TABLE II
ROBUSTNESS OF FILTERS

Image	σ^2, p	Canonical LWOS	M-LWOS 3x3	M-LWOS 5x5
Lena	400,4	138.7	130.8	117.6
Lena	400,12	201.3	180.6	134.3
Lena	200,8	115.8	109.5	100.4
Lena	600,8	200.4	188.8	150.0
Woman	400,8	150.3	140.7	117.4
Lighthouse	400,8	319.3	309.1	316.8

TABLE III
FILTERING RESULTS USING FILTERS DESIGNED BY IDEAL TRAINING

Image	σ^2, p	Canonical LWOS	M-LWOS 3x3	M-LWOS 5x5
Lena	400,4	131.6	128.5	117.4
Lena	400,12	188.2	171.9	134.1
Lena	200,8	108.6	104.6	98.0
Lena	600,8	191.8	186.8	149.0
Woman	400,8	142.9	140.2	117.0
Lighthouse	400,8	255.6	264.4	253.0

better visual performance than the other filters. We can also recognize that the 3×3 window size is insufficient for such image processing.

C. Robustness of M-LWOS Filters

Both canonical LWOS and M-LWOS filters are designed by training. In practical cases, the original images of corrupted images (i.e., input images) are not available. Thus, training images are different from corrupted images. It is necessary to assess the robustness of filters designed by training. Both the canonical LWOS filter and the M-LWOS filter ($P = 9$) are designed by using Lena which is corrupted by mixed noise with (σ^2, p) = (400,8). We process six corrupted images as listed in Table II by using these two filters. Table III shows the results for filters which are designed ideally (using corrupted images and corresponding original images).

The M-LWOS filter is superior to the canonical LWOS filter for all of the images. M-LWOS filters are robust against changing the noise condition. Moreover, the designed filters achieve an almost ideal performance, as can be seen from the results for Woman, if we choose training images which are similar to the corrupted images. On the basis of these results, M-LWOS filters are considered robust against the selection of training images.

V. CONCLUSIONS

The M-LWOS filter was introduced. The optimization of the M-LWOS filter under the MSE criterion is investigated. From experimental results, we can believe the effectiveness of the optimal M-LWOS filter. The M-LWOS filter gives almost the same performance at the same window size of the canonical LWOS filter. However, the number of parameters of the M-LWOS filter is superior to that of the canonical LWOS filter. The M-LWOS filter has fewer parameters than a Ll filter that gives the same performance.

REFERENCES

[1] F. Palmieri and C. G. Boncelet, Jr., "Ll-filter—A new class of order statistic filters," *IEEE Trans. Acoust., Speech, Signal Processing*, vol. 37, pp. 691–701, May 1989.
[2] L. Yin, R. Yang, M. Gabbouj, and Y. Neuvo, "Weighted median filters: A tutorial," *IEEE Trans. Circuits Syst. II*, vol. 43, pp. 157–192, Mar. 1996.
[3] L. Yin and Y. Neuvo, "Fast adaptation and performance characteristics of FIR-WOS hybrid filters," *IEEE Trans. Signal Processing*, vol. 42, pp. 1610–1628, July 1994.
[4] J. Song and Y. H. Lee, "Linear combination of weighted order statistic filters: Canonical structure and optimal design," *IEEE Trans. Circuits Syst. II*, vol. 43, pp. 349–362, May 1996.
[5] Y. T. Kim and G. R. Arce, "Permutation filter lattices: A general order statistics filtering framework," *IEEE Trans. Signal Processing*, vol. 42, pp. 2227–2241, Sept. 1994.
[6] A. Taguchi, "A design method of fuzzy weighted median filters," in *Proc. IEEE Int. Conf. Image Processing*, Lausanne, Switzerland, Sept. 1996, pp. 423–426.
[7] A. K. Jain, *Fundamentals of Digital Image Processing*. Englewood Cliffs, NJ: Prentice Hall, 1989.
[8] Y. H. Lee and S. A. Kassam, "Generalized median filtering and related nonlinear filtering techniques," *IEEE Trans. Acoust., Speech, Signal Processing*, vol. ASSP-33, pp. 672–683, June 1985.
[9] J. Astola and P. Kuosmanen, *Fundamentals of Nonlinear Digital Filtering*. Boca Raton, FL: CRC, 1997.

