Implementation of FFT Structures Using the Residue Number System

BEN-DAU TSENG, G. A. JULLIEN, MEMBER, IEEE, AND WILLIAM C. MILLER

Abstract—**This paper considers the implementation of a fast Fourier transform (FFT) structure using arrays of read-only memories. The arithmetic operations are based entirely on the residue number system. The most important aspect of the structure relates to the scaling arrays, which are required to prevent overflow. Because of the limitations of the number system, scaling factors have to be chosen on an *a priori* basis. This paper develops optimum procedures for choosing both scaling factors and the position of scaling arrays in the structure. Some examples are presented relating to the filtering of speech via a convolutional filter structure.**

Index Terms—**Error analysis, FFT structures, high-speed filters, optimum hardware realization, residue number system (RNS), ROM arrays.**

I. INTRODUCTION

A. Problem Statement

THE fast Fourier transform (FFT) algorithm has been used to compute the discrete Fourier transform (DFT) in a number of diverse applications [1]–[3]. The hardware realizations of these FFT processors have ranged from general purpose digital computers to dedicated large scale integrated circuits [4], [5]. Recently, read-only-memory (ROM) implementations of the FFT have been considered because of their potential for high-speed parallel architecture [6], [7].

In general, an FFT cannot be implemented exactly. An integer-based realization of the FFT implemented with a finite-word length processor has two sources of quantization error associated with it. Arithmetic operations may introduce an error related to the rounding or truncation of arithmetic results, and errors also arise when FFT calculations are performed with inexact coefficients. In addition, if the FFT processor is implemented in a real-world signal processing environment, quantization errors introduced by the associated analog-to-digital converter (A/D) must be considered.

The authors are engaged in the development of an ROM oriented realization of an FFT processor using the residue number system (RNS) [7]. This type of realization offers the advantages of using a simple hardware structure of parallel arrays of ROM's to perform exact integer based arithmetic operations at high speed. However, quantization errors associated with inexact coefficients and scaling operations, among others, must still be considered.

There are quantization problems associated with the proposed RNS implementation that have not been fully treated in the literature. The most closely related papers [8], [9], deal primarily with the radix-2 FFT and consider only fixed-point arithmetic. This paper 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.

B. Residue Number System

The RNS has received varying degrees of attention from workers in the field during the last two decades. The use of the RNS is undergoing a current revival [10], owing to the present availability of LSI hardware that appears eminently suited to performing high speed operations in the parallel RNS structure. Although some current work is being directed at the problem of building a general purpose floating point arithmetic processor, using the RNS [10], a number of more immediate applications appear to lie in the direction of special purpose digital signal processing hardware. Recent, independent, investigations have discussed the RNS realization of both nonrecursive digital filters [11], and recursive filters [12], [13]. The common feature that emerges from these works, is the use of ROM's to provide parallel arrays of look-up tables for performing the arithmetic operations.

A number in the RNS is represented by the L-tuple $X = (x_0, x_1, x_2, \cdots, x_{L-1})$ where $x_i = X$ modulo m_i; this is written $x_i = |X|_{m_i}$. Binary operations of $+$ or $\cdot$, modulo M, between X and Y have the following property $Z = X \circ Y$

$$|Z|_{m_i} = z_i = |x_i \circ y_i|_{m_i} \tag{1}$$

where $\circ$, $\triangleq$, $+$, or $\cdot$.

The binary operations of $+$ and $\cdot$ within a finite integer ring (in which results of binary operations are computed modulo M), may be realized within l independent rings in which the results of operations are computed modulo m_i, where M and the $\{m_i\}$ are related by

$$M = \prod_{i=0}^{l-1} m_i. \tag{2}$$

The only proviso is that the $\{m_i\}$ be relatively prime. If the $\{m_i\}$ are made small enough, then the results of operations on all combinations of inputs can be prestored in ROM's, and, by combining a sufficient number of rings, a viable dynamic range, M, for the number systems can be generated. The RNS is much more suitable for parallel structures than

---

Manuscript received March 9, 1978; revised October 24, 1978.

The authors are with the Department of Electrical Engineering, University of Windsor, Windsor, Ont., Canada.

weighted magnitude representations (such as the binary number system), because arithmetic operations are fully independent between digits. This allows computations performed over a large dynamic range, to be realized, independently, in systems with much smaller dynamic ranges.

There have been many previous reports of high speed FFT realizations [6], [14], [15], 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 ROM's, 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 $z = (a + b) \cdot (c + d)$. Figure 1 shows the array for one of the moduli in the system. The only control function required is a latch pulse. The 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 ith 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 ROM's, is that binary operations with constants can be precalculated and stored in the ROM. This turns out to be important when designing scaling arrays [13], and leads to a significant hardware saving. 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. Thus while arrays of ROM's can perform exact integer based arithmetic operations in a high speed parallel manner, quantization errors associated with inexact coefficients and especially scaling operations must still be considered. The use of ROM arrays in implementing RNS operations has been reported in [13].

In the RNS, scaling is difficult because the digits do not convey any immediate information about the magnitude of the number. Further, the scaled number is usually not an integer and this requires some form of estimation procedure to be implemented. Two techniques have been developed using look-up tables [13]. One is to iteratively modify the number in order to use exact division to obtain either truncated or rounded results. The other is to use a summation of scaled metric vector estimates to obtain the result. In terms of hardware saving the latter is better than the former. However, the error bound in the latter depends on the number of scaling moduli and does not lend itself to a general treatment. Therefore, throughout this paper, the first scaling algorithm is considered to be used. 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. For example, scaling a 6 moduli system by the product of 3 of the moduli, requires a minimum of 33 ROM's [13]. In comparison, the multiplication of 2, 6 digit numbers only requires 6 ROM's. If X is the original number, k, the scale factor, and Y, the scaled number, the scaling process can be represented by the following rational system,

$$X \simeq \frac{\hat{X}}{D}, \qquad |\hat{X}| \leq \frac{M}{2} - 1$$

$$Y = \frac{X}{K} \simeq \frac{\frac{\hat{X}}{D}}{K} \simeq \frac{\left[\frac{\hat{X}}{K}\right]_R}{D}. \tag{3}$$

Here, $\hat{X}$ is an integer with the denominator, D, to normalize its magnitude, and $[\ ]_R$ denotes that the expression is to be taken to the closest integer. A viable scaling procedure requires that the scale factor must be predetermined (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 predetermined 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. This approach has been adopted in the subsequent error analysis.

C. Summary of Paper

As residue arithmetic is carried out in the integer number system, 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 of the FFT is shown, in this paper, to be equal to 4^k, $k = 1, 2$.

While the paper is primarily concerned with the radix-4 FFT it is shown that the mean-square value of the number growth associated with each stage of a radix-r FFT is equal to r. A theoretical worst case upper bound of number growth is derived that is data independent. An experimental study of the upper bound of number growth associated with typical input signals is also presented. A quantization error analysis of the radix-4 decimation-in-time (DIT) FFT that includes three forms of quantization error is developed in the paper. Errors due to A/D quantization, coefficient (twiddle factor) rounding and scaling quantization are explicitly covered.

A generalized expression for the relative root-mean-square (RMS) 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 error expression is determined. A design procedure for selecting optimal values is also given. Errors arising in an actual filtering example are described. The specific results are summarized in the Conclusions.

