Thomas A. Kriz

Reduced Data Re-Order Complexity Properties of Polynomial Transform 2D Convolution and Fourier Transform Methods

This paper presents new results concerning the matrix data re-order requirements of polynomial-transform-based 2D convolution and 2D Fourier Transform methods which can be employed in digital processing of images and other 2D problems. The results indicate that several power-of-2 length-modified ring polynomial transform methods developed by Nussbaumer allow the total avoidance of the row-column matrix transpose commonly encountered in other algorithmic approaches, while also providing a number of other computational advantages. It is demonstrated that this property can be the source of significantly improved throughput on a number of existing data processing structures. An execution time comparison with an efficient Fast Fourier Transform algorithm base is made assuming the use of general register architecture and array processor units. It is also assumed that one makes use of recently developed efficient matrix transpose methods by Eklundh and Ari to support 2D FFT data re-order requirements. These comparisons demonstrate a two to four times throughput improvement for the use of the polynomial transform method in place of the 2D FFT approach to circularly convolve or generate 2D Fourier transforms for large 2D fields in the range 1024 × 1024 to 8192 × 8192.

Introduction

A number of multiply-free polynomial transform methods [1–3] have recently been devised to support efficient execution of convolution and 2D Fourier Transform (DFT) signal processing procedures. Analysis of this approach indicates that it can provide a significant number of attractive features: (1) reduced computational complexity, (2) Fast Fourier Transform (FFT)-like transform computation structures, (3) reduced computation noise, (4) minimum-multiply convolution algorithms, (5) real arithmetic instead of complex, (6) trigonometric coefficient table independence, and (7) efficient support for both real and complex data processes [3]. The results given in this paper indicate that another item should be added to this list which is of great importance in the execution of large field 2D convolution and 2D DFT processes, that of significantly reduced data re-order complexity. (A portion of these results were presented at the 1981 IEEE Conference on Acoustics, Speech and Signal Processing, Atlanta, GA, March 30–April 1, 1981.)

The data re-order complexity of 2D convolution and DFT processes can be characterized generally, in a practical sense, as the throughput impact of having to re-order matrix-organized data from one vector (or subvector) form to another, usually row to column and vice versa. Such data re-ordering complexities are commonly dominated by a requirement for transposition of the 2D data field when using non-polynomial transform-based 2D FFT and multi-dimensional CRT-mapped Fourier Transform methods. The impact of this transposition is obviously negligible if the existence of a random store is assumed which is large enough to contain the entire data field, so that any subvector can be conveniently addressed. For large field problems, however, where the data are stored with a particular vector orientation on a disk storage device (or tape), the impact can be severe. This has led to the development of a number of sophisticated methods to support efficient transpose of disk-stored matrix data in the execution of 2D DFT procedures [4–9]. These methods can be used to significantly reduce the impact of data re-order complexity compared to that of direct matrix-transpose methods for row-column re-ordering of disk-stored matrix data. It is shown, however, that the use of an appropriate polynomial transform method can totally

Table 1 Polynomial transform candidates.

