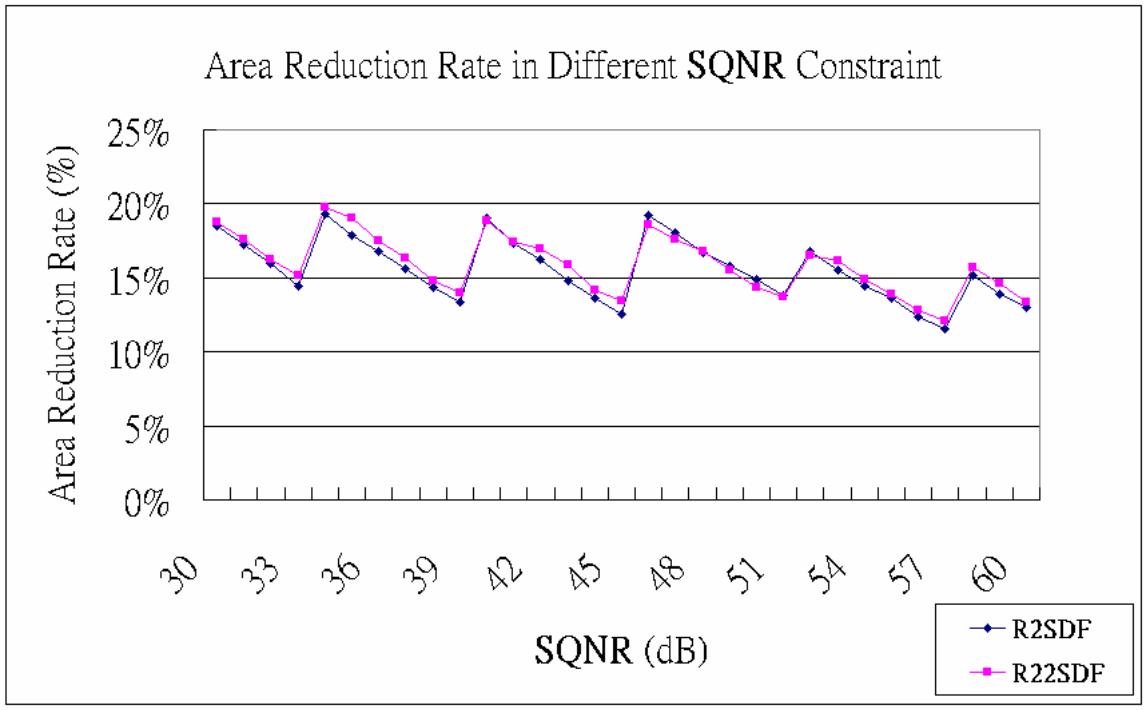


Figure 5.2 Area Reduction Rate vs. SQNR Constraint

5.2.1.4 SQNR Error

Table 5.4 shows the experimental results with the same constraints except SQNR error is 1.1 dB as that in Table 5.2(a). Since the allowable SQNR error is great than 1 dB, the pure statistical error analysis method will be used. The SQNR of these optimized wordlength sets had been verified by simulation based-method for accuracy, introduced in column “Post-SQNR”. The maximum insufficient error of SQNR is 0.18 dB. In other words, it is -0.4% of SQNR constraint.

R2SDF Constraints: SQNR=45 dB, **SQNR_error > 1 dB**, IO=18 bits, Throughput=50 MHz

