

國立交通大學

電子工程學系電子研究所

博士論文

單晶片網路系統平台設計最佳化之研究

  
**On the Study of Design Optimization for  
Network-on-Chip Platform**

研究生：王成業  
指導教授：周景揚

中華民國九十六年六月

單晶片網路系統平台設計最佳化之研究

**On the Study of Design Optimization for  
Network-on-Chip Platform**

研究生：王成業      Student：Cheng-Yeh Wang  
指導教授：周景揚      Advisor：Jing-Yang Jou

國立交通大學

電機學院

電子工程學系

電子研究所



A Dissertation

Submitted to Department of Electronics Engineering  
and Institute of Electronics

College of Electrical and Computer Engineering

National Chiao-Tung University

in partial Fulfillment of the Requirements

for the Degree of

**Doctor of Philosophy**

in

Electronics Engineering

June 2007

Hsin-Chu, Taiwan, Republic of China

中華民國九十六年六月

# 單晶片網路系統平台設計最佳化之研究

學生：王成業

指導教授：周景揚

國立交通大學

電機學院

電子工程學系

電子研究所



隨著系統晶片持續的成長，設計複雜度增加，降低設計時程是一個重要的課題。因此我們提出單晶片網路產生器、快速傅立葉轉換產生器、以及乘法產生器來降低設計所需時間。讓系統設計者可以快速的得知電路的特性，讓電路設計者可以得到一個良好的設計。

在本研究中，我們提出高效能單晶片網路系統。根據虛擬電路交換、使用專屬的通道、流量採用比重分配、管線匯流排，我們可以達到以下的特性：(1) 保證頻寬 (2) 節省記憶體使用 (3) 不會產生死結。此外我們還提出通訊導向的任務指派演算法。此演算法使用剖析驅動的策略，讓系統總處理能力最大化。因此實作了時間精準的網路模擬器用於分析網路流量。通訊競爭的資訊可以被分析，並且回

饋到最佳化流程。在最佳化的流程中，把時間與空間的通訊競爭一起考慮，讓系統的通訊競爭降到最低。此外由於頻寬能力有其極限，頻寬需求超出極限時，整個系統會被其極限所限制住。因此頻寬的需求會被考慮到。在此演算法中，考慮到通訊量以及通訊競爭的狀況。相較於沒有考慮的狀況下，系統效能提昇 20%。

此外，針對管線式快速傅立葉轉化器的字元長度最佳化提出混合式的方法。此方法使用快速統計分析以及精準模擬分析來達到加速最佳化的流程。發展出統計上的不同字元長度的錯誤模型用於此最佳化流程。此外，硬體相關的模型被抽取出來計算工作頻率，以達到使用者的頻率要求。在最佳化的過程中，根據使用者的限制，首先找出最佳解可能的範圍，還有最佳解的形式可以有效的降低最佳解的空間。此混合式方法相較於純粹的模擬分析可以有效的減少所需評估時間，相較於純粹的統計分析，可獲得更加精確的結果。

最後，一個有效率的合成演算法被提出用於乘法器的自動產生器。此合成演算法在非規則的樹狀合成問題上考慮到邏輯閘延遲以及繞線延遲。根據合成樹的到達時間以及應到時間的要求，可以平衡生成樹的路徑延遲。使用新穎的樹狀生成演算法，考慮到每一條路徑的延遲。所建構的合成器可以產生高速度以及小面積的乘法器。

# On the Study of Design Optimization for Network-on-Chip Platform

Student : Cheng-Yeh Wang      Advisor : Jing-Yang Jou

Department of Electronics Engineering

and Institute of Electronics

National Chiao-Tung University



## Abstract

As System-on-Chip (SoC) designs progressively grow, reducing the development time becomes a crucial challenge. Therefore, the Network-on-Chip (NoC) generator, the FFT generator, and the multiplier generator are developed for reducing the design time.

In this work, a high-performance Network-on-Chip platform is presented. To achieve (1) bandwidth guarantee (2) economical memory usage, and (3) deadlock free, the virtual-circuit switching with dedicated connection path, virtual channel flow control with weighted

round-robin scheduling, and the pipelined bus are used. To perform the task binding, a communication-driven task binding algorithm is proposed. This algorithm employs profile-driven strategy to bind tasks onto the NoC such that the overall system throughput can be maximized. To analyze the network traffic, a cycle-accurate network simulator is hence implemented. The traffic contention information is analyzed, and then fed back to the proposed optimization flow. The effects of communication amount, traffic contention, and bandwidth requirement are considered to perform the task binding. The overall system throughput is improved up to 20% for 100 test cases as compared with the task binding without considering the communication and contention effects.

This thesis also describes a novel hybrid method for the wordlength optimization of pipelined FFT processors which is the arithmetic kernel of OFDM-based systems. This methodology utilizes the rapid computation of statistical analysis, and the accurate evaluation of simulation-based analysis to investigate a speedy optimization flow. A statistical error model for varying wordlengths of PE stages of an FFT processor is developed to support this optimization flow. A technology-dependent model is extracted to support the FFT's operating frequency constraint. The wordlength boundary is found by constraints, and the optimal form is introduced to reduce computation time. Experimental results show that the wordlength optimization employing the speedy flow reduces the percentage of the total area of the FFT processor that increases with an increasing FFT length. The proposed hybrid method requires shorter prediction time than the absolute simulation-based method

does and achieves more accurate outcomes than a statistical calculation does.

Finally, an effective multiplier synthesis algorithm for cell-based multipliers is presented. The synthesis algorithm considers gate delay and wire delay for non-regular tree synthesis. Based on arrival time and required time of the tree constraints, the generated compressed tree can achieve balanced path delay. By using a novel tree generation algorithm with timing consideration for each vertical compressor slice (VCS), the developed synthesizer can automatically generate high-speed multipliers in small area.







# Acknowledgements

First and foremost, I would like to express my greatest appreciation to my advisor, Professor Jing-Yang Jou (周景揚教授) for his suggestions and guidances. He not only encourages the freedom thinking but also is our ideal. I would like to very much thank to Professor Lan-Da Van (范倫達教授) and Professor Juinn-Dar Huang (黃俊達教授), who give me valuable suggestions. Also, I would like to thank the involved members in the projects, Ya-Chi Yang (楊雅琪), Tson-Yee Lin (林忠毅), Chih-Bin Kuo (郭志彬), Pao-Jui Huang (黃保瑞), Chih-Chieh Chou (周志杰), and Guan-Hao Chen (陳冠豪). Without the seamless cooperation, the projects would not be so successful. Thanks to Shang-Wei Tu (涂尚璋), Geeng-Wei Lee (李耿維), and Liang-Yu Lin(林亮宇) for several useful discussions. Special thanks to all EDA members for the wonderful time we share together.

I would like express my sincere appreciation to Miss Zwei-Mei Lee (李瑞梅), my girlfriend, for her patient encouragement and the discussion of writing this thesis. I would appreciate my family for their patient wait and my father education philosophy ”no won-

der sometimes incompetent people win out in imperial examinations, but does this mean that the erudite ones will be neglected?(蓋有幸而獲選，孰云多而不揚)”。

CHENG-YEH WANG

*National Chiao-Tung University*

*2007, June*



# Contents

<b>中文摘要</b>	<b>i</b>
<b>English Abstract</b>	<b>iii</b>
<b>Acknowledgements</b>	<b>vii</b>
<b>List of Tables</b>	<b>xiii</b>
<b>List of Figures</b>	<b>xv</b>
<b>1 Introduction</b>	<b>1</b>
1.1 Motivation . . . . .	1
1.2 Thesis Organization . . . . .	6
<b>2 Network on Chip</b>	<b>9</b>
2.1 Overview of Network Switching . . . . .	12
2.1.1 Circuit Switching . . . . .	12



2.1.2	Connectionless Packet Switching . . . . .	14
2.1.3	Virtual-circuit Switching . . . . .	15
2.2	Architecture Models and Platform Design . . . . .	16
2.2.1	Network Switching . . . . .	18
2.2.2	Switch Design . . . . .	19
2.2.3	Quality of Service Modeling and Property . . . . .	23
2.3	Task Binding Methodology . . . . .	26
2.3.1	NoC platform Modeling . . . . .	27
2.3.2	Task Binding Problem . . . . .	28
2.3.3	Task Mapping . . . . .	33
2.3.4	Connection Path Assignment . . . . .	34
2.4	Experimental Results and Discussion . . . . .	36
2.5	Summary . . . . .	46
<b>3</b>	<b>FFT Processor</b>	<b>49</b>
3.1	Overview of FFT . . . . .	52
3.2	Error Analysis . . . . .	54
3.2.1	Statistical Analysis . . . . .	54
3.2.2	Simulation-Based Method . . . . .	62
3.2.3	Demonstration of Statistical and Simulation-Based Analysis . . . . .	63

3.3	Wordlength Optimization . . . . .	67
3.3.1	Evaluation of Upper Bound Wordlength . . . . .	69
3.3.2	Evaluation of Lower Bound Wordlength . . . . .	70
3.3.3	Optimized Wordlength Search . . . . .	73
3.4	Experimental Results . . . . .	79
3.4.1	Variant FFT Lengths . . . . .	81
3.4.2	Output SQNR Requirement . . . . .	83
3.4.3	Loose SQNR Error Requirement . . . . .	86
3.4.4	Comparisons of Three Methods . . . . .	86
3.5	Summary . . . . .	88
<b>4</b>	<b>Multiplier Generator</b>	<b>89</b>
4.1	Overview of Multiplier . . . . .	89
4.1.1	Non-Booth Encoding . . . . .	92
4.1.2	Wallace Tree Method . . . . .	92
4.2	Layout-Driven Multiplier Generation . . . . .	94
4.2.1	Delay Estimation . . . . .	97
4.2.2	Feasibility Checking Algorithm . . . . .	102
4.2.3	Vertical Compressor Slice (VCS) Generation . . . . .	106
4.3	Experimental Results and Summary . . . . .	109

<b>5</b>	<b>Conclusions and Future Works</b>	<b>111</b>
5.1	Conclusions . . . . .	111
5.2	Future Works . . . . .	112
	<b>Bibliography</b>	<b>115</b>



# List of Tables

3.1	Example of Random Verification . . . . .	64
3.2	Common Specifications of FFT for OFDM . . . . .	79
3.3	Constraints for Optimization . . . . .	81
3.4	Area Reduction of R2SDF with Different FFT Lengths Using Hybrid Method . . . . .	82
3.5	Area Reduction of R2 <sup>2</sup> SDF with Different FFT Lengths Using Hybrid Method . . . . .	83
3.6	Area Reduction of R2SDF with Different FFT Lengths Using Statistical Analysis . . . . .	85
4.1	Selection of partial product . . . . .	91
4.2	Experimental Results . . . . .	109





# List of Figures

1.1	OFDM transmitter data flow graph [1]	2
1.2	OFDM receiver data flow graph [1]	2
1.3	OFDM channel estimation [1]	2
1.4	System modeling graph[2]	4
1.5	Overview of NoC platform	6
2.1	A point-to-point network.	9
2.2	A bus-based network.	10
2.3	A switch-based network.	10
2.4	Circuit switched network.	13
2.5	Packet switched network.	14
2.6	Virtual-circuit switched network.	16
2.7	Mesh-based interconnection architecture of the NoC platform.	17
2.8	Transformation from relay station to switch.	18

2.9	An example of virtual channel scheme. . . . .	19
2.10	Bandwidth allocation of a physical channel using a weighted round-robin scheduler. . . . .	20
2.11	Interface transactions between two switches. . . . .	21
2.12	Real-time QoS modeling. . . . .	23
2.13	NoC platform model. . . . .	26
2.14	The task graph of MPEG-4 encoder. . . . .	27
2.15	Illustration of binding methodology. . . . .	29
2.16	Contention in time domain. . . . .	31
2.17	Proposed task binding methodology. . . . .	32
2.18	Pseudo code of the path assignment. . . . .	35
2.19	Histograms of normalized latency under different injection rate. . . . .	38
2.20	Histograms of normalized latency under different buffer size of virtual channel. . . . .	39
2.21	Fail rate under different communication factors. . . . .	41
2.22	Fail rate under different buffer size. . . . .	42
2.23	Latency under different communication factors. . . . .	43
2.24	Latency under different buffer size. . . . .	44
2.25	Throughput ratio under different communication factor. . . . .	45



3.1	Conventional R2SDF and R2 <sup>2</sup> SDF DIF implementations. . . . .	53
3.2	Error model of a PE stage. . . . .	56
3.3	Propagation of quantization and scaling errors. . . . .	57
3.4	Propagation of multiplication errors. . . . .	59
3.5	Propagation of noiseless multiplications. . . . .	60
3.6	Block diagram of simulation analysis. . . . .	62
3.7	Histogram of SQNR difference with randomly generated wordlengths. . .	65
3.8	Histogram of SQNR difference with partial exhaustive verification. . . .	66
3.9	Wordlength optimization flow of a PE stage. . . . .	68
3.10	Evaluation of the upper bound wordlength. . . . .	69
3.11	Evaluation of the lower bound wordlength. . . . .	71
3.12	Area increment of each PE stage as the wordlength increases 1 bit . . . .	72
3.13	The procedure to determine the optimized wordlength set candidates. . . .	74
3.14	Optimized wordlength selection. . . . .	76
3.15	An example of hybrid wordlength optimization. . . . .	78
3.16	An example of pure statistical analysis. . . . .	78
3.17	Area reduction rate versus SQNR constraints. . . . .	84
3.18	Comparisons of results using different analytical methods. . . . .	87
4.1	Multiplication steps. . . . .	90

4.2	Non-Booth encoding for $8 \times 8$ multiplication. . . . .	91
4.3	Partial product selection using a row of AND gates. . . . .	91
4.4	Illustration of Wallace tree for reducing 18 partial products. . . . .	93
4.5	Overview of multiplier generation. . . . .	94
4.6	Conceptual profile of input arrival time of final adder. . . . .	95
4.7	Overview of vertical compressor slice. . . . .	96
4.8	Evaluation of output arrival time . . . . .	97
4.9	Cell-based delay model of a 3-to-2 compressor . . . . .	98
4.10	$\pi$ -model of the $i$ -th wire ( $w_i$ ) . . . . .	98
4.11	L-shaped approximation between two cells. . . . .	99
4.12	An example of wire delay estimation using L-shaped approximation. . . . .	100
4.13	Full Decomposition (FD) procedure. . . . .	101
4.14	Feasibility Checking (FC) algorithm. . . . .	103
4.15	The first step of feasibility checking: decomposition. . . . .	105
4.16	The second step of feasibility checking: further decomposition. . . . .	105
4.17	The third step of feasibility checking: check the derived arrival time. . . . .	106
4.18	VCS generation algorithm. . . . .	107
4.19	Experimental flow. . . . .	108

# Chapter 1

## Introduction



### 1.1 Motivation

In SoC era, available gate count grows year by year. Effectively utilizing silicon area is a significant challenge. Merging chips into a single chip becomes the mainstream to improve the silicon utilization. This trend is especially obvious in computer industry. CPU companies have developed multi-core processors by implementing more than one cores into a processor [3]. Chipset companies intend to integrate more peripherals in a single chip. In addition to the mentioned above, embedded system designs are also conducted to integrate processors, accelerators, and peripherals into a single chip to implement handheld devices [4].

In application domain, Software-defined Radio (SDR) systems [5] are used to imple-

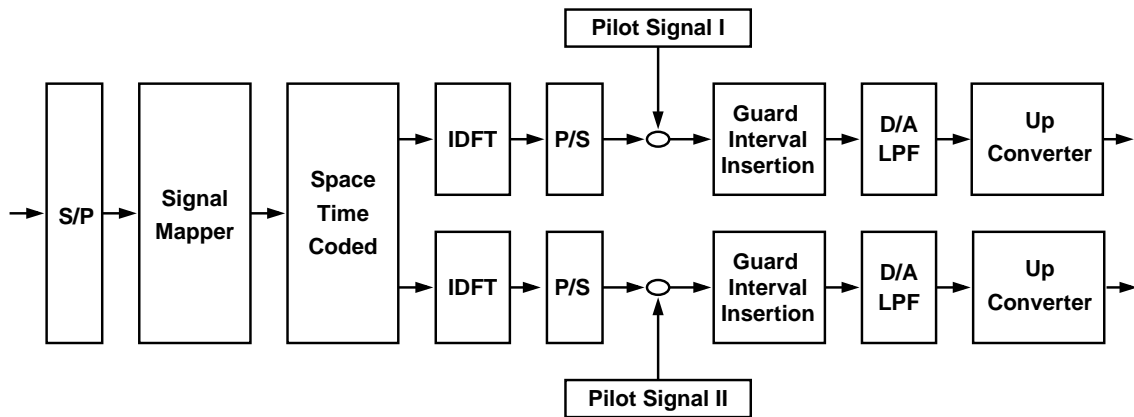


Figure 1.1: OFDM transmitter data flow graph [1]

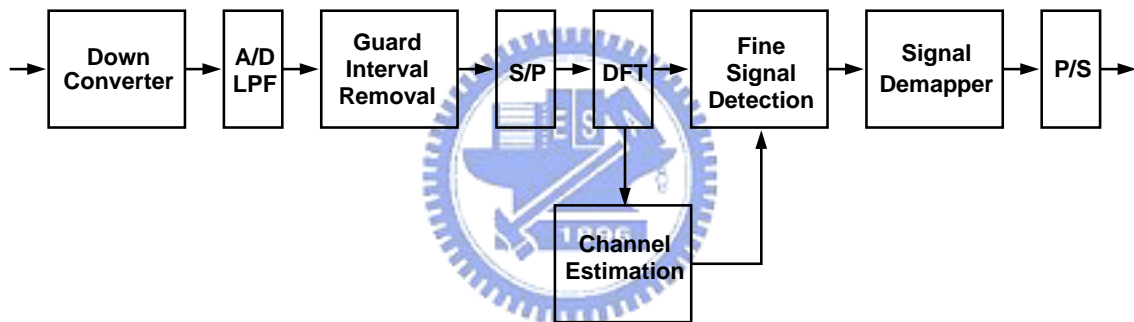


Figure 1.2: OFDM receiver data flow graph [1]

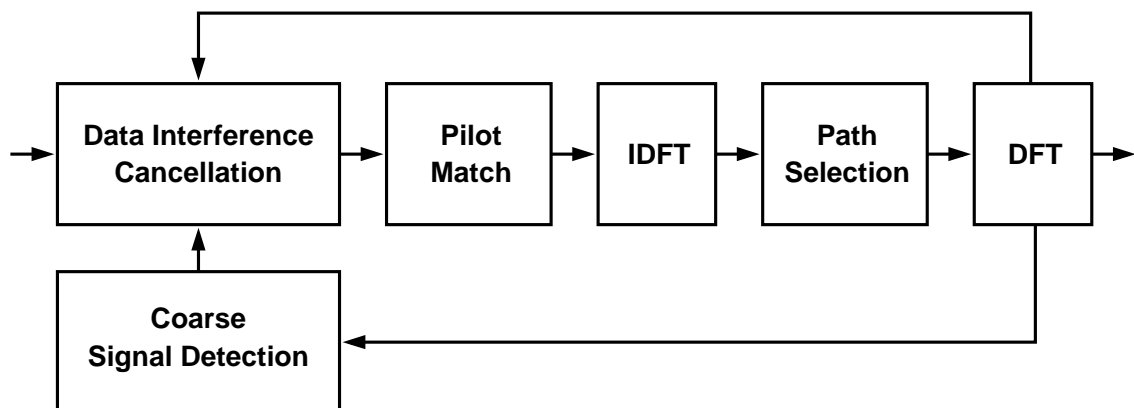


Figure 1.3: OFDM channel estimation [1]

ment multi-standard communications. These SDR systems produce a new radio by running new software. If the target standards use Orthogonal Frequency Division Multiplex (OFDM) to get high spectral efficiency, the computational complexity will be huge increasing. For example, the OFDM system as shown in Fig. 1.1, Fig. 1.2, and Fig. 1.3 was used to achieve high spectral efficiency [1]. This communication software was evaluated, and the design required multiple processors computing power to meet real-time constraint [6]. A programmable multi-core processor with accelerators is suitable for implementing such systems.

The integrated chip has more complex traffic than traditional single-core solution. Hence, on-chip communication is critical in the system implementation [7]. The complex communication scenario could cause the system performance out of control, and result in a failed system. Three major communication networks, the point-to-point connection, the bus-based communication, and the switching network, have been developed to process these complex traffic [8]. In this work, a virtual circuit switching network for SoC is proposed, and it can achieves performance guarantee and high utilization.

To optimize an SoC system, the tradeoff among hardware cost, system performance, and power dissipation can be performed at system level, register transfer level (RTL), and circuit level. When the tradeoff is assessed at system level, it has more flexibility to improve the system than that is done at other levels. Traditionally, system designers came to a compromise according to experience or manual estimation. However, systems are

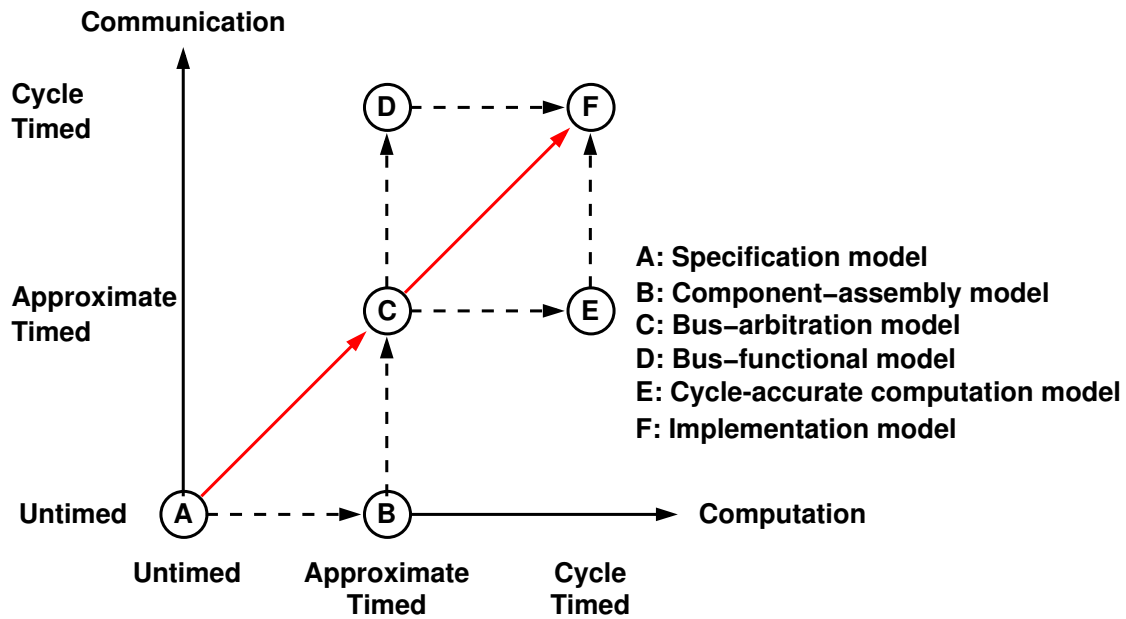


Figure 1.4: System modeling graph[2]

more and more complex along with the progressive technology. This tradeoff cannot just be manually handled, and thus Computer Aided Design (CAD) becomes more and more important in system designs [9].

There are several models proposed at system level as shown in Fig. 1.4 [2]. Researchers proposed a top-down refinement flow from untimed functional model to cycle accurate model. Architecture designers create the model and the systematic analysis is applied to optimize the architecture.

Design space exploration (DSE) can identify a suitable architecture for specific application, and then the architecture candidate is detailedly evaluated. The DSE is time-consuming if it is performed at low abstraction levels, e.g. RTL. Since reducing design time is crucial for designers in SoC era, the DSE is preferred to be performed at electronic



system level (ESL). At ESL, system's function and timing information are modeled for evaluating the system. The timing information of a building block can easily be modified for different implementation. For example, the single data rate (SDR) DRAM and double data rate (DDR) DRAM are both modeled as memory storage, but with different timing information. If system evaluation indicates that memory bandwidth is insufficient, and SDR DRAM is going to be replaced by DDR DRAM, designers only need to change the timing model of this building block to introduce a new component.

After obtaining the architecture candidate at ESL, the DSE of this candidate is then performed at RTL for refinement. Since the design space of each functional block needs to be explored, it is definitely time-consuming for a complex system. Therefore, it can significantly reduce the development time, if RTL functional blocks can be generated automatically.

In this thesis, a Network-on-Chip (NoC) platform are proposed for software-defined OFDM wireless communication. A NoC generator, a fast fourier transform (FFT) generator, and a multiplier are developed for speeding up the DSE of a system as shown in Fig. 1.5. These generators produce the optimized prototypes based on the given specifications. The produced prototypes have accurate function and timing information. The information is used to replace those of the corresponding functional blocks of a system at ESL for the DSE. Therefore, the DSE can be more accurate. When design flow advances to RTL, these blocks can just be replaced by the prototyping RTL designs produced by the

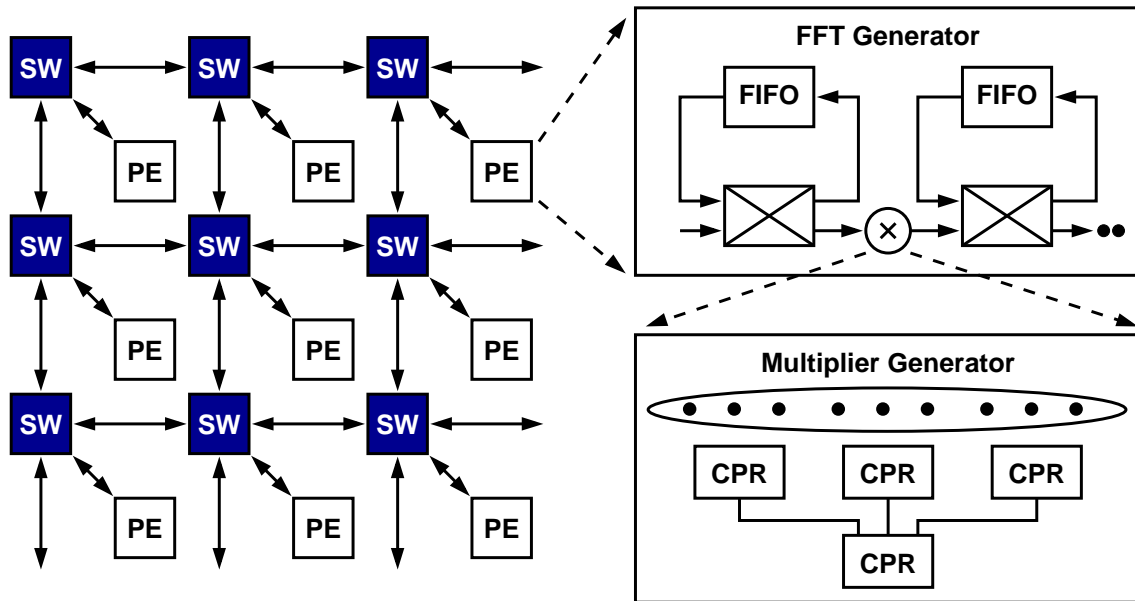


Figure 1.5: Overview of NoC platform

generators for system evaluation. Furthermore, when the flow advances to physical level, the layout of these blocks are replaced by the prototyping layout. The total development time is therefore reduced.

## 1.2 Thesis Organization

This thesis is organized into five chapters. Chapter 1 gives the introduction of the thesis from the motivation of automatic generators. Chapter 2 describes the development of NoC generator, and the methodology for optimizing NoC networks. The experimental results of the NoC generator are given in the rest of this chapter. Chapter 3 briefly reviews pipelined FFT processors, and then presents the FFT generator and the wordlength

optimization algorithm, following by the experimental results. Chapter 4 discusses the multiplier generator based on the gate delay and wire delay optimization. Finally, conclusions and future works are drawn in chapter 5.





# Chapter 2

## Network on Chip

Three types of networks were developed for on-chip communications [8]. A point-to-point communication network is shown in Fig. 2.1 which is constructed using a dedicated channel between the source and the destination. Without sharing channel with other communication traffic, this network has minimum run-time uncertainty, but it requires large silicon area due to the large amount of communication paths. These communication paths

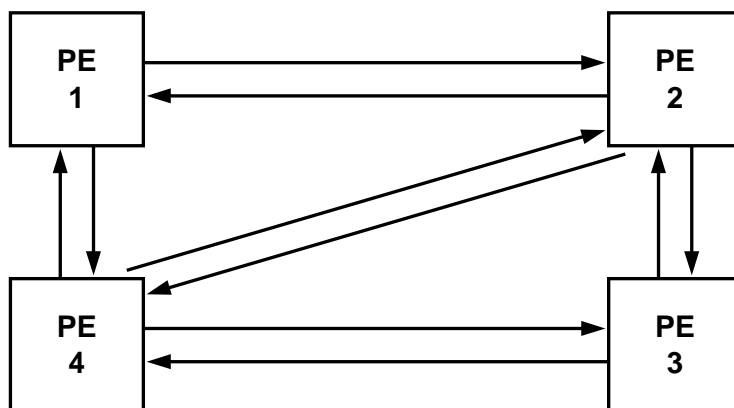


Figure 2.1: A point-to-point network.

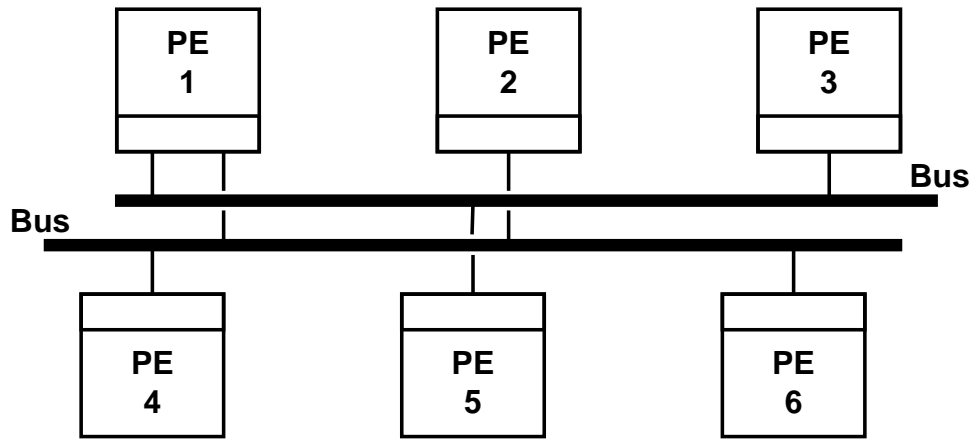


Figure 2.2: A bus-based network.

