quantization errors problem could be soon neglected. It is our experience, however, that as computers become more precise, faster, and cheaper, larger systems are implemented by them and the sampling rate becomes larger. This increase in dimensions and sampling rate would require, once again, the consideration of the quantization error problem.

References

[1] S. Y. Hwang, "On optimization of cascade fixed point digital filters," *IEEE Trans. Circuit Theory*, vol. CT-70, pp. 163-166, 1974.

[2] ——, "Roundoff noise in state-space digital filtering: A general analysis," *IEEE Trans. Acoust., Speech, Signal Processing*, vol. ASSP-24, pp. 256-262, 1976.

[3] ——, "Minimum uncorrelated unit noise in state space digital filters," *IEEE Trans. Acoust., Speech, Signal Processing*, vol. ASSP-25, pp. 273-281, 1977.

[4] W. S. Lee, "Optimization of digital filters for low roundoff noise," *IEEE Trans. Circuits Syst.*, vol. CAS-21, pp. 424-431, 1976.

[5] B. Liu and A. Peled, "Heuristic optimization of cascade realization of fixed point digital filters," *IEEE Trans. Acoust., Speech, Signal Processing*, vol. ASSP-23, pp. 464-473, 1975.

[6] C. T. Mullis and R. A. Roberts, "Synthesis of minimum roundoff noise fixed point digital filters," *IEEE Trans. Circuits Syst.*, vol. CAS-23, pp. 551-562, 1976.

[7] J. Fadavi-Ardekani, S. K. Mitra, and B. D. O. Anderson, "Extended state-space model of discrete-time dynamic systems," *IEEE Trans. Circuits Syst.*, vol. CAS-29, pp. 547-556, 1982.

[8] D. S. K. Chan, "Constrained minimization of roundoff noise in fixed-point digital filters," in *Proc. 1979 IEEE Int. Conf. Acoust., Speech, Signal Processing*, 1979, pp. 335-339.

[9] L. W. Bowmar and J. C. Hung, "Minimum roundoff noise digital filters with some powers-of-two coefficients," *IEEE Trans. Circuits Syst.*, vol. CAS-31, pp. 833-840, 1984.

[10] A. B. Stripad, "Performance degradation on digital implemented Kalman filter," *IEEE Trans. Aerosp. Electron. Syst.*, vol. AES-17, pp. 629-634, Sept. 1981.

[11] D. Williamson, "Finite wordlength design of digital Kalman filters for state estimation," *IEEE Trans. Automat. Contr.*, vol. AC-30, pp. 930-939, 1985.

[12] P. Moroney, A. S. Willsky, and P. K. Houpt, "Rounding noise and scaling in digital implementation of control compensators," *IEEE Trans. Acoust., Speech, Signal Processing*, vol. ASSP-31, pp. 1464-1477, 1983.

[13] G. Amit and U. Shaked, "Small roundoff noise realization of fixed-point digital filters and controllers," *IEEE Trans. Acoust., Speech, Signal Processing*, vol. 36, 1988.

[14] J. Zeman and A. G. Lindgren, "Fast digital filters with low roundoff noise," *IEEE Trans. Circuits Syst.*, vol. CAS-28, pp. 716-723, 1981.

A Pipelined FFT Processor for Word-Sequential Data

GUOAN BI AND E. V. JONES

Abstract—**A modified fast Fourier transform algorithm is described together with a real-time pipelined implementation. The approach is particularly suited to sequentially presented input data. For example, for the often-favored radix-4 implementation, the new method requires less data memory and only 1/3 of the complex multipliers of a conventional design.**

I. Introduction

Many applications of special purpose fast Fourier transform (FFT) processors arise from signal processing problems which have an inherent real-time requirement. Furthermore, data that originate from a distant source will, for reasons of transmission economy, usually be in a bit and/or word-sequential format. However, FFT algorithms [1]-[3] call for spatially global interconnections, and so do not naturally interface with such sequential input data. Thus, circuit intensive demultiplexing and input buffering techniques have to be employed. In the literature, two approaches have been proposed to deal with this data/processor mismatch. Taking an N-point radix-4 pipelined configuration as an example, one solution is shown in Fig. 1(a) where input delay stages before the first butterfly element provide the requisite data word timing [1]. Another approach is to write the input data into a buffer then read out at $1/4$ input bit rate, as shown in Fig. 1(b) [2]. To achieve real-time processing, four memory blocks (each storing $N/4$ complex data words) receive the input data, while another four blocks dump the previously received input data.