Point	Pre-Post	PE Stage WordLength (1 2 3 4 5 6 7 8 9 10 11 12 13)	Area (μm^2)	Area Reduction	Post-SQNR (dB) (Simulatiom)	Time (sec)
8	Pre	12 12 12	258831	16%	46.43	1
	Post	11 11 12	216184			
16	Pre	12 12 12 12	402693	11%	45.44	1
	Post	11 11 12 13	359378			
32	Pre	13 13 13 13 13	733434	9%	45.27	1
	Post	11 12 12 13 13	664339			
64	Pre	14 14 14 14 14 14	1152506	14%	45.58	1
	Post	11 12 13 13 13 14	993360			
128	Pre	14 14 14 14 14 14 14	1550048	12%	44.94	1
	Post	11 12 12 13 14 14 15	1370066			
256	Pre	15 15 15 15 15 15 15 15	2249617	16%	45.64	1
	Post	11 12 13 13 14 14 15 16	1884288			
512	Pre	15 15 15 15 15 15 15 15 15	2988883	15%	44.82	1
	Post	11 12 13 13 14 14 15 15 16	2546349			
1024	Pre	16 16 16 16 16 16 16 16 16 16	4474445	21%	45.00	1
	Post	11 12 13 14 14 14 15 15 16 17	3555154			
2048	Pre	16 16 16 16 16 16 16 16 16 16 16	6396581	19%	44.84	1
	Post	11 12 13 13 14 15 15 15 16 17 18	5153721			
4096	Pre	17 17 17 17 17 17 17 17 17 17 17 17	10255423	23%	45.30	2
	Post	11 12 13 13 14 15 15 16 17 17 18 19	7875389			
8192	Pre	17 17 17 17 17 17 17 17 17 17 17 17 17	16668224	24%	45.04	2
	Post	11 12 13 13 14 15 15 16 17 17 18 18 19	12676846			

Table 5.4 Area Optimization of Different FFT Point (SQNR Error = 1.1dB)

5.2.2 Special Cases of Optimization

5.2.2.1 Absolute Constraint Over

There is only one advice for conditions that are scaling down to meet the constraint of hardware library. There are two conditions about these cases. First, the throughput constraint is great then the maximum throughput of hardware library. The maximum throughput of hardware library is the throughput for the wordlength set with the minimum wordlength of hardware library for all stages. If 2 is the minimum wordlength of hardware library, then the {2 2 2 2 2 2 …} is the wordlength set of maximum throughout. Second, the SQNR constraint is great than the maximum SQNR of hardware library. The maximum SQNR of hardware library is the SQNR for the wordlength set with maximum wordlength of hardware library for all stages. If 32 is the maximum wordlength of hardware library then the {32 32 32 32 32 32 …} is the wordlength set of maximum SQNR.

Figure 5.3 shows the output messages. Figure 5.3 (a) is the output message when the related user constraints are N=1024, SQNR=45dB, the input wordlength and output wordlength are 18, and the throughput constraint is 200MHz. The throughput constraint, 200MHz, is over the maximum throughput, 171MHz, of hardware library. Figure 5.3(b) is the output message when the related user constraints are N=1024, SQNR=80dB, the input wordlength and output wordlength are 18, and the throughput constraint is 50MHz. The SQNR constraint, 80dB, is over the maximum SQNR, 69dB, of hardware library.

```
Sorry! Your throughput constraint is over!
Please scale down the throughput from 200 MHz to which under 171 MHz!!
```

(a)

```
Sorry! Your SQNR constraint is over!
Please scale down the SQNR from  80 dB to which under  69 dB!!
```

(b)

Figure 5.3 Output Message of Generator when There is No Solution

5.2.2.2 Partial Constraint Over

This case happens when some constraints are over and all constraints are within hardware library constraints. The proper ranges will be presented for tread-off. Figure 5.4 is the output message when the related user constraints are N=1024, SQNR=68dB, the input wordlength and output wordlength=18, and the throughput constraint is 77MHz. The SQNR constraint, 68dB, with the throughput constraint, 77MHz, can't be met. The output message is to advise user how to trade off.

```
Sorry! Your SQNR constraint  68 dB and throughput constraint  77 MHz are over!
The maximum SQNR of 1024 point FFT in this library is  69 dB!!
1. To meet the throughput the SQNR constraint must be under  48 dB!!
2. To meet the SQNR, the throughput constraint must be under  70 MHz!!
```

Figure 5.4 Output Message of Generator when There is No Solution

5.2.3 Methods Comparison

The area reduction and the computation time of optimization will be compared in this sub-section. First, the comparison between previous work [3] and our hybrid method will be shown. Then, the comparison between our hybrid method and the pure statistical method will be introduced.

