

University of Windsor

## Scholarship at UWindor

---

Electronic Theses and Dissertations

Theses, Dissertations, and Major Papers

---

1979

### A DESIGN PROCEDURE FOR A ROM IMPLEMENTATION OF A RESIDUE NUMBER SYSTEM BASED FFT PROCESSOR.

BEN-DAU. TSENG

*University of Windsor*

Follow this and additional works at: <https://scholar.uwindsor.ca/etd>

---

#### Recommended Citation

TSENG, BEN-DAU., "A DESIGN PROCEDURE FOR A ROM IMPLEMENTATION OF A RESIDUE NUMBER SYSTEM BASED FFT PROCESSOR." (1979). *Electronic Theses and Dissertations*. 596.

<https://scholar.uwindsor.ca/etd/596>

This online database contains the full-text of PhD dissertations and Masters' theses of University of Windsor students from 1954 forward. These documents are made available for personal study and research purposes only, in accordance with the Canadian Copyright Act and the Creative Commons license—CC BY-NC-ND (Attribution, Non-Commercial, No Derivative Works). Under this license, works must always be attributed to the copyright holder (original author), cannot be used for any commercial purposes, and may not be altered. Any other use would require the permission of the copyright holder. Students may inquire about withdrawing their dissertation and/or thesis from this database. For additional inquiries, please contact the repository administrator via email ([scholarship@uwindsor.ca](mailto:scholarship@uwindsor.ca)) or by telephone at 519-253-3000ext. 3208.





National Library of Canada  
Collections Development Branch

Canadian Theses on  
Microfiche Service

Bibliothèque nationale du Canada  
Direction du développement des collections

Service des thèses canadiennes  
sur microfiche

## NOTICE

The quality of this microfiche is heavily dependent upon the quality of the original thesis submitted for microfilming. Every effort has been made to ensure the highest quality of reproduction possible.

If pages are missing, contact the university which granted the degree.

Some pages may have indistinct print especially if the original pages were typed with a poor typewriter ribbon or if the university sent us a poor photocopy.

Previously copyrighted materials (journal articles, published tests, etc.) are not filmed.

Reproduction in full or in part of this film is governed by the Canadian Copyright Act, R.S.C. 1970, c. C-30. Please read the authorization forms which accompany this thesis.

THIS DISSERTATION  
HAS BEEN MICROFILMED  
EXACTLY AS RECEIVED

## AVIS

La qualité de cette microfiche dépend grandement de la qualité de la thèse soumise au microfilmage. Nous avons tout fait pour assurer une qualité supérieure de reproduction.

S'il manque des pages, veuillez communiquer avec l'université qui a conféré le grade.

La qualité d'impression de certaines pages peut laisser à désirer, surtout si les pages originales ont été dactylographiées à l'aide d'un ruban usé ou si l'université nous a fait parvenir une photocopie de mauvaise qualité.

Les documents qui font déjà l'objet d'un droit d'auteur (articles de revue, examens publiés, etc.) ne sont pas microfilmés.

La reproduction, même partielle, de ce microfilm est soumise à la Loi canadienne sur le droit d'auteur, SRC 1970, c. C-30. Veuillez prendre connaissance des formules d'autorisation qui accompagnent cette thèse.

LA THÈSE A ÉTÉ  
MICROFILMÉE TELLE QUE  
NOUS L'AVONS REÇUE

A DESIGN PROCEDURE FOR A ROM IMPLEMENTATION  
OF A RNS BASED FFT PROCESSOR

by

© BEN-DAU TSENG

A Dissertation

Submitted to the Faculty of Graduate Studies  
through the Department of Electrical Engineering  
in Partial fulfillment of the requirements for  
the Degree of Doctor of Philosophy at The  
University of Windsor

Windsor, Ontario

1979

© Ben-Dau Tseng 1979

723862

## ABSTRACT

This thesis considers some of the design problems associated with a read-only-memory implementation of a Residue Number System based Fast Fourier Transform processor. The principal design parameters identified in this work are the radix of the FFT structure, the number range associated with the processor, the values of scale factors, the manner in which scaling is distributed throughout the processor, the coefficients associated with the integer conversion of twiddle factors and the resolution, in terms of bits, that the related analogue-to-digital converter provides.

The value of the optimal radix that minimizes the number of cascaded multiplications within the FFT structure has been found to equal four. Using this radix an expression for the theoretical upper bound for number growth in the FFT processor has been derived. Simulation studies have also been used to determine the number growth associated with sample functions that characterize typical inputs that would be applied to the processor.

The specification of a desired mean-squared-error of the frequency domain estimate computed by the FFT processor has been found as a useful criterion of optimality for the design procedure. A comprehensive study of quantization error sources was required in order to provide the theoretical basis for determining how the various design parameters influence the error of the estimate. This analysis has ultimately led to

a design procedure in which the values of design parameters that simplify the hardware realization and meet a specified error criterion can be determined.

## ACKNOWLEDGEMENTS

The author wishes to express his sincere appreciation to his supervisor, Dr. W. C. Miller, for many valuable discussions and constructive criticisms throughout the study period. The advice and assistance of Dr. G. A. Jullien is gratefully acknowledged. Thanks are due to the other members of the department who helped the author in different ways.

Thanks and gratitude are also due to my wife for her diligence and perseverance in the typing of this dissertation.



## TABLE OF CONTENTS

	<u>Page</u>
ABSTRACT.....	i
ACKNOWLEDGEMENTS.....	iii
LIST OF FIGURES.....	vi
LIST OF TABLES.....	viii
LIST OF SYMBOLS.....	x
LIST OF APPENDICES.....	xii
CHAPTER 1: INTRODUCTION.....	1
1.1 The Residue Number System and A ROM Oriented Implementation.....	1
1.2 Conventional FFT Algorithms and Implementations.....	2
1.3 Objectives of The Research.....	9
1.4 Organization of The Thesis.....	10
CHAPTER 2: THE RESIDUE NUMBER SYSTEM AND ITS APPLICATION TO A FFT PROCESSOR.....	12
2.1 Residue Number System Concepts.....	12
2.2 The Fast Fourier Transform.....	17
2.3 A ROM Oriented FFT Structure.....	18
2.4 Identification of Design Parameters....	24
CHAPTER 3: FFT PROCESSOR CONSIDERATIONS.....	26
3.1 Optimal Radix Determination.....	26
3.2 Number Growth.....	29
2.1 Mean-square Bound.....	30
2.2 Theoretical Worst Case Upper Bound.....	32
2.3 Experimental Worst Case Upper Bound....	35
3.3 Quantization Error Sources.....	39
CHAPTER 4: A DESIGN PROCEDURE FOR A FFT PROCESSOR.	41
4.1 Design Criteria.....	41
4.2 General Design Parameter Relationships.....	41
2.1 Statistical Error Models.....	41
2.2 RMS Quantization Error Analysis.....	45
4.3 A Simplified Design Procedure.....	56
4.4 Other Considerations.....	63
4.1 Optimal Scaling Scheme:.....	63
4.2 Scale Factor Considerations.....	70
4.3 Sequential Realization Considerations..	77

	<u>Page</u>
4.5 An Example Design Problem.....	82
CHAPTER 5: DISCUSSION OF RESULTS.....	85
5.1 Introduction.....	85
5.2 Simulation of The FFT Processor.....	85
5.3 Comparison of Theoretical and Simulation Results.....	87
5.4 Ramifications.....	105
4.1 Fixed-point Arithmetic Number System.	105
4.2 The DIF Algorithm.....	109
CHAPTER 6: CONCLUSIONS.....	113
APPENDICES .....	116
REFERENCES .....	139
VITA AUCTORIS .....	143

## LIST OF FIGURES

<u>Figure</u>	<u>Page</u>
1.1 Block Diagram of A FFT Basic Calculation Unit and Its Auxiliary Memories.....	6
1.2 Block Diagrams of Five Different Realizations.....	6
2.1 Block Diagram of A ROM for A Modulus $m_i$ .....	14
2.2 Pipelined Array for The Function $ (a \cdot b) + (c \cdot d) _{m_i}$ ...	16
2.3a Simplified Representation of An r-Point Transform DIT Algorithm.....	19
2.3b Simplified Representation of An r-Point Transform DIF Algorithm.....	19
2.4a Simplified Integer Representation of An r-Point Transform DIT Algorithm.....	21
2.4b Simplified Integer Representation of An r-Point Transform DIF Algorithm.....	21
2.5 Conceptual Block Diagram of A FFT Processor.....	23
3.1 Radix-4 DIT Basic Calculation.....	36
4.1 Basic Module of The i-th Stage of A Radix-4 FFT Processor.....	47
4.2 Scaling Schemes of The FFT Processor.....	53
4.3 Relative RMS Output Error vs Number Range for Various Scaling Schemes; $N=1024$ .....	65
4.4 Relative RMS Output Error vs Scale Factor for Various Scaling Schemes; $N=1024$ .....	66
4.5 Relative RMS Scaling Output Error vs Scale Factor; $M=10^5$ , $N=1024$ .....	76

<u>Figure</u>	<u>Page</u>
4.6a Twiddle Factor Multiplications .....	80
4.6b Artificial Twiddle Factor Multiplications at The First Stage.....	80
5.1a Relative RMS Output Error vs Number Range for $S_1$ ; $k_1 = 1$ , $k = 2$ and $k_1 = 2$ , $k = 1$ .....	92
5.1b Relative RMS Output Error vs Number Range for $S_1$ ; $k_1 = 2$ , $k = 3$ and $k_1 = 3$ , $k = 1$ .....	93
5.2 Relative RMS Output Error vs Number Range for $S_1$ and $S_2$ ; $k_1 = 2$ , $k = 1$ .....	95
5.3 Relative RMS Output Error vs Number Range for $S_3$ ; $k_1 = 2$ , $k = 1$ .....	96
5.4 Relative RMS Output Error vs Number Range for $S_4$ ; $k_1 = 2$ , $k = 1$ .....	98
5.5 Relative RMS Output Error vs Number Range for $S_1$ and $S_2$ ; $p_2 \neq P$ , $k_1 = 2$ , $k = 1$ .....	99
5.6 Relative RMS Output Error vs Number Range for $S_1$ , $S_2$ and $S_4$ ; $p_1 = P$ , $k_1 = 1$ , $k = 1$ ....	100
5.7 Relative RMS Output Error vs Number Range for $S_1$ and $S_2$ with Real Input; $k_1 = 2$ , $k = 1$ .	102
5.8 Relative RMS Output Error vs Number Range for $S_3$ with Real Input; $k_1 = 2$ , $k = 1$ .....	103
5.9a Basic Module of An Original Radix-4 DIF FFT.	110
5.9b Basic Module of A Modified Radix-4 DIF FFT..	110

## LIST OF TABLES

<u>Table</u>	<u>Page</u>
1.1 Comparison of Five Different FFT Realizations.....	7
2.1 Number of Cascaded Non-integer Multiplications for Radices of 2, 4, 8 and 16.....	28
3.1a Theoretical Worst Case Upper Bound of The Number Growth at Each Stage of A Radix-4 DIT FFT.....	34
3.1b Theoretical Worst Case Upper Bound of The Number Growth at Each Stage of A Radix-4 DIF FFT.....	34
3.2 Simulation Study of Worst Case Upper Bound of The Number Growth at Each Stage of A Radix-4 DIT FFT.....	38
4.1 Optimum Functional Relationships between Various Parameters of A Radix-4 DIT FFT; N=1024.....	62
4.2 Comparison of The Total Number of Look-up Tables Required for The Cascade Realizations for Four Different Scaling Scheme; N=1024 $\alpha_1=0.01$ .....	69
4.3 Optimum Functional Relationships between $\alpha_1$ , M and P for Various N; $k_1=2$ , $k=1$ .....	71
4.4 Comparison of Optimum Functional Relationships of Various Parameters between Two Different Cases, $p_2=P$ and $p_2 \neq P$ ; $k_1=2$ , $k=1$ and N=1024.....	75

<u>Table</u>	<u>Page</u>
5.1 Properties of Simulation Input Sequences.....	90
5.2 Simulation Study of Worst Case Upper Bound of The Number Growth at Each Stage of A Real Input Radix-4 DIT FFT.....	104
5.3 Comparison of Word Length Required at Various Units of A Radix-4 FFT Processor for Two Different Methods Using Fixed-point Arithmetic...	108
5.4 Simulation Study of Worst Case Upper Bound of The Number Growth at Each Stage of A Modified Radix-4 DIF FFT.....	111

# LIST OF SYMBOLS

B	A/D converter width in bits
$E_{IN}^i(n)$	error at the input of the i-th stage
$E_{DFT}^i(n)$	error at the input to the 4-point DFT of the i-th stage
$E_{OUT}^i(n)$	error at the output of the i-th stage
$e_Q$	A/D converter quantization error
$e_I$	twiddle factor integer conversion rounding error
$e_S$	scaling rounding error
$j$	$\sqrt{-1}$
K	scale factor
k	number of stages between scaling operations
$k_i$	number of stages between the (i-1)-th and i-th scaling operations
$k_q$	number of stages after the (q-1)-th scaling operation
L	total number of moduli used in the RNS
M	number range of the RNS
m	total number of stages in the FFT
$m_i$	the i-th modulus of the RNS
N	total number of samples in the FFT
P	integer conversion constant
$P_i$	integer conversion factor at the i-th stage
q	number of scaling operations required to produce a scaled output
r	radix of the FFT

$S$	number of scaling moduli
$W_N$	$\exp(-j2\pi/N)$
$x_{IN}^i(n)$	true value at the input of the $i$ -th stage
$x_{DFT}^i(n)$	true value at the input to the 4-point DFT of the $i$ -th stage
$x_{OUT}^i(n)$	true value at the output of the $i$ -th stage
$\hat{x}_{IN}^i(n)$	scaled integer value at the input of the $i$ -th stage
$\hat{x}_{DFT}^i(n)$	scaled integer value at the input to the 4-point DFT of the $i$ -th stage
$\hat{x}_{OUT}^i(n)$	scaled integer value at the output of the $i$ -th stage
$\alpha_1$	relative RMS output error
$\alpha_Q^2$	relative mean-squared output error due to A/D converter quantization
$\alpha_I^2$	relative mean-squared output error due to integer conversion rounding
$\alpha_S^2$	relative mean-squared output error due to scaling rounding
$\sigma^2$	mean-squared value of the input sequence
$\text{Im}(\cdot)$	imaginary part of the term enclosed
$\text{Re}(\cdot)$	real part of the term enclosed
$[\cdot]_R$	the closest integer to the term enclosed



## LIST OF APPENDICES

	<u>Page</u>
APPENDIX A. Number Growth at Each Stage of A Radix-r DIF FFT.....	117
APPENDIX B. An Error Analysis of A Sequentially Realized Radix-4 DIT FFT.....	120
APPENDIX C. Simulation Details.....	122
APPENDIX D. An Error Analysis of A Radix-4 DIT FFT with Real Input.....	131
APPENDIX E. An Error Analysis of A Rounding Effect on The Output of The Twiddle Factor Multiplications.....	135

## CHAPTER 1

### INTRODUCTION

#### 1.1 The Residue Number System and A ROM Oriented Implementation

The residue number system (RNS) has received varying degrees of attention from workers in the field during the last two decades. In the early to middle 1960's, considerable work was done on implementation of RNS arithmetic for general purpose computing [1], [2]. After years of work much was learned about RNS arithmetic, it was essentially abandoned for general purpose computing because sign detection, magnitude comparison and general division are difficult operations, although fast and simple integer operations of addition and multiplication were demonstrated [1].

However, with the recent advances in high density memories technology that appears eminently suited to performing high speed operations in the parallel RNS structure, interest in RNS arithmetic has been revived. Although some current work is being directed at the problem of building a general floating point arithmetic processor, using the RNS [3], a number of more immediate applications appear to lie in the direction of special purpose digital signal processing hardware, where for many of their operations a pure integer number system such as RNS is all that is required.

Recent, independent, investigators have discussed the RNS based realization of high speed digital filters. Jenkins and Leon [4] investigated RNS techniques for non-recursive filters,

while Soderstrand [5] and Jullien [6] discussed the implementations of recursive filters. The concept of a read-only-memory (ROM) oriented implementation of the fast Fourier transform (FFT) based on the RNS has been described by Tseng, Miller, Jullien et al [7]. The common feature that emerges from these works, is the use of ROMs to provide parallel arrays of look-up tables for performing the RNS arithmetic operations.

In order to apply RNS principles to hardware structures, it is necessary to convert the conventional digital number system into and out of the residue code. The conversion from binary input into the residue code can be easily implemented using ROMs [6]. But the conversion from residue code into an analog voltage or a binary code is, in general, more difficult. A technique based on a mixed radix conversion to convert residue code into a binary code has been discussed by Baraniecka and Jullien [40]. Two approaches to the design of residue-to-analog conversion were described by Jenkins [8].

### 1.2 Conventional FFT Algorithms and Implementations

The FFT was first introduced by Cooley and Tukey [9], although it has a long complicated history as described by Cooley, Lewis and Welch [10]. The FFT algorithm is an efficient way of computing the discrete Fourier transform (DFT) of a time series data. Since the DFT is an important computation in most digital signal processing problems, the FFT algorithms have been explored very quickly. Detailed explanation and derivation of the FFT algorithms can be found

in [11] . Examples of applications to which the FFT has contributed were also discussed in [11] .

In general, there are two basic versions of the FFT algorithms, namely, the decimation-in-time (DIT) algorithm and the decimation-in-frequency (DIF) algorithm [11] . Both of the two versions have various structures which require ordered or scrambled input and generate scrambled or ordered output. The radix-2 DIT algorithm with scrambled input and ordered output, and the DIF algorithm with ordered input and scrambled output are two well-known ones, because their twiddle factors can be easily generated recursively.

In fact, the radix-2 algorithms are special cases for their simplicity. The efficiency of FFT computation can be improved by using a higher radix algorithm. If the number of samples,  $N$ , is an integer power of 2, then the higher the radix is, the better the efficiency is [12] . However, the program becomes more and more complicated for higher radix algorithms. Thus, in most applications only radix 2, 4, or 8 has been used. If, however, for some problems  $N$  is not an integer power of 2, one can always append enough number of zeros to make  $N$  an integer power of 2 such that a radix 2, 4 or 8 algorithm can be used. Alternatively, a mixed radix FFT algorithm can be used, if  $N$  can be decomposed into a product of some small factors [13] .

Recently, the concept of the conversion of a DFT to convolution has been used to develop two new algorithms, which are called prime-factor FFT and Winograd Fourier transform

algorithms [14], [15]. Using these algorithms, the number of samples,  $N$ , must be a product of some relatively pairwise prime factors. By using the Chinese Remainder Theorem to regroup the input data, these two new algorithms do not require any twiddle factors, while individual factors' DFTs are written in a very efficient way to minimize the number of multiplications.

Since the development of the FFT algorithm by Cooley and Tukey [1], many efforts have been spent on finding faster hardware implementations. Due to the sequential operations of a general purpose computer, software implementations can not operate at a very high speed. For some problems, which have an inherent real time constraint or have a large volume of data, speed becomes very important. Thus, a special purpose hardware designed for performing the FFT algorithm is required.

Figure 1.1 shows a block diagram of a basic calculation unit (BCU) and its auxiliary memories for a FFT processor. The BCU performs the required  $r$ -point DFT operations and twiddle factor multiplications. The data memory may consist of several submemories, which depends on how it is realized. In most implementations, the BCU is implemented using TTL logic. Recently, some of the BCUs are implemented by mixing TTL logic and ROMs [16].

Based on the forms of the realizations, Bergland [17] divided the FFT hardware processor into four different ones, namely, the sequential, cascade and parallel iterative processors, and the array analyzer. In addition, there is another

type of realization, which will be called partial parallel iterative processor. These five different realizations are shown in Figure 1.2. Table 1.1 also shows some of the features of the five different realizations, where  $N$  is the total number of stages. The first realization is the simplest but slowest one. The costs of the fourth and fifth realizations are very high, which limit their applications. For a higher speed system, the second realization is usually used. For efficient implementation, the number of BCUs required in the third realization,  $p$ , must be an integer power of  $r$ . When  $p$  is equal to  $r$ , they are called Dual-2 and Quad-4 processor for radices of 2 and 4, respectively [18]. In fact, when  $r \leq m \leq 2r$ , the third realization with  $p$  equal to  $r$  is better than the second realization as far as both the speed and cost are concerned.

Due to the huge amount of literature, a complete survey of FFT hardware implementations is not attempted. In order to show the tendency of FFT hardware realization, a discussion of some realization will now be given.

Bergland and Hale [19] first proposed a cascade FFT processor to compute spectra, where the twiddle factors were generated recursively. Using the principle introduced by Stockham [20], O'leary described a non-recursive digital filter using cascade FFT [21], where two independent data can be transformed simultaneously. Groginsky and Works [22] proposed a similar structure, and introduced a technique of pipelining, such that the new data can enter

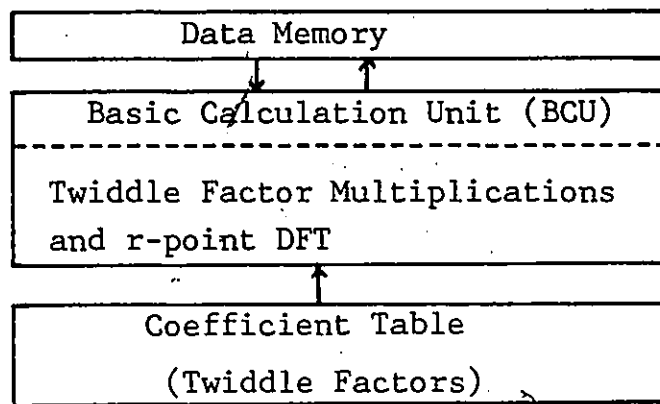


FIG. 1.1 BLOCK DIAGRAM OF A FFT BASIC CALCULATION UNIT AND ITS AUXILIARY MEMORIES

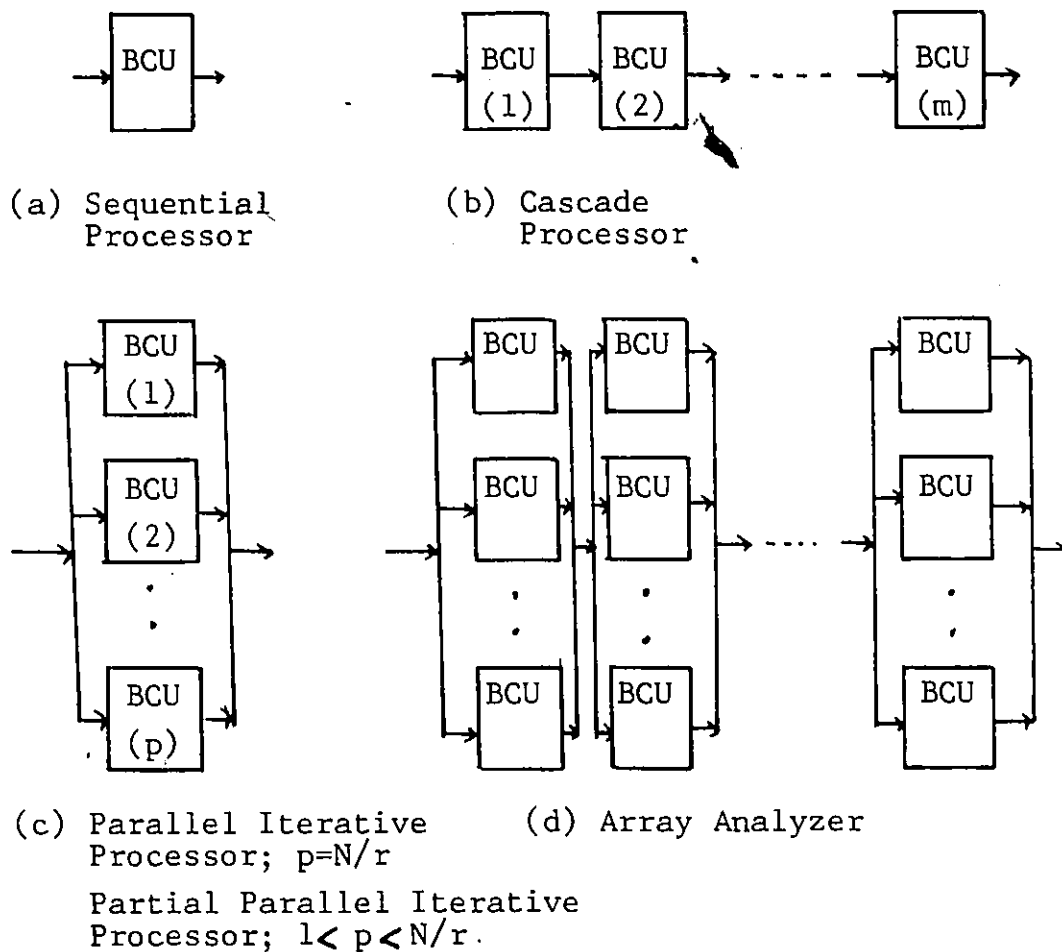


FIG 1.2 BLOCK DIAGRAMS OF FIVE DIFFERENT REALIZATIONS

TABLE 1.1  
COMPARISON OF FIVE DIFFERENT FFT REALIZATIONS

Realization ( $N = r^m$ )	Basic Calculation Unit	Operations in Parallel	Performed Sequential	Execution Time
(1) Sequential	1	1	$m \cdot (N/r)$	$m \cdot (N/r) \cdot D^*$
(2) Cascade	m	m	$N/r$	$2 \cdot (N/r) \cdot D$
(3) Partial Parallel Iterative	$p^{**}$	p	$m \cdot ((N/r)/p)$	$m \cdot ((N/r)/p) \cdot D$
(4) Parallel Iterative	$N/r$	$N/r$	m	$mD$
(5) Array	$m \cdot (N/r)$	$m \cdot (N/r)$	1	D

\* D is the time required for performing one basic calculation

\*\*  $1 < p < N/r$



continuously into the processor, while the processing of earlier data is carried out. Veenkant [18] introduced a different type of realization which uses  $r$  BCUs to increase the number of parallel operations at each stage.

Pease [23] brought out an idea to use slow memory efficiently by splitting main memory into several submemories. Corinthios [24 - 27] utilized Pease's idea and reported a series of improved realizations. He first proposed a sequential processor with a wired-in design to minimize control circuitry [24]. Two memories, input and output memories, were used. The output memory collects the intermediate results and feeds them back to the input memory, which becomes the input data for the next stage. Using a binary counter to control the ordering of feedback from the output memory, the processor can receive input data in natural order and generate output data in natural order, too. Then he developed a machine oriented FFT algorithm [25]; where both the input and output memories were partitioned into  $r$  submemories such that the shuffle operations were carried as an inherent part of the feedback of each stage. Furthermore, the feedback was completely eliminated by interchanging the role of the input and output memories [26]. Finally, a radix-4 256-point FFT processor was actually constructed, with a sampling frequency of 1.6 MHz [27]. Specifically, a fixed-point arithmetic was used.

Martinson and Smith [28] introduced a modified floating-point arithmetic, which uses fewer components than the fixed-point arithmetic with an equivalent performance level. They also suggested that a ROM may be used to implement a multiplier.

For example, a 5x5 bits multiplier with rounding to 8 bits can be implemented using 8K ROM. Liu and Peled [16] further utilized the idea of trading memories for logical gates. A new approach, which uses the technique of "bit slice" to store all possible outcomes of arithmetic operations into ROMs, was developed.

### 1.3 Objectives of The Research

The FFT algorithm has been used to compute the DFT in a number of diverse applications [29 - 31]. The hardware realizations of these processors have ranged from general purpose digital computer to dedicated larger scale integrated circuits. Recently, ROM implementations of the FFT have been considered because of their potential for a high speed parallel architecture [7].

This thesis considers some of the design problems associated with a ROM implementation of a RNS based FFT processor. Normally a ROM oriented scheme, based on table look-up methods, would not be feasible in a weighted magnitude system for any realistic dynamic range, due to the number of possible combinations. The use of the RNS allows a number to be represented with respect to a number of moduli. The operations with respect to the various moduli can be carried out independently of each other. The resulting small dynamic range makes it possible to use ROM table look-up techniques and independent parallel operations capability provides the basis for a very high speed processor. Since memory technology is evolving rapidly, the proposed realization can take immediate advantage of any decreased access times.

Since, in general, a FFT can not be implemented exactly, the specification of a desired root-mean-square (RMS) error of the frequency domain computed by the FFT processor can be used as a criterion of optimality of the design procedure. A comprehensive study of quantization error sources is required in order to provide the theoretical basis for determining how the various design parameters influence the error of the estimate.

There are quantization error problems associated with the proposed RNS implementation that have not been fully treated in the literature. The most closely related papers [32] , [33] deal primarily with the radix-2 FFT and consider only fixed-point arithmetic. This thesis presents an analysis of radix-4 FFT quantization error based on a consideration of scaling, integer conversion, dynamic range of the number system, number of stages and A/D quantization. This analysis then leads to a design procedure in which the values of all design parameters, that simplify the hardware realization and meet a specified error criterion, can be determined.

#### 1.4 Organization of The Thesis

In Chapter 2, the mathematical concepts of the RNS is introduced through a discussion of its basic properties. It is shown that the RNS is much more suitable for parallel structure than the weighted magnitude representations (such as the binary number system), because arithmetic operations are fully independent digits. The implementation of residue arithmetic using ROMs is described, and the advantages using

arrays of ROMs are also discussed. Following a brief description of the FFT, a ROM oriented FFT structure is proposed. Also, all design parameters are identified in this chapter.

As residue arithmetic is carried out in the integer number system, the magnitudes of numbers increase very rapidly after multiplication operation, and hence it is necessary to determine the radix of the FFT that minimizes the number of cascaded multiplications for a given number of samples. The optimal radix is shown to be equal to 4 in Chapter 3. Other considerations which include the number growth at each stage and different quantization error sources in the FFT processors are also discussed./

In Chapter 4, a quantization error analysis of the radix-4 FFT is developed. Errors due to A/D quantization, coefficient (twiddle factor) rounding and scaling quantization are explicitly covered. A general expression for the relative RMS output error has been derived which is a function of the parameters associated with the desired realization. Certain of the parameters have been set to practical values and a simplified design procedure has been obtained. Finally, an example is given to show how to use the design procedure.

A discussion of the theoretical and simulation results is treated in Chapter 5. This chapter also contains a description of some ramifications of the theoretically derived expressions.

In Chapter 6, the conclusions that can be obtained from the research are summarized.

## CHAPTER 2

### THE RESIDUE NUMBER SYSTEM AND ITS APPLICATION TO A FFT PROCESSOR

#### 2.1 Residue Number System Concepts

The RNS [1] is of particular interest because of the separable nature of the arithmetic. The arithmetic operations of addition and multiplication can be easily carried out using look-up tables [6]. In fact, using current technology, all the possible results of these arithmetic operations can be stored in high density ROMs.