II. THE RADIX-4 FFT

A. Introduction

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

$$X(k) = \sum_{n=0}^{N-1} x(n) W_N^{nk} \qquad k = 0, 1, \cdots, N-1 \tag{4}$$

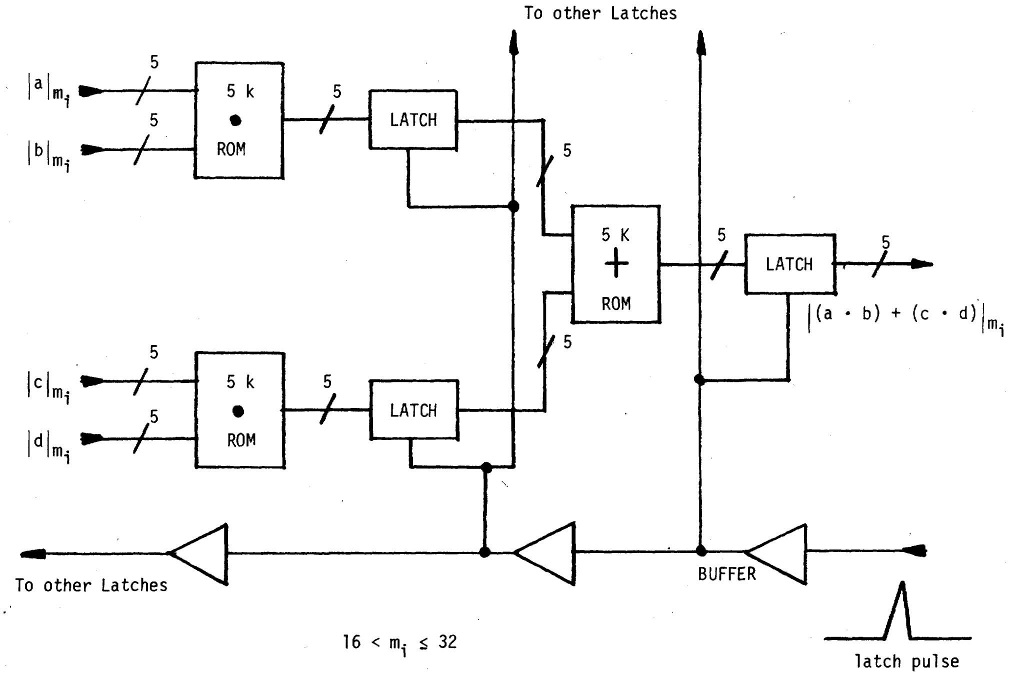


Fig. 1. Pipelined array for the function $|(a \cdot b) + (c \cdot d)|_{m_i}$.

and

$$x(n)=\frac{1}{N}\sum_{k=0}^{N-1} X(k)W_N^{-nk} \qquad n=0,1,\cdots,N-1 \qquad (5)$$

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 [16]. 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 form of the resultant r-point transform with a decimation-in-time (DIT) algorithm is given by

$$x_{i+1}(k)=\sum_{n=0}^{r-1} x_i(n)(W_N^t)^n W_r^{nk} \qquad (6)$$

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

The basic calculations of the r-point transform, as shown in (6), 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 decimation-in-frequency (DIF) algorithm. Fig. 2 shows a simplified representation of the r-point transform DIT algorithm. When the radix of the FFT is either 2 or 4, there will be no nontrivial internal coefficients requiring multiplication operations in the r-point DFT. This results from the fact that W_r^{nk} in (6) equals 0, ± 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.

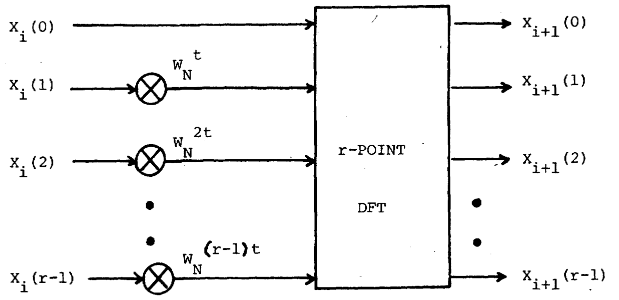


Fig. 2. Simplified representation of an r-point transform DIT algorithm.

B. Optimal Radix Considerations

In the integer number representation used in this paper, the authors consider that the arithmetic results are retained with full accuracy of each basic operation of addition, subtraction, and multiplication. When the FFT is implemented using integer number arithmetic, all noninteger coefficients, which include twiddle factors and nontrivial internal coefficients in the r-point DFT, must be normalized to integers. Since multiplication results are retained to full accuracy, the magnitudes 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

TABLE I

NUMBER OF CASCADED MULTIPLICATIONS FOR RADICES OF 2, 4, 8 AND 16

r \ N	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, ±1, ±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.

the number of scaling operations. This is especially important when the RNS is used, since scaling operations are cumbersome to implement in hardware [13]. Since the magnitudes 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.

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 nonnegative integer. Table I 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 [17], [18]. Thus, Table I 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 I 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 I, 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 basic calculation unit of a radix-16 FFT is much more complex than 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 (array, parallel iterative, cascaded, sequential) [19]. However, a radix-16 realization severely limits the viability of a processor due to the number of samples N that can be selected. Thus in this paper, radix 4 is considered as the optimal realization radix. A detailed radix-4 DIT basic calculation can be seen in Fig. 3.

C. Mean-Square Bound

Applying Parseval's theorem to (4), we have

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

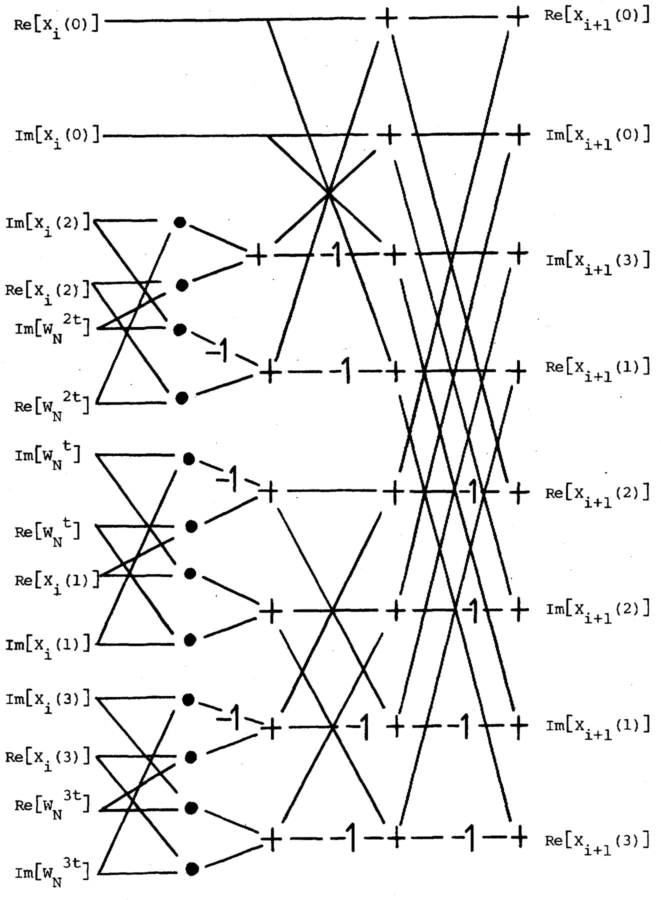


Fig. 3. Radix-4 DIT basic calculation.

or

$$\frac{1}{N}\sum_{k=0}^{N-1} |X(k)|^2 = N \cdot \frac{1}{N}\sum_{n=0}^{N-1} |x(n)|^2. \tag{7}$$

Equation (7) 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-4 FFT ($N = 4^m$), the mean-square value may increase by $(N)^{1/m}$ ($= 4$) at each stage. In fact, for a radix-r FFT, we 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 (6), one obtains

$$x_{i+1}(k) = \sum_{n=0}^{r-1} y_i(n) W_r^{nk}. \tag{8}$$

Equation (8) 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

$$\sum_{k=0}^{r-1} |x_{i+1}(k)|^2 = r \sum_{n=0}^{r-1} |y_i(n)|^2. \tag{9}$$

Since $|y_i(n)|^2 = |x_i(n)|^2 |(W_N^t)^n|^2$ and $|(W_N^t)^n|^2 = 1$, then (9) becomes

$$\sum_{k=0}^{r-1} |x_{i+1}(k)|^2 = r \sum_{n=0}^{r-1} |x_i(n)|^2. \tag{10}$$

Equation (10) can be generalized as

$$\frac{1}{N}\sum_{k=0}^{N-1} |x_{i+1}(k)|^2 = r \cdot \frac{1}{N}\sum_{n=0}^{N-1} |x_i(n)|^2. \tag{11}$$

Equation (11) 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, although the proof is not shown here.