A Modified Shuffle-Free Architecture for Linear Convolution

A. Elnaggar, M. Aboelaze, and A. Al-Naamany

***Abstract*—This paper presents a class of modified parallel very large scale integration architectures for linear convolution in shuffle-free forms. The proposed algorithms show that for 1-D convolution, the number of lower-order convolutions can be reduced from three to two allowing a hardware saving without slowing down the processing speed. The proposed partitioning strategy results in a core of data-independent convolution computations. Such computations can be overlapped in software pipelines, super pipelines, or executed concurrently on multiple functional units in a DSP chip.**

I. INTRODUCTION

Convolution is a very important operation in signal and image processing with applications to digital filtering, and video image

Manuscript received August 5, 1999; revised August 8, 2001. This paper was recommended by Associate Editor R. Geiger.

A. Elnaggar and A. Al-Naamany are with the Department of Information, Engineering, Sultan Qaboos University, Muscat 123, Oman (e-mail: ayman@squ.edu.om; naamany@squ.edu.om).
M. Aboelaze is with the Department of Computer Science, York University, Toronto, ON M3J 1P3, Canada (e-mail: aboelaze@cs.yorku.ca).

Publisher Item Identifier S 1057-7130(01)10422-2.

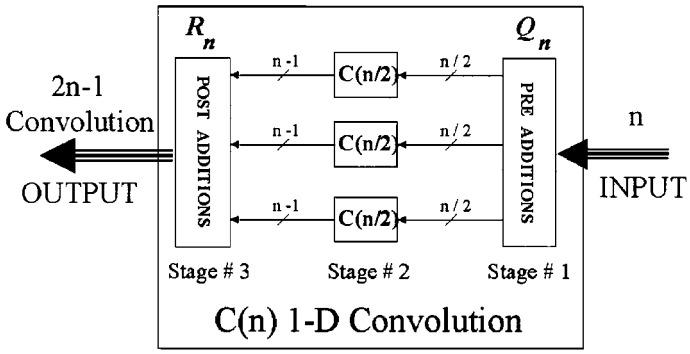


Fig. 1. The original realization of the 1-D convolution algorithm. (a) Direct realization. (b) RDS permutations.

processing. Thus, abundant approaches have been suggested to achieve high speed processing for linear convolution and to design efficient convolution architectures [1]–[6].

The proposed work is based on a nontrivial modification of the 1-D convolution algorithm presented in [1] and shown in Fig. 1. We show that using an alternative (permutation-free) construction, the number of lower-order parallel convolutions (Stage #2 in Fig. 1) can be reduced from three to two only, 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. Tensor products (or Kronecker products) have proven to be useful in providing a unified decomposable matrix formulations for multidimensional transforms, convolutions, matrix multiplication, and other fundamental computations [1], [4]–[6]. Some of the tensor product and permutation matrices properties that will be used throughout this paper are [6]: If $n = n_1 n_2 n_3$, then

$$P_{n,n_1} P_{n,n_2} = P_{n,n_1 n_2}. \tag{1}$$

Commutative property

$$(A \otimes B) \otimes C = A \otimes (B \otimes C). \tag{2}$$

Parallel Operations: if $n = n_1 n_2$ then

$$A_{n_1} \otimes B_{n_2} = P_{n,n_1}(I_{n_2} \otimes A_{n_1}) P_{n,n_2}(I_{n_1} \otimes B_{n_2}). \tag{3}$$

If $n = n_1 n_2$then

$$P_{n,n_1} P_{n,n_2} = I_n \tag{4}$$

where, $P_{n,s}$ is an $n \times n$ binary matrix specifying an n/s-shuffle (or s-stride) permutation.