A number in the RNS is represented by the L-tuple  $X = (x_1, x_2, \dots, x_L)$  where  $x_i = X \text{ modulo } m_i$ ; this is written  $x_i = |X|_{m_i}$ . If all the moduli,  $\{m_i\}$ , are relatively pairwise prime, the range of numbers that can be uniquely represented in residue code is equal to the product of all moduli

$$M = \prod_{i=1}^L m_i \quad (2.1)$$

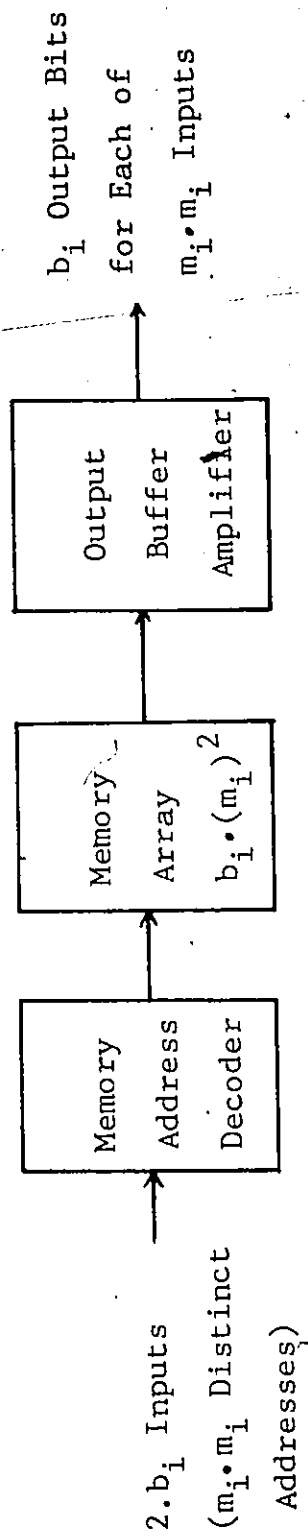
Thus, the RNS forms a ring of integers modulo  $M$ . If the RNS does not have a sufficient range of numbers to represent the result generated by arithmetic operation from a finite set of real integers, then the RNS will have overflow and the result will be a modulo  $M$  remainder of the true value.

In the RNS, binary operations of addition and multiplication, modulo  $M$ , between two numbers  $X$  and  $Y$  have the following property  $Z = |X \circ Y|_M$

$$|z|_{m_i} = z_i = |x_i \circ y_i|_{m_i} \quad (2.2)$$

where  $\circ = +$  or  $\cdot$ . In other words, the RNS does not have a carry mechanism and there is complete independence between operations on the residues. This separable nature of the arithmetic offers advantages to implementing multiplication and addition in a high speed parallel manner. Moreover, if the  $\{m_i\}$  are made small enough, then the results of operations, on all combinations of inputs, can be prestored in ROMs, and, by combining a sufficient number of rings, a viable dynamic range,  $M$ , for the number systems can be generated. This allows computations performed over a large dynamic range to be realized, independently, in systems with much smaller dynamic ranges. Figure 2.1 shows a block diagram of a ROM for a modulus  $m_i$ . In order to generate the  $m_i \cdot m_i$  possible input combinations (address),  $2 \cdot b_i$  input terminals are necessary at the memory address decoder, where  $b_i$  is the number of bits required to represent the maximum integer,  $m_i - 1$ , within the look-up tables of modulus  $m_i$ . Thus, if  $M = \prod_{i=1}^L m_i$ , then the memories required in terms of bits for the RNS is equal to  $\sum_{i=1}^L b_i \cdot m_i^2$ . The minimum memory requirements for various moduli have been discussed in [6].

The sign of numbers in the RNS is not explicitly shown. In general, the numbers in the range  $0$  to  $\frac{M}{2} - 1$  are assigned to be positive, and the numbers in the range  $\frac{M}{2}$  to  $M-1$  are assigned to be negative such that  $|x_i + y_i|_{m_i} = 0$  where

FIG. 2.1 BLOCK DIAGRAM OF A ROM FOR A MODULUS  $m_i$

$y_i = |-X|_{m_i}$  and  $X$  denotes a positive integer. Since the rules of ordinary signed arithmetic will be preserved, there is no real need for an explicit knowledge of the sign of a number during arithmetic operations. For the signed RNS, the range of positive and negative integers is reduced to approximately one-half of the total possible range of the residue representation.

The division process for residue code is complicated. If we define scaling as division by a predetermined set of constants, then this restricted operation is much easier than general division. In order to scale efficiently, the scale factor or predetermined set of constants must be chosen to be a product of some of the moduli used in the RNS [1].

There have been many previous reports of high speed FFT realizations [16], [27], [34], however, these have used the binary number system in which to perform the required arithmetic operations. This invariably leads to a proliferation of binary adders and multipliers which have to be interconnected in a pipeline arrangement, for high throughput; a job made difficult by the radically different structure of each network. In using an array of ROMs, an extremely simple structure emerges that offers identical characteristics for any required operation and is inherently simple to pipeline. As an example, consider the problem of pipelining an array designed to compute the function  $Z = (a \cdot b) + (c \cdot d)$ . Figure 2.2 shows the array for one of the moduli in the system. The only control function required is a latch pulse. The



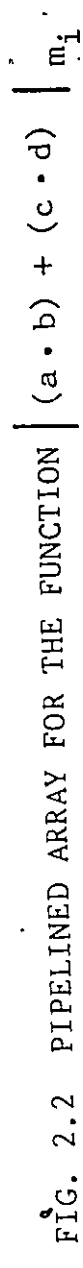


FIG. 2.2 PIPELINED ARRAY FOR THE FUNCTION  $\left| (a \cdot b) + (c \cdot d) \right|_{m_1}$

buffers are used to provide both power gain and delay. The delay allows the  $(i + 1)$ -th stage to capture data before the address lines of the  $i$ -th stage change. The system throughput rate is the inverse of the ROM access plus latch times. Conservatively 10 MHz. A further, hidden, advantage when using ROMs, is that binary operations with constants can be pre-calculated and stored in the ROM. This turns out to be important when designing scaling arrays [6], and leads to a significant hardware saving.

## 2.2 The Fast Fourier Transform

The DFT pair of the complex  $N$  point sequence  $\{x(n)\}$ , is defined as

$$X(k) = \sum_{n=0}^{N-1} x(n) W_N^{nk} \quad k = 0, 1, \dots, N-1 \quad (2.3)$$

and

$$x(n) = \frac{1}{N} \sum_{k=0}^{N-1} X(k) W_N^{-nk} \quad n = 0, 1, \dots, N-1 \quad (2.4)$$

where  $j = \sqrt{-1}$ ,  $W_N = \exp(-j2\pi/N)$ , and  $\{X(k)\}$  are complex.

The FFT is simply an efficient method for computing the DFT. In fact, when  $N$  is a composite number, the  $N$ -point DFT can be computed as a collection of smaller DFT's in conjunction with additional coefficients commonly called twiddle factors [35]. If  $N = r^m$ , where  $r$  and  $m$  are positive integers, the factors of  $N$  are equal to  $r$  and the algorithm is called a radix- $r$  algorithm. When an  $N$ -point DFT is computed using

a radix-r algorithm, a structure with  $m$  stages, where each stage includes  $N/r$  basic  $r$ -point transforms results. The basic forms of the resultant  $r$ -point transforms with DIT and DIF<sup>+</sup> are given by

$$x_{i+1}(k) = \sum_{n=0}^{r-1} x_i(n) (W_N^t)^n W_r^{nk} \quad (2.5)$$

and

$$x_{i+1}(k) = \sum_{n=0}^{r-1} x_i(n) W_r^{nk} (W_N^t)^k, \quad (2.6)$$

respectively, where  $\{x_i(n)\}$  denote the numbers at the input of the  $i$ -th stage, and  $(W_N^t)^n$ ,  $(W_N^t)^k$  are the appropriate twiddle factors.

The basic calculation of the  $r$ -point transform, as shown in equation (2.5), can be decomposed into two steps. First the  $r$ -input points are multiplied by twiddle factors. Then the  $r$ -point DFT is computed. The above procedure is reversed for the DIF algorithm. Figures 2.3(a) and (b) show the simplified representations of the  $r$ -point transform DIT and DIF algorithms, respectively.

### 2.3 A ROM Oriented FFT Structure

Since the RNS is an integer number system, all non-integer coefficients in the FFT, which include twiddle factors and non-trivial coefficients in the  $r$ -point DFT, must be converted to integers. These can be done by introducing integer conversion factors. Figure 2.4 shows the simplified integer

---

+ The DIT (or DIF) algorithms are based on the decomposition of the DFT computation by forming successively smaller subsequences of the input (or output) sequence.

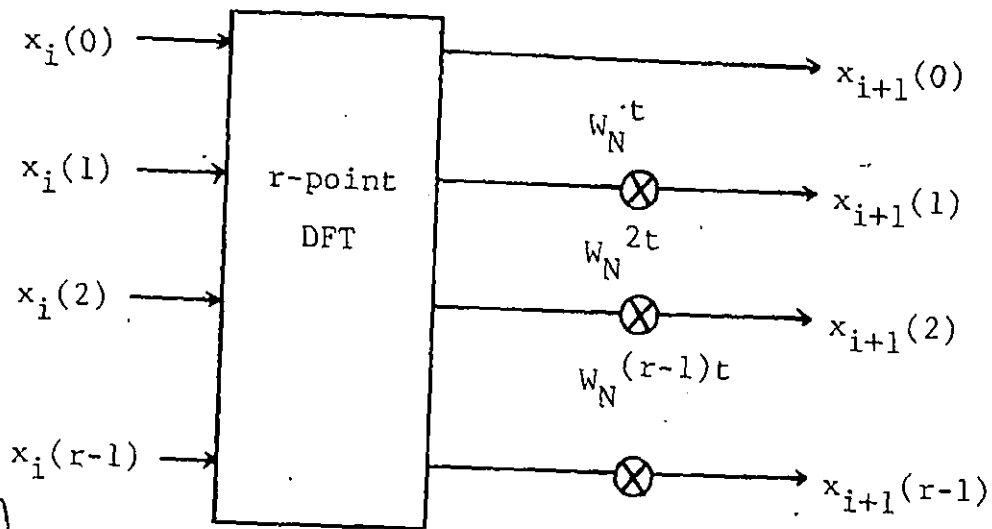
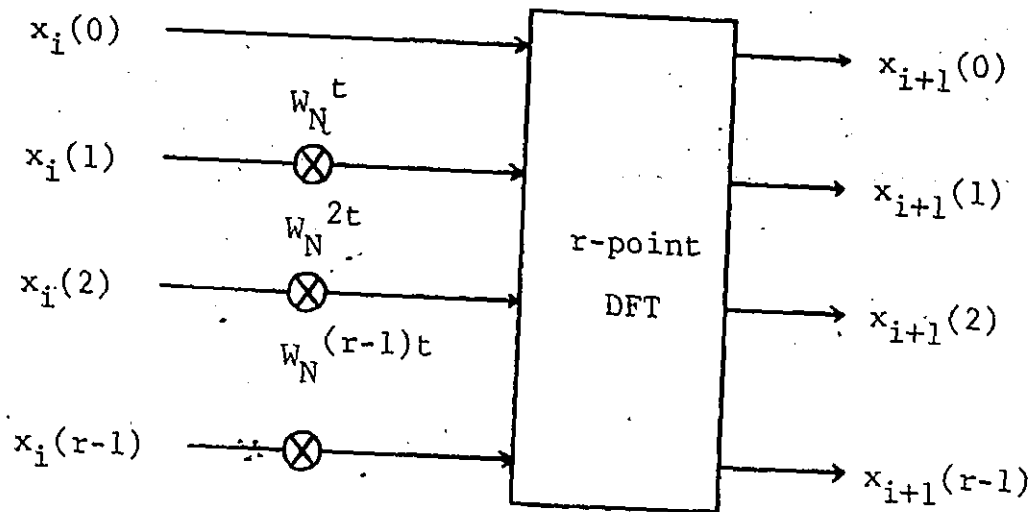


FIG. 2.3 SIMPLIFIED REPRESENTATION OF AN  $r$ -POINT TRANSFORM

(a) DIT ALGORITHM

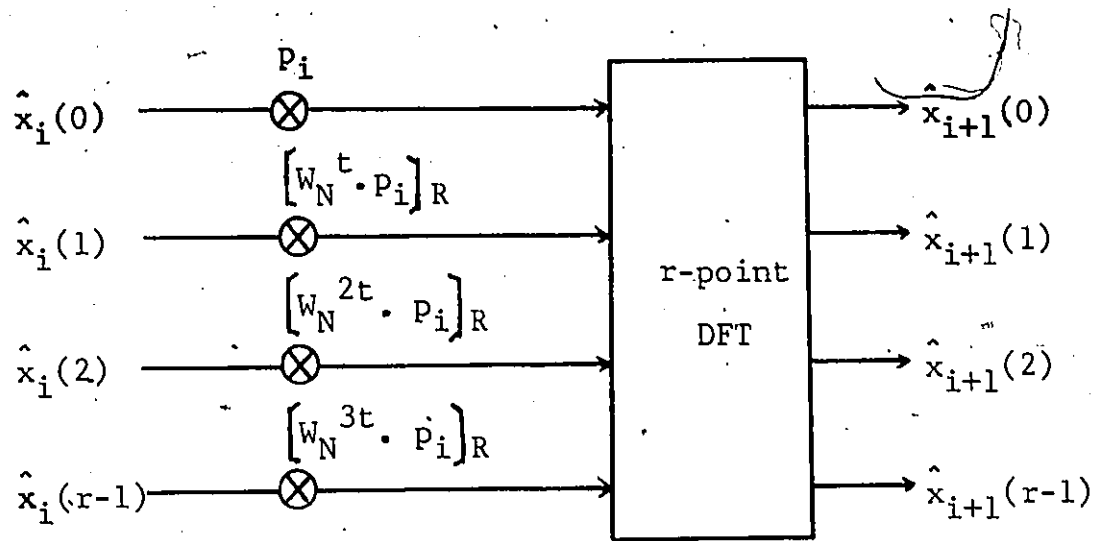
(b) DIF ALGORITHM

representations of  $r$ -point transform algorithms, where  $p_i$  is the integer conversion factor for twiddle factors at the  $i$ -th stage, which may be varied from stage to stage. It should be noted that if there are non-trivial coefficients required in the  $r$ -point DFT, then another integer conversion factor is also needed. In Figure 2.4, the superscript " $\cdot$ " denotes an integer value.

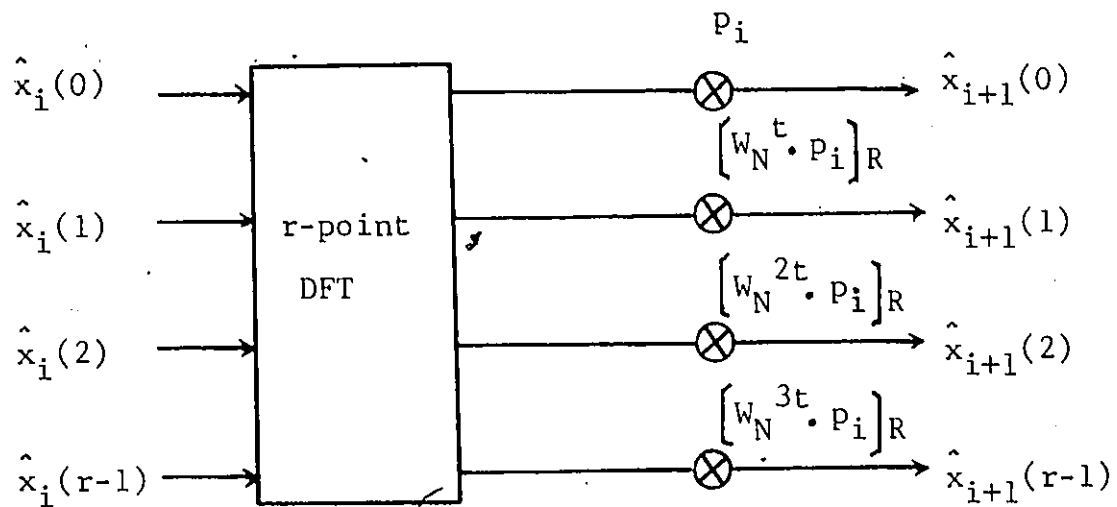
Without any loss of generality, we will assume that the binary number from the A/D converter is treated as an integer number; therefore, all the required arithmetic operations in the FFT can be computed using integer arithmetic, after introducing the integer conversion factors. In fact, the RNS concepts may be applied to the integer based FFT.

From Section 2.1, it is clear that the residue arithmetic can be efficiently implemented using table look-up ROM arrays, if the  $\{m_i\}$  are made small enough, and, by combining a sufficient number of moduli, a large dynamic range can also be generated. Thus, it is possible to have a ROM oriented FFT structure, if the number range of the RNS is sufficient for the dynamic range of the FFT. In practice, scaling operations are used to scale down the number within the limited range of the RNS.

In the RNS, scaling is difficult, because the digits do not convey any immediate information about the magnitude of the number. It is possible to scale fairly efficiently when the scale factor is a product of some of the moduli, but even in this case the hardware cost is fairly high [6]. If  $X$  is



(a)



(b)

FIG. 2.4 SIMPLIFIED INTEGER REPRESENTATION OF AN  $r$ -POINT TRANSFORM

(a) DIT ALGORITHM

(b) DIF ALGORITHM

the original number,  $K$  the scale factor, and  $Y$ , the scaled number, the scaling process can be represented by the following rational system,

$$X \approx \frac{\hat{X}}{D} \quad \text{with} \quad |\hat{X}| \leq \frac{M}{2} - 1$$

$$Y = \frac{X}{K} \approx \frac{\frac{\hat{X}}{D}}{K} \approx \frac{\left[ \frac{\hat{X}}{K} \right]_R}{D} \quad (2.7)$$

where  $\hat{X}$  is an integer with the denominator,  $D$ , to normalize its magnitude, and  $\left[ \right]_R$  denotes the integer round-off procedure. A viable scaling procedure requires that the scale factor must be pre-determined (no dynamic normalization is allowed) and scaling operations must be kept to a minimum. In order to satisfy both requirements, we will, in general, perform several exact arithmetic operations before scaling by a pre-determined constant. In this case, the dynamic range of the numerator may grow considerably after each operation. In order to preserve the rational system, it is necessary to increase the denominator to match the range growth of the numerator when cascading multiplications. Since the denominators are known a priori at every stage, one needs only to analyze the operations performed on the numerator. One can, therefore, consider that the number system is purely integer.

Figure 2.5 shows a conceptual block diagram of the proposed FFT structure with the indication of some parameters, which will be discussed in the next section. From Figure 2.5, it is seen that the analog input must be first converted

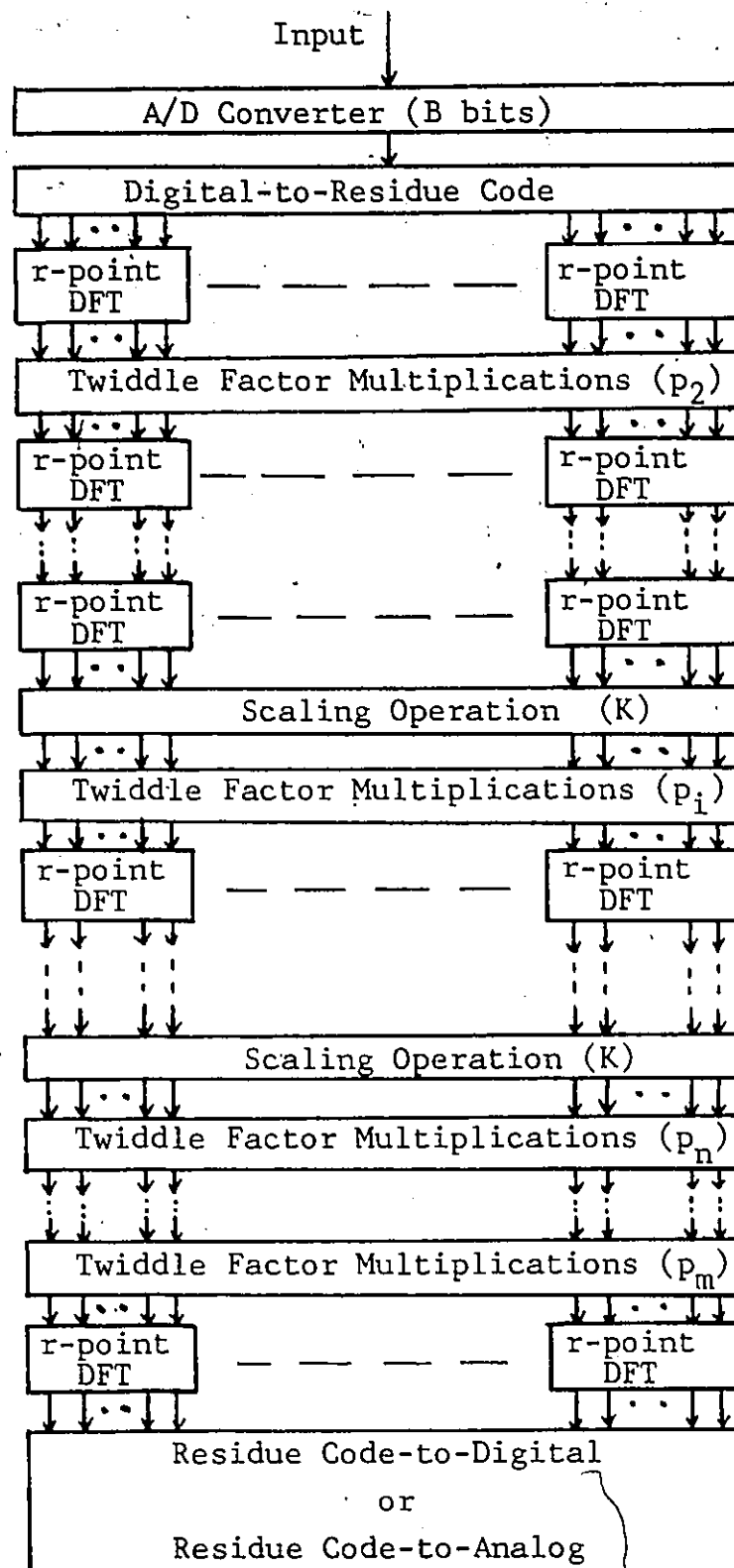


FIG. 2.5 CONCEPTUAL BLOCK DIAGRAM OF A FFT PROCESSOR



to binary digital form, and then to residue code. The number of  $r$ -point DFT's required depends on how the FFT is realized.

#### 2.4 Identification of Design Parameters

In order to construct a ROM oriented FFT processor, all the related design parameters have to be properly identified. From the proposed structure, as shown in Figure 2.4, some of design parameters are explicitly indicated. There are, however, several other parameters which can not be directly shown in the proposed structure. The importance of each parameter will be discussed in the following.

The most important parameter of the proposed structure is the number range,  $M$ , of the RNS. This parameter can also be recognized as the maximum word length for a conventional binary number implementation. In fact, the memory requirements for the ROM oriented FFT structure depend mainly on the value of  $M$ .

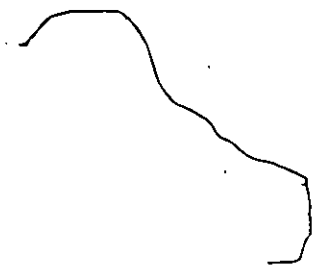
The A/D converter width in bits,  $B$ , is the number of bits used to represent the input data. To have accurate representations of analog input, a large number for  $B$  is required. Since this corresponds to a larger magnitude of the digitized input data, and the range of the RNS is limited, a compromise between the accuracy and the magnitude of the input data must be taken into consideration.

The integer conversion factor,  $p_i$ , is used to convert the non-integer coefficients into integers. Without introducing this factor, it is not possible to have an integer

based FFT. The consideration given for the A/D converter quantization bits also hold true here.

Scaling is important, since in practice we are not interested in computations modulo  $M$ , rather we require to approximate to calculations carried out in the infinite field of real (or complex) numbers. For a fixed number range, a smaller scale factor requires more frequent scaling operations. The choice of the scale factor results in different scaling distributions. In other words, we can use either larger scale factor with less scaling operations, or smaller scale factor with more scaling operations.

In addition to the above mentioned parameters, there is an inherent parameter of the FFT, which is the number growth at each stage. Once number growth has been determined, it can also be used to establish not only scale factor but also the manner in which gain in number growth is distributed through the system.



## CHAPTER 3

### FFT PROCRESSOR CONSIDERATIONS

#### 3.1 Optimal Radix Determination

Since integer based arithmetic is used to perform the required arithmetic operations in the FFT, and exact integer results are always obtained from the basic calculations of addition, subtraction and multiplication in the RNS, then the results at the output of each BCU in the FFT must be retained with full accuracy. As multiplication results are retained to full accuracy, the magnitude of numbers at subsequent stages of the FFT increase very rapidly due to the cascaded integer multiplications. In the RNS, the range of numbers that can be uniquely represented in residue code is equal to the product of all moduli, and hence all numbers must be properly scaled within the range of this number system to prevent overflows.

In terms of using integer arithmetic, we are interested in maximizing the ratio of the number of binary operations to the number of scaling operations. This is especially important when the RNS is used, as scaling operations are cumbersome to implement in hardware [6]. Since the magnitude of numbers in the FFT are increasing mainly due to multiplications, then a radix-r FFT having a minimum number of cascaded multiplications is obviously desirable from a scaling point of view.—In the FFT, the occurrence of multiplications are due to internal multiplications in the r-point DFT and

twiddle factor multiplications. When the radix of the FFT is either 2 or 4, there will be no non-trivial internal coefficients requiring multiplication operations in the  $r$ -point DFT. This results from the fact that  $W_r^{nk}$  in equations (2.5) and (2.6) equals  $0, \pm 1$ , or  $\pm j$  when  $r=2$  or  $4$ . It is thus apparent that 4 is the largest radix without internal multiplications occurring in the  $r$ -point DFT.

When the number of samples,  $N$ , is equal to a power of 2, the FFT may be realized using radices of  $2^k$ , where  $k$  is a non-negative integer. Table 2.1 shows the number of cascaded multiplications for various values of  $r$  and  $N$ . The analytic relationships are shown in the last column. Here we assume that the 8 and 16-point DFT's in the basic calculations of the radix-8 and -16 FFT's are computed in the same manner as small- $N$  WFTA's described by Silverman [15],[36]. Thus Table 2.1 has been generated on the basis that there is only one multiplication level within the  $r$ -point DFT for  $r=8$  and 16. For radices of 2 and 4, the numbers shown in Table 2.1 represent the number of cascaded twiddle factors only; while for radices of 8 and 16, the numbers represent the sum of the cascaded twiddle factors and cascaded internal multiplications in the  $r$ -point DFT's.

From Table 2.1, it can be seen that the radices of 4 and 16 have the smallest number of cascaded multiplications, and thus the use of these radices will minimize the number of scaling operations required. In general, the hardware necessary to realize the BCU of a radix-16 FFT is much more

TABLE 2.1  
NUMBER OF CASCADED NON-INTEGEE MULTIPLICATIONS FOR RADICES OF 2, 4, 8 AND 16

$\begin{array}{c} N \\ \hline r \end{array}$	64	128	256	512	1024	2048	4096	$N = r^m$
2	4	5	6	7	8	9	10	$m - 2^*$
4	2		3		4		5	$m - 1$
8	3			5			7	$(m-1) + m^{**}$
16			3				5	$(m-1) + m^{**}$

\* In the radix-2 FFT, there are two stages where the twiddle factors are trivial integers, such as 0,  $\pm 1$ ,  $\pm j$ . Otherwise, there is only one stage having these properties.

\*\* The second term,  $m$ , is the number of cascaded internal multiplications in the  $r$ -point DFT's.

complex than for a radix-4 FFT. The choice between radix 4 or 16 from a speed and cost point of view depends upon the manner in which the FFT processor is realized. However, a radix-16 realization severely limits the viability of a processor due to the number of samples,  $N$ , that can be selected. Thus radix 4 is considered as the optimal realization radix, when the number of samples,  $N$ , is a power of 2.

When the number of samples is not a power of 2, a mixed-radix FFT, prime-factor FFT or Winograd Fourier transform may be used. However, these algorithms have some disadvantages in the hardware implementations, although they may require less computation time than the radix- $r$  FFT as far as the software implementations are concerned. The main disadvantage is that the basic calculation at each stage is different, which makes the hardware very difficult to be pipelined. Furthermore, the realizations for these algorithms are not flexible, and, in fact, they must be realized in a cascade form, which is also due to the different structure of basic calculation at each stage.

Now we can conclude that radix 4 is the optimal realization radix. If, however, the number of samples is not a power of 4, then we can append enough number of zeros to the original data, and still perform a radix-4 algorithm.

### 3.2 Number Growth

In this section, some properties of the radix-4 FFT will be examined. Specifically, the mean-square value and the maximum magnitude of numbers at the output of each stage will be determined. The mean-square bound, which is

independent of input data, can be determined exactly. While the maximum magnitude can be determined both theoretically and experimentally for the worst cases only.

### 3.2.1 Mean-square Bound

In order to analyze the RMS error in the radix-4 FFT, the mean square value at the output of each stage must first be known. It has been shown in [32] that the mean square value will increase by 2 for a DIT radix-2 FFT. In the following, we will derive the mean-square bound in a more generalized manner.

Applying Parseval's theorem to equation (2.3), one obtains

$$\sum_{k=0}^{N-1} |X(k)|^2 = N \sum_{n=0}^{N-1} |x(n)|^2$$

or

$$\frac{1}{N} \sum_{k=0}^{N-1} |X(k)|^2 = N \cdot \frac{1}{N} \sum_{n=0}^{N-1} |x(n)|^2 \quad (3.1)$$

Equation (3.1) indicates that the mean-square value of the result is N times the mean-square value of the initial sequence. Since there are m similar stages for a radix-r FFT ( $N=r^m$ ), we can show that the mean square value will increase by r at each stage, as follows.

Letting  $y_i(n) = x_i(n)(W_N^t)^n$  in equation (2.5), one obtains

$$x_{i+1}(k) = \sum_{n=0}^{r-1} y_i(n) W_r^{nk} \quad (3.2)$$

Equation (3.2) can be recognized as an ~~r-point~~ DFT with  $y_i(n)$  and  $x_{i+1}(k)$  as the input and output, respectively. Applying Parseval's theorem again, one obtains

$$\sum_{k=0}^{r-1} |x_{i+1}(k)|^2 = r \sum_{n=0}^{r-1} |y_i(n)|^2 \quad (3.3)$$

Since

$$|y_i(n)|^2 = |x_i(n)|^2 |(W_N^t)^n|^2$$

and  $|(W_N^t)^n|^2 = 1$ , then equation (3.3) becomes

$$\sum_{k=0}^{r-1} |x_{i+1}(k)|^2 = r \sum_{n=0}^{r-1} |x_i(n)|^2 \quad (3.4)$$

Equation (3.4) can be generalized as

$$\frac{1}{N} \sum_{k=0}^{N-1} |x_{i+1}(k)|^2 = r \cdot \frac{1}{N} \sum_{n=0}^{N-1} |x_i(n)|^2 \quad (3.5)$$

Equation (3.5) shows that the mean-square value will increase by  $r$  at the output of each stage for a radix- $r$  DIT algorithm. This property also holds for a radix- $r$  DIF algorithm, and the proof is shown in Appendix A.



### 3.2.2 Theoretical Worst Case Upper Bound

In the proposed FFT processor the upper bound of number growth must be derived in order to be able to avoid overflow problems. Specifically, one would like to know the maximum magnitude of both the real and imaginary parts that occurs at the output of each stage.

