Time-Recursive Computation and Real-Time Parallel Architectures: A Framework

Emmanuel Frantzeskakis, John S. Baras, *Fellow, IEEE,* and K. J. Ray Liu, *Senior Member, IEEE*

Abstract—**The time-recursive computation has been proven a particularly useful tool in real-time data compression, in transform domain adaptive filtering, and in spectrum analysis. Unlike the FFT-based ones, the time-recursive architectures require only local communication. Also, they are modular and regular, thus they are very appropriate for VLSI implementation and they allow a high degree of parallelism. In this two-part paper, we establish an architectural framework for parallel time-recursive computation. We consider a class of linear operators that consists of the discrete time, time invariant, compactly supported, but otherwise arbitrary kernel functions. We show that the structure of the realization of a given linear operator is dictated by the decomposition of the latter with respect to proper basis functions. An optimal way for carrying out this decomposition is demonstrated. The parametric forms of the basis functions are identified and their properties pertinent to the architecture design are studied. A library of architectural building modules capable of realizing these functions is developed. An analysis of the implementation complexity for the aforementioned modules is conducted. Based on this framework, the time-recursive architecture of a given linear operator can be derived in a systematic routine way.**

I. Introduction

THE discipline of time-recursive computation embraces a number of algorithms and architectures introduced in the context of diverse applications and under different names. First, the Goertzel algorithms (or Goertzel filters), introduced in 1958 [14] and later explored by other researchers [2], [3], [15], can be used for implementing an N-point discrete Fourier transform (DFT) in cases where only a small subset of the N-frequency components is desired [29]. During the last two decades, the running transforms have been used in frequency domain filtering [31] and transform domain adaptive filtering [4]. Several data transforms, such as the DFT, DCT, DST, and variations of them have been employed for accelerating the convergence and improving the performance in applications such as channel equalization, echo cancellation, adaptive line enhancing, and others [4], [9], [28], [38], [27], [15]. The advantage of the running algorithms over the fast algorithms is that for N consequitive evaluations of an N-point sliding transform the computational complexity is $O(N^2)$ compared to $O(N^2 \log_2 N)$ for the fast algorithm implementation. The same rationale applies for realizing the sliding transforms that are used in spectrum analysis, the DFT being the most popular among them [31], [4]. A nonsinusoidal transform used in this context was realized in a time-recursive way independently in [1] and [11]. Nonrectangular data windowing can be embodied in the time-recursive implementation of the short time fourier transform (STFT) and the time-recursive design can be generalized for multiple dimensions [21].

The term "time-recursive" has first appeared in [8] in the context of real-time data compression. Unlike adaptive filtering and spectrum estimation, where a sliding transform is desired, in data compression schemes the transform coefficients have to be evaluated in a block by block manner. The subtle point in the real-time, time-recursive implementation of the block transforms hinges on the fact that the operators need to evaluate one result per time unit,[1] while an operator in the fully parallel and pipelined FFT needs to produce one result every N time units. Apparently, this is the reason that has discouraged the use of time-recursive computation in data coding until recently [8], [5]. The situation has been changed due to the advances in the VLSI technology that penalize more the global communication than the requirement for short internal clock cycle. In particular, note that the FFT-based architectures that employ global interconnection butterfly networks require area $O(N^2)$ [35, pp. 216–219], while the recursive computation in [15] has only $O(N)$ complexity. As a side effect, the speed of a (VLSI implemented) operator can match the input data rate, by adjusting the length of the clock cycle [7], [5]. As long as this synchronization constraint is satisfied for a real-time application, area minimization becomes the major concern in the design, while latency and power consumption should confine with application dependent restrictions. Under this light, the success of the time-recursive VLSI circuits in evaluating block transforms and the promise they show are mainly justified, apart from the modularity, regularity, and scalability of the design, by virtue of the area optimality property and the communication locality property. This has been clearly demonstrated recently for a number of individual examples, DCT being the most prevalent among them. Descriptions of the architecture details, complete VLSI layout floor plans, discussions on the finite word-length implementation effects, as well as comparisons with competent techniques have been provided [8], [22], [7], [5], [30]. Furthermore, the time-recursive architectures are very efficient for separable multidimensional data transforms.

Manuscript received July 22, 1993; revised April 10, 1995. This work was supported by NSFD CDR 8803012 through the Engineering Research Center's Program, as well as the ONR Grant N00014-93-1-0566 and NSF NYI Award MIP 9457397. The associate editor coordinating the review of this paper and approving it for publication was Prof. Keshab Parhi.

E. Frantzeskakis is with INTRACOM, Athens, Greece.

J. S. Baras and K. J. R. Liu are with the Electrical Engineering Department, Institute of Systems Research, University of Maryland, College Park, MD 20742 USA.

IEEE Log Number 9415072.

[1] The time unit is the time that lapses between two adjacent input data.

In particular, the implementation cost is linear in terms of operator counts and the communication requirement remains local. The induction procedure for designing multidimensional architectures based on the 1-D ones is described in [21] and [23], while a detailed example is given in [8].

The purpose of this paper is to identify the scope of applicability of the time-recursive computation, so that the exploitation of the well established advantages of this technique becomes feasible for the widest possible spectrum of applications. We establish an architectural framework for parallel time-recursive computation. We show that all the aforementioned algorithmic and architectural designs exhibit a common infrastructure. We consider a class of linear operators that consists of the discrete time, time invariant, compactly supported, but otherwise arbitrary kernel functions. We specify the properties of the linear operators that can be implemented efficiently in a time-recursive way. Based on these properties, one can develop a time-recursive architectural implementation for a given operator in a routine way. See, for example, [12], [13]. Here, we briefly interpret the results presented in [12] regarding the modulated lapped transform (MLT) [24], [26], oftentimes referred in the data coding community as modified DCT (MDCT) [17].