II. Replicated Dilated Shuffle (RDS) Permutations

For the purposes of this paper, the class of RDS permutations needs to be introduced. This class of shuffle is very effective for network partitioning and for deriving modular VLSI interconnection networks for convolution as will be shown in the following sections.

A shuffle permutation with dilation d operates on subvectors of size d rather than single elements. For example, the permutation $P^2_{4,2}$ operates on subvectors of size two. To illustrate this effect, consider the two operations $P^1_{4,2} X_4$ and $P^2_{4,2} X_8$, respectively,

$$P^1_{4,2} X_4 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_0 \\ x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} x_0 \\ x_2 \\ x_1 \\ x_3 \end{bmatrix}$$

$$P^2_{4,2} X_8 = \begin{bmatrix} I_2 & 0 & 0 & 0 \\ 0 & 0 & I_2 & 0 \\ 0 & I_2 & 0 & 0 \\ 0 & 0 & 0 & I_2 \end{bmatrix} \begin{bmatrix} \tilde{x}_0 \\ \tilde{x}_1 \\ \tilde{x}_2 \\ \tilde{x}_3 \end{bmatrix} = \begin{bmatrix} \tilde{x}_0 \\ \tilde{x}_2 \\ \tilde{x}_1 \\ \tilde{x}_3 \end{bmatrix}$$

where, $\boldsymbol{I_2}$ is the identity matrix of size 2, and

$$\tilde{x}_i = \begin{bmatrix} x_{2i} \\ x_{2i+1} \end{bmatrix}.$$

The main power of RDS permutations stems from the fact that standard shuffle permutations can be constructed from smaller RDS permutations in a simple recursive fashion. This is very useful property for enabling the modularization of VLSI networks that employ shuffle networks. For example, the permutations $P^1_{16,8}$ realized in Fig. 2(a) can be represented in RDS permutations as

$$P^1_{16,8} = \left(I_4 \otimes P^1_{4,2}\right)\left(I_2 \otimes P^2_{4,2}\right)\left(I_1 \otimes P^4_{4,2}\right)$$
$$= \prod_{i=0}^{2} \left(I_{2^i} \otimes P_{4,2} \otimes I_{2^{2-i}}\right).$$

The realization of $P^1_{16,8}$ in RDS permutation is shown in Fig. 2(b).

III. 1-D Shuffle-Free Convolution Algorithm

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

$$C(n) = R_n (I_3 \otimes C(n/2)) Q_n \tag{5}$$

where

$$Q_n = \left[P_{3(2^{\alpha-1}),3(2^{\alpha-2})}\right]^{\alpha-1} \left[(I_{2^{\alpha-1}} \otimes A)\, P_{2^\alpha,2^{\alpha-1}}\right] \tag{6}$$

$$R_n = R(\alpha) \left[P_{3(2^{\alpha-1}),3} \left(I_{2^{\alpha-1}} \otimes B\right) P_{3(2^{\alpha-1})(2^{\alpha-1})}\right] \tag{7}$$

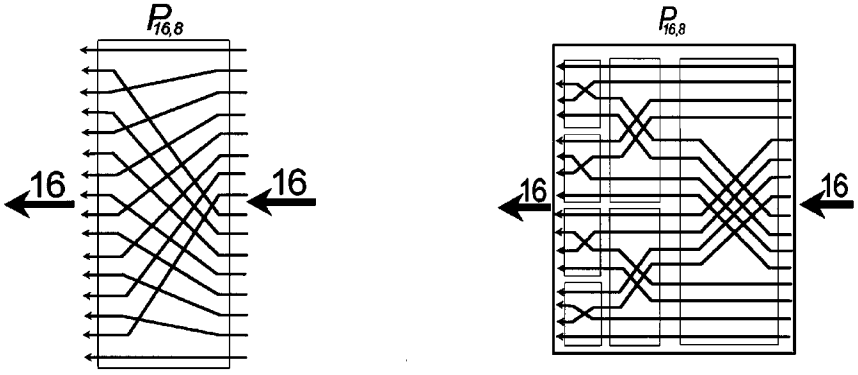
A, B, and $R(\alpha)$ are special matrices of 1's and 0's only and can be realized using simple adders [1]. The realization of the original algorithm is shown in Fig. 1. Observe that, we have drawn our networks such that data flows from right to left. We chose this convention to show the direct correspondence between the derived algorithms and the proposed VLSI networks.

