Transactions Briefs

An Efficient Architecture for Multi-Dimensional Convolution

A. Elnaggar and M. Aboelaze

***Abstract*—This paper presents modified parallel architectures for multi-dimensional (m-d) convolution. We show that for two-dimensional (2-d) convolutions, with careful design, the number of lower-order 2-d convolutions can be reduced from nine to six with a computation saving of 33%. Moreover, the original speed of the computations is not affected. The proposed partitioning strategy results in a core of data-independent convolution computations, and can be generalized to the m-d convolution. The resulting very large scale integration networks have very simple modular structure, highly regular topology, and use simple arithmetic devices.**

I. Introduction

Multi-dimensional (m-d) convolution is a very important operation in signal and image processing with applications to digital filtering and video processing. Thus, abundant approaches have been suggested to achieve high-speed processing for linear convolution, and to design efficient convolution architectures [3]–[6]. However, the majority of the previous approaches focused on expressing a two-dimensional (2-d) convolution in terms of two consecutive stages of one-dimensional (1-d) convolutions.

Our methodology employs tensor product decompositions and permutation matrices as the main tools for expressing the convolution algorithm. We employ several techniques to manipulate such decompositions into suitable expressions that can be mapped efficiently onto very large scale integration (VLSI) structures. Tensor products (or Kronecker products), when coupled with permutation matrices, have proven to be useful in providing a unified decomposable matrix, formulations for multidimensional transforms, convolutions, matrix multiplication, and other fundamental computations [2], [3], [6].

The proposed algorithm is based on a nontrivial modification of the 2-d convolution algorithm recently proposed in [3] and realized in Fig. 1. We show that, using the properties of tensor product and permutation matrices [6], a large 2-d convolution computation can be decomposed recursively into three cascaded stages. We show that the number of lower-order convolvers at the core computations can be reduced from nine to six with a computation saving of 33%. It should be also emphasized that our partitioning and combining method does not make any assumption about how the core convolution is computed. Indeed, any suitable convolution method can be used. This makes the proposed method very flexible and realizable over a wide range of hardware and software platforms.

Manuscript received July 1999; revised August 2000. This paper was recommended by Associate Editor J. LeBlanc.

A. Elnaggar is with the Department of Electrical Engineering, Sultan Qaboos University, Muscat 123, Oman (e-mail: ayman@squ.edu.om).

M. Aboelaze is with the Department of Computer Science, York University, Toronto, ON, Canada M3J 1P3 (e-mail: aboelaze@cs.yorku.ca).

Publisher Item Identifier S 1057-7130(00)11033-X.

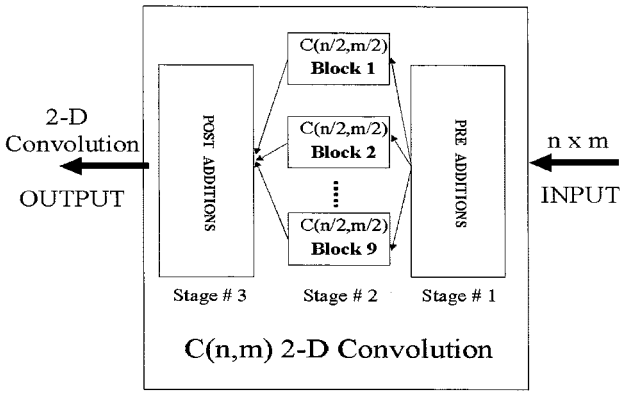

Fig. 1. The original realization of the 2-d convolution algorithm.