We now consider the radix-4 DIT algorithm. When  $r=4$  in equation (2.5), one obtains

$$x_{i+1}(k) = \sum_{n=0}^3 x_i(n) (W_N^t)^n W_4^{nk} \quad (3.6)$$

From equation (3.6) one can compute a worst case upper bound for the number growth at each stage. Recognizing that

$$(W_N^t)^n = \cos \frac{2\pi nt}{N} - j \sin \frac{2\pi nt}{N} \quad (3.7)$$

one can rewrite equation (3.6) as

$$x_{i+1}(k) = \sum_{n=0}^3 \left\{ (\operatorname{Re}(x_i(n)) \cos \frac{2\pi nt}{N} + \operatorname{Im}(x_i(n)) \sin \frac{2\pi nt}{N}) + j(\operatorname{Im}(x_i(n)) \cos \frac{2\pi nt}{N} - \operatorname{Re}(x_i(n)) \sin \frac{2\pi nt}{N}) \right\} W_4^{nk} \quad (3.8)$$

where  $\operatorname{Re}(\cdot)$  and  $\operatorname{Im}(\cdot)$  denote the real and imaginary parts of the terms enclosed, respectively. Since, for a 4-point DFT, each output point (real or imaginary part) is always computed by adding or subtracting 4 input points, then, from equation (3.8), one obtains

$$\text{Max} \left\{ \left| \text{Re}(x_{i+1}(k)) \right|, \left| \text{Im}(x_{i+1}(k)) \right| \right\} \leq$$

$$\text{Max} \left\{ \left| \text{Re}(x_i(n)) \right|, \left| \text{Im}(x_i(n)) \right| \right\} \sum_{n=0}^3 \left\{ \left| \cos \frac{2\pi nt}{N} \right| + \left| \sin \frac{2\pi nt}{N} \right| \right\}$$

or

$$\frac{\text{Max} \left\{ \left| \text{Re}(x_{i+1}(k)) \right|, \left| \text{Im}(x_{i+1}(k)) \right| \right\}}{\text{Max} \left\{ \left| \text{Re}(x_i(n)) \right|, \left| \text{Im}(x_i(n)) \right| \right\}} \leq \sum_{n=0}^3 \left\{ \left| \cos \frac{2\pi nt}{N} \right| + \left| \sin \frac{2\pi nt}{N} \right| \right\} \quad (3.9)$$

Similarly, one can show (see Appendix A) that the theoretical worst case upper bound associated with the DIF algorithm is given as

$$\frac{\text{Max} \left\{ \left| \text{Re}(x_{i+1}(k)) \right|, \left| \text{Im}(x_{i+1}(k)) \right| \right\}}{\text{Max} \left\{ \left| \text{Re}(x_i(n)) \right|, \left| \text{Im}(x_i(n)) \right| \right\}} \leq 4 \left\{ \left| \cos \frac{2\pi kt}{N} \right| + \left| \sin \frac{2\pi kt}{N} \right| \right\} \quad (3.10)$$

It is seen from (3.9) and (3.10) that the theoretical worst case upper bound depends only on the magnitudes of twiddle factors which are themselves independent of the magnitude of the input sequence. It is also seen that the upper bound associated with the DIF algorithm depends only on one particular twiddle factor, while it depends on all twiddle factors within one basic 4-point transform for the DIT algorithm. The upper bounds computed according to (3.9) and (3.10) are given in Table 3.1 for  $N$  ranging from 64 to 4096. From Table 3.1 it can be seen that the upper bound

TABLE 3.1a

THEORETICAL WORST CASE UPPER BOUND OF  
THE NUMBER GROWTH AT EACH STAGE OF A RADIX-4 DIT FFT

Stage No. N	1	2	3	4	5	6
64	4	5.027	5.042			
256	4	5.027	5.042	5.058		
1024	4	5.027	5.042	5.058	5.058	
4096	4	5.027	5.042	5.058	5.058	5.058

TABLE 3.1b

THEORETICAL WORST CASE UPPER BOUND OF  
THE NUMBER GROWTH AT EACH STAGE OF A RADIX-4 DIF FFT

Stage No. N	1	2	3	4	5	6
64	5.657	5.657	4			
256	5.657	5.657	5.657	4		
1024	5.657	5.657	5.657	5.657	4	
4096	5.657	5.657	5.657	5.657	5.657	4

on number growth for a given stage is independent of the number of samples,  $N$ . In addition, for any given value  $N$ , the upper bound associated with the DIT algorithm is seen to increase as the number of stages increase until, for all practical purposes a maximum value is reached, and no worst case upper bound for number growth greater than 5.058 was seen to occur. For the case of the DIF algorithm, the upper bound at each stage is kept to a constant value, 5.567, except at the last stage where it is equal to 4. The particular value, 5.567, is, in fact, due to a twiddle factor of angle  $\pi/4$  which exists at every stage, but not at the last stage. At the last stage, the twiddle factors are equal to 1 and hence the upper bound is equal to 4.

If we compare the upper bounds between the DIT and DIF algorithms, from Table 3.1(a) and (b), we see that overall number growth is smaller for the DIT algorithm. Therefore, in the following, we will only consider the radix-4 DIT algorithm. A detailed radix-4 DIT basic calculation is shown in Figure 3.1.

Scaling factors determined from the theoretical worst case bound will definitely ensure the overflows do not occur but at the same time unrealistically constrain the dynamic range of the hardware.

### 3.2.3 Experimental Worst Case Upper Bound

Besides the theoretical upper bound developed in section 3.2.2 it is desirable to experimentally determine the number growth that is associated with commonly occurring input sequences,

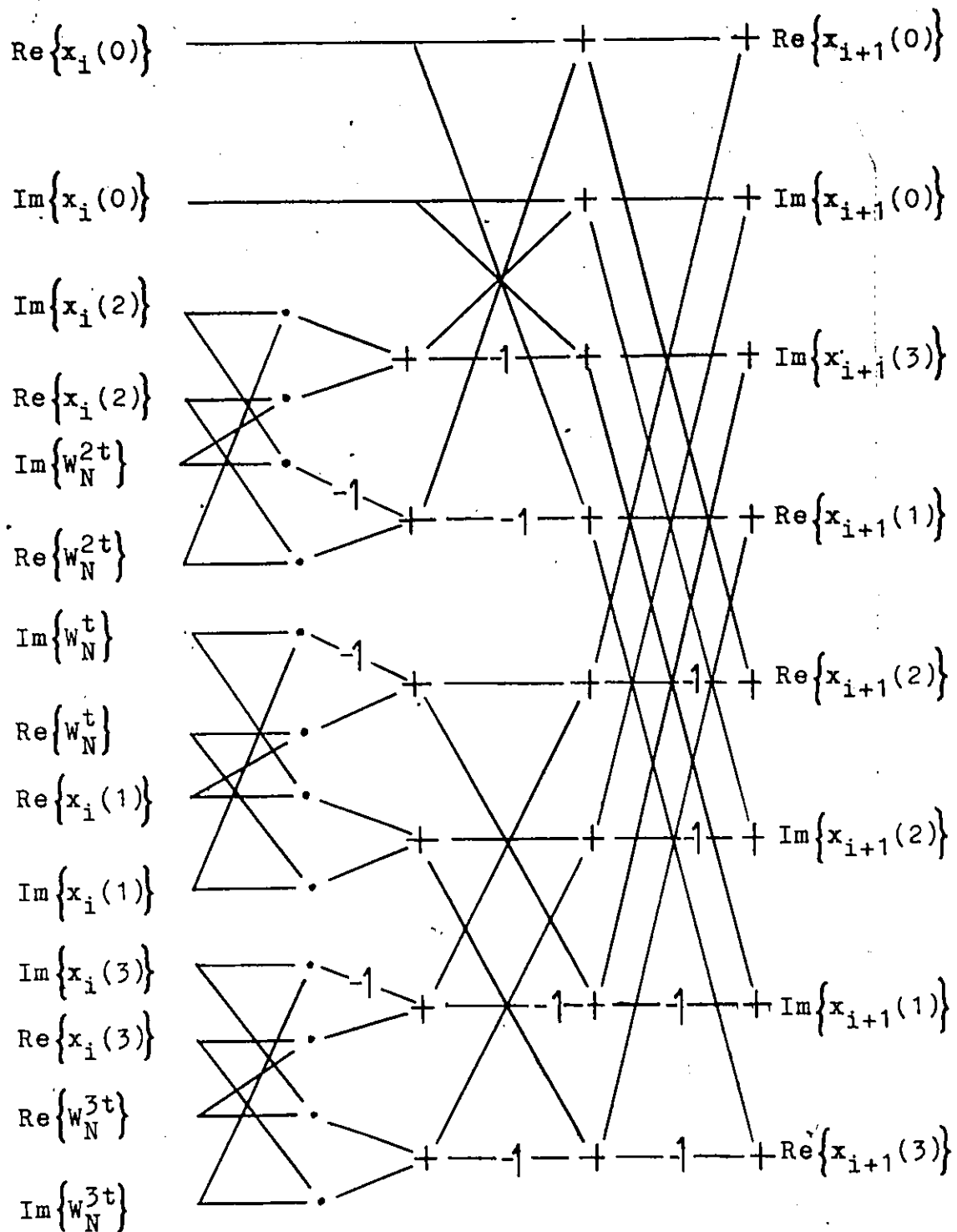


FIG. 3.1 RADIX-4 DIT BASIC CALCULATION

in order to obtain a more realistic value.

To facilitate the experimental determination of number growth, a radix-4 DIT FFT program was written and executed on a large scale general purpose digital computer. The number of samples,  $N$ , was chosen to be 1024. Three different sets of input sequences, including uniformly distributed pseudo random numbers, sine waves plus pseudo random number, and speech signals were used.

In one set of experiments, the inputs were comprised of pseudo random numbers for both the real and imaginary components. These components were uncorrelated and fell within the set  $(-1,1)$  for the first case, and  $(0,1)$  for the second case. The experimental upper bound on number growth determined for these two cases is shown in rows one and two, respectively, of Table 3.2.

In the second set of experiments two sine waves plus pseudo random numbers were used as the input sequences. The general form of the input sequence is given by

$$a(n) + 0.5 \sin \frac{n\pi}{256} + 0.25 \sin \frac{n\pi}{128} + 0.25 \sin \frac{n\pi}{64} \quad (3.11)$$

where in the first case  $\{a(n)\}$  are pseudo random numbers lying in the range  $-0.5 < a(n) < 0.5$  and in the second case pseudo random numbers lying in the range  $0 < a(n) < 1$ .

The experimental upper bound on number growth associated with these two cases are shown in rows three and four, respectively, of Table 3.2.

TABLE 3.2

SIMULATION STUDY OF WORST CASE UPPER BOUND OF  
THE NUMBER GROWTH AT EACH STAGE OF A RADIX-4 DIT FFT

Stage No. Input Type	1	2	3	4	5
1	3.97	2.93	2.72	2.74	2.62
2	3.98	3.64	3.79	3.98	4.04
3	3.17	3.04	3.35	3.78	3.95
4	3.37	2.59	3.79	3.98	4.04
5	2.66	2.95	3.32	4.01	4.00

The number growth associated with typical speech waveforms was also investigated. The corresponding upper bound on number growth is shown in the fifth row of Table 3.2.

Each of the experiments delineated in Table 3.2 were repeated sixty times. The upper bounds shown in Table 3.2 represent  $\mu + 3\sigma$ , where  $\mu$  and  $\sigma$  are the mean values and standard deviations for the samples generated, respectively.

From Table 3.2 it can be seen that an experimentally determined upper bound on number growth equal to 4 is quite reasonable for the input sequences considered.

### 3.3 Quantization Error Sources

The subject of error in the FFT has been reported in a number of papers. Error associated with the radix-2 FFT floating-point arithmetic have been analyzed in [37]. The error problems of FFTs using fixed-point arithmetic were first discussed by Welch [32], where different methods to avoid overflow were also examined. Recently, Brigham and Cecchini [33] developed a nomogram for determining fixed-point FFT system dynamic range. These papers were concerned primarily with the error characteristics of radix-2 FFT's. There are quantization problems associated with the proposed RNS implementation that have not been fully treated in the literature.

When the radix-4 FFT is implemented using the RNS, three forms of quantization errors, A/D converter quantization, coefficient rounding and scaling errors are presented.

Strictly speaking, there are several different error sources other than A/D quantization error in an A/D converter, such as



saturation and aperture errors [38], [39]. Since the input to the A/D converter can be properly adjusted, quantization error is usually considered to be the dominant one. The coefficient rounding error is due to finite precision (or integer) representations of twiddle factors. The scaling error is introduced by scaling to keep the data within the limited dynamic range.

## CHAPTER 4

### A DESIGN PROCEDURE FOR A FFT PROCESSOR

#### 4.1 Design Criteria

In the proposed FFT structure, overflow is not tolerable in the RNS. In section 2.4, all the design parameters associated with the proposed processor have been identified. Further, the number growth at each stage in the FFT has been studied. Therefore when the number range is fixed, it is always possible to determine the rest of the parameters using trial and error methods to avoid overflows. However, only

considering the overflow problem is not sufficient to obtain an efficient design. In fact, the arbitrary choice of parameters may lead to a waste of memory requirements.

It is well-known that when digital signal processing elements are implemented with hardware, errors due to finite word length always exist. In order to achieve an acceptable level of error for some chosen word length, the characteristics of these errors must be known. In this thesis, we will consider the relative RMS error at the output of the FFT processor as the design criterion.

#### 4.2 General Design Parameter Relationships

##### 4.2.1 Statistical Error Models

In this section quantization errors associated with A/D conversion, twiddle factors and scaling operations are considered. In order to determine their effects on a FFT processor, it is first necessary to establish the error models being used

to characterize each source of error.

### 1. A/D quantization error model

There are a number of different types of error that may be introduced by an A/D converter [38], [39].

Errors associated with quantization and saturation are the most common types. As the input level can always be adjusted to minimize saturation effects, only quantization error will be considered here.

If sampled data is represented by B bits, including the sign bits, and the input signal falls within the range  $\pm U$  then the converter step size, Q, is equal to

$$Q = \frac{U}{2^{B-1}} \quad (4.1)$$

It is noted that  $Q = 1$  in the integer number system. When the quantization noise is assumed to be uniformly distributed with zero mean value, the mean and variance of the converter quantization error have been shown [39] to be

$$\overline{e_Q} = 0, \quad \overline{e_Q^2} = \frac{1}{12} Q^2 = \frac{1}{12} \quad (4.2)$$

The statistical assumptions have been considered adequate to represent quantization error, even though correlation effects have been neglected, because for practical values of B the magnitude of the quantization error introduced by the converter, when compared with twiddle factor and scaling errors, is small.

## 2. Twiddle factor error model

In the proposed realization of the FFT processor, the error associated with the integer representation of the twiddle factor must be considered. An integer conversion factor,  $p_i$ , must be introduced as shown in Figure 2.4.

In the ROM oriented implementation envisaged, the integer conversion factor,  $p_i$ , for the twiddle factors, must be pre-determined and the integer representations of the twiddle factors stored in ROMs. The error associated with the integer conversion of a twiddle factor,  $c$ , is defined as

$$\begin{aligned}
 e_I &= \text{Re}(e_I) + j \text{Im}(e_I) \\
 &= p_i c - \left[ p_i c \right]_R \\
 &= (p_i \text{Re}(c) - \left[ p_i \text{Re}(c) \right]_R) + j (p_i \text{Im}(c) - \left[ p_i \text{Im}(c) \right]_R)
 \end{aligned}
 \tag{4.3}$$

where  $\left[ \right]_R$  denotes the integer round-off procedure. The round-off procedure is such that the errors  $\text{Re}(e_I)$  and  $\text{Im}(e_I)$  are uniformly distributed in the range  $(-0.5, 0.5)$  and thus

$$\overline{e_I} = 0$$

and

$$\overline{|e_I|^2} = \overline{(\text{Re}(e_I))^2} + \overline{(\text{Im}(e_I))^2} = \frac{1}{6} \tag{4.4}$$

### 3. Scaling error model

In the ROM oriented implementation considered, the magnitude of the numbers that occur at each stage must be known. The study of number growth given in section 3.2.3 allows one to choose a scale factor capable of constraining the number growth to a desired value.

The errors associated with a scaling operation are either round-off or truncation errors depending upon the nature of the hardware. There are two different scaling algorithms that have been developed for the look-up table approach, the original algorithm and the estimate algorithm [6]. The errors due to the latter depend upon the specific moduli. Throughout this thesis, we will assume that the former is used.<sup>+</sup> The output from this scaling procedure is

$$Y = \left\lfloor \frac{X}{K} \right\rfloor_R \quad (4.5)$$

where  $X$  is the input,  $K$  is the scale factor.

In order to define scaling error, let  $V$  be the complex integer to be scaled and  $K$  be the scale factor, where  $K$  is a positive integer. The scaling error,  $e_S$ , is given by

$$\begin{aligned} e_S &= \text{Re}(e_S) + j \text{Im}(e_S) = \frac{V}{K} - \left\lfloor \frac{V}{K} \right\rfloor_R \\ &= \left\{ \frac{\text{Re}(V)}{K} - \left\lfloor \frac{\text{Re}(V)}{K} \right\rfloor_R \right\} + j \left\{ \frac{\text{Im}(V)}{K} - \left\lfloor \frac{\text{Im}(V)}{K} \right\rfloor_R \right\} \quad (4.6) \end{aligned}$$

---

<sup>+</sup> The scaling algorithm generates a zero mean error by the addition of one half the scale factor to the input.

The same round-off procedure is used here as we used in equation (4.3), hence,  $\text{Re}(e_S)$  and  $\text{Im}(e_S)$  have the same probability distribution as  $\text{Re}(e_I)$  or  $\text{Im}(e_I)$  and the mean and variance are given by

$$\overline{e_S} = 0 \text{ and } \overline{|e_S|^2} = \frac{1}{6}, \quad (4.7)$$

respectively.

#### 4.2.2 RMS Quantization Error Analysis

In this section an expression for the relative RMS error,  $\alpha_1$ , is derived. If the true value of the DFT is given by the complex sequence  $\{X(k)\}$  and the sequence generated by the FFT processor is  $\{\hat{X}(k)\}$  then the relative RMS error,  $\alpha_1$ , is given by.

$$\alpha_1 = \left\{ \frac{\sum_k [\text{Re}(X(k)) - \text{Re}(\hat{X}(k))]^2 + [\text{Im}(X(k)) - \text{Im}(\hat{X}(k))]^2}{\sum_k (\text{Re}(X(k)))^2 + (\text{Im}(X(k)))^2} \right\}^{1/2}. \quad (4.8)$$

The RMS error,  $\alpha_1$ , shall now be developed in terms of converter quantization error, twiddle factor error and scaling error. In this analysis, it is assumed that these three sources of error are uncorrelated.

In the following analysis it is necessary to delineate

a number of variables with special care. A typical stage in the radix-4 processor consists of  $N/4$  basic modules that have the form shown in Figure 4.1. Each basic module has twiddle factors and a 4-point DFT associated with it. The variables associated with the analysis of the basic module are also included in Figure 4.1. A superscript indicates the stage number while the subscript indicates the appropriate input or output. The symbols  $\{x_{IN}^i(n)\}$  and  $\{x_{DFT}^i(n)\}$  are used to represent the true values of the complex sequence that exist at the input to the twiddle factor multipliers and to the 4-point DFT of the  $i$ -th stage of the FFT processor, respectively. Whereas,  $\{\hat{x}_{IN}^i(n)\}$  and  $\{\hat{x}_{DFT}^i(n)\}$  are the scaled integer representations of the corresponding computed value. The error between two points is given by

$$E_{IN}^i(n) = x_{IN}^i(n) - \theta \hat{x}_{IN}^i(n) \quad (4.9)$$

or

$$E_{DFT}^i(n) = x_{DFT}^i(n) - \frac{\theta}{P_i} \hat{x}_{DFT}^i(n) \quad (4.10)$$

where  $\theta$  is a factor that takes into account the integer conversions and scaling operations.

#### 1. Converter quantization error

The mean-square value of the input to the first stage is defined as

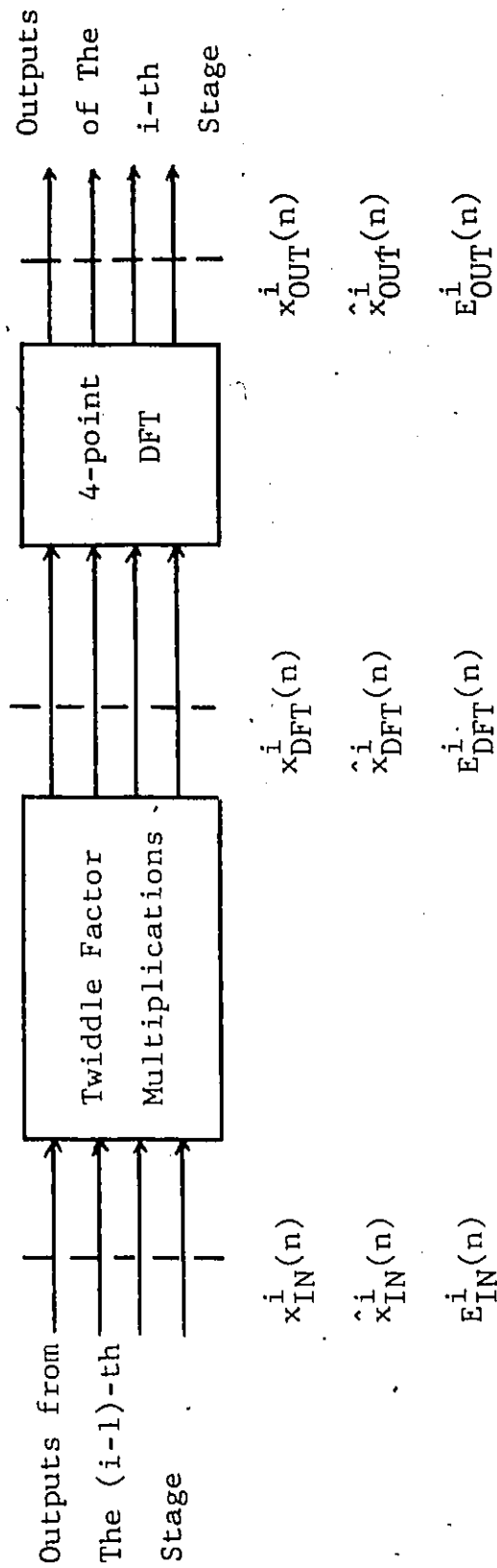


FIG. 4.1 BASIC MODULE OF THE i-th STAGE OF A RADIX-4 FFT PROCESSOR



$$\sigma^2 = \overline{|x_{IN}^1(n)|^2} \quad (4.11)$$

The error present at the input to the first stage are due to A/D converter quantization errors and thus from equations (4.2) and (4.9) one has

$$\overline{E_{IN}^1(n)} = 0 \text{ and } \overline{|E_{IN}^1(n)|^2} = 2 \overline{e_Q^2} = \frac{1}{6}, \quad (4.12)$$

since  $\theta$  equals 1 at the input to the first stage.

In the first stage of a radix-4 DIT FFT, the twiddle factors are equal to 1 and hence the integer conversion factor,  $p_1$ , is also equal to 1. Thus no multiplication operations are associated with the first stage, and one has

$$E_{DFT}^1(n) = E_{IN}^1(n) \quad (4.13)$$

There are, however, four input points associated with each complex output point. Therefore, the error associated with the output of the first stage is given by

$$\overline{E_{OUT}^1(n)} = \overline{E_{DFT}^1(n)} = \overline{E_{IN}^1(n)} = 0 \quad (4.14)$$

and

$$\overline{|E_{OUT}^1(n)|^2} = 4 \overline{|E_{DFT}^1(n)|^2} = 4 \overline{|E_{IN}^1(n)|^2} = 4 \cdot \frac{1}{6} \quad (4.15)$$

The A/D converter quantization error introduced into the input of the first stage continues to propagate through the remaining stages.

## 2. Twiddle factor error

For the second and subsequent stages the twiddle factors are no longer trivial and the errors introduced by integer conversions must be considered. Let  $c$  denote the true value of a complex twiddle factor in the second stage. From equation (4.10), one obtains

$$\begin{aligned} E_{\text{DFT}}^2(n) &= x_{\text{DFT}}^2(n) - \frac{\theta}{p_2} \cdot \hat{x}_{\text{DFT}}^2(n) \\ &= x_{\text{IN}}^2(n) \cdot c - \frac{1}{p_2} \hat{x}_{\text{IN}}^2(n) \cdot [p_2 c]_R \end{aligned} \quad (4.16)$$

where  $\theta = \frac{1}{p_1} = 1$ . Since  $E_{\text{OUT}}^1(n) = x_{\text{OUT}}^1(n) - \hat{x}_{\text{OUT}}^1(n) = x_{\text{IN}}^2(n) - \hat{x}_{\text{IN}}^2(n)$ , one can rewrite equation (4.16) as

$$E_{\text{DFT}}^2(n) = x_{\text{OUT}}^1(n) \cdot c - \frac{1}{p_2} (x_{\text{OUT}}^1(n) - E_{\text{OUT}}^1(n)) \cdot [p_2 c]_R \quad (4.17)$$

Using equation (4.3) one can write equation (4.17) as

$$E_{\text{DFT}}^2(n) = -E_{\text{OUT}}^1(n) \left( \frac{e_I}{p_2} - c \right) + x_{\text{OUT}}^1(n) \left( \frac{e_I}{p_2} \right) \quad (4.18)$$

Thus from equation (4.18), by noting that  $|c|^2 = 1$  and  $\bar{e}_I = 0$ , one obtains

$$\overline{E_{\text{DFT}}^2(n)} = 0 \quad (4.19)$$

and

$$\begin{aligned}
 \overline{|E_{DFT}^2(n)|^2} &= \overline{|E_{OUT}^1(n)|^2} \cdot \left( \frac{\overline{|e_I|^2}}{P_2} + 1 \right) + \overline{|x_{OUT}^1(n)|^2} \cdot \frac{\overline{|e_I|^2}}{P_2} \\
 &\approx \overline{|E_{OUT}^1(n)|^2} + \overline{|x_{OUT}^1(n)|^2} \cdot \frac{\overline{|e_I|^2}}{P_2} \\
 &= 4 \cdot 2 \overline{e_Q^2} + 4 \cdot \overline{|x_{IN}^1(n)|^2} \cdot \frac{1}{6} \cdot \frac{1}{P_2} \\
 &= 4 \cdot \frac{1}{6} + \frac{2}{3} \frac{1}{P_2} \sigma^2 \quad (4.20)
 \end{aligned}$$

However, since only 3 integer conversions associated with the twiddle factors contribute round-off error as shown in Figure 2.4, equation (4.20) can be written as

$$\begin{aligned}
 \overline{|E_{DFT}^2(n)|^2} &= 4 \cdot \frac{1}{6} + \frac{3}{4} \cdot \frac{2}{3} \frac{\sigma^2}{P_2} \\
 &= 4 \cdot \frac{1}{6} + \frac{1}{2} \frac{\sigma^2}{P_2} \quad (4.21)
 \end{aligned}$$

To generate the output of the second stage a 4-point DFT must be computed and thus one obtains

$$\overline{|E_{OUT}^2(n)|^2} = 0 \quad (4.22)$$

and

$$\overline{|E_{OUT}^2(n)|^2} = 4 \cdot \overline{|E_{DFT}^2(n)|^2} = 4^2 \cdot \frac{1}{6} + 2 \frac{\sigma^2}{P_2} \quad (4.23)$$

The operations associated with the third stage are similar to those associated with the second stage, hence

$$\overline{|E_{OUT}^3(n)|^2} = 4^3 \cdot \frac{1}{6} + 4 \cdot 2 \left( \frac{1}{P_2} + \frac{1}{P_3} \right) \sigma^2 \quad (4.24)$$

This development can be generalized such that the errors, due to converter quantization and to integer conversion, at the output of the k-th stage are given by

$$\overline{|E_{OUT}^k(n)|^2} = \begin{cases} 4 \cdot \frac{1}{6}, & k = 1 \\ 4^k \cdot \frac{1}{6} + 4^{k-2} \cdot 2 \left( \sum_{i=2}^k \frac{1}{P_i} \right) \sigma^2, & k \geq 2 \end{cases} \quad (4.25)$$

### 3. Scaling error

In order to prevent overflow due to number growth it is necessary to introduce a scale factor K. Due to hardware constraints associated with the realization of an RNS scaling algorithm [6] it is desirable to use only one scale factor and vary the number of stages between scaling operations.

Let  $k_i$  be the number of stages between the (i-1) and i-th scaling operation. Then

$$m = \sum_{i=1}^q k_i \quad (4.26)$$

where  $q$  is the number of scaling operations required to produce a scaled output at the  $m$ -th stage of the processor. In many applications full output precision rather than a scaled output is desired at the output of the final stage and hence a total of  $q-1$  scaling operations are performed. These scaling schemes are also shown in Figure 4.2.

The error due to the first scaling operation can be computed as

$$E_{s_1}(n) = x_{OUT}^{k_1}(n) - \frac{K}{\prod_{i=1}^{k_1} P_i} \cdot \left[ \frac{x_{OUT}^{k_1}(n)}{K} \right]_R \quad (4.27)$$

Substituting equation (4.6) into equation (4.27) and using the relationship between  $x_{OUT}^{k_1}(n)$  and  $\hat{x}_{OUT}^{k_1}(n)$ , one obtains

$$E_{s_1}(n) = E_{OUT}^{k_1}(n) - \frac{K e_s}{\prod_{i=1}^{k_1} P_i} \quad (4.28)$$

Then

$$\overline{|E_{s_1}(n)|^2} = \overline{|E_{OUT}^{k_1}(n)|^2} + \frac{K^2}{\prod_{i=1}^{k_1} P_i^2} \cdot \frac{1}{6} \quad (4.29)$$

In the above equation, the first term is the mean-square error generated before the first scaling operation, as given in equation (4.25), while the second term is the mean-square error due to the first scaling operation.