A. Shuffle-Free Algorithm

The shuffle permutations can be removed from the tensor forms of Q_n and R_n in (6) and (7) that represent the pre-additions and the post-additions, respectively. In this case, the effect of the shuffle permutations is implicitly embedded in the tensor expression.

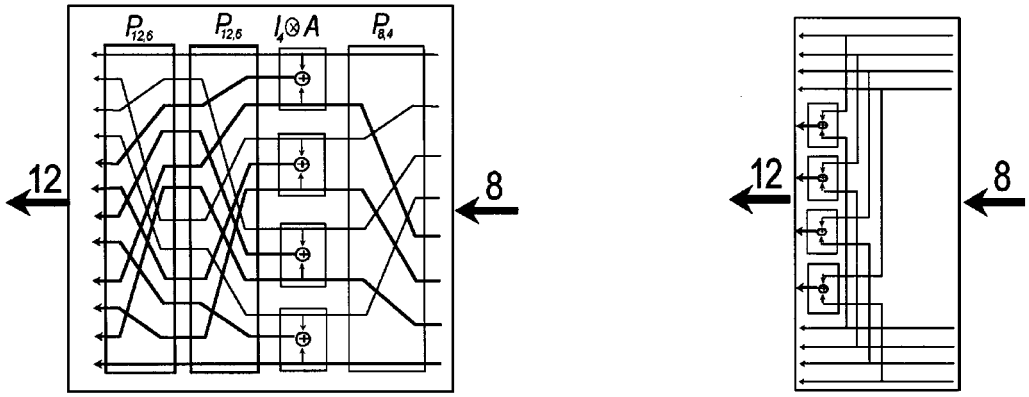
First, recall that the $(\alpha - 1)$ permutations in the term $\left(P_{3(2^{\alpha-1}),3(2^{\alpha-2})}\right)^{\alpha-1}$contained in (6) can be simplified by first using property (1), we have

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



(a) (b)

Fig. 2. Fig. 2 The realization of $\mathbf{P}^{1}_{16,8}$. (a) Direct realization. (b) RDS permutations.



(a) (b)

Fig. 3. The realization of Q_8. (a) Original realization. (b) Shuffle-free realization.

Therefore, using property (2)

$$\begin{aligned}&\left(P_{3(2^{\alpha-1}),\,3(2^{\alpha-2})}\right)^{\alpha-1}\\&=\left(P_{3(2^{\alpha-1}),\,3}\prod_{i=1}^{\alpha-2}P_{3(2^{\alpha-1}),\,2}\right)\left(P_{3(2^{\alpha-1}),\,3(2^{\alpha-2})}\right)^{\alpha-2}\\&=P_{3(2^{\alpha-1}),\,3}\left(P_{3(2^{\alpha-1}),\,3(2^{\alpha-2})}\right)^{\alpha-2}.\end{aligned}\tag{9}$$

But from property (4),

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

Hence,

$$\left(P_{3(2^{\alpha-1}),\,3(2^{\alpha-2})}\right)^{\alpha-1}=P_{3(2^{\alpha-1}),\,3}.\tag{11}$$

Therefore, we can write Q_n as

$$Q_n=P_{3(2^{\alpha-1}),\,3}\left(I_{2^{\alpha-1}}\otimes A\right)P_{2^{\alpha},\,2^{\alpha-1}}.\tag{12}$$

By decomposing the two terms $P_{3(2^{\alpha-1}),\,3}$ and $P_{2^{\alpha},\,2^{\alpha-1}}$ in (12) into their RDS permutations and applying properties (2) and (4) repetitively, Q_n can be simplified to

$$Q_n=A\otimes I_{2^{\alpha-1}}.\tag{13}$$

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 (6) and the modified shuffle-free representation of (13) are shown in Fig. 3(a) and (b), respectively.