Polynomial ring candidates $M(Z)$	*Transform length**	$G(Z)$ *root*	*Convolution dimension* $(N \times N)$	*Generalized products*	*Data re-order*
$(Z^q - 1)/(Z - 1)$	q	Z	$q \times q$	q mul $M(Z)$ 1 conv q	None
$(Z^q - 1)/(Z - 1)$	$2q$	$-Z$	$2q \times q$	$2q$ mul $M(Z)$ 1 conv $2q$	None
$(Z^{2q} - 1)/(Z^2 - 1)$	$2q$	$-Z^{q+1}$	$2q \times 2q$	$2q$ mul $M(Z)$ 1 conv $2 \times 2q$	$2 \times 2q$
$(Z^{q^2} - 1)(Z^q - 1)$	q	Z^q	$q \times q^2$	q mul $M(Z)$ q mul $(Z^q - 1)/(Z - 1)$ 1 conv q	$q \times q$
$(Z^{q^2} - 1)/(Z^q - 1)$	q^2	Z	$q^2 \times q^2$	$q(q + 1)$ mul $M(Z)$ q mul $(Z^q - 1)/(Z - 1)$ 1 conv q	$q \times q^2$ $q \times q$
$(Z^{q_1 q_2} - 1)/(Z^{q_2} - 1)$	q_1	Z^{q_2}	$q_1 \times q_1 q_2$	q_1 mul $M(Z)$ 1 conv $q_1 q_2$	None
$(Z^{2^{t-1}} + 1)$	2^t	Z	$2^t \times 2^t$	$3 \times 2^{t-1}$ mul $M(Z)$ 1 conv $2^{t-1} \times 2^{t-1}$	$2^n \times 2^{n-1}$ $n = t, \cdots, 2$
$(Z^{2^t} + 1)$	2^t	Z^2	$2^t \times 2^t$	2^t mul $M(Z)$ 2^{4t} scalar products	None

*q = odd prime,
t = positive integer.

eliminate the need for row-column re-ordering of disk-stored matrix data in large field 2D convolution and DFT problems, while providing many of the other attractive features cited above.

The paper is organized as follows to make this point: First, the data re-order properties of various polynomial transforms are reviewed. Next, a particular transform procedure is selected from this set as a preferred approach to reduce both computational and data re-order complexity in the execution of a large 2D circular convolution process. Finally, results are given for an execution time comparison of the selected polynomial transform method with 2D Fourier Transform methods in which it is assumed that sophisticated matrix transpose procedures [3–9] are used to convolve (or generate a 2D DFT) for 2D real, large fields in the range 1024×1024 to 8192×8192.

Theoretical background

The general definition of a polynomial transform which supports 2D convolution can be specified as [3]

$$A_k(Z) = \sum_{m=0}^{N-1} P_m(Z)\,[G(Z)]^{mk} \bmod M(Z)$$

$$k = 0, 1, \cdots, N, \qquad (1)$$

where

$$P_m(Z) = \sum_{n=0}^{N-1} a_{n,m} Z^n,$$

$G^N(Z) = 1 \bmod M(Z)$,

$N^{-1} \bmod M(Z)$ and $G^{-1}(Z) \bmod M(Z)$ exist, and

$$S \equiv \sum_{k=0}^{N} G(Z)^{qk} \bmod M(Z)$$

$\equiv 0$ for $q \not\equiv 0 \bmod N$

$\equiv N$ for $q \equiv 0 \bmod N$.

The inverse transform has a similar form:

$$P_m(Z) = \frac{1}{N} \sum_{k=0}^{N-1} A_k(Z)[G(Z)]^{-km} \bmod M(Z). \qquad (2)$$

The coefficients of $P_m(Z)$ in these expressions are the sampled values of the mth row (or column) in a 2D space, and the polynomial $M(Z)$ and root $G(Z)$ must be selected to satisfy the conditions given above. Some particularly useful choices of $M(Z)$ and $G(Z)$ which have been found to satisfy these conditions are listed in Table 1 [2, 3]. As indicated in this table, the selection of $M(Z)$ and $G(Z)$ is closely coupled to the size of the 2D convolution field, and typically involves the partition of a convolution into one or more polynomial transform pairs for complete implementation.

The data transpose requirements which can be used as a basis to define the data re-order complexity of particular polynomial ring candidates are also listed in the rightmost column of Table 1. The nature of the requirements for data re-order depends upon the algorithmic form of the computa-