5.2.3.1 Previous Work vs. Our Work

The previous work [3] is to optimize wordlength by the pure simulation-based method. And our hybrid method is combined with simulation-based and statistical method. Figure 5.5 presents the post area and computing time of these methods. It shows that results of optimized area of these methods are equally. But the computing time of our method is much faster especially when the FFT length is longer.

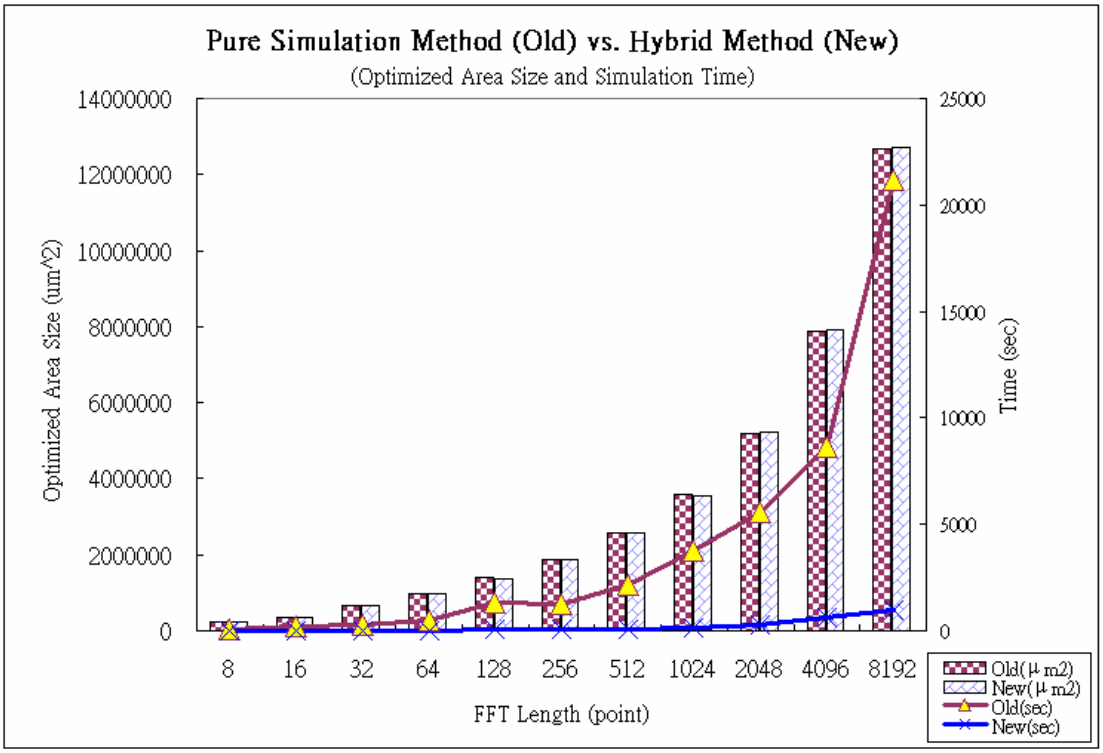


Figure 5.5 Comparison Result between Pure Simulation-Based and Hybrid Method

5.2.3.2 Our Hybrid Method vs. Our Pure Statistical Method

There are two kinds of optimization methods in our work. The hybrid method is the first one, used whenever the allowable maximum SQNR error constraint is less than 1 dB. Second, the pure statistical method is used whenever the allowable maximum SQNR error constraint is greater than 1 dB. The comparison result of these methods is presented in Figure 5.6. It is the figure of the area reduction rate and computing time. It can be found that the area reduction rates of these two method are equally but the computing time of pure statistical method is much faster.

It is interesting to note that the area reduction rate is better when there are insufficient SQNR error occurred in optimizations of 128, 512 and 2048 point FFT, in Table 5.4, of pure statistical method.