Even though the resulting circuits in Fig. 3(a) and (b) are topologically equivalent, removing the shuffle-permutations from the tensor formulations can simplify data movement. Moreover, the resulting shuffle-free networks are suited for implementation using CAD tools that employ automatic partitioning, placement, and routing.

Similarly, substituting $I_{2^{\alpha-1}}$ for A_{n_1} in (7) and applying properties (1)–(4), gives

$$\begin{aligned}R_n&=R(\alpha)\left[P_{3(2^{\alpha-1}),\,3}\left(I_{2^{\alpha-1}}\otimes B\right)P_{3(2^{\alpha-1}),\,(2^{\alpha-1})}\right]\\&=R(\alpha)(B\otimes I_{2^{\alpha-1}}).\end{aligned}\tag{14}$$

The permutation-free recursive realization of the 8-point convolution using three 4-point convolutions is shown in Fig. 4(a). We assume that each addition requires one unit of time.

B. Multiplexed Architecture of the 1-D Convolution

A careful scrutiny of the realization shown in Fig. 4(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 4-point convolution in the center stage. Thus, the top and the bottom 4-point convolutions can be computed one addition cycle ahead of the middle 4-point convolution. This means that only two 4-point convolvers are needed at a time. Therefore, through the use of a multiplexer, either the top or the bottom 4-point convolver can be removed from the architecture as shown in Fig. 4(b). A multiplexer is used to multiplex the top and middle parts of the signal flow. Input data are read from the host computer and are executed in a pipeline fashion. The top part of the signal flow will pass through the bottom 4-point convolver first; while the middle part will first perform signal additions with the bottom part, then pass through the bottom 4-point convolver. It can be verified that succeeding sets of input data can enter the hardware and can be executed in a pipeline fashion. In addition, lower order convolution will be at full utilization.

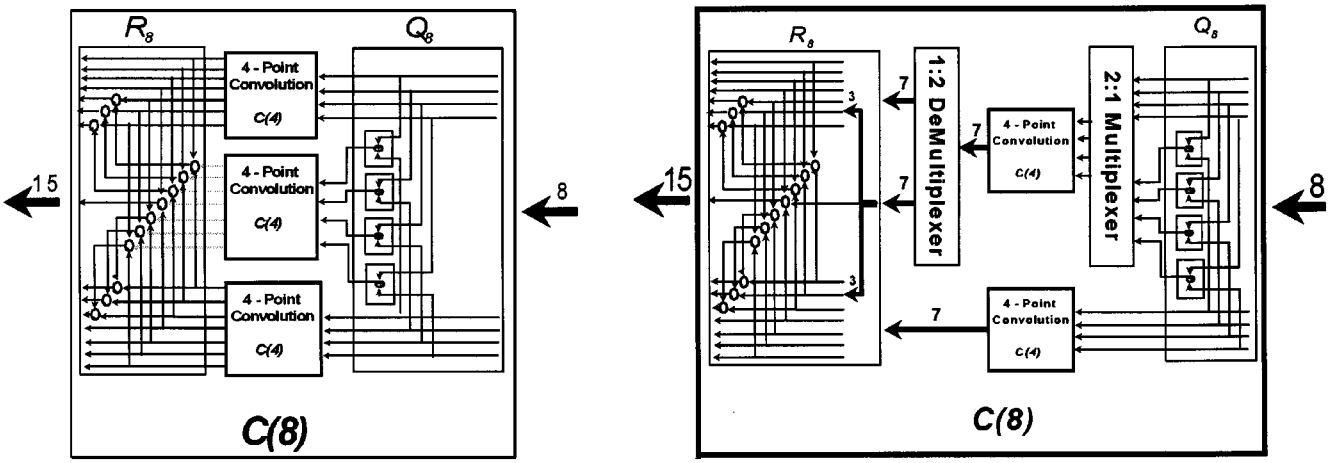


Fig. 4. Realization of 8-point convolution. (a) Shuffle-free architecture. (b) Multiplexed architecture.