The rest of this paper is organized as follows. In Section II, we introduce some terminology. In Section III, we study the time-recursive algorithmic structures and their properties. In Section IV, we focus on the architectural implementation of time-recursive architectures. In Section V, we briefly discuss the special features pertinent to block data transforms. We conclude with Section VI. In the Appendix, we give the proofs of some lemmas that are stated in the course of the paper.

II. Preliminaries

In many signal processing applications the key computation consists of a *mapping operator* $[h_0 \quad h_1 \quad \cdots h_{N-1}]: x(\cdot) \rightarrow X(\cdot)$, which operates on the semi-infinite sequence of scalar data $x(\cdot)$ and produces the sequence $X(\cdot)$ as follows:

$$X(t) = \sum_{n=0}^{N-1} h_n x(t+n-N+1), \qquad t = 0, 1, \cdots. \quad (1)$$

Note that all FIR filters can be considered as this type of computation. This is also true for a number of data transforms. For example, the kth frequency component of the N-point DFT is obtained for $h_n = e^{-j(2\pi/N)kn}$.

We can specify a mapping operator $[h_0 \quad h_1 \quad \cdots h_{N-1}]$ with a function $f(\cdot)$, for which the values at the points $0, 1, \cdots, N-1$ are the prescribed coefficients $h_n = f(n), n = 0, 1, \cdots, N-1$. In the sequel, we will use the term *kernel function* or simply *kernel* for this function $f(\cdot)$. For example, the kernel $f(n) = e^{\alpha n}$ is associated to the operator $[e^{\alpha n}, n = 0, 1, \cdots, N-1]$. Furthermore, we will call *kernel group* a vector of kernel functions $f_0(\cdot), f_1(\cdot), \cdots, f_{M-1}(\cdot)$:

$$\mathbf{f}(\cdot) = [f_0(\cdot) \quad f_1(\cdot) \quad \cdots \quad f_{M-1}(\cdot)]^T.$$

A *time-recursive implementation* of a mapping operator $[h_n \quad h_1 \quad \cdots \quad h_{N-1}]$ is the one that is based on a recursive update computation of the type

$$X(t+1) = \mathcal{U}(X(t), x(t-N+1), x(t+1)).$$

For example, the kth frequency component of the N-point DFT can be extracted as follows [31]:

$$X_k(t+1) = e^{j(2\pi/N)k}[X_k(t) + x(t+1) - x(t-N+1)].$$

III. Design of Time-Recursive Algorithm

A. Shift Property

In the course of our study, we will see that *all* mapping operators specified in (1) can be implemented in a time-recursive way. Nevertheless, such an implementation is not always attractive compared to feed-forward ones.

Let us first introduce the *shift property* of kernel groups.

Definition: A kernel group $\mathbf{f}(\cdot) = [f_0(\cdot) \quad f_1(\cdot) \quad \cdots \quad f_{M-1}(\cdot)]^T$, satisfies the shift property (SP) if it satisfies the (matrix) difference equation

$$\mathbf{f}(n-1) = \mathbf{R}\mathbf{f}(n), \qquad n = 1, 2, \cdots, N \quad (2)$$

with a specified final condition $\boldsymbol{bf}(N)$, where $\mathbf{R}$ is a constant matrix of size $M \times M$. Furthermore, we shall say that a kernel function $\phi(\cdot)$ satisfies SP if there is a kernel group $\mathbf{f}(\cdot)$ that satisfies SP and $\phi(\cdot)$ is an element of $\mathbf{f}(\cdot)$.

With the following lemma, we specify a family of kernels and kernel groups that can be implemented time-recursively in a way that will be determined shortly.

Lemma 3.1: A time-recursive implementation of a kernel group $\mathbf{f}(\cdot)$ is feasible if this kernel group satisfies the shift property.

Proof: Equation (2) gives

$$f_p(n-1) = \sum_{q=0}^{M-1} r_{qp} f_q(n), \qquad n = 1, 2, \cdots, N,$$
$$p = 0, 1, \cdots, M-1$$

where $r_{qp}, p, q = 0, 1, \cdots, M-1$ are the elements of the matrix $\mathbf{R}$. Let

$$X_p(t) = \sum_{n=0}^{N-1} f_p(n) x(t+n-N+1),$$
$$p = 0, 1, \cdots, M-1. \quad (3)$$

Suppose this is available at the time instant $t+1$. For the quantities $X_p(t+1), p = 0, 1, \cdots, M-1$ we have

$$\begin{aligned} X_p(t+1) &= \sum_{n=0}^{N-1} x(t+n+1-N+1) f_p(n) \\ &= \sum_{n=1}^{N} x(t+n-N+1) f_p(n-1) \\ &= \sum_{n=1}^{N} x(t+n-N+1) \sum_{q=0}^{M-1} r_{qp} f_q(n) \\ &= \sum_{q=0}^{M-1} r_{qp} \left(\sum_{n=1}^{N} x(t+n-N+1) f_q(n) \right) \end{aligned}$$

and therefore, we obtain the algorithm:

$$X_p(t+1) = \sum_{q=0}^{M-1} r_{qp}[X_q(t) - x(t-N+1)f_q(0) + x(t+1)f_q(N)] \quad (4)$$

where $p = 0, 1, \cdots, M-1$. If we assume knowledge of the boundary values $\{f_q(0), f_q(N), q = 0, 1, \cdots, M-1\}$, the algorithm specified in (4) will become the update computation we were after. Equation (2) implies that knowledge of $\mathbf{f}(N)$ yields $\mathbf{f}(0)$. Furthermore, note that if $\mathbf{R}$ is nonsingular, knowledge of $\mathbf{f}(0)$ yields $\mathbf{f}(N)$. □