D. 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. When $r = 4$ in (6), one obtains

$$x_{i+1}(k) = \sum_{n=0}^{3} x_i(n)(W_N^t)^n W_4^{nk}. \tag{12}$$

From (12) one can compute a worst case upper bound for the number growth at each stage. Recognizing that

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

one can rewrite (9) as

$$\begin{aligned} x_{i+1}(k) = \sum_{n=0}^{3} \Big\{ & [\mathrm{Re}\,(x_i(n)]\cos\frac{2\pi nt}{N} \\ & + \mathrm{Im}\,[x_i(n)]\sin\frac{2\pi nt}{N}\Big\} \\ & + j\Big\{\mathrm{Im}\,[x_i(n)]\cos\frac{2\pi nt}{N} \\ & - \mathrm{Re}\,[x_i(n)]\sin\frac{2\pi nt}{N}\Big\} W_4^{nk} \end{aligned} \tag{13}$$

where Re (·) and Im (·) denote the real and imaginary parts of the term 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 (13), one obtains

$$\begin{aligned} \max\{|\mathrm{Re}\,[x_{i+1}(k)]|,\ |\mathrm{Im}\,[x_{i+1}(k)]|\} & \\ \le \max\{|\mathrm{Re}\,[x_i(k)]|,\ &|\mathrm{Im}\,[x_i(n)]|\} \\ \cdot \sum_{n=0}^{3}\Big\{\Big|\cos\frac{2\pi nt}{N}\Big| &+ \Big|\sin\frac{2\pi nt}{N}\Big|\Big\} \end{aligned}$$

or

$$\frac{\max\{|\mathrm{Re}\,[x_{i+1}(k)]|,\ |\mathrm{Im}\,[x_{i+1}(k)]|\}}{\max\{|\mathrm{Re}\,[x_i(n)]|,\ |\mathrm{Im}\,[x_i(n)]|\}} \le \sum_{n=0}^{3}\Big\{\Big|\cos\frac{2\pi nt}{N}\Big| + \Big|\sin\frac{2\pi nt}{N}\Big|\Big\}. \tag{14}$$

It is seen from (14) 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. The upper bounds computed according to (14) are given in Table II for N ranging from 64 to 2048. From Table II it can be seen that the upper bound 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 is seen to increase as the number of stages increase until, for all practical purposes, a maximum value is reached. In Table II no worst case upper bound for number growth greater than 5.058 was seen to occur.

TABLE II
THEORETICAL WORST CASE UPPER BOUNDS 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	
2048	4	5.027	5.042	5.058	5.058	5.058

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.

E. Experimental Upper Bound

Besides the theoretical upper bound developed in Section D it is desirable to experimentally determine the number growth that is associated with commonly occurring input sequences, 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 digital computer. The number of samples N was chosen to be 1024. Three different sets of input sequences, including uniformly distributed pseudorandom numbers, sine waves plus pseudorandom numbers, and speech signals were used.

In one set of experiments, the inputs comprised pseudorandom 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 1 and 2, respectively, of Table III.

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

$$a(n) + 0.5\sin\frac{n\pi}{256} + 0.25\sin\frac{n\pi}{128} + 0.25\sin\frac{n\pi}{64} \tag{15}$$

where in the first case $\{a(n)\}$ are pseudorandom numbers lying in the range $-0.5 < a(n) < +0.5$ and in the second case pseudorandom numbers lying in the range $0 < a(n) < 1.0$. The experimental upper bound on number growth associated with these two cases are shown in rows 3 and 4, respectively, of Table III.

The number growth associated with typical speech waveforms was also investigated. The corresponding upper

TABLE III

EXPERIMENTAL WORST CASE UPPER BOUNDS OF THE NUMBER GROWTH AT EACH STAGE OF A RADIX-4 DIT FFT FOR VARIOUS INPUT SIGNALS

Input Type \ Stage No.	1	2	3	4	5
1	3.62	3.89	2.25	1.80	2.15
2	3.74	3.95	3.59	3.86	3.84
3	2.95	3.08	2.85	3.36	3.76
4	3.26	2.67	3.26	3.88	3.95
5	2.66	2.95	3.32	4.01	4.00

TABLE IV

DESIGN-ORIENTED RESULTS FOR VARIOUS SCALING SCHEMES; $N = 1024$

Scaling Schemes	α_1	P	Remarks
$k_1 = 5$	$11.01\ M^{-1/5}$	$0.203\ M^{1/5}$	$\alpha_S^2 = 0, \quad \alpha_Q^2 = 0.25\ \alpha_I^2$
$k_1 = 4$	$4.085\ M^{-1/4}$	$0.2\ M^{1/4}$	$\alpha_S^2 \gg \alpha_Q^2,\ \alpha_S^2 = 0.33\ \alpha_I^2$
$k_1 = 3,\ k = 1$	$4.365\ M^{-1/3}$	$0.198\ M^{1/3}$	$\alpha_Q^2 \gg \alpha_S^2,\ \alpha_Q^2 = 0.5\ \alpha_I^2$
$k_1 = 3,\ k = 2$	$3.804\ M^{-1/3}$	$0.152\ M^{1/3}$	$\alpha_Q^2 + \alpha_S^2 = 0.5\ \alpha_I^2$
$k_1 = 2,\ k = 1$	$10.18\ M^{-1/2}$	$0.0982\ M^{1/2}$	$\alpha_S^2 \gg \alpha_Q^2,\ \alpha_S^2 = \alpha_I^2$
$k_1 = 2,\ k = 2,\ k_q = 1$	$5.714\ M^{-1/3}$	$0.0491\ M^{1/3}$	$\alpha_S^2 \gg \alpha_Q^2,\ \alpha_S^2 = 0.5\ \alpha_I^2$
$k_1 = 2,\ k = 3$	$4.085\ M^{-1/4}$	$0.2\ M^{1/4}$	$\alpha_S^2 \gg \alpha_Q^2,\ \alpha_S^2 = 0.33\ \alpha_I^2$
$k_1 = 1,\ k = 1$	$10.21\ M^{-1/2}$	$0.0979\ M^{1/2}$	$\alpha_S^2 \gg \alpha_Q^2,\ \alpha_S^2 = \alpha_I^2$
$k_1 = 1,\ k = 2$	$5.554\ M^{-1/3}$	$0.156\ M^{1/3}$	$\alpha_S^2 \gg \alpha_Q^2,\ \alpha_S^2 = 0.5\ \alpha_I^2$
$k_1 = 1,\ k = 3,\ k_q = 1$	$4.628\ M^{-1/4}$	$0.141\ M^{1/4}$	$\alpha_S^2 \gg \alpha_Q^2,\ \alpha_S^2 = 0.33\ \alpha_I^2$
$k_1 = 1,\ k = 4$	$3.388\ M^{-1/5}$	$0.233\ M^{1/5}$	$\alpha_S^2 \gg \alpha_Q^2,\ \alpha_S^2 = 0.25\ \alpha_I^2$

bound on number growth is shown in the fifth row of Table III.

Each of the experiments delineated in Table III were repeated ten times. The upper bounds shown in Table III represent the worst cases that occurred within the ensemble of data generated.

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

III. ERROR ANALYSIS

A. Introduction

In this section the relative error associated with an FFT processor is derived from a consideration of errors due to A/D converter quantization noise, the integer conversion of twiddle factors and scaling. The analysis is based on the assumption that the arithmetic results are retained with full accuracy at each basic operation of addition, subtraction and multiplication. Since the magnitude of the numbers increase very rapidly with cascaded integer multiplications, a scaling operation to prevent overflow is required. In Section III-B, the statistical models for the various types of error are discussed. In Section III-C, a general error expression (46) is derived as a function of six parameters associated with the proposed structure. Section III-D, several practical assumptions and constraints are made on the parameters involved. This leads to (59), an expression that can be used as the basis for a practical design procedure. While the expressions are such that practically useful closed forms cannot be derived for many of the parameters, the essential interrelationships have been tabulated in Table IV. Finally, the manner in which the results of this section are used in the design of the proposed FFT processor, is explicitly discussed. The theoretical results have been experimentally verified with a high-speed convolution filter used to perform speech processing operations.

B. 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 an 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 [20], [21].

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

$$Q = \frac{U}{2^{B-1}}. \tag{16}$$

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 [21] to be

$$\overline{e_Q} = 0, \qquad \overline{e_Q^2} = \tfrac{1}{12}Q^2 = \tfrac{1}{12}. \tag{17}$$

Equation (17) has been considered adequate to represent converter 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 Fig. 4. Typical numerical values are also included in this figure.