A. Tensor Product Properties

Some of the properties of the tensor product that will be used throughout this paper are [3], [6] as follows:

$$\text{Distributive property:} \quad AB \otimes CD = (A \otimes C)(B \otimes D) \tag{1}$$

$$\text{Commutative property:} \quad (A \otimes B) \otimes C = A \otimes (B \otimes C) \tag{2}$$

$$\text{If } n = n_1 n_2 n_3, \qquad \text{then } P_{n,n_1} P_{n,n_2} = P_{n,n_1 n_2}. \tag{3}$$

Parallel Operations: For square matrices A_{n_1} and B_{n_2}, if $n = n_1 n_2$ then

$$A_{n_1} \otimes B_{n_2} = P_{n,n_1} (I_{n_2} \otimes A_{n_1}) P_{n,n_2} (I_{n_1} \otimes B_{n_2}). \tag{4}$$

If $n = n_1 n_2$, then

$$P_{n,n_1} P_{n,n_2} = I_n. \tag{5}$$

For nonsquare matrix $A_{n,m}$, we have

$$A_{n,m} \otimes A_{n,m} = P_{n^2,n}(I_n \otimes A_{n,m}) P_{nm,n}(I_m \otimes A_{n,m}). \tag{6}$$

Where $P_{n,s}$ is an $n \times n$ binary matrix specifying an n/s-shuffle (or s-stride) permutation, and I_n is the identity matrix of size n.

II. Reducing the Complexity of the m-d Convolution Algorithm

For an $n_1 \times n_2$ input data image, the 2-d convolution output is given by [3]

$$\tilde{C}_{n_1,n_2} = \tilde{\tilde{R}}_{n_1,n_2} \left(I_9 \otimes C_{n_1/2,n_2/2}\right) \tilde{\tilde{Q}}_{n_1,n_2} \tag{7}$$

where

$$C_{n_1/2,n_2/2} = C(n_1/2) \otimes C(n_2/2)$$

is the lower order 2-d convolution matrix for an $n_1/2 \times n_2/2$ input image,

$$\tilde{\tilde{Q}}_{n_1,n_2} = \left(P_{9(n_1/2),3} \otimes I_{n_2/2}\right) (Q_{n_1} \otimes Q_{n_2}) \tag{8}$$

and

$$\tilde{\tilde{R}}_{n_1,n_2} = (R_{n_1} \otimes R_{n_2}) \left(P_{9(n_1-1),3(n_1-1)} \otimes I_{n_2-1}\right). \tag{9}$$

are the 2-d pre- and post-additions, respectively.

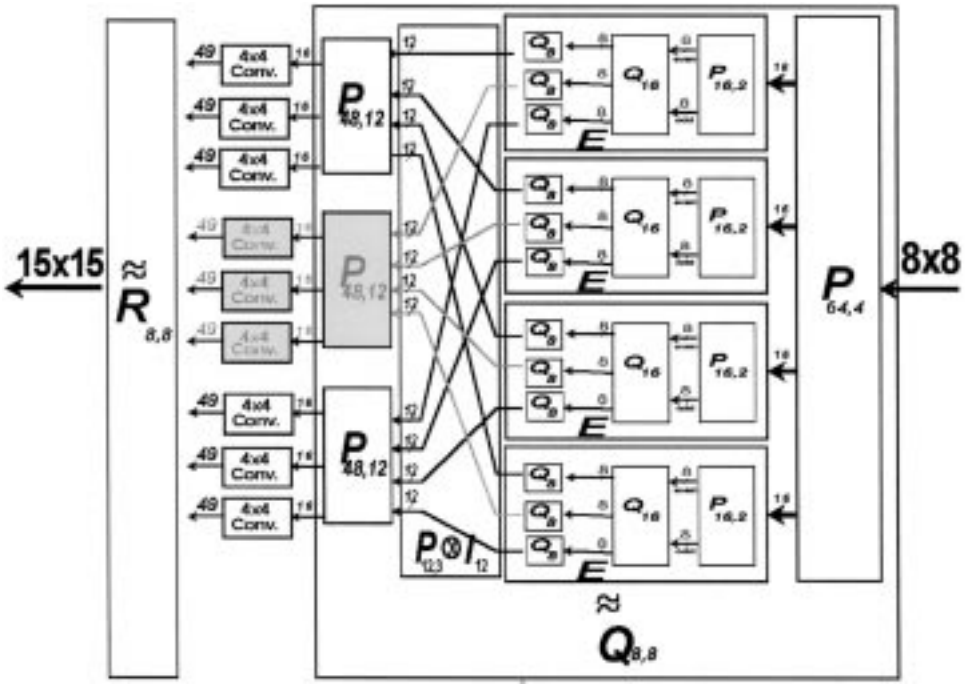

Fig. 2. The proposed 2-d convolution for an 8×8 input image with a detailed realization of $\tilde{\tilde{Q}}_{8,8}$.