TABLE I
COMPARISON OF THE NUMBER OF ADDITIONS REQUIRED BY THE PROPOSED ALGORITHM 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
COMPARISON OF THE NUMBER OF MULTIPLICATIONS REQUIRED BY THE PROPOSED ALGORITHM 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	6561	59,049

It should be mentioned that, it would not be possible to observe the resource sharing shown in Fig. 4(a) without the modified shuffle-free architecture. This is evident by comparing the realizations of Q_8 in the original form shown in Fig. 3(a) and the modified shuffle-free form shown in Fig. 3(b).

C. Computation Complexity of the Proposed 1-D Convolution Algorithm

The number of additions needed to compute the pre-additions stage is given by ($n = 2^\alpha$)

$$\sum_{i=0}^{\alpha-1}(3^{\alpha-i-1})(2^i)=3^{\alpha-1}\sum_{i=0}^{\alpha-1}(3^{-i})(2^i)=(3^\alpha-2^\alpha). \quad (15)$$

The number of additions needed to compute the post-additions stage is given by

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

Therefore, the total number of additions needed to compute n-point ($n = 2^\alpha$) convolution is given by

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

In the multiplexed architecture this number is reduced to

$$5(3^\alpha)-7(2^\alpha)-5\left(3^{\alpha-1}\right)+7\left(2^{\alpha-1}\right)=10\left(3^{\alpha-1}\right)-7\left(2^{\alpha-1}\right). \quad (18)$$

The total number of multiplications required is equal to 3^α. In the multiplexed architecture, this number is reduced to be $2(3^{\alpha-1})$ multiplication only. Tables I and II show the number of additions and multiplications required by the proposed algorithm, respectively. 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 $n = 1024$.

IV. CONCLUSIONS

In this brief, we presented a class of modified parallel VLSI architectures for linear convolution in shuffle-free forms. The proposed algorithms showed that for 1-D convolution, we use two smaller order convolutions only allowing a hardware saving without slowing down the processing speed. 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.

ACKNOWLEDGMENT

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

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, no. 4, 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 and Distributed Processing*, 1995, pp. 644–651.
[4] D. Rodriguez, "Tensor product algebra as a tool for VLSI implementation of the discrete Fourier transform," in *Proc. ICASSP'91*, 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'90*, pp. 1507–1510.
[6] R. Tolimieri, M. An, and C. Lu, *Algorithms for Discrete Fourier Transform and Convolution*. New York: Springer-Verlag, 1989.

A Low-Power Array Multiplier Using Separated Multiplication Technique

Chang-Young Han, Hyoung-Joon Park, and Lee-Sup Kim

***Abstract*—This brief proposes a separated multiplication technique that can be used in digital image signal processing such as finite impulse response (FIR) filters to reduce the power dissipation. Since the 2-D image data have high spatial redundancy, such that the higher bits of input pixels are hardly changed, the redundant multiplication of higher bits is avoided by separating multiplication into higher and lower parts. The calculated values of the higher bits are stored in memory cells, *caches*, such that they can be reused when a cache hit occurs. Therefore, the dynamic power is reduced by about 14% in multipliers by using the proposed separated multiplication technique (SMT) and in a 1-D 4-tap FIR filter by about 10%.**

***Index Terms*—FIR filter, low power, LRU, multiplier.**

I. INTRODUCTION

Digital signal processing (DSP) is the technology at the heart of the next generation of personal mobile communication systems. Most DSP systems incorporate a multiplication unit to implement algorithms such as convolution and filtering. In many DSP algorithms, the multiplier lies in the critical delay path and ultimately determines the performance of the algorithm. Present technologies possess computing capacities that allow the realization of computationally intensive tasks such as speech recognition and real-time digital video. However, the demand for high-performance portable systems incorporating multimedia capabilities has elevated design for low power to the forefront of design requirements in order to maintain reliability and provide longer hours of operation. As a result, a great deal of effort has been invested to reduce the energy dissipation in multipliers in various research fields [1]–[6].