In the ROM oriented implementation envisaged, the integer conversion factor p_i for the twiddle factors, must be predetermined and the integer representations of the twiddle factors stored in ROM's. The error associated with the integer conversion of a twiddle factor c is defined as

$$\begin{aligned} e_I &= \mathrm{Re}\,\{e_I\} + j\,\mathrm{Im}\,\{e_I\} \\ &= p_i c - [p_i c]_R \\ &= (p_i\,\mathrm{Re}\,\{c\} - [p_i\,\mathrm{Re}\,\{c\}]_R) \\ &\quad + j(p_i\,\mathrm{Im}\,\{c\} - [p_i\,\mathrm{Im}\,\{c\}]_R). \end{aligned} \tag{18}$$

The roundoff procedure is such that the errors $\mathrm{Re}\,\{e_I\}$ and $\mathrm{Im}\,\{e_I\}$ are uniformly distributed in the range $[-0.5, 0.5]$ and thus

$$\overline{e_I} = 0$$

and

$$\overline{|e_I|^2} = \overline{(\mathrm{Re}\,\{e_I\})^2} + \overline{(\mathrm{Im}\,\{e_I\})^2} = \tfrac{1}{6}. \tag{19}$$

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 II-D allows one to choose a scale factor capable of constraining the number growth to a desired value.

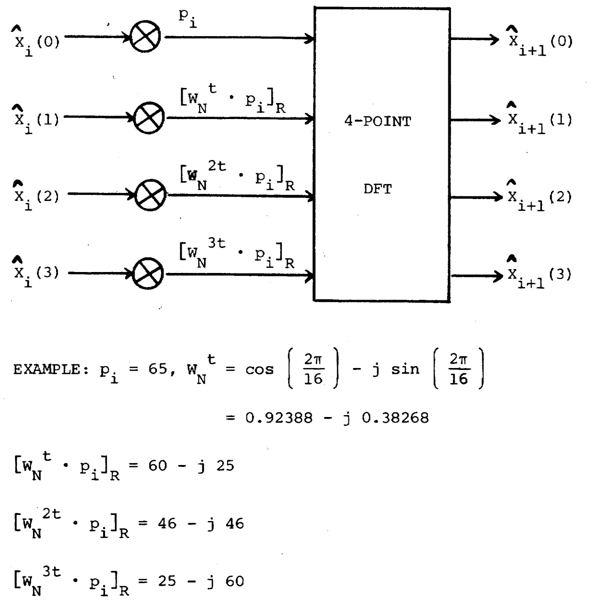


Fig. 4. Integer representation of a 4-point transform DIT algorithm.

The errors associated with a scaling operation are either roundoff or trunction 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 [13]. The errors due to the latter depend upon the specific moduli. Throughout this paper, we will assume that the former is used.[1] The output from this scaling procedure is

$$Y = \left[\frac{X}{K}\right]_R \tag{20}$$

where X is the input and 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

$$\begin{aligned} e_S &= \mathrm{Re}\,\{e_S\} + j\,\mathrm{Im}\,\{e_S\} \\ &= \frac{V}{K} - \left[\frac{V}{K}\right]_R \\ &= \left(\frac{\mathrm{Re}\,\{V\}}{K} - \left[\frac{\mathrm{Re}\,\{V\}}{K}\right]_R\right) \\ &\quad + j\left(\frac{\mathrm{Im}\,\{V\}}{K} - \left[\frac{\mathrm{Im}\,\{V\}}{K}\right]_R\right). \end{aligned} \tag{21}$$

The same roundoff procedure is used here as we used in (18), hence, $\mathrm{Re}\,\{e_S\}$ and $\mathrm{Im}\,\{e_S\}$ have the same probability distribution as $\mathrm{Re}\,\{e_I\}$ or $\mathrm{Im}\,\{e_I\}$ and the mean and variance are given by

$$\overline{e_S} = 0 \quad \text{and} \quad \overline{|e_S|^2} = \tfrac{1}{6}, \tag{22}$$

respectively.

C. *rms Quantization Error Analysis*

In this section an expression for the relative rms error α_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 α_1 is given by

---

[1] The scaling algorithm generates a zero mean error by the addition of one half the scale factor to the input.

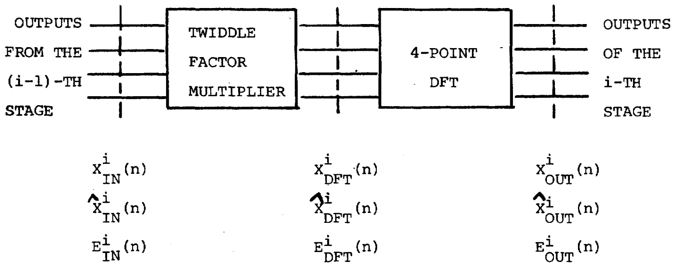


Fig. 5. Basic module of the *i*th stage of a radix-4 FFT.

$$\alpha_1 = \left(\frac{\sum_k \{\mathrm{Re}\,[X(k)] - \mathrm{Re}\,[\hat{X}(k)]\}^2 + \{\mathrm{Im}\,[X(k)] - \mathrm{Im}\,[\hat{X}(k)]\}^2}{\sum_k \{\mathrm{Re}\,[X(k)]\}^2 + \{\mathrm{Im}\,[X(k)]\}^2} \right)^{1/2} \tag{23}$$

The rms error α_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 FFT processor consists of $N/4$ basic modules that have the form shown in Fig. 5. 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 Fig. 5. A superscript indicates the stage number while the subscript indicates the appropriate input or output. The symbols $\{x^i_{\mathrm{in}}(n)\}$ and $\{x^i_{\mathrm{DFT}}(n)\}$ 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 ith stage of the FFT processor, respectively, whereas, $\{\hat{x}^i_{\mathrm{in}}(n)\}$ and $\{\hat{x}^i_{\mathrm{DFT}}(n)\}$ are the scaled integer representations of the corresponding computed value. The error between two points is given by

$$E^i_{\mathrm{in}}(n) = x^i_{\mathrm{in}}(n) - \theta \hat{x}^i_{\mathrm{in}}(n) \tag{24}$$

or

$$E^i_{\mathrm{DFT}}(n) = x^i_{\mathrm{DFT}}(n) - \frac{\theta}{p_i}\, \hat{x}^i_{\mathrm{DFT}}(n) \tag{25}$$

where θ 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

$$\sigma^2 = \overline{|x^1_{\mathrm{in}}(n)|^2}. \tag{26}$$

The error present at the input to the first stage are due to A/D converter quantization errors and thus from (17) and (24) one has

$$\overline{E^1_{\mathrm{in}}(n)} = 0 \qquad \text{and} \qquad \overline{|E^1_{\mathrm{in}}(n)|^2} = 2\overline{e_Q^2} = \tfrac{1}{6}, \tag{27}$$

since θ 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

$$E^1_{\mathrm{DFT}}(n) = E^1_{\mathrm{in}}(n). \tag{28}$$

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

$$\overline{E^1_{\mathrm{out}}(n)} = \overline{E^1_{\mathrm{DFT}}(n)} = \overline{E^1_{\mathrm{in}}(n)}$$

and

$$\overline{|E^1_{\mathrm{out}}(n)|^2} = 4 \cdot \overline{|E^1_{\mathrm{DFT}}(n)|^2} = 4 \cdot \overline{|E^1_{\mathrm{in}}(n)|^2} = 4 \cdot \tfrac{1}{6}. \tag{29}$$

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 (25), one obtains

$$\begin{aligned} E^2_{\mathrm{DFT}}(n) &= x^2_{\mathrm{DFT}}(n) - \frac{\theta}{p_2}\, \hat{x}^2_{\mathrm{DFT}}(n) \\ &= x^2_{\mathrm{in}}(n)c - \frac{1}{p_2}\, \hat{x}^2_{\mathrm{in}}(n)[p_2 c]_R \end{aligned} \tag{30}$$

where $\theta = 1/p_1 = 1$. Since

$$E^1_{\mathrm{out}}(n) = x^1_{\mathrm{out}}(n) - \hat{x}^1_{\mathrm{out}}(n) = x^2_{\mathrm{in}}(n) - \hat{x}^2_{\mathrm{in}}(n)$$

one can rewrite (30) as

$$\begin{aligned} E^2_{\mathrm{DFT}}(n) = x^1_{\mathrm{out}}(n)c &- \frac{1}{p_2} \\ &\cdot [x^1_{\mathrm{out}}(n) - E^1_{\mathrm{out}}(n)][p_2 c]_R. \end{aligned} \tag{31}$$

Using (18) one can write (31) as

$$E^2_{\mathrm{DFT}}(n) = E^1_{\mathrm{out}}(n)\left(\frac{e_I}{p_2} - c\right) - x^1_{\mathrm{out}}(n)\left(\frac{e_I}{p_2}\right). \tag{32}$$