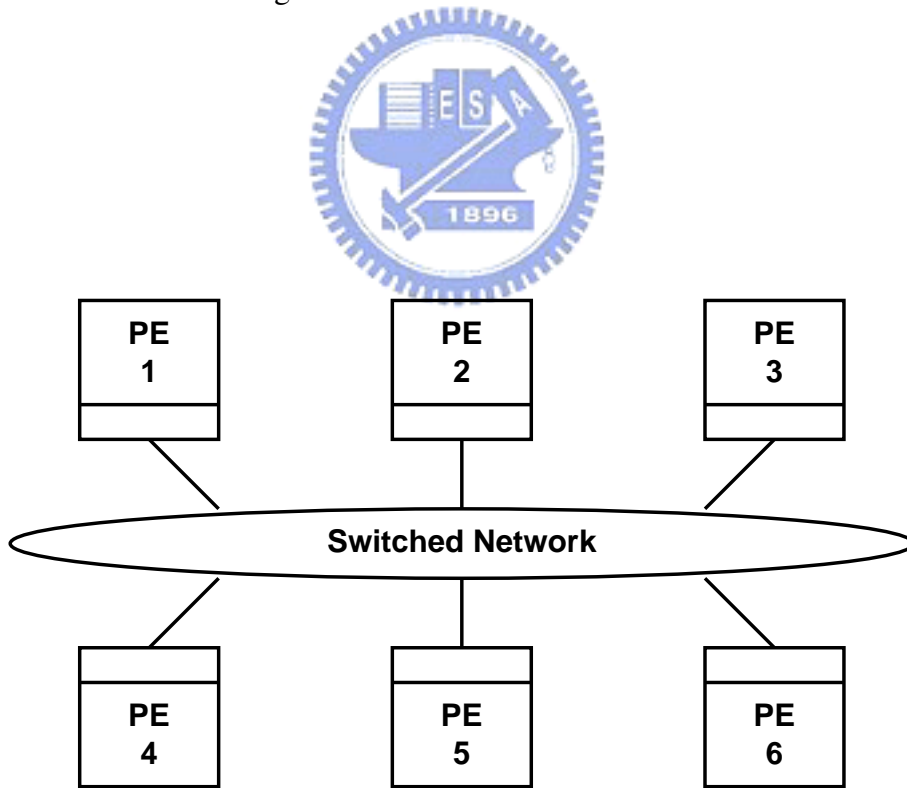


Figure 2.3: A switch-based network.

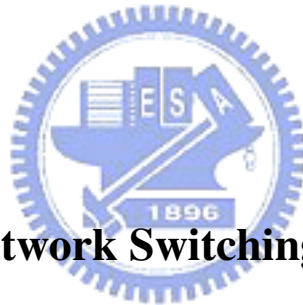
also need to be recognized at design time. Hence, this communication network is often employed in application-specific designs. Fig. 2.2 shows a bus-based communication network which is often used in IP-reused designs. It is a centralized network, and needs an arbitration mechanism to decide which processing element (PE) can use the bus. Such a centralized network will become a communication bottleneck as the number of PEs increases. Since on-chip communication is more and more complex, and traffic is heavier in a SoC design, the switch-based network is hence developed. A conceptual sketch of the switch-based network is shown in Fig. 2.3. This network is decentralized and concurrent, and hence its energy does not waste on meaningless signal transitions.

Circuit switching and packet switching are mostly used for network communications. The connectionless approach and connection-oriented approach are usually employed in packet switched networks. The store-and-forward switching, the virtual cut-through switching, and the wormhole switching are the connectionless approaches. The wormhole switching is suitable for on-chip communications due to good average latency and low memory usage, but it has unpredictable latency under heavy traffic. The connection-oriented approach employs the circuit switched concept in the packet switched network, and it is hence called as virtual-circuit switching.

For Network-on-Chip (NoC), circuit switching and wormhole switching are widely used for on-chip communications [10]. To achieve high resource utilization and performance guarantee, the hybrid method combining circuit switching and wormhole switching

were proposed [11]. However, some applications need both high resource utilization and performance guarantee in each path. In this work, an NoC generator is developed based on the virtual-circuit switching typically used in computer networks to achieve high resource utilization and performance guarantee.

The rest of this chapter is organized as follows. Section 2.1 introduces the network switching. Section 2.2 introduces the developed switch architecture, and the NoC platform design. In Section 2.3, the communication-aware task binding methodology is described. Then, experimental results and discussions are given in Section 2.4. Finally, a summary is remarked.



## 2.1 Overview of Network Switching

In this section, circuit switching, connectionless packet switching, and virtual-circuit switching are briefly reviewed.

### 2.1.1 Circuit Switching

Circuit switching uses the dedicated resources to meet the real-time requirement. However, the dedicated resources will be wasted if traffic is not continuous. Since on-chip traffic is usually a burst transaction, the circuit switching is therefore not adequate to such applications. On the other hand, it is satisfactory in real-time applications.



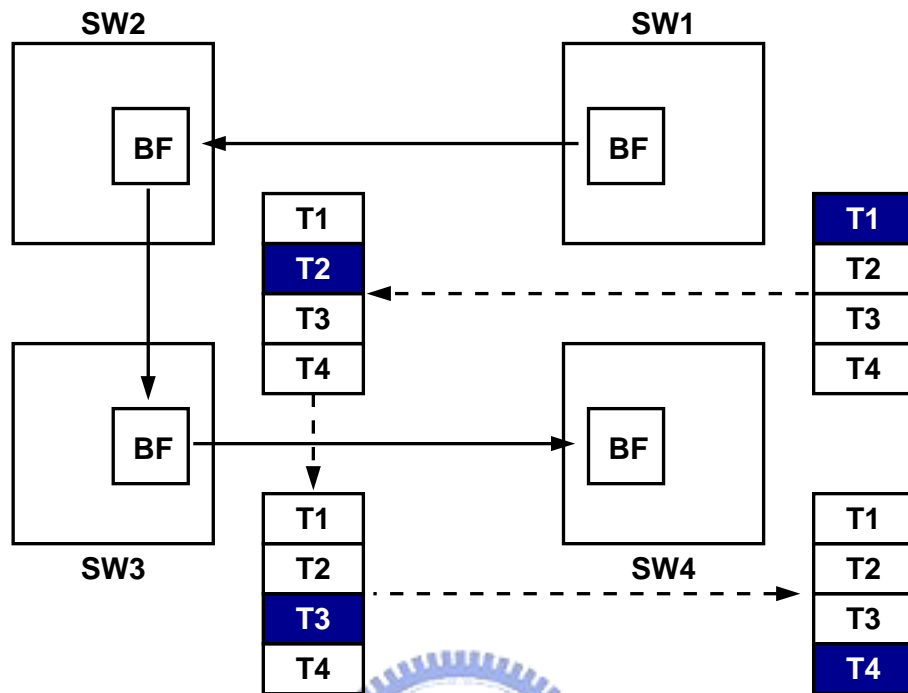


Figure 2.4: Circuit switched network.

Time-division multiplexing (TDM) is employed in circuit switching for transmitting data through dedicated channels. As connection paths are established, the required time slots and buffers will be reserved for data transactions. Hence, the contention will not happen and the performance can be guaranteed.

Fig. 2.4 shows the conceptual plot of a circuit switching network, where  $T1$  to  $T4$  are time slots in a round. The highlighted time slots and buffers are reserved for the path indicated using the solid arrows as shown in Fig. 2.4. If the time slots are well-arranged, the latency can be reduced, but the throughput will not be improved.

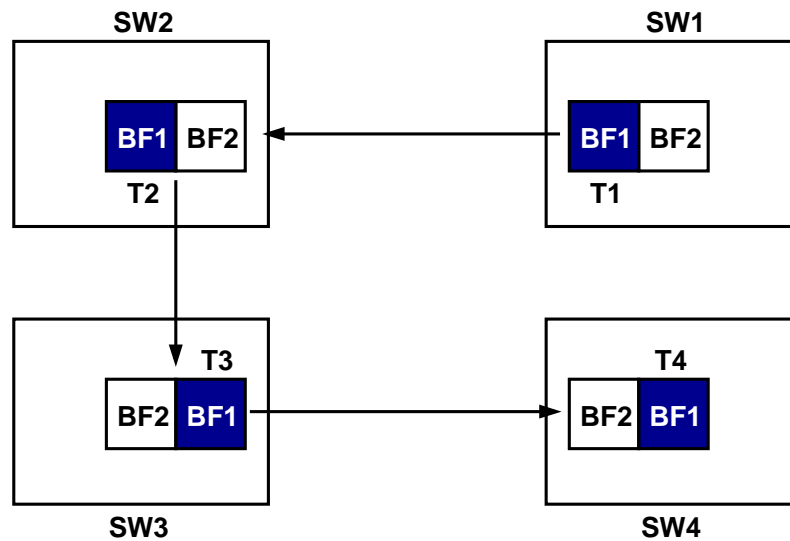


Figure 2.5: Packet switched network.

### 2.1.2 Connectionless Packet Switching

Connectionless packet switching is widely used in data communications. This switching approach employs the shared resources to achieve high resource utilization, but it has unpredictable latency. If there is heavy traffic, the resources will be occupied, and the packet switching network cannot work well.

In connectionless packet switching, buffers are shared by all transactions. Buffers may overflow and will drop packets if the network has no handshaking schemes. If there exists handshaking scheme, a transaction will stall until the buffers of the destination are released. The conceptual sketch of a packet switched network is shown in Fig. 2.5. BF1 and BF2 Buffers in a switch are shared by all packets regardless of the source and the destination. When the switch receives a packet, it reserves a buffer for this packet, and

then releases this buffer when the packet passes through to the destination. Hence, the buffers in the packet switched network can achieve high utilization.

For store-and-forward switched networks, a switch receives a complete packet, and then forwards to the destination. Hence, the switch requires to reserve sufficient buffers for this packet. For virtual cut-through switched networks, a switch needs to reserve enough buffers for a complete packet, but it can forward this packet to the destination directly without completely receiving this packet. For the wormhole switched networks, a switch can directly forward the received packet to the destination without reserving any buffers.



### 2.1.3 Virtual-circuit Switching

Virtual-circuit switching requires setting up a virtual connection from the source to the destination before sending packets. Fig. 2.6 shows the conceptual sketch of a virtual-circuit switched network, where virtual-circuit identifier (VCI) is introduced to specify which virtual-circuit access the physical wires. The VCI is not a global identifier; it has link local scope and is carried inside the header of the packet. As shown in Fig. 2.6, the virtual-circuit table of a switch is initially established based on routing paths, and is used to indicate the VCI (Out VCI) of the delivered packet according to this packet's original VCI (In VCI). The packet delivered from the SW1 switch to the SW4 switch will change the VCI of the packet header from In VCI to Out VCI according to each virtual circuit

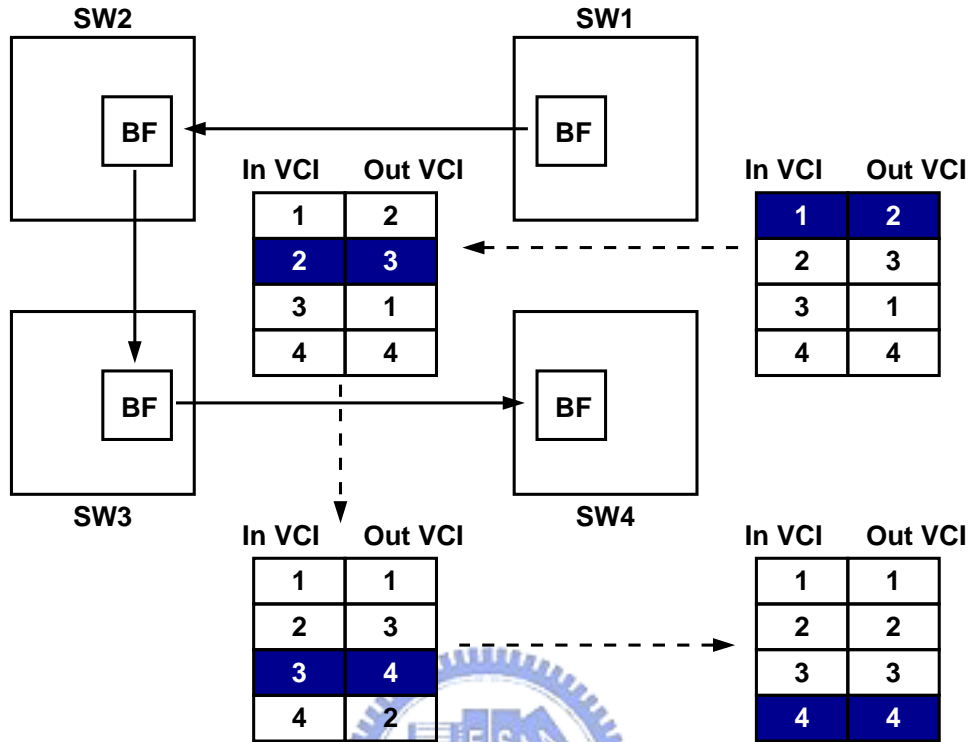


Figure 2.6: Virtual-circuit switched network.

table in the path. If enough buffers and bandwidth are reserved for this path, the quality of service (QoS) can be provided.

## 2.2 Architecture Models and Platform Design

There are many different interconnection architectures of NoC platform. P.P. Pande et al. [12] compare the performance and characteristics of a variety of NoC architectures and also obtain comparative results for a number of common NoC topologies. In this work, several assumptions are made in the following. First, we assume that our interconnection architecture of the NoC platform is a mesh-based topology where the platform

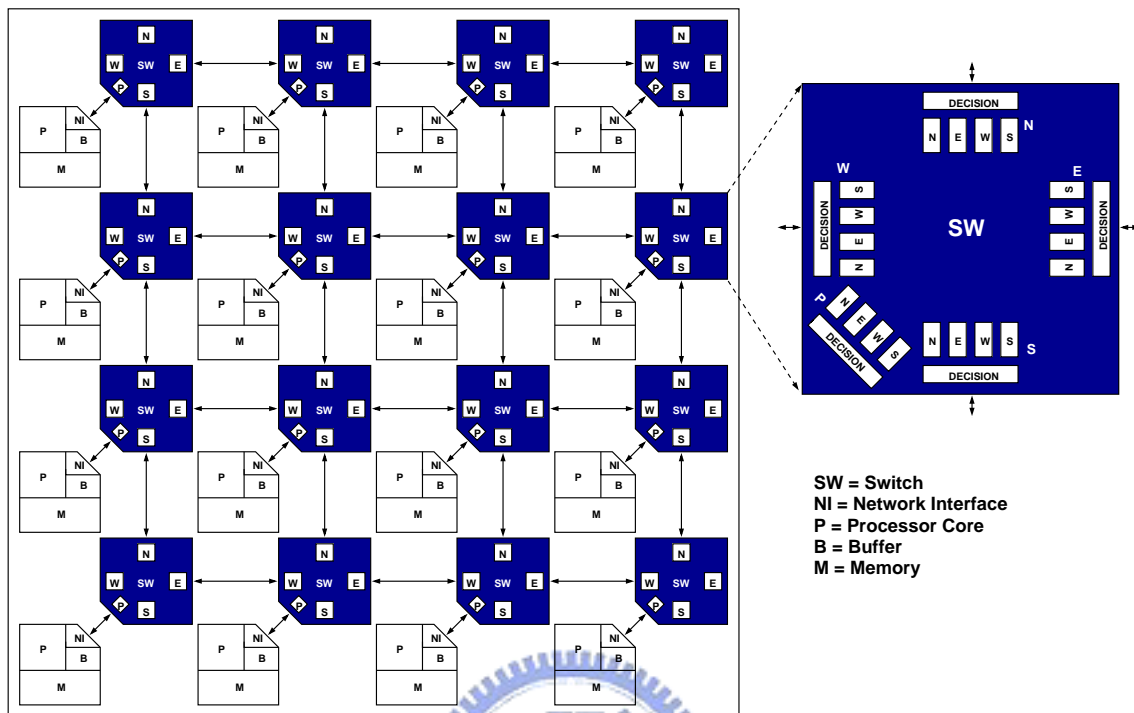


Figure 2.7: Mesh-based interconnection architecture of the NoC platform.

is illustrated in Fig. 2.7. Second, the platform that consists of two kinds of components: identical processors and switches. Third, each processor contains local memory and is connected to the local switch. Fourth, each switch connects to the neighboring switches and the local processor.

Three reasons are considered for choosing the 2-D mesh topology. First, the simple connection and easy routing are preferred in parallel computing platforms [13]. Next, the uniform interconnection among the nodes makes balanced propagation delay between switches and ensures the overall scalability of the network. Finally, this topology meets the plane manufacturing topology of IC technology.

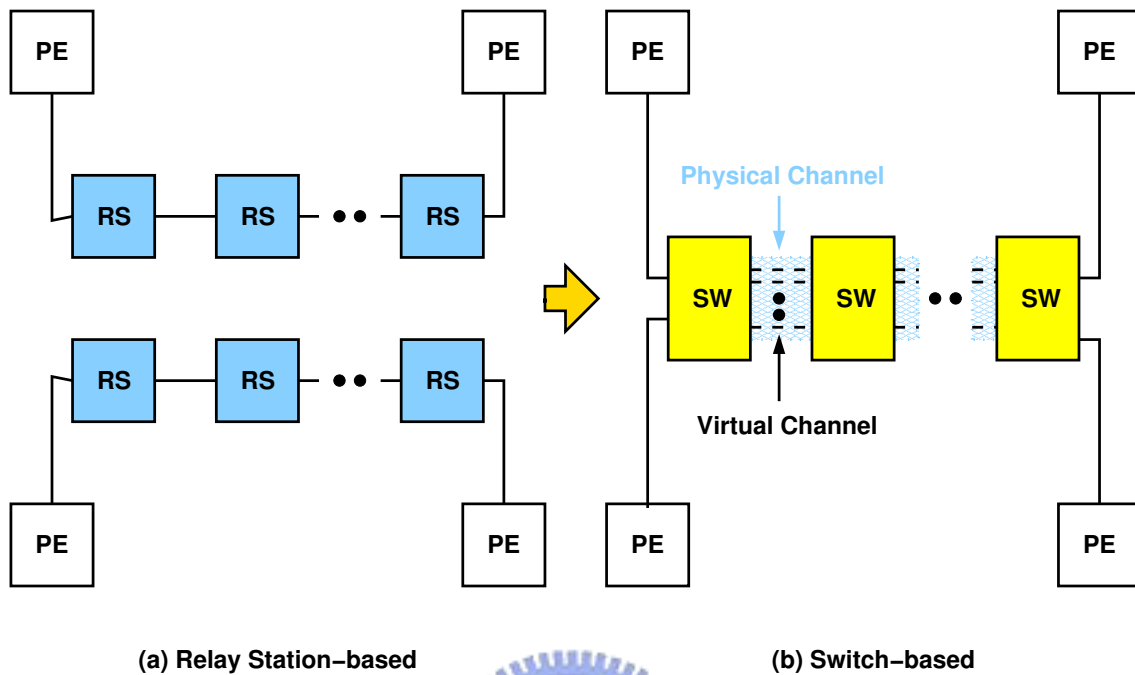


Figure 2.8: Transformation from relay station to switch.

### 2.2.1 Network Switching

We propose a switch architecture based on the latency-insensitive concepts [14] [15] and utilizes the virtual-circuit switching technique to achieve high bandwidth utilization, bandwidth guarantee and predictable latency under heavy traffic condition. Relay station (RS) is used for pipeline the long interconnect in latency-insensitive design. The topology of relay station connection is shown in the Fig. 2.8 [15]. In order to improve the low utilization of the dedicated peer-to-peer connections, the RSs are replaced by our switches and the virtual channels are substituted for the connections between RSs.

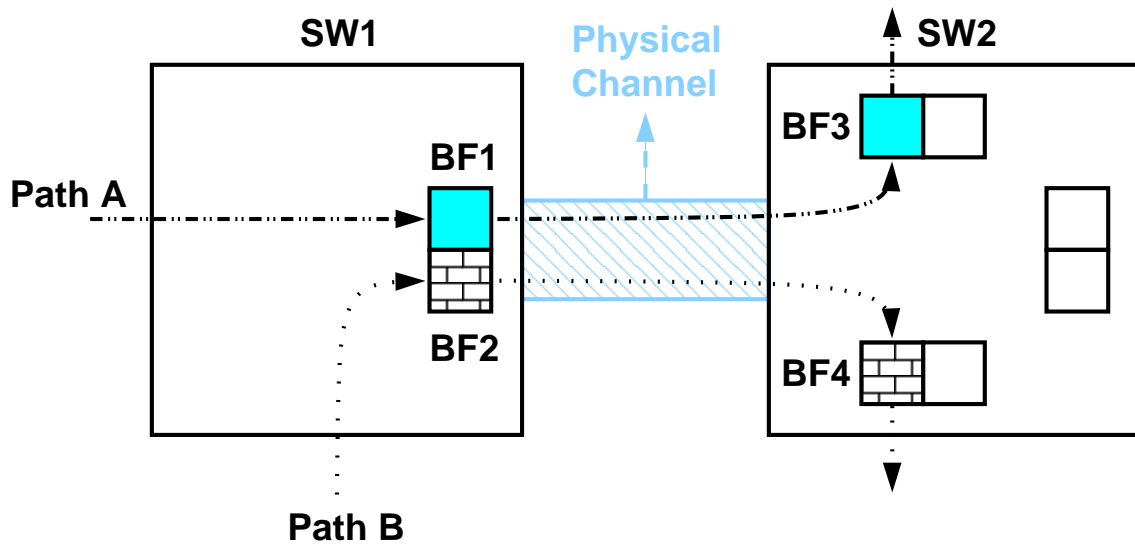


Figure 2.9: An example of virtual channel scheme.

### 2.2.2 Switch Design



The proposed switch architecture using the hybrid of virtual channel scheme, the weighted round-robin scheduling, and SRAM-based configuration is capable of providing high-throughput, bandwidth guarantee, economical memory usage, and deadlock free. We summarize the switch capabilities as follows:

First, each switch based on virtual-circuit switching owns the advantages of predictable behavior and the real-time response. The switches use the virtual channel flow control to enhance the overall latency and the throughput of a network. For example, in Fig. 2.9, there are two messages crossing the physical channel between switches SW1 and SW2. Without using the virtual channel technique, the message data will be buffered at the input or output of the physical channel. Moreover, the transfer in this channel will

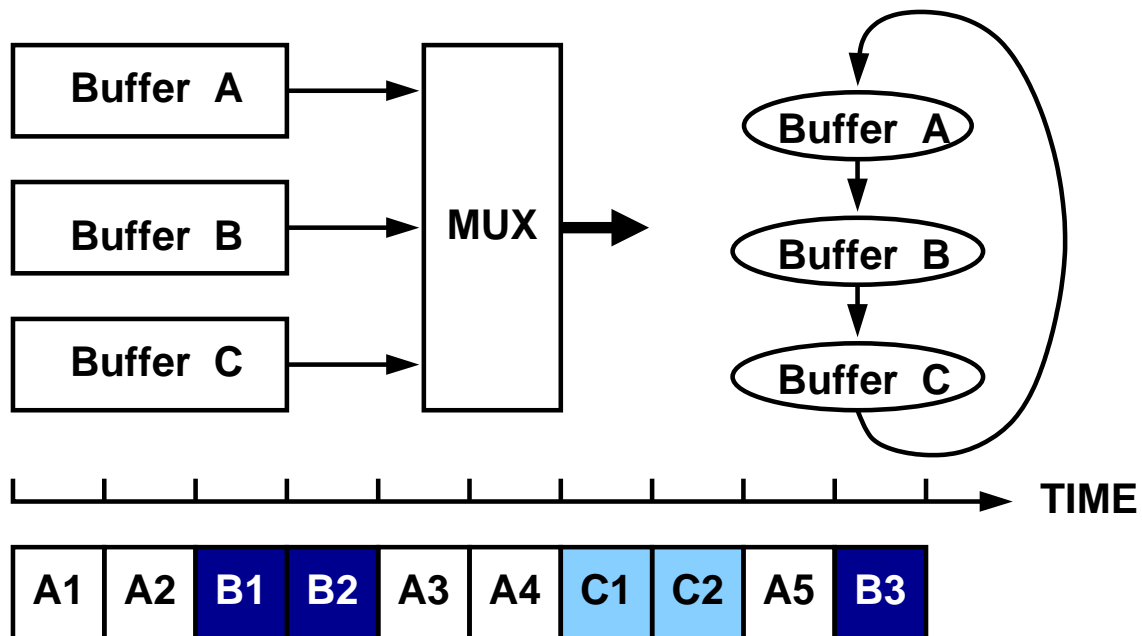


Figure 2.10: Bandwidth allocation of a physical channel using a weighted round-robin scheduler.

be blocked until the buffers are released. In this work, the messages can be delivered rather than blocked by dividing the physical channel into several virtual channels. The waiting time of the message transfer is reduced, and the average latency of this channel is decreased. Thus, the physical channel gets higher utilization and the network obtains a larger throughput.

Second, concerning the bandwidth sharing of a physical channel among all the virtual channels, we exploit the weighted round-robin scheduling scheme to grant the use of the physical channel to each virtual channel. Instead of using the time-division method, the weighted round-robin scheduler as shown in Fig. 2.10 allocates different bandwidth for each virtual channel by assigning different amount of the time slots. The higher weight of



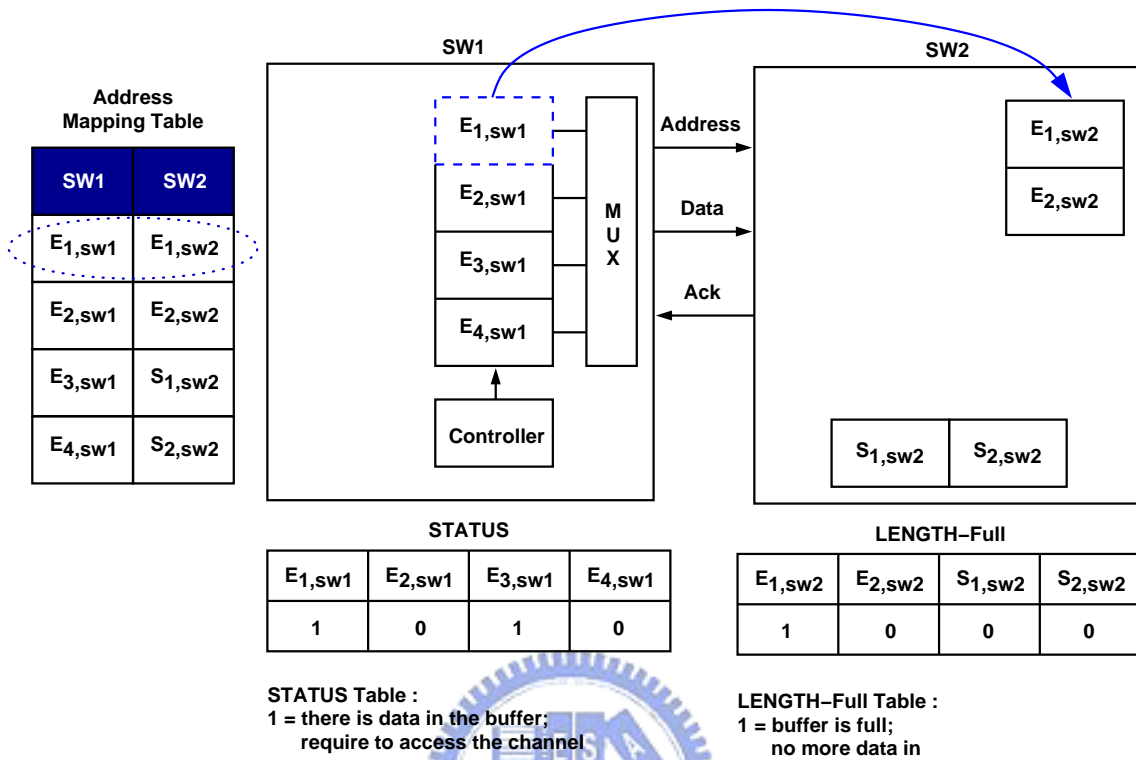


Figure 2.11: Interface transactions between two switches.

a channel means that more communication bandwidth is available.

Third, the data exchange protocol between two switches or between the switch and the network interface of the local processor is executed within four clock cycles. Fig. 2.11 shows that the interface transaction between two adjacent switches, SW1 and SW2. The address mapping table records the destination address to which the messages are transferred. At the first cycle, if the buffer,  $E_{1,sw1}$ , of SW1 has data inside, the system controller grants the channel priority to this data. At cycle 2, this  $E_{1,sw1}$  buffer sends the address of  $E_{1,sw2}$  through the Address-line to indicate that this transaction tries to deliver data to the buffer  $E_{1,sw2}$  of SW2. At the third clock cycle, the buffer  $E_{1,sw2}$  sends the acknowledge

signal, true or false, back to SW1 through the Ack-line according to its buffer status, full or available. Meanwhile, the buffer  $E_{1,sw1}$  sends the data through the Data-line, and the buffer  $E_{1,sw2}$  stores this data if it has spare space. However, this data may be discarded if  $E_{1,sw2}$  is already full. During the fourth cycle, the buffer  $E_{1,sw1}$  keeps this data until the transaction is successfully completed.

Fourth, our switch provides different memory configurations to improve the local memory utilization. The first reason is that not all buffers of the switches are reserved when the number of the connection paths is smaller than the number of the designed buffers. The second reason, the memory is a critical component for buffering data in a network. Therefore, in memory implementation, we use two-port SRAM instead of registers when the number of virtual channel is large in the physical channel. In the switch, the memory is divided into several different sizes of buffers to optimize the utilization. The memory in a switch port can be partitioned into 8 8-words blocks, 16 4-words blocks or 32 2-words blocks.

Finally, in order to support the real-time application, our switches is able to establish the dedicated connection paths in advance by reserving the corresponding virtual channels since the behavior of the communication and the number of the nodes can be predetermined in early stage of system design.