These radix-4 pipelined processors require $3\log_4 N - 3$ complex multipliers and $8\log_4 N$ complex adders [1]. Also, about $2.5N$ and $3N$ complex data stores are needed for Fig. 1(a) and (b), respectively; this includes the delay stages for input buffering. The efficiency or utilization of the butterfly elements in Fig. 1(a) is 25 percent for sequential input data, although full utilization can be achieved when four parallel input data streams are available. This poor efficiency for fully sequential data results from the processing throughput mismatch between the input data and the potentially (four times) higher rate of the butterfly elements. Full utilization of the butterfly elements can be achieved if a bit-rate conversion is performed as shown in Fig. 1(b), and the pipeline is operated at $1/4$ of the input rate. A pipelined discrete Fourier transform (DFT) linear systolic array has been reported which can naturally interface with the sequential input data [4]. For an N-point DFT, however, it needs N complex multipliers and adders, and $5.5N$ complex shift registers, a requirement which becomes impractical when N is large.

This correspondence presents a new efficient implementation which can be used for either bit-serial or bit-parallel word-sequential input data. An algorithm is first derived from the definition of the DFT, and then a radix-4 pipelined FFT processor is implemented to illustrate the use of our algorithm.

II. Algorithm

The N-point DFT of a finite duration complex sequence $x(n)$ is defined by

$$X(k) = \sum_{n=0}^{N-1} x(n) W_N^{nk} \qquad k = 0, 1, \cdots, N-1;$$
$$W_N = e^{-j(2\pi/N)}. \tag{1}$$

Let N be a composite number of v integers so that $N = r_1 r_2 \cdots r_v$ and define

$$N_t = \frac{N}{r_1 r_2 \cdots r_t} \qquad 1 \le t \le v-1 \tag{2}$$

where t is the stage number of the decomposed DFT and r_t its radix. Using the recursive property of (2) and the relationship $W_{N_i N_j}^{N_j k} = W_{N_i}^{k}$, for radix r_1 (1) becomes

$$\begin{aligned} X(k) &= \sum_{p=0}^{r_1-1} x(N_1 p) W_N^{N_1 pk} + \sum_{p=0}^{r_1-1} x(N_1 p + 1) W_N^{(N_1 p+1)k} \\ &\quad + \cdots + \sum_{p=0}^{r_1-1} x(N_1 p + N_1 - 1) W_N^{(N_1 p + N_1 - 1)k} \\ &= \sum_{q_1=0}^{N_1-1} W_N^{q_1 k} \sum_{p=0}^{r_1-1} x(N_1 p + q_1) W_{r_1}^{pk}. \end{aligned}$$

Now, defining indexes k_1 and m_1 by $k = r_1 k_1 + m_1$ where $0 \le k_1 \le N_1 - 1$ and $0 \le m_1 \le r_1 - 1$, (3) becomes

$$X(r_1 k_1 + m_1) = \sum_{q_1=0}^{N_1-1} x_1(q_1, m_1) W_{N_1}^{q_1 k_1} \tag{4}$$

Manuscript received February 3, 1988; revised February 5, 1989.

The authors are with the Department of Electronic Systems Engineering, University of Essex, Wivenhoe Park, Colchester, Essex, CO4 3SQ, U.K.

IEEE Log Number 8931332.