Thus,

$$\overline{E^2_{\text{DFT}}(n)} = 0 \tag{33}$$

and

$$\begin{aligned}\overline{|E^2_{\text{DFT}}(n)|^2} &= \overline{|E^1_{\text{out}}(n)|^2}\left(\frac{\overline{|e_I|^2}}{p_2^2} + 1\right) \\ &\quad + \overline{|x^1_{\text{out}}(n)|^2} \cdot \frac{\overline{|e_I|^2}}{p_2^2} \\ &\simeq \overline{|E^1_{\text{out}}(n)|^2} + \overline{|x^1_{\text{out}}(n)|^2} \cdot \frac{\overline{|e_I|^2}}{p_2^2} \\ &= 4 \cdot 2\overline{e_Q^2} + 4 \cdot \overline{|x^1_{\text{in}}(n)|^2} \cdot \frac{1}{6} \cdot \frac{1}{p_2^2} \\ &= 4 \cdot \frac{1}{6} + \frac{2}{3}\frac{1}{p_2^2}\,\sigma^2. \end{aligned} \tag{34}$$

However, since only 3 integer conversions associated with the twiddle factors contribute roundoff error as shown in Fig. 4, (34) can be written as

$$\begin{aligned}\overline{|E^2_{\text{DFT}}(n)|^2} &= 4 \cdot \frac{1}{6} + \frac{3}{4} \cdot \frac{2}{3}\frac{\sigma^2}{p_2^2} \\ &= 4 \cdot \frac{1}{6} + \frac{1}{2}\frac{\sigma^2}{p_2^2} \end{aligned} \tag{35}$$

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

$$\overline{E^2_{\text{out}}(n)} = 0 \tag{36}$$

and

$$\overline{|E^2_{\text{out}}(n)|^2} = 4\overline{|E^2_{\text{DFT}}(n)|^2} = 4^2 \cdot \frac{1}{6} + 2\frac{\sigma^2}{p_2^2}. \tag{37}$$

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

$$\overline{|E^3_{\text{out}}(n)|^2} = 4^3 \cdot \frac{1}{6} + 4 \cdot 2\left(\frac{1}{p_2^2} + \frac{1}{p_3^2}\right)\sigma^2. \tag{38}$$

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

$$\overline{|E^k_{\text{out}}(n)|^2} = \begin{cases} 4 \cdot \frac{1}{6}, & k = 1 \\ 4^k \cdot \frac{1}{6} + 4^{k-2} \cdot 2\left(\sum\limits_{i=2}^{k} \frac{1}{p_i^2}\right)\sigma^2, & k \geq 2. \end{cases} \tag{39}$$

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 rms scaling algorithm [13] 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 $(i-1)$th and ith scaling operation. Then

$$m = \sum_{i=1}^{q} k_i \tag{40}$$

where q is the number of scaling operations required to produce a scaled output at the mth 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.

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

$$E_{s_1}(n) = x^{k_1}_{\text{out}}(n) - \frac{K}{\prod\limits_{i=1}^{k_1} p_i} \cdot \left|\frac{x^{k_1}_{\text{out}}(n)}{K}\right|_R. \tag{41}$$

Substituting (21) into (41) and using the relationship between $x^{k_1}_{\text{out}}(n)$ and $\hat{x}^{k_1}_{\text{out}}(n)$, one obtains

$$E_{s_1}(n) = E^{k_1}_{\text{out}}(n) - \frac{Ke_s}{\prod\limits_{i=1}^{k_1} p_i}. \tag{42}$$

Then

$$\overline{|E_{s_1}(n)|^2} = \overline{|E^{k_1}_{\text{out}}(n)|^2} + \frac{K^2}{\prod\limits_{i=1}^{k_1} p_i^2} \cdot \frac{1}{6}. \tag{43}$$

In the above equation, the first term is the mean-square error generated before the first scaling operation, as given in (39), while the second term is the mean-square error due to the first scaling operation.

At the output of the $(k_1 + k_2)$th stage (just before the second scaling operation), the first term in (43) 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

$$\begin{aligned}\overline{|E^{k_1+k_2}_{\text{out}}(n)|^2} &= 4^{k_1+k_2} \cdot \tfrac{1}{6} + 4^{k_1+k_2-2} \\ &\quad \cdot 2\left\{\sum_{i=2}^{k_1+k_2} \frac{1}{p_i^2}\right\} \\ &\quad + 4^{k_2} \cdot \frac{K^2}{\prod\limits_{i=1}^{k_1} p_i^2} \cdot \frac{1}{6}. \end{aligned} \tag{44}$$

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

$$\begin{aligned}\overline{|E^m_{\text{out}}(n)|^2} &= 4^m \cdot \frac{1}{6} + 4^{m-2} \cdot 2\left\{\sum_{i=2}^{m} p_i^{-2}\right\}\sigma^2 \\ &\quad + \frac{K^{2q}}{6}\sum_{i=1}^{q-1}\left\{4^{\sum_{l=1}^{i} K_{q-l+1}} \cdot K^{-2i} \cdot \prod_{l=1}^{\sum_{l=1}^{q-1} k_l} p_l^{-2}\right\}. \end{aligned} \tag{45}$$

Then using (45), (26), and (7), one can obtain the total rms relative error,

$$\alpha_1 = \frac{\text{rms (error)}}{\text{rms (true result)}} = \left\{\frac{\overline{|E_{\text{out}}^m(n)|^2}}{4^m\sigma^2}\right\}^{1/2}$$
$$= \left\{\frac{1}{6\sigma^2} + \frac{1}{8}\sum_{i=2}^{m} p_i^{-2} + \frac{K^{2q}}{6\cdot 4^m\sigma^2}\right.$$
$$\left.\cdot \sum_{i=1}^{q-1}\left[4^{\sum_{l=1}^{i} K_{q-l+1}} \cdot K^{-2i} \prod_{l=1}^{\sum_{t=1}^{q-i} k_t} p_l^{-2}\right]\right\}^{1/2}. \tag{46}$$

The relative rms error, α_1, expressed in (46) is seen to be a function of the following:

a) σ^2, the mean-square value of the true 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 ith 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.

D. Design-Oriented Results

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

While (46) is a general expression it does not explicitly introduce the range of the integer number system being used or the upper bounds on number growth analyzed in Sections II-D and II-E. In addition, if (46) 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 (46) 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 $x_{\text{in}}^1(n)$ is assumed to be a uniformly distributed complex random 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 (26) becomes

$$\sigma^2 = \tfrac{1}{6}2^{2B}. \tag{47}$$

Second, from Table III, 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 $i = 2, 3, \cdots, m$ and $p_1 = 1$.

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 operation 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

$$k_1 \le m, \qquad k_2 = k_3 = \cdots = k_q = k \tag{48}$$

and the second class by

$$k_1 \ll m, \qquad k_2 = k_3 \cdots = k_{q-1} = k$$

and

$$1 \le k_q < k. \tag{49}$$

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

$$K = (4p)^k. \tag{50}$$

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

$$M = 4^{k_1}p^{k_1 - 1}2^B. \tag{51}$$

Equation (51) can be rewritten as

$$B = \log_2\left(\frac{M}{4^{k_1}p^{k_1-1}}\right) \tag{52}$$

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 $1/6\sigma^2$ in (42) be represented by α_Q^2. Similarly, let

$$\frac{1}{8}\sum_{i=2}^{m} p_i^{-2} = \alpha_I^2$$

and the last term equal α_S^2. Thus,

$$\alpha_1^2 = \alpha_Q^2 + \alpha_I^2 + \alpha_S^2. \tag{53}$$

These three terms will now be expressed in terms of the appropriate parameters.

Using (47) the relative error due to A/D converter quantization is given by

$$\alpha_Q^2 = \frac{1}{2^{2B}}. \tag{54}$$

Using (51) one can write

$$\alpha_Q^2 = 4^{2k_1}M^{-2}p^{2(k_1-1)}. \tag{55}$$

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

$$\alpha_I^2 = \frac{m-1}{8}\frac{1}{p^2}. \tag{56}$$

Using (48), (49), and (51), α_S^2 can be written as

$$\alpha_S^2 = A \cdot 4^{2k+k_1} \cdot M^{-2} p^{2k} \tag{57}$$

where

$$A = \begin{cases} \dfrac{4^{m-k_1} - 1}{4^k - 1}, & k_q = k \\[2ex] \dfrac{4^{m-k_1-k_q+k} - 1}{4^k - 1}, & k_q < k \end{cases} \tag{58}$$

takes into account the two general classes of scaling schemes.