Now, we will further manipulate (8) and (9) to exploit the resource sharing available and, consequently, realize the multiplexed architecture of the 2-d convolution using less number of lower-order convolvers.

Applying property (4), we can write Q_n in the form $Q_n = A \otimes I_{n/2}$ [3]. Consequently, $\tilde{\tilde{Q}}_{n_1,n_2}$ can be written as

$$\begin{aligned}\tilde{\tilde{Q}}_{n_1,n_2} &= \left(P_{9(n_1/2),3} \otimes I_{n_2/2}\right)\left(Q_{n_1} \otimes Q_{n_2}\right) \\ &= \left(P_{9(n_1/2),3} \otimes I_{n_2/2}\right)\left[A \otimes I_{n_1/2} \otimes A \otimes I_{n_2/2}\right]. \end{aligned} \quad (10)$$

Also, from properties (1) and (2), we have

$$\begin{aligned}\tilde{\tilde{Q}}_{n_1,n_2} &= \left(P_{9(n_1/2),3}\left(A \otimes I_{n_1/2} \otimes A\right)\right) \otimes \left(I_{n_2/2} I_{n_2/2}\right) \\ &= \left(P_{9(n_1/2),3}\left(A \otimes I_{n_1/2} \otimes A\right)\right) \otimes I_{n_2/2}. \end{aligned} \quad (11)$$

Since the matrix

$$A = \begin{bmatrix} 1 & 0 \\ 1 & 1 \\ 0 & 1 \end{bmatrix}$$

is of dimension 3×2 [3], we have

$$A = AI_2 \quad \left(I_{n_1/2} \otimes A\right) = I_{3n_1/2}\left(I_{n_1/2} \otimes A\right). \quad (12)$$

Substituting (12) in (11) then using property (1), we have

$$\begin{aligned}\tilde{\tilde{Q}}_{n_1,n_2} &= \left(P_{9(n_1/2),3}\left(\left(AI_2 \otimes I_{3n_1/2}\right)\left(I_{n_1/2} \otimes A\right)\right)\right) \otimes I_{n_2/2} \\ &= \left(P_{9(n_1/2),3}\left(A \otimes I_{3n_1/2}\right)\left(I_{n_1} \otimes A\right)\right) \otimes I_{n_2/2}. \end{aligned} \quad (13)$$

Using property (4), the parallel form $(I_{n_1} \otimes A)$ in (13) can be modified to

$$\left(I_{n_1} \otimes A\right) = P_{3n_1,n_1}\left(A \otimes I_{n_1}\right) P_{2n_1,2}. \quad (14)$$

Also, from property (4), we can write the term $(A \otimes I_{3n_1/2})$ in the form

$$\begin{aligned}\left(A \otimes I_{3n_1/2}\right) &= \left(\left(A \otimes I_{n_1/2}\right) \otimes I_3\right) \\ &= P_{9n_1/2,3n_1/2}\left(I_3 \otimes A \otimes I_{n_1/2}\right) P_{3n_1,3}. \end{aligned} \quad (15)$$

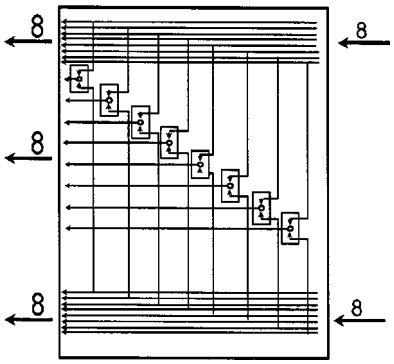

Fig. 3. The realization of Q_{16}.