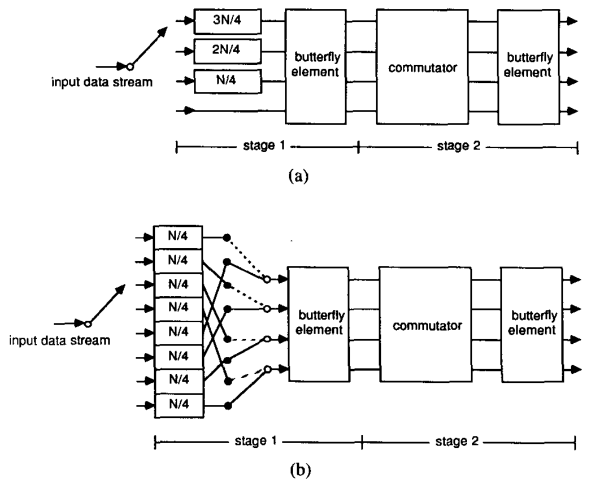

Fig. 1. Input interface arrangements for radix-4 FFT. (a) Demultiplexing input data stream. (b) Input buffering by alternatively writing and reading.

where

$$x_1(q_1, m_1) = W_N^{q_1 m_1} \sum_{p=0}^{r_1-1} x(N_1 p + q_1) W_{r_1}^{p m_1}. \tag{5}$$

This equation defines the computation for the first stage. Continuing the decomposing process of (4) for radix numbers other than r_1, the complete N-point DFT can be decomposed into $v - 1$ further stages of computation. The final stage is defined by

$$X(r_1 r_2 \cdots r_{v-1} m_v + r_1 r_2 \cdots r_{v-2} m_{v-1} + \cdots + r_1 m_2 + m_1)$$
$$= \sum_{q_{v-1}=0}^{r_v-1} x_{v-1}(q_{v-1}, m_{v-1}) W_{r_v}^{q_{v-1} m_v} \tag{6}$$

while intermediate stages (t) are given by the recursive equation

$$x_t(q_t, m_t) = W_{N_{t-1}}^{q_t m_t} \sum_{p=0}^{r_t-1} x_{t-1}(N_t p + q_t, m_{t-1}) W_{r_t}^{p m_t} \tag{7}$$

where, for both of (6) and (7),

$$2 \le t \le v - 1, \quad 0 \le m_i \le r_i - 1, \quad 0 \le q_i \le N_i - 1,$$
$$\text{and } 2 \le i \le v.$$

Each summation in these equations represents an r_t-point DFT computation. A "twiddle" factor ($W_{N_{t-1}}^{q_t m_t}$) is outside the summation, the decomposition thus corresponds to a decimation-in-frequency computation [2]. A decimation-in-time computation can also be obtained by slightly modifying the procedures used above. Since radix numbers can be any positive integer, (5), (6), and (7) can be used for either mixed radix computation or uniform radix computation (i.e., $r_1 = r_2 = r_3$, etc.). As an example, the flow graph of a 16-point radix-4 FFT is shown in Fig. 2. The corresponding equations are

$$X(4m_2 + m_1) = \sum_{q_1=0}^{3} x_1(q_1, m_1) W_4^{q_1 m_2} \tag{8}$$

$$x_1(q_1, m_1) = W_{16}^{q_1 m_1} \sum_{p=0}^{3} x(4p + q_1) W_4^{p m_1},$$
$$0 \le m_1, m_2 \le 3. \tag{9}$$

In Fig. 2, each open circle represents a summation while the dots define the stage boundaries. The number inside the open circle is the value of m_1 (for the first stage) or m_2 (for the second stage). The number outside the open circle is the twiddle factor applied.

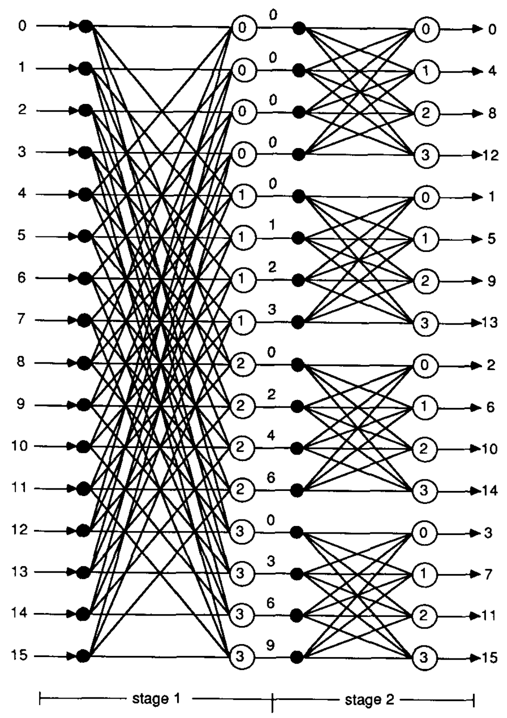

Fig. 2. Flow graph for a 16-point radix-4 FFT computation.

