Two-Dimensional Block Processors—Structures and Implementations

MAHMOOD R. AZIMI-SADJADI, MEMBER, IEEE, AND ROBERT A. KING

Abstract—Two-dimensional (2-D) block processing technique for linear filtering of digital images is introduced. New 2-D block structures are derived for 2-D recursive digital filters realized by difference equations and state-space formulations. Several special cases have also been considered and the relevant 2-D block structures are given. The computational costs of different implementation techniques employing high-speed convolution algorithms such as fast Fourier transform, number theoretic transform and polynomial transform have been studied. A comparison among the relative efficiencies of these implementation schemes is made and a suitable method is then proposed using short convolution algorithm which results in a minimized computational time.

I. Introduction

THE application of block processing to 1-D recursive (IIR) digital filters was first suggested by Gold and Jordan [1] who proposed a method for implementing IIR filters which requires only finite convolutions on the blocks of input sequence. Subsequently, a number of other techniques have also been introduced, but the most useful approach, from the point of view of implementation, was developed by Burrus [2]. He used matrix representation of 1-D difference equation and obtained a block recursive equation whereby the output blocks can recursively be computed from the past and present input blocks and past output block. This method was shown to provide an efficient means of recursive filtering operation especially when used in conjunction with fast transform techniques. Later, Mitra, and Gnanasekaran [3] have developed several other 1-D block structures which possess different properties with regard to roundoff error, sensitivity, etc.

More recently it has been shown that the block processing structure also exhibits several other prominent benefits such as reduced finite wordlength effects and increased data throughput rate. Barnes and Shinnaka [4] have derived a 1-D block-state structure from a simple 1-D state-space model and have shown that the roundoff noise variance for the block-state structure is reduced by a factor equal to the block length when compared with the noise variance of its scalar counterpart. They have also shown [5] that the poles of the 1-D block-state realization are the Lth power (L being the length of blocks) of the poles of an associated state-space realization, i.e., if the original scalar filter is assumed to be stable, then the poles of the corresponding block implemented filter move closer to the origin of the unit circle as the block length is increased. In [6] this block-state structure is used to develop a new 1-D block-state structure with high inherent parallelism which is ideally suited for implementation in a multiprocessor environment. Moreover, the number of computations required for this structure is shown to be much less than that of the canonical block-state structure [4].

Manuscript received January 15, 1985; revised June 4, 1985.
M. R. Azimi-Sadjadi is with the Department of Electrical Engineering, University of Michigan, Dearborn, MI 48128.
R. A. King is with the Department of Electrical Engineering, Imperial College of Science and Technology, London SW7 2BT, United Kingdom.
IEEE Log Number 8405847.

Block implementation for 2-D digital filters has been studied by several authors [7]–[13], over the past few years. Several new 2-D block structures have been proposed for digital filters realized by difference equation [9]–[12], convolution summations [9]–[12] and state-space formulations [13].

In this paper block implementation of 2-D recursive digital filters is considered. For digital filters described by 2-D difference equation a general 2-D block recursive equation is given in Section II. Several special cases have been studied. Specifically, these include; block structures for 2-D all-pole filters, separable product, separable denominator–non-separable numerator 2-D recursive filters and also 2-D nonrecursive digital filters. A new 2-D block-state realization model is derived in Section III from a single-input, single-output 2-D state-space model of Roesser [15]. This structure which has a form analogous to its scalar counterpart, lends itself for analyzing 2-D multi-input, multi-output systems. In Section IV, various possible implementation methods employing fast Fourier transform (FFT), number theoretic transform (NTT) and polynomial transform (PT) are examined on the 2-D block recursive structure. A comparison of their relative efficiencies based on number of multiplications, roundoff error, dynamic range constraint and storage requirements, is made and a suitable implementation scheme is then proposed which results in further enhancement of the computational efficiency.

II. 2-D Block Recursive Structure (General)

In this section, a general 2-D block recursive structure is introduced for 2-D recursive digital filters based on the idea of representing linear filtering operations in matrix form.

Consider a causal 2-D recursive (IIR) digital filter with transfer function:

$$H(z,w)=\frac{Y(z,w)}{X(z,w)}=\frac{(z,w)\{y_{m,n}\}}{(z,w)\{x_{m,n}\}}=\frac{\sum_{i=0}^{M_b}\sum_{j=0}^{N_b} b_{i,j}z^{-i}w^{-j}}{\sum_{i=0}^{M_a}\sum_{j=0}^{N_a} a_{i,j}z^{-i}w^{-j}} \quad (1)$$

where $\{x_{m,n}\}$ and $\{y_{m,n}\}$ are the input and output sequences, respectively, and $a_{i,j}$'s and $b_{i,j}$'s are the filter coefficients.

Equation (1) can be written in convolution form to relate the transfer function coefficients to the input and output elements.

$$a_{m,n} * y_{m,n} = b_{m,n} * x_{m,n}. \quad (2)$$

Consider the input sequence $\{x_{m,n}\}$ to be partitioned into nonoverlapping blocks of dimension K by L, where

$$K \geqslant \max(M_a, M_b), \qquad L \geqslant \max(N_a, N_b). \quad (3)$$

Now, if these blocks are arranged as vectors with element subscripts ordered lexicographically, Equation (2) can be written in a matrix form. Using this formulation, the following "2-D block recursive equation" can be obtained [9], [10]:

$$\begin{aligned} Y_{i,j} &= E_{10}Y_{i-1,j} + E_{01}Y_{i,j-1} + E_{11}Y_{i-1,j-1} \\ &\quad + F_{00}X_{i,j} + F_{10}X_{i-1,j} + F_{01}X_{i,j-1} + F_{11}X_{i-1,j-1}. \end{aligned} \quad (4)$$