One can write (53) as

$$\alpha_1^2 = 4^{2k_1} M^{-2} p^{2(k_1-1)} + \frac{m-1}{8} \frac{1}{p^2} + A \cdot 4^{2k+k_1} \cdot M^{-2} p^{2k}. \tag{59}$$

Thus the relative mean-square error α_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 (59) with respect to P, and equating the results to zero, one obtains

$$2(k_1 - 1) \cdot 4^{2k_1} \cdot M^{-2} p^{2k_1-3} - \frac{m-1}{4} p^{-3} + 2k \cdot A \cdot 4^{2k+k_1} \cdot M^{-2} p^{2k-1} = 0. \tag{60}$$

Equation (60) can be used to determine the value of P, expressed as a function of M, that minimizes the relative error α_1 for a given scaling scheme. The results cannot 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 values 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 IV. For example, if an acceptable level of relative error α_1 is equal to 0.01 and scaling scheme $k_1 = 3$ and $k = 1$ is used, then from Table IV the corresponding value of M is given by the relationship

$$\alpha_1 = 4.365 M^{-1/3}$$

and thus M is found to be equal to 0.832×10^8. In a similar manner the corresponding value of P is given by

$$P = 0.198 M^{1/3}$$

or $P = 0.864 \times 10^2$. Using (50) the scale factor K is found to be equal to 0.346×10^3, and using (52), the A/D converter accuracy, B, is found to be 7.44 bits. The values of M, P, K, and B determined are then rounded to the closest integer number to give the value that may be actually used in the practical realization.

An FFT processor was simulated on a general purpose digital computer in a manner that satisfied all the assumptions used in the development of (59). Simulation results were used to establish a relationship between the error α_1 and the range of the number system M being used. These results are shown in Fig. 6. The corresponding results predicted by the theoretically derived relationships tabulated in Table IV have also been plotted in Fig. 6. The small error between the theoretical and the simulation results shown in Fig. 6 indicates that (59) and the subsequent relationships tabulated in Table IV can be used with confidence in the practical design of an FFT processor.

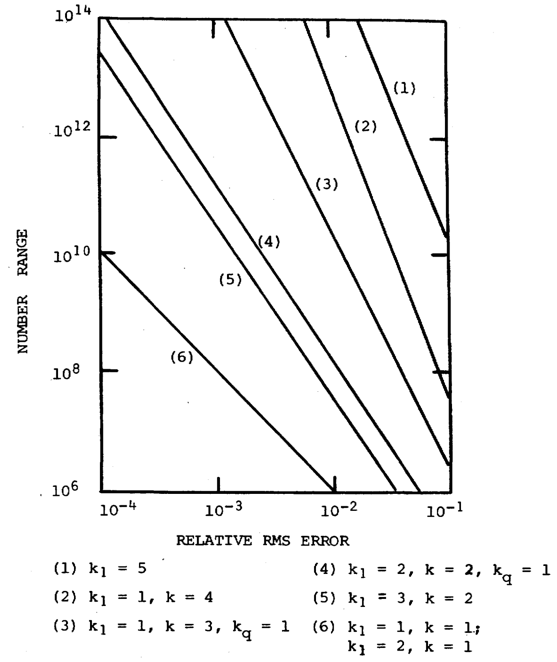


Fig. 6. Relative rms error versus number range for different scaling schemes; $N = 1024$.

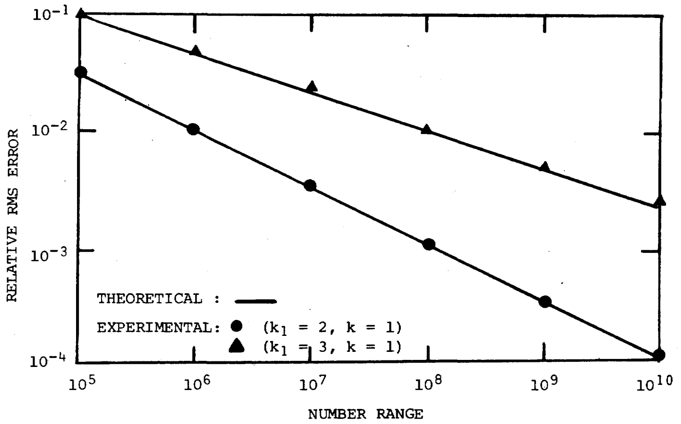


Fig. 7. Relative rms error versus number range for two scaling schemes; $M = 10^7$, $N = 1024$.

Since the scaling scheme must be specified to use Table IV, it is desirable to determine which scaling scheme will generate the smallest relative error for a given value of M. Fig. 7 shows plots of M versus α_1 for various scaling schemes. The values plotted were computed using the functional relationships between M and α_1 for the various scaling schemes shown in Table IV. In Fig. 7 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

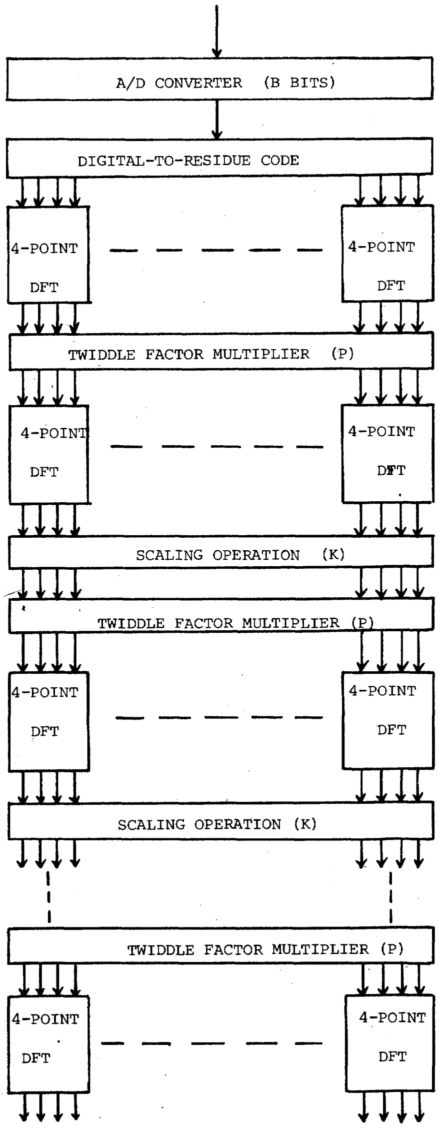


Fig. 8. Conceptual block diagram of a radix-4 FFT realization.

schemes $k_1 = 2$ and $k = 1$ 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.

The results developed in this section can be used as the basis of a design procedure for an FFT processor. The scaling scheme $k_1 = 2$, $k = 1$, is assumed to be used. Now, the design procedure is given as follows:

1) Specify the level of relative error α_1 that is acceptable for the processor.

2) Compute the required range of the number system M using the relationship $\alpha_1 = 10.18M^{-1/2}$ given in Table IV.

3) Select a set of relatively prime moduli which gives a product of all moduli M_0 close to the computed M from 2).

4) Compute the integer conversion constant P using the relationship $P = 0.0982M^{1/2}$ given in Table IV.

5) From the chosen set of moduli in 3), select some of them to obtain a scale factor K_0 close to $4P$. The integer conversion constant for the third and the following stages is equal to $[K_0/4]_R$.

TABLE V
TOTAL NUMBER OF LOOK-UP TABLES FOR VARIOUS RMS ERRORS

$\{m_i\}$	K	α_1	SEQUENTIAL REALIZATION	CASCADE REALIZATION
32, 31, 29 13, 3	31, 13	0.01	266	1048
32, 31, 27 23, 7	31, 27	0.005	266	1048
31, 19, 17 13, 11, 7	17, 13, 11	0.001	372	1416
THROUGHPUT RATE (τ IS THE ACCESS PLUS LATCH TIME)			$\frac{8}{5\tau}$	$\frac{8}{\tau}$

6) Substitute M_0 and P into (47a) 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 second stage 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. A conceptual block diagram of the implementation is shown in Fig. 8.

For completeness, the total number of look-up tables for the radix-4 1024-point FFT at various rms errors is given in Table V. The scaling scheme $k_1 = 2$, $k = 1$ is used. Referring to Fig. 3, it is seen that there are 34 operations in the radix-4 basic calculation and 16 operations in the 4-point DFT. Furthermore, as discussed in [13], the total number of look-up tables required for the scaling array is

$$T_s = (L-1)\left(S + \frac{L}{2}\right) - S^2$$

where L is the total number of moduli and S is the number of scaling moduli. The values given in Table V are calculated using the following formula.

