AN EFFICIENT ALGORITHM FOR MULTIDIMENSIONAL CONVOLUTION

A. Elnaggar

Department of Electrical Engineering
Sultan Qaboos University,
Muscat, Oman 123

M. Aboelaze

Department of Computer Science,
York University,
Toronto, Canada M3J 1P3

ABSTRACT

This paper presents a novel recursive algorithm for generating higher-order multidimensional (or m-D for short) convolution by combining the computation of 3^m identical lower-order (smaller size) convolution computations, and its implementation in parallel VLSI networks [2]. We show that for 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 35% [3]. The resulting VLSI architectures have very simple modular structure, highly regular topology, and use simple arithmetic units.

1. INTRODUCTION

Convolution is a very important operation in signal and image processing with applications to digital filtering, and video image processing. Thus, abundant approaches have been suggested to achieve high speed processing for linear convolution and to design efficient convolution architectures [4]-[11]. Most of these methods speed up the computations by parallelizing the computation of linear convolution based on direct strategies for parallelizing serial convolution. Such networks are not efficient for achieving high degree of parallelism.

The main result reported in this paper shows that for an $n \times m$ input image, a two dimensional (2-D) convolution computation, $C(n,m)$, can be decomposed recursively into three cascaded stages as shown in Fig. 1 [2]. The center stage is constructed recursively from 9 parallel (data-independent) blocks each realizing a smaller-size 2-D convolution ($C(n/2, m/2)$). The post-additions and the pre-additions stages serve as "glue" circuits that combine the 9 lower order convolution blocks to construct the higher order convolution architecture.

Our methodology is based on a non-trivial generalization of the 1-D convolution algorithm proposed in [1]. We show that the proposed algorithm can be extended such that an m-D convolution can be constructed from 3^m smaller size m-D convolutions. Our methodology employs tensor product (or Kronecker products) decompositions and permutation matrices as the main tools for expressing the proposed algorithm [4], [10]. We employ several techniques to manipulate such decompositions into suitable recursive expressions that can be mapped efficiently onto VLSI structures.

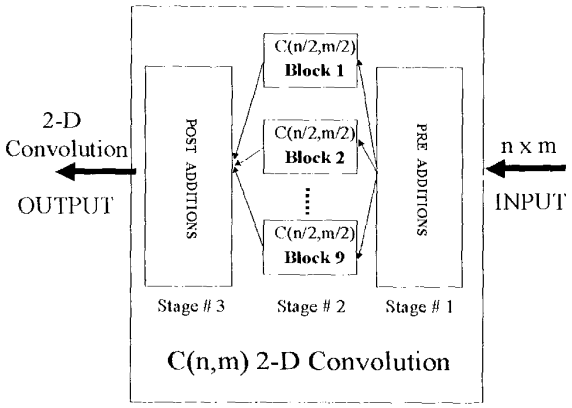

Fig. 1 The proposed 2-D convolution architecture

2. THE PROPOSED RECURSIVE ALGORITHMS

2.1 TOOM'S LINEAR CONVOLUTION ALGORITHM

In this section, we will briefly review the 1-D recursive formulation of Toom's convolution algorithm given in [1]. Let x_n and h_n be two sequences of length n, the linear convolution $y_{2n-1} = h_n * x_n$ is given by the matrix form

$$y_{2n-1} = C(n) X_n \tag{1}$$

For $n = 2^\alpha$, where α is an integer, the recursive form of Toom's algorithm is given by [1]

$$C(n) = R_n (I_3 \otimes C(n/2)) Q_n \tag{2}$$

where

$$Q_n = \left(P_{3.2^{\alpha-1},3.2^{\alpha-2}} \right)^{\alpha-1} \left[\left(I_{2^{\alpha-1}} \otimes A \right) P_{2^\alpha,2^{\alpha-1}} \right], \tag{3}$$

$$R_n = R(\alpha) \left[P_{3(2^\alpha-1),3} (I_{2^\alpha-1} \otimes B) P_{3(2^\alpha-1),(2^\alpha-1)} \right], \tag{4}$$

$$A = \begin{bmatrix} 1 & 0 \\ 1 & 1 \\ 0 & 1 \end{bmatrix}, \quad B = \begin{bmatrix} 1 & 0 & 0 \\ -1 & 1 & -1 \\ 0 & 0 & 1 \end{bmatrix} \tag{5}$$

0-7803-6293-4/00/$10.00 ©2000 IEEE.