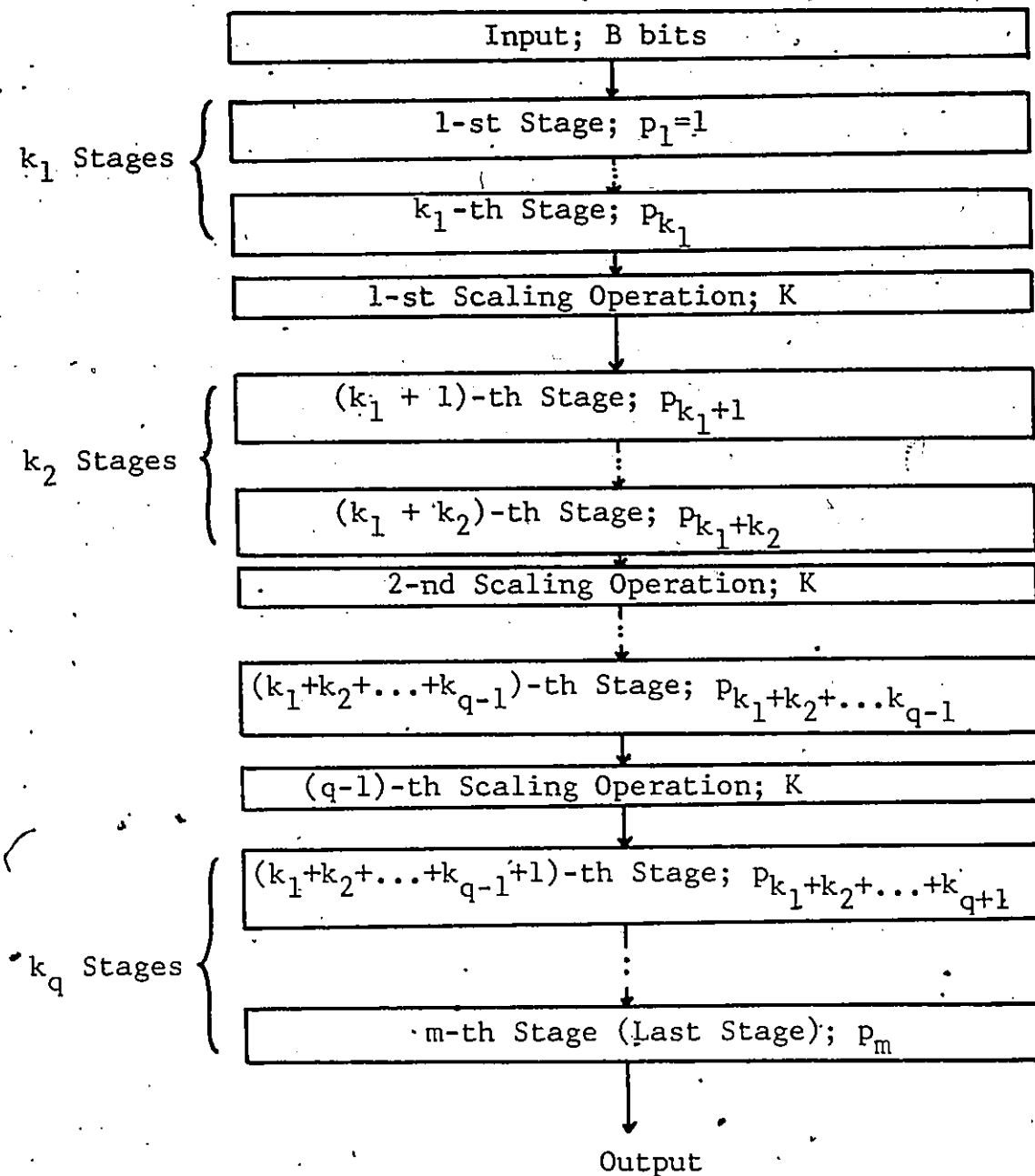


FIG. 4.2 SCALING SCHEMES OF THE FFT PROCESSOR

At the output of the  $(k_1 + k_2)$ -th stage (just before the second scaling operation), the first term in equation (4.29) will be propagated as discussed in the previous section, and the second term will grow by a factor of  $4^{k_2}$ . Thus, the error becomes

$$\begin{aligned} \overline{|E_{OUT}^{k_1+k_2}(n)|^2} &= 4^{k_1+k_2} \cdot \frac{1}{6} + 4^{k_1+k_2-2} \cdot 2 \left\{ \sum_{i=2}^{k_1+k_2} \frac{1}{P_i^2} \right\} \\ &\quad + 4^{k_2} \cdot \frac{K^2}{\prod_{i=1}^{k_1} P_i^2} \cdot \frac{1}{6} \end{aligned} \quad (4.30)$$

This development can be extended to give an expression for the mean square error at the output of the final stage due to converter quantization, integer conversion and scaling with the form

$$\begin{aligned} \overline{|E_{OUT}^m(n)|^2} &= 4^m \cdot \frac{1}{6} + 4^{m-2} \cdot 2 \cdot \left\{ \sum_{i=2}^m \frac{1}{P_i^2} \right\} \sigma^2 \\ &\quad + \frac{K^{2q}}{6} \sum_{i=1}^{q-1} \left\{ 4^{\sum_{h=1}^i k_{q-h+1}} \cdot K^{-2i} \cdot \frac{\sum_{t=1}^{q-1} k_t}{\prod_{h=1}^{q-1} P_h^2} \right\} \end{aligned} \quad (4.31)$$

Note that  $p_1=1$  in equation (4.31).

Then using equations (4.31), (4.11) and (3.1), one can obtain the total RMS relative error,

$$\alpha_1 = \frac{\text{RMS (error)}}{\text{RMS (true result)}} = \left\{ \frac{|E_{\text{OUT}}^m(n)|^2}{4^m \sigma^2} \right\}^{1/2}$$

$$= \left\{ \frac{1}{6 \sigma^2} + \frac{1}{8} \sum_{i=2}^m p_i^{-2} + \frac{K^{2q}}{6 \cdot 4^m \sigma^2} \cdot \sum_{i=1}^{q-1} \left[ 4^{\sum_{h=1}^i k_{q-h+1}} \right. \right. \\ \left. \left. K^{-2i \sum_{t=1}^{q-i} k_t} p_h^{-2} \right] \right\}^{1/2} \quad (4.32)$$

The relative RMS error,  $\alpha_1$ , expressed in equation (4.32) is seen to be a function of :

- (a)  $\sigma^2$ , the mean-square value of the input to the first stage,
- (b)  $m$ , the total number of stages in the FFT processor,
- (c)  $p_i$ , the integer conversion factor associated with the twiddle factors in the  $i$ -th stage,
- (d)  $q$ , the number of scaling operations required to produce a scaled output from the last stage of the processor,
- (e)  $K$ , the scale factor used to prevent overflow due to number growth (assumed constant for all scaling operations), and
- (f)  $k_i$ , the number of stages between scaling operations.



### 4.3 A Simplified Design Procedure

In the design of an FFT processor the range of the number system being used is first determined. Then the upper bound on number growth associated with each stage, for typical input sequences is established. The rest of the parameters  $p_i$ ,  $q$ ,  $k_i$  and  $K$  can then be computed.

While equation (4.32) is a general expression, it does not explicitly introduce the range of the integer number system being used or the upper bound on number growth analyzed in section 3.2. In addition, if equation (4.32) is to provide the basis for a design procedure it is convenient to set some parameters to practical values and make certain simplifying assumptions.

In this section equation (4.32) will be put in a form that is less general but that leads to more design-oriented results. The following simplifications will be made.

First, the input signal,  $x_{IN}^1(n)$ , is assumed to be a uniformly distributed complex sequence with zero mean values for both the real and imaginary parts. Then, if the accuracy of the A/D converter is given as  $B$  bits the mean-square value shown in equation (4.11) becomes

$$\sigma^2 = \frac{1}{6} 2^{2B} \quad (4.33)$$

Secondly, from Table 3.2, a practical upper bound on number growth at each stage will be set equal to 4. As a third simplifying assumption the integer conversion factor,  $p_i$ ,

will be set equal to  $P$  for  $i = 2, 3, \dots, m$  and  $p_1 = 1$ . In other words, all twiddle factors are represented with equal accuracy.

On the basis of the last two simplifications it is reasonable to restrict the number of possible scaling schemes to ones in which the number of stages between scaling operations is constant. This simplification still allows the number of stages before the first scaling operation, and after  $q-1$  scaling operations, to be variable. With these simplifications it is convenient to consider two general classes of scaling schemes.

Let the first class of scaling schemes be given by

$$k_1 \leq m, k_2 = k_3 = \dots = k_q = k \quad (4.34)$$

and the second class by

$$k_1 < m, k_2 = k_3 = \dots = k_{q-1} = k$$

and

$$1 \leq k_q < k \quad (4.35)$$

Also, on the basis of the last two assumptions and the fact that the scale factor,  $K$ , is a constant for all scaling operations, one can then set the scale factor equal to

$$K = (4P)^k \quad (4.36)$$

If  $M$  represents the range of the RNS allowed in the FFT processor, by noting that  $2^{B-1}$  is the maximum magnitude of the input along with a consideration of number growth, it follows that

$$\frac{M}{2} = 4 \cdot (4P)^{k_1-1} \cdot 2^{B-1}$$

or

$$M = 4^{k_1} \cdot P^{k_1-1} \cdot 2^B \quad (4.37)$$

Equation (4.37) can also be rewritten as

$$B = \log_2 \left( \frac{M}{4^{k_1} P^{k_1-1}} \right) \quad (4.38)$$

such that  $B$  is expressed as a function of the value of  $M$  and  $P$  for the specified scaling scheme.

In order to simplify the notation let the term,  $\frac{1}{6\sigma^2}$ , in equation (4.32) be represented by  $\alpha_Q^2$ . Similarly, let  $\frac{1}{8} \sum_{i=2}^m P_i^{-2} = \alpha_I^2$  and the last term equal  $\alpha_S^2$ . Thus

$$\alpha_1^2 = \alpha_Q^2 + \alpha_I^2 + \alpha_S^2 \quad (4.39)$$

These three terms will now be expressed in terms of the

appropriate parameters.

Using equation (4.33) the relative error due to A/D converter quantization is given by

$$\alpha_Q^2 = \frac{1}{2^{2B}} \quad (4.40)$$

In fact, the above result can be directly obtained by knowing the error present at the input to the first stage. Dividing  $|E_{IN}^1(n)|^2$  in equation (4.12) by  $\sigma^2$ , one obtains

$$\begin{aligned} \frac{|E_{IN}^1(n)|^2}{\sigma^2} &= \frac{1}{6\sigma^2} \\ &= \frac{1}{2^{2B}} \end{aligned} \quad (4.41)$$

Equations (4.40) and (4.41) indicate that the output relative error due to the error present in the input is equal to the input relative error.

Using equation (4.37) one can write equation (4.40) as

$$\alpha_Q^2 = 4^{2k_1} M^{-2} P^{2(k_1-1)} \quad (4.42)$$

Since  $p_i = P$ ,  $i \geq 2$  one can write the relative error due to integer conversion as

$$\alpha_I^2 = \frac{m-1}{8} \frac{1}{P^2} \quad (4.43)$$

Using equations (4.34), (4.35) and (4.37),  $\alpha_S^2$  can be written as

$$\alpha_S^2 = A \cdot 4^{2k+k_1} \cdot M^{-2} P^{2k} \quad (4.44)$$

where

$$A = \begin{cases} \frac{4^{\frac{m-k_1}{4} - 1}}{4^k - 1}, & k_q = k \\ \frac{4^{\frac{m-k_1-k_q+k}{4} - 1}}{4^k - 1}, & k_q < k \end{cases} \quad (4.45)$$

takes into account the two general classes of scaling schemes.

Combining equations (4.42), (4.43) and (4.44), one can write equation (4.39) as

$$\alpha_1^2 = 4^{2k_1} M^{-2} P^{2(k_1-1)} + \frac{m-1}{8} \frac{1}{P^2} + A \cdot 4^{2k+k_1} \cdot M^{-2} P^{2k} \quad (4.46)$$

Thus the relative mean-square error,  $\alpha_1$ , is expressed as a function of the maximum magnitude,  $M$ , the integer conversion constant,  $P$ , and the scaling schemes as given by  $k_1$ ,  $k$  and  $k_q$ .

By differentiating equation (4.46) with respect to  $P$ , and equating the results to zero one obtains

$$2(k_1 - 1) \cdot 4^{2k_1} \cdot M^{-2} P^{2k_1-3} - \frac{m-1}{4} P^{-3} + 2k \cdot A \cdot 4^{2k+k_1} \cdot M^{-2} P^{2k-1} = 0 \quad (4.47)$$

Equation (4.47) can be used to determine the value of  $P$ , expressed as a function of  $M$ , that minimizes the relative error,  $\alpha_1$ , for a given scaling scheme. The results can not be put in a closed form expression that is practically useful. However, for a given value of  $N$ , a given desired level of relative error and a given specified scaling scheme, it is possible to tabulate the interrelationships that define the corresponding value of  $M$  and  $P$ . A knowledge of  $M$  and  $P$  allow one to compute the corresponding value of the scale factor,  $K$ , and the required A/D converter accuracy,  $B$ . For  $N = 1024$  and various scaling schemes these relationships have been tabulated in Table 4.1.

The results developed in this section can be used as the basis of a design procedure for the proposed FFT processor. We will assume that  $N = 1024$ . Now, the design procedure is given as follows:

- (1) Specify the level of relative error,  $\alpha_1$ , that is acceptable for the processor.
- (2) Select a desired scaling scheme.
- (3) Compute the required range of the number system,  $M$ , using the relationship given in Table 4.1.
- (4) Select a set of relatively prime moduli which gives a product of all moduli,  $M_0$ , close to the computed  $M$  from (3).
- (5) Compute the integer conversion constant,  $P$ , using the relationship given in Table 4.1.

TABLE 4.1  
OPTIMUM FUNCTIONAL RELATIONSHIPS BETWEEN  
VARIOUS PARAMETERS OF A RADIX-4 DIT FFT; N=1024

Scaling Schemes	$\alpha_1$	P	Remarks
$k_1=5$	$11.01 M^{-1/5}$	$0.203 M^{1/5}$	$\alpha_S^2=0, \alpha_Q^2=0.25 \alpha_I^2$
$k_1=4$	$4.085 M^{-1/4}$	$0.2 M^{1/4}$	$\alpha_S^2 \gg \alpha_Q^2, \alpha_S^2=0.33 \alpha_I^2$
$k_1=3, k=1$	$4.365 M^{-1/3}$	$0.198 M^{1/3}$	$\alpha_Q^2 \gg \alpha_S^2, \alpha_Q^2=0.5 \alpha_I^2$
$k_1=3, k=2$	$3.804 M^{-1/3}$	$0.152 M^{1/3}$	$\alpha_Q^2 + \alpha_S^2 = 0.5 \alpha_I^2$
$k_1=2, k=1$	$10.18 M^{-1/2}$	$0.0982 M^{1/2}$	$\alpha_S^2 \gg \alpha_Q^2, \alpha_S^2 = \alpha_I^2$
$k_1=2, k=2, k_q=1$	$5.714 M^{-1/3}$	$0.0491 M^{1/3}$	$\alpha_S^2 \gg \alpha_Q^2, \alpha_S^2=0.5 \alpha_I^2$
$k_1=2, k=3$	$4.085 M^{-1/4}$	$0.2 M^{1/4}$	$\alpha_S^2 \gg \alpha_Q^2, \alpha_S^2=0.33 \alpha_I^2$
$k_1=1, k=1$	$10.21 M^{-1/2}$	$0.0979 M^{1/2}$	$\alpha_S^2 \gg \alpha_Q^2, \alpha_S^2 = \alpha_I^2$
$k_1=1, k=2$	$5.554 M^{-1/3}$	$0.156 M^{1/3}$	$\alpha_S^2 \gg \alpha_Q^2, \alpha_S^2=0.5 \alpha_I^2$
$k_1=1, k=3, k_q=1$	$4.628 M^{-1/4}$	$0.141 M^{1/4}$	$\alpha_S^2 \gg \alpha_Q^2, \alpha_S^2=0.33 \alpha_I^2$
$k_1=1, k=4$	$3.388 M^{-1/5}$	$0.233 M^{1/5}$	$\alpha_S^2 \gg \alpha_Q^2, \alpha_S^2=0.25 \alpha_I^2$

- (6) From the chosen set of moduli in (4), select some of them to obtain a scale factor,  $K_0$ , close to  $(4P)^k$ . The integer conversion constant for the stages after the first scaling operation is equal to

$$\left[ \frac{(K_0)^{1/k}}{4} \right]_R.$$

- (7) Substitute  $M_0$  and  $P$  into equation (4.38) and solve for  $B$ . Round  $B$  to an integer,  $B_0$ . Substitute  $M_0$  and  $B_0$  into the same equation and solve for  $P$ . The integer conversion constant for the stages before the first scaling operation is equal to  $[P]_R$ .

For a specified level of relative error, it is always possible to select a set of relatively prime moduli such that their product is close to the required value of  $M$ .

However, this may not be true for the selection of the scale factor, because the scale factor must be determined from the fixed chosen moduli. As a general comment, it is better to choose some smaller moduli to form the moduli set such that it has more flexibility to obtain the desired value of the scale factor.

#### 4.4 Other Considerations

##### 4.4.1 Optimal Scaling scheme

Since the scaling scheme must be specified to use Table 4.1 it is desirable to determine which scaling scheme will generate the smallest relative error for a given value of  $M$ .



Figure 4.3 shows plots of number range versus relative RMS error for various scaling schemes. The values plotted were computed using the functional relationships between  $M$  and

$\alpha_1$  for the various scaling schemes shown in Table 4.1.

In Figure 4.3 the lowest curve, representing scaling schemes  $k_1 = 1, k = 1$  and  $k_1 = 2, k = 1$ , indicates the smallest value of error for a given number system range. It is desirable to choose the scaling schemes  $k_1 = 2, k = 1$  as the better of the two candidates since this scheme requires one less scaling operation and also requires less A/D converter accuracy for a given error.

When the FFT is implemented with hardware, one always wants to minimize the cost for a given acceptable performance, such as output error. This consideration can be stated as: if the errors of different schemes are the same, which one will require the least number of look-up ROM tables to implement? The answer to the question is not straightforward. In fact, the total number of look-up tables required for implementation will depend on how the FFT is realized. As mentioned in section 1.2, there are five different realizations. We will only consider the first three realizations, since they represent a medium cost/speed trade off.

The sequential processor requires one BCU and 8 (4x2) scaling arrays. Thus no extra scaling arrays are required, even if there are two or more scaling operations. Therefore, the scaling scheme with smaller  $M$  and smaller  $K$  will be our choice for the hardware realization. Figure 4.4 shows the

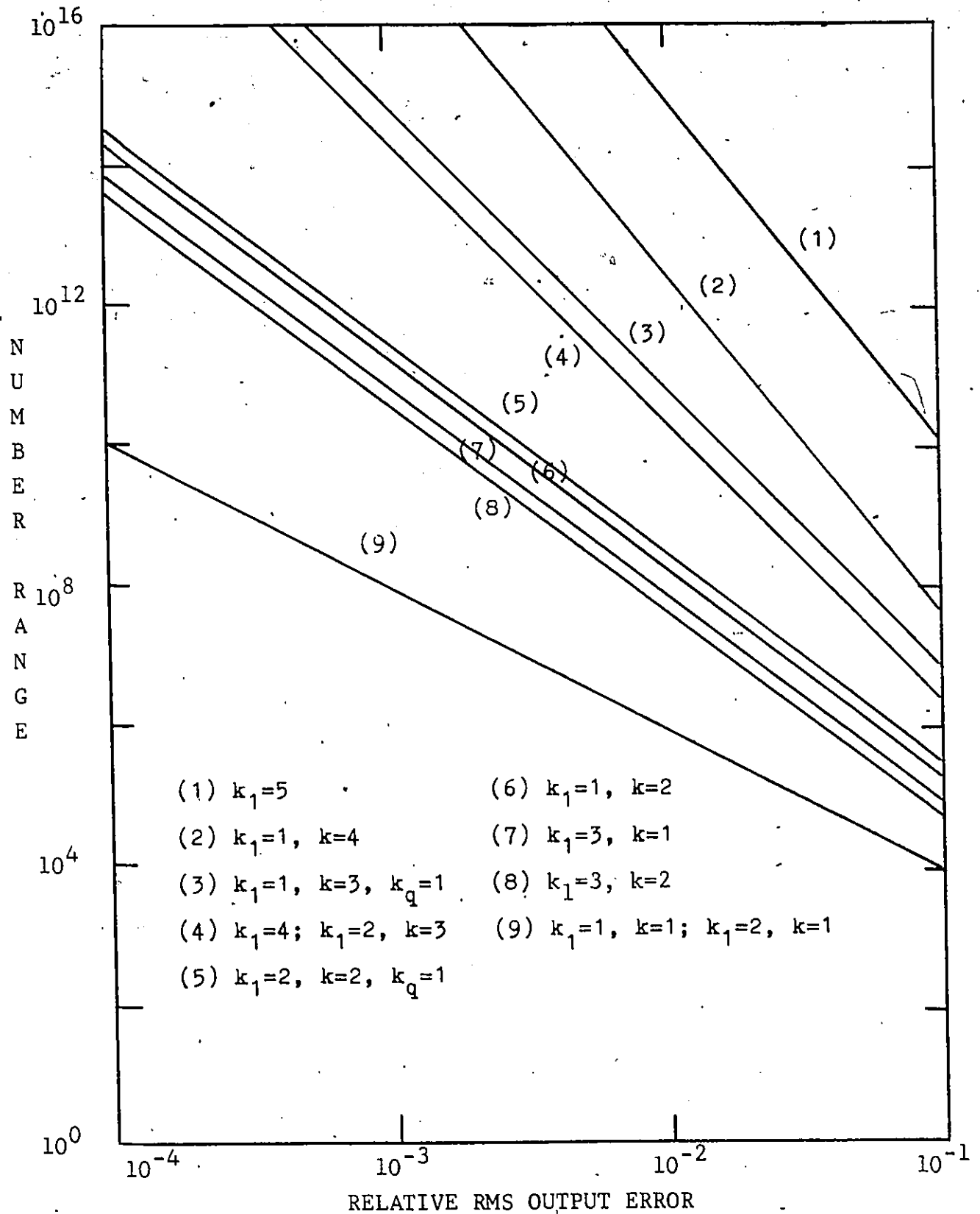


FIG. 4.3 RELATIVE RMS OUTPUT ERROR VS NUMBER RANGE FOR VARIOUS SCALING SCHEMES;  $N=1024$

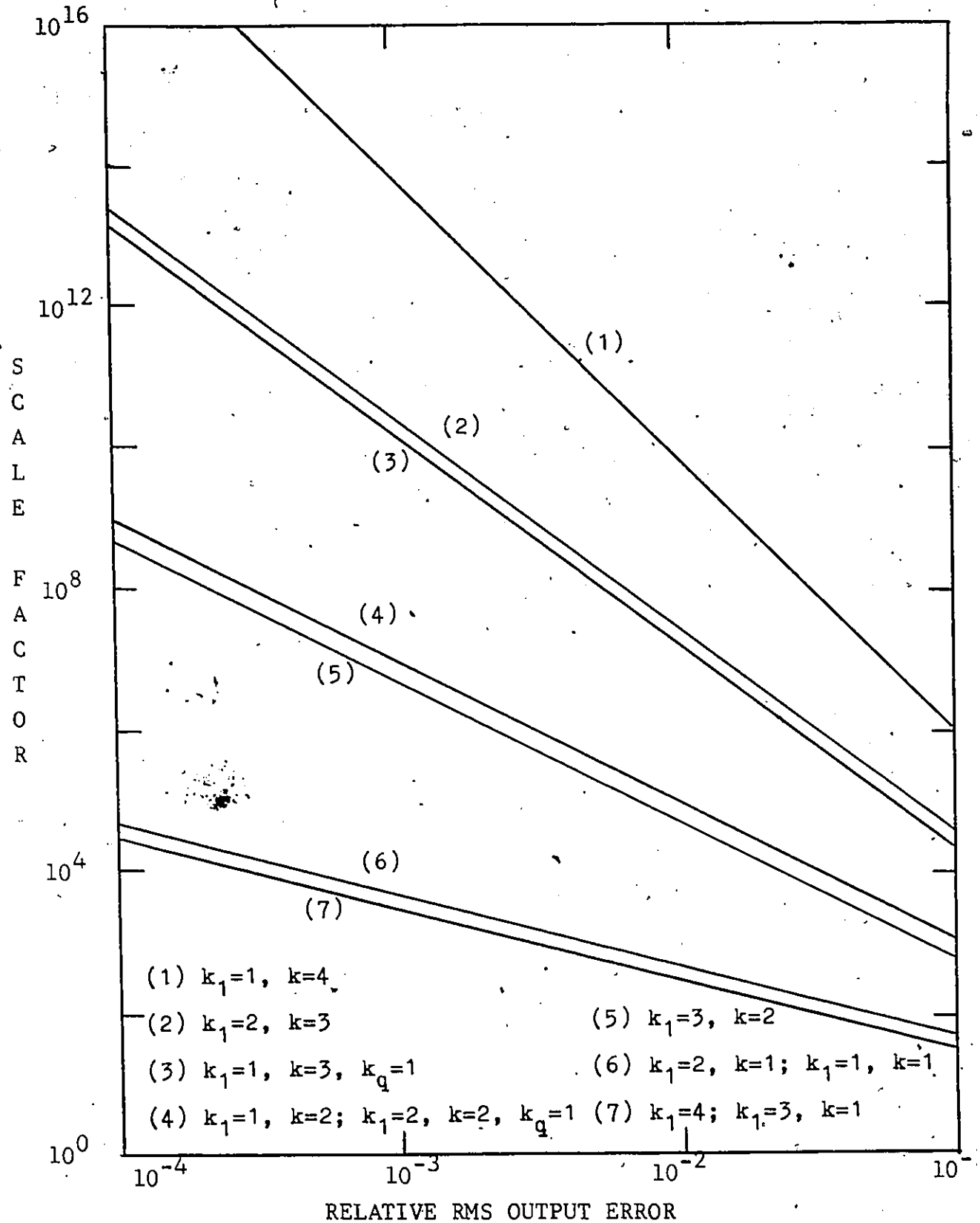


FIG. 4.4 RELATIVE RMS OUTPUT ERROR VERSUS SCALE FACTOR FOR VARIOUS SCALING SCHEMES;  $N=1024$

plots of scaling factor versus relative RMS error. The values plotted were computed using the relationships between  $M$ ,  $P$ ,  $K$  and  $\alpha_1$ . From Figure 4.4, it is seen that the scaling schemes  $k_1 = 4$  and  $k_1 = 3$ ,  $k = 1$  use the smallest  $K$  for a given error, but they require very large  $M$  as shown in Figure 4.3. Since the total number of look-up tables required depends mainly on the value of  $M$ , we then, again, consider that the two scaling schemes,  $k_1 = 1$ ,  $k = 1$  and  $k_1 = 2$ ,  $k = 1$  are better candidates. Based on the previously mentioned advantages for the scheme  $k_1 = 2$ ,  $k = 1$  as comparing to the scheme  $k_1 = 1$ ,  $k = 1$ , the best scheme for sequential realization is still  $k_1 = 2$ ,  $k = 1$ .

Since the total number of look-up tables required for the Quad-4 processor is just four times those in the sequential processor, the best scheme for the Quad-4 processor is  $k_1 = 2$ ,  $k = 1$ .

For the case of cascade realization, it requires  $m$  BCUs and  $8(q-1)$  scaling arrays, where  $q-1$  is the number of scaling operations for  $m$  stages. From Figure 3.1, it is seen that there are 34 operations involved in a basic calculation. In addition, when the FFT is implemented using the integer number system, it is necessary to multiply the first two points,  $\text{Re}(x_i(0))$  and  $\text{Im}(x_i(0))$ , by the integer conversion factor,  $p_i$ , as shown in Figure 2.4. Thus, there are total 36 operations required in the BCU. Since each operation can be implemented using  $L$  look-up tables for an  $L$  moduli number system, then the number of look-up tables required for a radix-4 BCU

is  $36L$ .<sup>+</sup> When the number of scaling moduli is  $S$ , it was shown in [6] that the number of look-up tables for a scaling array is  $(L-1)(S+\frac{L}{2}) - S^2$ . Therefore the number of look-up tables required for cascade realization is

$$T = 16L + (m-1) \cdot 36L + 8(q-1)((L-1) \cdot (S+\frac{L}{2}) - S^2) \quad (4.48)$$

where the first term takes the consideration that the twiddle factors at the first stage are equal to one. It is noted that the memory required for input/output conversion, temporary storage, and coefficient storage is not considered. From Figure 4.3 and Figure 4.4 four schemes with smaller  $M$  and smaller  $K$  are chosen to compare the values of  $T$ , when

$\alpha_1 = 0.01$ . These four schemes are shown in Table 4.2. Also in this table are shown the moduli used for the number system, the scaling factor, and the value of  $T$ . For the results in Table 4.2, the first scheme,  $k_1 = 2$ ,  $k = 4$ , requires the least number of look-up tables.

The total look-up table requirements shown in Table 4.2 appear to be quite large. The typical data rates we are anticipating, however, are of such a magnitude that the number of ROMs is quite justified and, in fact, represents a fairly cost effective solution. Since we are operating an independent radix-4 butterfly in hardware at each stage in the FFT array,

---

<sup>+</sup> If all the  $p_i$ 's are equal, then the multiplication by  $p_i$  can be precalculated in the look-up tables. Therefore, only  $34L$  look-up tables are required.

TABLE 4.2

COMPARISON OF THE TOTAL NUMBER OF LOOK-UP TABLES REQUIRED FOR  
THE CASCADE REALIZATIONS FOR FOUR DIFFERENT SCALING SCHEME;

$$N=1024, \alpha_1=0.01$$

Scheme No.	$k_1$	$k$	$q$	M	K	T
1	2	1	4	$32 \times 31 \times 13 \times 11 \times 7 = 9.93 \times 10^5$	$31 \times 13 = 4.03 \times 10^2$	1116
2	3	1	3	$32 \times 31 \times 29 \times 27 \times 13 \times 11 = 1.11 \times 10^8$	$29 \times 11 = 3.19 \times 10^2$	1296
3	3	2	2	$32 \times 31 \times 29 \times 27 \times 11 \times 7 = 5.98 \times 10^7$	$31 \times 27 \times 11 \times 7 = 6.44 \times 10^4$	1344
4	4	1	2	$32 \times 31 \times 29 \times 25 \times 17 \times 11 \times 7 = 2.54 \times 10^{10}$	$29 \times 11 = 3.19 \times 10^2$	1592

and we are assuming that the results of each ROM are latched to allow pipelining, then the data rate is given by  $8/\tau_L$ , where  $\tau_L$  is the access time of each ROM plus output data latch time. A conservative estimate for  $\tau_L$ , using current hardware, is 100 nS. We should, therefore, be able to produce data rates through the array of at least 80 MHz.

Based upon the above considerations, we feel that the scaling scheme,  $k_1=2$ ,  $k=1$  may be the best choice for both error minimization and minimum cost of hardware realization.

The choice between the selections of different realizations will, however, depends on the designer's demand. For completeness, Table 4.3 gives the optimum functional relationships between  $\alpha_1$ , M and P for various N from 64 to 4096, when the scheme  $k_1=2$ ,  $k=1$  is used.

#### 4.4.2 Scale Factor Considerations

As we mentioned in section 4.3, it is usually difficult to have the chosen scale factor close to the derived value. If, however, the chosen value of the scale factor deviates from the derived value by a very large amount, then the use of the corresponding values of the integer conversion constant and the A/D quantization bit, according to the design procedure given in section 4.3, no longer minimizes the relative error,  $\alpha_1$ . In this section, we will modified the third simplifying assumption made in section 4.3 such that we have more flexibility to determine the values of some parameters.