Sequential Realization:

$$\text{Number of tables} = 34L + 8T_s.$$

Cascade Realization:

$$\text{Number of tables} = 16L + (m-1) \times 34L + 8(q-1)T_s$$

where $m = 5$ and $q = 4$ for $N = 1024$ and $k_1 = 2$, $k = 1$. It should be noted that the number of look-up tables calculated do not take into consideration the size of ROM for different moduli. When the FFT is implemented in a pipelined manner, the data throughput rate is a more important parameter than number of look-up cycles (latency time). The throughput rates for two different realizations are given in Table V.

E. High-Speed Convolution Filter

In this section the analysis of a finite impulse response (FIR) digital filter realized using the RNS is considered. When an FIR digital filter is implemented using the FFT, there is always an inverse FFT (IFFT) to be computed. Hence another set of hardware to compute the IFFT is also required. If we can tolerate a factor of 2 reduction in data throughput, then the same hardware can be used for both forward and inverse transforms. In order to utilize the single hardware transform effectively, the magnitude of the input to the IFFT must be equal to the magnitude of the original input such that there will be no overflow. This can be achieved by properly choosing the gain G_F of the filter. Fig. 9 shows the block diagram of a high-speed convolution filter.

We can show that the rms relative error of the input to the IFFT is

$$\alpha_Q'^2 = \alpha_1^2 + \frac{G_S^2}{6N\sigma_0^2} + \frac{1}{6G_F^2} + \frac{G_S^2K^2}{6\beta N\sigma_0^2 G_F^2} \tag{61}$$

where β is the ratio of the passband 3 dB cutoff frequency to the folding frequency and

$$G_S = \frac{K^q}{\prod_{i=1}^{m} p_i}.$$

From the previous analysis, we can easily obtain the overall ratio

$$\begin{aligned}\alpha_2^2 &= \frac{G_S^2K^2}{\beta N G_F^2}\alpha_S^2 + \alpha_I^2 + \alpha_Q'^2\\ &= \left(\frac{G_S^2K^2}{\beta N G_F^2} + 1\right)\alpha_S^2 + 2\alpha_I^2\\ &\quad + \left(\frac{G_S^2K^2}{\beta N G_F^2} + \frac{G_S^2}{N} + 1\right)\alpha_Q^2\\ &\quad + \frac{1}{6G_F^2}.\end{aligned} \tag{62}$$

The simplifications made in Section III-D will also be used here to simplify the expression of α_2^2. Let $N = 1024$ and we will apply scaling scheme $k_1 = 2$, $k = 1$. When the numbers are multiplied by the filter coefficients, their magnitudes can increase to an upper bound of $1.4G_F$. Because of the pessimistic bound for the magnitude at the output of the first FFT, we have found, empirically, that the upper bound on the output of the filter can be set to G_F. Then the value of G_F can be determined using the following equation:

$$\frac{\left(\frac{M}{2}\right)\cdot G_F}{K^2} = 2^{B-1}. \tag{63}$$

Substituting (51) and $K = 4P$ into (63), we find that

$$G_F = P. \tag{64}$$

The optimum results for this case are

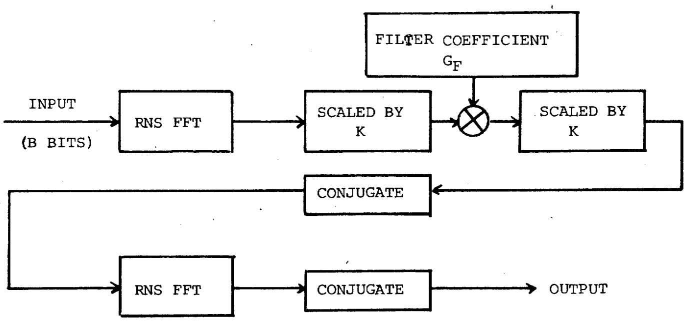


Fig. 9. Block diagram of a high-speed convolution filter.

$$\begin{aligned}\alpha_2 &= 84.66\beta^{-1/4}M^{-1/2}\\ p &= 0.018\beta^{1/4}M^{1/2}.\end{aligned} \tag{65}$$

As we compare the results of (65) to the results in Table V, we can see that the optimum results for the high-speed convolution filter may not be the same as the results for the single FFT. Thus, as far as the isolated FFT is concerned, our results for the FFT and IFFT combination are suboptimal. This tradeoff will always exist.

An experiment for subjective error analysis by filtering speech through a wide-band filter and comparing the input to the output, for quality of reproduction, was performed. An interesting result from this subjective analysis is that there is a direct correlation between the subjective quality of the output speech and the magnitude of the theoretical error. As a datum for this subjective error analysis the authors have found the following scheme yields wide-band ($\beta = 0.5$) filtered speech signals which are audibly indistinguishable from the original input: $k_1 = 2$ and $k = 1$; $M = 10^7$.

Using Fig. 10 a theoretical error of 3.2 percent is obtained; this error can serve as a guide for useful filter implementations. Two RNS realizations have been found which have errors below or about the same value as this error.

RNS Realization 1:

$M = 31 \times 16 \times 15 \times 13 \times 11 \times 7 = 7447440$

$K = 15 \times 11 = 165$

$\alpha_1 = 3.7$ percent

RNS Realization 2:

$M = 31 \times 29 \times 16 \times 15 \times 13 \times 11 = 30853680$

$K = 31 \times 11 = 341$

$\alpha_1 = 1.8$ percent

Both of these realizations are 6 moduli systems with two moduli used for the scale factor. The look-up tables required for these realizations consist of a mixture of 5K (8K in practice) for the moduli 31 and 29 and 1K for the other moduli [13]. In order to compute a data rate for filtering, we note that for a 1024-point radix-4 transform, 5 stages are required for both the FFT and the IFFT, hence 10 stages are required for filtering. The data rate is therefore given by

$$\frac{4 \times 2}{10\tau_L} = 8 \text{ MHz}$$

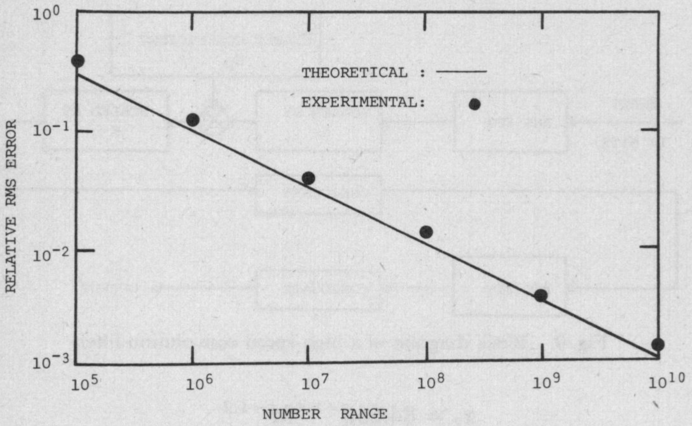


Fig. 10. Relative rms error versus number range for the FFT convolution filter on speech signal; $\beta = 0.5$; $M = 10^7$; $k_1 = 2$, $k = 1$, $N = 1024$.

for a sequential realization where $\tau_L = 100$ ns. This, of course, is a very optimistic data rate since we have not taken into account the nonzero size of the convolution kernel (which will reduce the data rate by a factor $(N - h)/N$ where h is the kernel size) and the auxiliary control equipment and random access memories which may slow down the array.

IV. CONCLUSIONS

This paper has considered a quantization error analysis of an RNS based FFT processor. It was found that a radix-4 FFT structure minimized the number of cascaded multiplication and was the most viable choice. A theoretical and experimental study of number growth in the 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 the relative mean square 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 paper. This procedure allows one to specify the relative output error and then determine the appropriate values of the number system range, the number of A/D converter quantization bits and the scale and integer conversion factors.

The analysis concepts developed in this paper were applied to the design of a high speed convolution filter implemented with the RNS. The theoretical results predicted for the filter were experimentally verified in a speech processing application. The design-oriented expressions given in the papers proved quite useful in a practical signal processing environment.

REFERENCES

[1] J. L. Flanagan, "Spectrum analysis in speech coding," *IEEE Trans. Audio Electroacoust.*, vol. Au-15, pp. 66–69, June 1967.

[2] H. D. Helms, "Fast Fourier transform method of computing difference equations and simulating filters," *IEEE Trans. Audio Electroacoust.*, vol. Au-15, pp. 85–90, June 1967.