Corollary 1: A kernel group $\mathbf{f}(\cdot)$ that satisfies SP can be implemented time-recursively as follows:

1) Compute the matrix $\mathbf{R}$ by evaluating $\mathbf{f}(n-1)$ and using (2).
2) Evaluate $\mathbf{f}(n)$ at the points $n = 0$ and $n = N$.
3) At each time instant t evaluate (4).

Note that the first two steps of the above algorithm belong to the initialization phase (off-line computation).

B. Scope of Time-Recursive Computation

The issue of specifying a family of kernel groups that satisfy SP is addressed by Lemma 3.2:

Lemma 3.2: The shift property is satisfied by the following:

1) The singleton kernel group $[cb^N]$, where b and c are nonzero free parameters.
2) The kernel group
$$[c_{00}b^n + c_{10}b^{-n}, c_{01}b^n + c_{11}b^{-n}]^T \quad (5)$$
where b is a nonzero parameter and the coefficients are free parameters, such that $c_{00}c_{11} - c_{01}c_{10} \neq 0$.
3) The kernel group $[c_0, c_1 n, \cdots, c_{M-1}n^{M-1}]^T$, where the coefficients are nonzero parameters.

Proof: One can readily verify that the associated matrices $\mathbf{R}^{(i)}, i = 1, 2, 3$, respectively, are

$$\mathbf{R}^{(1)} = \frac{1}{b},$$

$$\mathbf{R}^{(2)} = \begin{bmatrix} c_{00} & c_{10} \\ c_{01} & c_{11} \end{bmatrix} \begin{bmatrix} b^{-1} & 0 \\ 0 & b \end{bmatrix} \begin{bmatrix} c_{00} & c_{10} \\ c_{01} & c_{11} \end{bmatrix}^{-1} \quad \text{and}$$

$$\mathbf{R}^{(3)} = [r_{qp}]_{p,q=0,1,\cdots,M-1},$$

$$r_{qp} = \begin{cases} \dfrac{c_p}{c_q}\binom{p}{q}(-1)^{p-q}, & q \leq p \\ 0, & q > p. \end{cases}$$ □

Suppose now that we are given a mapping operator $[h_0 \quad h_1 \quad \cdots \quad h_{N-1}]$ for which we have the following linear decomposition:

$$h_n = \alpha\phi(n) + \beta\psi(n), \qquad n = 0, 1, \cdots, N-1$$

where $\phi(\cdot)$ and $\psi(\cdot)$ are kernel functions that satisfy SP. Since we have

$$X(t) \triangleq \sum_{n=0}^{N-1} h_n x(t+n-N+1) = \alpha X_\phi(t) + \beta X_\psi(t)$$

where $X_\phi(t)$ and $X_\psi(t)$ have the obvious definitions, we can obtain an efficient time-recursive implementation for $[h_0 \quad h_1 \quad \cdots \quad h_{N-1}]$ The mapping operators generated by this linearity property supplement the family of the operators that can be computed in a time-recursive way dictated by Lemma 3.2.

One can generate all the transform kernels that have been employed in the literature referenced in Section I with proper choice of the kernel parameters specified by Lemma 3.2. In particular, for $c = 1$ and $b = e^{j2k\pi/N}$, Statement 1 yields the kernel functions of the DFT.

By virtue of the fact that every mapping operator of finite length N can be expressed as a combination of exponential functions (by taking for example the DFT of the mapping operator coefficients), we conclude that all such operators can be implemented in a time-recursive way. In this perspective, Lemma 3.2 provides a completeness result. In other words, it provides a basis of kernel functions so that every mapping operator of finite length can be expressed as a linear combination of the basis functions.

C. Systematic Design I

In what follows, we summarize the steps to be taken in order to formulate the computation specified by a mapping operator in a time-recursive manner. We assume here that the given operator can be expressed by inspection (and use of Lemma 3.2) as a linear combination of kernel functions that satisfy SP. For example, the kernel functions of the discrete sinusoidal transforms belong in this class of operators (cf. Lemma 3.2, Statement 2).

Design Procedure

Input:

$$h_n = \sum_i c_i\phi_i(n) \quad (6)$$

where $\{\phi_i(n)\}$ is a set of kernel functions that satisfy the shift property SP and $\{c_i\}$ is a set of known constants.

Step 1: Specify the kernel groups $\mathbf{f}_i(\cdot)$ in which the kernel functions $\phi_i(\cdot)$ belong. For example, if $\phi_i(n) = n^2$ then, according to Lemma 3.2, Statement 3, we get $\mathbf{f}_i(n) = [1 \quad n \quad n^2]^T$.

Step 2: For each kernel group $\mathbf{f}_i(\cdot)$ use (2) in order to compute the matrix of parameters $\mathbf{R}_i$ and evaluate $\mathbf{f}_i(n)$ at the points $n = 0$ and $n = N$.

The outcome of this design procedure is the following algorithm:

1) Evaluate (4) in order to obtain $X_i(t+1)$, where $X_i(t)$ is defined as $X_i(t) = \Sigma_{n=0}^{N-1} \phi_i(n)x(t+n-N+1)$.
2) Evaluate
$$X(t) = \sum_i c_i X_i(t). \quad (7)$$

Detailed examples along the lines of this procedure are discussed in [11] and [10].