From equation (4.38), it can be seen that the required A/D quantization bits, B, is related to the values of M and

TABLE 4.3  
 OPTIMUM FUNCTIONAL RELATIONSHIPS BETWEEN  
 $\alpha_1$ , M AND P FOR VARIOUS N;  $k_1=2$ ,  $k=1$

N	$\alpha_1$	P
64	$4.0 M^{-\frac{1}{2}}$	$0.1487 M^{\frac{1}{2}}$
256	$9.118 M^{-\frac{1}{2}}$	$0.1250 M^{\frac{1}{2}}$
1024	$10.18 M^{-\frac{1}{2}}$	$0.0982 M^{\frac{1}{2}}$
4096	$15.27 M^{-\frac{1}{2}}$	$0.0732 M^{\frac{1}{2}}$



the integer conversion factors at the stages before the first scaling operation,  $p_i$  for  $i \leq k_1$ . It is also known that the scale factor depends only on the integer conversion factors at the stages between scaling operations. Thus by letting the values of  $p_i$ ,  $i \leq k_1$  be variable and not equal to  $P$ , we can decouple the relationship between  $p_i$ ,  $i \leq k_1$  (or  $B$ ) and  $p_i$ ,  $i > k_1$  (or  $K$ ). By doing so, even if the derived value of  $K$  can not be obtained, the errors, due to the A/D quantization and the integer conversion before the first scaling operation, can still be minimized by choosing proper values of  $p_i$ ,  $i \leq k_1$  and  $B$ , which are now independent of the value of the  $K$ . In other words, a suboptimum structure is easily obtained.

From the previous analysis, we concluded that the best scheme is  $k_1 = 2$ ,  $k = 1$ , which generated the smallest error for a fixed value of  $M$ . In this section we will assume that this best scheme is used. For this scheme there are two stages before the first scaling operation, and hence the integer conversion factor at the second stage,  $p_2$ , will be treated as a variable, and the rest of the integer conversion factors is set equal to a constant  $P$ , i.e.,  $p_3 = p_4 = \dots = p_m = P$ . Recall that  $p_1 = 1$ .

Since  $p_2 \neq P$ , equations (4.42) and (4.43) should be written as

$$\alpha_Q^2 = \frac{4^4 p_2^2}{M^2} \quad (4.49)$$

and

$$\alpha_I^2 = \frac{1}{8p_2^2} + \frac{m-2}{8P^2} \quad (4.50)$$

Substituting  $k_1 = 2$  and  $k = 1$  into equation (4.44), one obtains

$$\alpha_S^2 = \frac{4^4 (4^{m-2} - 1) P^2}{3M^2} \quad (4.51)$$

Combining equations (4.49), (4.50) and (4.51), equation (4.39) becomes

$$\alpha_1^2 = \frac{4^4 p_2^2}{M^2} + \frac{1}{8p_2^2} + \frac{m-2}{8P^2} + \frac{4^4 (4^{m-2} - 1) P^2}{3M^2} \quad (4.52)$$

From equation (4.52), it is interesting to see that the first two terms depend on  $p_2$ , while the last two terms depend on  $P$ . Thus, when  $M$  is fixed,  $\alpha_1$  can be minimized with respect to  $p_2$  and  $P$  separately. The corresponding values of  $p_2$  and  $P$  are given as follows for  $N$  equal to 1024,

$$p_2 = 0.1486 M^{\frac{1}{2}} \quad (4.53)$$

$$P = 0.0913 M^{\frac{1}{2}} \quad (4.54)$$

Also, the minimum value of  $\alpha_1$  is

$$\alpha_1 = 10.06 M^{-\frac{1}{2}} \quad (4.55)$$

As a comparison, Table 4.4 shows the correspondingly derived values of some parameters by minimizing the relative error,  $\alpha_1$ , for two different cases,  $p_2=P$  and  $p_2 \neq P$ . From Table 4.4 it is seen that the second case requires smaller A/D quantization bits,  $B$ , and larger integer conversion factor at the second stage,  $p_2$ , than the first case.

It is also seen that the second case produces slightly smaller error than the first case for a given value of  $M$ .

In fact, a most important advantage for the modified second case is that the relative error,  $\alpha_1$ , is minimized with respect to  $p_2$  and  $P$  separately. Thus, the error with respect to  $p_2$  is still minimized even if the derived value of  $P$  or  $K$  can not be obtained exactly.

In addition, it is best to have an expression to show the effects of the variation of the value of the scale factor,  $K$ , on the output error. Let  $\alpha_K^2$  denote the last two terms in equation (4.52)

$$\alpha_K^2 = \frac{m-2}{8p^2} + \frac{4^4 (4^{m-2} - 1) P^2}{3M^2} \quad (4.56)$$

Substituting  $P = \frac{K}{4}$  into equation (4.56), one obtains

$$\alpha_K^2 = \frac{2(m-2)}{K^2} + \frac{4^4 (4^{m-2} - 1) K^2}{3M^2} \quad (4.57)$$

When  $M$  and  $m$  are fixed, equation (4.57) can be used to minimize the value of  $\alpha_K$  by choosing a proper value of  $K$ . As an example, Figure 4.5 shows the plot of  $\alpha_K$  versus  $K$  for  $M=10^5$  and  $m=5$ .

TABLE 4.4  
COMPARISON OF OPTIMUM FUNCTIONAL RELATIONSHIPS OF  
VARIOUS PARAMETERS BETWEEN TWO DIFFERENT CASES,

$P_2 = P$  AND  $P_2 \neq P$ ;  $k_1 = 2$ ,  $k = 1$  AND  $N = 1024$

Integer Conversion Factor at The Second Stage	$\alpha_1$	$P_2$	P	B
$P_2 = P$	10.18 M $^{-\frac{1}{2}}$	0.0982 M $^{-\frac{1}{2}}$	0.0982 M $^{-\frac{1}{2}}$	0.5 $\log_2 M - 0.6519$
$P_2 \neq P$	10.06 M $^{-\frac{1}{2}}$	0.1486 M $^{-\frac{1}{2}}$	0.0913 M $^{-\frac{1}{2}}$	0.5 $\log_2 M - 1.2495$

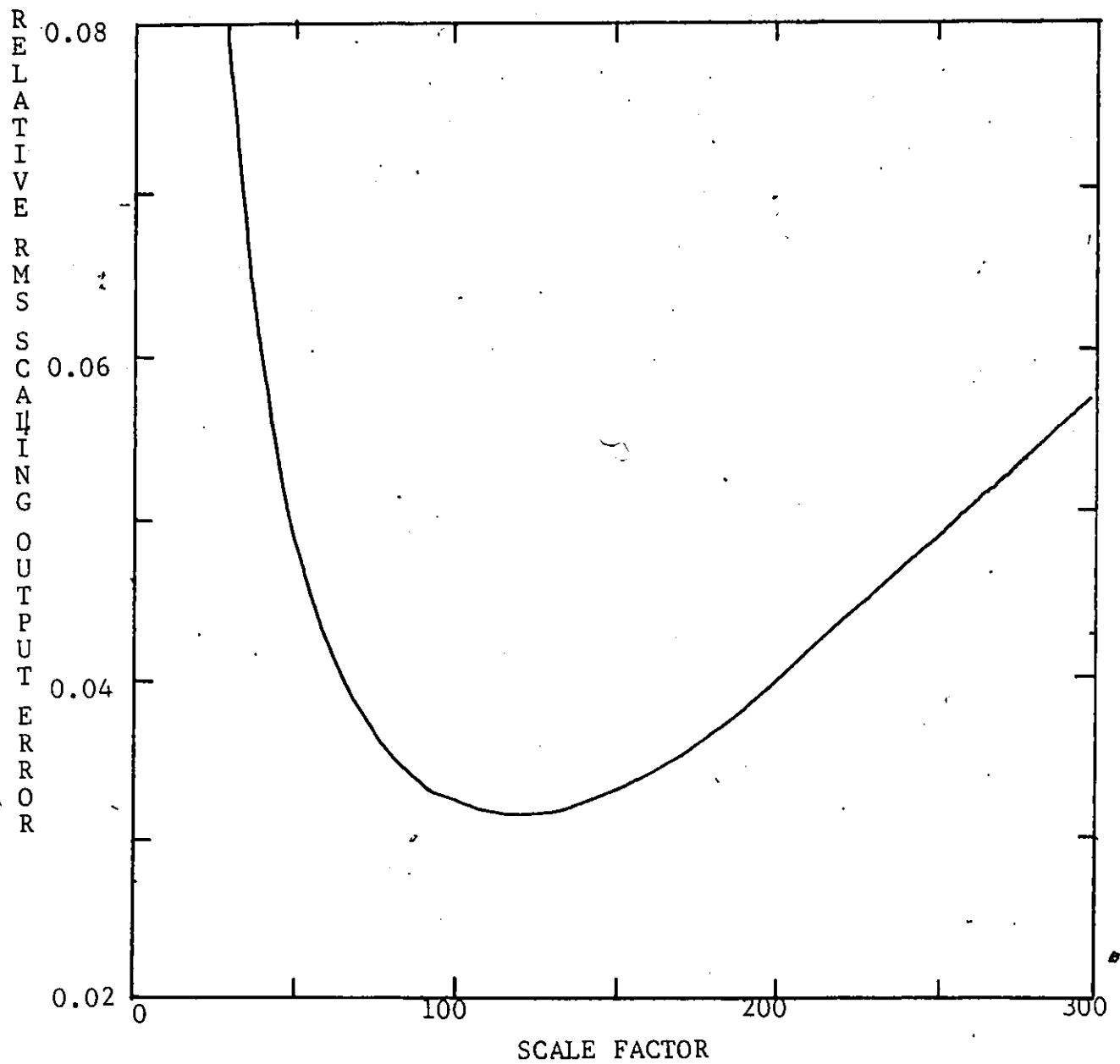


FIG. 4.5 RELATIVE RMS SCALING OUTPUT ERROR VERSUS SCALE FACTOR;  
 $M=10^5$ ,  $N=1024$

From Figure 4.5, it can be seen that if the desired value of  $K$  can not be obtained, one may choose a value of  $K$  either larger or smaller than the desired value to minimize the error,  $\alpha_K$ . For a practical realization, one would like to use the number of look-up tables as small as possible to implement the scaling array. However, the relationship between the magnitude of the scale factor and the required number of look-up tables is not trivial, and, in fact, the number of look-up tables required for the scaling array depends on both the total number of moduli used in the system and the number of moduli used for the scale factor [6]. Moreover, the error,  $\alpha_K$ , will be affected by changing the derived value of  $K$  to the closest suitable value. Thus, a trade off between the error,  $\alpha_K$ , and the required number of look-up tables with respect to a chosen scale factor must be taken into consideration.

The design procedure for this case is similar to the previous one given in section 4.3, but the values of  $M$  and  $P$  are computed using equations (4.54) and (4.55), respectively. In step (6), the plot of  $\alpha_K$  versus  $K$  for the chosen number range,  $M_0$ , must first be drawn, and based on the plot we choose a suitable scale factor to compromise the above mentioned trade off.

#### 4.4.3 Sequential Realization Considerations

The sequential realization is the simplest and slowest one, but its cost is the lowest. If speed is not very important for some applications, then the sequential realization

may be a good choice for its simplicity and lower cost.

In section 4.4.1, we considered that the scheme  $k_1=2$ ,  $k=1$ , is better than the scheme  $k_1=1$ ,  $k=1$  for the sequential realization, because the latter requires larger A/D converter quantization bits and also one more scaling operation. In this section, we will compare the two schemes from a different point of view.

Since only one calculation unit is used to perform the required arithmetic operations in the sequential realization, then, for efficient realization, it is best to have an identical structure for each stage in the FFT. When the scheme  $k_1=2$ ,  $k=1$ , is used, there are two stages before the first scaling operation and then scaling after each following stage. Thus, the structure of the first stage is different from the other stages, and special circuitry is required to control this scheme. Instead, when the scheme  $k_1=1$ ,  $k=1$ , is used, there is a scaling operation after each stage. In other words, the structure of each stage, including scaling operation, is identical. This indicates that the scheme  $k_1=1$ ,  $k=1$  is more suitable than the scheme  $k_1=2$ ,  $k=1$  for the sequential realization, as far as simple structure is concerned. In addition, we can also modify this scheme such the requirement of a large number of A/D converter quantization bits can be eliminated. This modification will be discussed in the following.

Since the twiddle factors in the first stage of the DIT algorithm are equal to one, then, in order to have a scaling operation after the first stage, the magnitude of input to

the first stage must be very large. Thus, a very large number of A/D converter quantization bits is usually required for the scaling scheme  $k_1=1$ ,  $k=1$ . This shortcoming can be eliminated by introducing some dummy constant integers in the first stage such that all input points to the first stage are pre-multiplied by a constant integer,  $G$ , and then 4-point DFTs are computed. Specifically, the constant integers,  $G$ , are stored in the real parts and zeros are stored in the imaginary parts, which form the "artificial" twiddle factors as shown in Figure 4.6. It is noted that no error will be introduced after the multiplication by an integer.

The determination of a desired constant integer,  $G$ , to minimize the output error,  $\alpha_1$ , can be seen in Appendix B. Here some relationships between  $G$ ,  $P$ ,  $M$  and  $\alpha_1$  are given in the following

$$G = p_2 = p_3 = p_4 = p_5 = P = 0.0979 M^{\frac{1}{2}}$$

and

(4.58)

$$\alpha_1 = 10.21 M^{-\frac{1}{2}}$$

If we compare the relationships given in equation (4.58) to those given in Table 4.1 for the same scaling scheme  $k_1=1$ ,  $k=1$ , we see that they are exactly the same. However, the required number of A/D converter quantization bits are different for the two cases, and are given by:



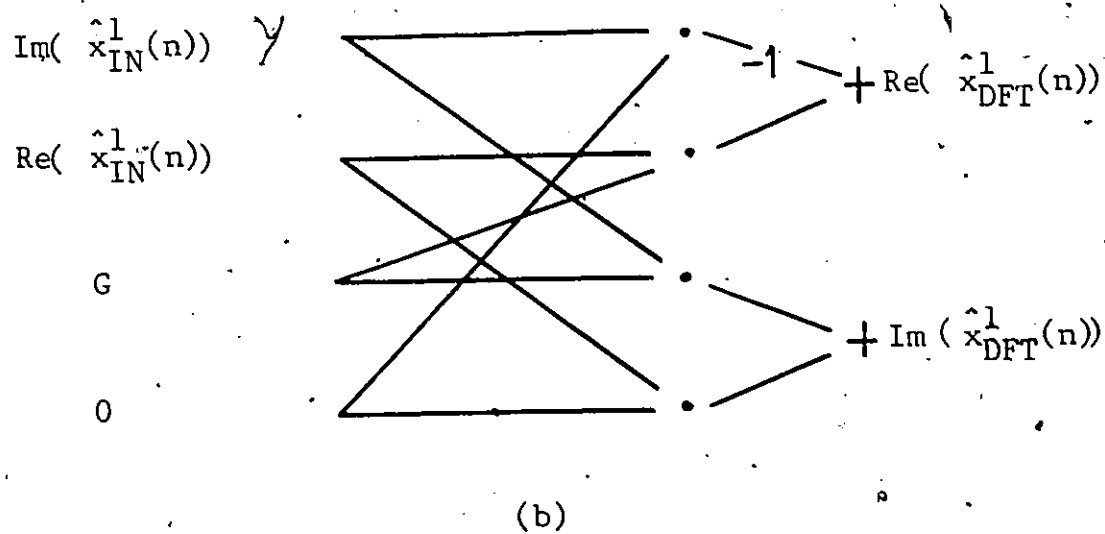
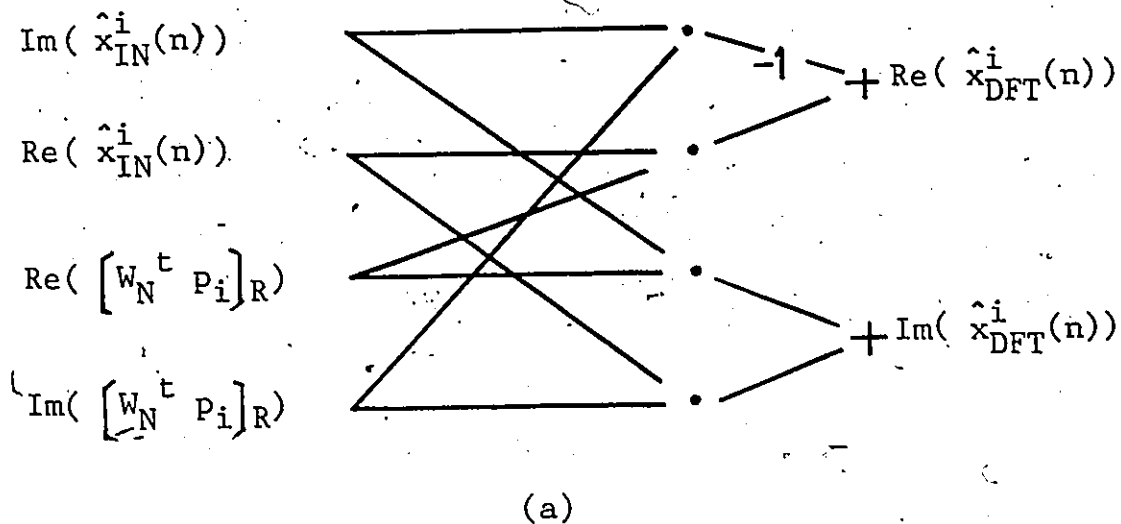


FIG. 4.6 (a) TWIDDLE FACTOR MULTIPLICATIONS  
 (b) ARTIFICIAL TWIDDLE FACTOR MULTIPLICATIONS  
 AT THE FIRST STAGE

$$B = 0.5 \log_2 M + 1.35 \quad (4.59)$$

and

$$B = \log_2 M - 2 \quad (4.60)$$

for the modified scheme with  $p_1=G$  and original scheme with  $p_1=1$ , respectively. It can be easily shown that when  $M \geq 104$ ,

$$\log_2 M - 2 \geq 0.5 \log_2 M + 1.35 \quad (4.61)$$

Since the required number range,  $M$ , is usually larger than 104, then the modified scheme uses less A/D quantization bits with the same error as compare to the original scheme. In other words, the reduction of A/D converter quantization bits does not increase the error,  $\alpha_1$ . This is because the error due to A/D quantization,  $\alpha_Q^2$ , is negligible as shown in the last column of Table 4.1.

We now conclude that the scaling scheme  $k_1=1$ ,  $k=1$  accompanied by the "artificial" twiddle factors in the first stage is much more suitable for sequential realization. In addition, the concept of storing constant integers in the first stage can be applied to spectral estimation, where the input to the FFT is usually weighted by some window function to reduce leakage effects, i.e. the window coefficients can also be stored in the first stage and treated as twiddle factors.

#### 4.5 An Example Design Problem

To illustrate how to use the design procedure given in the previous sections, we will consider the following example. An acceptable level of relative error,  $\alpha_1$ , is set to be equal to 0.01, and the scaling scheme  $k_1=2$ ,  $k=1$  is used. From Table 4.1 the corresponding value of  $M$  is given by the relationship

$$\alpha_1 = 10.18 M^{-\frac{1}{2}}$$

and thus  $M$  is found to be equal to  $1.036 \times 10^6$ . A possible set of moduli which satisfies the number range is  $\{32, 31, 29, 13, 3\}$ . Then

$$M_0 = 32 \times 31 \times 29 \times 13 \times 3 = 1.122 \times 10^6$$

Using the value of  $M_0$  and relationship between  $M$  and  $P$  given in Table 4.1, the integer conversion factor is calculated by

$$\begin{aligned} P &= 0.0982 M^{\frac{1}{2}} \\ &= 1.04 \times 10^2 \end{aligned}$$

Then, using equation (4.36), the scale factor,  $K$ , is found to be equal to  $4.161 \times 10^2$ . From the chosen set of moduli,  $\{32, 31, 29, 13, 3\}$ , the scaling moduli can be  $\{31, 13\}$

which gives  $K_0 = 403$ . The value of  $K_0$  may be not close enough to the derived value,  $4.161 \times 10^2$ . This can be possibly improved by either selecting a different set of moduli or using the method described in section 4.4.2. For this example, we will use the chosen scale factor, 403, and continue the design procedure to determine the rest of the parameters.

The integer conversion constant for the stage after the first scaling operation is equal to

$$P_3 = P_4 = P_5 = \left[ \frac{K_0}{4} \right]_R = 101$$

Substituting  $M_0$  and  $P$  into equation (4.38), one obtains

$$B = 9.40$$

or

$$B_0 = 9 \text{ bits}$$

Finally, using equation (4.37) or (4.38), the integer conversion constant for the second stage is found to be

$$P_2 = 137$$

Based on the calculated parameters, we also compute the relative RMS error using the general expression as shown in equation (4.32)

$$\alpha_1 = 0.00979$$

which is slightly less than the specified error, 0.01. The reason is that a larger value of the number range,  $M_0$ , than the required value has been used.

## DISCUSSION OF RESULTS

5.1 Introduction

The quantization error associated with the radix-4 DIT FFT processor has been analyzed in Chapter 4. This analysis has led to a practically useful design procedure.

In order to justify our predicted error results, the proposed FFT processor will be simulated by computer. Based on the simulated processor, relative output RMS error will be measured for several different input sequences, and also compared with the predicted results.

Finally, some ramifications of our theoretically derived error expressions will be described.

5.2 Simulation of The FFT Processor

In the RNS, exact integer results are always obtained from the basic operations of addition, subtraction and multiplication, hence these operations can be easily simulated by using entirely integer arithmetic. However, for a given range of the RNS,  $M$ , and a given scaling scheme, the corresponding required values of integer conversion and scale factors, and the A/D converter quantization bits, according to the derived theoretical interrelationships, need not be integers.

Although, in practice, they must be integers. In addition, it is not necessary to consider the range of the RNS,  $M$ , as a product of some relatively pairwise prime moduli, since this requirement is not explicitly shown in the

theoretical expressions. Therefore, for the purpose of comparison between theoretical and experimental results, we used  $M$  in ascending powers of 10, and the residue arithmetic was simulated using double precision floating point arithmetic instead of using exact integer arithmetic. To justify our simulated residue arithmetic using floating point arithmetic, a preliminary comparison between the error results obtained using exact integer arithmetic and using floating point arithmetic has been made, and it was found that the difference between them was negligible.

To facilitate the experimental simulation, two radix-4 DIT FFT programs were written, which can be seen in Appendix C. One was written in double precision floating point for arithmetic operations and also for the representations of twiddle factors. The results from this program were considered to be true results. In the other program, the arithmetic operations were performed using double precision floating point arithmetic, but the residue scaling scheme was incorporated and quantization errors due to A/D converter, integer conversion and scaling operation were properly simulated. The errors introduced by three different sources are all round-off errors, which can be simulated using the following FORTRAN statements,

$$X = X * 2^{B-1} + \text{DSIGN}(0.5, X)$$

$$C = C * P + \text{DSIGN}(0.5, C)$$

$$X = X / K + \text{DSIGN}(0.5, X)$$

where  $B$ ,  $P$  and  $K$  denote the A/D converter quantization bits,

integer conversion factor and scale factor, respectively, and

$$\text{DSIGN}(A1, A2) = \begin{cases} -A1 & \text{for } A2 < 0 \\ 0 & \text{for } A2 = 0 \\ A1 & \text{for } A2 > 0 \end{cases}$$

Moreover, before performing any scaling operation, a magnitude test routine was executed to detect if any number had magnitude larger than  $M/2$ , which is the maximum magnitude in the signed RNS. This is important, since the numbers in the RNS are performed over a finite range,  $M$ , and if there is an overflow the complete error simulation is flawed.

### 5.3 Comparison of Theoretical and Simulation Results

In order to characterize different types of input sequences, we will first define some terminologies which are useful to interpret experimental results. Since the level of the input to the A/D converter can be properly adjusted such that the maximum magnitude of the input is corresponding to the maximum integer from the output of the A/D converter,  $2^{B-1}$ , then it is best to define a nominal mean square value of the input sequence as a function of  $B$ . Let  $\sigma_a^2$  denote the actual mean square value of a input sequence, then the nominal mean square value,  $\sigma_n^2$ , is defined as

$$\sigma_n^2 = \sigma_a^2 \left( \frac{2^{B-1}}{x_{\max}} \right)^2 \quad (5.1)$$

where  $x_{\max}$  is the maximum magnitude of the input sequence. As an example, we consider that the input sequence is a uniformly distributed random number between  $-A$  and  $A$  for



both the real and imaginary parts, then

$$\sigma_a^2 = 2 \cdot \frac{A^3 + A^3}{3(A+A)} = \frac{2}{3} A^2 \quad (5.2)$$

and

$$\sigma_n^2 = \frac{2}{3} A^2 \cdot \left( \frac{2^{B-1}}{A} \right)^2 = \frac{2^{2B}}{6} \quad (5.3)$$

From equation (5.3), it is seen that the defined nominal mean square value is independent of the actual range of the input sequence. Since the theoretical relative RMS error was derived based on the assumption that the input mean square value is equal to  $\frac{2^{2B}}{6}$ , then an adjustment of the theoretical prediction can be made, if the ratio of the assumed nominal mean square value,  $\frac{2^{2B}}{6}$ , to the nominal mean square value of the input sequence,  $\sigma_n^2$ , is known. We now define an adjusting factor as

$$D = \sqrt{\frac{2^{2B}/6}{\sigma_n^2}} \quad (5.4)$$

such that the adjusted relative RMS error is equal to the product of the adjusting factor,  $D$ , and the theoretical predicted RMS error,  $\alpha_1$ .

Similarly, we define a nominal mean value of an input sequence to be the ratio of the actual mean value to the maximum magnitude of the sequence, which is also independent of the actual range of the input sequence. In addition,

the correlation coefficient between the real and imaginary parts of the input to the FFT processor is defined as

$$\rho = \frac{\overline{\text{Re}(x_{IN}^1(n)) \cdot \text{Im}(x_{IN}^1(n))} - \overline{\text{Re}(x_{IN}^1(n))} \cdot \overline{\text{Im}(x_{IN}^1(n))}}{\left\{ \left( \overline{(\text{Re}(x_{IN}^1(n)))^2} - \overline{\text{Re}(x_{IN}^1(n))}^2 \right) \cdot \left( \overline{(\text{Im}(x_{IN}^1(n)))^2} - \overline{\text{Im}(x_{IN}^1(n))}^2 \right) \right\}^{1/2}} \quad (5.5)$$

where  $-1 \leq \rho \leq 1$ . If  $\rho=0$ , the real and imaginary parts are uncorrelated.

Four different types of input sequences, including uniformly distributed pseudo random numbers, sine waves plus pseudo random numbers and speech signals, which were used to study the experimental upper bound on number growth in section 3.2.3, were also used as experimental input sequences. It is noted that the real and imaginary parts of the inputs to the FFT processor contain the same type of sequence in the experiments.

The first and second input sequences were pseudo random numbers. Both of them have the same nominal mean square values which are very close to the assumed value,  $\frac{2^{2B}}{6}$ . The only difference between them is the nominal mean value, where it is approximately equal to zero in the first case, and is equal to 0.5 in the second case. They are denoted as  $S_1$  and  $S_2$  in Table 5.1, respectively. Since the correlation coefficients are very small as shown in Table 5.1, the real and imaginary parts of  $S_1$  and  $S_2$  were considered to be uncorrelated.

The third type of input sequence was a sine waves plus

TABLE 5.1  
PROPERTIES OF SIMULATION INPUT SEQUENCES

Input	Nominal Mean Value	Nominal Mean Square Value	Correlation Coefficient	Adjusting Factor
S <sub>1</sub>	0.01	$\frac{2^2}{6.15}$	0.02	1.01
S <sub>2</sub>	0.50	$\frac{2^2}{6.15}$	0.02	1.01
S <sub>3</sub>	0.01	$\frac{2^2}{8.72}$	0.69	1.21
S <sub>4</sub>	0.01	$\frac{2^2}{14.96}$	0.20	1.58

zero-mean pseudo random numbers, which was given in equation (3.11) and denoted as  $S_3$  in Table 5.1. The nominal mean value is close to zero, and the nominal mean square value is slightly smaller than  $\frac{2^{2B}}{6}$ . Also, this sequence is well-correlated, because  $\rho = 0.69$  as shown in Table 5.1.

The last type of experimental sequence was a typical speech waveform. It is denoted as  $S_4$  in Table 5.1. From Table 5.1, it can be seen that the nominal mean value is also close to zero, but the nominal mean square value is only about half the value of  $\frac{2^{2B}}{6}$ . In addition, from Table 5.1, it is seen that  $\rho = 0.20$  for  $S_4$ , thus the real and imaginary parts of this input are slightly correlated.

Each of the values delineated in Table 5.1 was taken from the ensemble average value. The adjusting factors for each sequence are also shown in the last column of Table 5.1.