Substituting (14) and (15) in (13), we have

$$\begin{aligned}\tilde{\tilde{Q}}_{n_1,n_2} = \left(P_{9(n_1/2),3}\left(P_{9n_1/2,3n_1/2}\left(I_3 \otimes A \otimes I_{n_1/2}\right) P_{3n_1,3}\right)\right. \\ \left. \times \left(P_{3n_1,n_1}\left(A \otimes I_{n_1}\right) P_{2n_1,2}\right)\right) \otimes I_{n_2/2}. \end{aligned} \quad (16)$$

However, from property (5), we have

$$\begin{aligned}P_{9n_1/2,3} \times P_{9n_1/2,3n_1/2} &= I_{9n_1/2} \\ P_{3n_1,3} \times P_{3n_1,n_1} &= I_{3n_1}. \end{aligned} \quad (17)$$

Substituting (17) in (16), we can write $\tilde{\tilde{Q}}_{n_1,n_2}$ in the form

$$\begin{aligned}\tilde{\tilde{Q}}_{n_1,n_2} &= \left(\left(I_3 \otimes A \otimes I_{n_1/2}\right)\left(A \otimes I_{n_1}\right) P_{2n_1,2}\right) \otimes I_{n_2/2} \\ &= \left(\left(I_3 \otimes Q_{n_1}\right) Q_{2n_1} P_{2n_1,2}\right) \otimes I_{n_2/2}. \end{aligned} \quad (18)$$

Using property (4), we can write $\tilde{\tilde{Q}}_{n_1,n_2}$ in the final form

$$\begin{aligned}\tilde{\tilde{Q}}_{n_1,n_2} &= P_{9n_1/2 \cdot n_2/2,9n_1/2}\left[I_{n_2/2} \otimes \left(\left(I_3 \otimes Q_{n_1}\right) Q_{2n_1} P_{2n_1,2}\right)\right] \\ &\quad \times P_{n_1 \cdot n_2,n_2/2} \\ &= P_{9n_1/2 \cdot n_2/2,9n_1/2}\left(I_{n_2/2} \otimes E\right) P_{n_1 \cdot n_2,n_2/2} \end{aligned} \quad (19)$$

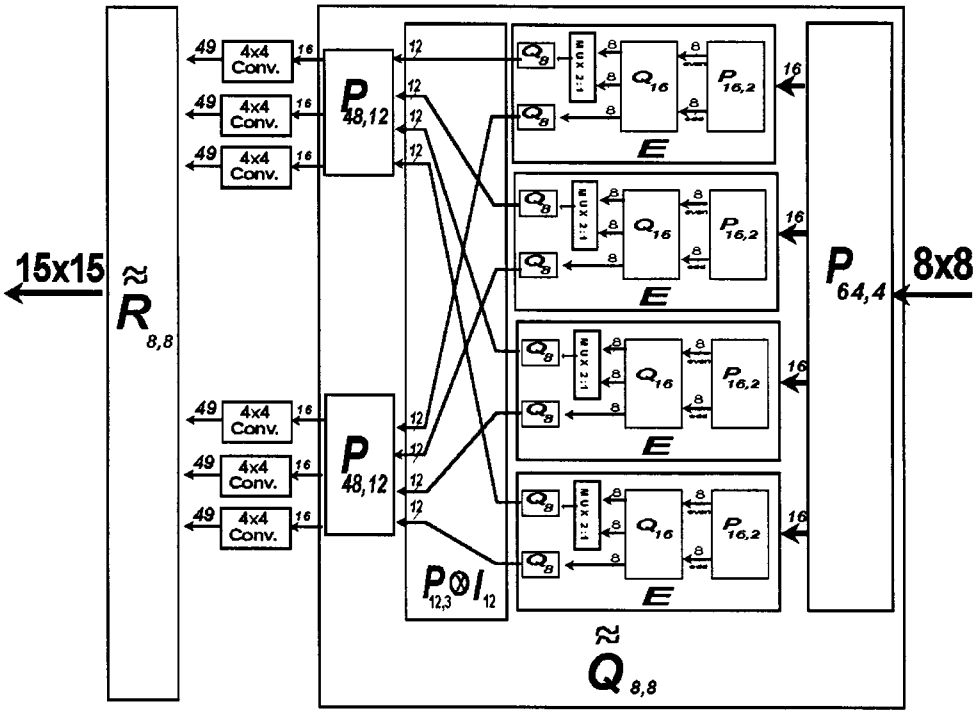

Fig. 4. The proposed 2-d convolution for an 8×8 input image with a multiplexed realization of $\tilde{\tilde{Q}}_{8,8}$.