III. Implementation

A pipelined N-point radix-4 FFT processor for bit-serial word-sequential input data built according to (5), (6), and (7) will have $v = \log_4 N$ stages corresponding to the v equations. Each stage produces one output within each word cycle rather than 4 outputs as in a conventional pipelined implementation. As an example, Fig. 3 shows the block diagram of an N-point uniform radix-4 processor. Each stage contains a commutator, a butterfly element (performing the summations), and a complex multiplier (for twiddle factors). The sequential outputs at each stage must be ordered in accordance with the value of m_t (for example, from Fig. 2, at stage 1, the outputs associated with $m_1 = 0$ are produced in the first four word cycles, then those associated with $m_1 = 1$ in the next four word cycles, and so on). In general, the computation at stage $t + 1$ can start after N_t output data words have been produced by stage t rather than after $3N_t + 1$ words as in a conventional radix-4 design; thus, the delay elements preceding the commutator in [3] can be saved.

A suitable bit-serial complex multiplier will be found in [5] while the commutator and the butterfly element required are described below.

A. Commutator

As seen in (7), the input data for each summation at stage t are separated in time by N_t words. The requisite commutator is shown in Fig. 4 (this is required for both real and imaginary parts). It consists of six shift registers each providing N_t word delays. Control signals (denoted c_1, c_2, and c_3) select the appropriate data via 2:1 multiplexers in accordance with the value of m_t. By way of example, the timing and operation of the commutator for stage 1 for a 16-point FFT is shown in Fig. 5, where t' is the instant when the first input word arrives. As shown at the top of the figure, each input word occupies a word slot of duration T and is numbered according to its appearance in time. Below this, the 2:1 multiplexer operations of Fig. 4 for appropriate values of m_1 are indi-

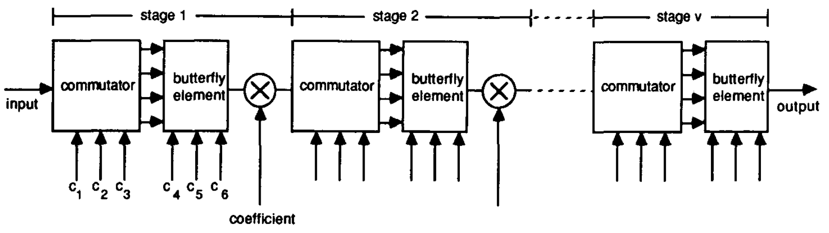

Fig. 3. N-point radix-4 pipelined FFT processor.

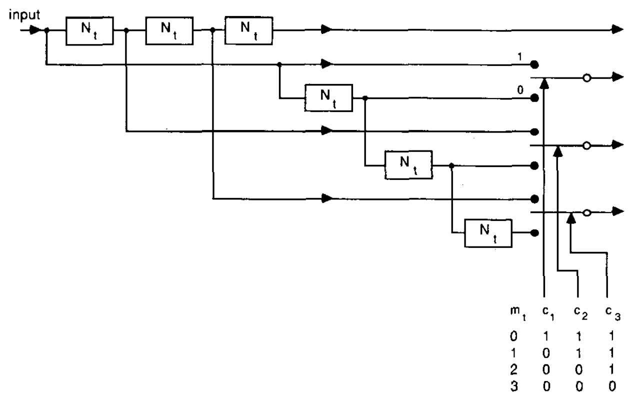

Fig. 4. Commutator for stage t.