The first experiment was to have the random numbers,  $S_1$ , as the input sequence, since it satisfied the first simplifying assumption made in section 4.3. Simulation results were used to establish a relationship between the error,  $\alpha_1$ , and the range of the number system,  $M$ , being used. These results are shown in Figure 5.1. The corresponding results predicated by the theoretically derived relationships tabulated in Table 4.1 have also been plotted in Figure 5.1. The small error between the theoretical and the simulation results shown in Figure 5.1 for various scaling schemes indicates that equation (4.46) and the subsequent relationships tabulated in Table 4.1 can be used with confidence in the practical design

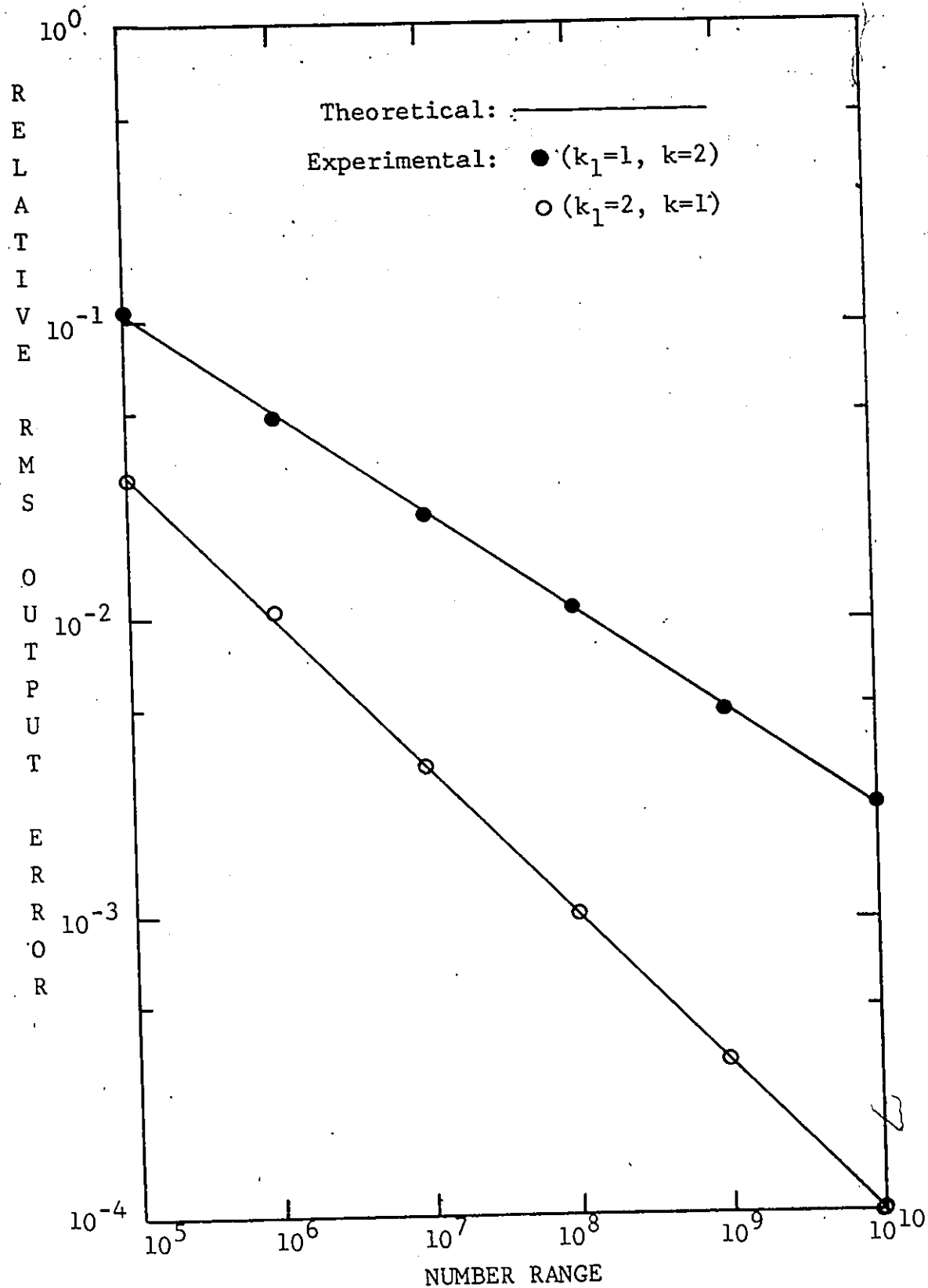


FIG. 5.1(a) RELATIVE RMS OUTPUT ERROR VERSUS NUMBER RANGE  
 FOR  $S_1$ ;  $k_1=1, k=2$  AND  $k_1=2, k=1$

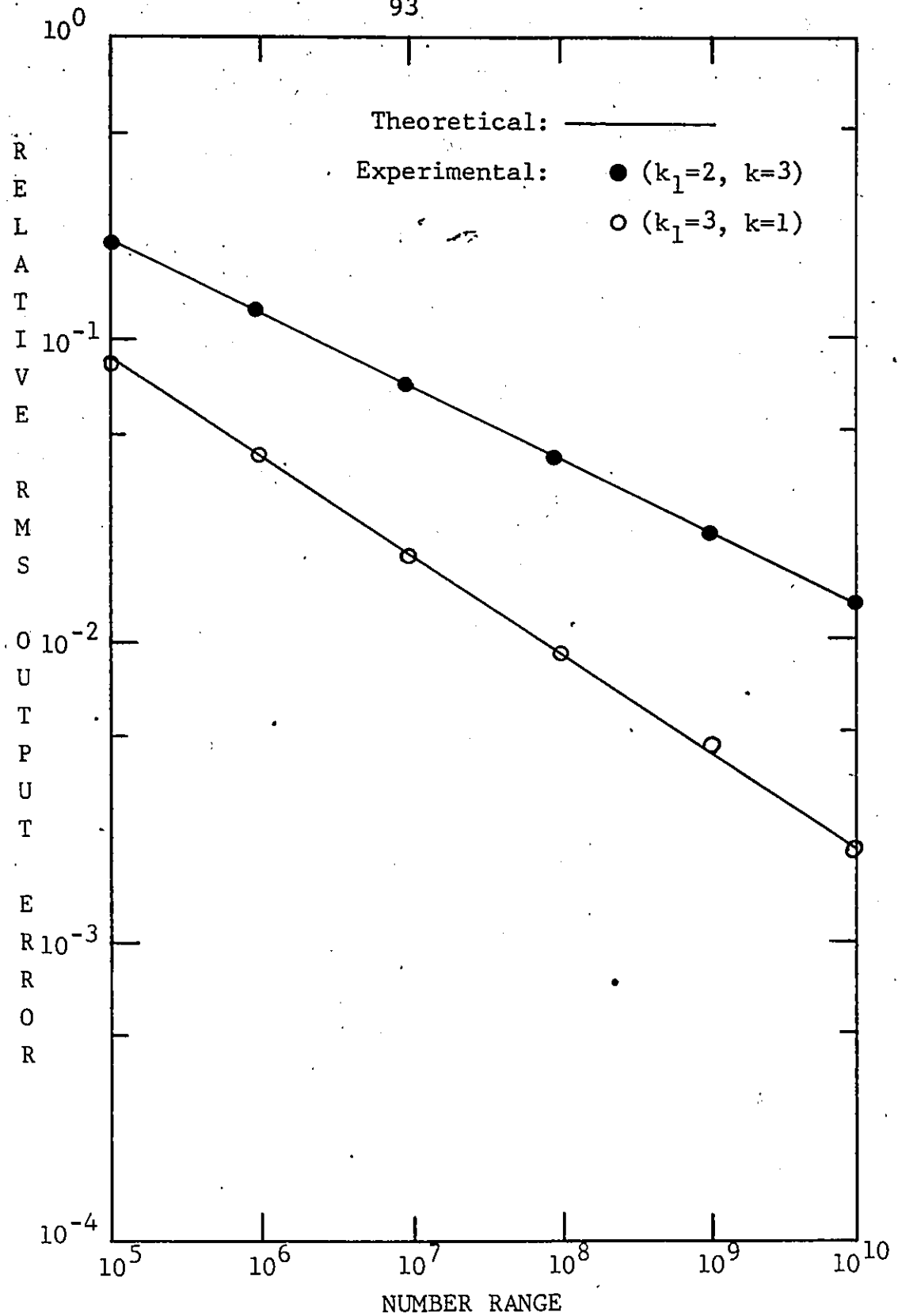


FIG. 5.1(b) RELATIVE RMS OUTPUT ERROR VERSUS NUMBER RANGE  
FOR  $S_1$ ;  $k_1=2, k=3$  AND  $k_1=3, k=1$

of a FFT processor.

In the second experiment, we chose  $S_2$  as the input sequence. As we can see from Table 5.1, the only difference between  $S_1$  and  $S_2$  is their nominal mean values. The simulation and corresponding theoretical results were plotted in Figure 5.2 for both  $S_1$  and  $S_2$ . From Figure 5.2, it is seen that the simulation results of  $S_2$  lie below the theoretical curve. The reason may be that the actual mean value of the errors associated with the integer conversion of the twiddle factors,  $e_I$ , is not zero as assumed in the statistical error models. This will not affect the previous results using zero mean input sequence,  $S_1$ , because for inputs with this property,  $x_{OUT}^i(n) = 0$  except at the last stage. Thus equations (4.19) and (4.20) can still be obtained by using  $x_{OUT}^1(n) = 0$ , and  $E_{OUT}^1(n) = 0$ . For a non-zero mean input, equations (4.19) and (4.20) are not correct unless  $\overline{e_I} = 0$ . Although a closed form to show the effect of non-zero mean value of the input sequence can not be obtained, the theoretically derived expression may serve as a pessimistic bound for non-zero mean input.

So far, we have only considered the cases of uncorrelated input sequences,  $S_1$  and  $S_2$ . The comparison between theoretical and simulation results for correlated input sequence with zero-mean value,  $S_3$ , is shown in Figure 5.3. The adjusted theoretical results are also shown in Figure 5.3. From Figure 5.3, it is seen that the simulation results lie slightly below the adjusted theoretical results. This is possibly

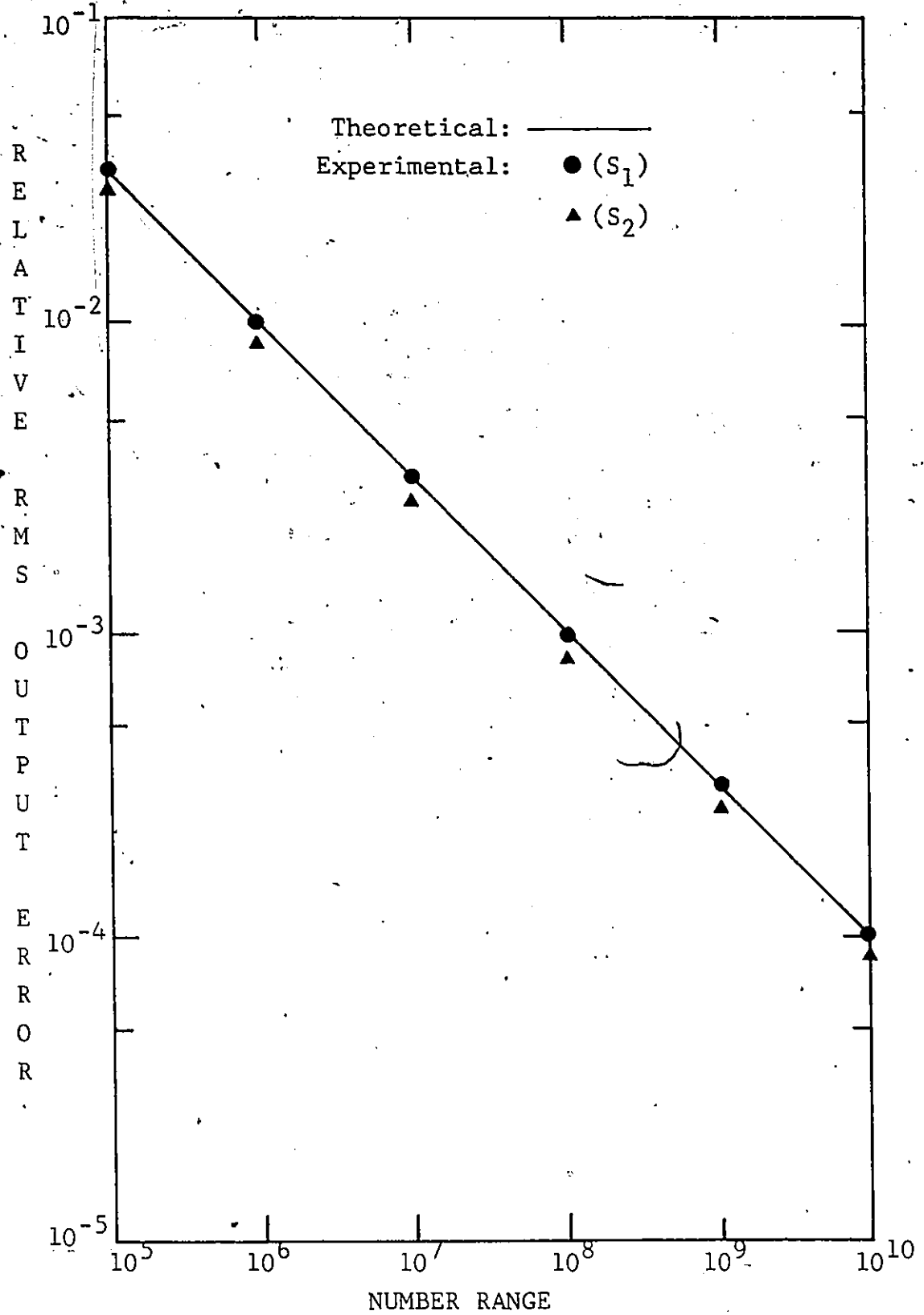


FIG. 5.2 RELATIVE RMS OUTPUT ERROR VERSUS NUMBER RANGE  
FOR  $S_1$  AND  $S_2$ ;  $k_1=2$ ,  $k=1$



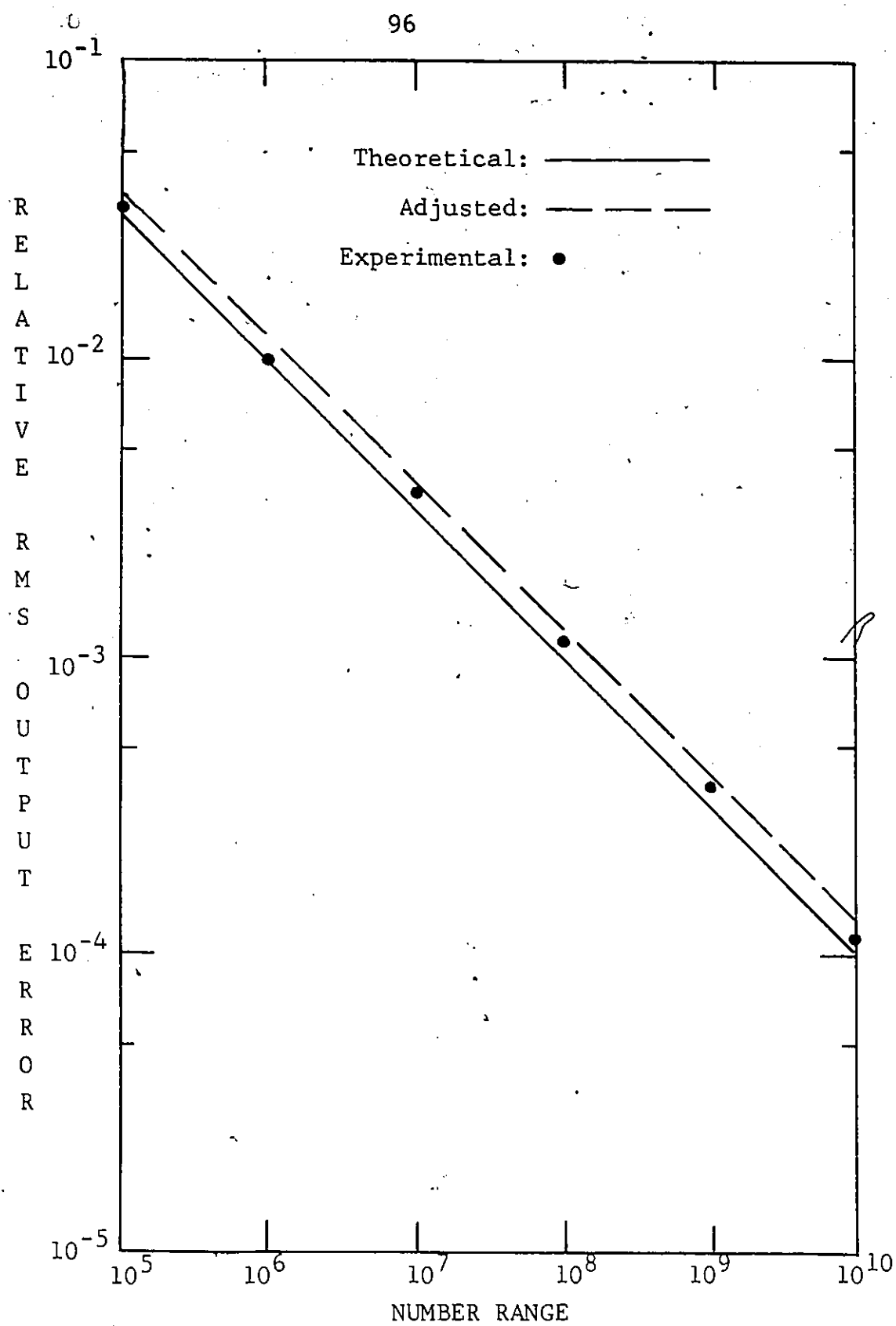


FIG. 5.3 RELATIVE RMS OUTPUT ERROR VERSUS NUMBER RANGE  
FOR  $S_3$ ;  $k_1=2$ ,  $k=1$

due to the correlation of the real and imaginary components of the input sequence, which will produce correlated quantization error. However, even though the correlation coefficient,  $\rho$ , is as high as 0.69, the effect is not significant. Thus our theoretical results accompanied by the adjustment factor are quite accurate for predicting errors of a correlated input sequence.

Figure 5.4 shows the results of transforming speech signal,  $S_4$ . Since the real and imaginary parts of this input sequence is correlated, the simulation results should lie below the adjusted theoretical results. This is verified in Figure 5.4.

In section 4.5.2, we have discussed a modified scheme which allows  $p_2$  to be variable and leads to a better result. The simulation and theoretical results associated with the modified scheme were plotted in Figure 5.5 for the input sequences,  $S_1$  and  $S_4$ . Again, the simulation and theoretical results for  $S_1$  are in good agreement, and the simulation results of  $S_4$  lie slightly below the theoretical results.

By storing some dummy constant integers in the first stage, identical structures for each stage is obtained for the scaling scheme  $k_1=1$ ,  $k=1$ . The theoretical and simulation results of transforming  $S_1$ ,  $S_2$  and  $S_4$  using this scheme are shown in Figure 5.6, which are similar to the previous results.

In addition to the above experiments, where both real and imaginary parts were comprised the same type of input sequence, we have also considered the case where only real parts contained

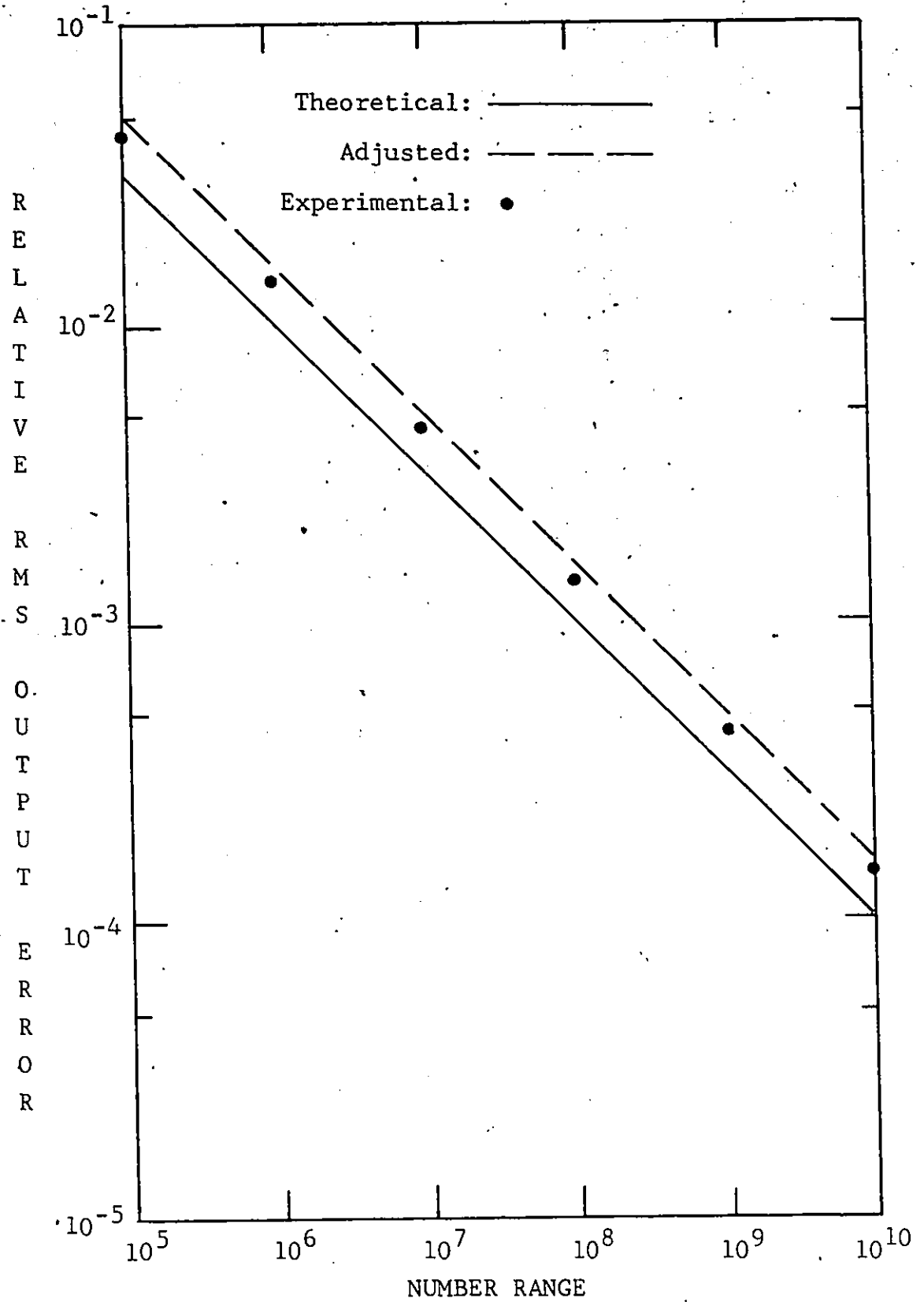


FIG. 5.4 RELATIVE RMS OUTPUT ERROR VERSUS NUMBER RANGE  
FOR  $S_4$ ;  $k_1=2$ ,  $k=1$

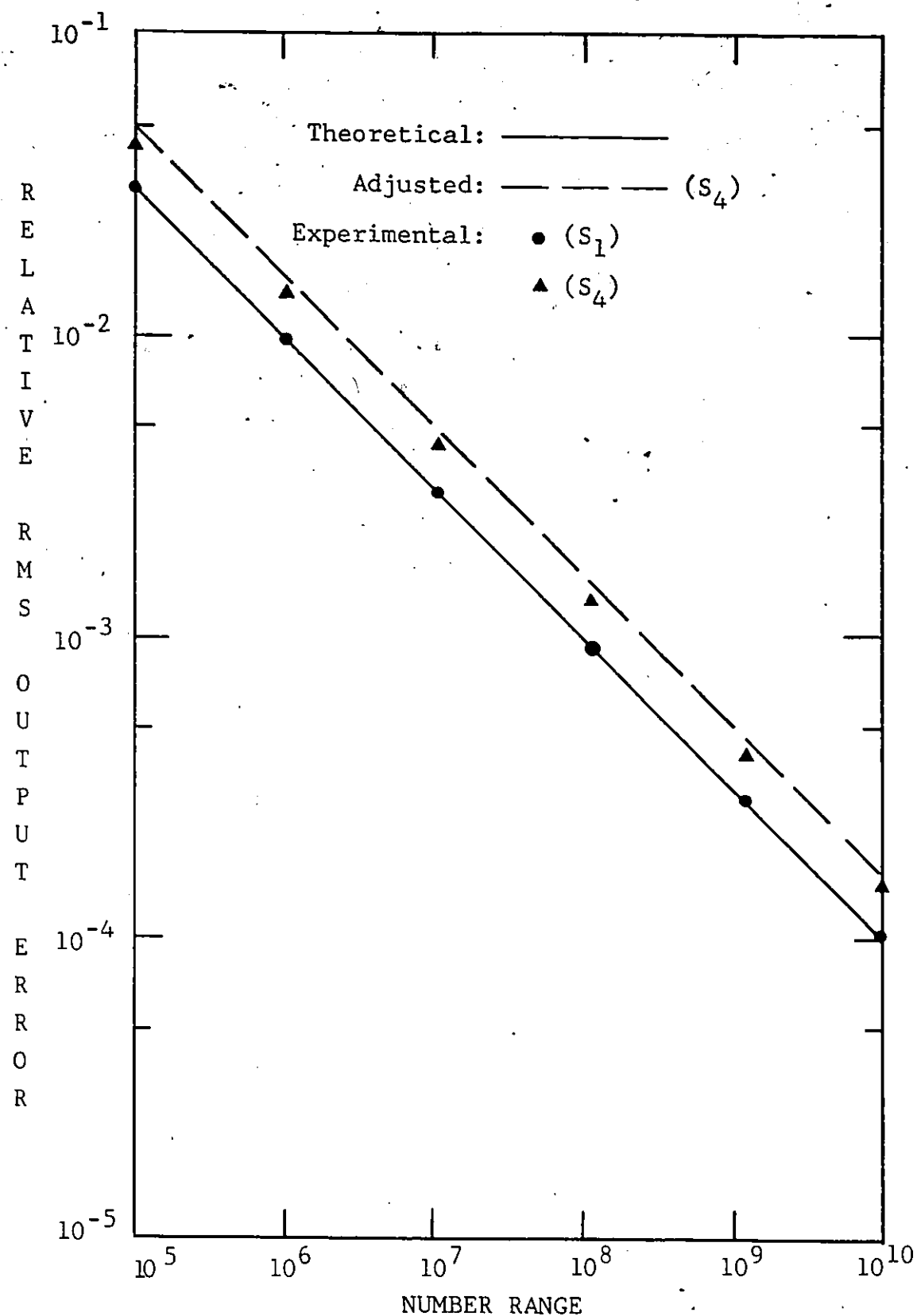


FIG. 5.5 RELATIVE RMS OUTPUT ERROR VERSUS NUMBER RANGE  
FOR  $S_1$  AND  $S_2$ ;  $p_2=P$ ,  $k_1=2$ ,  $k=1$

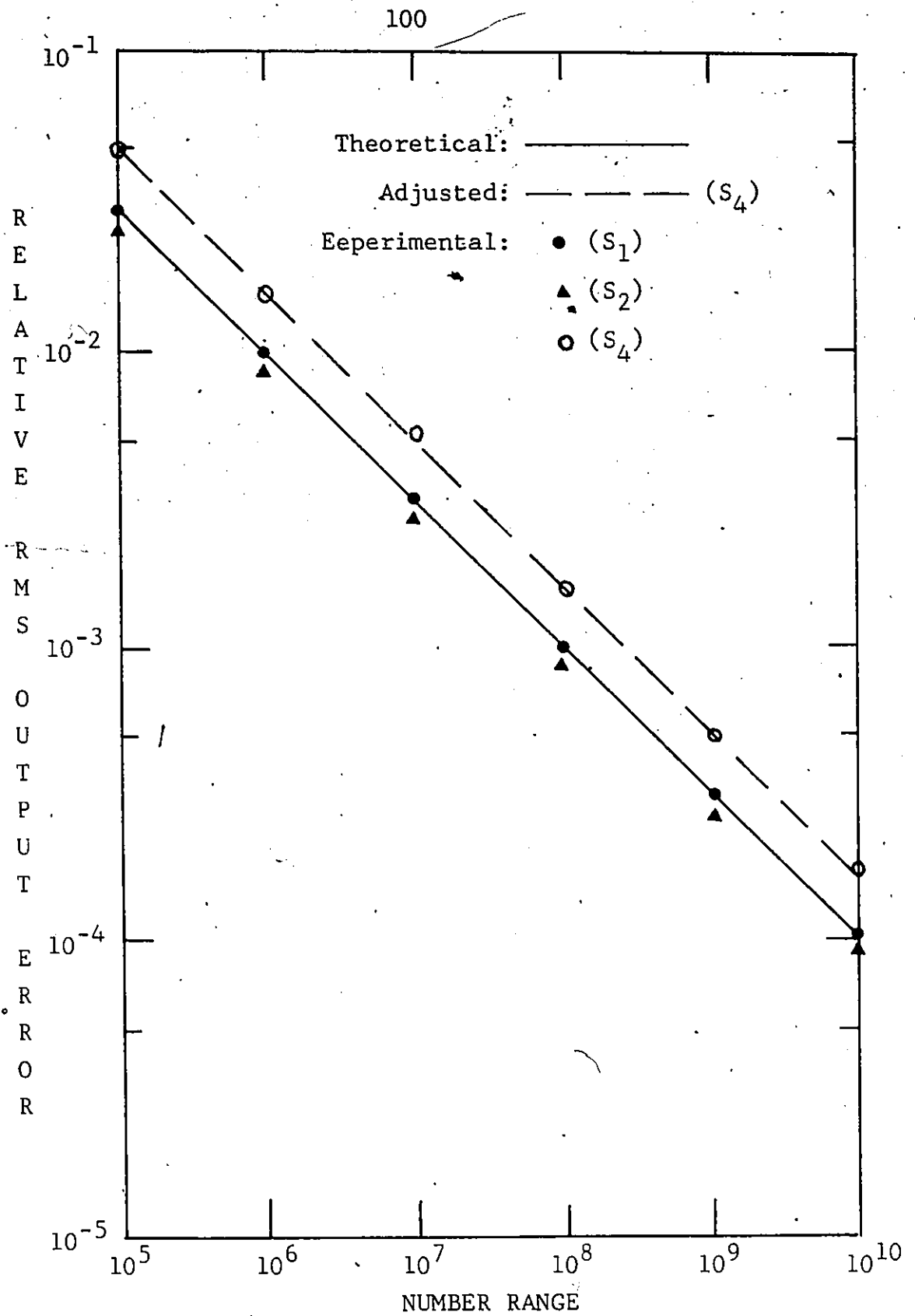


FIG. 5.6 RELATIVE RMS OUTPUT ERROR VERSUS NUMBER RANGE  
FOR S<sub>1</sub>, S<sub>2</sub> AND S<sub>4</sub>; p<sub>1</sub>=P, k<sub>1</sub>=1, k=1

data, while imaginary parts were filled up with zeros. For this case, the predicted results for  $S_1$  and  $S_2$  also have to be adjusted. The adjusted factors for  $S_1$ ,  $S_2$  and  $S_3$  are 1.414, 1.414 and 1.711, respectively. Figure 5.7 shows the results of transforming  $S_1$  and  $S_2$ . From Figure 5.7, it is seen that the simulation results of  $S_2$  still lie slightly below the results of  $S_1$ . From the previous experience, there was always a very good agreement between the simulation and predicted results for  $S_1$ . But, from Figure 5.7, it is seen that there is a small error between them. The reason is that the adjusting factor defined in equation (4.4) is not suitable for a real input FFT. The derivation of a more reasonable adjusting factor can be seen in Appendix D. The newly adjusted results are also shown in Figure 5.7. Again, we obtain a very good agreement between experimental and correctly adjusted theoretical results ( $D=1.22$ ). The results of transforming the real input sequence,  $S_3$ , are shown in Figure 5.8. From Figure 5.8, it is seen that the correctly adjusted results ( $D=1.48$ ) is much closer to the experimental results than the previously adjusted results ( $D=1.71$ ).

Finally, the experimental upper bound at each stage for real input sequences  $S_1$ ,  $S_2$  and  $S_3$  is also shown in Table 5.2. If we compare the bounds in Table 5.2 to Table 3.2, we see that they are close each other. Therefore, 4 is also a reasonable bound for the real input FFT.