Although both the traditional circuit-switching and the proposed switching configuration have latency guarantee, the proposed one has smaller average latency and higher

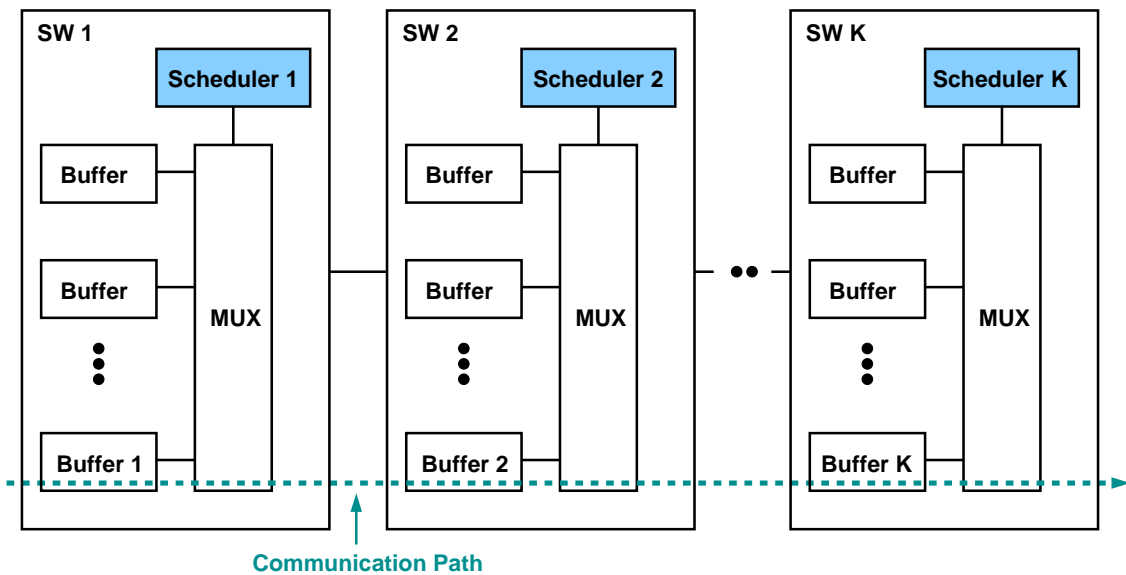


Figure 2.12: Real-time QoS modeling.

hardware utilization. The proposed one has the worst case guarantee as compared to the worm-hole packet switching while both switches have small buffer size and high hardware utilization.

### 2.2.3 Quality of Service Modeling and Property

In the real-time system, the latency guarantee is the essential requirement of the quality of service (QoS) while the scheduling algorithm enables the appropriate task scheduling to satisfy the real-time requirement in the worst case condition. On the other hand, the QoS also plays a critical role even in a non-real time system. Generally, when using the communication fabric without performance guarantee, designers have to expend more design efforts to estimate the communication latency to make sure that the communication

loading is not underestimated for the given on-chip network communication architecture. As a consequence, the communication system infrastructure is usually over-designed to avoid the communication congestion. In this work, we use the weighted round-robin scheduling for our QoS model as shown in Fig. 2.12, where the weighted round-robin scheduling is a minimal resources scheduling scheme. Each master has a weight number  $N_i$  in the controlled scheduler. The scheduler grants the master if the master proposes the request. The master can transmit at most  $N_i$ -word data in a round. After that, the scheduler grants the next master until the round is complete.

Our switches support to establish a predictable communication quality of NoC platform and also provide a simple communication model for reducing the design complexity. As shown in Fig. 2.12, the communication path from Buffer 1 to Buffer  $K$  is established. The transactions from Buffer  $i$  is granted by the weighted round-robin Scheduler  $i$ . Before analyzing the properties of QoS model as exposed in Fig. 2.12, the useful definitions are revealed in the following:

- 1)  $w_{i,j}$  is the weight of the Buffer  $i$  in the weighted round-robin Scheduler  $j$ .
- 2)  $W_j$  is the sum of weight of the buffers controlled by the Scheduler  $j$ .
- 3)  $D_{max}$  denotes the maximum delay of a 1-word transmission.
- 4)  $R$  denotes the provided throughput rate of a buffer.

- 5)  $L_{max}$  denotes the maximum communication path latency of a 1-word transmission.
- 6)  $R_{path}$  denotes the throughput rate of a path.
- 7)  $L_{burst}$  denotes the maximum burst data latency.

Using the above definitions, the proposed network switch design has six properties to guarantee QoS, where the six QoS properties are described as follows:

- Property 1: If Buffer  $i$  is empty, the maximum delay from the data arrival to the transfer is the time period of a round in the round-robin scheduler, i.e.,  $D_{max} = W_j$ .
- Property 2: If there are data in Buffer  $i$ , the  $D_{max}$  between the transactions is  $W_j$ .
- Property 3: If the buffer size is the double of the buffer's weight or more, the provided lower-bound throughput rate is the ratio of the weight and the sum of the weights in the round-robin scheduler, i.e.,  $R \geq \frac{w_{i,j}}{W_j}$ .
- Property 4: The maximum path latency of 1-word transmission is the sum of maximum node latency of 1-word transmission, i.e.,  $L_{max} = \sum_{j=1}^k W_j$ .
- Property 5:  $R_{path}$  is dominated by the minimum throughput of the buffers in the path, i.e.,  $R_{path} = \min \left\{ \frac{w_{i,j}}{W_j} \right\}$ , where  $j = 1, 2, 3, \dots, k$ .
- Property 6: Using Property 4 and Property 5, the burst data delay can be obtained as  $L_{burst} = L_{max} + \frac{N}{R_{path}}$ , where  $N$  is data size.

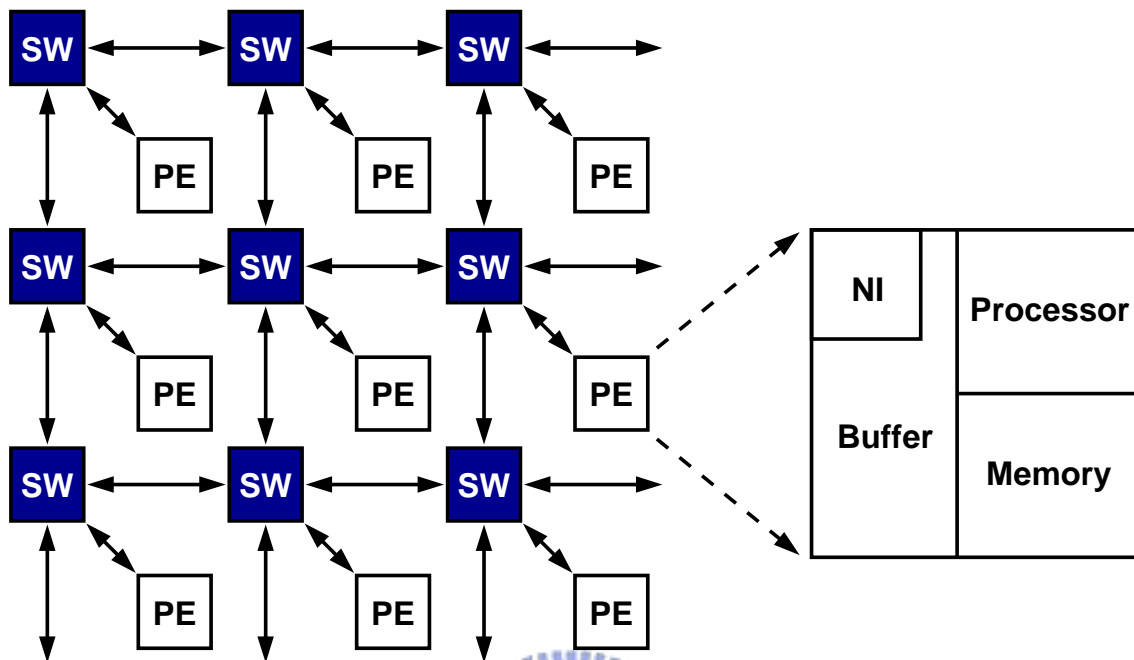


Figure 2.13: NoC platform model.

The Property 6 means that our switch has an upper bound of the burst data delay such that system designers can design target systems to meet real-time constraints.

## 2.3 Task Binding Methodology

In this section, we present the communication-aware methodology to solve the task binding problem based on the NoC platform which is constructed by the proposed switch as mention before.

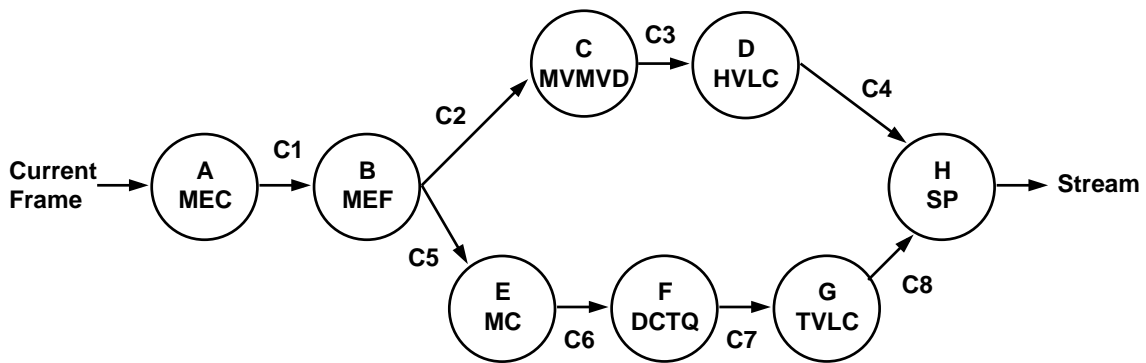
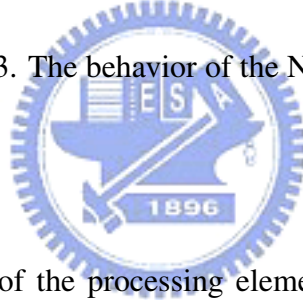


Figure 2.14: The task graph of MPEG-4 encoder.

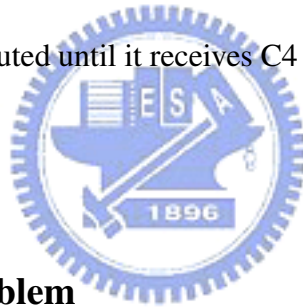
### 2.3.1 NoC platform Modeling

Without loss of generality, the proposed NoC platform using an efficient switch architecture can be modeled in Fig. 2.13. The behavior of the NoC platform model are described in the following:



- 1) The platform composed of the processing element (PE) and switch (SW) is the mesh-based communication architecture. Each PE contains one processor, memory, and network interface (NI).
- 2) All processors in this platform are identical.
- 3) The processors have limited buffer to store input and output data.
- 4) Each PE has local memory to store the execution code and the data.
- 5) The local memory of a PE is adequate for a task.

We employ the task graph to model applications and assume that applications are able to be partitioned into many communicated tasks due to the parallelism. Fig. 2.14 shows a task graph of an MPEG-4 encoder [16]. A vertex represents a task and the functionality labeled in the vertex. For example, task C denoted as MVMVD performs the motion vector to motion vector difference calculation. The edge represents a data transmission and the corresponding communication amount. After task B is finished, task B transmits C2 unit data to task C and transmits C5 unit data to task E. An edge also indicates the data dependency. A task cannot be executed until it receives the data from the predecessor. For example, task H cannot be executed until it receives C4 unit data from task D and C8 unit data from task G.



### 2.3.2 Task Binding Problem

The task binding problem is formulated in the following descriptions:

- Given:
  - 1) The application is modeled as a task graph  $G(V,E)$ , where  $V$  is a task and the weight denote the computation amount and  $E$  is the dependence and the communication amount between the tasks.
  - 2) The NoC platform  $(PE,S)$ , where  $PE$  and  $S$  contain the position information of processors and the communication architecture information, respectively.



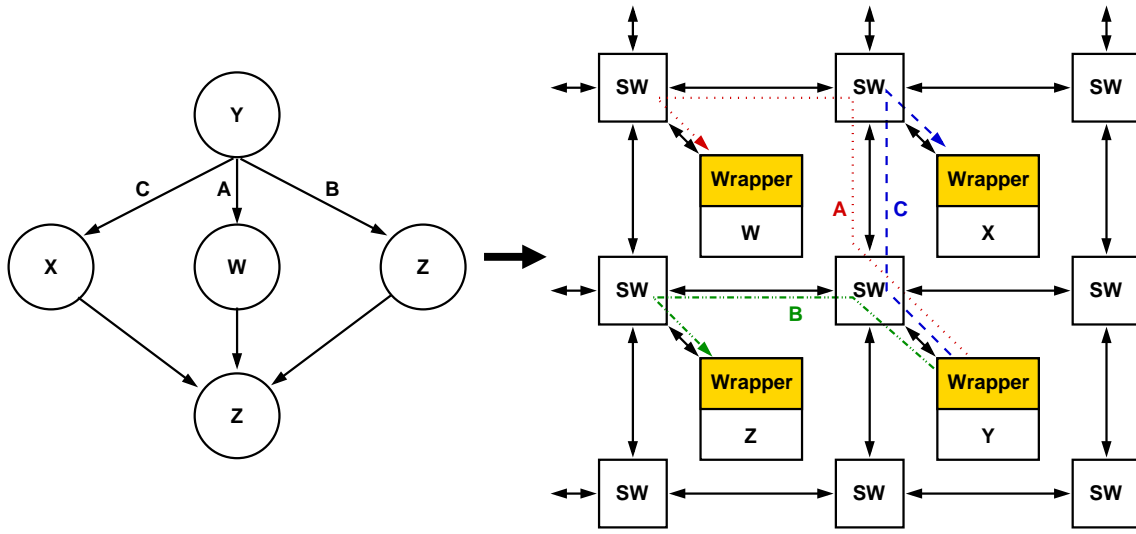


Figure 2.15: Illustration of binding methodology.

- Goal:

To bind each  $V$  to PE and to assign the routing path between tasks such that the overall system throughput is maximum.

An example of binding result is shown in Fig. 2.15 that tasks bind to PEs and the communication paths are established. The path A and path B are overlapped and will contend in temporal domain.

In this work, according to Fig. 2.15, we propose a more accurate communication cost calculation scheme that will be exploited by the task mapping and the routing path assignment processes. The more complete communication cost function  $\xi$  is described in the following.

$$\xi = \sum_{X \in \text{each path}} \sum_{\text{each channel in X}} \left( 1 + \frac{C_{\text{amount}, X} + \rho_X}{C_{\text{amount}, MAX}} + B_{\text{penalty}} \right) \quad (2.1)$$

where  $C_{amount}$ ,  $\rho_X$ , and  $B_{penalty}$  denote the communication amount, contention factor which models the total effect of communication contention on a physical channel, and bandwidth penalty respectively. The detailed expression of  $\rho_X$ , and  $B_{penalty}$  are revealed in the following:

$$\rho_X = \sum_{Y \in \text{other paths in the channel}} C_{amount,Y} \times \Gamma_{Y \rightarrow X} \quad (2.2)$$

and

$$B_{penalty} = \begin{cases} \alpha \times \frac{B_{demanded} - B_{provided}}{B_{provided}} & , \text{ if } B_{demanded} > B_{provided} \\ 0 & , \text{ if } B_{demanded} < B_{provided} \end{cases} \quad (2.3)$$

where  $\Gamma$ ,  $\alpha$ ,  $B_{demanded}$ , and  $B_{provided}$  denotes the contention density, the penalty weight, the demanded bandwidth used by communication paths in a physical channel, and the provided bandwidth of a physical channel respectively. The contention density is formulated as

$$\Gamma_{b \rightarrow a} = \frac{t_{(a,b)overlap}}{t_{commun.time,a}} \quad (2.4)$$

The density of each pair of the communication paths can be derived from the communication profile in the time domain as shown in Fig. 2.16.

The communication cost proposed in (2.1) means that the efficiency of a communication path is affected by the distance of the path, the number of paths to share the channel and the bandwidth usage.

Therefore, we propose a design methodology as shown in Fig. 2.17 to solve the task binding problem. The first block, Task Graph, contains the computation and communi-

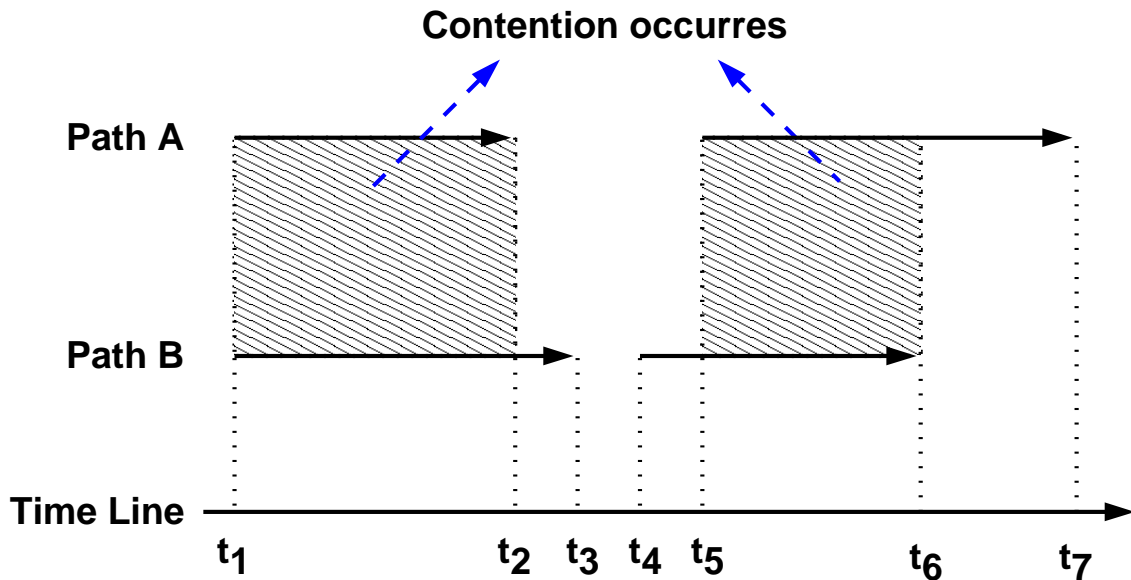


Figure 2.16: Contention in time domain.

cation information for all tasks to support the requirements of the task mapping. The mapping approach employs the placement techniques used in FPGA to map tasks onto PE [17]. The main idea is that if the traffic loading of any two tasks is heavy, these two tasks are allocated as next to each other. After the task mapping, the shortest path technique is applied to configure all the connection paths for these tasks. Then, the simulation is performed to obtain the profile of the communication in time domain and the contention parameters mentioned above are calculated. Finally, the profile feeds back to the task mapping process and the path assignment proceeds this profile to achieve the better assignment. The profile referred to as profile-driven optimization provides the more accuracy contention information than the system simulation without routing information. This design flow can be proceeded iteratively to enhance the system performance.

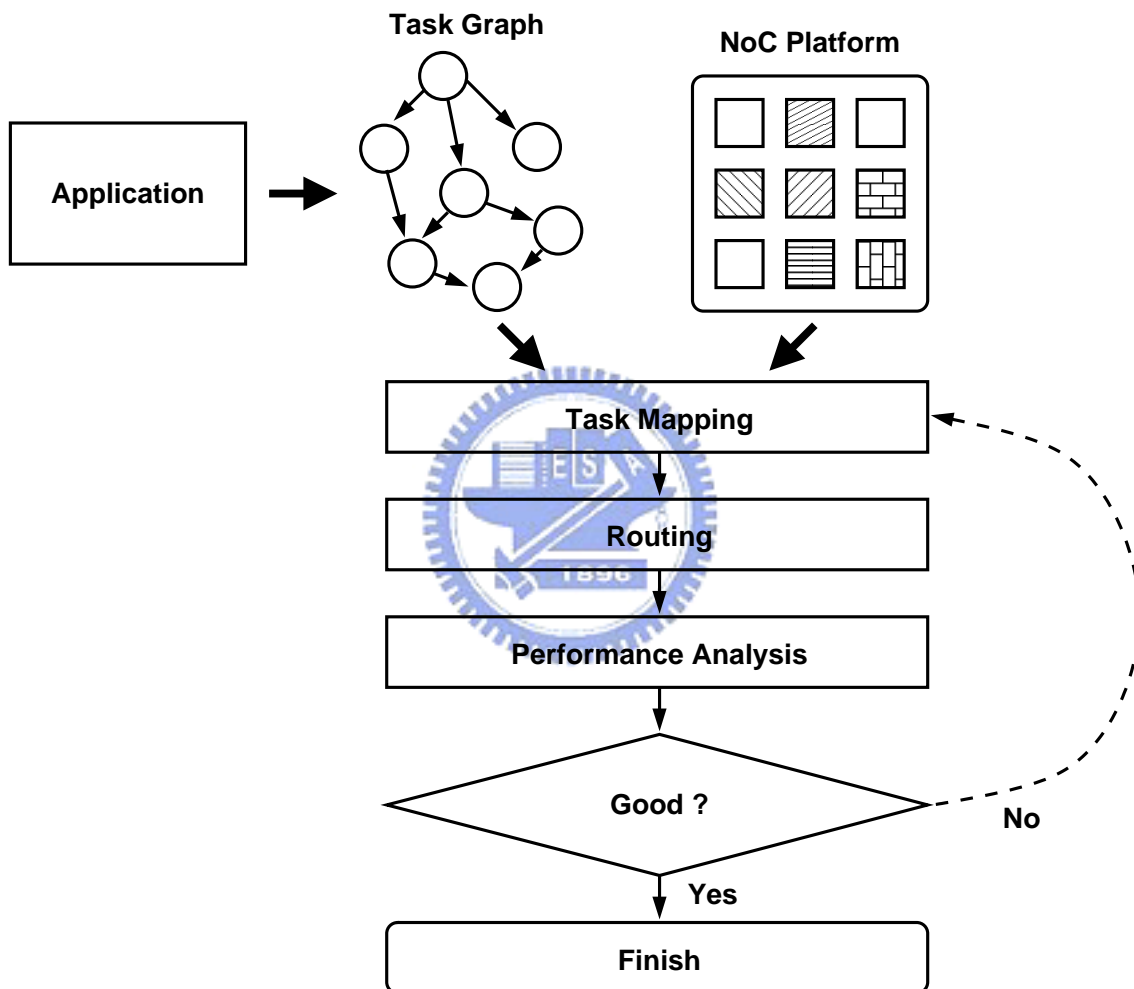


Figure 2.17: Proposed task binding methodology.

### 2.3.3 Task Mapping

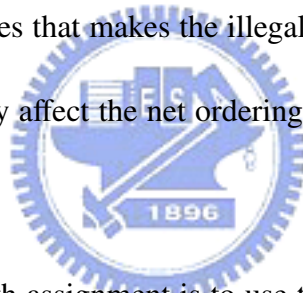
The task mapping of the proposed task binding method utilizes the simulated annealing technique since this technique is simple and suitable for diverse architectures developed in the on-chip networks. This mapping technique can be easily adapted to different cost requirements for the optimization. In this work, the goal of the task mapping is to minimize the overall communication resource usage. In the task mapping stage, because the routing is not applied, the path information in the physical channel cannot be obtained so that the simplified cost function as listed in (2.5) is derived by neglecting the contention factor  $\rho_X$  and the bandwidth penalty  $B_{penalty}$  in (2.1).

$$\xi = \sum_{\text{Path of pair of processors}} distance \times \left( 1 + \frac{C_{amount,X}}{C_{amount,MAX}} \right) \quad (2.5)$$

It is worthy of noting that we can consider (2.5) as the simplified version of (2.1). On the other hand, the distance in (2.5) is the Manhattan distance between the source node and the destination. The first term,  $(distance \times 1)$ , of the cost function describes the resource usage of the virtual channels. This term is also the conventional cost in the FPGA placement algorithm. In our mapping algorithm, not only the distance, but also the communication amounts between the tasks affect the system performance. The second term,  $(distance \times \frac{C_{amount,x}}{C_{amount,MAX}})$ , expressed as the normalized result represents the communication effect of the physical channel.

### 2.3.4 Connection Path Assignment

After finishing the task mapping, a router is used to assign the connection paths between any pair of interconnected processors. The algorithm proposed in [18] is applied to solve the routing problem. This router is essentially a variant of the maze router [19], where Dijkstra's algorithm [20] is applied to find the lowest cost path between the transmitting and the receiving processors. The Pathfinder algorithm [17] then performs multiple routing iterations to rip up some or all nets and reroute them by different paths if there exists a competition for routing resources that makes the illegal routing. Please note that ripping up and rerouting these nets only affect the net ordering. These nets are all routed by the same maze routing algorithm.



The cost function of the path assignment is to use the equation (2.1). There are two differences between cost functions (2.1) and (2.5). One is that the distance of the path assignment is the real routing length rather than the Manhattan distance used in the task mapping. The other difference is that the contention density and bandwidth penalty are included in the path assignment cost to describe the contention effect and total bandwidth introduced by other paths.

Fig. 2.18 shows the procedure of the path assignment algorithm. At the first, the all-shortest-path algorithm is applied to this path assignment. In Step 2, the overused virtual channels can be solved so that all the nets of the paths are routed. However, the overused

### **Path Assignment**

**Input:** Given an NoC architecture with the locations of transmitter and receiver. Number of virtual channel in a physical channel is assigned.

**Output:** Route all communication paths such that the communication cost is minimized.

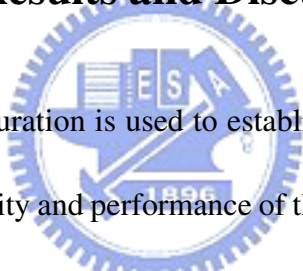
**Algorithm:**

1. The all shortest path algorithm
2. Fix virtual channel overuse
3. Sort paths by cost function
4. while (1)
5.     re-route the paths in orderly
6.     if (no improvement in this iteration)
7.         exit

Figure 2.18: Pseudo code of the path assignment.

bandwidth of some physical channels is not overcome in this step. From Step 3 to Step 7, these communication paths are redistributed into the physical channels to avoid overuse. The redistributed method is to find the path which has the maximum cost and then reroute this path to reduce its cost. The cost of this critical path may not be improved due to the possible heavy contention with other paths, so other paths need to be rerouted alternately depending on the measurement cost.

## 2.4 Experimental Results and Discussion



In this work, a 2D-mesh configuration is used to establish the network infrastructure for the experiments. The functionality and performance of the proposed platform with 4-by-4 nodes are evaluated. The traffic on this platform is generated by a random traffic generating function. The major performance evaluated in this experiment is the communication information of the platform. Then, the proposed task binding method is applied to the platform with the randomly generated task graph. Finally, the performance of our task binding algorithm will be evaluated based on this platform.

The proposed switch architecture is modeled in both cycle-accurate C++ and Verilog HDL. The C++ model is used for system design and the platform evaluation. The Verilog model is used for hardware design. After the synthesis with 0.25um standard CMOS technology, this switch can operate at 185 MHz in the typical-case corner.



In order to evaluate the traffic performance of the switch-based network platform, the C++ model of the switch is combined into the overall model of the network, and each original processing element (PE) is replaced by a random traffic pattern generator. This pattern generator generates random size packets which move from the arbitrary chosen source to the random destination.

The latency in this study means the elapsed time required for the data packet transmitted from the source node to the destination node [21]. Maximum latency is defined as the predicted worst case latency and the maximum latency can be obtained by Property 6. Normalized latency is defined as the latency divided by the maximum latency and normalized latency indicates the average performance. Injection rate is defined as the required bandwidth of the generated traffic divided by the guarantee bandwidth of a communication path. By changing the value of the injection rate, the different communication loads are available to evaluate the platform.

Fig. 2.19 shows the experimental histograms of the normalized latency versus different injection rates while each virtual channel of the platform only has a 2-word buffer. The network latency guarantee (normalized latency  $\leq 1$ ) is achieved even at the high injection rate (injection rate = 1). This means that the proposed NoC platform has the property of the minimum bandwidth guarantee for each transmission. The normalized latency approaches to zero when the injection rate decreases. This indicates that the average latency reduces as the injection rate decreases. With the property of the latency guarantee,

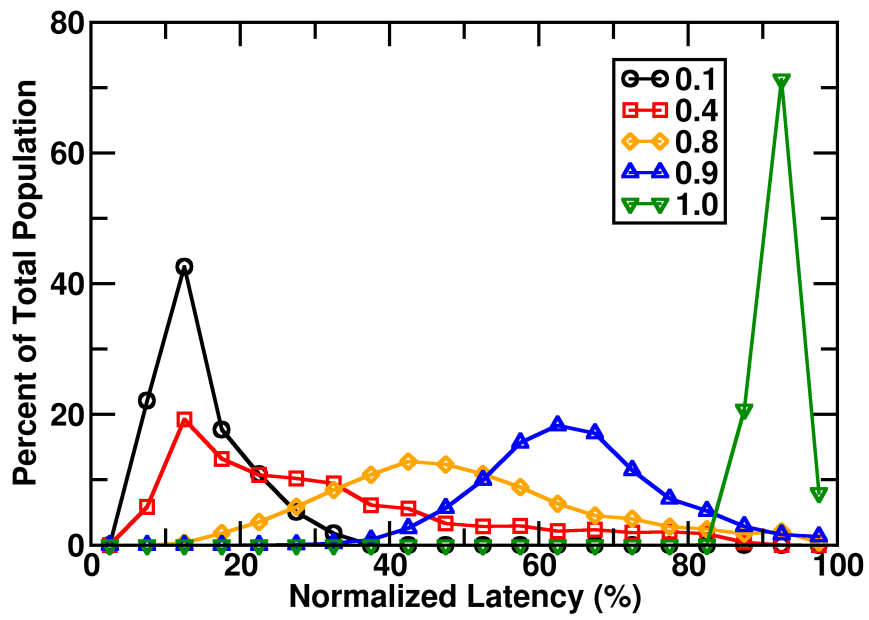


Figure 2.19: Histograms of normalized latency under different injection rate.

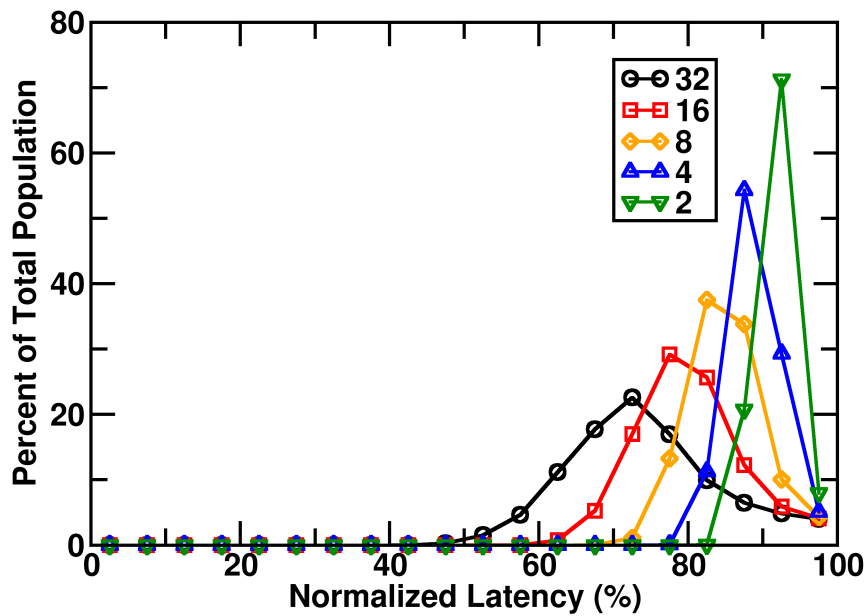


Figure 2.20: Histograms of normalized latency under different buffer size of virtual channel.

the predictability of the proposed platform can be obtained and the real-time systems can be realized.

