An Improved Toom's Algorithm for Linear Convolution

A. Elnaggar, *Associate Member, IEEE,* and M. Aboelaze

***Abstract*—This letter presents an improved Toom's algorithm that allows hardware savings without slowing down the processing speed. We derive formulae for the number of multiplications and additions required to compute the linear convolution of size $n = 2^{\alpha}$. We demonstrate the computational advantage of the proposed improved algorithm when compared to previous algorithms, such as the original matrix–vector multiplication and the FFT algorithms.**

***Index Terms*—Convolution, permutation matrices, shuffle networks, tensor products, Toom's algorithm.**

I. Introduction

CONVOLUTION is a very important operation in signal and image processing with applications to digital filtering and video image processing. Many approaches have been suggested to achieve high-speed processing for linear convolution and to design efficient convolution architectures [1]–[5].

The proposed work is based on a nontrivial modification of the one-dimensional (1-D) convolution algorithm presented in [1] and shown in Fig. 1. Using an alternative (permutation-free) construction, we show that the number of lower order parallel convolutions (Stage #2 in Fig. 1) can be reduced from three to only two, while keeping the regular topology and simple data flow of the original very large scale integration (VLSI) architecture. Our methodology employs tensor-product decompositions and permutation matrices as the main tools for expressing DSP algorithms.

Let x_n and h_n be two sequences of length $n = 2^{\alpha}$. The linear convolution $y_{2n-1} = h_n * x_n$ in matrix form is given by $Y_{2n-1} = C(n)x_n$, where $C(n)$ is the convolution matrix defined by [1]

$$C(n) = R_n \tilde{D}(n) Q_n \tag{1}$$

where

$$\tilde{D}(n) = \mathrm{diag}[Q_n h_n] \tag{2}$$

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

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

Manuscript received June 13, 2001; revised May 6, 2002. The associate editor coordinating the review of this manuscript and approving it for publication was Dr. Geert Leus.

A. Elnaggar is with the Department of Information Engineering, Sultan Qaboos University, Muscat, Oman 123 (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 10.1109/LSP.2002.801718.

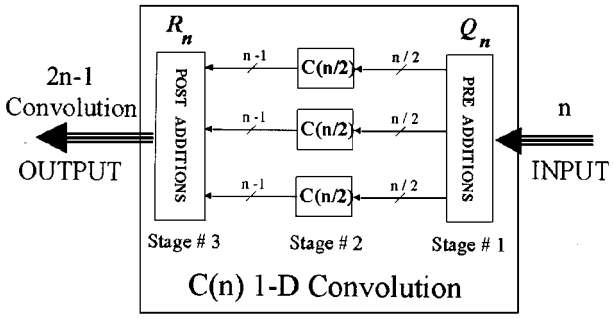


Fig. 1. Original realization of the 1-D convolution algorithm.

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

$P_{n,m}$ represents a shuffle permutation matrix on n inputs with stride m, and $\otimes$ represents the tensor-product [6]. Equation (1) can be realized by cascading the three stages: preaddition stage Q_n followed by one stage of multiplications $\tilde{D}(n)$, followed by a postaddition stage R_n as shown in Fig. 1 [1].

II. Improved Preaddition and Postaddition

Using the tensor-product property $P_{n,n_1}P_{n,n_2} = P_{n,n_1n_2}$, where $n = n_1n_2n_3$ [4]–[6], the term $(P_{3(2^{\alpha-1}),\,3(2^{\alpha-2})})$ contained in (3) can be simplified to

$$P_{3(2^{\alpha-1}),\,3(2^{\alpha-2})} = P_{3(2^{\alpha-1}),\,3} \prod_{i=1}^{\alpha-2} P_{3(2^{\alpha-1}),\,2}. \tag{6}$$

Since the tensor-product is commutative and from (6), then

$$\begin{aligned} &\left(P_{3(2^{\alpha-1}),\,3(2^{\alpha-2})}\right)^{\alpha-1} \\ &\quad = \left(P_{3(2^{\alpha-1}),\,3} \prod_{i=1}^{\alpha-2} P_{3(2^{\alpha-1}),\,2}\right) \left(P_{3(2^{\alpha-1}),\,3(2^{\alpha-2})}\right)^{\alpha-2} \\ &\quad = P_{3(2^{\alpha-1}),\,3} \left(P_{3(2^{\alpha-1}),\,3(2^{\alpha-2})}\right)^{\alpha-2}. \end{aligned} \tag{7}$$