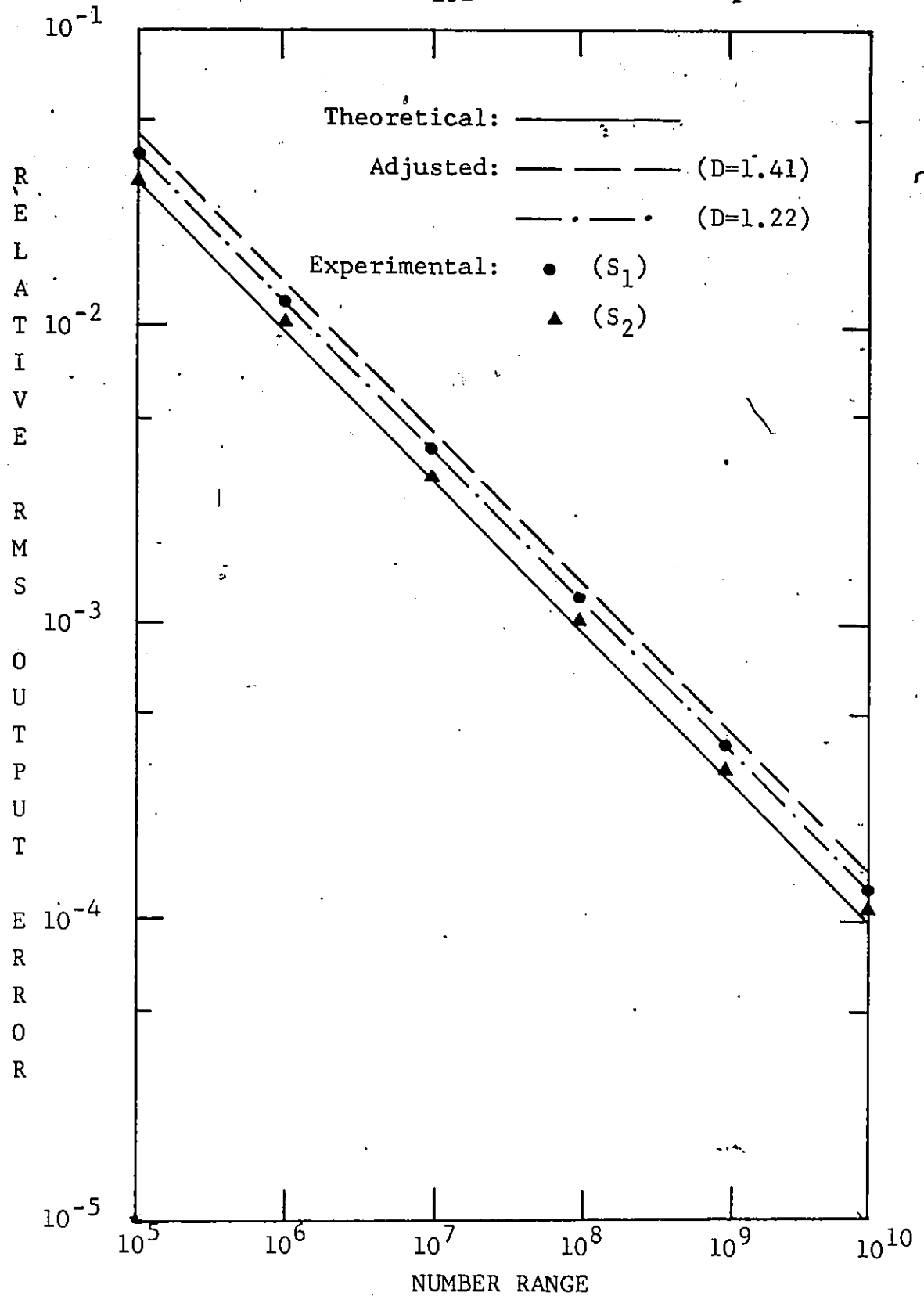


FIG. 5.7 RELATIVE RMS OUTPUT ERROR VERSUS NUMBER RANGE  
FOR  $S_1$  AND  $S_2$  WITH REAL INPUT;  $k_1=2$ ,  $k=1$

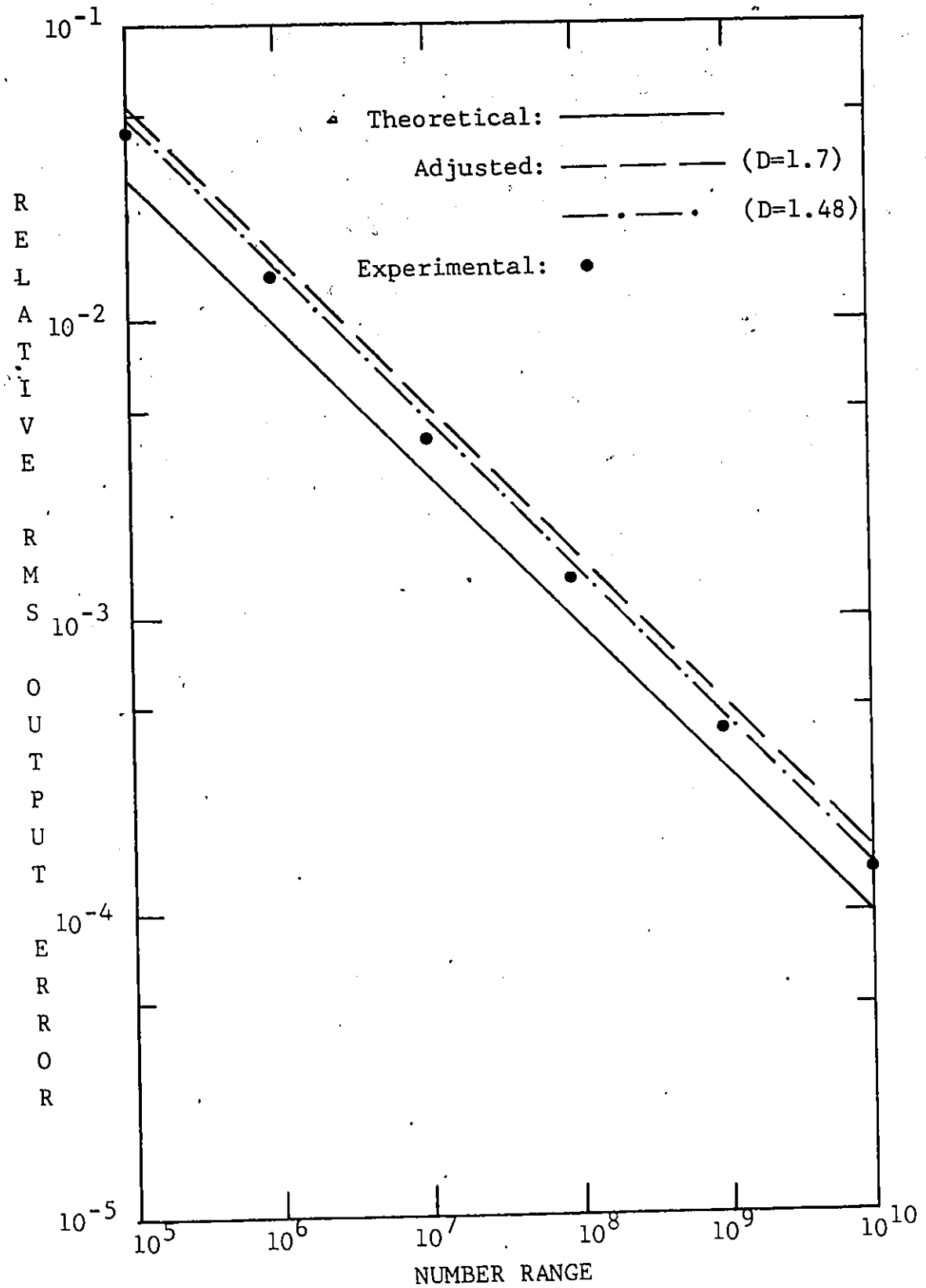


FIG. 5.8 RELATIVE RMS OUTPUT ERROR VERSUS NUMBER RANGE  
FOR  $S_3$  WITH REAL INPUT;  $k_1=2$ ,  $k=1$



TABLE 5.2  
SIMULATION STUDY OF WORST CASE UPPER BOUND OF  
THE NUMBER GROWTH AT EACH STAGE OF A REAL INPUT RADIX-4 DIT FFT

Stage No. Input Type	1	2	3	4	5
$S_1$	3.62	3.08	2.25	1.57	1.98
$S_2$	3.55	3.70	3.40	3.88	3.91
$S_3$	3.00	2.56	3.22	3.63	3.87

## 5.4 Ramifications

### 5.4.1 Fixed-point Arithmetic Number System

In the experimental simulation, we had the number range,  $M$ , in ascending power of 10, since the requirement of  $M$  to be a product of some relatively pairwise prime moduli is not explicitly shown in the theoretical expressions. In fact, our derived error expression can be applied to any integer based arithmetic number system. Since the most common integer based arithmetic number system is the fixed-point arithmetic number system, we will, in the following, discuss how the error results can be applied to it.

First, some similarities between our error analysis and other workers' fixed-point error analysis will be indicated. In our analysis, scale factors were chosen on a priori basis and the position of scaling arrays was predetermined. Thus, the scaling scheme and scale factor were prefixed, that is similar to the first scheme, shifting right one bit at every iteration, as described in [32], or the so-called automatic array scaling in [33]. Furthermore, we considered that several exact arithmetic operations were performed before scaling by a predetermined constant. In other words, the results at the output of the BCU were retained to full accuracy. This type of arithmetic unit for radix-2 FFT has also been considered in [33].

When the FFT is implemented using fixed-point arithmetic, it is more convenient to express the number range, integer conversion factors and scale factor in terms of number of

bits. If the scale factor is equal to  $2^{b_s}$ , then the scaling process can be implemented by shifting right  $b_s$  bits. To illustrate the design procedure for the fixed-point arithmetic using our theoretical error expressions, we will consider the following example. The error,  $\alpha_1$ , is specified as 0.01, and the scaling scheme  $k_1=2$ ,  $k=1$  is used for  $m=5$ . From Table 4.1, the corresponding value of  $M$  is given by the relationship

$$\alpha_1 = 10.18 M^{-\frac{1}{2}}$$

and hence  $M = (10.18 / \alpha_1)^2 = 1.036 \times 10^6$ . Since at least 20 bits are required to represent the value,  $1.036 \times 10^6$ , then the chosen value of  $M$ ,  $M_0$ , is

$$M_0 = 2^{20} = 1.049 \times 10^6.$$

The integer conversion constant,  $P$ , can be determined using the relationship,  $P = 0.0982 M^{1/2}$ , given in Table 4.1. The value of  $P$  is found to be 100.56. Since there is one stage between scaling operation, then

$$K = 4P = 402.24$$

The corresponding number of bits for the value, 402.24, is 8.65 bits. The choice can be 8 or 9 bits. We will assume that 9 bits is used. Therefore, we obtain

$$P = \frac{2^9}{4} = 2^7$$

which indicates that 7 bits are used to represent the twiddle factors. After the values of M and P have been determined, the required number of A/D converter quantization bits can be calculated using equation (4.38). For this example, B=9 bits.

Since the required word length for different units were given in the fixed-point radix-4 FFT described in [27], then a comparison between our chosen word length and that given in [27] can be made, which is shown in Table 5.3. The scaling scheme used in [27] is a conditional array scaling<sup>+</sup>, which employs different normalization procedures at each stage. This type of scaling scheme should produce less output error than the fixed scheme. From Table 5.3, it can be seen that the required word length for different parameters or units are comparable for two schemes. Thus our design procedure can also be used to select proper word length for a fixed-point FFT.

In the above fixed-point implementations, we consider that the results at the output of the twiddle factor multiplications retain with full word length. In addition, there is another method, which is quite often used to implement the fixed-point FFT. That is, the results at the output of twiddle factor multiplications are rounded to a certain number

---

<sup>+</sup> In [27], it is called automatic array scaling, but according to [33], it should be called a conditional scaling scheme.

TABLE 5.3  
COMPARISON OF WORD LENGTH REQUIRED AT VARIOUS UNITS OF  
A RADIX-4 FFT PROCESSOR FOR TWO DIFFERENT METHODS USING FIXED-POINT ARITHMETIC

Scheme Unit (Parameter)	Our Scaling Scheme $k_1=2, k=1$ ; DIT FFT	Corinthios' Scaling Scheme; DIF FFT
A/D converter	8 bits + sign bit	8 bits + sign bit
Twiddle Factor	7 bits + sign bit	10 bits + sign bit
Maximum Word Length	16 bits + sign bit	20 bits + sign bit *
Scaling Scheme	Shift Right 9 Bits at Every Stage after The Second Stage	Normalize The Numbers at The Input to Each Stage; Truncate The Results at The Output of Each Stage to 9 Bits
Error	0.01	Less Than 0.01

\* This word length is at the output of the multiplier. The maximum word length may require one more bit to take into account the addition or subtraction after the multiplication to yield the outputs of the BCU.

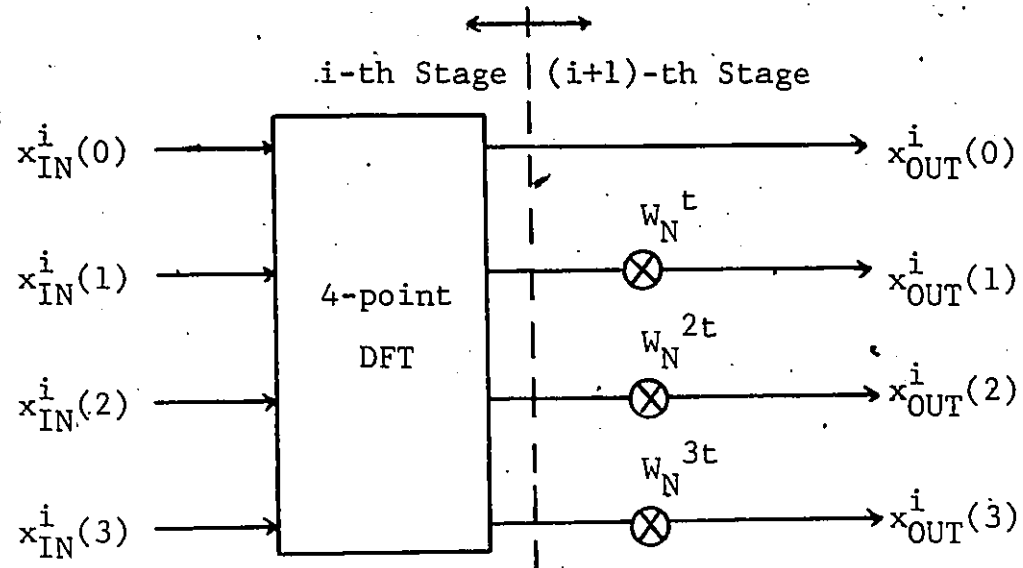
of bits. The latter generates larger error but requires less word length than the former.

Since exact integer results are always obtained from the multiplication operation of the RNS, our derived results based on the RNS can be easily adopted to the first type of fixed-point FFT. For the second type, our results can not be used directly. However, the change associated with the rounding effects is not too difficult, which is given in Appendix E.

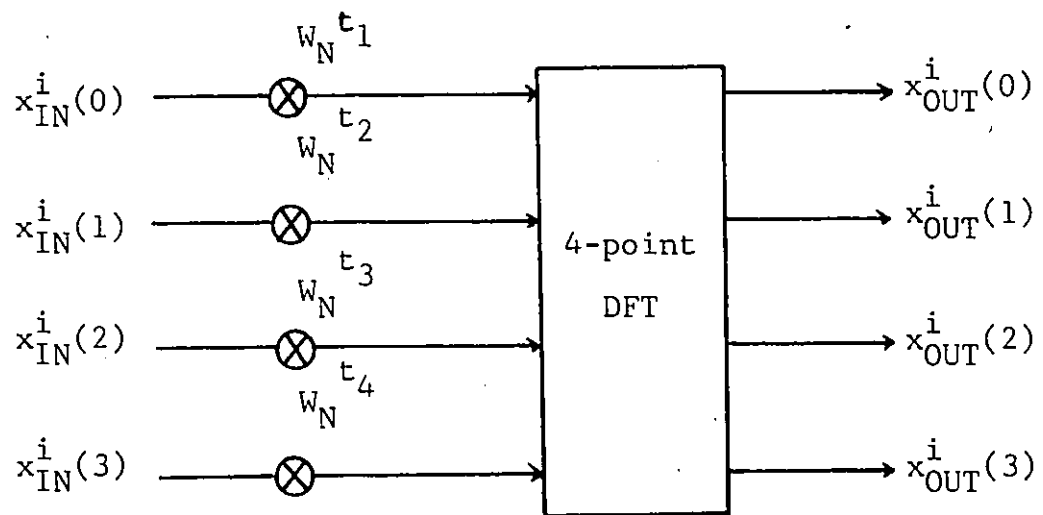
#### 5.4.2 The DIF Algorithm

In section 3.2, we considered that the DIT algorithm is better than the DIF algorithm, because the overall worst case upper bound is larger for the DIF algorithm. If, however, the DIF algorithm is to be used, can our error analysis also be applied to it? The answer is yes, although some modifications are required.

From Figure 2.3, it can be seen that the basic structures for the DIT and DIF algorithms are different. Thus, in order to apply our error results to the DIF algorithm, the structure of the DIF algorithm must be changed such that it will be similar to the DIT algorithm. This can be done if the twiddle factor multiplications in the DIF basic calculation are treated as part of the following stage. This is shown in Figure 5.9. It is noted that the twiddle factors in the first stage of the modified DIF algorithm are equal to one. Since the twiddle factor multipliers are equally distributed over the 4 input points, this algorithm was termed the symmetric algorithm in [25]. Table 5.4 also shows the experimental upper bounds for 5



(a)



(b)

FIG. 5.9 (a) BASIC MODULE OF AN ORIGINAL RADIX-4 DIF FFT  
 (b) BASIC MODULE OF A MODIFIED RADIX-4 DIF FFT

TABLE 5.4  
SIMULATION STUDY OF WORST CASE UPPER BOUND OF  
THE NUMBER GROWTH AT EACH STAGE OF A MODIFIED RADIX-4 DIF FFT

Stage No. Input Type	1	2	3	4	5
1	3.62	3.42	2.20	1.64	1.88
2	3.77	3.73	3.19	3.75	3.90
3	2.90	3.64	3.23	3.61	3.91
4	3.22	2.64	3.19	3.74	3.90
5	1.95	2.14	3.32	3.69	3.98



different inputs used in section 3.2. From Table 5.4, it can be seen that 4 is also a reasonable upper bound on number growth at each stage of the modified radix-4 DIF FFT.

## CHAPTER 6

### CONCLUSIONS

This thesis has considered some of the design problems associated with a ROM implementation of a RNS based FFT processor. The specification of a desired RMS error of the frequency domain estimate computed by the FFT processor has been used as a criterion of optimality for the design procedure.

Based on the minimum number of cascaded multiplications, it was found that a radix-4 FFT structure was the most viable choice. Since the RNS is an integer based number system, the results from the operations of addition, subtraction and multiplication are exact. However, due to a limited number range in the RNS, all numbers must be properly scaled to avoid overflows. Theoretical and simulation studies of number growth in the radix-4 FFT processor indicated that growth by a factor of 4 at each stage can be used as the basis for scaling.

A general expression for predicting the relative RMS error at the output of the FFT processor has been derived. By making a number of simplifying assumptions, it has been possible to develop a number of expressions that can be used directly in the practical design of the FFT processor described in the thesis. The design procedure allows one to specify the relative RMS output error,  $\alpha_1$ , and then determine the appropriate values of the number system range,  $M$ , the number of A/D quantization bits,  $B$ , the scale factor,  $K$ , and the integer

conversion factor,  $P$ . Since the scaling scheme must be specified in the design procedure, it was found that the scaling scheme that provides two stages before the first scaling operation and then scaling after each following stage, was the best scheme, as far as the values of  $M$ ,  $K$  and  $B$  were concerned.

The use of RNS, constrains the scale factor,  $K$ , to be a product of some of the moduli used in the system rather than the exact value given by the derived functional relationships between  $M$ ,  $K$  and  $\alpha_1$ . Since the values of the A/D quantization bits,  $B$ , and the integer conversion factor,  $P$ , are normally functions of  $K$ . They will be affected by changing the derived value of  $K$  to the closest suitable value. However, by letting the integer conversion factor at the second stage,  $p_2$ , be a variable, a modified scheme that eliminates the joint dependency of  $B$  and  $p_2$  on  $K$  has been developed.

It is known that the simplest and cheapest realization of a FFT processor is a sequential realization. For this type of realization, only one computation unit is required, hence it is best to have similar computations in each stage. Obviously, the best candidate for this realization is the scaling scheme that requires one scaling operation at the output of each stage. Unfortunately, this scheme needs a very large number of A/D converter quantization bits because of the lack of twiddle factors in the first stage. This problem can be resolved by introducing some dummy constant integers to serve as twiddle factors. This concept can also be applied to the case of

storing windowing coefficients in spectrum analyzer applications.

The theoretical results derived for various scaling schemes were verified for various types of input signals using simulation techniques. Although the analysis developed in this thesis was based on the assumption that the input sequence was a set of uniformly distributed pseudo-random numbers, it has been shown that the theoretical expressions associated with some of the adjusting factors can still predict the relative RMS output error for other types of input signals accurately.

The design expressions derived in this thesis are related to a radix-4 DIT FFT using the RNS, however these relationships can be extended directly to the modified radix-4 DIF FFT case. In addition, the design expressions are also applicable to any integer based arithmetic system including a binary fixed-point representation.

3

APPENDICES

## APPENDIX A

### NUMBER GROWTH AT EACH STAGE OF A RADIX-4 DIF FFT

#### A.1 Mean-Square Bound of A Radix-r DIF FFT

In this section, we will show that the mean-square value at the output of each stage will increase by  $r$  for a radix-4 DIF algorithm.

The basic form of the  $r$ -point transform with DIF is given by

$$x_{i+1}(k) = \left( \sum_{n=0}^{r-1} x_i(n) W_r^{nk} \right) (W_N^t)^k \quad (\text{A.1})$$

where  $\{x_i(n)\}$  denote the number at the input of the  $i$ -th stage, and  $(W_N^t)^k$  are the appropriate twiddle factors.

By letting

$$y_i(k) = \sum_{n=0}^{r-1} x_i(n) W_r^{nk} \quad (\text{A.2})$$

in equation (A.1), one obtains

$$x_{i+1}(k) = y_i(k) (W_N^t)^k \quad (\text{A.3})$$

Taking the squared value of both sides in equation (A.3), one obtains

$$\left| x_{i+1}(k) \right|^2 = \left| y_i(k) \right|^2 \cdot \left| (W_N^t)^k \right|^2$$

or

$$|x_{i+1}(k)|^2 = |y_i(k)|^2 \quad (\text{A.4})$$

where  $|(W_N^t)^k|^2 = 1$ . Since equation (A.2) represents an  $r$ -point DFT, then

$$\sum_{k=0}^{r-1} |y_i(k)|^2 = r \sum_{n=0}^{r-1} |x_i(n)|^2 \quad (\text{A.5})$$

Substituting equation (A.4) into equation (A.5), one has

$$\sum_{k=0}^{r-1} |x_{i+1}(k)|^2 = r \sum_{n=0}^{r-1} |x_i(n)|^2 \quad (\text{A.6})$$

Equation (A.6) can be generalized as

$$\frac{1}{N} \sum_{k=0}^{N-1} |x_{i+1}(k)|^2 = r \cdot \frac{1}{N} \sum_{n=0}^{N-1} |x_i(n)|^2 \quad (\text{A.7})$$

This complete the proof.

#### A.2 Theoretical Worst Case Upper Bound on Number Growth for A Radix-4 DIF FFT

Since equation (A.2) can be recognized as a 4-point DFT with  $x_i(n)$  and  $y_i(k)$  as the input and output, respectively, then one should have

$$\text{Max} \left\{ \left| \text{Re}(y_i(k)) \right|, \left| \text{Im}(y_i(k)) \right| \right\} \leq 4 \text{Max} \left\{ \left| \text{Re}(x_i(n)) \right|, \left| \text{Im}(x_i(n)) \right| \right\} \quad (\text{A.8})$$

Substituting  $(W_N^t)^k = \cos \frac{2\pi kt}{N} - j \sin \frac{2\pi kt}{N}$  into equation (A.3), one obtains

$$x_{i+1}(k) = \left\{ \operatorname{Re}(y_i(k)) \cos \frac{2\pi kt}{N} + \operatorname{Im}(y_i(k)) \sin \frac{2\pi kt}{N} \right\} + j \left\{ \operatorname{Im}(y_i(k)) \cos \frac{2\pi kt}{N} - \operatorname{Re}(y_i(k)) \sin \frac{2\pi kt}{N} \right\} \quad (A.9)$$

Equation (A.9) can be written as

$$\begin{aligned} & \operatorname{Max} \left\{ \left| \operatorname{Re}(x_{i+1}(k)) \right|, \left| \operatorname{Im}(x_{i+1}(k)) \right| \right\} \\ & \operatorname{Max} \left\{ \left| \operatorname{Re}(y_i(k)) \right|, \left| \operatorname{Im}(y_i(k)) \right| \right\} \cdot \left\{ \left| \cos \frac{2\pi kt}{N} \right| + \left| \sin \frac{2\pi kt}{N} \right| \right\} \end{aligned} \quad (A.10)$$

Combining (A.10) and (A.8), one obtains

$$\frac{\operatorname{Max} \left\{ \left| \operatorname{Re}(x_{i+1}(k)) \right|, \left| \operatorname{Im}(x_{i+1}(k)) \right| \right\}}{\operatorname{Max} \left\{ \left| \operatorname{Re}(x_i(n)) \right|, \left| \operatorname{Im}(x_i(n)) \right| \right\}} \leq 4 \left\{ \left| \cos \frac{2\pi kt}{N} \right| + \left| \sin \frac{2\pi kt}{N} \right| \right\} \quad (A.11)$$



## APPENDIX B

### AN ERROR ANALYSIS OF

### A SEQUENTIALLY REALIZED RADIX-4 DIT FFT

In this appendix, the optimum functional relationships between some parameters associated with the storing of some constant integers in the first stage of the FFT will be derived. The scaling scheme  $k_1=1$ ,  $k=1$  is considered to be used.

Since the stored integers in the first stage can be treated as twiddle factors, we will consider that the multiplication by a constant is part of the required operations in the first stage. Thus, the constant integer,  $G$ , can also be denoted as  $p_1$ .

Similar to the scale factor consideration discussed in section 4.4.2, we may let  $G=p_1 \neq P$ , and  $p_2=p_3=\dots p_m=P$ . However, by doing so, we found that an extreme condition,  $G=p_1=1$ , will minimize the error;  $\alpha_1$ . Thus we obtained the same relationships between  $M$ ,  $P$  and  $\alpha_1$  as given in Table 4.1. Therefore in the following we will consider that all  $p_i$ 's are equal to a constant, including the value of  $G$ .

From equations (4.40) and (4.41), one can easily obtain the relative output mean square error due to A/D converter quantization,  $\alpha_Q^2$ , as

$$\alpha_Q^2 = \frac{.1}{2^{2B}} = \frac{(4P)^2}{M^2} \quad (B.1)$$

Since the multiplication by a constant integer,  $p_1$  or  $G$ , does not introduce any error, then

$$\alpha_I^2 = \frac{1}{8} \sum_{i=2}^m \frac{1}{p_i^2} = \frac{m-1}{8P^2} \quad (\text{B.2})$$

The error due to scaling operations,  $\alpha_S^2$ , will have the same form as that in equation (4.32) but  $p_1=G=P \neq 1$ . Using the simplifying assumptions made in section 4.3, one has

$$\alpha_S^2 = \frac{4}{3} (4^{m-1} - 1) \frac{(4P)^2}{M^2} \quad (\text{B.3})$$

Adding equations (B.1), (B.2) and (B.3) together, one obtains

$$\begin{aligned} \alpha_1^2 &= \alpha_Q^2 + \alpha_I^2 + \alpha_S^2 \\ &= \frac{(4P)^2}{M^2} + \frac{m-1}{8P^2} + \frac{4}{3} (4^{m-1} - 1) \frac{(4P)^2}{M^2} \end{aligned} \quad (\text{B.4})$$

The optimal value of  $P$  is obtained by minimizing the error,  $\alpha_1$ , which gives

$$P = 0.0979 M^{\frac{1}{2}} \quad (\text{B.5})$$

for  $m=5$ . Also, the minimum error is

$$\alpha_1 = 10.22 M^{-\frac{1}{2}} \quad (\text{B.6})$$

## APPENDIX C

### SIMULATION DETAILS

The predicted RMS relative output error associated with the radix-4 DIT FFT described in this thesis, has been simulated on a Data General Nova minicomputer. Listings of two programs are given here. The first program was written in double precision floating point for arithmetic operations and also for the representations of twiddle factors. The results from this program were considered to be true results. The second program was also written in double precision floating point to simulate integer arithmetic, but the residue scaling scheme was incorporated and quantization errors were simulated.

```

C      *****PROG1*****
C      THIS PROGRAM COMPUTES THE RADIX-4 DIT FFT USING
C      DOUBLE PRECISION FLOATING-POINT ARITHMETIC
C      SUBROUTINE REQUIRED: SCRAMB, FFT4D
C      DOUBLE PRECISION A(1024),B(1024)
C      A IS THE REAL PART OF THE INPUT
C      B IS THE IMAGINARY PART OF THE INPUT
C      INTEGER NAME(6),IA(1024),IB(1024)
C      DIMENSION C(1024),D(1024)
C      COMMON /B1/ A,B
C      EQUIVALENCE (IA,C),(IB,D)
C      OBTAIN INPUT DATA FROM A PROPER FILE
C      WRITE(10,9000)
9000  FORMAT(' 3X'/'INPUT DATA FILE NAME = ',2)
      READ(11,9001) (NAME(I),I=1,6)
9001  FORMAT(6A2)
      ACCEPT 'DATA: INTEGER(2), SNGL PRECISION(4)='/,IBY1
      ACCEPT '# OF RECORDS (1024 POINTS) = ',NREC
      OPEN 1,NAME,LEN=1024*IBY1,REC=NREC
C      STORE OUTPUT DATA IN THE FILE, TRUE
      ACCEPT '# OF RECORDS TO BE PROCESSED = ',NREC
      OPEN 2, 'DP1 TRUE', ATT='C', LEN=4096, REC=NREC
      ACCEPT 'FIRST RECORD TO BE PROCESSED = ',IFIRST
      DO 500 ILOOP=1,NREC/2
      WRITE(10,100) ILOOP
100  FORMAT(1X,'RECORD = ',I4)
      ILOOP1=ILOOP+IFIRST-1
      IF (IBY1 .EQ. 4) GO TO 200
      READ(1,REC=ILOOP1) (IA(I),I=1,1024)
      READ(1,REC=ILOOP1+1) (IB(I),I=1,1024)
      DO 200 I=1,1024
      A(I)=IA(I)
200  B(I)=IB(I)
      GO TO 500
300  READ(1,REC=ILOOP1) (C(I),I=1,1024)
      READ(1,REC=ILOOP1+1) (D(I),I=1,1024)
      DO 400 I=1,1024
      A(I)=C(I)
400  B(I)=D(I)
500  CALL SCRAMB(5)
      CALL FFT4D(5,1)
      WRITE(2,REC=ILOOP) (C(I)=A(I),I=1,1024)
      WRITE(2,REC=ILOOP+1) (D(I)=B(I),I=1,1024)
600  CONTINUE
      CLOSE 1
      CLOSE 2
      STOP
      END

```

```

SUBROUTINE SCRAMB(NS)
C   THIS SUBROUTINE PERFORMS THE REQUIRED SCRAMBLING
C   PROCEDURE FOR RADIX-4 DIT OR DIF ALGORITHM
C   NS IS THE NUMBER OF STAGES
C   DOUBLE PRECISION A(1024), B(1024), R
C   A IS THE REAL PART OF THE INPUT
C   B IS THE IMAGINARY PART OF THE INPUT
COMMON /B1/ A, B
INTEGER L(7)
EQUIVALENCE (L7, L(1)), (L6, L(2)), (L5, L(3)), (L4, L(4)),
1(L3, L(5)), (L2, L(6)), (L1, L(7))
DO 70 J=1, 7
L(J)=4
IF(J-NS) 71, 70, 70
71 L(J)=4**((NS+1)-J)
70 CONTINUE
JN=1
DO 60 J2=1, L2, L1
DO 60 J3=J2, L3, L2
DO 60 J4=J3, L4, L3
DO 60 J5=J4, L5, L4
DO 60 J6=J5, L6, L5
DO 60 JR=J6, L7, L6
IF(JN .GE. JR) GO TO 62
R=A(JN)
A(JN)=A(JR)
A(JR)=R
R=B(JN)
B(JN)=B(JR)
B(JR)=R
JN1=JN+L6
JR1=JR+1
DO 10 I=1, 3
IF(JR1 .LE. JN1) GO TO 62
R=A(JN1)
A(JN1)=A(JR1)
A(JR1)=R
R=B(JN1)
B(JN1)=B(JR1)
B(JR1)=R
JN1=JN1+L6
JR1=JR1+1
10 CONTINUE
62 JN=JN+1
60 CONTINUE
RETURN
END