Fig. 2.20 shows the normalized latency under the different buffer sizes of the virtual channels. The normalized latency reduces when the buffer size is increased. In general, the bigger buffer size of the virtual channels in the switches, the better communication performance can be achieved for the system in various applications.

In this switch-based network infrastructure, assume that the processor can begin to operate only when all input data are available and its output buffer size is enough for the data generated by itself.

The Task Graph For Free (TGFF) [22], a user-controllable, general-purpose, pseudo-random task graph generator, is employed to generate the task graphs used in the experiments. 100 task graphs are used, each task graph has at least 60 to 100 tasks, and the maximum inputs/outputs of each task are 7 to 10. The task graph is generated by TGFF and the communication amount is modified by multiplying the communication factor. Higher communication factor means the higher ratio of the communication and computation. On the other hand, the communication factor also indicates the provided physical bandwidth. The larger communication factor, the smaller physical bandwidth. In this experiment, the communication factor is set to 0.25, 0.5, 1, 2, and 4. The task graph with the lower communication factor implies that the application is computation intensive. Therefore, the higher factor means the application is communication intensive. Fail rate is defined as the number of fail transactions over the number of total transactions. Higher fail rate means that more power consumption is used for useless (meaningless) transaction and thus total power increases.

Fig. 2.21 shows the relationship between the fail rate and the communication factor by using traditional approach and the proposed one. The traditional approach is build without considering communication information. In Fig. 2.21, the fail rate increases along with the communication factor and saturates about 16%. This means that even under communication intensive, the fail rate is under-controlled. In Fig. 2.21, as compared with the traditional method, our proposed approach can improve the fail rate by 12.3% and

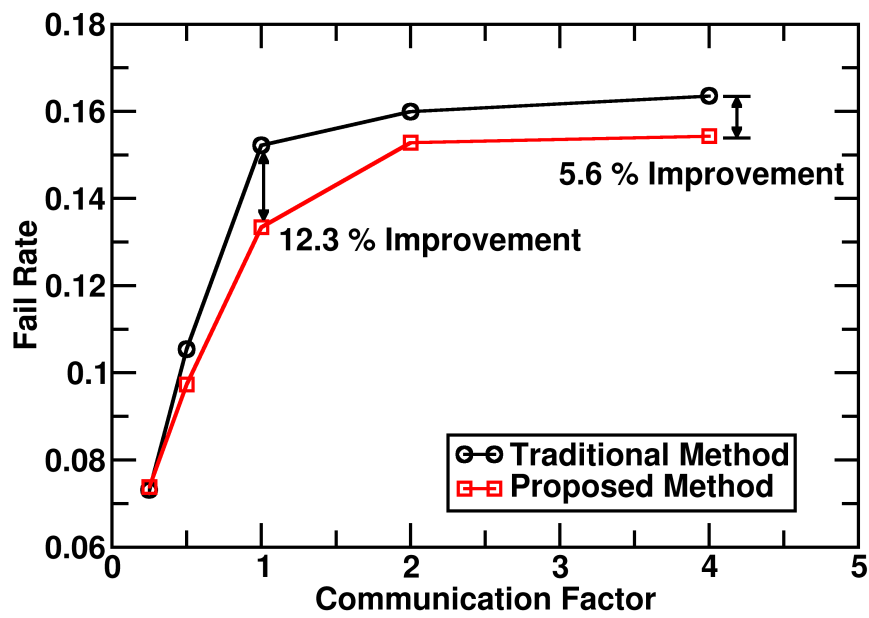


Figure 2.21: Fail rate under different communication factors.

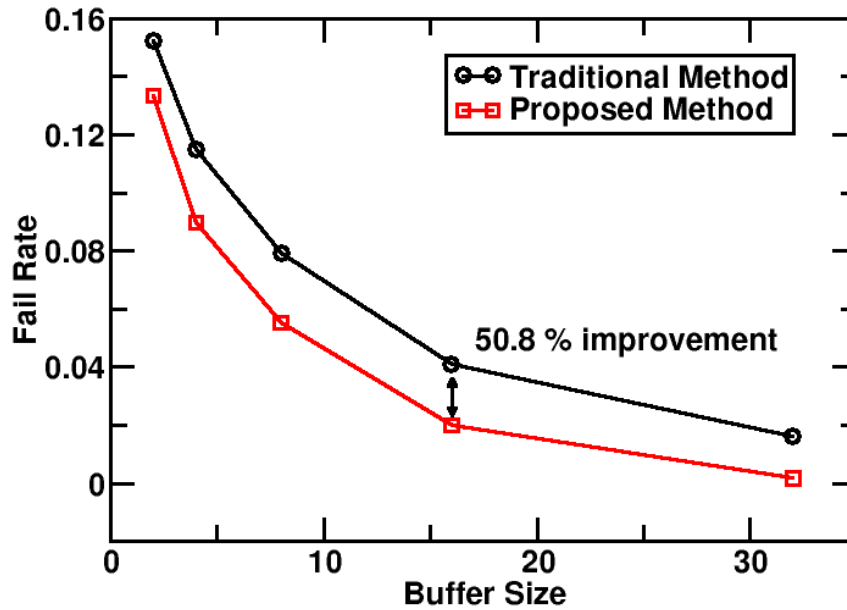


Figure 2.22: Fail rate under different buffer size.

5.6% with respect to the communication factor 1 and 4 respectively.

In order to reduce fail rate, the buffer size is increased. The result shows that the fail rate decreases along with the increasing buffer size and is shown in Fig. 2.22. The transaction is failed when the buffers of the destination are full. This means that larger buffer size gets lower fail probability. Compared with the traditional method, 50.8% improvement of fail rate versus 16 buffers can be obtained via the new approach.

Fig. 2.23 shows the communication latency versus the communication factor. The latency grows linearly with the increasing communication factor. In Fig. 2.23, as compared with the traditional method, our proposed approach can improve the latency by 9.8% with

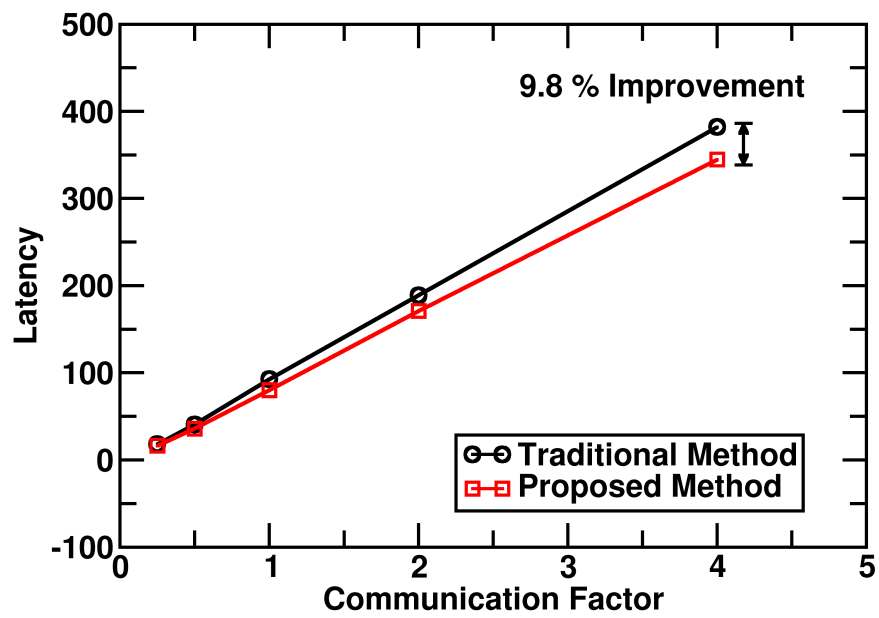


Figure 2.23: Latency under different communication factors.

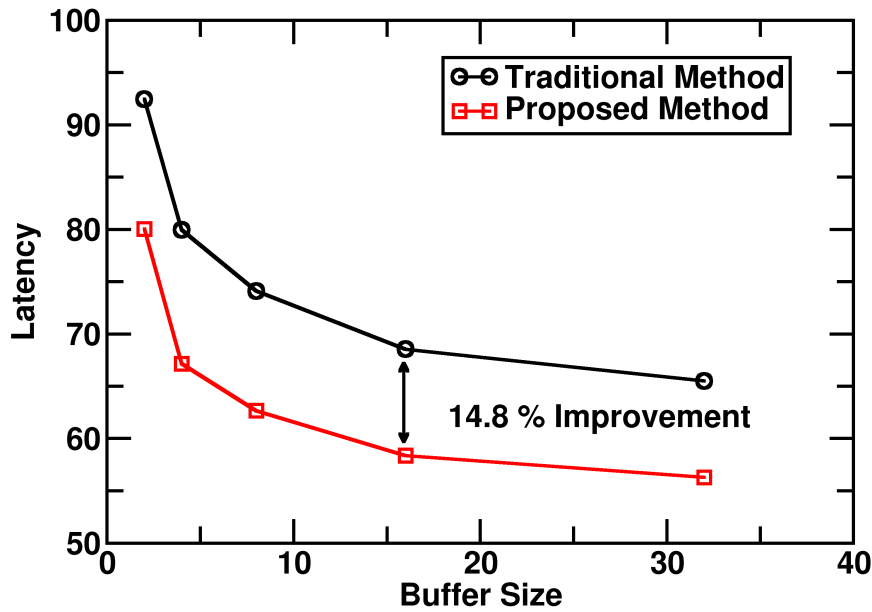


Figure 2.24: Latency under different buffer size.

respect to the communication factor 4.

Increasing the buffer size can also reduce the latency. As shown in Fig. 2.24, the latency decreases when the buffer size increases. This also means that lower fail rate gets smaller latency. Compared with the traditional method, 14.8% improvement of latency versus 16 buffers can be obtained via the new approach.

Fig. 2.25 shows the comparison of the throughput ratio versus the communication factor between the proposed algorithms. The throughput ratio is defined as the throughput resulted from the proposed algorithm divided by this one obtained from the traditional approach. The first curve shows the result by applying (2.5). Then, add contention factor



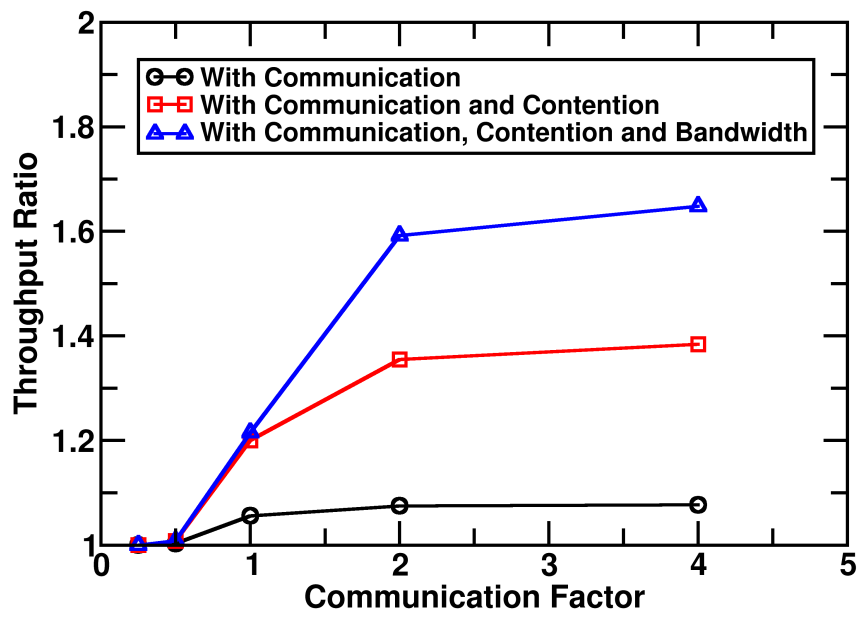
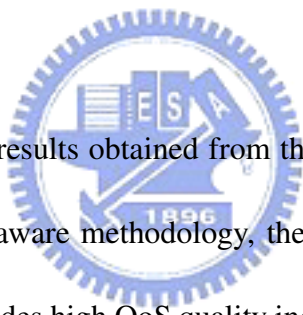


Figure 2.25: Throughput ratio under different communication factor.

in the connection path assignment in the second curve. Finally, add bandwidth penalty in the connection path assignment to improve the performance under heavy load. In the computation-intensive task graph, the corresponding throughputs are almost equal under different algorithms and this implies that the communication effect can be neglected. On the other hand, in the communication-intensive task graph, the bandwidth penalty becomes dominating because the required bandwidth is more than the provided bandwidth. To balance the bandwidth becomes the most important issue. As shown in Fig. 2.25, the proposed algorithm improve throughput 20% under normal communication (communication factor = 1).



From the above simulation results obtained from the new model of switch-based architecture and communication-aware methodology, the proposed approach outperforms the traditional method and provides high QoS quality including high-throughput, latency-insensitive, bandwidth guarantee, and high memory utilization.

## 2.5 Summary

This work proposes a switch-based network platform design that adopts the hybrid of the latency-insensitive concept, virtual-circuit switching, weighted round-robin scheduling, and pipeline bus. The platform has the latency guarantee and the low average latency. The proposed task binding algorithm employs the iterative profile-driven optimization

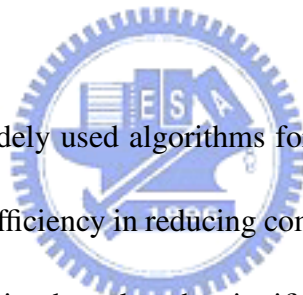
technique to reduce the effect of the communication amount and the communication contentions, so that the high system throughput is achieved. The experimental results indicate that the task binding approach increases the system utilization and effectively improves the network throughput up to 20% on average.





# Chapter 3

## FFT Processor



The FFT is one of the most widely used algorithms for calculating the Discrete Fourier Transform (DFT) owing to its efficiency in reducing computation time [23]. Recently, the FFT requiring real-time processing has played a significant role in many communication systems based on Orthogonal Frequency Division Multiplexing (OFDM) technology such as HDTV, xDSL modems and wideband mobile terminals.

Pipelined FFT implementations are highly appropriate for real-time applications since pipelined FFT can be easily merged with the sequential nature of sampling. Several FFT architectures were developed, such as Radix-2 Multi-path Delay Commutator (R2MDC) [24], Radix-2 Single-path Delay Feedback (R2SDF) [25], Radix-2<sup>2</sup> Single-path Delay Feedback (R2<sup>2</sup>SDF) [26][27], Radix-4 Single-path Delay Feedback (R4SDF) and Radix-4 Multi-path Delay Commutator (R4MDC) [24]. Among these architectures, delay feed-

back approaches are always more efficient than the corresponding delay commutator approaches in terms of required memory size [26] [28]. The R4SDF requires fewer multipliers than those required by R2SDF; however, the R2SDF architecture is simple and regular. The R2<sup>2</sup>SDF architecture is a compromise endowed with the R2SDF structure and the multiplicative complexity of the R4SDF. This study focuses on R2SDF and R2<sup>2</sup>SDF architectures.

Since the pipeline FFT architecture is memory-consuming, reducing its memory requirement will save a significant amount of chip area. Several studies have employed regular module implementations and have attempted to reduce the area-consuming elements in the FFT design. The design of [29] reduces the amount of memory used to store the twiddle factors by employing canonic signed digit (CSD) constant multipliers. A new FFT architecture, the radix-2 single deep delay feedback (R2SD<sup>2</sup>SF) presented in [30], has smaller complex multipliers and adders than other FFT designs. Both the designs of [29] and [30] have fixed wordlength for data and coefficients for each pipeline stage. The possibility to use varying wordlengths for these stages is frequently ignored when achieving modularized solutions. However, the increasing use of intellectual property (IP) makes the non-module implementation viable, allowing for the further exploitation of pipelined architectures.

In general, an FFT cannot be implemented exactly. Each multiplier and adder in the pipelined FFT architecture can introduce errors due to rounding or truncation of arith-

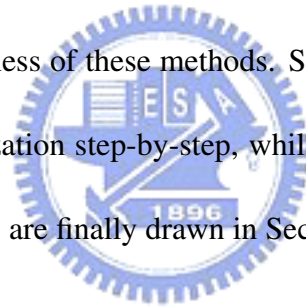
metic results. Errors typically accumulate successively over FFT stages. That is, errors from early stages can affect performance in latter stages. The wordlengths of data and coefficients chiefly affect precision, quantization errors, and hardware complexity. Increased wordlengths increase the precision and reduce quantization error at the cost of area and power. Conversely, to maintain a lower hardware cost, a shorter wordlength can be chosen at the sacrifice of precision. Therefore, identifying an optimized solution of wordlength is necessary.

Two conventional methods for FFT error analysis of signal to quantization noise ratio (SQNR) and wordlengths are statistical error analysis and simulation-based analysis. Although the SQNR can be calculated efficiently by employing statistical models [31] [32] [33], the accuracy of the calculated result heavily depends on the model used. A more precise model yields more accurate results. The simulation-based method evaluates the FFT by comparing simulation results of the fixed-point computations with those obtained using the floating-point arithmetic [34]. Although simulation increases the accuracy of the evaluation results, it is time-consuming.

According to error analysis, optimizing wordlengths of pipeline stages in FFT processors for given specifications is feasible. Optimization of an 8192-point FFT processor using the simulation method has shown that progressive wordlengths and scaling in the early stages can achieve a good compromise between SQNR and hardware cost [35]. However, this approach requires a long time to run the simulation.

This work presents a statistical model for error analysis at the stage level with varying wordlengths in the pipeline FFT processor. Furthermore, a hybrid method for reducing the required simulation time is introduced. The optimized wordlength parameters at each stage are generated automatically according to design specifications of FFT processors, such as the length of FFT, SQNR and the real-time processing requirements. Finally, the optimization flow using the proposed error model and the hybrid method is demonstrated.

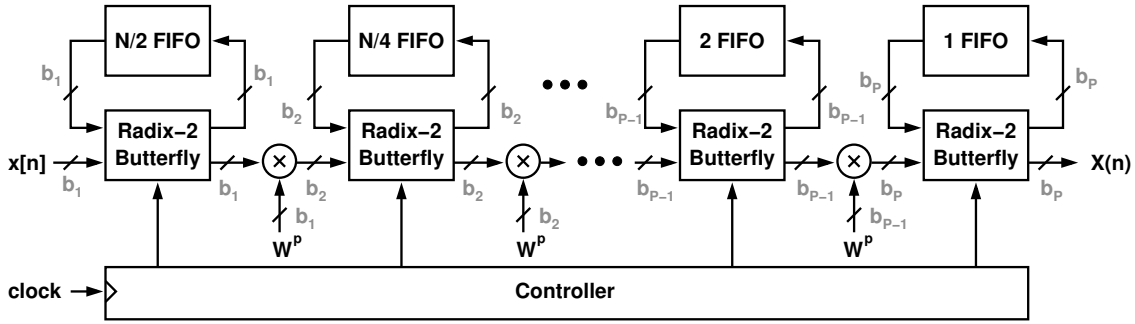
The rest of this chapter is organized as follows. Section 3.1 gives a brief review of the FFT. Section 3.2 then introduces statistical and simulation-based error analyses and demonstrates the effectiveness of these methods. Section 3.3 describes the proposed method for wordlength optimization step-by-step, while Section 3.4 summarizes the experimental results. Conclusions are finally drawn in Section 3.5.



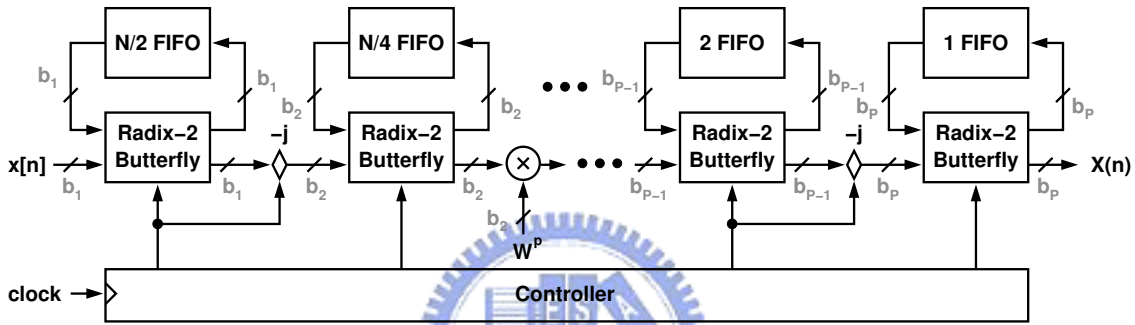
### 3.1 Overview of FFT

An FFT based on structuring the DFT computation by forming increasingly smaller subsequences of the input sequence  $x[n]$  is called a decimation-in-time (DIT) FFT. Alternatively, an FFT can also be decomposed using a first-half/second-half approach that divides the output sequence  $X(r)$  into increasingly smaller subsequences; this procedure is called a decimation-in-frequency (DIF) FFT [36]. Since both of these schemes are similar in nature, their performance cannot be exactly compared without a given architecture [33].





(a) R2SDF architecture.



(b) R2<sup>2</sup>SDF architecture.

Figure 3.1: Conventional R2SDF and R2<sup>2</sup>SDF DIF implementations.

In this work, the DIF algorithm is used to illustrate the architectural implementations.

This work examines the architectures of R2SDF and R2<sup>2</sup>SDF for the fixed-point DIF pipeline FFT processor to demonstrate the effectiveness of the proposed optimization method. Their block diagrams are shown in Fig. 3.1, where  $N$  is FFT length,  $b_k$  is the wordlength of stage  $k$ ,  $k \in \{1, 2, \dots, P\}$ , and  $P = \log_2 N$ . Due to spatial regularity, both controllers in these architectures can be implemented by using simple  $P$ -bit counters [25] [27]. Since the valid output range of the  $\pm$  operation of the FFT butterfly is double that of the valid input range, a scaling by  $1/2$  is applied to eliminate the overflow. With-

out scaling by  $1/2$ , the overflow will cause excessive error. Therefore, scaling by  $1/2$  is employed for each stage in this work.

Although the area and power consumption in these pipeline architectures are dominated by memory (FIFOs) and multipliers, to progressively adjust the wordlength of pipeline stages can reduce the area of memory and multipliers and hence the overall power consumption. To adjust wordlengths based on maintaining the SQNR requirement, error analysis is required.

## 3.2 Error Analysis

### 3.2.1 Statistical Analysis

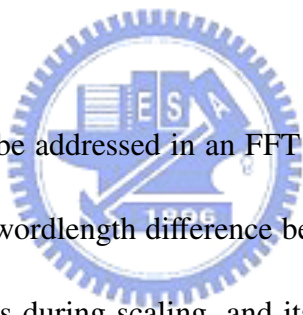


This section introduces the statistical error model for varying wordlengths of pipeline stages. The precision of the FFT processor is then discussed according to the SQNR derivation. This derivation has two major steps. The first step involves finding error sources based on the architectures adopted and the fixed-point arithmetic schemes, e.g., truncation, and scaling by  $1/2$  or not. The next step entails searching the paths of error propagation and combining all of these errors along paths to evaluate the variance of the errors propagating to the output. Moreover, the SQNR of the FFT output is given.

In the following analyses, two architectures, R2SDF and R2<sup>2</sup>SDF, are illustrated for deriving the statistical error models. Assume a fixed-point arithmetic with  $(b_k + 1)$  bit

wordlengths and a signed fraction, where  $k$  is the stage number of PE stage. The input to an  $N$ -point FFT, denoted by  $x[n]$ , where  $n = 0, 1, 2, \dots, N - 1$ , is a sequence of finite valued complex numbers. Numbers consist of  $2N$  real random variables that are uncorrelated and are uniformly distributed in  $(-1/\sqrt{2}, 1/\sqrt{2})$ . One db is added to the SQNR constraint (iSQNR) to allow for the SQNR error in the statistical model. The effect of inaccuracy in the twiddle factor,  $W^p$ , is not addressed here. The truncation operations are modeled as uncorrelated.

### Error Sources



Four major error sources must be addressed in an FFT processor. The first error source is the quantization error of the wordlength difference between PE stages whose variance is  $\sigma_{q,k}^2$ . The second error occurs during scaling, and its variance is represented by  $\sigma_{s,k}^2$ . Another error results from the complex multiplication of the twiddle factor, and  $\sigma_{m,k}^2$  is used to denote its variance. The last error, the insufficient output wordlength error  $\sigma_{q,o}^2$ , is only considered for the last stage of the FFT processor.

Fig. 3.2 shows the error model of a PE stage with stage-by-stage scaling of  $1/2$ .  $\sigma_{q,k}^2$  occurs when the wordlength,  $b_{k-1}$ , of the stage  $k - 1$  is longer than that of the stage  $k$ .  $\sigma_{q,k}^2$  is the variance of the truncated bits from  $b_{k-1}$  to  $b_k$ . Scaling error is produced when  $b_k < b_{k-1} + 1$ . Scaling by a factor of  $1/2$  involves one bit right shift and truncation of the last significant bit (LSB).  $\sigma_{s,k}^2$  is defined as the variance of this truncated bit. Both  $\sigma_{q,k}^2$

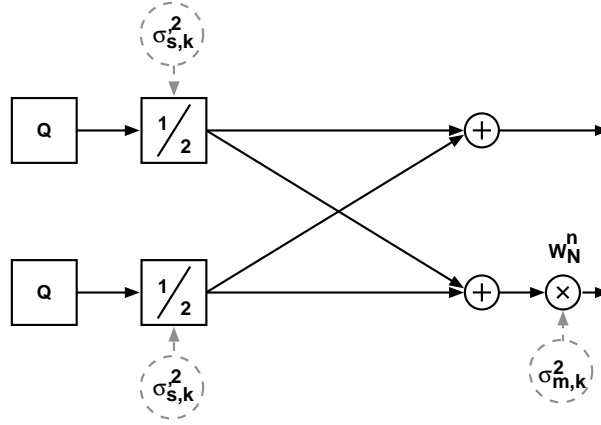


Figure 3.2: Error model of a PE stage.

and  $\sigma_{s,k}^2$  can be combined as an error of directly scaling the output data of stage  $k - 1$  and truncating the scaled result to  $b_k$  bits. Since complex scaling can be achieved by separately scaling the real and imaginary parts of the data, the combined error  $\sigma_{s,k}^2$  (Fig. 3.2) can be expressed as

$$\sigma_{s,k}^2 = \begin{cases} 0 & ; b_{k-1} + 1 \leq b_k \\ 2 \times \left( \frac{1}{M} \cdot 2^{-2(b_{k-1}+1)} \cdot \sum_{v=0}^{M-1} v^2 \right) & ; b_{k-1} + 1 > b_k \end{cases} \quad (3.1)$$

where  $M = 2^{(b_{k-1}+1)-b_k}$ .

If a complex multiplication is implemented by using four real multiplications and the results of real multiplications are truncated individually,  $\sigma_{m,k}^2$  is defined as the variance of truncated bits of the result after finishing a complex multiplication. This variance can be represented by wordlengths,  $b_{k-1}$  and  $b_k$ , and is obtained as

$$\sigma_{m,k}^2 = \begin{cases} 4 \times \left( \frac{1}{M} \cdot 2^{-2(b_{k-1}+1+b_k)} \cdot \sum_{v=0}^{M-1} v^2 \right), M = 2^{(b_{k-1}+1+b_k)-b_k} ; b_{k-1} + 1 \leq b_k \\ 4 \times \left( \frac{1}{M} \cdot 2^{-2 \cdot 2b_k} \cdot \sum_{v=0}^{M-1} v^2 \right), M = 2^{2b_k-b_k} ; b_{k-1} + 1 > b_k \end{cases} \quad (3.2)$$

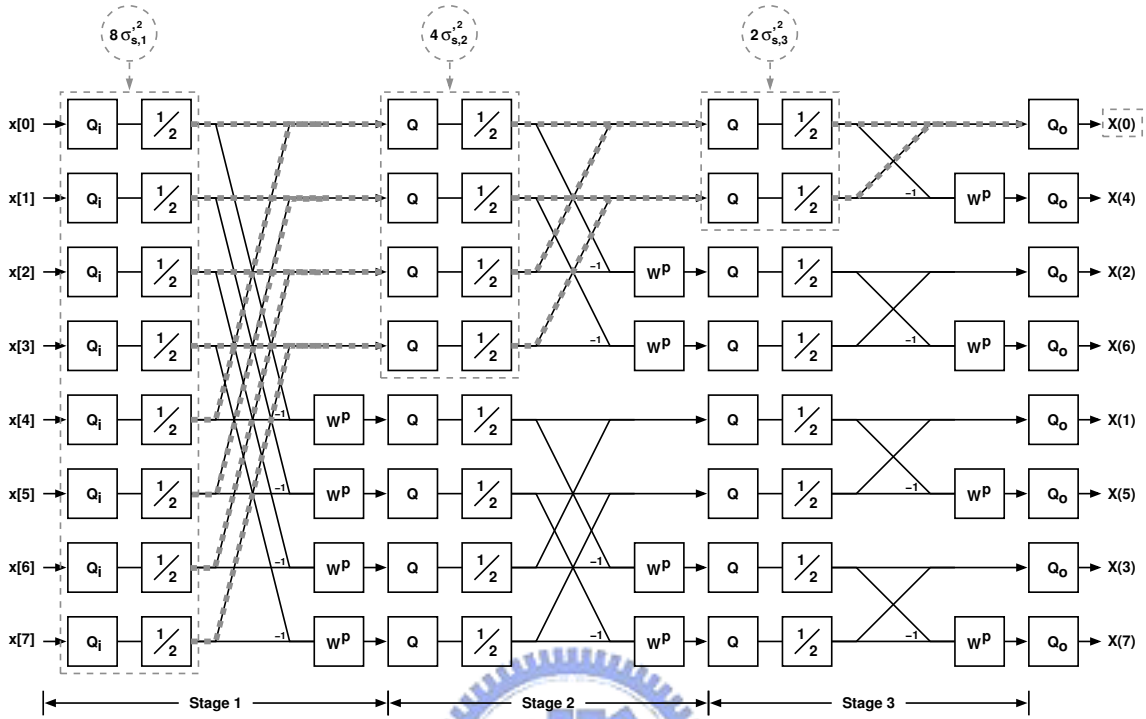


Figure 3.3: Propagation of quantization and scaling errors.

If the required output wordlength,  $b_o$  of the FFT processor, is too short, the output of the last PE stage must be truncated. The quantization error is then generated and its variance,  $\sigma_{q,o}^2$ , can be described by  $b_o$  and wordlength of the last stage,  $b_p$ ; the formula is expressed as

$$\sigma_{q,o}^2 = \begin{cases} 0 & ; b_p \leq b_o \\ 2 \times \left( \frac{1}{M} \cdot 2^{-2b_p} \cdot \sum_{v=0}^{M-1} v^2 \right) & ; b_p > b_o \end{cases} \quad (3.3)$$

where  $M = 2^{b_p - b_o}$ .