In this equation, $X_{i,j}$ is the (i,j)th input block, defined as a column vector, i.e.,

$$X_{i,j} = \left[\hat{X}_{iK}^{(j)}\hat{X}_{iK+1}^{(j)}\cdots\hat{X}_{(i+1)K-1}^{(j)}\right]^t \quad (5a)$$

where $\hat{X}_m^{(j)} = [x_{m,jL}x_{m,jL+1}\cdots x_{m,(j+1)L-1}]$ and similarly for $Y_{i,j}$. Matrices E_{pq}'s and F_{pq}'s are defined by

$$\begin{aligned} E_{pq} &= -C_{00}^{-1}C_{pq}, \qquad p,q=0,1 \text{ and } (p,q)\neq(0,0) \\ F_{pq} &= C_{00}^{-1}D_{pq}, \qquad p,q=0,1. \end{aligned} \quad (5b)$$

where C_{00} and C_{10} are, respectively, lower and upper triangular, KL by KL, block Toeplitz matrices defined by

$$C_{00_{i,j}} = \begin{cases} A_m, & m=i-j, \quad 0\leqslant m\leqslant M_a \\ 0, & \text{otherwise}, \quad i,j=1,\cdots,K \end{cases}$$

and

$$C_{10_{i,j}} = \begin{cases} A_m, & m=K-(j-i), \quad 1\leqslant m\leqslant M_a \\ 0, & \text{otherwise} \quad i,j=1,\cdots,K. \end{cases} \quad (5c)$$

This also holds for C_{01} and C_{11}, where the constituent block matrices are A'_m's. The block matrices A_m and A'_m are lower and upper triangular, L by L, Toeplitz given by

$$A_{m_{i,j}} = \begin{cases} a_{m,n}, & n=i-j \quad 0\leqslant n\leqslant N_a \\ 0, & \text{otherwise} \quad i,j=1,\cdots,L \end{cases}$$

$$A'_{m_{i,j}} = \begin{cases} a_{m,n}, & n=L-(j-i) \quad 1\leqslant n\leqslant N_a \\ 0, & \text{otherwise} \quad i,j=1,\cdots,L. \end{cases} \quad (5d)$$

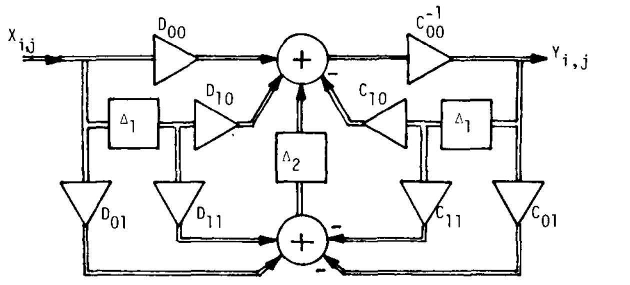


Fig. 1. Canonical Block Realization Form.

Matrices D_{pq}'s can be defined, in a similar manner, in terms of B_m and B'_m where these latter matrices may be defined in terms of $b_{m,n}$ similar to A_m and A'_m.

Equation (4) is a general 2-D block recursive structure for 2-D quarter plane recursive digital filters. The processing scheme involves partitioning the input image into a sequence of "nonoverlapping" blocks and then computing each output block recursively from the past and present input blocks and past output blocks via the block recursive equation of (4). The specific structure of C's and D's matrices makes it possible to employ high-speed convolution techniques to accelerate the processing time. Note that, the convolution operator C_{00}^{-1} can be expressed in terms of the truncated impulse response elements of the associated all-pole filter. This allows the block recursive equation to be implemented without any matrix inversion. Furthermore, the roundoff noise due to the finite word-length effects for this structure is shown [14] to be very much reduced when compared with that of direct filtering operation using the 2-D difference equation. These advantages make this structure to be a very efficient and accurate means of recursive filtering operation. In Section IV), this structure is used in conjunction with several high-speed convolution algorithms to enhance the computational efficiency.

The 2-D block recursive equation of (4) can be realized in a variety of ways. The canonical block realization for this structure is shown in Fig. 1. In this figure Δ_1 and Δ_2 represent the delay operators associated with the first and second subscripts, respectively.

Special Cases

1) 2-D Recursive All-Pole Filter

For an all-pole filter $b_{00}=1$ and $b_{i,j}=0$, $\forall(i,j)\neq(0,0)$ or in matrix form

$$D_{00}=I_{KL} \quad \text{and} \quad D_{01}=D_{10}=D_{11}=0$$

Thus the block recursive equation in this case becomes

$$Y_{i,j} = E_{10}Y_{i-1,j} + E_{01}Y_{i,j-1} + E_{11}Y_{i-1,j-1} + C_{00}^{-1}X_{i,j} \quad (6)$$

2) Separable Product 2-D Recursive Filter

The transfer function of a separable product 2-D filter is given by

$$H(z,w) = H_1(z)H_2(w) = \left(\frac{\sum_{i=0}^{M_b} \beta_i z^{-i}}{\sum_{i=0}^{M_a} \alpha_i z^{-i}}\right)\left(\frac{\sum_{j=0}^{N_b} \beta'_j w^{-j}}{\sum_{j=0}^{N_a} \alpha'_j w^{-j}}\right). \quad (7)$$

It can easily be shown that the matrices C_{pq}'s and D_{pq}'s will also be separable product, i.e.,

$$C_{pq} = \Lambda_p \otimes \Lambda'_q$$
$$D_{pq} = \Pi_p \otimes \Pi'_q, \qquad p,q = 0,1 \quad (8)$$

where Λ_p, Λ'_q, Π_p, and Π'_q are Toeplitz matrices associated with the corresponding 1-D block structures and contain as their elements, α_i, α'_i, β_i, and β'_i, respectively [3]. In addition, $\otimes$ denotes the Kronecker product operation.