But since $P_{n,n_1}P_{n,n_2} = I_n$ where $n = n_1n_2$

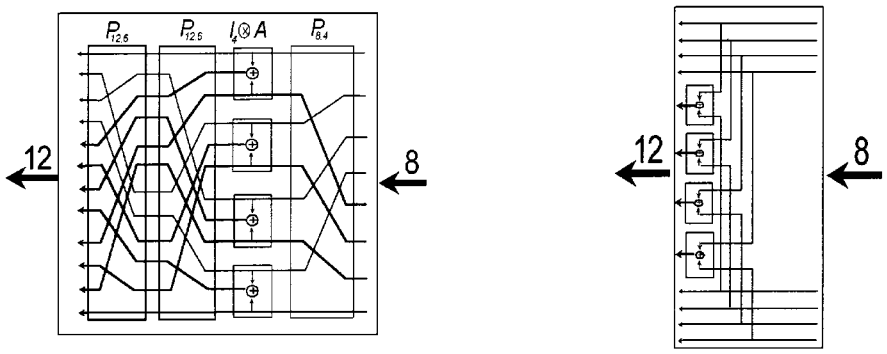
$$P_{3(2^{\alpha-1}),\,2} \cdot P_{3(2^{\alpha-1}),\,3(2^{\alpha-2})} = I_{3(2^{\alpha-1})}. \tag{8}$$

From (7) and (8), we have

$$\left(P_{3(2^{\alpha-1}),\,3(2^{\alpha-2})}\right)^{\alpha-1} = P_{3(2^{\alpha-1}),\,3}. \tag{9}$$

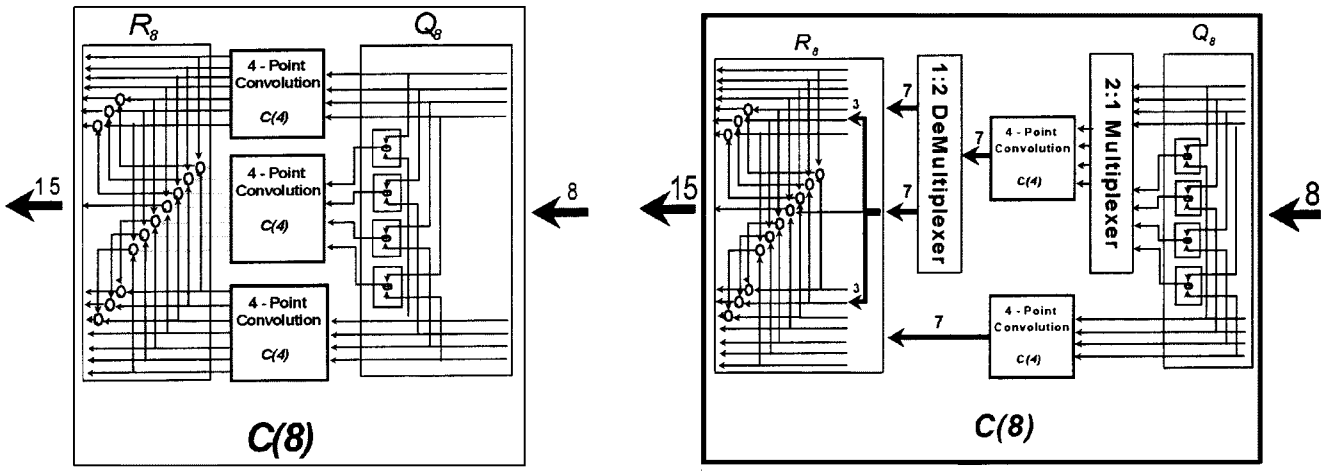
Therefore, we can write Q_n as

$$Q_n = P_{3(2^{\alpha-1}),\,3}\,(I_{2^{\alpha-1}} \otimes A)\, P_{2^{\alpha},\,2^{\alpha-1}}. \tag{10}$$



(a) The original realization. (b) The shuffle-free realization.

Fig. 2. Realization of Q_8. (a) Original realization. (b) Shuffle-free realization.



(a) The shuffle-free architecture. (b) The multiplexed architecture.

Fig. 3. Realization of eight-point convolution. (a) Shuffle-free architecture. (b) Multiplexed architecture.

By decomposing the two permutations $P_{3(2^{\alpha}-1),3}$ and $P_{2^{\alpha},2^{\alpha}-1}$ in (10) into their serial forms [6], Q_n can be simplified to

$$Q_n = A \otimes I_{2^{\alpha}-1}. \tag{11}$$

Note that the above expression for Q_n does not include any (explicit) permutations. The realization of Q_8 ($n = 2^3$) using the original form of (3) and the modified shuffle-free representation of (11) is shown in Figs. 2(a) and 3(b), respectively.