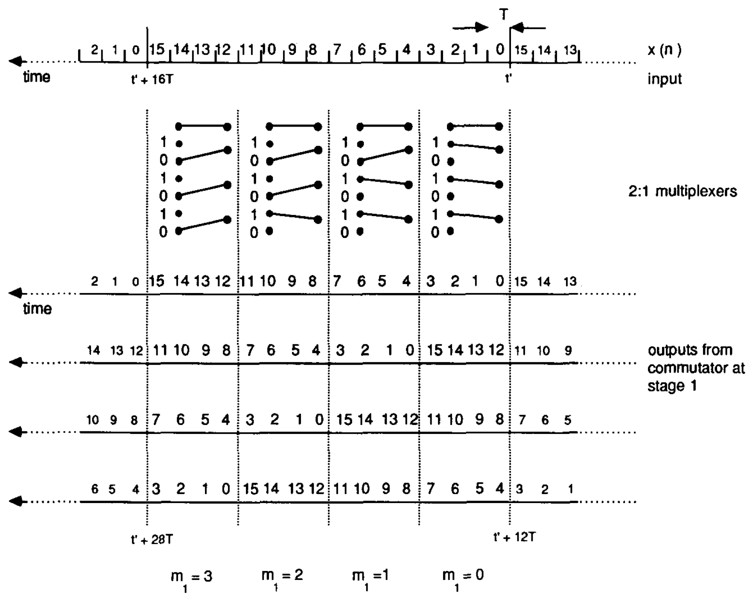

Fig. 5. Timing and operation of the commutator at stage 1 for a 16-point radix-4 FFT.

cated by the lines between dots. The commutator output timing is shown at the bottom of the figure.

Our commutator is different from that used in other algorithms ([3], for example) in that the same set of data is supplied to its associated butterfly element for every N_{t+1} word cycles, as shown in Fig. 5. With the exception of the first stage, this simplifies the commutator buffering so that only $6N_t$ delay elements are required at stage t in comparison to $12N_t$ in [3].

B. Butterfly Element

The butterfly element performs the summations of (5), (6), and (7). For radix-4, the complex multiplication within the sum can be replaced by the combination of addition, subtraction, and swapping between the real and imaginary parts, as shown in Fig. 6. Three complex adder/subtractors (each comprising a real and imaginary element) are used instead of eight complex adders [1]-[3]. Control

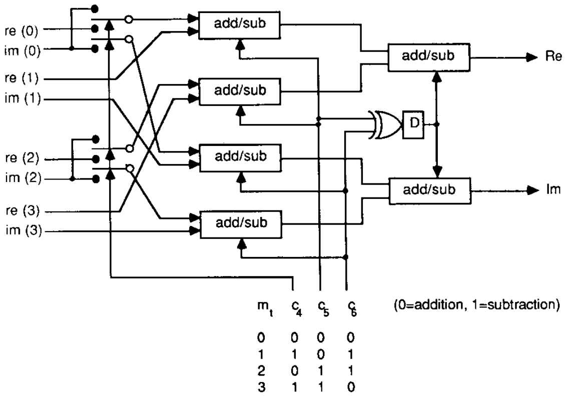

Fig. 6. Butterfly element for stage t.

signals again select data and functions in accordance with the value of m_t. Our butterfly element produces N outputs consecutively over N word cycles in contrast to conventional configurations [as in Fig. 1(a)] which must produce N outputs within $N/4$ word cycles leaving $3N/4$ word cycles unused. Thus, only one complex multiplier is needed for the twiddle rotation at each stage instead of three in other designs of butterfly realizations [1]-[3].

IV. Discussion

Our algorithm attempts to minimize the hardware required, consistent with matching the processing throughput to the sequential data input rate. Inspection of the commutator shows that the number of complex data stores for stage t is $2(r_t - 1)N_t$. Thus, the total number of complex data stores in the whole pipeline is

$$D_{\text{total}} = \sum_{t=1}^{t'} 2(r_t - 1)N_t = 2\left[\frac{(r_1 - 1)N}{r_1} + \frac{(r_2 - 1)N}{r_1 r_2} + \cdots + \frac{(r_{t'} - 1)N}{r_1 r_2 \cdots r_{t'}}\right] = 2(N - 1). \tag{10}$$

Since r_t can be any positive integer, this result can be applied to any uniform or mixed radix algorithm. The total number of complex data stores D_{total} for our algorithm increases with N, while for conventional algorithms it increases with both N and radix number r_t [1]. In the case of radix-4 implementations, conventional designs as shown in Fig. 1(a) and (b) require about $2.5N$ or $3N$ complex data stores, respectively.