Here, $A \otimes B$ represents the tensor products of the two matrices A and B, if $n = r\,s$ then $P_{n,s}$ is an $n \times n$ binary matrix specifying an n/s-shuffle (or s-stride) permutation. $R(\alpha)$ is a special matrix of 1's and 0's [1], [4]. Note that, Q_n and R_n represent the pre-additions and the post-additions stages, respectively, and they serve as glue structures that combine three identical lower order convolution architectures $(C(n/2))$ to construct the higher order 1-D convolution $(C(n))$ as shown in Fig. 2.

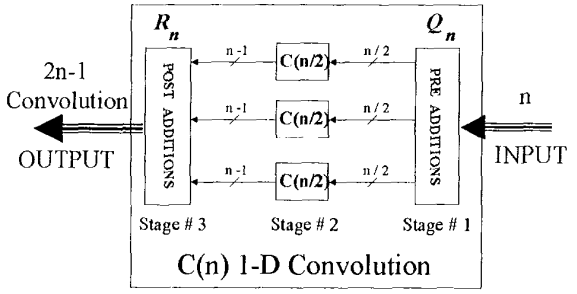

Fig. 2 The 1-D recursive convolution architecture

Observe that, we have drawn our networks such that data flows from right to left. We chose this convention to show the direct correspondence between the derived algorithms and the proposed VLSI networks.

2.2 THE 2-D CONVOLUTION ALGORITHM

For an $n_1 \times n_2$ input data image, Pratt [9] has shown that the 2-D convolution output is given by

$$p = C_{n_1,n_2} f \tag{6}$$

where, C_{n_1,n_2} is the 2-D convolution transform matrix; and p and f are the output and input column-scanned vectors, respectively of size $n = n_1 n_2$ each. Pratt has also shown that, for separable transforms, the matrix C_{n_1,n_2} can be decomposed into the tensor form

$$C_{n_1,n_2} = C(n_1) \otimes C(n_2) \tag{7}$$

where $C(n_1)$ and $C(n_2)$ represent row and column 1-D convolution operators on f, respectively, as defined in (2). Substituting (2) in (7), we can write the 2-D convolution transform matrix in the form

$$\begin{aligned} C_{n_1,n_2} = &\left[R_{n_1}\left(I_3 \otimes C(n_1/2)\right)Q_{n_1}\right] \otimes \\ &\left[R_{n_2}\left(I_3 \otimes C(n_2/2)\right)Q_{n_2}\right] \end{aligned} \tag{8}$$

Applying the distributive and accumulative properties of the tensor products leads to [2], [4], [10]

$$\begin{aligned} C_{n_1,n_2} = &\left[R_{n_1} \otimes R_{n_2}\right]\left[\left(I_3 \otimes C(n_1/2)\right) \otimes \left(I_3 \otimes C(n_2/2)\right)\right] \otimes \\ &\left[Q_{n_1} \otimes Q_{n_2}\right]. \\ = &\tilde{R}\,\tilde{C}_{n_1,n_2}\,\tilde{Q} \end{aligned} \tag{9}$$

where

$$\tilde{R} = \left[R_{n_1} \otimes R_{n_2}\right], \tag{10}$$

$$\tilde{C}_{n_1,n_2} = \left[\left(I_3 \otimes C(n_1/2)\right) \otimes \left(I_3 \otimes C(n_2/2)\right)\right], \tag{11}$$

$$\tilde{Q} = \left[Q_{n_1} \otimes Q_{n_2}\right]. \tag{12}$$

Note that the matrix $\tilde{C}_{n_1,n_2}$ in (11) contains the lower-order 1-D convolutions matrices $C(n_1/2)$ and $C(n_2/2)$ in an involved tensor product expression. Our next task is to manipulate $\tilde{C}_{n_1,n_2}$ to derive a more efficient lower-order 2-D expression. We can rewrite $\tilde{C}_{n_1,n_2}$ as

$$\begin{aligned} \tilde{C}_{n_1,n_2} &= \left[\left(I_3 \otimes C(n_1/2)\right) \otimes I_3\right) \otimes \left(C(n_2/2)\right)\right], \\ &= \left[P_{9(n_1-1),3(n_1-1)}\left(I_9 \otimes C(n_1/2)\right)P_{9(n_1/2),3}\right] \otimes \\ &\quad C(n_2/2) \end{aligned} \tag{13}$$