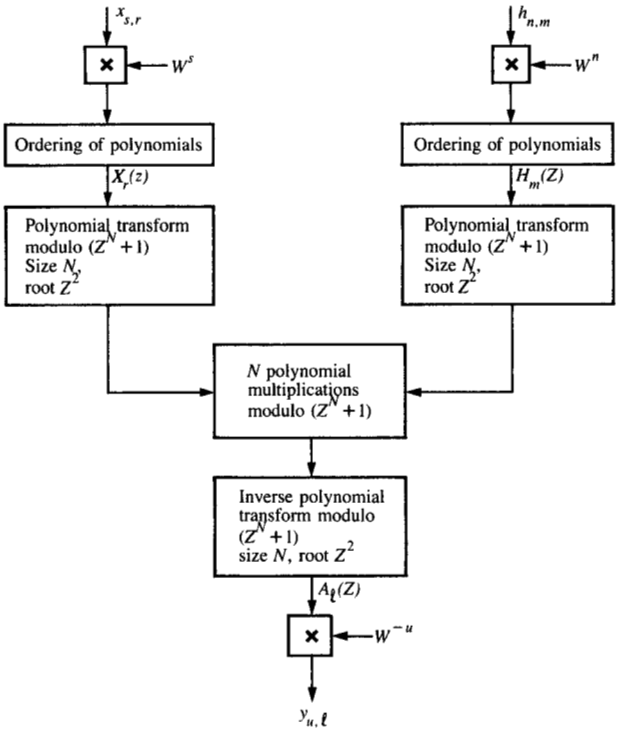

Figure 1 $(2^t \times 2^t)$ convolution via PT$(2^t, Z^2)$.

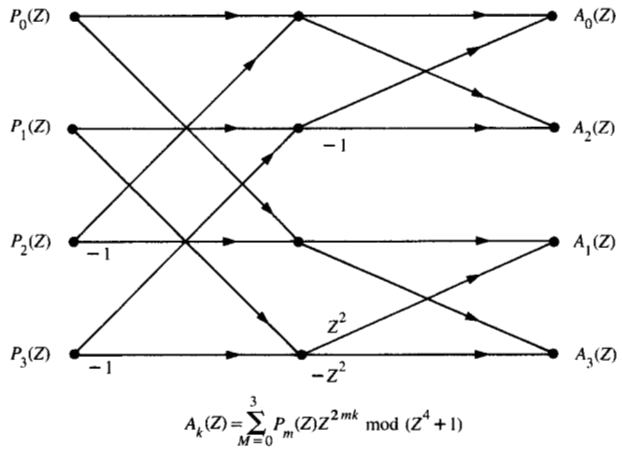

Figure 2 Four-point PT$(2^t, Z^2)$ decimation in time example.

tion structure used to implement the transforms in (1, 2) and the way that the spectral convolution process, when partitioned, is mapped to a set of transforms yielding partial results which can be constructed together via a CRT procedure to obtain the desired convolution result. The data re-order requirements in Table 1 presume the use of the particular convolution process partitions which have been defined by Nussbaumer in the development of the transform methods [3].

The polynomial coefficient matrix data re-ordering requirements in Table 1 are based upon the following assumptions. The first six transforms are computed by accessing the coefficients of $P_m(Z)$ from disk storage in natural order with $k = 0, 1, \cdots, N$, without re-ordering of the polynomial coefficient set $P_m(Z)$. It is important here to note that the apparent requirement for rotation of the coefficients modulo $M(Z)$ does not require an actual rotation, but can be accomplished with the aid of an index pointer. It is evident, then, that all the data re-order requirements for these first six transforms must stem from the nature of the partitioning procedure which has been assumed. The form of this partition can be inferred from the form of the generalized product data for each transform in Table 1. Since the assumed transform algorithm must make repeated use of the same polynomial coefficient input data stream, it can be seen that one must use a separate storage region to collect the transform result. Thus, the reduced level ($<<N \times N$) of the data transpose requirements for the first six transforms in comparison to that of a 2D DFT method appears to be gained at the expense of requiring twice the storage space, not to mention an $o(N^3)$ arithmetic operation requirement.

Although some of the first six transforms in Table 1 require no transpose operation, from an arithmetic point of view for very large values of N none of these are competitive with the $o(N \log N)$ arithmetic operation level offered by some 2D DFT methods. Such is not the case, however, for the lower two transforms defined in the table, which permit the use of FFT-like computation structures to achieve an arithmetic load significantly below that of FFT-organized DFT methods. Of these two transforms, the lower one, based upon the use of a modified ring, exhibits both superior arithmetic and data re-order complexity properties and is, therefore, preferred for large field 2D convolutions. This transform, which is designated PT$(2^t, Z^2)$, based upon length and root attributes, is now examined in more detail.