Using the properties of Kronecker product, the 2-D block recursive equation simplifies to 1-D block equations of form

$$\Psi_0 Z_i + \Psi_1 Z_{i-1} = \Theta_0 W_i + \Theta_1 W_{i-1}, \quad (9a)$$

where

$$Z_i = \Psi'_0 Y_{i,j} + \Psi'_1 Y_{i,j-1} \quad (9b)$$
$$W_i = \Theta'_0 X_{i,j} + \Theta'_1 X_{i,j-1} \quad (9c)$$

and

$$\Psi_p = (\Lambda_p \otimes I_L), \quad \Theta_p = (\Pi_p \otimes I_L)$$
$$\Psi'_q = (I_K \otimes \Lambda'_q), \quad \Theta'_q = (I_K \otimes \Pi'_q), \qquad p,q = 0,1 \quad (9d)$$

and I_K and I_L are identity matrices of order K and L, respectively. Let us rewrite (9a) and (9b) in the following form:

$$Z_i = \Psi_0^{-1}(-\Psi_1 Z_{i-1} + \Theta_0 W_i + \Theta_1 W_{i-1}) \quad (10a)$$
$$Y_{i,j} = \Psi_0'^{-1}(-\Psi'_1 Y_{i,j-1} + Z_i). \quad (10b)$$

As a result, the 2-D block processor in this case is separated into two 1-D block processors. Using (10a), Z_i is computed by applying 1-D convolution operator

(a) to all the rows of $[X_{i,j}X_{i,j-1}]$, $[X_{i-1,j}X_{i-1,j-1}]$ and $[Y_{i-1,j}Y_{i-1,j-1}]$ blocks using equation (9b) and (9c) in order to obtain W_i, W_{i-1}, and Z_{i-1}, respectively,

(b) and then to all the columns of $[W_i \quad W_{i-1}]$ and Z_{i-1} using (10a). Note that Ψ_0^{-1} is itself a convolution operator.

Once Z_i has been computed, the desired output block $Y_{i,j}$ can be obtained using (10b).

3) Separable denominator 2-D recursive filter

In this case the operations on the output blocks can be separated into two 1-D convolutions whereas the operations on the input blocks will be of general nonseparable form.

4) 2-D non-recursive (FIR) filter

The block structure for these filters can be obtained by setting $a_{0,0}=1$ and $a_{i,j}=0$, $\forall (i,j) \neq (0,0)$ and representing $b_{i,j}$ by the impulse response sequence of the FIR filter. Thus $C_{00} = I_{KL}$ and $C_{01} = C_{10} = C_{11} = 0$ and the block equation becomes

$$Y_{i,j} = F_{00}X_{i,j} + F_{10}X_{i-1,j} + F_{01}X_{i,j-1} + F_{11}X_{i-1,j-1}. \quad (11)$$

III. 2-D Block-State Structure

Recursive digital filters can alternatively be described by a state-space realization model. A novel 2-D block-state realization model is introduced in this section. This multi-input, multi-output (MIMO) structure is derived [13] from Roesser's 2-D single-input, single-output (SISO) state-space model [15], [16] and is shown to have a form analogous to its scalar counterpart.

Let $u(i,j)$ and $y(i,j)$ represent the scalar input[1] and output, respectively. $R(i,j)$ is an $n_1 \times 1$ vector denoting the vertical state and $S(i,j)$ is an $n_2 \times 1$ vector denoting the horizontal state. The Roesser 2-D SISO state-space model is then given by

$$\begin{cases} R(i+1,j) = A_1 R(i,j) + A_2 S(i,j) + B_1 u(i,j) \\ S(i,j+1) = A_3 R(i,j) + A_4 S(i,j) + B_2 u(i,j) \end{cases}$$
$$y(i,j) = C_1 R(i,j) + C_2 S(i,j) + Du(i,j), \qquad \forall i,j \geq 0 \quad (12)$$

where A_1, A_2, A_3, A_4, B_1, B_2, C_1, C_2, and D are matrices of appropriate dimensions. For a global initial conditions, $R(0,j)$ and $S(i,0)$ which refer to the state along the edges should be externally specified. Let

$$A = \begin{bmatrix} A_1 & A_2 \\ A_3 & A_4 \end{bmatrix}_{n\times n} \quad B = \begin{bmatrix} B_1 \\ B_2 \end{bmatrix}_{n\times 1} \quad C = [C_1 \quad C_2]_{1\times n}$$

and

$$X(i,j) = \begin{bmatrix} R(i,j) \\ S(i,j) \end{bmatrix}_{n\times 1}, \qquad \text{where } n \triangleq n_1 + n_2.$$

Using these notations (12) may be written as

$$X(i,j) = A^{1,0}X(i-1,j) + A^{0,1}X(i,j-1) + B^{1,0}u(i-1,j) + B^{0,1}u(i,j-1)$$
$$y(i,j) = CX(i,j) + Du(i,j) \quad (13)$$

[1]Note that, we have denoted throughout this work, the input elements by $x(i,j)$ or $x_{i,j}$ but for the sake of consistency with Roesser's model, in this section the input is represented by $u(i,j)$.

where

$$A^{1,0}=\begin{bmatrix}A_1 & A_2\\ 0 & 0\end{bmatrix}\qquad A^{0,1}=\begin{bmatrix}0 & 0\\ A_3 & A_4\end{bmatrix}$$

$$B^{1,0}=\begin{bmatrix}B_1\\ 0\end{bmatrix}\quad \text{and}\quad B^{0,1}=\begin{bmatrix}0\\ B_2\end{bmatrix}.$$

As a result, a SISO state-space structure is characterized by a 4-tuple $\{A,B,C,D\}$ which realizes the original transfer function of the system, i.e.,

$$H(z,w)=C\left(\begin{bmatrix}zI_{n_1} & 0\\ 0 & wI_{n_2}\end{bmatrix}-A\right)^{-1}B+D \qquad (14)$$

where I_{n_1} and I_{n_2} are identity matrices of order n_1, n_2, respectively.

Definition 1

The state-transition matrix $A^{i,j}$ is defined as follows:

(a) $A^{i,j}=A^{1,0}A^{i-1,j}+A^{0,1}A^{i,j-1} \qquad (i,j)>(0,0)$

(b) $A^{0,0}=I$

(c) $A^{-i,j}=A^{i,-j}=0, \qquad \text{for } i\geqslant 1 \text{ and } j\geqslant 1.$ (15)

Having defined the 2-D state-space realization model, the problem of block-state realization involves finding a 4-tuple $\{\tilde{A},\tilde{B},\tilde{C},\tilde{D}\}$ which will completely describe MIMO systems and which can also yield the original scalar transfer function.