### Output Signal-to-Quantization Noise Ratio (SQNR)

Since all error sources are assumed uncorrelated, the variance of the errors at the FFT output can be obtained by summing all contributions from the individual error sources that propagate to the output. For an  $N$ -point FFT processor employing either the Radix-2 algorithm or Radix- $2^2$  algorithm, the scaling error  $\sigma_{s,k}^2$  propagating to any output node can be given by

$$\begin{aligned}\sigma_S^2 &\approx N \left(\frac{1}{4}\right)^{P-1} \times \sigma_{s,1}^2 + \frac{N}{2} \left(\frac{1}{4}\right)^{P-2} \times \sigma_{s,2}^2 + \cdots + \frac{N}{2^{P-1}} \left(\frac{1}{4}\right)^0 \times \sigma_{s,P}^2 \\ &= \sum_{k=1}^P \frac{N}{2^{k-1}} \cdot \left(\frac{1}{4}\right)^{P-k} \times \sigma_{s,k}^2\end{aligned}\quad (3.4)$$

where  $(1/4)^{P-k}$  is the effect of scaling on the error propagating at stage  $k$ . The  $\sigma_{s,k}^2$  propagation can be illustrated using an 8-point DIF Radix-2 algorithm signal flow graph (SFG) (Fig. 3.3). The number of scaling errors  $\sigma_{s,k}^2$  propagating to any output node, e.g.,  $X(0)$ , from the first, second, and third stages are 8, 4, and 2 respectively. Thus, the error variance of  $X(0)$  can be obtained from (3.4), and is given by  $\sigma_{S|X(0)}^2 = (1/2)\sigma_{s,1}^2 + \sigma_{s,2}^2 + 2\sigma_{s,3}^2$ .

To derive the variance of the FFT output due to multiplication errors, all multiplications are assumed noisy. Fig. 3.4 shows an 8-point DIF Radix-2 algorithm SFG. There are 4, i.e., half of 8,  $\sigma_{m,k}^2$ s in each stage, and each  $\sigma_{m,k}^2$  of the first ( $k = 1$ ), second ( $k = 2$ ), and last ( $k = 3$ ) stage will propagate to 4, 2, and 1 output nodes, respectively. On the other hand, for a general case with  $N$ -point FFT, there are half of the  $N$   $\sigma_{m,k}^2$ s error sources in

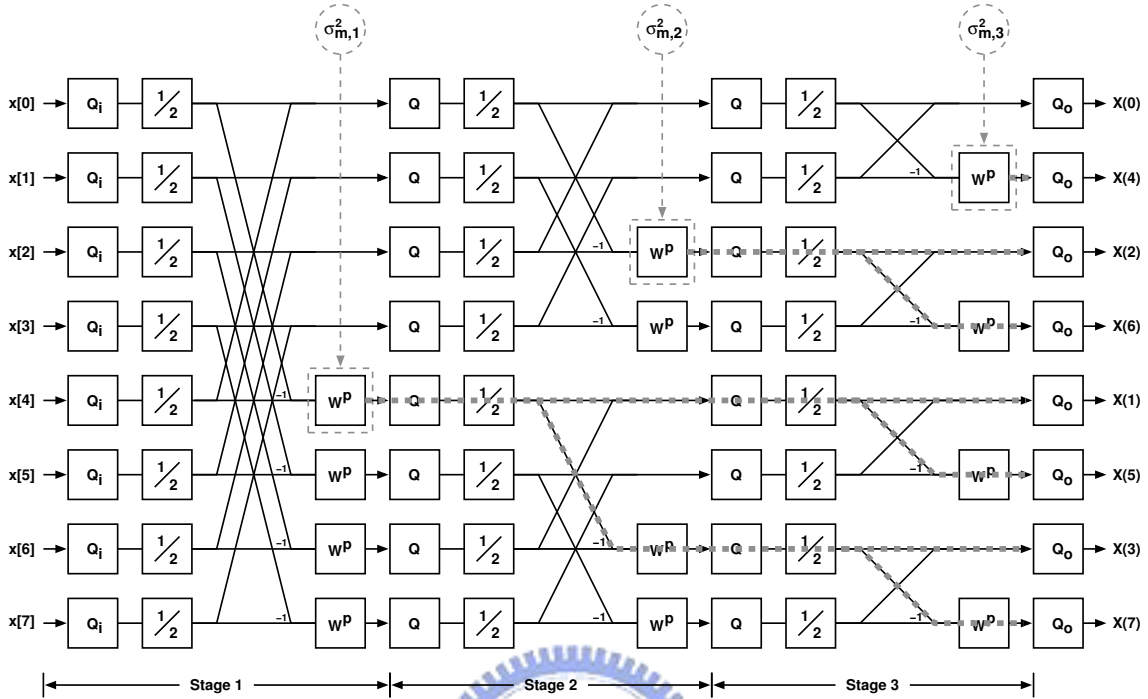


Figure 3.4: Propagation of multiplication errors.

each stage and each  $\sigma_{m,k}^2$  from the first stage to the last,  $P$ -th, stage propagates to  $\frac{N}{2}, \frac{N}{4}, \dots, \frac{N}{2^P}$  output data, respectively. Hence, one can easily derive the output variance caused by multiplication errors,  $\sigma_M^2$ , with the Radix-2 algorithm; the expression of  $\sigma_M^2$  is given by

$$\begin{aligned} \sigma_M^2 &\approx \frac{1}{N} \cdot \frac{N}{2} \left[ \frac{N}{2} \left(\frac{1}{4}\right)^{P-1} \cdot \sigma_{m,1}^2 + \frac{N}{2^2} \left(\frac{1}{4}\right)^{P-2} \cdot \sigma_{m,2}^2 + \dots + \frac{N}{2^P} \left(\frac{1}{4}\right)^0 \cdot \sigma_{m,P}^2 \right] \\ &= \frac{1}{2} \times \sum_{k=1}^P \frac{N}{2^k} \cdot \left(\frac{1}{4}\right)^{P-k} \times \sigma_{m,k}^2 \end{aligned} \quad (3.5)$$

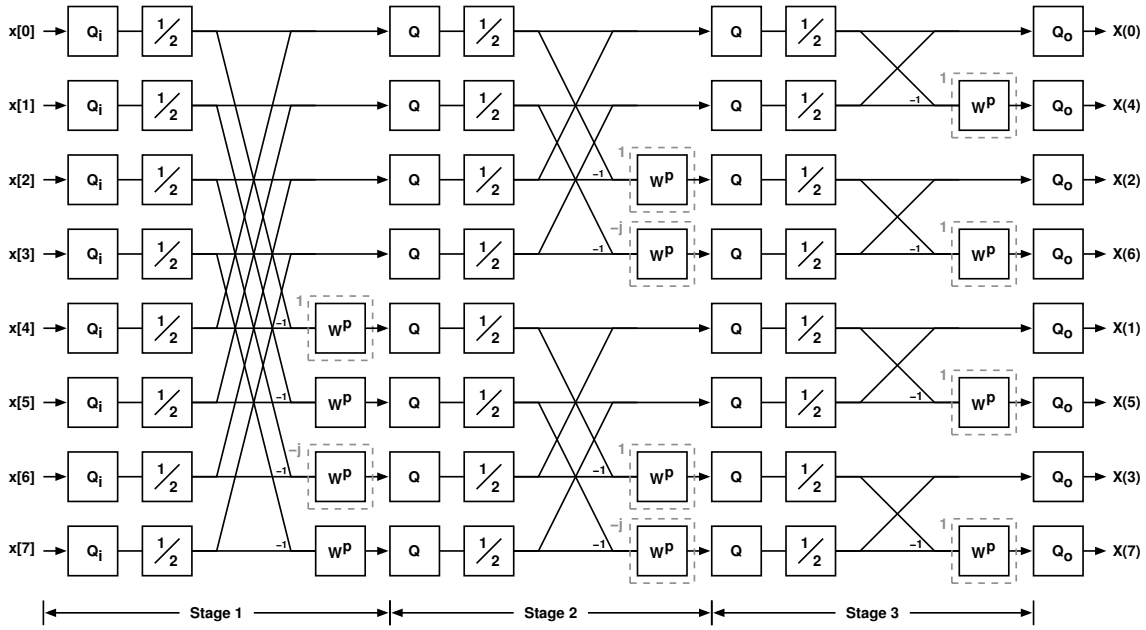


Figure 3.5: Propagation of noiseless multiplications.

For the Radix-2<sup>2</sup> algorithm, the corresponding expression of  $\sigma_M^2$  is modified as

$$\begin{aligned}
 \sigma_M^2 &\approx \frac{1}{N} \cdot \frac{3N}{4} \left[ \frac{N}{4} \left(\frac{1}{4}\right)^{P-2} \sigma_{m,2}^2 + \frac{N}{4^2} \left(\frac{1}{4}\right)^{P-4} \sigma_{m,4}^2 + \cdots + \frac{N}{4^{P/2}} \left(\frac{1}{4}\right)^0 \sigma_{m,P}^2 \right] \\
 &= \frac{3}{4} \times \sum_{k=1}^{P/2} \frac{N}{4^k} \cdot \left(\frac{1}{4}\right)^{P-2k} \times \sigma_{m,2k}^2 \quad (3.6)
 \end{aligned}$$

(3.5) and (3.6) are derived by assuming that all the multiplications are noisy. However, the multiplications associated with twiddle factors  $W^p = \pm 1$  or  $W^p = \pm j$  introduce no errors. Fig. 3.5 shows the position of noiseless multiplications in an 8-point Radix-2 algorithm SFG. The variances of these noiseless multiplications denoted as  $\sigma_C^2$  in the Radix-2 algorithm SFG and the Radix-2<sup>2</sup> algorithm SFG, are individually re-derived and



can be given by (3.7) and (3.8).

$$\begin{aligned}
\sigma_C^2 &\approx \frac{1}{N} \times \left[ 2 \cdot \frac{N}{2} \left(\frac{1}{4}\right)^{P-1} \cdot \sigma_{m,1}^2 + 2^2 \cdot \frac{N}{2^2} \left(\frac{1}{4}\right)^{P-2} \cdot \sigma_{m,2}^2 + \dots \right. \\
&\quad \left. + 2^{P-1} \cdot \frac{N}{2^{P-1}} \left(\frac{1}{4}\right) \cdot \sigma_{m,P-1}^2 \right] + \frac{1}{N} \times \left[ \frac{N}{2} \times \frac{N}{2^P} \cdot \sigma_{m,P}^2 \right] \\
&= \left[ \sum_{k=1}^{P-1} \left(\frac{1}{4}\right)^{P-k} \cdot \sigma_{m,k}^2 \right] + \frac{1}{2} \times \frac{N}{2^P} \cdot \sigma_{m,P}^2 \tag{3.7}
\end{aligned}$$

$$\begin{aligned}
\sigma_C^2 &\approx \frac{1}{N} \cdot \frac{3N}{4} \times \left[ \frac{N}{4} \cdot \left(\frac{1}{2^{P-2}}\right) \cdot \left(\frac{1}{4}\right)^{P-2} \sigma_{m,2}^2 \right. \\
&\quad \left. + \frac{N}{4^2} \cdot \left(\frac{1}{2^{P-4}}\right) \cdot \left(\frac{1}{4}\right)^{P-4} \sigma_{m,4}^2 + \dots + \frac{N}{4^{P/2}} \cdot \sigma_{m,P}^2 \right] \\
&= \frac{3}{4} \times \sum_{k=1}^{P/2} \frac{N}{4^k} \cdot \left(\frac{1}{2^{P-2k}}\right) \times \left(\frac{1}{4}\right)^{P-2k} \cdot \sigma_{m,2k}^2 \tag{3.8}
\end{aligned}$$

According to the assumption of no correlations in these error sources, the total output error variance can be obtained by summing the variance of each error propagating to the output; this summation is expressed as

$$\sigma_T^2 = \sigma_S^2 + (\sigma_M^2 - \sigma_C^2) + \sigma_{qo}^2 \tag{3.9}$$

Furthermore, the variance of output data in an N-point FFT processor is given in (3.10) [36].

$$\sigma_X^2 = \frac{1}{3N} \tag{3.10}$$

Hence, the output SQNR is obtained by

$$SQNR_1 = 10 \log_{10} \left( \frac{\sigma_X^2}{\sigma_T^2} \right) \tag{3.11}$$

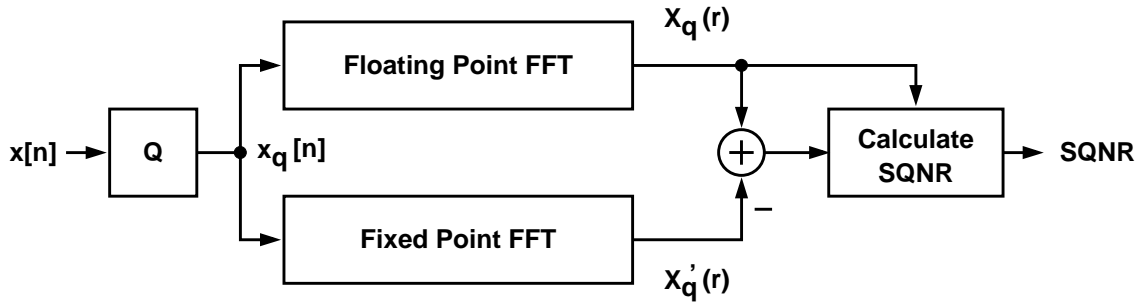


Figure 3.6: Block diagram of simulation analysis.

This  $SQNR_1$  model is used as the performance index, whereas statistical error analysis is employed in the FFT processor.

### 3.2.2 Simulation-Based Method

Fig. 3.6 presents a conceptual block diagram of the simulation-based analysis. To perform this simulation, floating-point and fixed-point C models with a given FFT algorithm were developed. According to system constraints, e.g., wordlength of each stage, rounding or truncation of stages, number of scaling stages, input/output wordlength, the C models obtain the proper fixed-point results. Then, SQNR can be evaluated by comparing the fixed-point output with the floating-point output; the formula of the calculation is given by

$$SQNR_2 = 10 \log_{10} \frac{\sum_{r=0}^{N-1} [X_q(r)]^2}{\sum_{r=0}^{N-1} [X_q(r) - X'_q(r)]^2} \quad (3.12)$$

During simulation, random patterns are generated as inputs, and then the resulting  $SQNR_2$ 's are averaged as an estimated average of the SQNR distribution. During sim-

ulating analysis, there is a trade-off between the accuracy of the  $\overline{SQNR}_2$  and the required number of simulation times. The required simulation times are investigated as follows. First, the random variable  $\bar{S}$  is used as an estimate of  $\overline{SQNR}_2$ . Then, according to the central limit theorem, the sampling distribution of  $\bar{S}$  is approximately normally distributed. Therefore, we can be  $(1 - \alpha)100\%$  confident that the SQNR error will not exceed a specified amount,  $e$ , when the number of  $SQNR_2$ 's equals

$$\left(\frac{z_{\alpha/2} \cdot \sigma}{e}\right)^2 \quad (3.13)$$

where  $\sigma$  is the standard deviation of the distribution of  $SQNR_2$ , and  $z_{\alpha/2}$  satisfies the probability equation,  $\text{Prob}(z_{\alpha/2} < Z) = \alpha/2$ , when  $Z$  is a random variable with a standard normal distribution [37]. In this work,  $e$  is the constraint of SQNR error (SQNR\_Error).

### 3.2.3 Demonstration of Statistical and Simulation-Based Analysis

The proposed error model was verified using the simulation-based error analysis and the result was compared with that obtained by statistical error analysis. In this analysis, the SQNR is calculated using statistical method and is also evaluated using simulation for 8-, 16-,  $\dots$ , 8192-point DIF Radix-2 FFT and for 16-, 64-,  $\dots$ , 4096-point DIF Radix-2<sup>2</sup> FFT with the freely chosen wordlength from 8 – 32 bits for each stage.

Table 3.1 shows a summary of the comparison between the two methods with 20 randomly generated wordlength sets in a 1024-point DIF Radix-2 FFT, where input wordlength

Table 3.1: Example of Random Verification

Wordlength Set no.	Wordlength of PE Stage										SQNR (dB)		
	1	2	3	4	5	6	7	8	9	10	Simulation	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

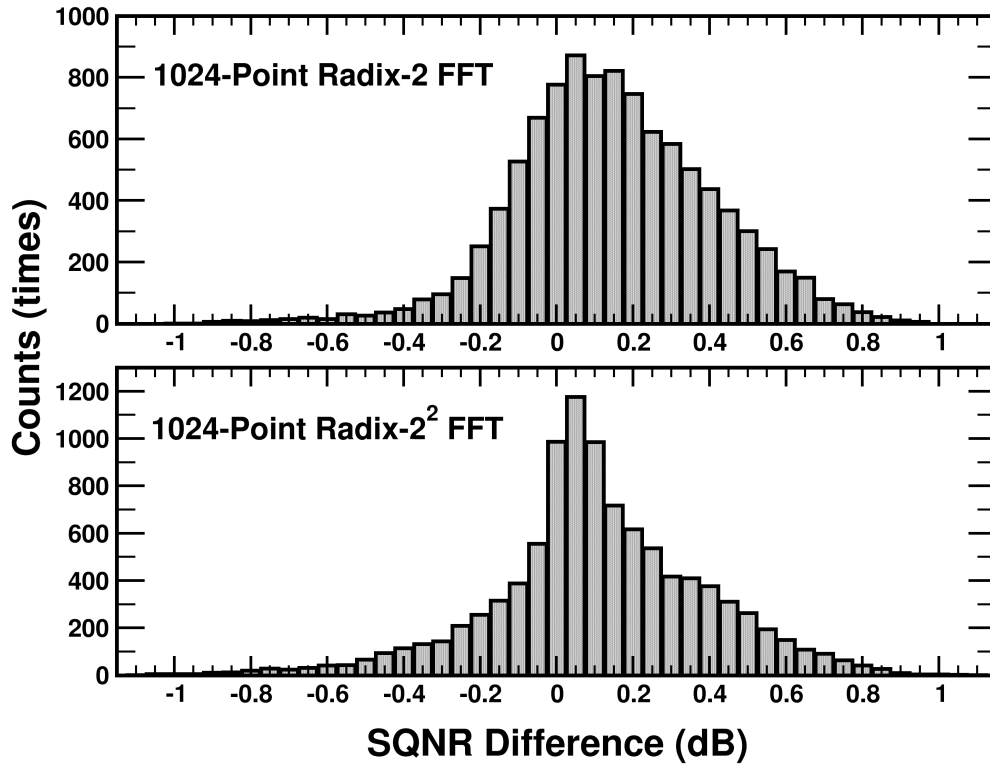


Figure 3.7: Histogram of SQNR difference with randomly generated wordlengths.

is set to be equal to that of the first stage, and output wordlength is set to be the same as that of the last stage. The SQNR difference is obtained by subtracting the SQNR of simulation analysis from that of statistical analysis.

Fig. 3.7 shows the histogram of SQNR difference with  $10^4$  randomly generated wordlength sets for the 1024-point FFT of Radix-2 and Radix-2<sup>2</sup> algorithm. The difference in comparison is within  $\pm 1.0$  dB in Radix-2 FFT and within  $\pm 1.1$  dB for the Radix-2<sup>2</sup> FFT.

Exhaustively comparing all wordlength sets of 8 – 32 bits is impractical because the simulation time is unendurable. Therefore, partial exhaustive verification for wordlengths

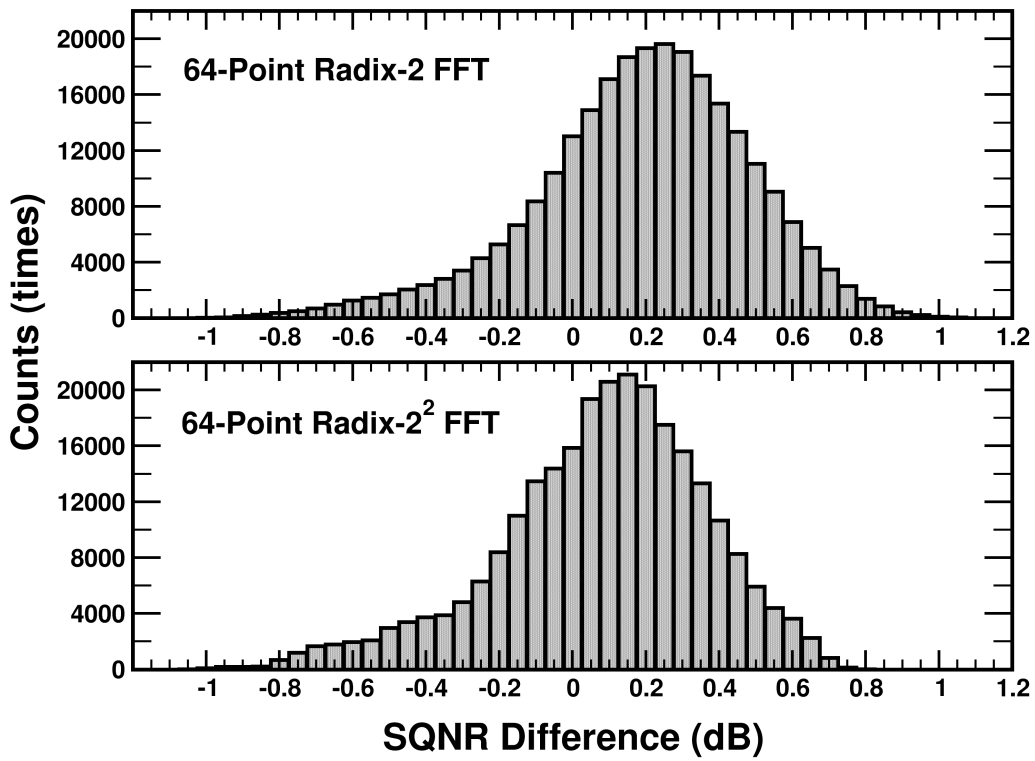


Figure 3.8: Histogram of SQNR difference with partial exhaustive verification.

of 11 – 18 bits was employed in the comparison of 64-point Radix-2 and Radix-2<sup>2</sup> FFT. This comparison required 130 hours. Fig. 3.8 shows the results of the comparison; the SQNR difference is within  $\pm 1.1$  dB.

Both Fig. 3.7 and Fig. 3.8 present a bias shift in SQNR difference. This shift is produced because the noise model of multipliers in statistical analysis is an approximation of the actual noise distribution of multipliers. However, this is not an important issue as the shift is much smaller than the maximum SQNR difference. On the other hand, a parameter  $\Delta$  is introduced to indicate the maximum SQNR difference for the optimization process. The amount of SQNR difference is not analytically expressed, and is obtained by an experiment. Thus, according to experimental results, the value of  $\Delta$  is suggested 1 dB for the R2SDF architecture and 1.1 dB for the R2<sup>2</sup>SDF architecture when statistical analysis is mixed with simulation-based analysis.

### 3.3 Wordlength Optimization

Fig. 3.9 presents the flow of the proposed automatic wordlength optimization in the pipelined FFT processor. There are four major steps in the process. First, the upper bound wordlength (UBW) for each PE stage is evaluated based on the operating frequency requirement and the SQNR constraint (iSQNR) of the processor. Next, the UBWs of stages are fed into the lower bound wordlength (LBW) evaluation as an additional constraint for

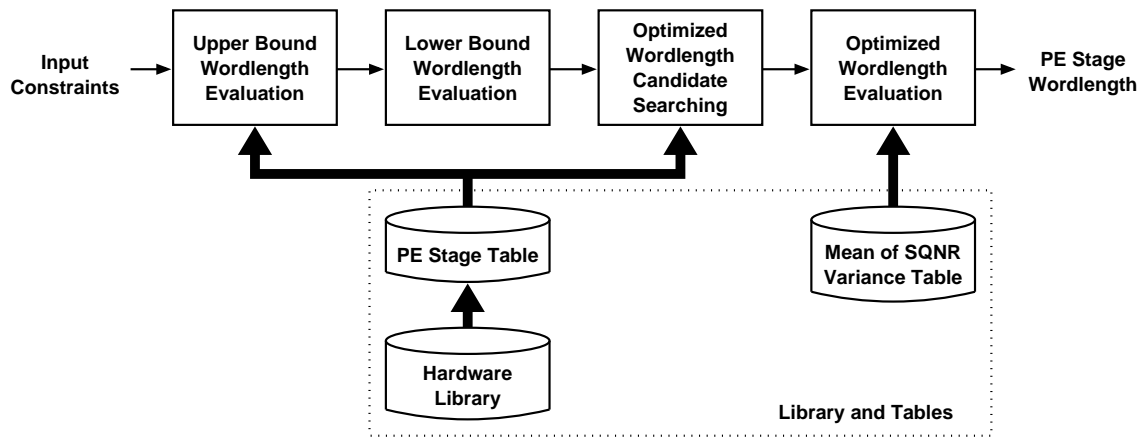


Figure 3.9: Wordlength optimization flow of a PE stage.

determining the LBWs. Both UBW and LBW evaluations employ the statistical analysis. Then, use the statistical analysis to determine optimized wordlength candidates (OWCs) based on  $iSQNR-\Delta$ . Finally, the optimized wordlength (OW) evaluation is performed based on the two primary procedures: (a) If the  $SQNR\_Error$  is  $\leq 1$  dB, a simulation analysis is used to select a solution with the smallest area. As the candidates are arranged in ascending order in area, the algorithm terminates after finding the first solution. (b) If the  $SQNR\_Error$  is  $> 1$  dB, a benefit function is introduced and the best benefit function is selected.

A hardware library, and two tables, a PE stage table and a mean of SQNR variance table, are prepared prior to activating the optimizing process. To optimize the hardware cost, one hardware library such as the TSMC  $0.25\mu\text{m}$  cell library, is chosen to determine the area size and critical timing delay of a PE stage in the FFT processor by synthesizing versus different wordlengths. The obtained data are recorded in the PE stage table to



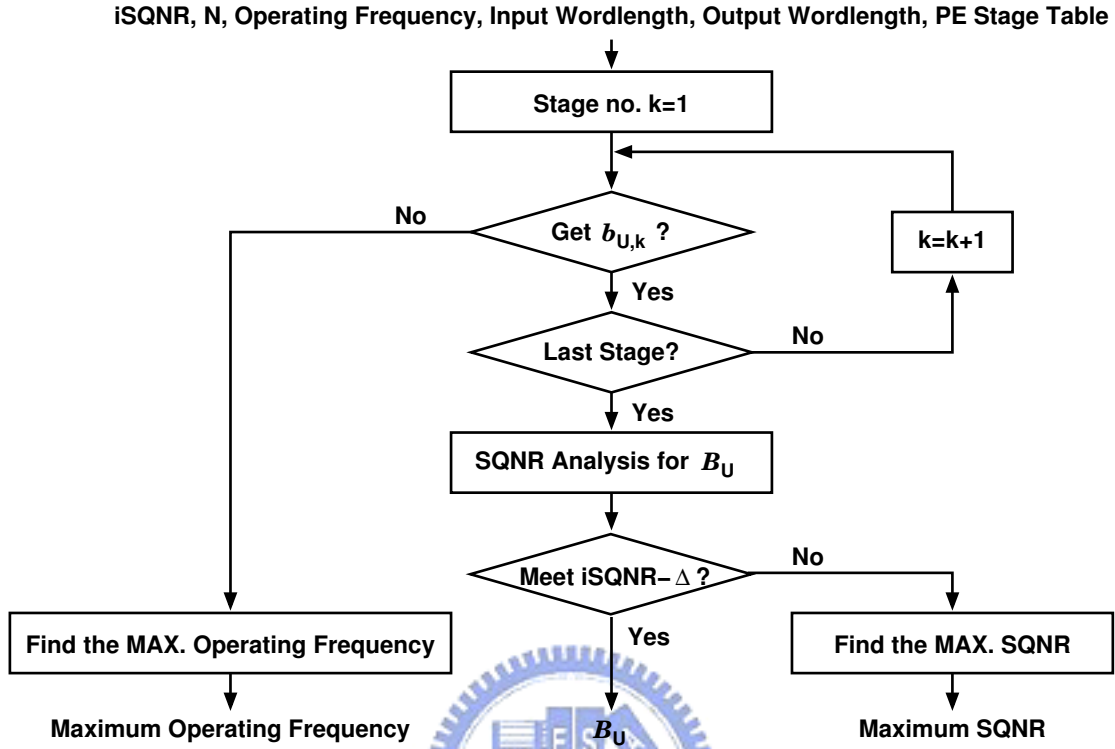


Figure 3.10: Evaluation of the upper bound wordlength.

speed up automation. The mean of the SQNR variance table is used to derive lengths of simulation at different simulated confidences according to (3.13). This table is established by calculating the mean of 100 simulated SQNR variances of a PE stage with wordlengths of 8-32 bits versus distinct FFT lengths ( $N$ ).

### 3.3.1 Evaluation of Upper Bound Wordlength

The UBW of the  $k$ -th PE stage, named as  $b_{U,k}$ , is defined as the maximum possible wordlength, such that the critical path satisfies the timing constraint, which is the inverse of the operating frequency of the FFT processor. Since the upper bound is obtained based

on the operating frequency and throughput, the lower throughput constraint and faster hardware library exactly increase the UBW. Conversely, the increased wordlength will require increased time for the operation of the PE stage. That is, increasing wordlength reduces the allowed operating frequency, and thus, the operating frequency requirement can be violated. Additionally, short wordlength results in a poor SQNR.

Fig. 3.10 shows the process of UBW evaluation. First, the UBW corresponding to the operating frequency requirement is evaluated stage by stage. When the operating frequency requirement is achieved for each stage,  $\{b_{U,k}\}$ s are obtained. Otherwise, the maximum allowable operating frequency is reported. When all  $\{b_{U,k}\}$ s are given, they are used to analyze the SQNR of the FFT processor. If the evaluated SQNR meets the SQNR constraint, the UBW set denoted by  $B_U$  comprising these  $\{b_{U,k}\}$ s is output, or the maximum achievable SQNR is reported.

### 3.3.2 Evaluation of Lower Bound Wordlength

When the UBW set is obtained, it is used to support the evaluation of the lower bound wordlength (LBW). The LBW set,  $B_L$ , is derived from  $B_U$ , such that the optimized solution must be above  $B_L$ . That is, if the solution is not above  $B_L$ , the solution will not meet the SQNR constraint.