Since the convolution matrix $C(n_2/2)$ is of dimension $[(n_2-1) \times n_2/2]$, we can write it as

$$C(n_2/2) = I_{n_2-1}\,C(n_2/2)\,I_{n_2/2} \tag{14}$$

substituting (14) in (13), we have

$$\begin{aligned} \tilde{C}_{n_1,n_2} &= \left[P_{9(n_1-1),3(n_1-1)}\left(I_9 \otimes C(n_1/2)\right)P_{9(n_1/2),3}\right] \otimes \\ &\quad \left[I_{n_2-1} C(n_2/2) I_{n_2/2}\right]. \\ &= \left[P_{9(n_1-1),3(n_1-1)} \otimes I_{n_2-1}\right] \\ &\quad \left[\left(I_9 \otimes C(n_1/2)\right) \otimes C(n_2/2)\right] \\ &\quad \left[P_{9(n_1/2),3} \otimes I_{n_2/2}\right]. \end{aligned} \tag{15}$$

Now, substituting (15) in (9) gives

$$C_{n_1,n_2} = \tilde{\tilde{R}}\left(I_9 \otimes C_{n_1/2,n_2/2}\right)\tilde{\tilde{Q}} \tag{16}$$

where

$$C_{n_1/2,n_2/2} = C(n_1/2) \otimes C(n_2/2), \tag{17}$$

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

$$\tilde{\tilde{Q}} = \left(P_{9(n_1/2),3} \otimes I_{n_2/2}\right)\left(Q_{n_1} \otimes Q_{n_2}\right), \tag{18}$$

and

$$\tilde{\tilde{R}} = \left(R_{n_1} \otimes R_{n_2}\right)\left(P_{9(n_1-1),3(n_1-1)} \otimes I_{n_2-1}\right) \tag{19}$$

are the new 2-D pre- and post-additions, respectively and they serve as glue structures that combine 9 identical lower order

convolution architectures $\left(C(n_1/2)\otimes C(n_2/2)\right)$ to construct the higher order 2-D convolution $\left(C(n_1)\otimes C(n_2)\right)$ as shown previously in Fig. 1.

2.3 THE m-D CONVOLUTION ALGORITHM

We can extend the derivation of the 2-D convolution algorithm in section 2.2 to the m-D case, for any $\boldsymbol{m} \succ 2$. From (6) and (7), we can write the m-D convolution in the form

$$\begin{aligned} p &= \left(C(n_1)\ \otimes\ C(n_2)\ \otimes\ \cdots\ \otimes\ C(n_m)\right) f \\ &= \left(\bigotimes_{i=1}^{m} C(n_i)\right) f \quad . \end{aligned} \tag{20}$$

Following the derivation steps in section 2.2, we can write (20) in the form

$$p = \left[\hat{R}\left(I_{3^m} \bigotimes_{i=1}^{m} C(n_i/2)\right)\hat{Q}\right] f \tag{21}$$

where,

$$\begin{aligned} \hat{Q} &= \left[\prod_{i=1}^{m-1}\left(P_{u_1,u_2}\bigotimes_{k=1}^{i} I_{u_3}\right)\right]\left(\bigotimes_{i=1}^{m} Q_i\right) \quad , \\ \hat{R} &= \left(\bigotimes_{k=1}^{m} R_k\right)\left[\prod_{i=1}^{m-1}\left(P_{w_1,w_2}\bigotimes_{l=i}^{m-1} I_{w_3}\right)\right] \quad , \\ u_1 &= 3^{m-i+1}\prod_{j=1}^{m-i}\left(n_j/2\right) \quad , \qquad u_2 = 3 \quad , \\ w_1 &= 3^{i+1}\prod_{j=1}^{i}\left(n_j-1\right) \quad , \qquad w_2 = 3i\prod_{k=1}^{i}\left(n_k-1\right) \quad , \end{aligned} \tag{22}$$

Q_i and R_k are defined in (3) and (4), respectively. Equation (20) represents the recursive m-D convolution computation. In this case, we use 3^m of the lower-order architectures in parallel. Similar analysis to that of deriving $\tilde{\tilde{Q}}$ and $\tilde{\tilde{R}}$ in section 2.2 can be applied to $\hat{Q}$ and $\hat{R}$, respectively [2].

3. REDUCING THE COMPLEXITY OF THE *m*-D CONVOLUTION ALGORITHM

Now, we will further manipulate (3) and (4) to exploit the resource sharing available and consequently, realize the multiplexed architecture of the 2-D convolution using less number of the lower-order convolvers.