In order to derive such a model which describes the system of (12) or (13) in terms of blocks of size $K\times L$ of input and output data, the following "block-state" vectors are defined along the vertical and horizontal directions, respectively.

$$\tilde{R}(i,j)=\begin{bmatrix}\hat{R}(iK,jL)\\ \hat{R}(iK,jL+1)\\ \vdots\\ \hat{R}(iK,(j+1)L-1)\end{bmatrix}$$

$$\tilde{S}(i,j)=\begin{bmatrix}\hat{S}(iK,jL)\\ \hat{S}(iK+1,jL)\\ \vdots\\ \hat{S}((i+1)K-1,jL)\end{bmatrix} \qquad (16)$$

Where $\hat{R}(m,n)$ and $\hat{S}(m,n)$ are defined as

$$\hat{R}(m,n)=\begin{bmatrix}R(m,n)\\ 0\end{bmatrix},\qquad \hat{S}(m,n)=\begin{bmatrix}0\\ S(m,n)\end{bmatrix}.$$

The elements of block-state vector $\tilde{R}(i,j)$, which are mutually independent, represent the states associated with the boundary elements of the (i,j)th block along the horizontal direction. Similarly the elements of $\tilde{S}(i,j)$ represent the states associated with the boundary elements of the relevant block along the vertical direction.

Using these two vectors the block-state realization model can be completely characterized. In other words, the elements of $\tilde{R}(i+1,j)$ and $\tilde{S}(i,j+1)$ can be expressed in terms of the state elements of $\tilde{R}(i,j)$ and $\tilde{S}(i,j)$. Recursive applications of (12) and (15) for each element of the block-state vectors and arranging these elements in an appropriate manner yields (see Appendix 1) the following 2-D block-state realization model [13]

$$\begin{cases}\tilde{R}(i+1,j)=\tilde{A}_1\tilde{R}(i,j)+\tilde{A}_2\tilde{S}(i,j)+\tilde{B}_1TU(i,j)\\ \tilde{S}(i,j+1)\ =\tilde{A}_3\tilde{R}(i,j)+\tilde{A}_4\tilde{S}(i,j)+\tilde{B}_2U(i,j)\end{cases}$$

$$Y(i,j)=\tilde{C}_1\tilde{R}(i,j)+\tilde{C}_2\tilde{S}(i,j)+\tilde{D}U(i,j) \qquad (17)$$

where $\tilde{A}_1$ and $\tilde{A}_4$ are lower triangular block Toeplitz matrices of size $(Ln\times Ln)$ and $(Kn\times Kn)$, respectively, defined by their (i,j)th elements

$$\tilde{A}_{1_{i,j}}=\begin{cases}0, & i<j\\ A^{1,0}A^{K-1,i-j}, & i\geqslant j, \qquad i,j=1,\cdots,L\end{cases}$$

and

$$\tilde{A}_{4_{i,j}}=\begin{cases}0, & i<j\\ A^{0,1}A^{i-j,L-1}, & i\geqslant j, \qquad i,j=1,\cdots,K.\end{cases} \qquad (18a)$$

Matrice $\tilde{A}_2$ and $\tilde{A}_3$ that are of dimension $(Ln\times Kn)$ and $(Kn\times Ln)$, respectively, can be defined by their (i,j)th elements

$$\tilde{A}_{2_{i,j}}=A^{1,0}A^{K-j,i-1}, \qquad i=1,\cdots,L;\ j=1,\cdots,K$$

and

$$\tilde{A}_{3_{i,j}}=A^{0,1}A^{i-1,L-j}, \qquad i=1,\cdots,K;\ j=1,\cdots,L. \qquad (18b)$$

Matrices $\tilde{B}_1$ and $\tilde{B}_2$ are also lower triangular block Toeplitz of size $(Ln\times LK)$ and $(Kn\times KL)$, respectively, given by

$$\tilde{B}_{1_{i,j}}=\begin{cases}0, & i<j\\ \hat{B}_{i-j}, & i\geqslant j \qquad i,j=1,\cdots,L\end{cases}$$

and

$$\tilde{B}_{2_{i,j}}=\begin{cases}0, & i<j\\ \hat{B}'_{i-j}, & i\geqslant j, \qquad i,j=1,\cdots,K\end{cases} \qquad (19a)$$

Where the block matrices $\hat{B}_m$ and $\hat{B}'_m$ are defined by

$$\hat{B}_m=[A^{1,0}B^{K-1,m}\quad A^{1,0}B^{K-2,m}\cdots A^{1,0}B^{0,m}], \qquad m\geqslant 1$$

and

$$\hat{B}'_m=[A^{0,1}B^{m,L-1}\quad A^{0,1}A^{m,L-2}\cdots A^{0,1}B^{m,0}], \qquad m\geqslant 1.$$

For $m=0$ we have

$$\hat{B}_0=[B^{K,0}B^{K-1,0}\cdots B^{1,0}]$$

and

$$\hat{B}'_0=[B^{0,L}B^{0,L-1}\cdots B^{0,1}] \qquad (19b)$$

Moreover $B^{m,n}$ is defined by

$$B^{m,n}=A^{m-1,n}B^{1,0}+A^{m,n-1}B^{0,1}. \qquad (19c)$$

$\tilde{C}_1$ is a matrix of size $(KL \times Ln)$ given by

$$\tilde{C}_1 = \begin{bmatrix} \hat{C}_0 \\ \hat{C}_1 \\ \vdots \\ \hat{C}_{K-1} \end{bmatrix} \tag{20a}$$

where the block matrices $\hat{C}_m$ are lower triangular Toeplitz of size $(L \times Ln)$ defined by

$$\hat{C}_{m_{i,j}} = \begin{cases} 0, & i < j \\ CA^{m,i-j}, & i \geqslant j \quad i,j = 1,\cdots,L \end{cases} \tag{20b}$$

Matrix $\tilde{C}_2$ is also a lower triangular Toeplitz of size $(KL \times Kn)$ defined by

$$\tilde{C}_{2_{i,j}} = \begin{cases} 0, & i < j \\ \hat{C}'_{i-j}, & i \geqslant j \quad i,j = 1,\cdots,K \end{cases} \tag{21a}$$

where

$$\hat{C}'_m = \begin{bmatrix} CA^{m,0} \\ CA^{m,1} \\ \vdots \\ CA^{m,L-1} \end{bmatrix}. \tag{21b}$$

Matrix $\tilde{D}$ is a lower triangular block Toeplitz of size $(KL \times KL)$ given by

$$\tilde{D}_{i,j} = \begin{cases} 0, & i < j \\ \hat{D}_{i-j}, & i \geqslant j \quad i,j = 1,\cdots,K \end{cases} \tag{22a}$$

and

$$\hat{D}_{m_{p,q}} = \begin{cases} 0, & p < q \\ CB^{m,p-q} + D\delta(m, p-q), & p \geqslant q \end{cases} \quad p,q = 1,\cdots,L \tag{22b}$$

$\delta(\cdot,\cdot)$ denotes the Kronecker delta.