Even though the resulting circuits in Figs. 2(a) and 3(b) are topologically equivalent, removing the shuffle-permutations from the tensor formulations can simplify data movement.

Similarly, substituting $I_{2^{\alpha}-1}$ for A_{n_1} in (4) and applying similar tensor-product properties [6], R_n can be simplified to

$$\begin{aligned} R_n &= R(\alpha)\left[P_{3(2^{\alpha}-1),3}(I_{2^{\alpha}-1} \otimes B)P_{3(2^{\alpha}-1),(2^{\alpha}-1)}\right] \\ &= R(\alpha)(B \otimes I_{2^{\alpha}-1}). \end{aligned} \tag{12}$$

The permutation-free recursive realization of the eight-point convolution using three four-point convolutions is shown in Fig. 3(a). We assume that each addition requires one unit of time.

III. Multiplexed Architecture of the 1-D Convolution

A careful scrutiny of the realization shown in Fig. 3(a) reveals that the data movement through the computational stages encounters different amounts of delays. In particular, the computations involved in the Q_8 matrix affect only the middle four-point convolution in the center stage. Thus, the top and the bottom four-point convolutions can be computed one addition cycle ahead of the middle four-point convolution. This means that only two four-point convolvers are needed at a time. Therefore, through the use of a multiplexer, either the top or the bottom four-point convolver can be removed from the architecture [as shown in Fig. 3(b)], allowing hardware savings without slowing down the processing speed.

It should be mentioned that it would not be possible to observe the resource sharing shown in Fig. 3(a) without the improved shuffle-free architecture. This is evident by comparing the realizations of Q_8 in the original form [Fig. 2(a)] and the modified shuffle-free form [Fig. 2(b)].

TABLE I
A COMPARISON OF THE NUMBER OF ADDITIONS WITH OTHER APPROACHES

Method	Formula Used	$n=8$	$n=256$	$n=1024$
Direct	$n(n-1)$	56	65,280	$1.047\text{x}10^6$
FFT	$12n\log 2n$	384	27,648	$0.135\text{x}10^6$
The proposed Algorithm	$5(3^{\log n}) - 7(2^{\log n}) + 2$	81	31,015	$0.288\text{x}10^6$

TABLE II
A COMPARISON OF THE NUMBER OF MULTIPLICATIONS WITH OTHER APPROACHES

Method	Formula Used	$n=8$	$n=256$	$n=1024$
Direct	n^2	64	65,536	$1.048\text{x}10^6$
FFT	$12n\log 2n+8n$	448	29,696	$0.143\text{x}10^6$
The proposed algorithm	$3^{\log n}$	27	6,561	59,049

IV. COMPUTATION COMPLEXITY OF THE IMPROVED ALGORITHM

From (3), the number of additions required is the number of additions to compute the term $(I_{2^{\alpha-1}} \otimes A)$. The rest are permutations only. Therefore, the number of additions needed to compute the preaddition stage is given by ($n = 2^\alpha$)

$$\sum_{i=0}^{\alpha-1} (3^{\alpha-i-1})(2^i) = 3^{\alpha-1} \sum_{i=0}^{\alpha-1} (3^{-i})(2^i) = (3^\alpha - 2^\alpha). \quad (13)$$

From (4), the number of additions is the number of additions to compute the term $(I_{2^{\alpha}-1} \otimes B)$ plus the additions to compute $R(\alpha)$. The rest are permutations only. Since each of the three-input adders used in computing the matrix B can be realized by two two-input adders, the number of additions is given by

$$2\sum_{i=1}^{\alpha} 3^{i-1}(2^{\alpha-i+1} - 1). \quad (14)$$

The matrix $R(\alpha)$ is a special matrix of zeros and ones and has the property that the number of one-entries per row is either one or two; all other entries are zeros. Moreover, the number of one-entries per column is exactly one; all other entries are zeros. Using these properties, we can derive modular realizations for the matrix $R(\alpha)$ at different stages. For a convolution of size $n = 2^\alpha$, the coordinates of the one-entries are

$$[i(2^\alpha - 1) + j,\ i(2^\alpha - 1) + j]$$

where, $i = 0, 1, 2, j = 0, 1, \ldots, (2^\alpha - 2)$.