Applying the parallel-serial properties on (3), we can write Q_n in the simplified form

$$Q_n = A \otimes I_{n/2} \tag{23}$$

From (18) and (23), we can rewrite $\tilde{\tilde{Q}}_{n_1,n_2}$ as

$$\begin{aligned} \tilde{\tilde{Q}}_{n_1,n_2} &= \left(P_{9(n_1/2),3}\otimes I_{n_2/2}\right)\left(Q_{n_1}\otimes Q_{n_2}\right) \\ &= \begin{bmatrix} \left(P_{9(n_1/2),3}\otimes I_{n_2/2}\right) \\ \left[A\otimes I_{n_1/2}\otimes A\otimes I_{n_2/2}\right] \end{bmatrix} \end{aligned} \tag{24}$$

Also, from the distributive property of tensor-product, we can write (24) in the form [2]

$$\begin{aligned} \tilde{\tilde{Q}}_{n_1,n_2} &= \left(P_{9(n_1/2),3}\left(A\otimes I_{n_1/2}\otimes A\right)\right)\otimes \\ &\quad \left(I_{n_2/2}\ I_{n_2/2}\right) \\ &= \left(P_{9(n_1/2),3}\left(A\otimes I_{n_1/2}\otimes A\right)\right)\otimes I_{n_2/2} \end{aligned} \tag{25}$$

But from (5), the matrix A is of dimension 3×2. Therefore, we have

$$\begin{aligned} A &= A I_2 \quad , \\ \left(I_{n_1/2}\otimes A\right) &= I_{3n_1/2}\left(I_{n_1/2}\otimes A\right) \end{aligned} \tag{26}$$

Substituting (26) in (25) then using the distributive and the associative properties [2], we have

$$\begin{aligned} \tilde{\tilde{Q}}_{n_1,n_2} &= \left(P_{9(n_1/2),3}\left(\left(AI_2\ \otimes\ I_{3n_1/2}\right)\left(I_{n_1/2}\otimes A\right)\right)\right)\otimes \\ &\quad I_{n_2/2} \\ &= \left(P_{9(n_1/2),3}\left(A\otimes I_{3n_1/2}\right)\left(I_{n_1}\otimes A\right)\right)\otimes I_{n_2/2} \end{aligned} \tag{27}$$

Using the parallel-serial property [2], the parallel form $\left(I_{n_1}\otimes A\right)$ in (27) can be modified to

$$\left(I_{n_1}\otimes A\right) = P_{3n_1,n_1}\left(A\otimes I_{n_1}\right)P_{2n_1,2} \tag{28}$$

Substituting (28) in (27), we have

$$\begin{aligned} \tilde{\tilde{Q}}_{n_1,n_2} &= P_{9(n_1/2),3}\left(P_{9n_1/2,3n_1/2}\left(I_3\otimes A\otimes I_{n_1/2}\right)P_{3n_1,3}\right) \\ &\quad \left(P_{3n_1,n_1}\left(A\otimes I_{n_1}\right)P_{2n_1,2}\right)\otimes I_{n_2/2} \end{aligned} \tag{29}$$

But since

$$\begin{aligned} P_{9n_1/2,3}\times P_{9n_1/2,3n_1/2} &= I_{9n_1/2} \quad , \\ P_{3n_1,3}\times P_{3n_1,n_1} &= I_{3n_1} \end{aligned} \tag{30}$$

Substituting (30) in (29), we can write $\tilde{\tilde{Q}}_{n_1,n_2}$ in the form

$$\tilde{\tilde{Q}}_{n_1,n_2} = P_{9n_1/2,n_2/2,9n_1/2}\left(I_{n_2/2}\otimes E\right)P_{n_1,n_2,n_2/2} \tag{31}$$

where, $E = \left(I_3\otimes Q_{n_1}\right)Q_{2n_1}P_{2n_1,2}$.

The detailed realization of $\tilde{\tilde{Q}}_{8,8}$ (for an 8×8 input image) is shown in Fig. 3 in which the computations involved in any of the 4 parallel E blocks are realized by the permutation $P_{16,2}$, followed by the computation stage of Q_{16}, followed by 3 parallel blocks of the computation Q_8. The detailed realization of Q_{16} is shown in Fig. 4.

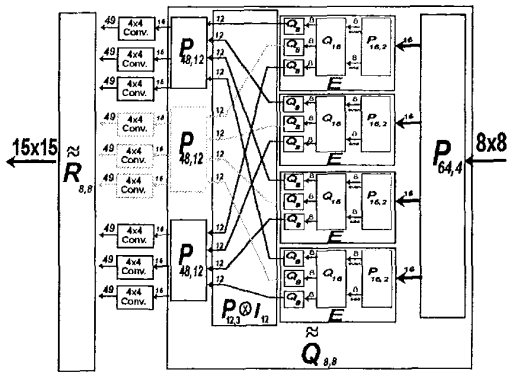

Fig.3 The detailed realization of $\tilde{\tilde{Q}}_{8,8}$ for an 8×8 input.