D. Mapping Operator Decomposition

If the mapping operator is not specified in the form (6), for example, if we are given the vector of the coefficients instead

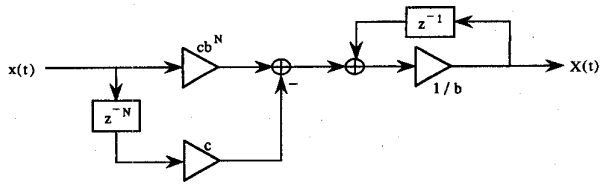


Fig. 1. Architecture for kernel group of size $M = 1$.

of a closed-form expression, an elaborate technique must be employed in order to obtain the linear expression required as the input of the design procedure. For any mapping operator a number of different time-recursive realizations exist, since the above mentioned decomposition is not unique. Given a mapping operator, we would like to obtain the optimal time-recursive implementation in terms of the architectural cost. Unfortunately, this is not an easy problem, since a variety of ad hoc designs may exist for a specified operator. Here, we address the question of optimality with respect to the number of kernels that are used in a linear decomposition of a given mapping operator.

Lemma 3.3: The size of the smallest kernel group that can be used to implement the mapping operator $[h_0 \quad h_1 \quad \cdots \quad h_{N-1}]$ in a time-recursive way is equal to the size of the minimal order partial realization of the linear time invariant (LTI) system with the N first Markov parameters[2] being equal to the coefficients of the specified operator.

Proof: Given a mapping operator $[h_0 \quad h_1 \quad \cdots h_{N-1}]$ we can have the following coefficient expansion:

$$h_n = \mathbf{c}\mathbf{A}^n\mathbf{b}, \qquad n = 0, 1, \cdots, N-1 \tag{8}$$

where $\mathbf{A}$ is the system matrix of size $M \times M$ and $\mathbf{b}, \mathbf{c}$ are the input and output vectors, respectively [18], [20]. Let

$$\mathbf{f}(n) = \mathbf{A}^n\mathbf{b} \tag{9}$$

be a kernel group of size M. Since $\mathbf{f}(n-1) = \mathbf{A}^{n-1}\mathbf{b} = \mathbf{A}^{-1}\mathbf{f}(n)$, this kernel group satisfies the shift property with

$$\mathbf{R} = \mathbf{A}^{-1} \quad \text{and} \quad \mathbf{f}(0) = \mathbf{b}. \tag{10}$$

From (8) and (9), we get the linear decomposition of the mapping operator coefficients $h_n = \mathbf{c}\mathbf{f}(n)$. Therefore, the time-recursive implementation of the mapping operator can be based on the kernel group $\mathbf{f}(\cdot)$. In our construction, the size of the kernel group M is equal to the order of the realization $\{\mathbf{A}, \mathbf{b}, \mathbf{c}\}$. □

Thus, by using Lemma 3.3 we can obtain a time-recursive algorithm for an arbitrary mapping operator based on the minimum number of kernels. The extended algorithm design procedure is described in the following subsection.

E. Systematic Design II

For the time-recursive implementation of an arbitrary mapping operator $h_1 \quad \cdots h_{N-1}]$ three steps need to be added at the beginning of the design procedure in Section III-C:

Design Procedure Supplement

Input: The mapping operator $h_1 \quad \cdots h_{N-1}]$

Step 0.1: Compute the quantities $\mathbf{A}, \mathbf{b}$, and $\mathbf{c}$ in (8) [18], [20].

Step 0.2: Use the similarity transform that will yield $\{\mathbf{A}, \mathbf{b}, \mathbf{c}\}$ in the modal canonical form.[3]

Step 0.3: Calculate the closed-form expression for the operator coefficients.

The expression specified in Step 0.3 can be used as the input in the design procedure described in Section III-C.

Note that Step 0.1 returns a state space description of an LTI system in the controller canonical form. By transforming this system in the modal canonical form we are able to compute the closed form of the elements in matrix $\mathbf{A}^n$ (since this is a block diagonal matrix where the blocks are either rotation matrices or real scalars). Consequently, Step 0.3 can be carried out by simple algebraic manipulations.

In conclusion, the above design procedure yields a realization for which the associated matrix $\mathbf{R}$, first, has the minimum possible size, and second, it is block diagonal with block elements either real scalars or 2×2 plane rotation matrices. In Section IV, we will see that both of these features are very desirable for the architectural implementation.

F. Difference Equation Property

A fundamental property of the Markov parameters $\{h_n = \mathbf{c}\mathbf{A}^n\mathbf{b}, n = 0, 1, \cdots\}$ of LTI systems dictates [18]:

$$h_{n+M} + \alpha_1 h_{n+M-1} + \cdots + \alpha_M h_n = 0$$

where $\alpha_p, p = 1, 2, \cdots, M$ are the constants specifying the system matrix $\mathbf{A}$ in the controller canonical form [18]. Equivalently, this can be written in a difference equation format as follows:

$$h_n = \gamma_1 h_{n-1} + \cdots + \gamma_M h_{n-M} \tag{11}$$

where

$$\gamma_p = -\alpha_p, \qquad p = 1, 2, \cdots, M. \tag{12}$$

Let $\mathbf{e}_p$ be the row vector of length M, for which the pth element is unity and all other elements equal zero. If vector $\mathbf{c}$ equals $\mathbf{e}_p$, then (8) implies that h_n is the pth kernel function of the kernel group $\mathbf{f}(\cdot)$. Suppose now that $\mathbf{A}$ and $\mathbf{b}$ are of the form specified in controller canonical form. Then, all kernel functions in (9) satisfy the same difference equation (11). Lemma 3.4, which follows, states that this is true even if $\mathbf{A}$ and $\mathbf{b}$ do not have any special structure. Thus, it introduces the difference equation property of a kernel group:

Definition: A kernel group $\mathbf{f}(\cdot) = [f_0(\cdot)\, f_1(\cdot) \cdots f_{M-1}(\cdot)]^T$,satisfies the difference equation property (DEP) if there are scalars $\gamma_p, p = 1, 2, \cdots, M$, independent of n, such that the kernel functions $f_q(\cdot), q = 0, 1, \cdots, M-1$ satisfy the following difference equation

$$f_q(n) = \gamma_1 f_q(n-1) + \cdots + \gamma_M f_q(n-M), \qquad n = 1, 2, \cdots, N \tag{13}$$

with specified initial conditions $f_q(n), n = -1, -2, \cdots, -M$.

[2] For the definition of the Markov parameters of an LTI system, see [18, pp. 92–93].

[3] For the definition of similarity transforms and the canonical realization forms for LTI systems one may refer to [18].

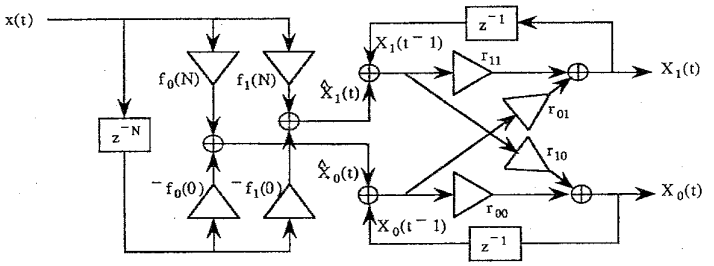


Fig. 2. Lattice architecture for kernel group of size $M = 2$.

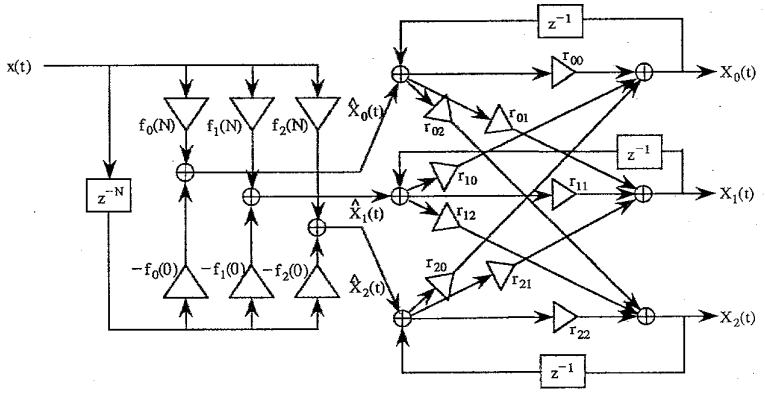


Fig. 3. Lattice architecture for kernel group of size $M = 3$.

Lemma 3.4: A kernel group satisfies DEP if and only if it satisfies SP.

The proof of this Lemma is given in the Appendix.

IV. Design of Time-Recursive Architecture

A. Lattice Architecture Design for Mapping Operators

In Section III, we introduced a unifying approach for formulating the computation specified by a mapping operator in a time-recursive manner. A key role in this formulation is played by the evaluation of the expression in (4). The architectural implementation of (4) will have a lattice structure if the size of the associated kernel group is $M = 2$ (see Fig. 2). An example of this architecture appears in [22]. In an abuse of terminology, we will call *lattice architectures* the architectures that implement (4) regardless of the size of the kernel group. The lattice architecture that implements a kernel group of size $M = 3$ is depicted in Fig. 3. The overall architecture design is completed by a simple weighted-sum circuit that evaluates (7). We can observe that this architecture consists of M two-tap FIR filters and a $M \times M$ weighted interconnection network with M feedback loops. The total cost of this structure is no more than $M^2 + 2M$ multipliers and $M(M-1) + 2M = M^2 + M$ two-input adders. The weighted-sum circuit consists of M multipliers and $M - 1$ adders. The cost of the overall implementation is given on Table I (lattice architecture).

The $M \times M$ weighted interconnection network is characterized by the matrix R specified in (10). If we follow all five steps of the design procedure described in Sections III-C and III-E, the matrix R will be block diagonal with blocks consisted of plane rotations. Consequently, we can implement the interconnection network very efficiently, with locally interconnected rotation circuits. The latter can be realized either with CORDIC processors [16] or with distributed arithmetic techniques [34]. The cost for implementing a mapping operator with this approach is shown on Table I (lattice/modal). Furthermore, with this setup we can exploit the fact that the absolute values of all the eigenvalues of a lossless system have the same magnitude [36], [37]. The lossless QMF bank implementation presented in [11] takes advantage of this fact to reduce the number of multipliers to be implemented.