In this brief, we exploit the spatial redundancy of images to reduce the energy dissipation in multipliers and apply this novel multiplier implementation technique to decimated FIR filters by factor 2, which are popularly used in discrete wavelet transform (DWT).

Differential coefficients method (DCM) [7]–[9] is a similar algorithm, which uses highly correlated data. DCM uses differential coefficients to multiply inputs, and compensates for the effect of differential coefficients by adding the previously computed partial product. However, the use of DCM necessitates careful consideration when coefficients are determined in the FIR filter design process. Furthermore, this method is not adequate for a sub-block based 2-D image due to the large memory requirement. Therefore, this brief proposes a new power reduction method for 2-D image processing using the separated multiplication technique.

The remainder of this brief is organized as follows. Section II briefly introduces the motivations for finding a new power reduction in the multiplier. The key idea for reducing power in the multiplication is described in Section III. The proposed idea is simulated and optimal solutions are applied to the decimated FIR filter module in Section IV. The proposed architecture is described in Section V and the results are analyzed in Section VI. Finally, conclusions are presented in Section VII.

II. MOTIVATIONS

A. Review of DCM Algorithm and Its Drawbacks

The direct form of the FIR filter uses coefficients and inputs directly. The DCM computes partial products with difference coefficients first, and then adds the previously computed partial products. We can rewrite the general FIR filter outputs Y_{j+1} with the first-order difference DCM algorithm and obtain (1). The graphical descriptions of direct form and DCM form are shown in Fig. 1.

$$\begin{aligned} Y_{j+1} &= C_0X_{j+1} + C_1X_j + \cdots + C_{N-1}X_{j-N+2} \\ &= C_0X_{j+1} + ((C_1X_j - C_0X_j) + C_0X_j) + \cdots \\ &\quad + (C_{N-1}X_{j-N+2} - C_{N-2}X_{j-N+2}) + C_{N-2}X_{j-N+2} \\ &= C_0X_{j+1} + (dC_1^1X_j + C_0X_j) + \cdots \\ &\quad + (dC_{N-1}^1X_{j-N+2} + C_{N-2}X_{j-N+2}) \end{aligned} \tag{1}$$

where

$$dC_k^1 = C_k - C_{k-1},\ k = 1\ to\ (N-1).$$

The DCM needs small bit-width coefficients to reduce the power consumption in computing the partial products. However, power reduction cannot be expected if the bit-width of the coefficient differences is not significantly shorter than the original coefficient bit-width. This scheme cannot be applied to a system if the system uses the existing FIR filter and its coefficient difference is not small. Another drawback of the DCM is that it is not a suitable technique for sub-block based image processing using overlapping memory. After the last data of a certain row is filtered, the data contained in the registers, shown in Fig. 1(b), must be flushed to process the next row. Since the new row processing requires the previous sub-block pixels rather than the current, it requires additional logic to handle the flushing scheme. As a result, the latency increases if DCM is used in sub-block based image processing. Therefore, a new method is needed for efficiently handling sub-block based 2-D image processing.

Manuscript received May 10, 2000; revised September 13, 2001. This paper was recommended by Associate Editor V. Owall.

The authors are with the Multimedia VLSI Laboratory, Korea Advanced Institute of Science and Technology (KAIST), Taejon, 305–701 Korea (e-mail: quark@mvlsi.kaist.ac.kr).

Publisher Item Identifier S 1057-7130(01)10421-0.