[3] C. M. Rader, "An improved algorithm for high speed autocorrelation with applications to spectral estimation," *IEEE Trans. Audio Electroacoust.*, vol. Au-18, pp. 439–442, Dec. 1970.

[4] G. D. Bergland, "Fast Fourier transform hardware implementations—A survey," *IEEE Trans. Audio Electroacoust.*, vol. Au-17, pp. 109–119, June 1969.

[5] G. D. Bergland and D. E. Wilson, "A fast Fourier transform algorithm for a global highly parallel processor," *IEEE Trans. Audio Electroacoust.*, vol. Au-17, pp. 125–127, June 1969.

[6] B. Liu and A. Peled, "A new hardware realization of high-speed fast Fourier transform," *IEEE Trans. Acoust., Speech, Signal Processing*, vol. ASSP-23, pp. 543–547, Dec. 1975.

[7] G. A. Jullien *et al.*, "Hardware realization of digital signal processing elements using the residue number system," presented at the IEEE Int. Conf. Acoust., Speech, Signal Processing, Hartford, CT, May 9–11, 1977.

[8] P. D. Welch, "A fixed-point fast Fourier transform error analysis," *IEEE Trans. Audio Electroacoust.*, vol. AU-17, pp. 151–157, June 1969.

[9] E. O. Brigham and L. R. Cecchini, "A Nomogram for Determining FFT System Dynamic Range," presented at the IEEE Int. Conf. Acoust., Speech, Signal Processing, Hartford, CT, May 9–11, 1977.

[10] J. Huang and F. Taylor, Unpublished presentation at the 1st Ohio State University Workshop on Residue Arithmetic, Columbus, OH, May 1978.

[11] W. K. Jenkins and B. J. Leon, "The use of residue number systems in the design of finite impulse response digital filters," *IEEE Trans. Circuit Syst.*, vol. CAS-24, pp. 191–201, Apr. 1977.

[12] M. A. Soderstrand, "A high speed low-cost recursive digital filter using residue number arithmetic," in *Proc. IEEE*, vol. 65, pp. 1065–1067, July 1977.

[13] G. A. Jullien, "Residue number scaling and other operations using ROM arrays," *IEEE Trans. Comput.*, vol. C-27, pp. 325–337, Apr. 1978.

[14] M. J. Corinthios *et al.*, "A parallel radix-4 fast Fourier transform computer," *IEEE Trans. Comput.*, vol. C-24, Jan. 1975.

[15] A. Pomerleau *et al.*, "On the design of a real time modular FFT processor," *IEEE Trans. Circuit Syst.*, vol. CAS-23, pp. 630–633, Oct. 1976.

[16] L. R. Rabiner *et al.*, "Terminology in digital signal processing," *IEEE Trans. Audio Electroacoust.*, vol. AU-20, pp. 322–337, Dec. 1972.

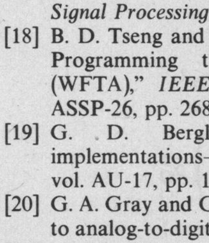
[17] H. F. Silverman, "An introduction to programming the Winograd Fourier transform algorithm (WFTA)," *IEEE Trans. Acoust., Speech, Signal Processing*, vol. ASSP-25, pp. 152–165, Apr. 1977.

[18] 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.

[19] G. D. Bergland, "Fast Fourier transform hardware implementations—An overview," *IEEE Trans. Audio Electroacoust.*, vol. AU-17, pp. 104–108, June 1969.

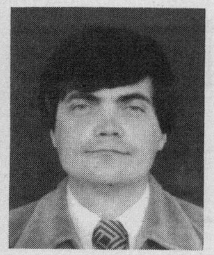
[20] G. A. Gray and G. W. Zeoli, "Quantization and saturation noise due to analog-to-digital conversion," *IEEE Trans. Aerosp. Electron. Syst.*, pp. 222–223, Jan. 1971.

[21] L. D. Enochson and R. K. Otnes, "Programming and analysis for digital time series data," US Dep. of Defense, 1968.



Ben-Dau Tseng was born in Hangchow, China. He received the B.S. degree in electronic engineering from the National Chiao-Tung University, Hsinchu, Taiwan, in 1969, and the M.S. degree in electrical engineering from the University of North Dakota, Grand Forks, in 1973.

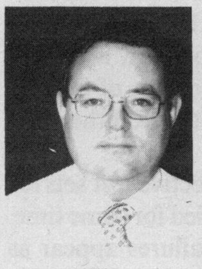
He is currently working towards the Ph.D. degree at the University of Windsor, Windsor, Ont. His interests are in digital signal processing and spectral analysis.



G. A. Jullien (M'70) was born in Wolverhampton, England on June 16, 1943. He received the B.Tech. degree from Loughborough University of Technology, Loughborough, in 1965, the M.Sc. degree from the University of Birmingham, Birmingham, in 1967 and the Ph.D. degree from Aston University in 1969, all in electrical engineering.

From 1961 until 1966, he worked for English Electric Computers at Kidgrove, England, first as a student apprentice and then as a data processing engineer. From 1967 until 1969 he was employed as a Research Assistant at Aston University in England. Since 1969 he has been in the Department of Electrical Engineering at the University of Windsor, Windsor, Ont., Canada and currently holds the rank of Professor. He is currently engaged in research in the areas of one- and two-dimensional digital signal processing, high-speed digital hardware and microprocessor systems. He also teaches courses on electronic circuits, microcomputer systems and digital signal processing. From 1975 until 1976 he was a visiting senior research engineer at the Central Research Laboratories of EMI Ltd., Hayes, Middlesex, England.

Dr. Jullien is a member of the Association of Professional Engineers of Ontario, and the American Society for Engineering Education.



William C. Miller was born in Toronto, Ont., Canada on September 20, 1937. He received the B.S.E. degree from the University of Michigan, Ann Arbor, and the M.A.Sc. and Ph.D. degrees from the University of Waterloo, Waterloo, Ont., all in electrical engineering.

Since 1968 he has been with the Department of Electrical Engineering at the University of Windsor, Windsor, Ont., where he is currently a Professor and member of the Signal Processing Laboratory Staff.

Dr. Miller is a registered Professional Engineer in the Province of Ontario.

Detection of Faults in Programmable Logic Arrays

JAMES E. SMITH, MEMBER, IEEE

Abstract—**A new fault model is proposed for the purpose of testing programmable logic arrays. It is shown that a test set for all detectable modeled faults detects a wide variety of other faults. A test generation method for single faults is then outlined. Included is a bound on the size of test sets which indicates that test sets are much smaller than would be required by exhaustive testing. Finally, it is shown that many interesting classes of multiple faults are also detected by the test sets.**

Index Terms—**Programmable logic arrays, fault detection, fault modeling, test generation.**

I. INTRODUCTION

PROGRAMMABLE logic arrays (PLA's) provide the logic designer with an economical way of realizing combinational switching functions [1]–[3]. The economy is achieved by manufacturing standard "blank" arrays, and, as a final step, "programming" the array to perform a particular set of functions. In some technologies programming is performed by using a custom mask for the final metalization step. Field programmable logic array (FPLA) technologies allow the user to program the array by blowing fusible links within the array.

As with any other logic circuit, PLA's must be tested to insure that they operate correctly. Essentially, three different testing schemes are possible. The first is to place special test circuitry on the array [3]. This special circuitry is then enabled by placing voltage levels on inputs and outputs which are beyond normal levels. This method is used to avoid the addition of pins for testing and checks the presence or absence of connections in the PLA. While this method allows the PLA to be tested quickly, a PLA tested in this way has not been tested under normal operating conditions. Furthermore, after such a PLA is placed in a system it may be difficult to apply the appropriate testing signals which require abnormal signal levels.

A second test method is to exhaustively apply all possible input vectors to the array and check to see if it responds correctly. Exhaustive testing more adequately reflects normal operating conditions, but it has the disadvantage of requiring rather sophisticated high-speed test equipment. While a manufacturer may have such equipment, a user in the field, who often needs to test a PLA, may not. In addition, as PLA technology advances and the number of inputs increases, exhaustive testing will become impractical even with high-speed equipment. Exhaustive testing can also be very difficult in a system environment where other logic typically separates the PLA from the "outside world." Here,

Manuscript received August 9, 1977; revised September 1978.

The author is with the Department of Electrical and Computer Engineering, University of Wisconsin, Madison, WI 53706.