Fig. 3.11 shows the flow of LBW evaluation. First, the UBW set,  $B_U$ , is assigned as the initial set  $B'_U$ . The evaluation is then conducted stage by stage. For the  $k$ -th stage, the

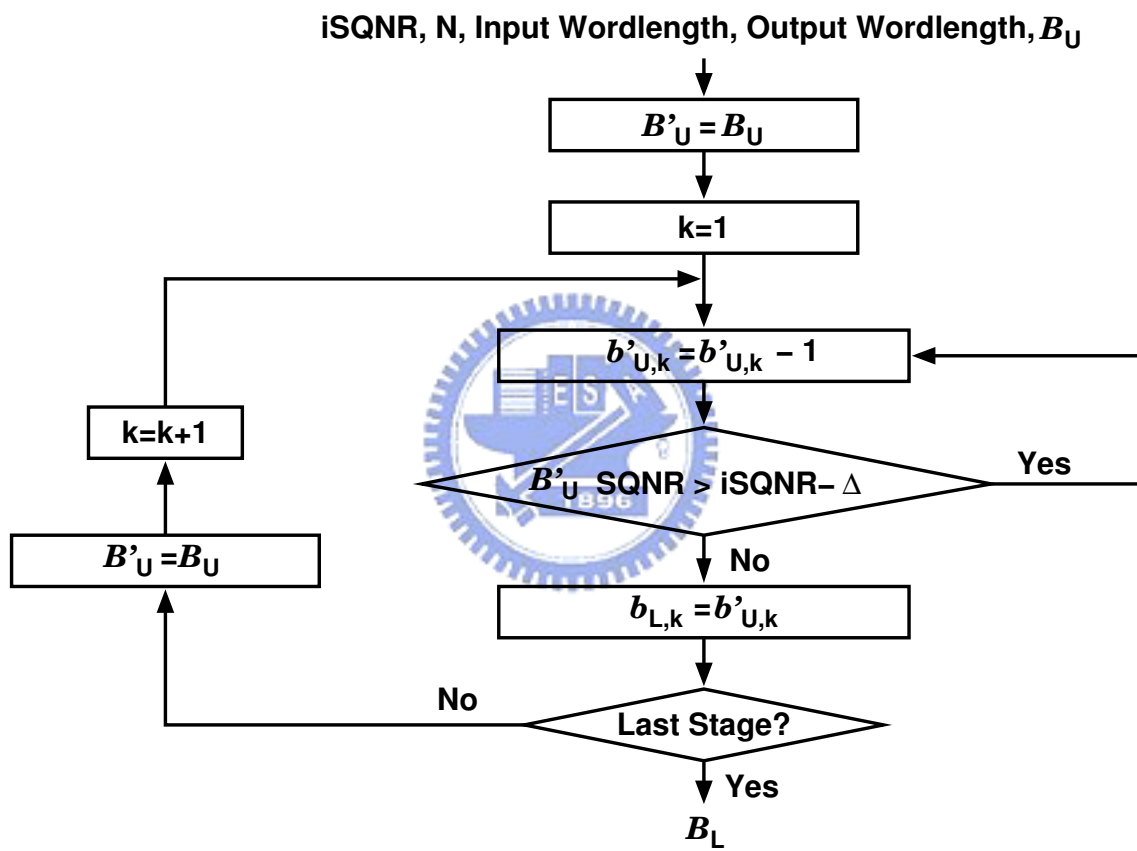


Figure 3.11: Evaluation of the lower bound wordlength.

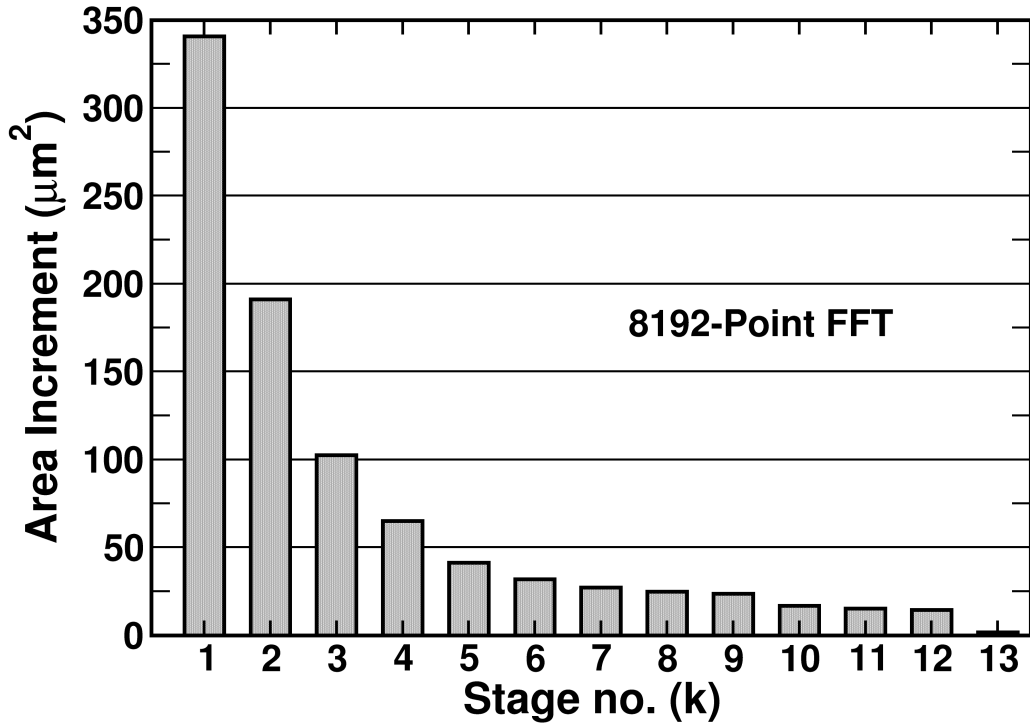


Figure 3.12: Area increment of each PE stage as the wordlength increases 1 bit

element of  $B'_U$ ,  $b'_{U,k}$ , progressively decreases until the SQNR of its corresponding set is smaller than the SQNR constraint; this  $b'_{U,k}$  is now stored and assigned as  $b_{L,k}$ . Notably, at the beginning of the evaluation of each stage,  $B'_U$  is reset as  $B_U$ . When the evaluation of the last stage is complete, the final LBW set,  $B_L$ , consisting of  $\{b_{L,k}\}$ s, is reported, and this LBW evaluation is terminated.

### 3.3.3 Optimized Wordlength Search

There are two chief considerations when searching the candidates of optimized wordlength.

First, since the DIF FFT processor requires large amount of memory, especially in the early stages, it is better to keep the wordlength of these stages short for area optimization.

This is demonstrated in Fig. 3.12, which shows that the area increment of each PE stage in an 8192-point FFT of Radix-2 when wordlength with 20 bits of each stage is added 1 bit.

The second issue is the output SQNR. The output SQNR of a pipelined FFT processor can be obtained by replacing these variances in (3.11) with corresponding values obtained from (3.1) to (3.10) and rearranged. The final expression is obtained as

$$SQNR \approx 10 \log_{10} \frac{c}{(a_1 2^{-2b_1+1} + a_2 2^{-2b_2+2} + \dots + a_P 2^{-2b_P+P})} \quad (3.14)$$

where  $c$  is a constant,  $a_k$  is a constant for each PE stage, and  $k \in \{1, 2, \dots, P\}$ . If

there exists the  $m$ -th PE stage such that  $a_m 2^{-2b_m+m} \gg a_k 2^{-2b_k+k}$ , for all  $k$  and  $k \neq$

$m$ , this stage will be a bottleneck degrading the output SQNR. That is, the value of

$(a_1 2^{-2b_1+1} + a_2 2^{-2b_2+2} + \dots + a_P 2^{-2b_P+P})$  is dominated by the term  $a_m 2^{-2b_m+m}$ . Thus, it

is efficient to choose the sizes of the wordlength to be close, so that each stage has an approximate quantity of errors.

In addition to the consideration already mentioned, there are two major properties of the R2SDF and R2<sup>2</sup>SDF architectures that must be considered when determining the

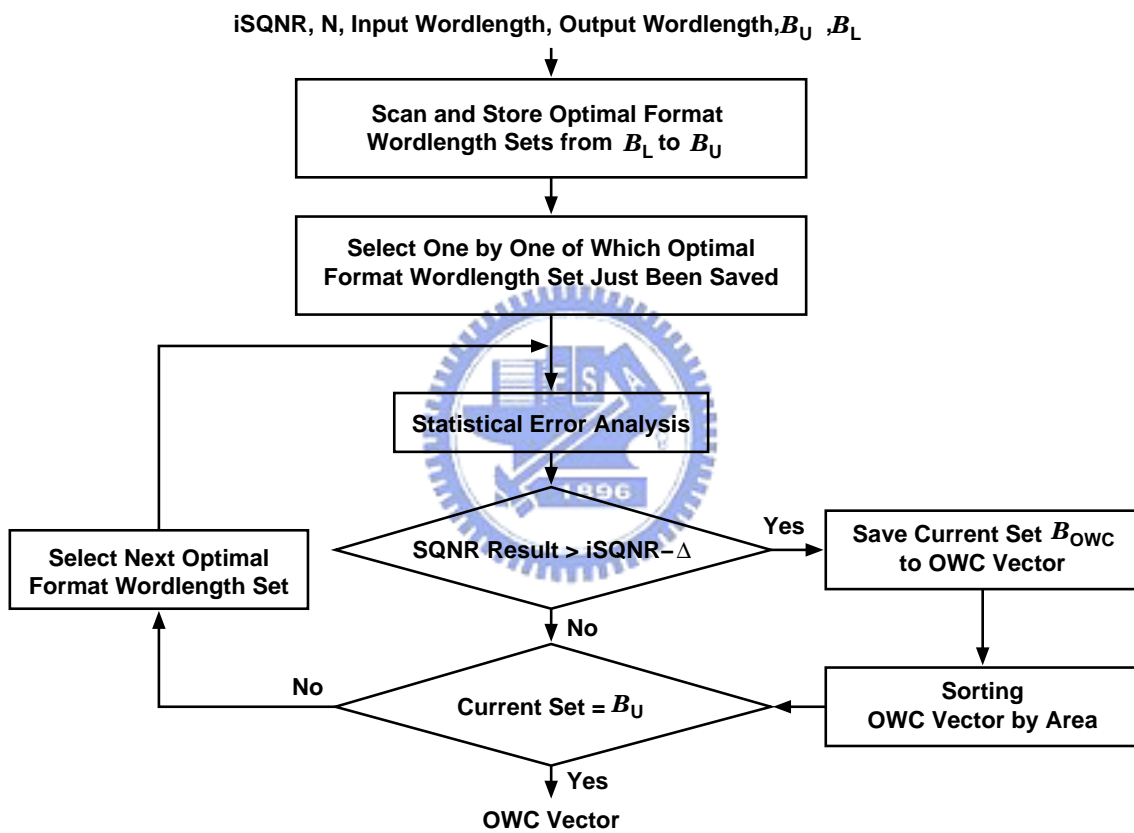


Figure 3.13: The procedure to determine the optimized wordlength set candidates.

format of the optimized wordlength. The required FIFO memories decrease stage by stage (Fig. 3.1), and the errors degrading SQNR increase due to the amount of  $a_k \cdot 2^k$  in each product term of (3.14) increasing along as  $k$  increases. According to these properties and considerations, the optimization format is in ascending order. This optimization format is true only for the architectures with the characteristics that the required memories decrease stage by stage. For other implementations, the optimization format needs to be modified, and then the flow for optimization described in the following can be employed.

An optimized wordlength set candidate (OWC),  $B_{OWC}$ , of a pipelined FFT processor has three properties: (a)  $B_L \leq B_{OWC} \leq B_U$ ; (b)  $B_{OWC}$  is in ascending order; and (c) the SQNR of the FFT processor with this wordlength set meets the SQNR constraint. To find the OWC vector, all wordlength sets from the LBW set,  $B_L$ , to the UBW set,  $B_U$ , are scanned, and those sets that are in ascending order survive. These sets are then evaluated using the statistical analysis. If the evaluated SQNR meets the constraint, its corresponding set is chosen as an OWC and is stored in the OWC vector sorted by area. This process is iteratively performed until the wordlength set reaches  $B_U$  and its evaluation is completed. Fig. 3.13 shows this search flow. The final output of the flow is the OWC vector consisting of  $B_{OWC}$  sets sorted by area size.

Fig. 3.14 presents the flow to determine the optimized wordlength (OW) set,  $B_{OW}$ . First, choosing which method to employ is according to the constraint of SQNR error. When simulation-based analysis is used, the  $B_{OWC}$ 's in the OWC vector are simulated

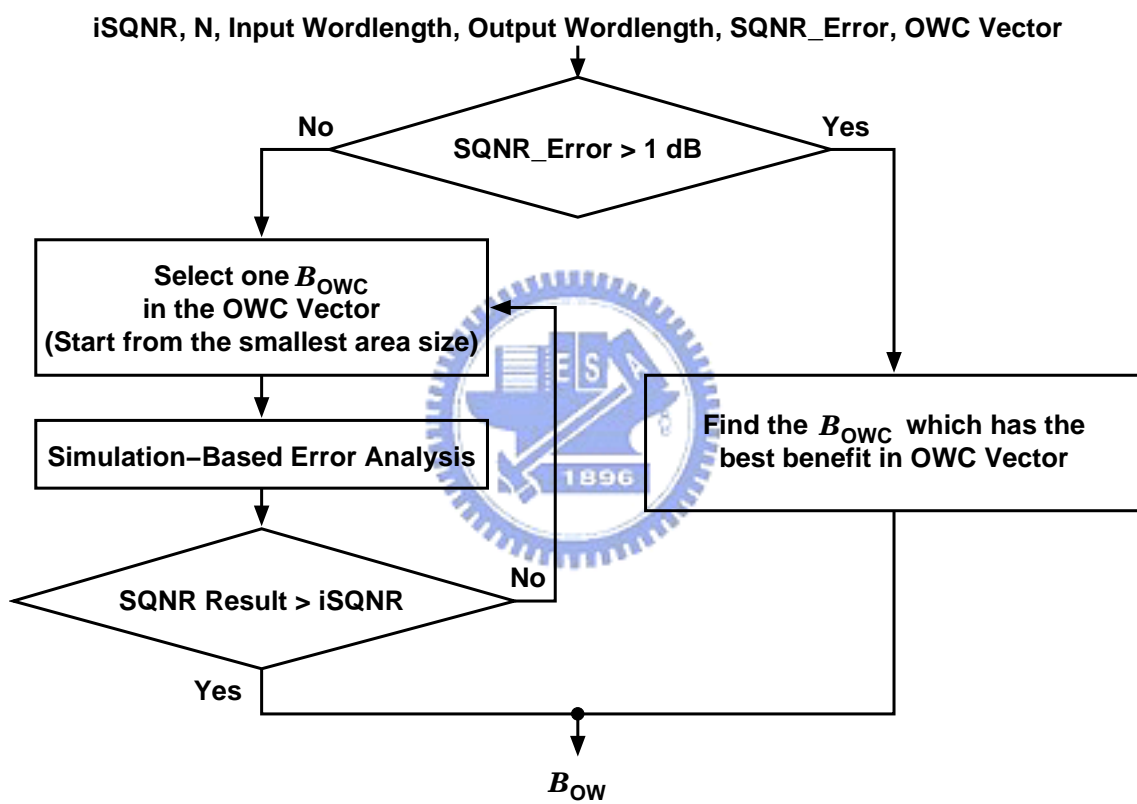


Figure 3.14: Optimized wordlength selection.



starting with the  $B_{OWC}$  with the smallest area until the SQNR of the simulated  $B_{OWC}$  meets the SQNR constraint,  $iSQNR$ . The entire process of optimization is named a hybrid method, whereas the analysis employed is the statistical method before finding the  $B_{OW}$ .

When the SQNR\_Error is  $> 1$  dB, the whole optimized process using the statistical technique is called the pure statistical method. In this method,  $B_{OW}$  is determined based on the benefit function that is defined as

$$Benefit = \frac{\Delta_{SQNR}}{\Delta_{Area}} \quad (3.15)$$

where  $\Delta_{SQNR}$  is the difference in SQNR between the  $B_L$  and the analyzed  $B_{OWC}$ , and  $\Delta_{Area}$  is their area difference. The  $B_{OWC}$  with the best benefit is the optimized wordlength,  $B_{OW}$ .



As the statistical analyzer is less accurate due to the error in the analytical model,  $Benefit$  is therefore introduced to fine tune the solution to get an enhanced SQNR with little area overhead. In the trade-off between SQNR and area,  $Benefit$  is a better cost function than the area cost function.

Fig. 3.15 shows an example of the steps in the wordlength optimization by using the hybrid method when the SQNR error constraint is under 1 dB. The system constraints are the same as those in Table 3.3. The  $sim\_SQNR$  means the result of simulation, and  $iSQNR$  is the SQNR constraint. After sorting the OWC vector with ascending area size, the  $B_{OW}$  selection is performed one-by-one until the  $sim\_SQNR$  is  $> iSQNR$  and then the

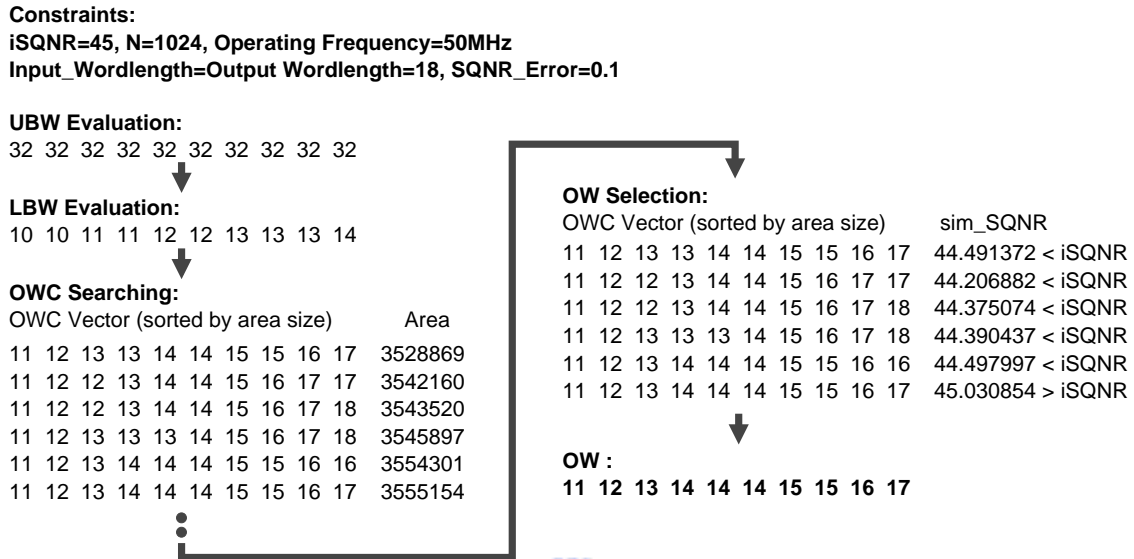


Figure 3.15: An example of hybrid wordlength optimization.

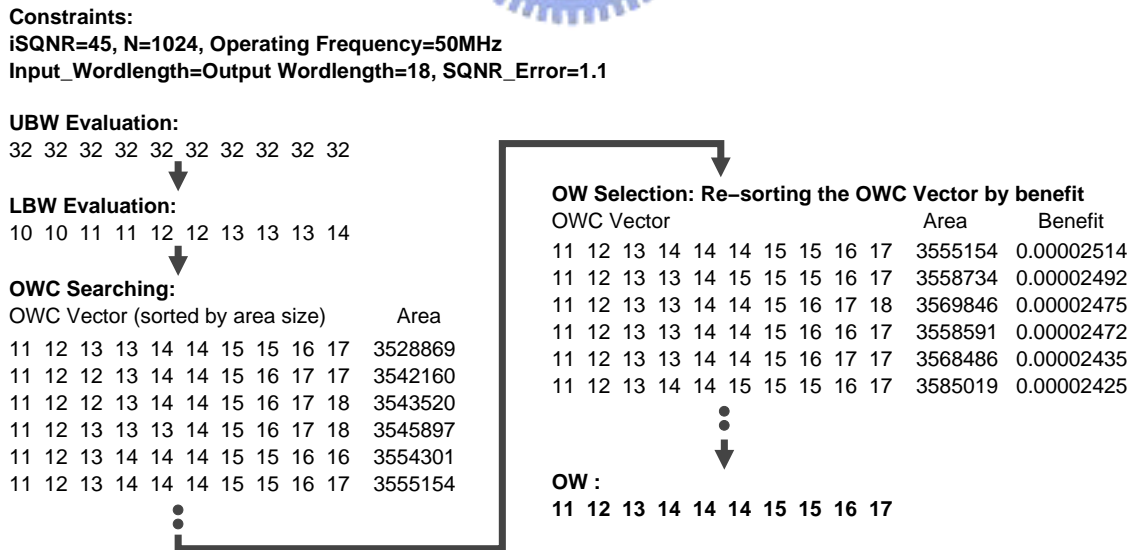
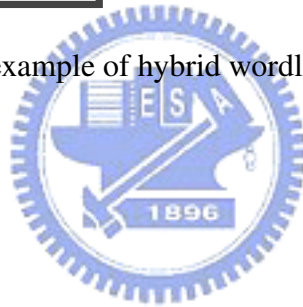


Figure 3.16: An example of pure statistical analysis.

Table 3.2: Common Specifications of FFT for OFDM

	FFT Length	Operating Frequency	I/O Format
Short Length	16~256 points	50 MHz	Complex, Word-Sequential
Long Length	256~8192 points	20 MHz	Complex, Word-Sequential

outcome of  $B_{OW}$  is obtained.

Fig. 3.16 shows the example of optimization employing the pure statistical method. Table 3.3 the system constraints, except for SQNR\_Error = 1.1 dB. Because the constraint of SQNR error is  $> 1$  dB, the pure statistical method is employed to identify  $B_{OW}$ . The  $B_{OWC}$ 's in the OWC vector are re-sorted based on the benefit that is calculated according to (3.15). The  $B_{OWC}$  with the best benefit is  $B_{OW}$ .

To determine the OWC vector, it requires full scan of the sets with the ascending order, and it therefore has the complexity of  $O(2^{P+\max\{b_{U,k}\}})$ . To perform the optimized wordlength evaluation, a greedy algorithm is used if the SQNR\_Error is  $\leq 1$  dB. If the SQNR\_Error is  $> 1$  dB, the evaluation requires to calculate the benefit function of the OWC vector. Hence, it has the same complexity as the determination of the OWC vector.

### 3.4 Experimental Results

In this work, FFT architectures DIF R2SDF and DIF R2<sup>2</sup>SDF were implemented by using C++ language with SystemC [38] [39] library to demonstrate the proposed flow for op-

timization. The FFT length,  $N$ , can be adjusted from 8-8192, and the wordlength of each stage can be altered from 8-32 bits. Since the SQNR range is 40-60 dB and the specifications (Table 3.2) are typically used in most OFDM-based systems [40], the SQNR of 45 dB and operating frequency of 50 MHz were adopted as constraints in these experiments.

Synthesis was conducted without any constraints using Synopsys Design Analyzer [41]. The models of the hardware functional blocks including adders, multipliers, and multiplexers, employ Synopsys DesignWare [42] and TSMC 0.25 $\mu$ m cell library, as well as the memory models, including the shift registers and ROMs [43] [44]. The fast carry look-ahead synthesis model for adders, booth-encoded Wallace tree synthesis model for multipliers, and the universal multiplexer synthesis model for multiplexers are adopted. The area and timing reports of Synopsys Design Analyzer are used for these models.

To verify the proposed optimization flow of the FFT processor, the C++ language with SystemC library is used to build the fixed-point model of the FFT hardware. In this experiment, the quantization mode is always truncation, which is SC\_TRN in the SystemC library, and the overflow mode is saturation, which is SC\_SAT in the library.

Finally, the platform is built in a PC with an Intel 2.4GHz CPU and 768MB RAM memory. The operating system (OS) is Microsoft Windows 2000.

Table 3.3: Constraints for Optimization

SQNR	SQNR_Error	Operating Frequency	I/O Wordlength	SQNR Simulation Confidence Level
45 dB	0.1 dB	50 MHz	18 bits	95%

### 3.4.1 Variant FFT Lengths

Table 3.4 and Table 3.5 present experimental results for area optimization versus FFT lengths of 8-8192 points. Table 3.4 shows the experimental results for R2DSF and Table 3.5 shows the experimental results for R2<sup>2</sup>SDF. Table 3.3 presents the optimization constraints, in which the SQNR\_Error is the value of  $e$  in (3.13) and the SQNR simulation confidence level is the amount of  $(1 - \alpha)100\%$ . Since the constraint of SQNR error is smaller than 1 dB, the hybrid method is employed. In both tables, the first column lists the FFT length. The second column indicates whether the choice of wordlength is optimized; w/o indicates that wordlength choice is based on using the same wordlength for each stage, which is the minimum length such that all the constraints are met, and w/ is the wordlength is decided using the proposed optimization method. The column “Area Reduction” presents the reduction rate of area calculated by

$$\frac{Area_{\{w/o\ optimization\}} - Area_{\{w/\ optimization\}}}{Area_{\{w/o\ optimization\}}} \times 100\% \quad (3.16)$$

The last column, “Time”, is the required time of the used computer for performing optimization. The maximum and minimum area reduction rates are 24% and 9% for R2SDF, and 23% and 6% for R2<sup>2</sup>SDF. In summary, these tables show that as  $N$  increased, the area

Table 3.4: Area Reduction of R2SDF with Different FFT Lengths Using Hybrid Method

		I/O Wordlength = 18 bits													Area ( $\mu\text{m}^2$ )	Area Reduction	Time (sec)
FFT Length (N)	w/ or w/o Optimization	Wordlength of PE Stage															
		1	2	3	4	5	6	7	8	9	10	11	12	13			
8	w/o	12	12	12											258831	16%	8
	w/	11	11	12											216184		
16	w/o	12	12	12	12										402693	11%	11
	w/	11	11	12	13										359378		
32	w/o	13	13	13	13	13									733434	9%	18
	w/	11	12	12	13	13									664339		
64	w/o	14	14	14	14	14	14								1152506	14%	20
	w/	11	12	13	13	13	14								993360		
128	w/o	14	14	14	14	14	14	14							1550048	10%	41
	w/	11	12	13	13	14	14	15							1388215		
256	w/o	15	15	15	15	15	15	15	15						2249617	16%	46
	w/	11	12	13	13	14	14	15	15						1882859		
512	w/o	15	15	15	15	15	15	15	15	15					2988883	14%	67
	w/	11	12	13	13	14	15	15	15	16					2569228		
1024	w/o	16	16	16	16	16	16	16	16	16	16				4474445	20%	139
	w/	11	12	13	13	14	14	15	16	17	17				3568487		
2048	w/o	16	16	16	16	16	16	16	16	16	16	16			6396581	19%	310
	w/	11	12	13	14	14	15	15	15	16	17	18			5184678		
4096	w/o	17	17	17	17	17	17	17	17	17	17	17	17		10255423	23%	616
	w/	11	12	13	13	14	15	15	16	17	17	17	18		7858489		
8192	w/o	17	17	17	17	17	17	17	17	17	17	17	17	17	16668224	24%	971
	w/	11	12	13	13	14	15	15	16	17	17	18	18	19	12676846		

Table 3.5: Area Reduction of R2<sup>2</sup>SDF with Different FFT Lengths Using Hybrid Method

		I/O Wordlength = 18 bits												Area ( $\mu m^2$ )	Area Reduction	Time (sec)
FFT Length (N)	w/ or w/o Optimization	Wordlength of PE Stage														
		1	2	3	4	5	6	7	8	9	10	11	12			
16	w/o	12	12	12	12									214156	11%	10
	w/	11	11	12	13									189706		
64	w/o	13	13	13	13	13	13							760033	6%	20
	w/	11	12	12	13	13	14							717914		
256	w/o	15	15	15	15	15	15	15	15					1732185	16%	31
	w/	11	12	12	13	13	14	15	16					1461204		
1024	w/o	16	16	16	16	16	16	16	16	16	16			3695209	20%	65
	w/	11	12	13	13	14	14	15	15	16	17			2973710		
4096	w/o	17	17	17	17	17	17	17	17	17	17	17	17	9271212	23%	317
	w/	11	12	12	13	14	14	15	16	17	18	18	18	7120428		

reduction rate increases.



### 3.4.2 Output SQNR Requirement

Fig. 3.17 shows the achieved area reduction rate versus different SQNR constraints for R2SDF and R2<sup>2</sup>SDF. In the conventional designs with the same wordlength for all pipeline stages, the SQNR increases about 6 dB when wordlength increase by 1 bit. In Fig. 3.17, a similar phenomenon can be observed when 6 dB is a cycle area reduction rate for different SQNR requirements. The reduction rate varied from 12%-20%.

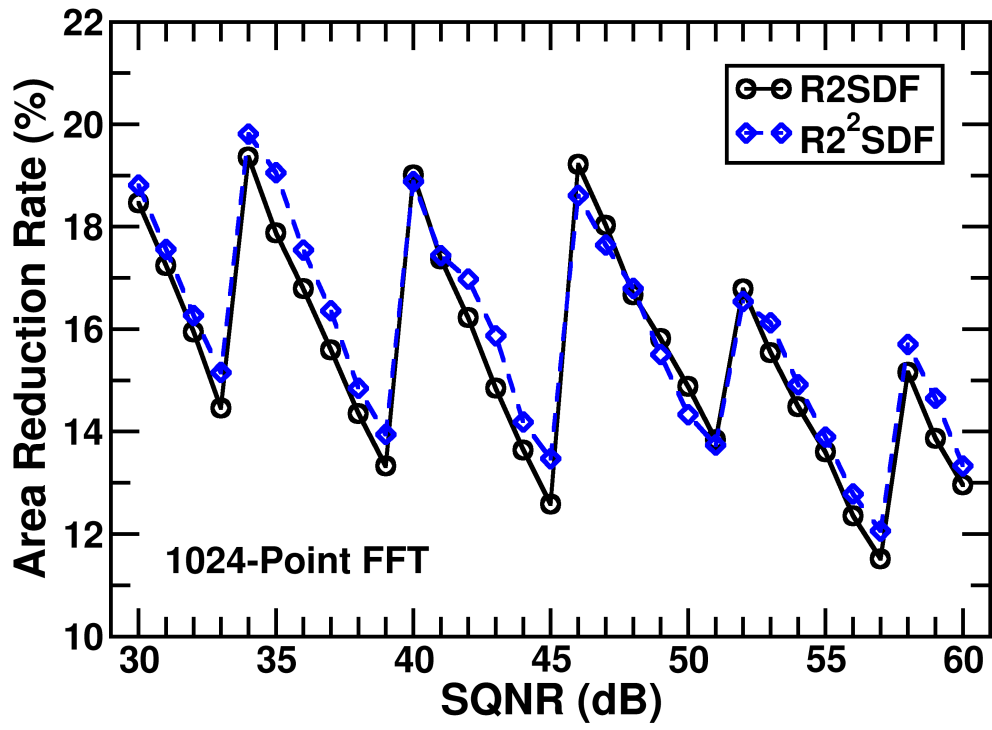


Figure 3.17: Area reduction rate versus SQNR constraints.