TABLE I
COMPARISON OF THE NUMBER OF MULTIPLICATIONS REQUIRED FOR THE 2-d CONVOLUTION (KERNEL OF DIMENSION 8×8)

$n \times n$	The direct method	The FFT	The original algorithm	The proposed algorithm
8×8	162	96	104	70
64×64	1147	144	140	94
128×128	4246	156	148	99

where, $E = (I_3 \otimes Q_{n_1})Q_{2n_1}P_{2n_1,2}$. The detailed realization of $\tilde{\tilde{Q}}_{8,8}$ (for an 8×8 input image) is shown in Fig. 2 in which the computations involved in any of the four parallel E blocks are realized by the permutation $P_{16,2}$, followed by the computation stage of Q_{16}, followed by three parallel blocks of the computation Q_8. The detailed realization of Q_{16} is shown in Fig. 3. A careful scrutiny of the realizations of Q_{16} shown in Fig. 3 reveals that the data movement through Q_{16} encounters different amounts of delays. In particular, the computations involved in Q_{16} affect only the middle Q_8 in any of the E blocks (shown with dotted lines in Fig. 2). Thus, the top and the bottom Q_8 can be computed one addition cycle ahead of the middle Q_8. This means that only two of the three parallel Q_8 blocks in the realization of E are needed at a time. Therefore, through the use of multiplexers, the middle Q_8 can be removed from the architecture of E without affecting the speed of the computations as shown in Fig. 4, reducing number of the lower-order 2-d convolvers from nine to six with a computation saving of 33%.

A multiplexed architecture of $\tilde{\tilde{R}}_{n_1,n_2}$ can be also derived using a similar procedure. Applying property (4), we can write R_n in the form $R_n = R(\alpha_1)(B \otimes I_{n_1-1})$ [3]. Consequently, we can write $\tilde{\tilde{R}}_{n_1,n_2}$ in the form

$$\begin{aligned}\tilde{\tilde{R}}_{n_1,n_2} &= (R_{n_1} \otimes R_{n_2})\left(P_{9(n_1-1),3(n_1-1)} \otimes I_{n_2-1}\right) \\ &= [(R(\alpha_1)(B \otimes I_{n_1-1})) \otimes (R(\alpha_2)(B \otimes I_{n_2-1}))] \\ &\quad \times \left(P_{9(n_1-1),3(n_1-1)} \otimes I_{n_2-1}\right) \end{aligned} \tag{20}$$

which, using property (1), can be modified to

$$\begin{aligned}\tilde{\tilde{R}}_{n_1,n_2} &= (R(\alpha_1) \otimes R(\alpha_2))\left((B \otimes I_{n_1-1}) \otimes (B \otimes I_{n_2-1})\right) \\ &\quad \times \left(P_{9(n_1-1),3(n_1-1)} \otimes I_{n_2-1}\right). \end{aligned} \tag{21}$$

Applying properties (1) and (2), we have

$$\begin{aligned}\tilde{\tilde{R}}_{n_1,n_2} &= (R(\alpha_1) \otimes R(\alpha_2))\left((B \otimes I_{n_1-1} \otimes B) \otimes I_{n_2-1}\right) \\ &\quad \times \left(P_{9(n_1-1),3(n_1-1)} \otimes I_{n_2-1}\right) \\ &= (R(\alpha_1) \otimes R(\alpha_2))\left(\left(B \otimes I_{3(n_1-1)}\right)\left(I_{3(n_1-1)} \otimes B\right)\right. \\ &\quad \left. \times\ P_{9(n_1-1),3(n_1-1)} \otimes I_{n_2-1}\right) \end{aligned} \tag{22}$$