One of several 2D-convolution implementations developed by Nussbaumer [3] which uses PT$(2^t, Z^2)$ is illustrated in Fig. 1. The use of scaling by W in the process modifies the natural negative wrap property of PT$(2^t, Z^2)$ modulo $(Z^N + 1)$ to support a circular convolution with positive wrap. The use of this scaling procedure is analogous to that which is well known in Fourier Transform theory for modification of a circular wrap property. As indicated in Fig. 1, no process partition is employed which would require data transposition.

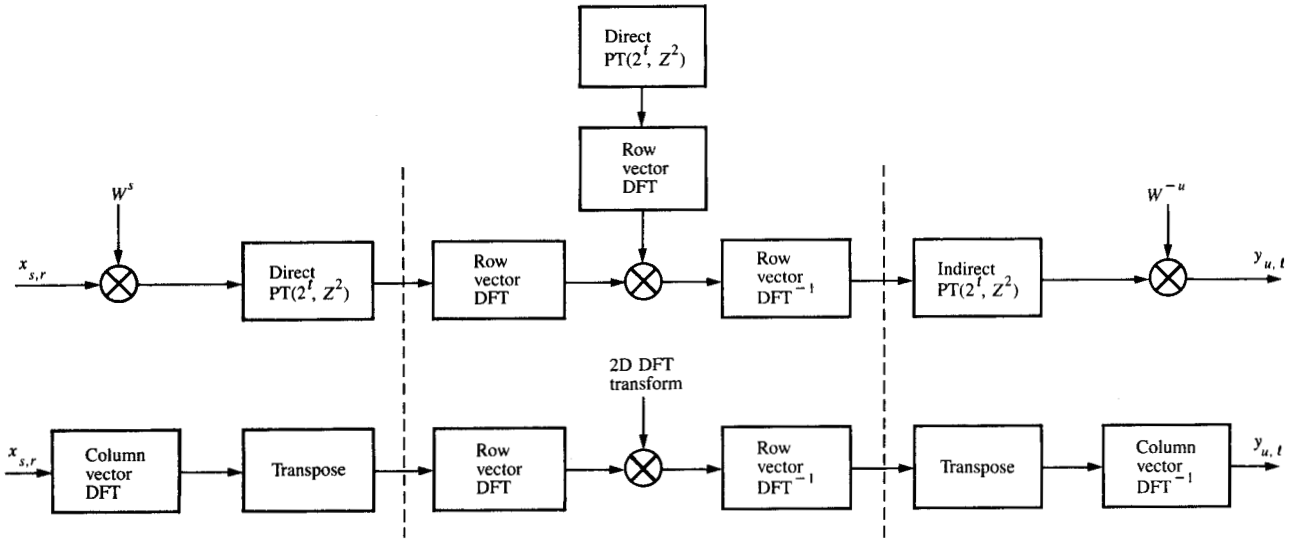

Figure 3 2D convolution via $PT(2^t, Z^2)$ with form similar to that via 2D DFT.

A remaining point which must now be demonstrated, however, is that there is an efficient algorithmic form of implementation of $PT(2^t, Z^2)$ which avoids the need to transpose the matrix data. This is achieved by rewriting (1) as

$$A_k(Z) = \sum_{m=0}^{N/2-1} P_{2m}(Z)Z^{4mk} + Z^{2k} \sum_{m=0}^{N/2-1} P_{2m+1}(Z)Z^{4mk} \bmod (Z^N + 1) \qquad (3)$$

and

$$A^{(Z)}_{k+N/2} = \sum_{m=0}^{N/2-1} P_{2m}(Z)Z^{4mk} - Z^{2k} \sum_{m=0}^{N/2-1} P_{2m+1}(Z)Z^{4mk} \bmod (Z^N + 1)$$
$$k = 0, 1, \cdots, N/2 - 1, \qquad (4)$$