Table 3.6: Area Reduction of R2SDF with Different FFT Lengths Using Statistical Analysis

FFT Length (N)	w/ or w/o Optimization	I/O Wordlength = 18 bits											Area ( $\mu\text{m}^2$ )	Area Reduction	Time (sec)	Post SQNR (dB)		
		Wordlength of PE Stage																
		1	2	3	4	5	6	7	8	9	10	11	12	13				
8	w/o	12	12	12											258831	16%	1	46.43
	w/	11	11	12											216184			
16	w/o	12	12	12	12										402693	11%	1	45.44
	w/	11	11	12	13										359378			
32	w/o	13	13	13	13	13									733434	9%	1	45.27
	w/	11	12	12	13	13									664339			
64	w/o	14	14	14	14	14	14								1152506	14%	1	45.58
	w/	11	12	13	13	13	14								993360			
128	w/o	14	14	14	14	14	14	14							1550048	12%	1	44.94
	w/	11	12	12	13	14	14	15							1370066			
256	w/o	15	15	15	15	15	15	15	15						2249617	16%	1	45.64
	w/	11	12	13	13	14	14	15	16						1884288			
512	w/o	15	15	15	15	15	15	15	15	15					2988883	15%	1	44.62
	w/	11	12	13	13	14	14	15	15	16					2546349			
1024	w/o	16	16	16	16	16	16	16	16	16	16				4474445	21%	1	45.00
	w/	11	12	13	14	14	14	15	15	16	17				3555154			
2048	w/o	16	16	16	16	16	16	16	16	16	16	16			6396581	19%	1	44.84
	w/	11	12	13	13	14	15	15	15	16	17	18			5153721			
4096	w/o	17	17	17	17	17	17	17	17	17	17	17	17		10255423	23%	2	45.30
	w/	11	12	13	13	14	15	15	16	17	17	18	19		7875389			
8192	w/o	17	17	17	17	17	17	17	17	17	17	17	17	17	16668224	24%	2	45.04
	w/	11	12	13	13	14	15	15	16	17	17	18	18	19	12676846			

### 3.4.3 Loose SQNR Error Requirement

Table 3.6 shows the wordlength optimization results based on the constraints presented in Table 3.3, except SQNR error=1.1 dB. Since the constraint of SQNR error equals 1.1 dB, the statistical analysis method is employed for optimization. The achieved area reduction rate versus the FFT lengths drifts toward the same direction as the results shown in Table 3.4. The SQNR corresponding to the optimized wordlengths is evaluated by simulation to verify its accuracy. Table 3.6 lists these evaluated results in the last column. The obtained SQNRs of FFT lengths, 128-point, 512-point, and 2048-point violate the SQNR constraint.



### 3.4.4 Comparisons of Three Methods

Fig. 3.18 shows the comparisons of the experimental results by employing the pure statistical method, pure simulation-based method, and the hybrid method for the R2SDF optimization. The pure simulation-based method is the proposed optimization without statistical error analysis and applies greedy algorithm to find the local optimum. One db is added to the SQNR constraint (iSQNR) to allow for the SQNR error in the statistical model. This is done for overdesign to cover the SQNR error of the statistical model. These experimental results suggest that overdesign with extra 1 dB has larger hardware costs than other two methods. This difference is reflected by comparisons of the area re-

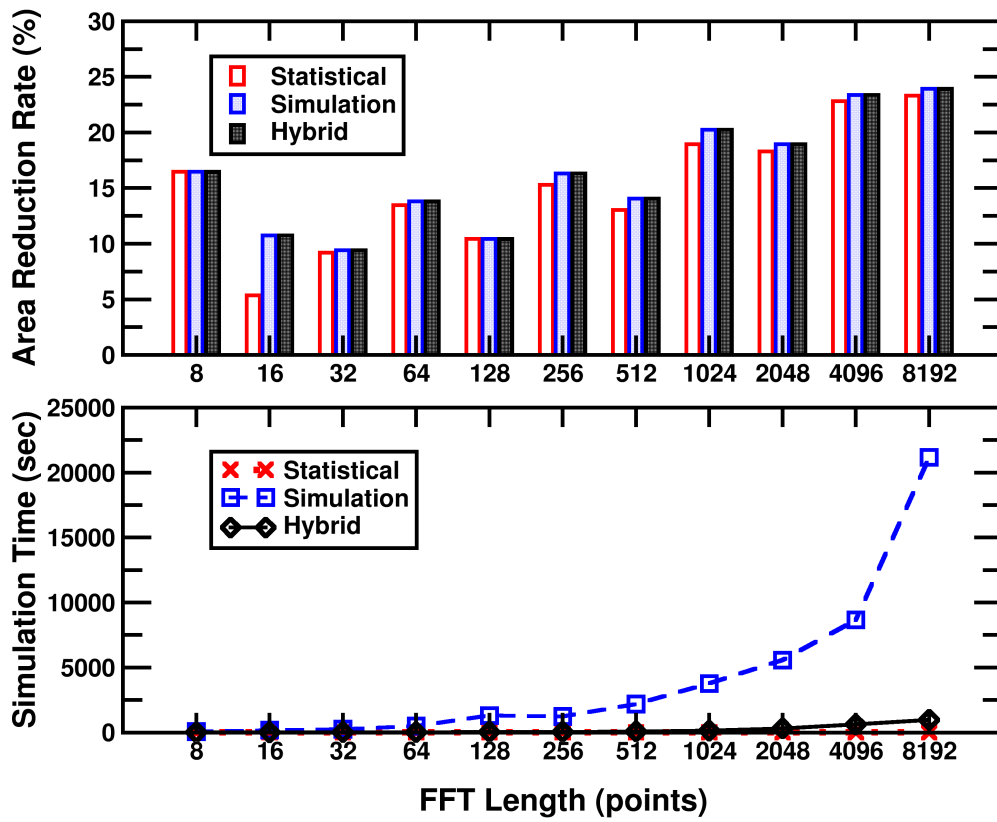


Figure 3.18: Comparisons of results using different analytical methods.


duction rate versus FFT lengths shown in the top frame. The bottom frame presents that the required time for performing optimization can be dramatically reduced by using the hybrid methods compared with that when using the pure simulation-based analysis with approximately the same optimized area. In summary, employing the hybrid method for the wordlength optimization is faster than the pure simulation-based method and provides a more accurate determination of wordlengths than the pure statistical method.

### 3.5 Summary

This work presented a modified statistical error model for varying wordlengths of individual stages of a pipelined FFT processor with Radix-2 and Radix-2<sup>2</sup> algorithms. The error model is used to investigate the hybrid method for wordlength optimization at the stage level. The proposed hybrid method combined with statistical and simulation-based error analyses performs the wordlength optimization based on minimizing the area size cost. A generator of pipelined FFT processors was developed for experiments to validate the proposed method and error model. The generator gives the suggested value of the maximum available SQNR or operating frequency at the UBW evaluation when constraints cannot be met. Experiments indicate that the hybrid method can obtain optimized results faster than the conventional simulation-based method, thereby reducing design time for the FFT processor. Furthermore, the area size of the processor is also minimized.

# Chapter 4

## Multiplier Generator



Multiplication is a basic arithmetic operation for digital signal processing. The speed of a multiplier is a critical issue in determining the performance of digital signal processors (DSPs). Therefore, high speed multipliers are required for high performance DSPs. In this work, a high performance multiplier generator was developed. The generator try to optimize the gate delay and the wire delay to get optimized solution in deep-submicron design.

### 4.1 Overview of Multiplier

Multiplication is a three-step process, and its conceptual sketch is shown in Fig. 4.1, where CPA is the abbreviation of *carry propagate adder* [45]. The first step is generation

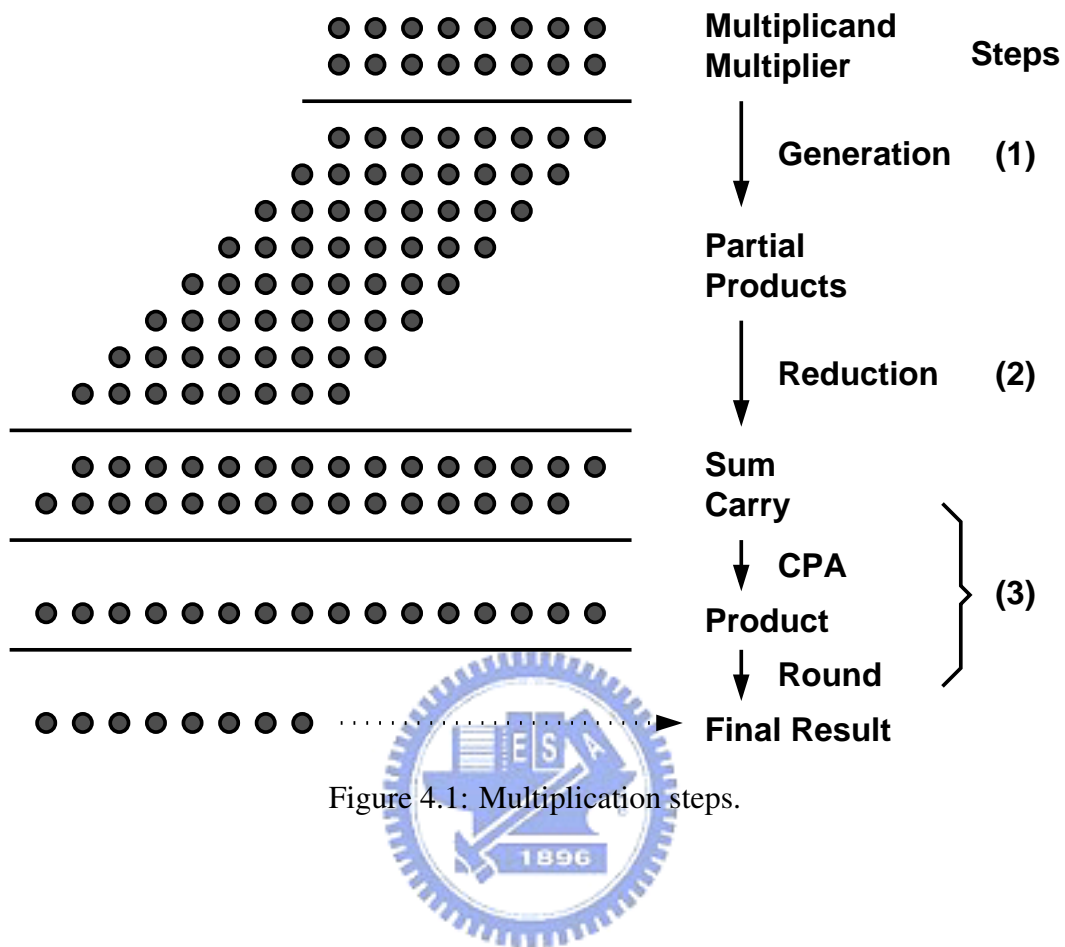


Figure 4.1: Multiplication steps.

of partial products. Then, the step advances on reduction of partial products to produce "Sum" and "Carry" rows. Finally, the result of multiplication is obtained by adding "Sum" and "Carry" rows using a CPA. This result is then rounded into a valid representation according to the system's specification. In this work, the generation of partial products uses non-Booth encoding, and the reduction of partial products employs Wallace-tree method [46].

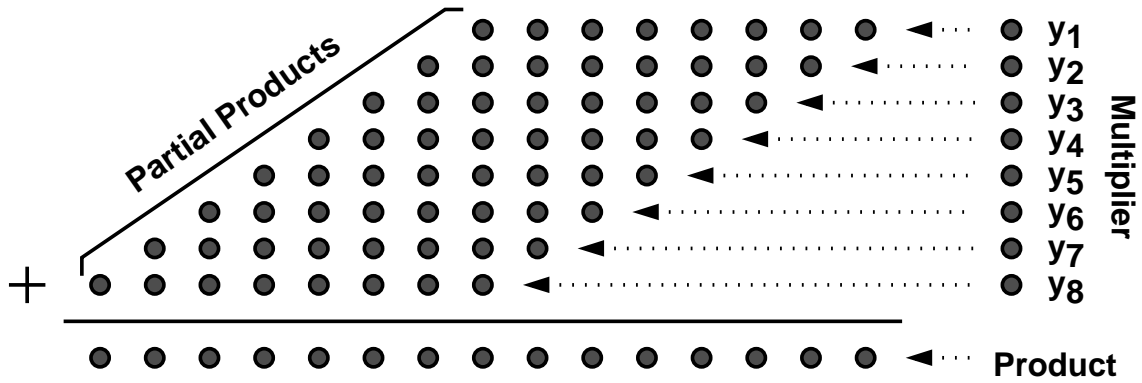


Figure 4.2: Non-Booth encoding for 8 × 8 multiplication.

Table 4.1: Selection of partial product

Multiplier Bit ( $y_i$ )	$M$ -bit Partial Product
0	0
1	X

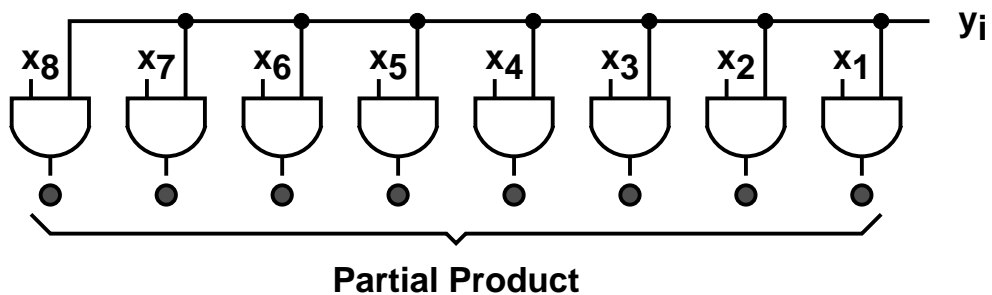


Figure 4.3: Partial product selection using a row of AND gates.

### 4.1.1 Non-Booth Encoding

The non-Booth encoding is the simplest method for producing partial products. Fig. 4.2 shows the illustration of non-Booth encoding for  $8 \times 8$  multiplication. Each row of dots represents a partial product. Each dot stands for a bit in the partial product. For ( $M$ -bit  $X$ )  $\times$  ( $N$ -bit  $Y$ ), this non-Booth algorithm selects the partial products from the set  $\{0, X\}$  according to each bit of  $Y$ . That is each partial product is either  $M$ -bit 0 or shifted  $X$ , as shown in Table 4.1. The non-Booth encoder is typically implemented using a logical AND gate for each bit of partial products as shown in Fig. 4.3.

### 4.1.2 Wallace Tree Method



The Wallace-tree method [46] is widely used for constructing high speed multipliers. Fig. 4.4 shows an illustration of Wallace tree with 3-to-2 compressors which are typically implemented using full adders (FA's) for reducing 18 partial products. This method reduces the partial products by connecting  $n$ -to- $m$  compressors in parallel in what is known as tree structure. Wallace tree-structure multipliers are higher speed than array-structure multipliers, since its delay time is proportional to the logarithm of the number of bits of a multiplier. However, Wallace tree is irregular, and hence, the complex interconnection makes its physical design crucial.

To build the reduction tree, the  $n$ -to- $m$  compressors are designed with the assistance



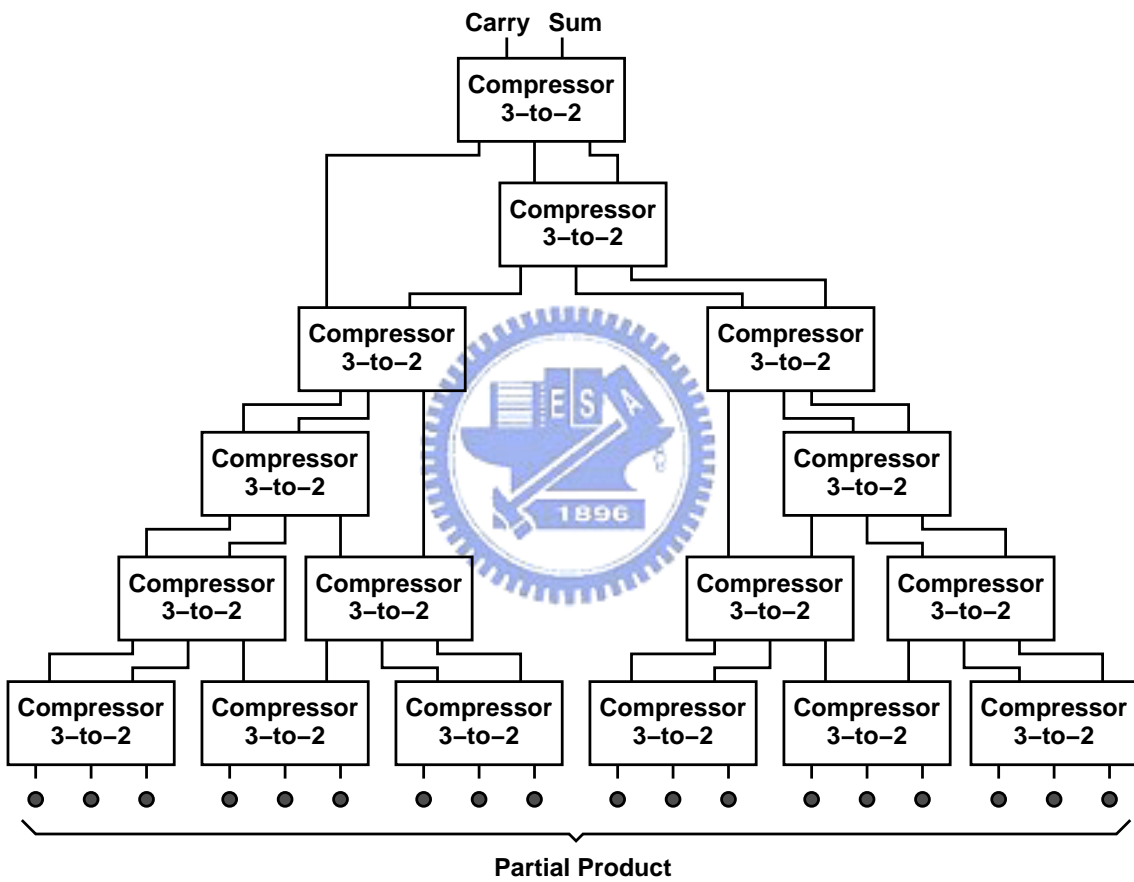


Figure 4.4: Illustration of Wallace tree for reducing 18 partial products.

```

Procedure: Multiplier_Generation
begin
    for (each VCS in increasing-weight order)
01    partial product generation;
02    placement for partial products;
03    VCS generation;
04    timing driven placement;
05    final adder adjustment;
end

```

Figure 4.5: Overview of multiplier generation.

of software that balances the different connecting paths [45] [47] [48]. This is a global approach to minimize the total delay for all possible paths, and is examined in the following sections.



## 4.2 Layout-Driven Multiplier Generation

To optimize multiplier design addressed in hardware size and speed, the generation flow is developed incorporating synthesis and placement. Fig. 4.5 shows an overview of the flow of multiplier generation. The flow is performed starting from the LSB vertical compressor slice (VCS), and ending at the MSB VCS. The first step is to generate partial products. Next, the logic of partial product terms are placed as compact as possible, and at the same time, the placement of half adders and the insertion of buffers are implemented. To compress the partial products, the VCS's are then produced based on balancing path

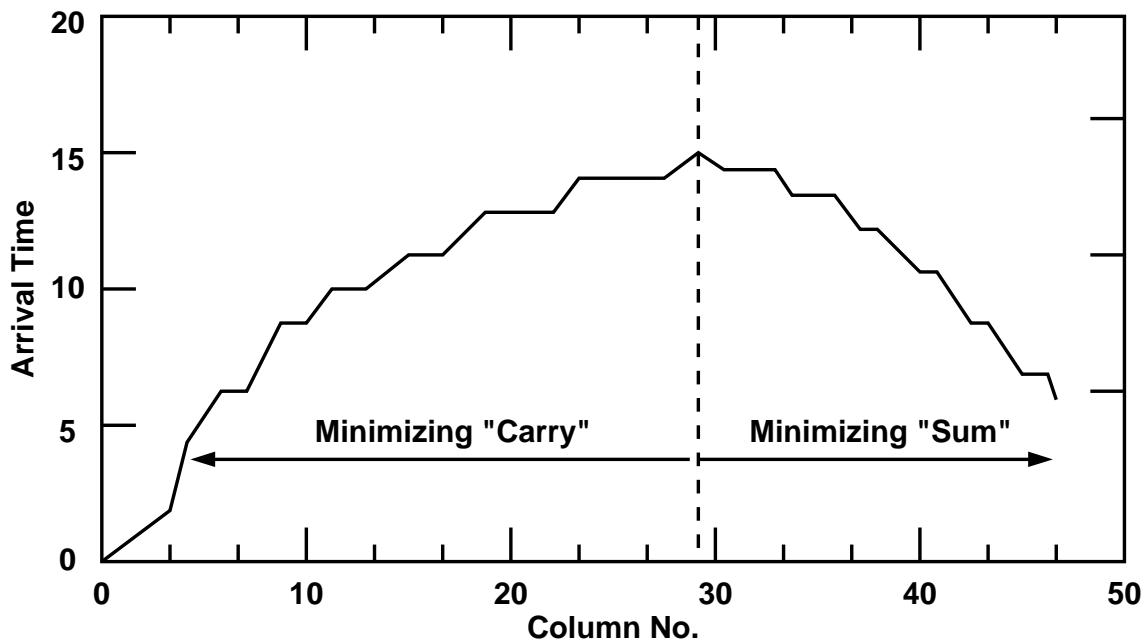
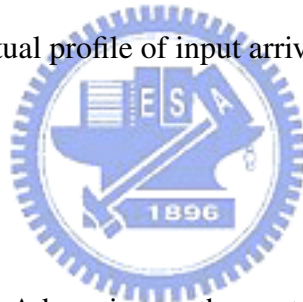


Figure 4.6: Conceptual profile of input arrival time of final adder.



delays according to gate delays. Advancing on the next step, the compressors of VCS are placed based on balancing path delays according to wire delays. Finally, the high-speed final adder is added, and the adder is typically implemented employing a carry lookahead adder (CLA).

After compression, the input arrival time of the final adder has a typical profile as shown in Fig. 4.6. In the compressing process, the optimization is performed to minimize the delays of "Carry" of those VCS's before that one with maximum delay, and minimize the delays of "Sum" and then minimize the delays of "Carry" of other VCS's.

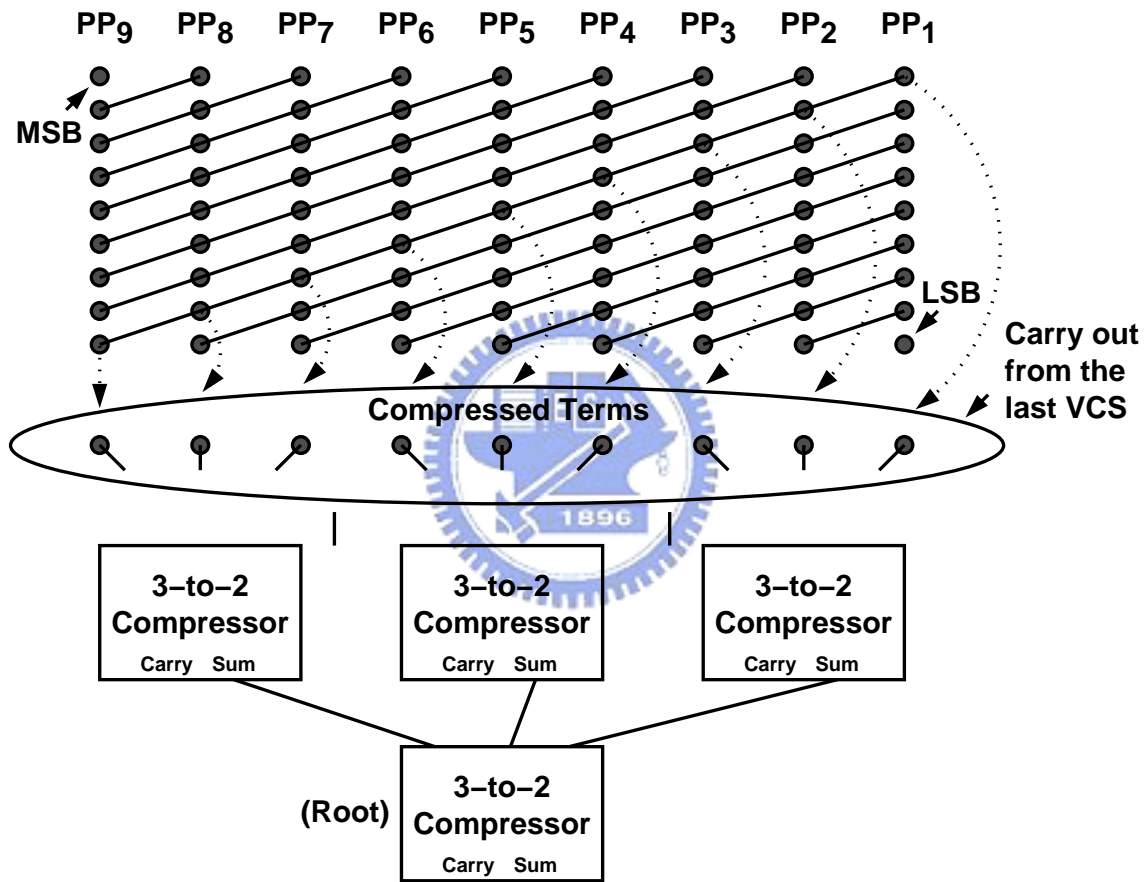


Figure 4.7: Overview of vertical compressor slice.

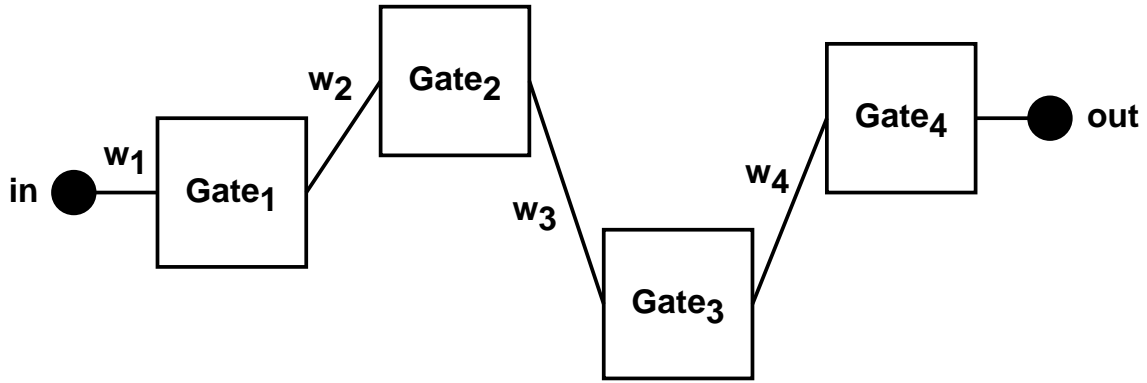


Figure 4.8: Evaluation of output arrival time

### 4.2.1 Delay Estimation

To build a VCS, the root is defined as the compressor that produces the final "Carry" and "Sum", and the leaf nodes are all compressed terms including the propagated carries from the previous VCS. The conceptual sketch using a  $9 \times 9$  multiplier with 3-to-2 compressors as an example is shown in Fig. 4.7, where  $PP_i$ , for all  $i$ , are the partial products.

Fig. 4.8 shows an example of the evaluation of output arrival time when signal propagates from node  $I$  to node  $O$ . As shown in this figure, the output data arrival time is given by

$$t_{out} = t_{in} + \sum_{i=1}^4 t_{wd,i} + \sum_{j=1}^4 t_{gd,j} \quad (4.1)$$

where  $t_{in}$  is the input data arrival time,  $t_{wd,i}$  is the wire delay of  $w_i$ , for all  $i$ , and  $t_{gd,j}$  is the gate delay of  $Gate_j$ , for all  $j$ . For the developed method, the output arrival time is modified by extending (4.1), and is given by

$$t_{out} = t_{in} + \sum_{i=1}^{K_w} t_{wd,i} + \sum_{j=1}^{K_g} t_{gd,j} \quad (4.2)$$

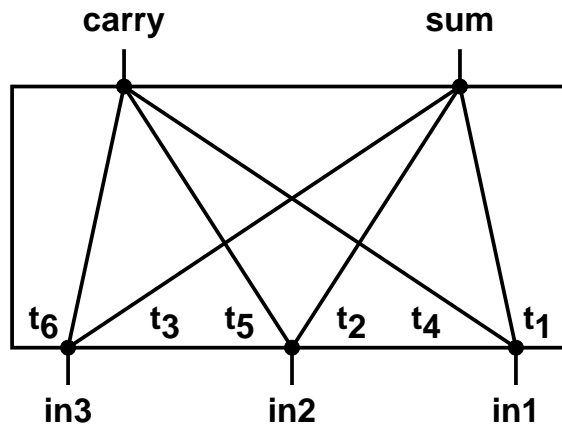


Figure 4.9: Cell-based delay model of a 3-to-2 compressor

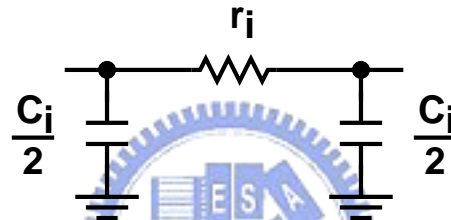


Figure 4.10:  $\pi$ -model of the  $i$ -th wire ( $w_i$ )

where  $K_w$  is the total number of wires, and  $K_g$  is the total number of gates. The VCS is generated based on balancing the output arrival time of VCS that is evaluated by roughly estimating the wire delay according to the number of compressors of a path.

In this work, the cell-based model rather than the conventional XOR-based model is used as the timing model, such that the timing analysis can be more accurate. For a 3-to-2 compressor, six path delays are modeled as shown in Fig. 4.9. These six paths are data from the inputs,  $in1$ ,  $in2$ , and  $in3$ , to the output,  $carry$  and  $sum$ . The arrival time of  $carry$  and  $sum$  is calculated by summing the corresponding wire delays, gate delays, and input arrival time as expressed in (4.2).