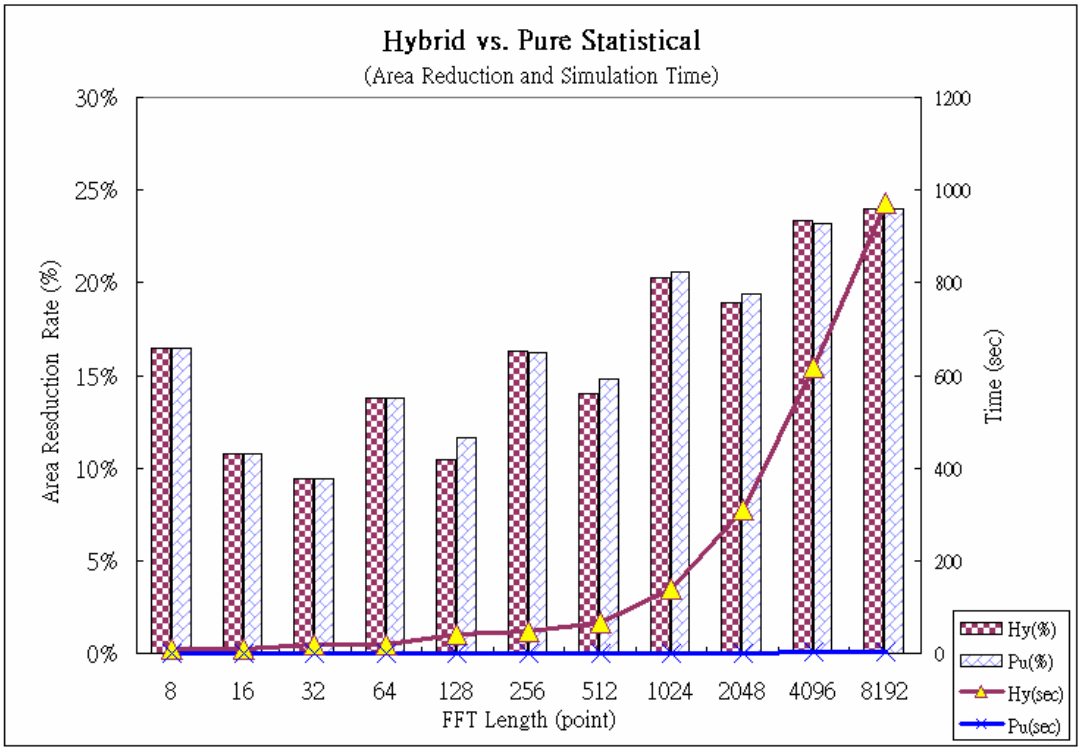


Figure 5.6 Comparison Result between Hybrid and Pure Statistical Method

Chapter 6

Conclusions and Future Works

In this thesis, a statistical error analysis method between SQNR and wordlength of each PE stage of pipelined FFT processors is presented. New hybrid wordlength optimization method on area reduction for pipelined FFT processors based on statistical and simulation-based error analysis is introduced, which is fast then the pure simulation-based method. We also presented a pure statistical wordlength optimization method. It generates the optimized wordlength of FFT processors just in several seconds even the point number of FFT is 8192. With our generator, the advice will still be given even there are no solution under user constraints.

Increase wordlength of FFT processors will increase the power consumption. Therefore, wordlength optimization for power consumption is another attractive topic. Actually, the accuracy of our optimization method depends on the accuracy of the given hardware library. And to build a precise hardware library for area or power is a difficult challenge.

Reference

[1] J. W. Cooley and J. W. Turkey, “An Algorithm for Machine Computation of Complex Fourier Series,” Math. Computation, Vol. 19, pp. 297-301, April 1965.

[2] Oppenheim, Alan V., and Schafer, Ronald W, Discrete Time Signal Processing, Second Edition, Prentice Hall, 1999.

[3] Tson-Yee Lin, On Wordlength Optimization of Pipelined FFT Processors, NCTU, Master Thesis, 2003.