with $G(Z) = Z^2$ and $M(Z) = (Z^N + 1)$. The form of (3), (4) indicates that $PT(2^t, Z^2)$ can take an algorithmic form which is exactly analogous to that of an FFT by replacing each complex element of the FFT with a vector of polynomial coefficients and the complex rotational operator W with a vector rotational operator Z^2. This point is illustrated graphically in Fig. 2 for $N = 4$. Because of this exact analogy, use of well-known FFT computation schemes [10] can be made to facilitate efficient execution of the computation of $PT(2^t, Z^2)$ and to avoid transpose of the data. FFT-like butterfly computation structures can also be used which operate on as few data items as a vector pair or a section of a vector pair at a time. Typically, in executing a program for $PT(2^t, Z^2)$ on disk-stored data, at each of the log N computation stages the computation stream can be organized to execute interlaced (first half/last half alternate) access of the previously generated disk data stream to support (odd/even-element-ordered) generation of the data stream used in the next stage. This interlaced access of input data implies that additional disk storage is required to store generated results (assuming ordinary disk read/write capabilities). Usually four disk or tape units would be used to efficiently implement this scheme [10]. Thus, the avoidance of a data transpose appears to be gained at the expense of increased storage requirements. It is emphasized, once again, that explicit shifting to accomplish rotation of polynomial coefficients is unnecessary, since in practice the use of index register pointers in accessing and storing elements of a stationary vector can be employed to accomplish the same result.

The convolution results just presented can be extended to cover 2D DFT methods generated via the expressions (1), (2). As indicated in Fig. 3, if the use of a vector DFT procedure is assumed to implement the polynomial ring "spectrum multiply" process for the convolution shown in Fig. 1, the computation structure can be made to take a form which is quite similar to that of a standard row-column-partitioned DFT approach. Unfortunately, this approach does not provide a 2D DFT as an intermediate result, but is close in form to one which does. The algorithmic form of such a polynomial-transform-generated 2D DFT [3] which makes use of $PT(2^t, Z^2)$ is shown in Fig. 4. A comparison of the data re-order properties of this algorithm with other polynomial transform choices leads to a result which is similar to that given previously, and for 2D DFT processes provides the same advantages as those already cited for 2D convolution processes.

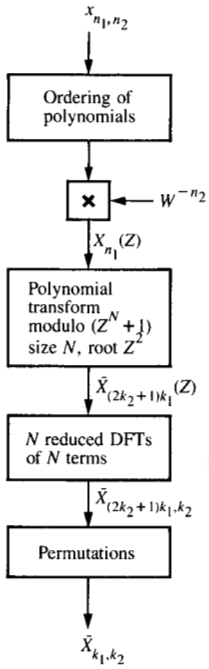

Figure 4 $N \times N$ 2D DFT via PT(2^t, Z^2) where $N = 2^t$.

Table 2 Matrix transpose execution time (seconds).

N × N real field	*GR machine (1024K word) (direct)*	*GR machine (1024K word) (Eklundh)*	*GR machine (1024K word) (Ari)*	*Array processor (256K word) (Eklundh)*
1024 × 1024	5	—	—	4
2048 × 2048	168	27	32	26
4096 × 4096	1312	111	127	242
8192 × 8192	10495	584	608	795

Table 3 2D-convolution DFT and spectrum multiply execution time (seconds) excluding matrix transpose.

N × N real field	*GR machine (1024K word)**	*Array processor (256K word)***
1024 × 1024	62	9
2048 × 2048	272	36
4096 × 4096	1184	144
8192 × 8192	5312	576

Execution constraint:
*arithmetic rate,
**data transfer rate.

Performance comparison results

This section gives results which demonstrate the reduced complexity impact of PT(2^t, Z^2) in terms of expected execution performance. Comparison is made with FFT-based 2D DFT methods assuming the use of efficient matrix transpose techniques.