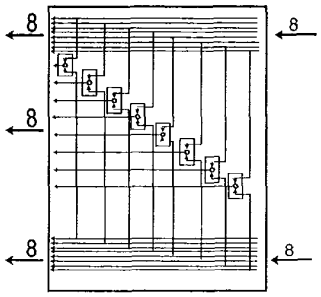

Fig. 4 The detailed realization of Q_{16}

A careful scrutiny of the realizations of Q_{16} shown in Fig. 4 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 gray color in Fig. 3). Thus, the top and the bottom Q_8 can be computed one addition cycle ahead of the middle Q_8. 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. 5, reducing number of the lower-order 2-D convolvers (4×4 in our case) from 9 to 6 with a computation saving of 35%.

A multiplexed architecture of $\tilde{\tilde{R}}_{n_1,n_2}$ can be also derived using a similar procedure [3].

4. CONCLUSIONS

In this paper, we presented a class of highly modular parallel VLSI architectures for multi-dimensional convolution. We showed that the multidimensional convolution is generated recursively by combining the computation of 3^m identical lower-order (smaller size) convolution computations. We showed also that, the number of lower-order 2-D convolutions can be reduced from nine to six with a computation saving of 35%. Moreover, the original speed of the computations is not affected.

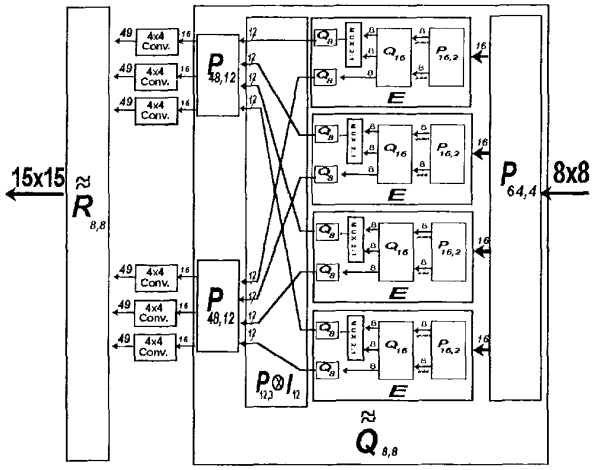

Fig. 5 The multiplexed realization of $\tilde{\tilde{Q}}_{8,8}$ for an 8×8 input.

5. REFERENCES

[1] A. Elnaggar, H. M. Alnuweiri, M. R. Ito, "A New Tensor Product Formulation for Toom's Convolution Algorithm," *IEEE Transactions on signal Processing*, Vol. 47, No. 4, April 1999.

[2] A. Elnaggar, H. M. Alnuweiri, M. R. Ito, "A New Recursive Algorithm for Multidimensional Convolution," *IEEE Transactions on Circuits and Systems-Part II: Analog and Digital Signal Processing*, Vol. 46, No. 5, May 1999.

[3] A. Elnaggar, M. Aboelaze, "An Efficient Architecture for Multidimensional Convolution," in review, *IEEE Transactions on Circuits and Systems-Part II: Analog and Digital Signal Processing.*

[4] J. Granata, M. Conner, R. Tolimieri, "Recursive Fast Algorithms and the Role of the Tensor Product," *IEEE Trans. On Signal Processing*, Vol. 40, No. 12, December 1992.

[5] I-S. Lin, S. K. Mitra, "Fast FIR Filtering Algorithms Based on Overlapped Block Structure," *Proc. of ISCAS'93*, 1993.

[6] F. Marino, "A Fixed-Point Parallel Convolver without Precision Loss for the Real-Time Processing of Long Numerical Sequences," *Proc. of the IEEE 7th Symposium on Parallel and Distributed Processing*, 1995.

[7] Z. J. Mou, P. Duhamel, "A Unified Approach to the Fast FIR Filtering Algorithms," *Proc. of ICAASP'88*, 1988.

[8] S. Muramatsu, H. Kiya, "Multidimensional Parallel Processing Methods for Rational Lattice Alteration," *Proc. of ISCAS'95*, 1995.

[9] W. K. Pratt, "*Digital Image Processing*," John Wiley & Sons, Inc., 1991.

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

[11] M. Vetterli, "Running FIR and IIR Filtering Using Multirate Filter Banks," *IEEE Trans. on ASSP*, Vol. 36, No. 5, 1988.