B. Periodicity Property

With regard to the structure depicted in Fig. 3, suppose that there are two constants D_1 and D_2 such that the relation

$$\hat{X}_p(t) = D_p \hat{X}_0(t) \tag{14}$$

is true for $p = 1, 2$ and $t = 1, 2, \cdots$. Then, one can verify that the three two-tap filters in Fig. 3 can be replaced by the structure shown in Fig. 4(a). The corresponding circuit for $M = 2$ is given in Fig. 4(b). In this way, $M - 1$ multipliers and an equal number of adders are saved. Obviously, the same modification can be applied for a kernel group of arbitrary size. The resulted cost metrics are depicted in Table I (case b). In Lemma 4.1, which follows, we state a condition on the

TABLE I
IMPLEMENTATION COST OF A MAPPING OPERATOR, BASED ON A KERNEL GROUP OF SIZE M: CASE a, THE OPERATOR DOES NOT SATISFY THE PERIODICITY PROPERTY AND IT IS UTILIZED BY A SLIDING TRANSFORM. CASE b, THE OPERATOR SATISFIES THE PERIODICITY PROPERTY AND IT IS UTILIZED BY A SLIDING TRANSFORM. CASE c, THE OPERATOR IS UTILIZED BY A BLOCK TRANSFORM

		multipliers	adders	rotations
Case a.	lattice architecture	M^2+3M	M^2+2M-1	-
	lattice / modal	$2M$	$\lfloor 5M/2+1 \rfloor$	M
	IIR architecture	$3M$	$3M-1$	-
Case b.	lattice architecture	M^2+2M+1	M^2+M	-
	lattice / modal	$2M$	$\lfloor 5M/2+1 \rfloor$	$M/2$
	IIR architecture	$2M$	$2M$	-
Case c.	lattice architecture	M^2+2M+1	M^2+M-1	-
	lattice / modal	$2M$	$\lfloor 5M/2 \rfloor$	$M/2$
	IIR architecture	$2M$	$2M-1$	-

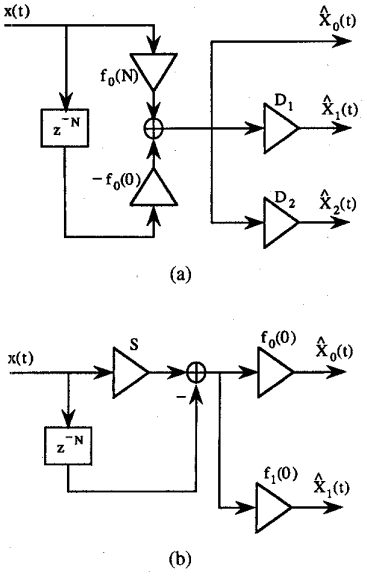


Fig. 4. Part of lattice architecture if the periodicity property is satisfied.

kernel functions that imply (14) and consequently the savings mentioned above can be obtained.

First, let us introduce the *periodicity property* of kernel groups.

Definition: A kernel group $\mathbf{f}(\cdot) = [f_0(\cdot) \quad f_1(\cdot) \quad \cdots \quad f_{M-1}(\cdot)]^T$, satisfies the periodicity property (PP) if the following relation holds:

$$\frac{f_0(N)}{f_0(0)} = \frac{f_1(N)}{f_1(0)} = \cdots = \frac{f_{M-1}(N)}{f_{M-1}(0)} = \frac{1}{S} \tag{15}$$

for some nonzero constant S.

Lemma 4.1: Given a kernel group $\mathbf{f}(\cdot)$ relation (14) holds for $p = 1, 2, \cdots, M-1$ and $t = 0, 1, \cdots$ if $\mathbf{f}(\cdot)$ satisfies the periodicity property.

The proof of Lemma 4.1 is given in the Appendix.

The name *periodicity property* is justified by the following special case: consider the kernel group specified by Statement 2 in Lemma 3.2. In the Appendix we prove the following Lemma:

Lemma 4.2: If parameter b of the kernel group (5) is of the form $b = e^{j\beta}$, then (5) satisfies the periodicity property if and only if $\beta = k\pi/N$, that is, if the kernel functions are periodic with period equal to N. Furthermore, if PP is satisfied the ratio value in (15) is equal to $1/S = (-1)^k$.

An example of kernel group that satisfies PP is the one that consists of the DCT and DST kernels

$$\mathbf{f}_k(n) = \left[\cos \frac{k\pi}{N}\left(n+\frac{1}{2}\right) \sin \frac{k\pi}{N}\left(n+\frac{1}{2}\right)\right]^T.$$

C. IIR Architecture Based on Shift Property

The *lattice architecture* we have seen in Section IV-A constitutes a direct translation of (4) into an architectural implementation. If a transfer function approach is adopted instead, we obtain an IIR filter structure implementation for (1) [23]. In this subsection, we show how we can specify the IIR implementation of a kernel group based on the shift property, while the IIR architecture design based on the difference equation property is the subject of the following subsection. The *IIR architecture* often involves less implementation cost in comparison to the lattice one, especially if the associated kernel group exhibits *the periodicity property* we have seen in the previous subsection.