The particular machines selected for this exercise are a general register (GR) architecture machine and an array processor. The performance assumptions for these machines are based upon actual execution time experience with FFT-like computation structures in real machines. The GR machine is assumed to possess a 1024K-word random access store and average throughput capability for such structures to execute 4×10^6 SRS floating point ADD/SUBTRACT operations per second (or 2×10^6 MULTIPLY operations). The array processor is assumed to possess a 256K-word bulk store and a maximum arithmetic execution capability of both 20×10^6 floating point ADD/SUBTRACT operations and 10×10^6 floating point MULTIPLY operations per second. Disk track size is taken as 4K words with a one-megaword-per-second transfer rate. Disk access time is presumed to be 20 ms average per segment for transpositions and 4 ms track to track for support of FFT-like processes.

The matrix transpose execution time for both machines is given in Table 2 for a variety of transpose methods, including that which is currently believed to be optimum [8, 9]. As can be seen, the relatively large store and rapid data item exchange capability of both machines supports rapid reorder of the disk-stored matrix data. These execution times, which presume a 1-μs data item exchange execution time in the GR machine and a 0.2-μs exchange time in the array processor, were computed using execution time formulas given in the cited references [4, 8, 9, 11]. The dominant execution constraint for the matrix transpose process is the 20-ms-per-segment disk data access time.

Execution times for DFT and spectrum multiply processes are tabulated in Table 3 assuming various-size real 2D fields. The arithmetic execution rates quoted above would lead one to expect a factor of 10 difference in the throughput rate of the two machines. For smaller 2D fields, this objective is not met because of the data transfer rate limitation which has been specified for the array processor. It is presumed in Table 3 and those which follow that a reference transform has been pre-computed.

Estimated GR machine 2D circular convolution execution times are given in Table 4 for the methods identified. It may be seen from the results in this table and the previous two tables that the arithmetic execution load is extremely dominant in GR machine execution of the 2D convolution problem under consideration. Also demonstrated is the fact that

the polynomial transform method offers a performance gain of almost 2 over the DFT-based method.

Table 5 lists estimated execution times for the 2D circular convolution problem assuming use of the array processor, alone and in combination with the GR machine. The results show that the impact of data re-order complexity is quite dominant in the DFT-executed methods on the array processor and increases significantly as one proceeds from the smallest to the largest 2D field for a stand-alone array. Use of PT(2^t, Z^2), therefore, significantly reduces the execution time, especially as we proceed from smallest to largest 2D field, but the result indicates that total array processor stand-alone execution time using PT(2^t, Z^2) is essentially equivalent to that of the DFT method arithmetic execution time (ignoring the transpose) given in the rightmost column of Table 3. Thus, very little benefit is realized for the impact of the reduced computation load provided by PT(2^t, Z^2). This occurs because those portions of the polynominal transform which cannot be overlapped with generalized product execution in the array processor are extremely I/O transfer rate bound. This problem can be alleviated as illustrated in the rightmost column of Table 5 by executing a portion of the PT(2^t, Z^2) transform load in a GR machine host and the remaining part plus polynomial products in the array processor. It can be seen from the results in Table 5 that the use of PT (2^t, Z^2) provides a performance boost of 2 to 4 over the best DFT method. The principal source of this improved throughput is from a reduction in data re-order complexity.

As a final item, it should be noted that performance results for use of PT(2^t, Z^2) to implement a 2D DFT process on the same GR and array processor machines is about half that of the execution time given for the PT(2^t, Z^2)-based convolution with a slight additional amount to accommodate bit-reversal re-ordering of the generated result. This result follows from the form of the algorithm given in Fig. 4 for 2D DFT generation in comparison to that for the 2D convolution shown in Fig. 3.

It should also be emphasized that the performance results presented in this paper have been based upon a particular set of assumed machine characteristics and the use of a popularly employed DFT method as a comparison basis. A change in these assumptions could potentially lead one to a different set of conclusions.