Finally matrix T in the first equation of (17) represents a column-to-row transformation matrix.

As a result, (17)–(22) give the desired 2-D block-state realization model which has a form similar to the Roesser 2-D state-space model. Moreover, the structure of this new model lends itself for analyzing 2-D MIMO systems and allows the use of linear transformations to generate additional new 2-D block structures. The schematic representation of this structure is shown in Fig. 2.

The necessary conditions for the 2-D SISO system of (12) to be stable is that the eigenvalues of matrices A_1 and A_4 lie in the interior of the unit circle in the complex plane [17]. Now referring to definitions of $\tilde{A}_1$ and $\tilde{A}_4$ in (18a), it can easily be deduced that the eigenvalues of these latter matrices are equal to those of A_1 and A_4, respectively, but raised to the power KL, where K and L are the block dimensions. In other words if the original 2-D SISO system is assumed to be stable the resultant MIMO system in (17) is somewhat more stable since the eigenvalues move closer to the origin of unit circle (for $K, L > 1$).

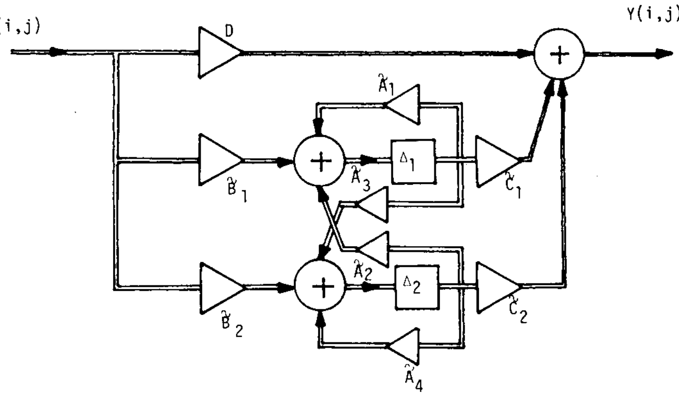


Fig. 2. Block-State Realization Form.

IV. Implementation Schemes for 2-D Block Recursive Processor

The implementations which have been suggested in the literature [2] for performing 1-D block processing usually involve the use of the FFT to accomplish some of the matrix–vector products by linear convolution.

In this section several possible implementation methods for 2-D block processing are considered. These methods differ in the amount of computation required (especially multiplications), the amount of storage required, and the effect of arithmetic roundoff. It is somewhat difficult to make a deterministic comparison between these methods, because of the tradeoffs between the various factors which depend on the complexity of the available hardware or software. However, because of the complexity of performing multiplications, the number of multiplications necessary to implement block recursive equation is often an important factor to be minimized.

In what follows, different implementation schemes using FFT, NTT, and PT are considered and their relative efficiencies are compared; an optimal implementation scheme is then proposed and used to perform the computations involved in 2-D block processing.

4.1. Implementation Using Fast Fourier Transform (FFT)

If the operations involved in the block recursive equation are all accomplished by direct matrix vector multiplications, using the definitions of C_{ij}'s and D_{ij}'s matrices in (5c) and (5d), the total number of multiplications required, N_{DMVP}, is

$$\begin{aligned} N_{\text{DMVP}} = {} & (M_a+1)(N_a+1) \\ & \cdot \left(\frac{M_a}{2}L + \frac{N_a}{2}K - \frac{M_a}{2} \cdot \frac{N_a}{2} \right) \\ & + (M_b+1)(N_b+1)KL \\ & + \frac{K(K+1)}{2} \cdot \frac{L(L+1)}{2}. \end{aligned} \tag{23}$$

Note that the number of multiplications required for the inversion of C_{00} is not included, since this operation can be

TABLE I

(K,L)	n_{FFT}	n_{PT}	n_{SC} $(M,N)=(2,2)$	n_{SC} $(M,N)=(4,4)$	n_{SC} $(M,N)=(8,8)$
(2,2)	20	27.5	19.75	*	*
(4,4)	32	40.62	24.94	*	*
(8,8)	62	60.78	35.11	56.36	*
(16,16)	97	74.26	48.1	65.77	*
(32,32)	134.5	107.16	62.55	78.22	*
(64,64)	173.25	121.71	77.78	92.38	144.17
(128,128)	212.63	141.17	93.4	107.45	155.9

*(Not efficient)
Total number of real multiplications per output points for implementation via FFT, PT and SC for different values of block sizes (K,L) and filter's order (M,N).

carried out beforehand using efficient transforms. The number of multiplications for this method increases largely with the order of the filter. Moreover, in this implementation the large dimension of the matrices (KL by KL) involved, places severe constraints on the choice of block sizes K and L, due to limitation of the amount of primary memory available. The operations in Equation (4) can efficiently be carried out using FFT. The operator C_{00}^{-1} represents a (K,L)-linear convolution which can be carried out, by appending an appropriate number of zeros, via a $(2K,2L)$-circular convolution; the rectangular operator $[D_{11}D_{01}D_{10}D_{00}]$ can be performed by $(K+M_b, L+N_b)$-circular convolution and similarly the operator $[C_{11}C_{01}C_{10}]$ can also be implemented by circular convolution of size $(K+M_a, L+N_a)$. The sequences $\{a_{i,j}\}$ and $\{b_{i,j}\}$ may be extended to dimension K by L, by adding a proper set of zeros, so that all these operations can be carried out by 2-D FFT of length $2K$ by $2L$.