[4] Chao-Kai Chang, Investigation and Design of FFT Core for OFDM Communication Systems, NCTU, Master Thesis, 2002.

[5] P. Duhamel, H. Hollmann, “Split Radix FFT Algorithm,” Electronics Letters, vol. 20, pp. 14-16, January 1984.

[6] L.R. Rabiner and B.Gold, Theory and Application of Digital Signal Processing, Prentice-Hall, Inc., 1975.

[7] E.H. Wold and A.M. Despain, “Pipelined and Parallel-Pipeline FFT Processors for VLSI Implementation,” IEEE Transactions on Computers, C-33(5):414-426, May 1984.

[8] Shousheng He and Mats Torkelson, “A New Approach to Pipeline FFT Processor,” Proceeding of International Parallel and Distributed Processing Symposium(IPDPS), The 10th International, pp. 766-770, 1996.

[9] Shousheng He and Mats Torkelson, “Designing Pipeline FFT Processors for OFDM (de)Modulation,” Proceeding of 1998 URSI International Symposium on Signals, Systems, and Electronics, pp. 256-262, 1998.

[10] P. D .Welch, “A Fixed-Point Fast Fourier Transform Error Analysis,” IEEE Transaction on Audio and Electro Acoustics, vol. AU-17, pp. 151-157, June 1969.

[11] A.V. Oppenheim and C. W. Weinstein, “Effects of Finite Register Length in Digital Filters and the Fast Fourier Transform,” Proceeding IEEE, vol. 60, pp. 957-976, Aug. 1972.

[12] M. Sundaramurthy and V. Umapathi Reddy, “Some Results in Fixed-Point Fast Fourier Transform Error Analysis,” IEEE Transactions on Computers, pp. 305-307, March 1977.

[13] Nuthalapati Chowdary and Willem Steenaart, “Accumulation of Product Roundoff Errors in Modified FFT’s,” IEEE Transactions on Circuits and Systems, vol. CAS-33, No.1, pp. 103-107, January 1986.

[14] R. Meyer, “Error Analysis and Comparison of FFT Implementation Structures,” IEEE Proceeding of 1989 ICASSP, vol. 2, pp. 888-891, 1989.

[15] N. S. Jayant and P. Noll, Digital Coding of Waveforms Principles and Applications to Speech and Vidio, Prentice Hall, 1984.

[16] K. Sayood, Introduction to Data Compression, Second Edition, Morgan Kaufmann, 2000.

[17] Stefan Johansson, Shousheng He, and Peter Nilsson, “Wordlength Optimization of a Pipelined FFT Processor,” Proceeding of Midwest Symposium on Circuits and Systems (MWSCAS), pp. 501-5.3, 1999.

[18] Synopsys DesignWare, http://www.synopsys.com.

[19] Synopsys Design Analyzer, http://www.synopsys.com.

[20] Artisan TSMC 0.25um Process High-Density Dual-Port SRAM (HD-SRAM-DP) Generator User Manual, Release 1.0, June 2000, http://www.artisan.com.

[21] Artisan TSMC 0.25um Process High-Speed Single-Port SRAM (HD-SRAM-SP) Generator User Manual, Release 3.0, June 2000, http://www.artisan.com.

[22] W. C. Yeh, “Arithmetic Module Design and Its Application to FFT”, PhD. Dissertation, National Chiao Tung University, Taiwan, Jul. 1, 2001.

Vita

Chih-Bin Kuo was born in Miaoli, Taiwan, in 1974. He graduate from National Yunlin Industrial Junior College, Yunlin, Taiwan, in June 1994 and entered the Institute of Electronics, NCTU in September 2001. His major studies were computer aided design (CAD) and electronic design automation (EDA). He received the M.S. degree from NCTU in August 2004.

93

碩士論文

利用混合方法進行管線化快速傅利葉轉換處理器的字元長度最佳化之研究

電機資訊學院
電子與光電學程

郭志彬