which, using property (4), can be reformulated in the parallel form

$$\begin{aligned}\tilde{\tilde{R}}_{n_1,n_2} &= (R(\alpha_1) \otimes R(\alpha_2))P_{9(n_1-1)(n_2-1),9(n_1-1)} \\ &\quad \times \left(I_{n_2-1} \otimes \left(B \otimes I_{3(n_1-1)}\right)\left(I_{3(n_1-1)} \otimes B\right)\right. \\ &\quad \left. \times\ P_{9(n_1-1),3(n_1-1)}\right) P_{9(n_1-1)(n_2-1),(n_2-1)} \\ &= (R(\alpha_1) \otimes R(\alpha_2))P_{9(n_1-1)(n_2-1),9(n_1-1)} \\ &\quad \times (I_{n_2-1} \otimes K)\, P_{9(n_1-1)(n_2-1),(n_2-1)} \end{aligned} \tag{23}$$

where $K = (B \otimes I_{3(n_1-1)})(I_{3(n_1-1)} \otimes B)P_{9(n_1-1),3(n_1-1)}$. Equation (23) represents $\tilde{\tilde{R}}_{n_1,n_2}$ in a multiplexed form similar to that of (19) for $\tilde{\tilde{Q}}_{n_1,n_2}$.

We can extend the derivation of the multiplexed 2-d convolution algorithm presented in this section to the m-d convolution algorithm in [3]. Following the same steps that are used to modify $\tilde{\tilde{Q}}_{n_1,n_2}$ and

$\tilde{\hat{R}}_{n_1,n_2}$ for the 2-d convolution, we can exploit the resource sharing available and derive multiplexed forms for the m-d pre-additions $(\hat{Q})$ and the post-additions $(\hat{R})$ in [3], reducing the number of lower-order m-d convolvers form 3^m to $(2/3)3^m$.

III. The Computational Complexity of the 2-d Convolution Algorithm

For simplicity, assume that $n_1 = n_2 = n$ and the input image is of size $n \times n$. From (19), the number of additions in the pre-additions stage $\tilde{\hat{Q}}_{n_1,n_2}$ is given by

$$\frac{n_2}{2}\left[3\left(\frac{n_1}{2}\right)+n_1\right]=\frac{5}{4}n^2. \tag{24}$$

Using property (6), we can write the term $R_{n_1} \otimes R_{n_2}$ in (20) as [2]

$$\begin{aligned} &R_{(2n-1)\times 3(n-1)} \otimes R_{(2n-1)\times 3(n-1)} \\ &\quad = P_{(2n-1)^2,(2n-1)}\left(I_{(2n-1)} \otimes R\right) P_{(2n-1)3(n-1),(2n-1)} \\ &\qquad \times \left(I_{3(n-1)} \otimes R\right). \end{aligned} \tag{25}$$

Therefore, the number of additions in the post-additions stage $\tilde{\hat{R}}_{n_1,n_2}$ is given by

$$\begin{aligned} &(2n-1)\ (\text{The number of additions in } R) \\ &\qquad + 3(n-1)\ (\text{The number of additions in } R) \\ &\quad = (5n-4)(3n-4). \end{aligned} \tag{26}$$

It should be mentioned that the number of additions in both the pre-addition and post-addition stages remains unchanged in the proposed algorithm. Moreover, the communication complexity is reduced by removing one of the three $P_{48,12}$ blocks, as shown in Fig. 4, for the case $n = 8$.

Since the multiplication stages are centered at the core lower-order parallel blocks [2], [3], removing one-third of these blocks in the proposed algorithm guarantees a 33% saving in the number of multiplications. It should be mentioned that the computation complexity in our proposed algorithm depends on the computations involved in the core computations $C_{n_1/2,n_2/2} = C(n_1/2) \otimes C(n_2/2)$. The number of multiplications in our proposed algorithm based on [1] as a core, compared to the direct method and FFT method, is shown in Table I. The table shows a significant reduction of the number of multiplications of the proposed multiplexed algorithm over the direct and FFT methods.