Now, if all the operations involved in Equation (4) are carried out by an efficient FFT algorithm for real-valued data [18] then the total number of real multiplications required, N_{FFT}, for $K=2^t$ and $L=2^s$ will be

$$N_{\text{FFT}} = 10[N(t+1)L + N(s+1)K] + 12KL \quad (24)$$

where

$$N(p) = 2^p(p-3)+4.$$

For different block sizes (K,L), the number of real multiplications per output point, n_{FFT} is tabulated in Table I.

As in the 1-D case, the use of conventional FFT algorithm for 2-D block processing reduces the amount of computations considerably when compared with the direct method using matrix–vector multiplication. In addition, unlike the direct method, this implementation allows the blocks of large dimensions to be processed. However, the main disadvantage of this method relates to the use of trigonometric functions and of complex arithmetic even for real convolutions. This consequently introduces a significant amount of roundoff error in the computations. Moreover, the FFT implementations necessitates storing all the complex basis functions, hence requiring a considerable amount of storage which may be an important factor for a small minicomputer or a special purpose hardware.

4.2. Implementation Using Number Theoretic Transform (NTT)

In recent years, new transforms have been defined on the ring of integers which provide efficient and error-free means of computing circular convolutions. These transforms are called "number theoretic transforms" [9].

The computation of the NTT requires $N \log_2 N$ bit shifts, additions and subtractions, but no multiplications. The only multiplications required for an NTT implementation of circular convolution are the N multiplications required to multiply the transforms [19]. The major drawback of this transform is that the maximum possible transform length is limited by the word length of the machine used. This consequently imposes a rigid requirement on the length of the sequences to be convolved. Moreover, the implementation of modular arithmetic in general-purpose computers is usually inefficient.

A number of methods have been suggested [20], [21] to circumvent the problems associated with convolutions of long sequences by means of number theoretic transforms. The method proposed by Agarwal and Cooley [20] makes use of the Chinese Remainder Theorem (CRT) to convert a 1-D circular convolution of size $N = N_1 \cdot N_2 \cdot N_3 \cdots$ into a M-D convolution of dimension $N_1 \times N_2 \times N_3 \cdots$, circular in all dimensions, provided that N_i are "mutually prime." The length-N convolution is then computed by the nesting of several short convolutions having lengths $N_1, N_2, N_3, \cdots$.

The total number of multiplications for this transform equals to $M_1 M_2 M_3 \cdots$, where $M_1, M_2, M_3, \cdots$ are the number of multiplications required to compute the short convolutions of lengths $N_1, N_2, N_3, \cdots$ and are such that $M_i \simeq 2N_i - K_i$, where K_i is the number of divisors of N_i, including 1 and N_i.

When this technique is used to compute a 2-D circular convolution of size $P \times Q$, one should always remember that P and Q must not have a common factor, thus if

$$P = p_1 \cdot p_2 \cdots p_r$$

$$Q = q_1 \cdot q_2 \cdots q_s$$

p_i's and q_i's have to be mutually prime. In this case, using CRT index mapping and nesting various small convolutions, the total number of multiplications is

$$M_{\mathrm{RT}} = M^{(1)} \times M^{(2)} \tag{25}$$

where $M^{(1)}$ and $M^{(2)}$ are the total number of multiplications required for the convolutions along the first and second dimensions, respectively, and we have:

$$\begin{aligned} M^{(1)} &= M_{p_1} \cdot M_{p_2} \cdots M_{p_r} \\ M^{(2)} &= M_{q_1} \cdot M_{q_2} \cdots M_{q_s}. \end{aligned} \tag{26}$$

In order to use this technique for performing linear convolutions, the dimensions of the transforms should be chosen such that (a) the convolution along each dimension is circular (b) the dimensions do not contain a common factor. The number of multiplications for short convolutions can be extracted from [20].

The implementation of 2-D block recursive processing by this composite rectangular transform is found to be computationally very inefficient. Moreover, the rigid requirements on the size of these transforms give rise to severe problems in choosing their appropriate values. This transform requires storing a number of different short convolution algorithms with great structural complexity (due to CRT index mapping).

However, as far as the effect of roundoff error is concerned, the multidimensional mapping techniques implemented in modular arithmetic, introduces no roundoff error at any stage of the computation. Even if ordinary arithmetic is used, the rectangular transform implementation technique is likely to have less arithmetical roundoff error than an FFT implementation [20].

The method developed by Martens [21] is particularly useful for long integer sequences. However, when working in the fields of rationals, this transform reduces to polynomial transform which is described next.

4.3. *Implementation Using Polynomial Transforms (PT)*

More recently, new transforms defined in the rings of polynomials have been introduced, which provide fast computation of M-D convolutions. These transforms which are called polynomial transforms (PT) [22], [23], use ordinary arithmetic without multiplications. The only multiplications required to compute a 2-D convolution by polynomial transform correspond to performing a number of polynomial products. Moreover, unlike NTT, PT's do not have any limitations on the length of the sequences to be convolved.

The earliest work, that of Nussbaumer [22] was concerned with the computation of 2-D convolution of dimension $q \times q$, with q an odd prime, by polynomial transform using short convolution algorithms. Subsequently, he introduced [23] an efficient algorithm for large 2-D convolutions of dimension $N \times N$, with $N = 2^m$ which can be implemented with FFT-type algorithm, while avoiding the use of trigonometric functions and complex arithmetic as with FFT techniques, and which do not require storing a number of different short convolution algorithms as with other fast convolution techniques [20].

Now, if the 2-D block recursive equation is implemented by these polynomial transforms, since there are five 2-D convolutions of dimension $2K \times 2L$ which have to be computed, the total number of multiplication will be

$$N_{\mathrm{PT}} = 5 \cdot M_{\mathrm{PT}} \tag{27}$$

where M_{PT} is the total number of multiplications required to compute a 2-D circular convolution of dimension $2K \times 2L$ by polynomial transforms. Using the values of M_{PT} given in Table II in [22] for short convolutions and Table III in [23] for large convolutions, the number of multiplications per output point, n_{PT}, for this implementation are given in Table I. Comparing with the FFT algorithm, this transform is shown to provide a more efficient implementation scheme for 2-D block processing especially for large order filters (greater than 8×8). In addition, the roundoff error in this implementation is much lower than that of the FFT technique [23].

4.4. *Implementation Using Short Convolution Algorithm (SC)*

The short convolution (length 2) algorithm [24] is used in this section to develop an efficient implementation method for 2-D block processing.