In both the conventional and the proposed constructions, the number of stages in the pipeline is the same, but in each stage three complex multipliers and eight adders are necessary in the former compared to only one multiplier and three adder/subtractors in the latter. Thus, significant savings are achieved at the expense of providing controllable adder/subtractors within the butterfly element and a few associated simple control signals. The implementation can be naturally operated in real time and has 75 and 100 percent utilization of complex multipliers and adder/subtractors, respectively. Table I summarizes the hardware requirements and the achieved computational efficiency for various designs. Simulation of the pipelined implementation as shown in Fig. 3 has been carried out from $N = 16$ to 1024.

V. Conclusion

A new design for a real-time pipelined FFT processor for word-sequentially presented data is proposed. This method can be used for both mixed and uniform radix number implementations. A radix-4 example shows that in addition to a high computational efficiency, significant savings have been achieved on complex multipliers, adders, and data stores.

TABLE I
A Comparison of Hardware Requirements and Computational Efficiency

	Radix 4 FFT shown in		DFT systolic array [4]	Proposed radix 4 FFT
	figure 1(a)	figure 1(b)		
multipliers	3 ($\log_4$ N -1)	3 ($\log_4$ N -1)	N	$\log_4$ N -1
adders	8 $\log_4$ N	8 $\log_4$ N	N	3 $\log_4$ N **
data memory	2.5 N	3N	5.5 N shift registers *	2N
computational efficiency	25 %	100 % at 1/4 input rate	100 %	add/sub 100 % multiplier 75 %

* The shift register length is equal to one multiply - add cycle

** Adder/ subtractors

References

[1] B. Gold and T. Bially, "Parallelism in fast Fourier transform hardware," *IEEE Trans. Audio Electroacoust.*, vol. AU-21, pp. 5-16, 1973.
[2] L. R. Rabiner and B. Gold, *Theory and Application of Digital Signal Processing.* Englewood Cliffs, NJ: Prentice-Hall, 1975, ch. 10.
[3] E. E. Swartzlander, *VLSI Signal Processing Systems.* Boston, MA: Kluwer Academic, 1986, ch. 6.
[4] G. H. Allen, P. B. Denyer, and D. Renshaw, "A bit serial array DFT," in *Proc. IEEE ICASSP'84*, vol. 1, San Diego, 1984, pp. 44A.1.1-4.
[5] G. Bi, "Application of mode controlled logic to pipelined serial signal processing," Ph.D. dissertation, Univ. Essex, U.K., 1988.

Simple Method for Generation of Multiple Normal Random Signals

TERUYUKI IZUMI

Abstract—**Uncorrelated multiple normal random signals are generated from a single binary random signal with proposed digital filters. A set of N weighting functions of the filters is derived from an even-shift orthogonal sequence. The implementation of the filters is very simple due to the binary property of the weights.**

I. Introduction

Random signals are used as an excitation source in various fields such as system identification. If a system to be tested has N inputs, N excitation sources are required. Random excitation is also applied to modal testing of structures. In this test, a simultaneous excitation at the multiple points is effectively applied to rapid and accurate identification [1]. Uncorrelated multiple random signals with a normal distribution are required for excitation. N uncorrelated random signals can be generated by passing a single normal random signal through a parallel cascade of N zero-memory nonlinear filters [2]. But all of them do not possess a normal distribution. Another method of generating N random signals is to pass a binary random signal through digital filters. In that case, the generated signals are normal and uncorrelated to each other if the weighting functions of the N filters are chosen based on nth-order

Manuscript received February 19, 1988; revised February 11, 1989.
The author is with the Department of Electrical Engineering, Yamaguchi University, Tokiwadai, Ube City, Yamaguchi 755, Japan.
IEEE Log Number 8931345.