```

```

SUBROUTINE FFT4D(NS, ISN)
C   THIS SUBROUTINE PERFORMS THE RADIX-4 DIT FFT WITH
C   SCRAMBLED INPUT AND ORDERED OUTPUT
C   X IS THE REAL PART OF THE INPUT
C   Y IS THE IMAGINARY PART OF THE INPUT
C   NS IS THE NUMBER OF STAGES (N = 4**M)
C   ISN=1 FOR FFT; ISN=-1 FOR INVERSE FFT
DOUBLE PRECISION T1, T2, T3, T4, T5, T6, T7, T8
DOUBLE PRECISION X(1024), Y(1024)
DOUBLE PRECISION ARG, SCL, P12, C1, C2, C3, S1, S2, S3
COMMON /B1/ X, Y
N=4**NS
P12=ISN*8. D0*DATAN(1. D0)
DO 30 L=1, NS
  LE1=4** (L-1)
  LE=4*LE1
  SCL=P12/LE
  DO 20 J=1, LE1
    ARG=SCL*(J-1)
    C1=DCOS(ARG)
    S1=DSIN(ARG)
    S2=2. *S1
    C2=1. -S2*S1
    S2=C1*S2
    S3=2. *C2
    C3=C1*(S3-1. )
    S3=S1*(S3+1. )
    DO 20 I=J, N, LE
      I1=I+LE1
      I2=I1+LE1
      I3=I2+LE1
      T1=X(I1)*C1+Y(I1)*S1
      T2=Y(I1)*C1-X(I1)*S1
      T3=X(I2)*C2+Y(I2)*S2
      T4=Y(I2)*C2-X(I2)*S2
      T5=X(I3)*C3+Y(I3)*S3
      T6=Y(I3)*C3-X(I3)*S3
      T8=X(I)+T3
      T7=X(I)-T3
      T3=T1+T5
      T1=T1-T5
      X(I)=T8+T3
      X(I2)=T8-T3
      T8=Y(I)+T4
      T4=Y(I)-T4
      T3=T2+T6
      T6=T2-T6
      Y(I)=T8+T3
      Y(I2)=T8-T3
      IF (ISN .EQ. -1) GO TO 10
      X(I1)=T7+T6
      Y(I1)=T4-T1
      Y(I3)=T4+T1
      X(I3)=T7-T6

```

```

      GO TO 20
10  X(I1)=T7-T6
    Y(I1)=T4+T1
    Y(I3)=T4-T1
    X(I3)=T7+T6
20  CONTINUE
30  CONTINUE
    RETURN
    END

```

```

C      *****PROG2*****
C      THIS PROGRAM SIMULATES THE RESIDUE ARITHMETIC
C      (OR INTEGER ARITHMETIC) RADIX-4 DIT ALGORITHM
C      ASSOCIATED WITH THE SCALING SCHEME, AND ALSO
C      COMPUTES THE RELATIVE OUTPUT RMS ERROR, ALPHA1
C      SUBROUTINE REQUIRED: SCRAMB, INTFFT, ALPHA1
C      DOUBLE PRECISION A(1024), B(1024), M, K, M2
C      DOUBLE PRECISION FMAX, FAC1, THETA, P(5)
C      INTEGER IA(1024), IB(1024), NAME(6), KI(5), Q
C      DIMENSION C(1024), D(1024)
C      COMMON /B1/ A, B
C      COMMON /B2/ P, K, M2, KI
C      EQUIVALENCE (IA, C), (IB, D)
C      OBTAIN INPUT DATA FROM A PROPER FILE
C      WRITE(10, 9000)
9000  FORMAT(' ', 3X, 'INPUT DATA FILE NAME = ', Z)
      READ(11, 9001) (NAME(I), I=1, 6)
9001  FORMAT(6A2)
      ACCEPT "DATA: INTEGER(2), SNGL PRECISION(4)=", IBY1
      ACCEPT "# OF RECORDS = ", NREC
      OPEN 1, NAME, LEN=1024*IBY1, REC=NREC
C      STORE OUTPUT DATA IN THE FILE: EXPT
      ACCEPT "# OF RECORDS TO BE PROCESSED = ", NREC
      OPEN 2, "DP1:EXPT", ATT="C", LEN=4096, REC=NREC
      OPEN 3, "DP1:TRUE", LEN=4096, REC=NREC
      ACCEPT "FIRST RECORD TO BE PROCESSED = ", IFIRST
C      OBTAIN DESIRED PARAMETERS FROM FILE: DPARA
      OPEN 4, "DP1:DPARA"
C      M IS THE NUMBER RANGE OF THE RNS
C      K IS THE SCALE FACTOR
C      ADB IS THE A/D CONVERTER QUANTIZATION BITS
C      P(I) ARE INTEGER CONVERSION FACTORS
C      Q IS THE NUMBER OF SCALING OPERATIONS REQUIRED
C      TO PROVIDE SCALED OUTPUT
C      KI(I) IS THE NUMBER OF STAGES BETWEEN (I-1)-TH
C      AND I-TH SCALING OPERATIONS
1000  READ(4) M, K, ADB, (P(I), I=1, 5), Q, (KI(I), I=1, Q)
      IF(M .LT. 0) GO TO 9999
      WRITE(12, 6666) M, K, ADB, (P(I), I=1, 5)
      WRITE(12, 7777) (KI(I), I=1, Q)
6666  FORMAT(3X, 'M=', E14.7, ' K=', E14.7, ' B=', F6.2,
1' P1, P2, P3, P4, P5=', 5(2X, F9.2))
7777  FORMAT(3X, 'K1, K2, ..., KQ=', 5(2X, I2))

```

```

C      M2 IS THE MAXIMUM MAGNITUDE OF THE SIGNED RNS
      M2=M/2.D0
      FMAX=0.D0
      DO 600 ILOOP=1,NREC,2
      WRITE(10,100) ILOOP
100  FORMAT(1X,'RECORD = ',I4/)
      ILOOP1=ILOOP+IFIRST-1
      IF(1BY1.EQ. 4) GO TO 300
      READ(1,REC=ILOOP1) (IA(I),I=1,1024)
      READ(1,REC=ILOOP1+1) (IB(I),I=1,1024)
      DO 200 I=1,1024
      A(I)=IA(I)
200  B(I)=IB(I)
      GO TO 405
300  READ(1,REC=ILOOP1) (C(I),I=1,1024)
      READ(1,REC=ILOOP1+1) (D(I),I=1,1024)
      DO 400 I=1,1024
      A(I)=C(I)
400  B(I)=D(I)
405  DO 410 I=1,1024
      IF(FMAX.LT. DABS(A(I))) FMAX=DABS(A(I))
410  IF(FMAX.LT. DABS(B(I))) FMAX=DABS(B(I))
      FAC1=2.D0**((ADB-1.D0)/FMAX)
      DO 415 I=1,1024
      A(I)=A(I)*FAC1+DSIGN(0.5D0,A(I))
      IF(DABS(A(I)).LT. 32767.D0) A(I)=IDINT(A(I))
      B(I)=B(I)*FAC1+DSIGN(0.5D0,B(I))
415  IF(DABS(B(I)).LT. 32767.D0) B(I)=IDINT(B(I))
      CALL SCRAMB(5)
      CALL INTFFT(5,1)
      THETA=K**((Q-1)/(P(1)*P(2)*P(3)*P(4)*P(5)*FAC1)
      WRITE(2,REC=ILOOP) (C(I)=A(I)*THETA,I=1,1024)
600  WRITE(2,REC=ILOOP+1) (D(I)=B(I)*THETA,I=1,1024)
      CALL ALPHA1(NREC)
      GO TO 1000
9999 CLOSE 1
      CLOSE 2
      CLOSE 3
      CLOSE 4
      STOP
      END

```

```

      SUBROUTINE INTFFT(NS,ISN)
C      THIS SUBROUTINE SIMULATES THE SCALING SCHEMES
C      NS=NUMBER OF STAGES
C      ISN=1 FOR FFT, ISN=-1 FOR INVERSE FFT
C      X IS THE REAL PART OF THE INPUT
C      Y IS THE IMAGINARY PART OF THE INPUT


```



```

DOUBLE PRECISION X(1024), Y(1024), P(5), K, M2
INTEGER KI(5), P1, P2, P3, Q1, Q2, Q3
DOUBLE PRECISION T1, T2, T3, T4, T5, T6, T7, T8
DOUBLE PRECISION ARG, SCL, P12, C1, C2, C3, S1, S2, S3
COMMON /B1/ X, Y
COMMON /B2/ P, K, M2, KI
N=4**NS
PI2=8. D0/DATAN(1. D0)
KS=1
LL=1
IOVER=0
IF(ISN .EQ. 1) GO TO 100
DO 60 I=1, N
60 Y(I)=-Y(I)
100 DO 900 L=1, NS
    LE1=4**((L-1)/KS)
    LE=4*LE1
    SCL=PI2/LE
    DO 300 J=1, LE1
        IF(J .EQ. 1) GO TO 200
        ARG=SCL*(J-1)
        C1=DCOS(ARG)
        S1=DSIN(ARG)
        S2=2. *S1
        C2=1. -S2*S1
        S2=C1*S2
        S3=2. *C2
        C3=C1*(S3-1.)
        S3=S1*(S3+1.)
        P1=C1*P(L)+DSIGN(0. 500, C1)
        Q1=S1*P(L)+DSIGN(0. 500, S1)
        P2=C2*P(L)+DSIGN(0. 500, C2)
        Q2=S2*P(L)+DSIGN(0. 500, S2)
        P3=C3*P(L)+DSIGN(0. 500, C3)
        Q3=S3*P(L)+DSIGN(0. 500, S3)
200 DO 300 I=J, N, LE
        I1=I+LE1
        I2=I1+LE1
        I3=I2+LE1
        IF(L .EQ. 1 .AND. I/I1 .EQ. 1. D0) GO TO 210
        IF(J .EQ. 1) GO TO 210
        T1=X(I1)*P1+Y(I1)*Q1
        T2=Y(I1)*P1-X(I1)*Q1
        T3=X(I2)*P2+Y(I2)*Q2
        T4=Y(I2)*P2-X(I2)*Q2
        T5=X(I3)*P3+Y(I3)*Q3
        T6=Y(I3)*P3-X(I3)*Q3
        GO TO 220
210 T1=X(I1)*P(L)
    T2=Y(I1)*P(L)
    T3=X(I2)*P(L)
    T4=Y(I2)*P(L)
    T5=X(I3)*P(L)
    T6=Y(I3)*P(L)

```



```

220 X(I)=X(I)+P(L)
   Y(I)=Y(I)+P(L)
   GO TO 240
230 T1=X(I1)
   T2=Y(I1)
   T3=X(I2)
   T4=Y(I2)
   T5=X(I3)
   T6=Y(I3)
240 T8=X(I)+T3
   T7=X(I)-T3
   T3=T1+T5
   T1=T1-T5
   X(I)=T8+T3
   X(I2)=T8-T3
   T8=Y(I)+T4
   T4=Y(I)-T4
   T3=T2+T6
   T6=T2-T6
   Y(I)=T8+T3
   Y(I2)=T8-T3
   Y(I1)=T2+T6
   Y(I1)=T4-T1
   X(I3)=T7-T6
   Y(I3)=T4+T1
300 CONTINUE
C   TEST THE MAXIMUM MAGNITUDE
C   M2 IS MAXIMUM MAGNITUDE FOR SIGNED MIS
   DO 400 I=1,1924
   IF (DABS(X(I)) .GT. M2) IOWER=IOWER+1
   IF (DABS(Y(I)) .GT. M2) IOWER=IOWER+1
400 CONTINUE
   IF (L .EQ. NS) GO TO 500
   IF (L .LT. FILL) GO TO 500
   KS=KS+1
   DO 500 I=1,N
   X(I)=X(I)*K+DSIGN(M, SDB, X(I))
   IF (DABS(X(I)) .LT. 2275) DO X(I)=IDINT(X(I))
   Y(I)=Y(I)*K+DSIGN(M, SDB, Y(I))
500 IF (DABS(Y(I)) .LT. 2275) DO Y(I)=IDINT(Y(I))
   LL=1
   GO TO 600
600 LL=LL+1
900 CONTINUE
   IF (ISN .EQ. 1) GO TO 950
   DO 950 I=1,N
   Y(I)=-Y(I)
950 CONTINUE
9999 IF (IOWER .NE. 0) WRITE(10,11111) IOWER
11111 FORMAT(1X, 'WARNING: OVERFLOWS OCCUR ', I5, ' TIMES')
      RETURN
      END

```

```

SUBROUTINE ALPHA1(NREC)
C THIS SUBROUTINE COMPUTES THE RMS ERROR
  DIMENSION A(1024), B(1024)
  DOUBLE PRECISION E1, E(20), SUM(20)
  DO 55 ILOOP=1, NREC
    READ(3, REC=ILOOP) (A(J), J=1, 1024)
    READ(2, REC=ILOOP) (B(J), J=1, 1024)
    E(ILOOP)=0. D0
    SUM(ILOOP)=0. D0
    DO 40 I=1, 1024
      E(ILOOP)=(A(I)-B(I))**2+E(ILOOP)
      SUM(ILOOP)=A(I)**2+SUM(ILOOP)
40  CONTINUE
    E1=DSQRT(E(ILOOP)/SUM(ILOOP))
    WRITE(12, 50) ILOOP, E1
50  FORMAT(4X, 'RECORD = ', I4, ' RMS(ERROR)/RMS
1(RRESULT)= ', D14, 7/)
55  CONTINUE
    E1=0. D0
    DO 60 J=1, NREC, 2
      E1=DSQRT((E(J+1)+E(J))/(SUM(J)+SUM(J+1)))+E1
60  CONTINUE
    E1=E1*2. D0/(FLOAT(NREC))
    WRITE(12, 70) E1
70  FORMAT(4X, 'AVERAGE = ', D14, 7/)
    RETURN
  END

```

## APPENDIX D-

### AN ERROR ANALYSIS OF A RADIX-4 DIT FFT WITH REAL INPUT

In this appendix, we will consider the case that only the real parts of the input contain data, while the imaginary parts are filled up with zeros. For this case, the expression derived in equation (4.32) is no longer valid.

Since the imaginary parts of the input contain all zeros, there will be no A/D quantization error for them. Therefore, the error present at the input to the first stage should be halved, i.e.

$$\overline{|E_{IN}^1(n)|^2} = \overline{e_Q^2} = \frac{1}{12} \quad (D.1)$$

In the first stage of a radix-4 DIT FFT, no multiplications are required, and hence the error at the output of the first stage is governed by the 4-point DFT. For the case of complex input, equation (4.15) indicated that the output mean-square error is 4 times the input error, because there are 4 input points associated with each output point for the 4-point DFT. When the imaginary parts of the input are equal to zero, there are, in the average sense, only 2 input points associated with each output point. Thus one should have

$$\overline{|E_{OUT}^1(n)|^2} = 2 \cdot \overline{|E_{DFT}^1(n)|^2} = 2 \cdot \overline{|E_{IN}^1(n)|^2} = \frac{1}{6} \quad (D.2)$$

When we compare equation (D.2) to equation (4.15), we can see that the overall RMS error due to A/D quantization is  $4^{m-1} \cdot \frac{1}{6}$  where  $m$  is the total number of stages. Furthermore, the errors due to integer conversion and scaling rounding will be the same as those derived in equation (4.31).

Let  $\sigma_R^2$  denote the mean square value of the real input sequence, then

$$\sigma_R^2 = \frac{1}{2} \sigma^2 = \frac{1}{12} 2^{2B} \quad (D.3)$$

where  $\sigma^2$  was defined in equation (4.33). Using similar technique described in section 4.3, one obtains

$$(\alpha_Q^2)_R = \frac{1}{2} \cdot 4^{2k_1} M^{-2} P^{2(k_1-1)} \quad (D.4)$$

$$(\alpha_I^2)_R = \frac{m-1}{8P^2} \quad (D.5)$$

and

$$(\alpha_S^2)_R = 2 \cdot A \cdot 4^{2k+k_1} M^{-2} P^{2k} \quad (D.6)$$

where  $A$  was defined in equation (4.45), and the subscript  $R$  refers the term to the real input FFT, which will also be used throughout this appendix.

Combining equations (D.4), (D.5) and (D.6), one obtains the optimum functional relationships between  $M$ ,  $(P)_R$ ,  $(\alpha_1)_R$  as follows,

$$(\alpha_1)_R = 12.11 M^{-\frac{1}{2}} \quad (D.7)$$

and

$$(P)_R = 0.0825 M^{\frac{1}{2}} \quad (D.8)$$

for  $k_1=2$ ,  $k=1$  and  $N=1024$ .

If, however, we choose the parameters for a real input FFT according to the functional relationships given in Table 4.1, instead of using equations (D.7) and (D.8), then the predicted error using Table 4.1 and also an adjusting factor is still not correct. In the following, we will derive another adjusting factor,  $D_R$ , when the input contains only the real parts.

Comparing equations (D.4), (D.5) and (D.6) to equations (4.42), (4.43) and (4.44), one obtains

$$(\alpha_Q^2)_R = -\frac{1}{2} \alpha_Q^2 \quad (D.9)$$

$$(\alpha_I^2)_R = \alpha_I^2 \quad (D.10)$$

and

$$(\alpha_S^2)_R = 2 \alpha_S^2 \quad (D.11)$$

From the last column of Table 4.1, it is seen that

$$\alpha_S^2 \gg \alpha_Q^2 \quad \text{and} \quad \alpha_S^2 = \alpha_I^2 \quad (\text{D.12})$$

for  $k_1=2$ ,  $k=1$ . Therefore one should have the following relationships,

$$(\alpha_S^2)_R = 2\alpha_S^2 \gg \alpha_Q^2 \gg (\alpha_Q^2)_R$$

or

$$(\alpha_S^2)_R \gg (\alpha_Q^2)_R \quad (\text{D.13})$$

and

$$(\alpha_I^2)_R = \alpha_S^2 \quad (\text{D.14})$$

Now the adjusting factor,  $D_R$ , can be calculated by

$$\begin{aligned} D_R &= \sqrt{\frac{(\alpha_I^2)_R}{\alpha_I^2}} \approx \sqrt{\frac{(\alpha_I^2)_R + (\alpha_S^2)_R}{\alpha_I^2 + \alpha_S^2}} = \sqrt{\frac{\alpha_S^2 + 2\alpha_S^2}{2\alpha_S^2}} \\ &= \sqrt{\frac{3}{2}} = 1.22 \end{aligned}$$

It should be noted that the adjusting factor, 1.22 is correct only when  $\sigma_R^2 = \frac{1}{12} 2^{2B}$ . Otherwise, it is given by

$$D_R = 1.22 \sqrt{\frac{\frac{1}{12} 2^{2B}}{\sigma_R^2}}$$

## APPENDIX E

### AN ERROR ANALYSIS OF A ROUNDING EFFECT ON THE OUTPUT OF THE TWIDDLE FACTOR MULTIPLICATIONS

One common method of the fixed-point FFT implementations is to round the results at the output of twiddle factor multiplications to a smaller number of bits. For example, the result of an 8-bits number multiplied by another 8-bit number may be rounded to 8 bits. Since the word length of the rounded number drops very rapidly, the scaling operation, which is placed between stages to prevent overflow, becomes almost unnecessary. In fact, the worst case bit growth from the rounded output to the output of the 4-point DFT in the same stage is only 2 bits. Thus if 2 extra bits are provided to allow numbers to grow up between multiplications, the scaling operation is not required. This is the case that will be considered in the following.

Since the errors due to rounding and scaling operations are all round-off errors, we can also consider that the rounding operation after multiplication is a kind of scaling operation. Therefore there is a scaling operation at each stage except the first stage, and there are total  $m-1$  scaling operations. This scaling scheme is similar to the scheme,  $k_1=1$ ,  $k=1$  as described in Chapter 4.

The errors due to A/D quantization and integer conversion are independent of scaling schemes. Hence, from equation (4.32), one has



$$\alpha_I^2 = \frac{m-1}{8P^2} \quad (E.1)$$

and

$$\alpha_Q^2 = \frac{1}{6\sigma^2} \quad (E.2)$$

It is noted we assume  $p_2=p_3=\dots=p_m=P$ , and  $p_1=1$ .

From equation (4.29), it is seen that the mean square error due to the first scaling operation is

$$\frac{K^2}{\prod_{i=1}^m p_i^2}$$

Since the rounding operation is executed before computing the 4-point DFT, its mean square error will enlarge by 4 at the output of the 4-point DFT. Therefore the relative mean square rounding error will be in the form

$$\alpha_S^2 = \frac{\sum_{i=1}^{m-1} 4^{m-i} \left(\frac{K^2}{P^2}\right)^i}{4^m \sigma^2} \quad (E.3)$$

Adding equations (E.1), (E.2) and (E.3), one obtains the overall relative mean square error as

$$\alpha_I^2 = \frac{1}{6\sigma^2} + \frac{m-1}{8P^2} + \frac{\sum_{i=1}^{m-1} 4^{m-i} \left(\frac{K^2}{P^2}\right)^i}{4^m \sigma^2} \quad (E.4)$$

We will again assume that

$$\sigma^2 = \frac{2^{2B}}{6} \quad (E.5)$$

$$\text{and } K = 4P, \quad (E.6)$$

Furthermore, the relationship between B and M is given by

$$2^B \cdot 4P = M \quad (E.7)$$

Using equations (E.5), (E.6) and (E.7), one can rewrite equation (E.4) as

$$\begin{aligned} \alpha_1^2 &= \frac{16P^2}{M^2} + \frac{m-1}{8P^2} + 8 \cdot (4^{m-1} - 1) \frac{16P^2}{M^2} \\ &= \frac{m-1}{8P^2} + 8 \cdot (4^{m-1} - 1) \frac{16P^2}{M^2} \end{aligned} \quad (E.8)$$

such that  $\alpha_1^2$  is a function of M and P. When  $m=5$ , the optimum results are

$$P = 0.0625 M^{\frac{1}{2}} \quad (E.9)$$

$$\alpha_1 = 15.98 M^{-\frac{1}{2}} \quad (E.10)$$

Equations (E.9) and (E.10) can be used to choose optimum Parameters for the considered fixed-point FFT implementation. However the parameters M and K should be carefully interpreted,

which will be given through the following example. First we will assume that the tolerable RMS error is equal to 0.01, i.e.,  $\alpha_i = 0.01$ . Then from equation (E.10) the value of M is found to be 2553604. Because 22 bits are required to represent 2553604, we will choose M to be  $2^{22}$ . Then using equations (E.9) and (E.6), it is found that

$$K = 4P = 512 = 2^9$$

and

$$P = 2^7.$$

Finally, the required A/D quantization bit can be determined using equation (E.7), which gives

$$B = 13.$$

In this fixed-point implementation, the number of bits required to represent the number range, M, is, in fact, the number of full bit at the output of the multiplication. Since the scale factor is represented by 9 bits, the results at the output of twiddle factor multiplications are rounded to 13 (=22-9) bits.

## REFERENCES

1. N.S. Szabo and R.I. Tanaka, "Residue Arithmetic and its Application to Computer Technology", McGraw-Hill, New York, 1967.
2. R.M. Guffin, "A Computer for Solving Linear Simultaneous Equations Using the Residue Number System", IRE Trans. Electron Computers, vol. EC-11, pp.164-173, April 1962.
3. J. Huang and F. Taylor, Unpublished presentation at the First Ohio State University Workshop on Residue Arithmetic, Columbus, Ohio, May 1978.
4. W.K. Jenkins and B.J. Leon, "The Use of Residue Number Systems in the Design of Finite Impulse Response Digital Filters", IEEE Trans. Circuits and Systems, vol. CAS-24, pp.191-201, April 1977.
5. M.A. Soderstrand, "A High Speed Low-cost Recursive Digital Filter Using Residue Number Arithmetic", Proceedings of IEEE, vol.65, pp.1065-1067, July 1977.
6. G.A. Jullien, "Residue Number Scaling and Other Operations Using ROM Arrays", IEEE Trans. Computers, vol.C-27, pp. 325-337, April 1978.
7. G.A. Jullien, W.C. Miller, B.D. Tseng et al, "Hardware Realization of Digital Signal Processing Elements Using The Residue Number System", IEEE Int. Conf. Acoust, Speech, Signal Processing, Hartford, CT. May 9-11, 1977.
8. W.K. Jenkins, "Techniques for Residue-to-Analog Conversion for Residue-Encoded Digital Filters", IEEE Trans. Circuits and Systems, vol.CAS-25, pp.555-562, July 1978.
9. J.W. Cooley and J.W. Tukey, "An Algorithm for The Machine Calculation of Complex Fourier Series", Math. Computation, vol.19, pp.297-301, 1965.
10. J.W. Cooley, P.A.W. Lewis, and P.D. Welch, "Historical Notes on The Fast Fourier Transform", IEEE Trans. Audio Electroacoust., vol.AU-15, pp.76-79, June 1967.
11. B. Gold and C.M. Rader, "Digital Processing of Signals", McGraw-Hill Book Company, New York, 1969.
12. G.D. Bergland, "A Fast Fourier Transform Algorithm Using Base 8 Iterations", Math. Comp. vol.22, pp.275-278, April 1968.

13. R.C. Singleton, "An Algorithm for Computing The Mixed radix Fast Fourier Transform", IEEE Trans. Audio Electroacoust., vol.AU-17, pp.93-103, June 1969.
14. D.P. Colba and T.W. Parks, "A Prime Factor FFT Algorithm Using High-Speed Convolution", IEEE Trans. Acoust., Speech, Signal Processing, vol.ASSP-25, pp.281-294, August 1977.
15. H.F. Silverman, "An Introduction to Programming The Winograd Fourier Transform Algorithm (WFTA)", IEEE Trans. Acoust., Speech, Signal Processing, vol.ASSP-25, pp.152-165, April 1977.
16. B. Liu and A. Peled, "A New Hardware Realization of High-Speed Fast Fourier Transforms", IEEE Trans. Acoust., Speech, Signal Processing, vol.ASSP-23, pp.543-547, December 1975.
17. G.D. Bergland, "Fast Fourier Transform Hardware Implementations - An Overview", IEEE Trans. Audio Electroacoustics, vol.AU-17, pp.104-108, June 1969.
18. R.L. Veenkant, "A Serial Minded FFT", IEEE Trans. Audio Electroacoust., vol.AU-20, pp.180-185, August 1972.
19. G.D. Bergland and H.W. Hale, "Digital Real-Time Spectral Analysis", IEEE Trans. Electronic Computers, vol.EC-16, pp.180-185, April 1967.
20. T.G. Stockham, "High Speed Convolution and Correlation", 1966 Spring Joint Computer Conf., AFIPS Proc., vol.28, Washington, D.C. : Thompson, pp.229-233, 1966.
21. G.C. O'leary, "Nonrecursive Digital Filtering Using Cascade Fast Fourier Transforms", IEEE Trans. Audio Electroacoust., vol.AU-18, pp.177-183, June 1970.
22. H.L. Groginsky and G.A. Works, "A Pipeline Fast Fourier Transform", IEEE Trans. Comput., vol.C-19, pp.1015-1019, November 1970.
23. M.C. Pease, "Organization of Large Scale Fourier Processors", J. ACM, vol.16, pp.474-482, July 1969.
24. M.J. Corinthios, "A Time-Series Analyzer", Proc. Symp. Comput. Processing in Communication, vol.19, MRI Symposia Ser. New York: Polytechnic Press, 1969.
25. M.J. Corinthios, "The Design of A Class of Fast Fourier Transform Computers", IEEE Trans. Computers, vol.C-20, pp.617-623, June 1971.

26. M.J. Corinthios, "A Fast Fourier Transform for High-Speed Signal Processing", IEEE Trans. Computers, vol.C-20, pp.843-946, August 1971.
27. M.J. Corinthios, "A Parallel Radix-4 Fast Fourier Transform Computer", IEEE Trans. Computers, vol.C-24, pp.80-92, January 1975.
28. L.W. Martinson and R.J. Smith, "Digital Matched Filtering with Pipelined Floating Point Fast Fourier Transforms (FFT's)", IEEE Trans. Acoust., Speech, Signal Processing, vol.ASSP-23, pp.222-234, April 1975.
29. J.L. Flanagan, "Spectrum Analysis in Speech Coding", IEEE Trans. Audio and Electroacoust., vol.15, pp.66-69, June 1967.
30. H.D. Helms, "Fast Fourier Transform Method of Computing Difference Equations and Simulating Filters", IEEE Trans. Audio and Electroacoust., vol.15, pp.85-90, June 1967.
31. C.M. Rader, "An Improved Algorithm for High Speed Autocorrelation with Applications to Spectral Estimation", IEEE Trans. Audio and Electroacoust., vol.18, pp.439-442, December 1970.
32. P.D. Welch, "A Fixed-Point Fast Fourier Transform Error Analysis", IEEE Trans. Audio and Electroacoust., vol.17, pp.151-157, June 1969.
33. E.O. Brigham and L.R. Cecchini, "A Nomogram for Determining FFT System Dynamic Range", IEEE Int. Conf. Acoust., Speech, Signal Processing, Hartford, CT, May 9-11, 1977.
34. A. Pomerleau, M. Fournier and H.L. Buijs, "On the Design of A Real Time Modular FFT Processor", IEEE Trans. Circuits, Systems, vol.CAS-23, pp.630-633, October 1976.
35. L.R. Rabiner et al, "Terminology in Digital Signal Processing", IEEE Trans. Audio and Electroacoust., vol.AU-20, pp.322-337, December 1972.
36. B.D. Tseng and W.C. Miller, "Comments on An Introduction to Programming The Winograd Fourier Transform Algorithm (WFTA)", IEEE Trans. Acoust., Speech, Signal Processing, vol.ASSP-26, pp.268-269, June 1978.
37. C.J. Weinstein, "Roundoff Noise in Floating Point Fast Fourier Transform Computation", IEEE Trans. Audio Electroacoust., vol.AU-17, pp.209-215, September 1969.
38. G.A. Gray and G.W. Zeoli, "Quantization and Saturation Noise due to Analog-to-Digital Conversion", IEEE Trans. Aerospace and Electronic Systems, pp.222-223, Jan. 1971.

39. L.D. Enochson and R.K. Otnes, "Programming and Analysis for Digital Time Series Data", United States Department of Defense, 1968.
40. A. Baraniecka and G.A. Jullien, "On Decoding Techniques for Residue Number System Realizations of Digital Signal Processing Hardware", IEEE Trans. Circuits and Systems, vol. CAS-25, pp. 935-936, November 1978.

VITA AUCTORIS

BEN-DAU TSENG

- 1947 Born on February 15th, in Hangchow, Cheking, China.
- 1959 Completed Elementary School Education at Ancan Public School, Hsintien, Taipei, Taiwan, Republic of China.
- 1965 Completed High School Education at Chienkuo High School, Taipei, Taiwan, Republic of China.
- 1969 Graduated from the National Chiao-Tung University, Hsinchu, Taiwan, Republic of China, with the degree of B.Sc. in Electronic Engineering.
- 1973 Graduated from the University of North Dakota, Grand Forks, North Dakota, USA, with the degree of M.Sc. in Electrical Engineering.
- 1979 Candidate for the degree of Ph.D., Electrical Engineering, University of Windsor, Windsor, Ontario, Canada.