Assuming that block with dimensions larger than the order of the filter are used, the Toeplitz matrices in Equation (4) will become relatively sparse. Now if these matrices are partitioned into blocks of size 2×2, each block containing non-zero elements, represents a length two, non-circular convolution operator, which can be carried out by three multiplications and five additions. This algorithm is optimal because, according to the Winograd Theorem [25] the minimum number of multiplications necessary for computing a length N noncircular convolution is $2N-1$.

Since the lower triangular elements of C_{00}^{-1} matrix are all nonzero, a substantial reduction in the total number of multiplications can be achieved, if this operation is carried out by FFT. In this case, the total number of multiplications, N_{SC}, is

$$\begin{aligned} N_{\mathrm{SC}} = {} & \frac{3KL}{2}(n_b+1)(M_b+1) \\ & + \frac{3K}{2} n_a (n_a+1)(M_a+1) \\ & + 3(n_a+1)\frac{(L-n_a)}{2}\frac{M_a(M_a+1)}{2} + N_{\mathrm{CC}} \end{aligned} \tag{28}$$

where

$$N_a = 2n_a, \quad N_b = 2n_b$$

and N_{CC} is the total number of real multiplications required to perform a circular convolution of dimension $2K$ by $2L$ by FFT, i.e.,

$$N_{\mathrm{CC}} = 4[N(t+1)L + N(s+1)K + KL] \tag{29}$$

and $N(\cdot)$ is defined in (24). A similar method can be used for odd values of N_a and N_b.

For different block sizes $K = 2^t$ and $L = 2^s$, and different order of the filter (M, N), the number of multiplications per output points, n_{SC}, given in Table I.

Comparison with the previously described method reveals that the method introduced in this section provides an efficient implementation scheme particularly when the order of the 2-D recursive filter is not greater than (8,8). For filters with orders equal or higher than (8,8), the implementation using polynomial transform would be more efficient.

The computations in this method are performed in ordinary arithmetic, hence resulting in less roundoff error than FFT or PT techniques. The computational time and computer memory allocations for this method would become minimum when the minimum values for block sizes are chosen. It has been shown that [14] the lower bound on the norm and mean-square value of the error produced due to roundoff of multiplication for a 2-D block implemented digital filter will be obtained when blocks of minimum dimensions are processed.

V. Conclusion

The block processing technique for 2-D recursive digital filters, is considered. For 2-D recursive digital filters realized by difference equation and state-space formulation, two different 2-D block recursive structures are introduced. In addition, several special cases have also been considered.

The block recursive processor, due to its unique structure, is shown to provide an accurate and efficient means of 2-D recursive filtering operation, when compared with available conventional methods. A number of different algorithms employing FFT, NTT and PT are employed in order to speed up the operations involved in 2-D block processing. The comparison between these algorithms is made based on several deterministic factors, and an efficient implementation technique for block recursive processing is then proposed, which makes use of length 2 short convolution algorithms. This method is particularly efficient when the order of the filter is not higher than (8,8). For filters with orders equal or greater than (8,8), the implementation using 2-D polynomial transforms is found to be more adequate.

As a result, the 2-D block recursive processor is shown to be a very powerful and versatile method in the implementation of 2-D recursive digital filters, especially when used in conjunction with fast transform techniques suggested in this paper.

Appendix 1

Consider the following relationships that are obtained by recursive application of (12) and (15):

$$\begin{aligned}\hat{R}(iK+K, jL+s) = {} & A^{K,0}\hat{R}(iK, jL+s) \\ & + \sum_{m=0}^{K-1} A^{K-m,0}\hat{S}(iK+m, jL+s) \\ & + \sum_{m=0}^{K-1} B^{K-m,0}u(iK+m, jL+s)\end{aligned} \tag{A.1}$$

and

$$\begin{aligned}\hat{S}(iK+r, jL+L) = {} & A^{0,L}\hat{S}(iK+r, jL) \\ & + \sum_{n=0}^{L-1} A^{0,L-n}\hat{R}(iK+r, jL+n) \\ & + \sum_{n=0}^{L-1} B^{0,L-n}u(iK+r, jL+n).\end{aligned} \tag{A.2}$$

For $\forall(i, j) \geqslant 0$ and $s \in [0, L-1]$, $r \in [0, K-1]$. To express the second terms of (A.1) and (A.2) in terms of the elements of $\tilde{S}(i, j)$ and $\tilde{R}(i, j)$, we can write

$$\begin{aligned}\hat{S}(iK+m, jL+s) & \\ = {} & \sum_{h=0}^{m} A^{0,1}A^{h,s-1}\hat{S}(iK+m-h, jL) \\ & + \sum_{t=0}^{s-1} A^{0,1}A^{m,s-1-t}\hat{R}(iK, jL+t) \\ & + \sum_{\substack{t=0 \\ (h \neq 0,\, t \neq s-1)}}^{s-1} \sum_{h=0}^{m} A^{0,1}B^{h,s-1-t}u(iK+m-h, jL+t) \\ & + B^{0,1}u(iK+m, jL+s-1), \qquad s \geqslant 1\end{aligned} \tag{A.3}$$

and similarly,

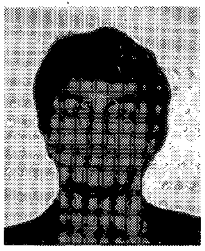
$$\begin{aligned}\hat{R}(iK+r, jL+n) & \\ = {} & \sum_{q=0}^{n} A^{1,0}A^{r-1,q}\hat{R}(iK, jL+n-q) \\ & + \sum_{v=0}^{r-1} A^{1,0}A^{r-1-v,n}\hat{S}(iK+v, jL) \\ & + \sum_{\substack{v=0 \\ (q \neq 0,\, v \neq r-1)}}^{r-1} \sum_{q=0}^{n} A^{1,0}B^{r-1-v,q}u(iK+v, jL+n-q) \\ & + B^{1,0}u(iK+r-1, jL+n), \qquad r \geqslant 1.\end{aligned} \tag{A.4}$$

Substituting (A.3) and (A.4) into (A.1) and (A.2) and forming $\tilde{R}(i+1, j)$ and $\tilde{S}(i, j+1)$, yields the desired block-state equation (17). For $Y(i, j)$ a similar technique can be employed.