Let $\alpha - 1 = \beta$. Then, the coordinates for the one-entries become $[i(2)^\beta + j,\ i(2^{\beta+1} - 1) + j]$. Now, observe that if $i = 0$ while j varies over its entire range, then the set of coordinates for the one-entries is given by $[i, j] = I^{(1)}_{2^\beta - 1}$, which describes an identity matrix that occupies rows 0 to $(2^{\beta+1} - 2)$ and columns 0 to $(2^{\beta+1} - 2)$ of the matrix $R(\alpha)$. Similarly, when $i = 1$ and j varies over its entire range, the set of coordinates for the one-entries is given by $[2^\beta + j,\ 2^{\beta+1} - 1 + j] = I^{(2)}_{2^{\beta+1}-1}$, which describes another identity matrix $I_{2^{\beta+1}-1}$ placed in rows 2^β to $[3(2^{\beta+1}) - 2]$ and columns $2^{\beta+1} - 1$ to $2^{\beta+2} - 3$. Finally, when $i = 2$ and j varies over its entire range, then the set of coordinates for the one-entries is given by

$$[2^{\beta+1} + j,\ 2(2^{\beta+1} - 1) + j] = I^{(3)}_{2^{\beta+1}-1}$$

which describes the third identity matrix $I_{2^{\beta+1}-1}$ placed in rows $2^{\beta+1}$ to $(2^{\beta+2} - 2)$ and columns $(2^{\beta+2} - 2)$ to $[3(2^{\beta+1}) - 4]$.

Finally, notice that there is no overlapping between the row coordinates of $I^{(1)}$ and $I^{(3)}$, which means that each row of the matrix $R(\alpha)$ will contain only two one-entries at the row coordinates specified above and only one one-entry in the remaining rows. Since the number of additions required is equal to the number of these rows with two one-entries, the number of additions to compute $R(\alpha)$ is given by

$$2\sum_{i=1}^{\alpha} 3^{i-1}(2^{\alpha-i} - 1). \quad (15)$$

Therefore, from (14) and (15), the number of additions needed to compute the postadditions is given by

$$2\sum_{i=1}^{\alpha} 3^{i-1}(2^{\alpha-i} - 1) + 2\sum_{i=1}^{\alpha} 3^{i-1}(2^{\alpha-i+1} - 1) = 2 + 4(3)^\alpha - 6(2)^\alpha. \quad (16)$$

From (13) and (16), the total number of additions needed to compute n-point ($n = 2^\alpha$) convolutions is given by

$$(3^\alpha - 2^\alpha) + 2 + 4(3)^\alpha - 6(2)^\alpha = 5(3^\alpha) - 7(2^\alpha) + 2. \quad (17)$$

Since $\tilde{D}(n)$ is a diagonal matrix of order 3^α, applying the matrix $\tilde{D}(n)$ implies performing independent elemenwise multiplications, which can be done in parallel. Therefore, the total number of multiplications required is equal to the number of multiplications in the single-core stage and is equal to 3^α [1].

Tables I and II show the computational advantage of the proposed improved algorithm when compared to previous algorithms, such as the original matrix–vector multiplication and the fast Fourier transform (FFT) algorithms. For example, although the number of additions of the proposed algorithm is nearly twice that of the FFT, the number of multiplications is less by 70% for the case $n = 1024$.

V. Conclusions

In this letter, we presented an improved Toom's algorithm that allows hardware savings without slowing down the processing speed. We demonstrated the computational advantage of the proposed improved algorithm when compared to previous algorithms, such as the original matrix–vector multiplication and the FFT algorithms.

References

[1] 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, pp. 1202–1204, Apr. 1999.

[2] X. Liu and L. T. Bruton, "High-speed systolic ladder structures for MD recursive digital filters," *IEEE Trans. Signal Processing*, vol. 44, pp. 1048–1055, Apr. 1996.

[3] F. Marino, "A fixed-point parallel convolver without precision loss for the real-time processing of long numerical sequences," in *Proc. IEEE 7th Symp. Parallel Distrib. Process.*, 1995, pp. 644–651.

[4] D. Rodriguez, "Tensor product algebra as a tool for VLSI implementation of the discrete Fourier transform," in *Proc. ICASSP*, Toronto, ON, Canada, 1991, pp. 1025–1028.

[5] H. V. Sorensen, C. A. Katz, and C. S. Burrus, "Efficient FFT algorithms for DSP processing using tensor product decompositions," in *Proc. ICASSP*, Albuquerque, NM, 1990, pp. 1507–1510.

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