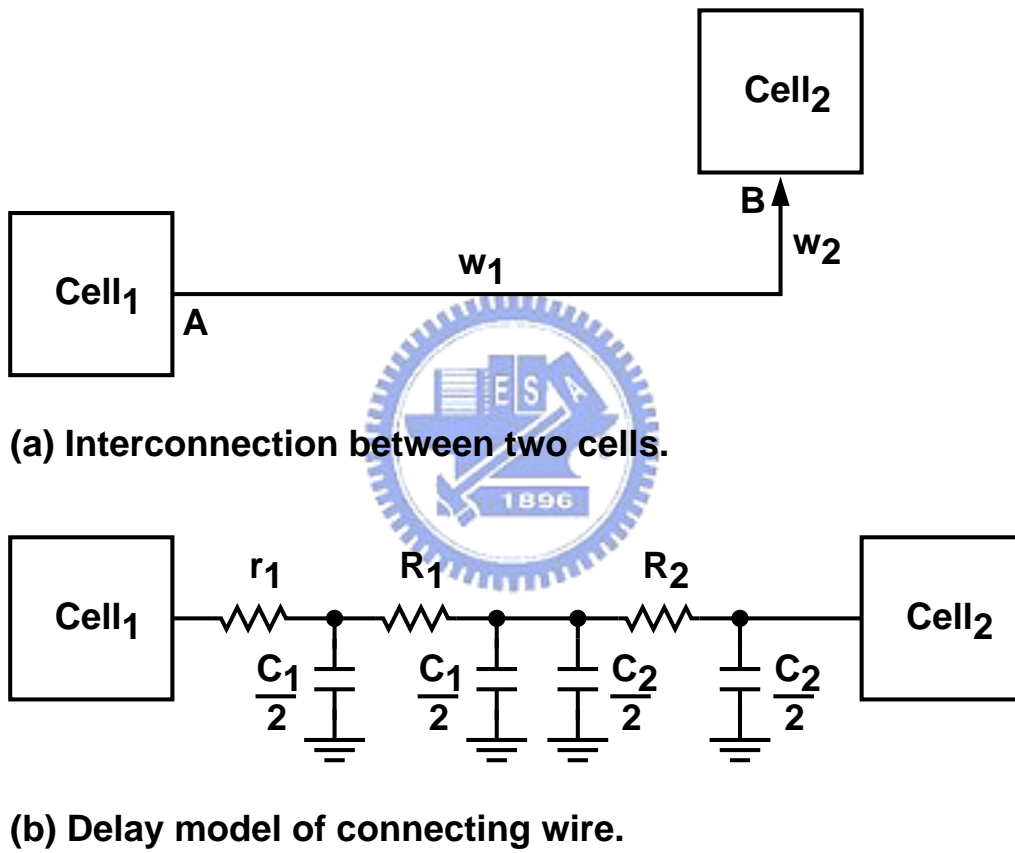


Figure 4.11: L-shaped approximation between two cells.

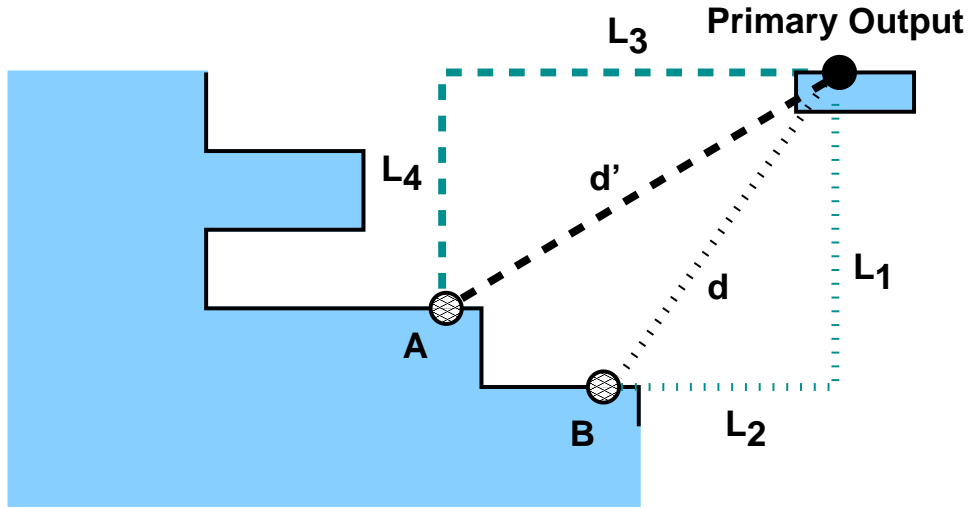


Figure 4.12: An example of wire delay estimation using L-shaped approximation.

Fig. 4.10 shows the  $\pi$ -model used as the wire delay model. Each  $w_i$  wire is modeled by a  $r_i$  resistor and two capacitors with the capacitance value of  $C_i/2$ . For two cells, the L-shape model shown in Fig. 4.11(a) is used for evaluate the delay. Use teh Elmore delay model [49] shown in Fig. 4.11(b) for calculating the delay given by

$$t_{d,A \rightarrow B} = r_1 C_l + \sum R_k C_k \quad (4.3)$$

where  $r_1$  is the output resistance of  $Cell_1$ ,  $C_l$  is the summed capacitance from  $r_1$  to the node B,  $R_k$  is the resistance of the  $w_k$  wire between nodes A and B,  $C_k$  is the summed capacitance from  $R_k$  to the node B. Hence, the  $t_{d,A \rightarrow B}$  is calculated as

$$\begin{aligned} t_{d,A \rightarrow B} = & r_1 \times \left( \frac{C_1}{2} + \frac{C_1}{2} + \frac{C_2}{2} + \frac{C_2}{2} + C_B \right) \\ & + R_1 \times \left( +\frac{C_1}{2} + \frac{C_2}{2} + \frac{C_2}{2} + C_B \right) + R_2 \times \left( \frac{C_2}{2} + C_B \right) \end{aligned} \quad (4.4)$$

Fig. 4.12 shows an example of wire delay estimation using L-shaped approximation.



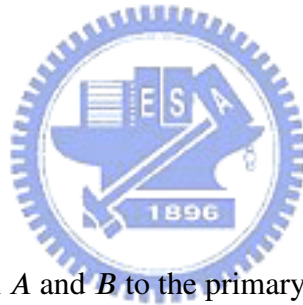
```

Procedure: Full Decomposition
Input:  {A, B} sets
Output: {C} set

begin
01  max ← max(A); // max: largest value of all elements in A
02  temp =  $\phi$ ;
03  for i = 1 to |B|
04    temp ← temp U { max - Bi };
05  C ← ( A \ { max } ) U temp;
end

```

Figure 4.13: Full Decomposition (FD) procedure.



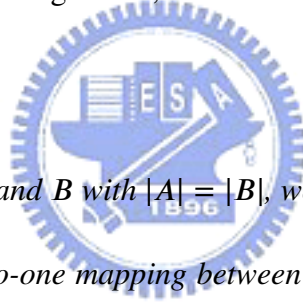
To estimate the wire delay from  $A$  and  $B$  to the primary output, the  $L_1 + L_2$  and  $L_3 + L_4$  wire lengths are used to calculate the delays respectively, when  $d \approx L_1 + L_2$  and  $d' \approx L_3 + L_4$  according to the L-shaped approximation.

By using 3-greedy algorithm [50] with  $\pi$ -model and L-shaped approximation, the upper bound of the arrival time for the final "Sum" of a VCS can be estimated before the VCS generation. To ensure the feasibility of the arrival time estimation during its reduction iteration in the VCS generation, the Feasibility Checking (FC) algorithm is developed, and is described in the following section.

### 4.2.2 Feasibility Checking Algorithm

To perform the FC algorithm, the *Full Decomposition* (FD) procedure is proposed to support VCS generation. The steps of the FD procedure are shown in Fig. 4.13. Firstly, the maximum element of set  $A$  is assigned to the  $max$  variable. Then, each element of set  $B$  is subtracted from  $max$ , and then all the resultant elements are collected into the  $temp$  set, which initially is an empty set. Finally, all elements of  $temp$  and those of  $A$  set except  $max$  are merged together to be  $C$  set.

Moreover, to examine the FC algorithm, the terminology "cover" is firstly defined as the following.



**Definition 1** Given two sets  $A$  and  $B$  with  $|A| = |B|$ , we say that  $A$  covers  $B$ , denoted as  $A \supseteq B$ , if there exists an one-to-one mapping between the element  $A_p$  and  $B_q$  such that  $A_p \geq B_q$ .

After describing the FD procedure, and defining the "cover" ( $\supseteq$ ), the feasibility is verified employing the FC algorithm shown in Fig. 4.14. As shown in Fig. 4.14, the feasibility checking is performed to check whether  $D$  set can be "decomposed" by  $F$  set through FD procedure such that the decomposed set,  $D'$ , satisfies  $D' \supseteq E$ . If it is true, the FC returns "TRUE"; else it returns "FALSE". As long as the number of elements in set  $D$  is less than that in set  $E$ , the FC algorithm continue to decompose set  $D$  using the FD procedure. If set  $D$  cannot be decomposed any more, find an element of set  $D$  to match

**Algorithm: Feasibility Checking****Input: { D, E, F } sets****Output: TRUE or FALSE****begin****01 while ( |D| < |E| )****02     if ( max(D) – min(F) >= max(E) )****03         D = Full Decomposition( D, F );****04     else****05         find  $d \in D$  and  $d \geq \max(E)$  such that  
           $|\max(E) - d|$  is minimum;****06         D  $\leftarrow$  D \ {d}, E  $\leftarrow$  E \ { max(E) };****07     if ( D  $\supseteq$  E )****08         return TRUE;****09     else****10         return FALSE;****end**

Figure 4.14: Feasibility Checking (FC) algorithm.

the maximum element of set  $E$ . Finally, the number of set  $D$  equals that of set  $E$ . Then, the FC algorithm returns "TRUE" or "FALSE" depending on whether  $D \supseteq E$  or not.

### Illustration of Feasibility Checking

Before constructing the VCS, we have to ensure the arrival time of the final "Sum" is shorter than its required time. A following example is illustrated to verify the FC algorithm.

- Given 3 sets,  $D$ ,  $E$  and  $F$ , where  $D$  is the set of initial required time for the final "Sum" of a VCS,  $E$  contains the arrival time of the compressed terms in this VCS, and  $F$  contains the path delays of a compressor. Assume that  $D = \{6\}$ ,  $E = \{1, 2, 2, 3, 4\}$  and  $F = \{1, 2, 3\}$ .

- 1) Fig. 4.15 shows the first step of the feasibility checking. Since  $|D| < |E|$ , and  $\max(D) = 6$ ,  $\min(F) = 1$ ,  $\max(E) = 4$ , we have  $\max(D) - \min(F) = 5 \geq \max(E)$ , which means the derived arrival time for the compressed term from the required time of the final "Sum" is larger than that of the compressed term which requires the maximum arrival time. Therefore, the timing violation will not happen, and  $\max(D)$  can be decomposed to three elements.
- 2) Fig. 4.16 shows the second step of the feasibility checking. Since  $|D| < |E|$ ,  $\max(D) = 5$ ,  $\min(F) = 1$ ,  $\max(E) = 4$ , we have  $\max(D) - \min(F) = 4 \geq$

$$D'_1 = \max(D) - F_1 = 6 - 1 = 5$$

$$D'_2 = \max(D) - F_2 = 6 - 2 = 4$$

$$D'_3 = \max(D) - F_3 = 6 - 3 = 3$$

$$D = \{5, 4, 3\}$$

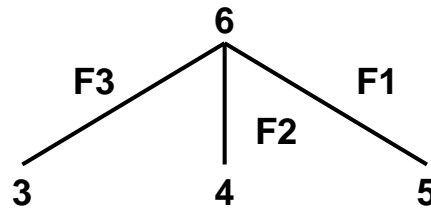


Figure 4.15: The first step of feasibility checking: decomposition.

$$D'_1 = \max(D) - F_1 = 5 - 1 = 4$$

$$D'_2 = \max(D) - F_2 = 5 - 2 = 3$$

$$D'_3 = \max(D) - F_3 = 5 - 3 = 2$$

$$D = \{4, 3, 2, 4, 3\}$$

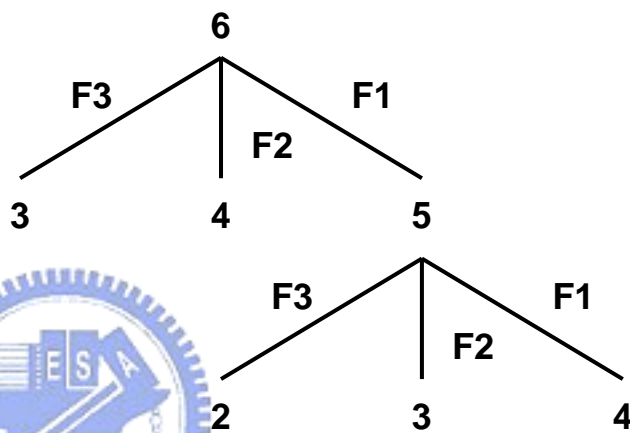


Figure 4.16: The second step of feasibility checking: further decomposition.

$\max(E)$ . Hence,  $\max(D)$ , can be further decomposed to three elements.

- 3) Fig. 4.17 shows the third step of the feasibility checking. Since  $|D| = |E|$ ,  $D$  is not decomposed any more. Through the first and the second steps, the set  $D$  contains the required time of the compressed terms of the VCS. For each  $D_i$  element of  $D$ , there exists an  $E_j$  element of  $E$ , such that  $D_i \geq E_j$  after the decomposition. If it is not true, the timing violation happens because the required time is smaller than the arrival time for some compressed terms.



```

Algorithm: VCS Generation
Input: sets { A, P } are the arrival time of compressed terms
          and the path delays of compressors respectively
Output: a VCS

begin
01  ini_sum <-- the arrival time of the final sum calculated by
    3-greedy algorithm;
02  while ( Feasibility Checking( ini_sum, A, P ) == TRUE)
03    ini_sum = ini_sum - 1;
04  ini_sum = ini_sum + 1;
    /* ini_sum is treated as the required time of sum of the final adder */
05  insert CPR0 ; /* CPR0 is the root of VCS */
06  L <-- the required time of 3 inputs of CPR0;
    /* L is the set of uncommitted leaves */
07  while ( there are uncommitted CPRs)
08    do
09    max_carry = max( req_carry_out of all uncommitted CPRs);
10    max_carry--;
11    if ( Feasibility Checking ( L, A, P ) == FALSE )
12      max_carry++;
13      set CPR1 to be committed;
        /* CPR1 is the CPR with max_carry */
14    for i = each(L)
15      if ( |L| < |A|)
16        try decomposition CPRi ;
17        if ( Feasibility Check (L, A, P ) == FALSE )
18          undo the decomposition CPRi ;
19        if ( Carry of CPRi < max_carry )
20          undo the decomposition CPRi ;
end

```

Figure 4.18: VCS generation algorithm.

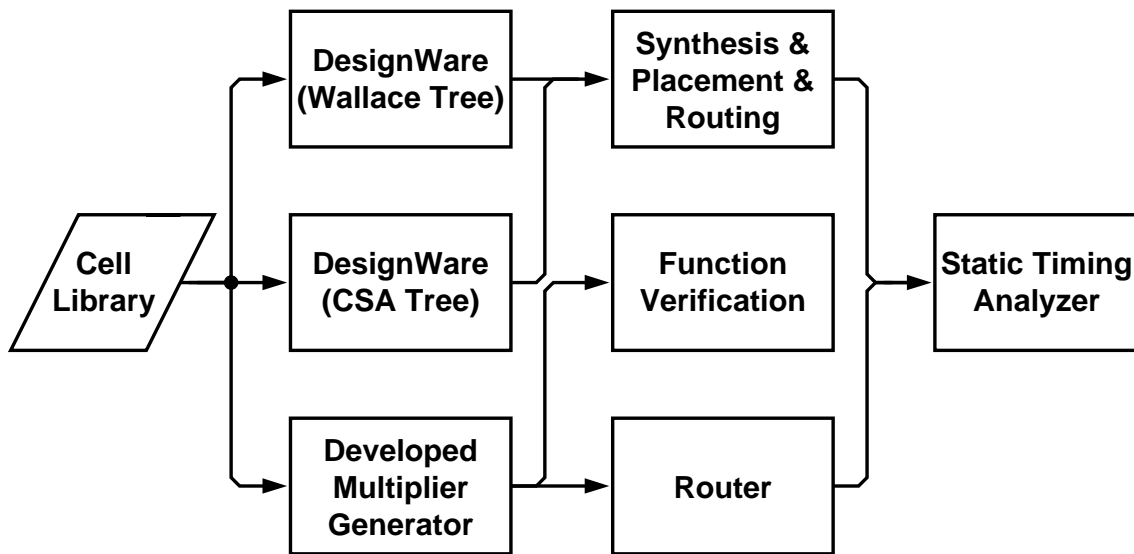


Figure 4.19: Experimental flow.



In the VCS Generation algorithm as shown in Fig. 4.18, line 1 applies the 3-greedy algorithm to evaluate the initial arrival time of the final "Sum", and then an upper bound of required time is obtained. Lines 2-4 are the calculation for the optimized required time of the final "Sum" using the FC algorithm. As long as there still exist uncommitted CPRs, VCS generation is performed iteratively. Lines 9-10 minimize the maximum required time of "Carry". If the required time of "Carry" is minimized, line 13 will set the CPR committed. If the number of uncommitted leaves is less than the number of compressed terms, lines 14-20 will attempt to insert CPRs. As long as a new CPR is inserted, the FC algorithm is used to make sure the inserted CPR is allowed.



Table 4.2: Experimental Results

		This Work	CSA	Wallace Tree
8 × 8	delay (ns)	3.369	3.888	3.298
	area (um <sup>2</sup> )	94343.34	146600.16	178478.16
12 × 12	delay (ns)	4.086	5.275	3.925
	area (um <sup>2</sup> )	156251.53	222530.84	258966.96
16 × 16	delay (ns)	4.660	6.943	4.405
	area (um <sup>2</sup> )	235366.42	324727.06	372566.54
24 × 24	delay (ns)	5.782	9.967	5.373
	area (um <sup>2</sup> )	424949.84	601216.0	716139.10
32 × 32	delay (ns)	6.235	12.775	5.523
	area (um <sup>2</sup> )	703340.65	930469.75	1105947.92

### 4.3 Experimental Results and Summary

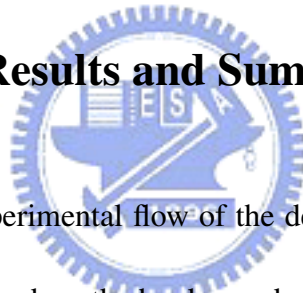


Fig. 4.19 shows the overall experimental flow of the developed layout-driven multiplier generator. This generator can produce the hardware description language (HDL) Verilog code, and the placed layout of a multiplier. Then, the Verilog code is used to verify the function of the multiplier. The placed layout is generated using the cell library with 0.35  $\mu\text{m}$ , 1P4M CMOS technology. This layout is then routed by the Synopsys's Apollo CAD tool. Finally, the static timing analysis is performed using Synopsys's Pathmill CAD tool.

Table 4.2 shows the comparisons of the critical delay and area of three schemes for variant sizes of multipliers. The layout-driven generated multiplier has smaller critical delay than the multiplier with CSA structure produced by commercial tools. Although

the delay of the developed multiplier is little longer than the multiplier in Wallace-tree structure produced by commercial tools, the area in the developed multiplier is much smaller than that of the Wallace-tree multiplier. Therefore, the layout-driven multiplier has better trade-off between performance and area than other two multipliers.



# Chapter 5

## Conclusions and Future Works

### 5.1 Conclusions



To handle the complexity of SoC designs, the system level design methodology, becomes an emerging issue. Design space exploration at system level is a critical step for trade-off between constraints. In this work, a NoC generator, an FFT generator, and a multiplier generator are developed to automatically produce the corresponding prototypes. These prototypes are used at system level, RTL and circuit level in the top-down design to reduce design time.

These generators rapidly generate optimized IPs according to the design constraints. The NoC generator is developed based on latency-insensitive concepts and employing the virtual-circuit switching to achieve high utilization, bandwidth guarantee, and the pre-

dictable latency under heavy traffic load. This NoC generator also employs the communication-aware task binding algorithm with the profile-driven optimization technique to reduce communication amount and contentions. The FFT generator uses the hybrid method combining the statistical analysis and simulation-based analysis to optimize wordlengths of an FFT processor. Hence, it can achieve high accuracy in the analysis, and also reduce the design exploration time. The Multiplier generator can produce a high-speed multiplier with small area based on gate delay optimization with considering wire delay.

## 5.2 Future Works



NoC is a trend for on-chip communication. Its contention effect is required to be analyzed, and hence, the ESL design methodology is used to construct NoC's modeling and simulation. If system designers can get the information at ESL, the architecture exploration will be possible. Hence, design, modeling, and verification at ESL are future study.

Wordlength optimization is important for system designs. This thesis studied wordlength in FFT module, but the crucial challenge is the wordlength optimization of a system. For communication systems, designers can model a system as a process network. For process networks, data can be exchanged using fixed-point representation in system optimization. Hence, to decide the fixed-point representation becomes an issue for system designs. This is difficult if the system is not a linear time-invariant (LTI) system. The system's charac-

teristics is not predictable if the system is not LTI. However, communication systems are usually not LTI systems. Therefore, to reduce the design time for system optimization is a research topic.

Multipliers are mostly often used for VLSI design. However, for deep sub-micron designs, local wire delay can affect the design performance, and hence, the wire delay must be tightly merged to gate delay, and the optimization of multipliers becomes complex. Simulation-annealing, and genetic algorithm are hence suitable for solving this problem.





# Bibliography

- [1] M.-L. Ku and C.-C. Huang, "A complementary codes pilot-based transmitter diversity technique for OFDM systems," *IEEE Transactions on Wireless Communications*, vol. 5, pp. 504–508, March 2006.
- [2] L. Cai and D. Gajski, "Transaction level modeling: An overview," *IEEE/ACM/IFIP International Conference on Hardware/Software and System Synthesis*, pp. 19–24, October 2003.
- [3] J. Dorsey, S. Searles, M. Ciraula, S. Johnson, N. Bujanos, D. Wu, M. Braganza, S. Meyers, E. Frang, and R. Kumar, "An integrated quad-core Opteron processor," *IEEE International Solid-State Circuits Conference Digest of Technical Papers*, pp. 102–103, 2007.
- [4] J. Song, T. Shepherd, M. Chau, A. Huq, I. Syed, S. Roy, A. Thippana, K. Shi, and U. Ko, "A low power open multimedia application platform for 3G wireless," *IEEE International Conference on Systems-on-Chip*, pp. 377–380, 2003.

- [5] M. Dillinger, K. Madani, and N. Alonistioti, “Software defined radio : architectures, systems, and functions,” *John Wiley & Sons Ltd*, 2003.
- [6] G.-H. Chen, “Design methodology at electronic system level: A case study of OFDM system,” *Master Dissertation, National Chiao Tung University, Taiwan*, August 2006.
- [7] L. Benini and G. D. Micheli, “Networks on chips: a new SoC paradigm,” *IEEE Computer Magazine*, vol. 35, pp. 70–78, January 2002.
- [8] A. Jantsch and H. Tenhunen, “Networks on chip,” *Kluwer Academic Publishers*, 2003.
- [9] D. Densmore, A. Sangiovanni-Vincentelli, and R. Passerone, “A platform-based taxonomy for ESL design,” *IEEE Design & Test of Computers*, vol. 23, pp. 359–374, September–October 2006.
- [10] J. Hu and R. Marculescu, “Energy- and performance-aware mapping for regular NoC architecture,” *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 24, pp. 551–562, April 2005.
- [11] E. Rijpkema, K. Goossens, A. Radulescu, J. Dielissen, J. van Meerbergen, P. Wielage, and E. Waterlander, “Trade-offs in the design of a router with both guar-



- anteed and best-effort services for networks on chip,” *Proceedings of ACM/IEEE Design Automation and Test in Europe*, pp. 350–355, 2003.
- [12] P. P. Pande, C. Grecu, A. Ivanov, R. Saleh, and G. D. Micheli, “Design, synthesis, and test of networks on chips,” *IEEE Design & Test of Computers*, vol. 22, pp. 404–413, September–October 2005.
- [13] W. J. Dally, “Performance analysis of a k-ary n-cube interconnect networks,” *IEEE Transactions on Computers*, vol. 39, pp. 775–785, June 1990.
- [14] L. P. Carloni, K. L. McMillan, A. Saldanha, and A. L. Sangiovanni-Vincentelli, “A methodology for correct-by-construction latency insensitive design,” *Proceedings of IEEE/ACM International Conference on Computer-Aided Design*, pp. 309–315, 1999.
- [15] L. P. Carloni and A. L. Sangiovanni-Vincentelli, “Coping with latency in soc design,” *IEEE Micro*, vol. 22, pp. 24–35, October 2002.
- [16] J. Y. Chang, W. J. Kim, Y. H. Bae, J. H. Han, H. J. Cho, and H. B. Jung, “Performance analysis for MPEG-4 video codec based on on-chip network,” *ETRI Journal*, vol. 27, pp. 497–503, 2005.
- [17] A. Marquardt, V. Betz, and J. Rose, “Timing-driven placement for FPGAs,” *ACM Symposium on FPGAs*, pp. 203–213, 2000.

- [18] C. Ebeling, L. McMurchie, S. A. Hauck, and S. Burns, "Placement and routing tools for the Triptych FPGA," *IEEE Transactions on Very Large Scale Integration Systems*, vol. 3, pp. 473–482, December 1995.
- [19] C. Y. Lee, "An algorithm for path connections and its applications," *IRE Transactions Electron Computing*, vol. 10, pp. 346–365, 1961.
- [20] E. Dijkstra, "A note on two problems in connexion with graphs," *Numerical Math*, vol. 1, pp. 269–271, 1959.
- [21] J. Duato, S. Yalamanchili, and L. Ni, "Interconnection networks: An engineering approach," *Morgan Kaufmann*, 2003.
- [22] R. P. Dick, D. L. Rhodes, and W. Wolf, "TGFF: task graphs for free," *Proceedings of the 6th IEEE International Workshop on Hardware/Software Codesign*, pp. 97–101, 1998.
- [23] J. W. Cooley and J. W. Tukey, "An algorithm for the machine calculation of complex fourier series," *Mathematics Computation*, vol. 19, pp. 297–301, 1965.
- [24] L. R. Rabiner and G. Gold, "Theory and application of digital signal processing," *Prentice-Hall, Inc.*, 1975.
- [25] E. H. Wold and A. M. Despain, "Pipelined and parallel-pipeline FFT processors for

- VLSI implementation,” *IEEE Transactions on Computers*, vol. C-33(5), pp. 414–426, May 1984.
- [26] S.-S. He and M. Torkelson, “A new approach to pipeline FFT processor,” *Proceedings of the 10<sup>th</sup> IEEE International Parallel and Distributed Processing Symposium*, pp. 766–770, 1996.
- [27] S.-S. He and M. Torkelson, “Designing pipeline FFT processors for OFDM (de)modulation,” *Proceedings of 1998 URSI International Symposium on Signals, Systems, and Electronics*, pp. 257–262, October 1998.
- [28] J.-C. Choi, W.-C. Choi, S.-G. Hwang, M. M.-O. Lee, and K.-R. Cho, “Design of a new IFFT/FFT for IEEE 802.11a WLAN based on the statistics distribution of the input data,” *IEEE Proceedings on High-Speed Networks and Multimedia Communications*, vol. 3079, pp. 589–597, June 2004.
- [29] J. yeol Oh and M. seob Lim, “Area and power efficient pipeline FFT algorithm,” *IEEE Workshop on Signal Processing Systems Design and Implementation*, pp. 520–525, November 2005.
- [30] L. Yang, K. Zhang, H. Liu, J. Huang, and S. Huang, “An efficient locally pipelined FFT processor,” *IEEE Transactions on Circuits and Systems II: Express Briefs*, vol. 53, pp. 585–589, July 2006.

- [31] P. D. Welch, "A fixed-point fast fourier transform error analysis," *IEEE Transactions on Audio and Electroacoustics*, vol. AU-17, pp. 151–157, June 1969.
- [32] A. V. Oppenheim and C. J. Weinstein, "Effects of finite register length in digital filtering and the fast fourier transform," *IEEE Proceedings*, vol. 60, pp. 957–976, August 1972.
- [33] R. Meyer, "Error analysis and comparison of FFT implementation structures," *IEEE International Conference on Acoustics, Speech and Signal Processing*, vol. 2, pp. 888–891, May 1989.
- [34] A. Pomerleau, H. L. Buijs, and M. Fournier, "A two-pass fixed point fast fourier transform error analysis," *IEEE Transactions on Acoustics, Speech and Signal Processing*, vol. 25, pp. 582–585, December 1977.
- [35] S. Johansson, S. He, and P. Nilsson, "Wordlength optimization of a pipelined FFT processor," *Proceedings of IEEE International Midwest Symposium on Circuits and Systems*, vol. 1, pp. 501–503, 1999.
- [36] A. V. Oppenheim and R. W. Schaffer, "Discrete-time signal processing," *Second Edition*, Prentice Hall, 1999.
- [37] R. E. Walpole, R. H. Myers, and S. L. Myers, "Probability and statistics for engineers and scientists," *Sixth Edition*, Prentice Hall, 1998.

- [38] “SystemC Version 2.0 User’s Guide,” <http://www.systemc.org>, 2002.
- [39] J. Bhasker, “A SystemC Primer,” *Star Galaxy*, 2002.
- [40] W. C. Yeh, “Arithmetic module design and its application to FFT,” *PhD. Dissertation, National Chiao Tung University, Taiwan*, July 2001.
- [41] “Synopsys Design Analyzer,” <http://www.synopsys.com>, 2000.
- [42] “Synopsys Design Ware,” <http://www.synopsys.com>, 2000.
- [43] “Artisan TSMC 0.25 $\mu$ m process high-density dual-port SRAM (HD-SRAM-DP) generator user manual, release 1.0,” <http://www.artisan.com>, June 2000.
- [44] “Artisan TSMC 0.25 $\mu$ m process high-speed single-port SRAM (HS-SRAM-SP) generator user manual, release 3.0,” <http://www.artisan.com>, June 2000.
- [45] H. A. Al-Twaijry, “Area and performance optimized CMOS multipliers,” *PhD Thesis, Stanford University*, August 1997.
- [46] C. S. Wallace, “A suggestion for a fast multiplier,” *IEEE Transactions on Computers*, vol. 13, pp. 14–17, February 1964.
- [47] V. G. Oklobdzija, D. Vileger, and S. S. Liu, “A method for speed optimized partial product reduction and generation of fast parallel multipliers using an algorithmic approach,” *IEEE Transactions on Computers*, vol. 45, pp. 294–306, March 1996.

- [48] P. F. Stelling, C. U. Martel, V. G. Oklobdzija, and R. Ravi, "Optimal circuits for parallel multipliers," *IEEE Transactions on Computers*, vol. 47, pp. 273–285, March 1998.
- [49] W. C. Elmore, "The transient response of damped linear networks with particular regard to wide-band amplifiers," *Journal of Applied Physics*, vol. 19, pp. 55–63, January 1948.
- [50] S. F. Oberman and M. J. Flynn, "Design issues in division and other floating-point operations," *IEEE Transactions on Computers*, vol. 46, pp. 154–161, February 1997.