References

[1] B. Gold and K. L. Jordan, Jr. "A note on digital filter synthesis," *Proc. IEEE*, vol. 56, pp. 1717–1718, Oct. 1968.
[2] C. S. Burrus, "Block realization of digital filters," *IEEE Trans. Audio Electroacoust.*, vol. AU-20, pp. 230–235, Oct. 1972.
[3] S. K. Mitra and R. Gnanasekaran, "Block implementation of recursive digital filters—New structures and properties," *IEEE Trans. Circuits Syst.*, vol. CAS-25, pp. 200–207, Apr. 1973.
[4] C. W. Barnes and S. Shinnaka, "Finite word effects in block-state realization of fixed-point digital filters," *IEEE Trans. Circuits Syst.*, vol. CAS-27, pp. 345–349, May 1980.
[5] C. W. Barnes and S. Shinnaka, "Block-shift invariance and block implementation of discrete-time filters," *IEEE Trans. Circuits Syst.*, vol. CAS-27, pp. 667–672, Aug. 1980.
[6] C. L. Nikias, "Fast block data processing via a new IIR digital filter structure," *IEEE Trans. Acoust., Speech, Signal Processing*, vol. ASSP-32, pp. 770–779, Aug. 1984.

[7] J. Loney and M. G. Strintzis, "Block implementation of two-dimensional digital filters," in *Proc. Int. Commun. Conf.*, 1978.
[8] S. K. Mitra and R. Gnanasekaran, "Block implementation of recursive two-dimensional digital filters," presented at 20th Midwest Symp. Circuits & Syst., Lubbock, TX, Aug. 1977.
[9] M. R. Azimi-Sadjadi and R. A. King, "Block implementation of 2-D digital filters," in *22nd Midwest Symp. on Circuits Syst.*, pp. 658–662, June 1979.
[10] M. R. Azimi-Sadjadi, "Block implementation of two-dimensional digital filters," in *Proc. MEDCOMP, 82*, Philadelphia, PA, pp. 160–169, Sept. 1982.
[11] S. K. Mitra and R. Gnanasekaran, "Block implementation of two-dimensional digital filters," *J. Franklin Inst.*, vol. 316, pp. 299–316, Oct. 1983.
[12] B. G. Mertzios, "Block realization of 2-D IIR digital filters," *Signal Processing*, vol. 7, no. 2, pp. 135–149, Oct. 1984.
[13] M. R. Azimi-Sadjadi, "A 2-D block-state realization model," in *Proc. IEEE* Int. 5th Symp. on Circuits and Syst., *Newport Beach, CA, pp.* 919–922, 1983.
[14] M. R. Azimi-Sadjadi and R. A. King, "An error analysis for 2-D block implemented digital filters," *IEEE Trans. Acoust. Speech, Signal Processing*, vol. ASSP-31, pp. 1570–1573, Dec. 1983.
[15] R. P. Roesser, "A discrete state-space model for linear image processing," *IEEE Trans. Automat. Contr.*, vol. AC-20, pp. 1–10, Feb. 1975.
[16] S. Y. Kung, B. C. Levy, M. Morf, and T. Kailath, "New results in 2-D systems theory, Part II: 2-D state space models—Realization and the notions of controllability, observability and minimality," *Proc. IEEE*, vol. 65, pp. 945–961, June 1977.
[17] W. S. Lu and E. B. Lee, "Stability analysis for two-dimensional systems via a Lyapunov approach," *IEEE Trans. Circuits Syst.*, vol. CAS-32, pp. 61–68, Jan. 1985.
[18] J. B. Martens, "Discrete Fourier transform algorithms for real-valued sequences," *IEEE Trans. Acoust., Speech, Signal Processing*, vol. ASSP-32, pp. 390–396, April 1984.
[19] R. C. Agarwal and C. S. Burrus, "Number theoretic transforms to implement fast digital convolution," *Proc. IEEE*, vol. 63, pp. 550–560, Apr. 1975.
[20] R. C. Agarwal and J. W. Cooley, "New algorithms for digital convolution," *IEEE Trans. Acoust., Speech, Signal Processing*, vol. ASSP-25, pp. 392–410, Oct. 1977.
[21] J. B. Martens, "Two-dimensional convolutions by means of number theoretic transforms over residue class polynomial rings," *IEEE Trans. Acoust., Speech, Signal Processing*, vol. ASSP-32, no. 4, Aug. 1984.
[22] H. J. Nussbaumer and P. Quandalle, "Computation of convolutions and discrete Fourier transforms by polynomial transforms," *IBM J. Res. Develop.*, vol. 22, pp. 134–144, Mar. 1978.
[23] H. J. Nussbaumer, "Fast polynomial algorithms for digital convolution," *IEEE Trans. Acoust., Speech, Signal Processing*, vol. ASSP-28, pp. 205–215, Apr. 1980.
[24] R. C. Agarwal and C. S. Burrus, "Fast one-dimensional digital convolution by multidimensional techniques," *IEEE Trans. Acoust., Speech, Signal Processing*, vol. ASSP-22, pp. 1–10, Feb. 1974.
[25] J. H. McClelland and C. M. Radar, *Number Theory in Digital Signal Processing.* Englewood Cliffs, NJ: Prentice-Hall, 1979.



Mahmood R. Azimi-Sadjadi (M'81) was born in Tehran, Iran, in 1952. He received the B.Sc. Degree from University of Tehran, Iran, in 1977, the M.Sc. and Ph.D. degrees from Imperial College University of London, England, in 1978 and 1982, respectively, all in electrical engineering.

Since September 1984, he has been with the Department of Electrical Engineering at The University of Michigan-Dearborn as an Assistant Professor of Electrical Engineering. His main research interests are in the areas of Digital Signal Processing, Multi-dimensional System Theory and Analysis, and Digital Image Processing.

Dr. Azimi-Sadjadi is the recipient of the 1984 Dow Chemical Outstanding Young Faculty Award of St. Lawrence Section of the American Society for Engineering Education.

Robert A. King, photograph and biography not available at time of publication.