Lemma 4.3: Let $f_p(\cdot)$ be a kernel function in the kernel group $\mathbf{f}(\cdot) = [f_0(\cdot) \quad f_1(\cdot) \quad \cdots \quad f_{M-1}(\cdot)]^T$ of size M. If $\mathbf{f}(\cdot)$ satisfies SP, the kernel function $f_p(\cdot)$ can be implemented by an IIR filter with transfer function $H_p(z)$

$$H_p(z) = \frac{b_p^0(z)}{a(z)} - z^{-N}\frac{b_p^1(z)}{a(z)}, \qquad p = 0, 1, \cdots, M-1 \tag{16}$$

where $a(z)$ is a polynomial in z^{-1} of degree M and $b_p^i(z), i = 0, 1$ are polynomials in z^{-1} of degree $M-1$. These are defined as follows: $a(z) = |\mathbf{A}(z)|, b_p^i(z) = |\mathbf{b}_p^i(z)|, i = 0, 1$, where (17), as given at the bottom of the page $\mathbf{b}_p^i(z)$ is an $M \times M$ matrix formed by substituting the pth column of $\mathbf{A}(z)$ with $[s_0^i \quad s_1^i \quad \cdots \quad s_{M-1}^i]^T, i = 0, 1$, and

$$s_p^0 = -\sum_{q=0}^{M-1} r_{qp} f_q(0),$$

$$s_p^1 = -\sum_{q=0}^{M-1} r_{qp} f_q(N), \qquad p = 0, 1, \cdots, M-1.$$

Note that $|\mathbf{X}|$ denotes the determinant of the matrix $\mathbf{X}$.

$$\mathbf{A}(z) = \begin{bmatrix} -1+r_{00}z^{-1} & r_{10}z^{-1} & \cdots & r_{P-1,0}z^{-1} \\ r_{01}z^{-1} & -1+r_{11}z^{-1} & \cdots & r_{P-1,1}z^{-1} \\ \vdots & \vdots & \ddots & \vdots \\ r_{0,P-1}z^{-1} & r_{1,P-1}z^{-1} & \cdots & -1+r_{P-1,P-1}z^{-1} \end{bmatrix} \tag{17}$$

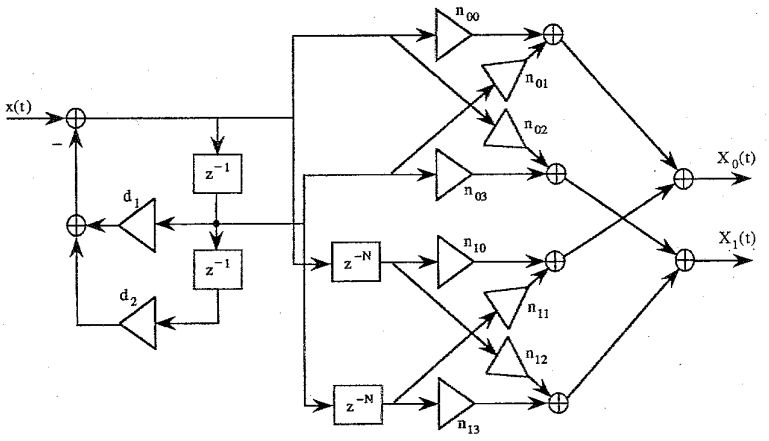


Fig. 5. IIR architecture for $M = 2$.

The proof is given in the Appendix. As a direct consequence of this Lemma we have the following corollary.

Corollary 2: Let $\mathbf{f}(\cdot) = [f_0(\cdot) \quad f_1(\cdot) \quad \cdots \quad f_{M-1}(\cdot)]^T$ be a kernel group of size M that satisfies PP. Then, the transfer function $H_p(z)$ of the linear system that models (4) is

$$H_p(z) = (S - z^{-N})\frac{b_p^1(z)}{a(z)}, \qquad p = 0, 1, \cdots, M-1 \quad (18)$$

where $a(z)$ and $b_p^1(z)$ are specified in Lemma 4.3, and S is the constant specified in (15).

For the sake of clarity, we will consider the special case of a kernel group of size $M = 2$ in detail. Let $H_p(z)$ be the transfer function of the linear system that models the mapping operators

$$[f_p(0) \quad f_p(1) \quad \cdots \quad f_p(N-1)]$$

for $p = 0, 1$. From (17), for $M = 2$ we get:

$$a(z) = \begin{vmatrix} -1 + r_{00}z^{-1} & r_{10}z^{-1} \\ r_{01}z^{-1} & -1 + r_{11}z^{-1} \end{vmatrix}.$$

Furthermore, we have

$$b_0^i(z) = \begin{vmatrix} s_0^i & r_{10}z^{-1} \\ s_1^i & -1 + r_{11}z^{-1} \end{vmatrix},$$

$$b_1^i(z) = \begin{vmatrix} -1 + r_{00}z^{-1} & s_0^i \\ r_{01}z^{-1} & s_1^i, \end{vmatrix}$$

where

$$s_p^0 = -r_{0p}f_0(0) - r_{1p}f_1(0) \quad \text{and}$$
$$s_p^1 = -r_{0p}f_0(N) - r_{1p}f_1(N), \qquad p = 0, 1.$$

The architectural implementation resulting from (16) is shown in Fig. 5, while for the case where the periodicity property is satisfied, the architecture associated to (18) is depicted on Fig. 6. We observe that the IIR architecture consists of a feedback structure with $M = 2$ delay elements. The parameters $d_i, i = 1, 2$ and $n_{ij}, i = 0, 1, j = 0, 1, 2, 3$ are given by the expressions shown in (19), at the bottom of the page.

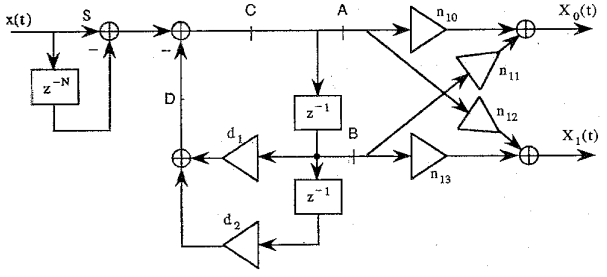


Fig. 6. IIR architecture for $M = 2$ if the periodicity property is satisfied.