IV. Conclusion

In this paper, we presented modified parallel architectures for m-d convolution. The proposed algorithm showed that for 2-d convolution, the number of lower-order convolutions is reduced from nine to six with a computation saving of 33%. The proposed partitioning strategy results in a core of data-independent convolution computations, and does not make any assumptions on how the core convolutions are computed. Indeed, any suitable convolution method can be used, which makes the proposed method very flexible and realizable over a wide range of hardware and software platforms.

Acknowledgment

The authors would like to thank the anonymous reviewers for their valuable comments and suggestions.

References

[1] P. C. Balla, A. Antoniou, and S. D. Morgera, "Higher radix aperiodic convolution algorithms," *IEEE Trans. Acoust., Speech, Signal Processing*, vol. 34, Jan. 1989.

[2] A. Elnaggar, H. M. Alnuweiri, and M. R. Ito, "A new tensor product formulation for Toom's convolution algorithm," *IEEE Trans. Signal Processing*, vol. 47, Apr. 1999.

[3] ——, "A new recursive algorithm for multidimensional convolution," *IEEE Trans. Circuits Syst. II*, vol. 46, May 1999.

[4] V. Hecht, K. Ronner, and P. Pirsch, "An advanced programmable 2-D-convolution chip for real-time image processing," in *Proc. ISCAS'91*.

[5] X. Liu and L. T. Bruton, "High-speed systolic ladder structures for MD recursive digital filters," *IEEE Trans. Signal Processing*, vol. 44, Apr. 1996.

[6] R. Tolimieri, M. An, and C. Lu, *Algorithms for Discrete Fourier Transform and Convolution*. New York: Springer-Verlag, 1989.

New Algorithm for Multidimensional Type-III DCT

Yonghong Zeng, Guoan Bi, and Alex C. Kot

***Abstract*—New algorithms are proposed for the type-III multidimensional discrete cosine transform (MD-DCT-III). The polynomial transform is used to convert the type-III MD-DCT into a series of one-dimensional type-III discrete cosine transforms (1-D-DCT-III). The algorithms achieve considerable savings on the number of operations compared to the row-column method. For computing an r-dimensional DCT-III, the number of multiplications required by the proposed algorithm is only $1/r$ times that needed by the row-column method, and the number of additions is also reduced. Compared to other known fast algorithms for two-dimensional- and MD-DCTs, the proposed one uses about the same number of operations. However, advantages such as better computational structure and flexibility on the choice of dimensional sizes can be achieved.**

***Index Terms*—Discrete cosine transform, fast algorithm, multidimensional signal processing, polynomial transform.**

I. Introduction

Discrete cosine transform has a wide range of applications, such as data compression, feature extraction, image reconstruction and multiframe detection. For example, the three-dimensional discrete cosine transform (3-D-DCT) coding is an alternative approach to the motion compensation transform coding technique used in video coding standards [7]. The multidimensional transforms are also used in the areas of computer vision, high definition television (HDTV), and vision telephone to process or analyze motion images (i.e., multiframe detection) [6], [7]. For example, the four-dimensional DCT is generally required for three-dimensional motion images. Although modern technologies have increased computing speed dramatically over the recent years, there still exist many difficulties in processing multidimensional signals at a throughput required by most practical applications. A good fast algorithm is extremely important to cope with the prohibitive computational complexity of the multidimensional transform.

Recently, several fast algorithms (other than the well-known row-column method) have been proposed for two-dimensional discrete cosine transform (2-D-DCT) or multidimensional discrete cosine (MD-DCT). Among them, the polynomial transform (PT) based algorithms for 2-D-DCT in [2] and [3], the algorithms in [4] and [5], and the algorithm for MD-DCT in [1] are reported to offer savings on the required

Manuscript received December 1999; revised August 2000. This paper was recommended by Associate A. Skodras.

The authors are with the School of Electrical and Electronics Engineering, Nanyang Technological University, Singapore (e-mail: eyhzeng@ntu.edu.sg; egbi@ntu.edu.sg; eackot@ntu.edu.sg).

Publisher Item Identifier S 1057-7130(00)11026-2.