Table 4 GR machine 2D-convolution execution time (seconds) (1024K word store).

N × N real field	*2D DFT (direct)*	*2D DFT (Eklundh)*	*Polynomial transform*
1024 × 1024	72	68	43
2048 × 2048	608	326	184
4096 × 4096	3808	1406	808
8192 × 8192	26302	6480	3578

Table 5 Array processor 2D-convolution execution time (seconds): (bulk store = 256K word); (alone and with GR machine and 1024K word main store).

N × N real field	*Array only 2D DFT (Eklundh)**	*Both machines 2D DFT (Eklundh)***	*Array only polynomial transform**	*Both machines polynomial transform**
1024 × 1024	17	10	9	5
2048 × 2048	88	54	35	20
4096 × 4096	628	222	138	83
8192 × 8192	2166	1168	560	340

Execution constraint:
*data transfer rate,
**GR machine transpose process.

References

1. H. J. Nussbaumer, "Digital Filtering Using Polynomial Transforms," *Electron. Lett.* **13,** 386–387 (November 1977).
2. H. J. Nussbaumer and P. Quandalle, "Computation of Convolutions and Discrete Fourier Transforms by Polynomial Transforms," *IBM J. Res. Develop.* **22,** 134–144 (March 1978).
3. H. J. Nussbaumer, *Fast Fourier Transform and Convolution Algorithms,* Springer-Verlag New York, Inc., New York, 1981.
4. J. O. Eklundh, "A Fast Computer Method for Matrix Transposing," *IEEE Trans. Computers* **C-21,** 801–803 (July 1972).
5. R. E. Twogood and M. P. Ekstrom, "An Extension of Eklundh's Matrix Transposition Algorithm and Its Application in Digital Image Processing," *IEEE Trans. Computers* **C-25,** 950–952 (September 1976).
6. M. Onoe, "A Method for Computing Large-Scale Two-Dimensional Transform Without Transposing Data Matrix," *Proc. IEEE* **63,** 196–197 (January 1975).
7. D. B. Harris *et al.,* "Vector Radix Fast Fourier Transform," *Proc. 1977 IEEE Conf. ASSP,* Hartford, CT, May 1977, pp. 548–551.
8. M. B. Ari, "On Transposing Large $2^n \times 2^n$ Matrices," *IEEE Trans. Computers* **C-27,** 72–75 (January 1979).
9. R. E. Twogood, "2D Digital Signal Processing with an Array Processor," *Proc. 1979 IEEE Conf. ASSP,* Denver, CO, April 1979, pp. 426–429.
10. R. C. Singleton, "A Method for Computing the FFT with Auxiliary Memory and Limited High Speed Storage," *IEEE Trans. Audio Electroacoust.* **AU-15,** 91–97 (June 1967).
11. R. E. Twogood, "2D FFTs of Large Images with the AP-120B," *Report VCRL-83674,* Lawrence Livermore Laboratory, Livermore, CA, February 29, 1980.

Received March 5, 1982; revised June 30, 1982

Thomas A. Kriz *IBM Instruments, Inc., Orchard Park, P.O. Box 332, Danbury, Connecticut 06810.* Dr. Kriz is chief architect of the advanced data processing system development group at IBM Instruments. He joined IBM in 1968 at the Federal Systems Division Owego, New York, laboratory, where he initially worked on the development of design automation tools for digital system testability

analysis and diagnostic data generation. In 1974, he transferred to the computer systems architecture group, where he was responsible for high-level design and system trade-off analysis of various array processor and VLSI chip structures for sensor system signal processing, image processing, and matrix-vector scientific data processing support. He joined IBM Instruments in 1981 to take responsibility for system architecture definition and advanced processor system development. He received the Ph.D. in electrical engineering in 1968 from Marquette University, Milwaukee, Wisconsin. Dr. Kriz is a member of the Association for Computing Machinery, Eta Kappa Nu, the Institute of Electrical and Electronics Engineers, Sigma Xi, and Tau Beta Pi.