D. IIR Architecture Based on Difference Equation Property

An alternative approach to the problem of designing the IIR architecture is based on the defining equation of $X_p(t)$ (3) and the difference equation property of the kernel group introduced in Subsection III-F. In more concrete terms, we can compute the $\mathcal{Z}$ transform of a kernel function $f_p(n)$ based on the difference equation (13) and then calculate the transfer function of the system specified by (3). The following lemmas describe how we can obtain the desired transfer function if we are specified the difference equation parameters. The special case of a difference equation of order $M = 2$ is first

$$\begin{array}{lll} d_1 = -r_{00} - r_{11} & n_{00} = f_0(N)r_{00} + f_1(N)r_{10} & n_{10} = f_0(0)r_{00} + f_1(0)r_{10} \\ d_2 = r_{00}r_{11} - r_{01}r_{10} & n_{01} = -f_0(N)d_2 & n_{11} = -f_0(0)d_2 \\ & n_{02} = f_0(N)r_{01} + f_1(N)r_{11} & n_{12} = f_0(0)r_{01} + f_1(0)r_{11} \\ & n_{03} = -f_1(N)d_2 & n_{13} = -f_1(0)d_2. \end{array} \quad (19)$$

considered, the reason being both its importance for a number of practical applications [23] and its simplicity.

Lemma 4.4: Let the kernel function $f_p(\cdot)$ satisfy the second order difference equation

$$f_p(n) = \gamma_1 f_p(n-1) + \gamma_2 f_p(n-2), \quad n = 1, 2, \cdots, N. \tag{20}$$

The transfer function $H_p(z)$ of the system specified in (3) is

$$H_p(z) = \frac{f_p(N-1) + \frac{1}{\gamma_2} f_p(N) z^{-1}}{1 - \frac{\gamma_1}{\gamma_2} z^{-1} - \frac{1}{\gamma_2} z^{-2}} - z^{-N} \frac{f_p(-1) + \frac{1}{\gamma_2} f_p(0) z^{-1}}{1 - \frac{\gamma_1}{\gamma_2} z^{-1} - \frac{1}{\gamma_2} z^{-2}}. \tag{21}$$

A variation of this lemma was originally given in [23]. In the Appendix, we present a proof that enables the generalization considered in Lemma 4.5.

The parameter values of the associated IIR architecture in Fig. 5 is a direct outcome of Lemma 4.4:

$$\begin{aligned} d_1 &= -\gamma_1/\gamma_2 & n_{00} &= f_0(N-1) & n_{10} &= f_0(-1) \\ d_2 &= -1/\gamma_2 & n_{01} &= f_0(N)/\gamma_2 & n_{11} &= f_0(0)/\gamma_2 \\ & & n_{02} &= f_1(N-1) & n_{12} &= f_1(-1) \\ & & n_{03} &= f_1(N)/\gamma_2 & n_{13} &= -f_1(0)/\gamma_2. \end{aligned} \tag{22}$$

The generalization of Lemma 4.4 for arbitrary values of the order M of the difference equation follows:

Lemma 4.5: Let the kernel function $f_p(\cdot)$ satisfy the Mth-order difference equation (13). Then, the transfer function $H_p(z)$ of the system specified in (3) is given by the expression in (16), where

$$\begin{aligned} a(z) &= 1 + \sum_{n=0}^{M-1} \frac{\gamma_{M-n}}{\gamma_M} z^{-n} - \frac{1}{\gamma_M} z^{-M}, \\ b_p^0(z) &= \sum_{n=0}^{M-1} \left[\frac{1}{\gamma_n} \sum_{q=M-n}^{M} \gamma_q f_p(N+M-n-q-1) \right] \cdot z^{-n} \quad \text{and} \\ b_p^1(z) &= \sum_{n=0}^{M-1} \left[\frac{1}{\gamma_n} \sum_{q=M-n}^{M} \gamma_q f_p(M-n-q-1) \right] z^{-n}. \end{aligned} \tag{23}$$

Lemma 4.5 gives a means for computing the IIR parameter values that is considerably easier from the alternative way of carrying out the algebraic computations involved in (16). Finally, as a direct consequence of Lemma 4.5 we have the following corollary.

Corollary 3: Let the kernel function $f_p(\cdot)$ satisfy

1) The Mth-order difference equation (13).
2) The condition

$$\frac{f_p(N)}{f_p(0)} = \frac{f_p(N-1)}{f_p(-1)} = \cdots = \frac{f_p(N-M+1)}{f_p(-M+1)} = S \tag{24}$$

for some constant S.

Then, the transfer function $H_p(z)$ of the system specified in (3) is given by (18), where $a(z)$ and $b_p^1(z)$ are specified in (23) and S in (24).

We may observe that (24) has the same effect on the IIR architecture with (15), the defining equation of the periodicity property for a kernel group. This fact suggests the following extension of the definition of the periodicity property:

Definition: We shall say that a kernel function $\phi(\cdot)$ satisfies the periodicity property (PP) if there is a positive integer M and a nonzero constant S such that

$$\frac{\phi(N)}{\phi(0)} = \frac{\phi(N-1)}{\phi(-1)} = \cdots = \frac{\phi(N-M+1)}{\phi(-M+1)} = S$$

is satisfied.

Interestingly, (15) and (24) imply the following corollary.

Corollary 4: If a kernel group satisfies the periodicity property, then the ratio value S in (15) will be either $S = 1$ *or* $S = -1$.

E. IIR Architecture Design for Mapping Operators

Thus far, we have discussed the procedure for computing the transfer function that is associated to a given kernel group. We have shown how this transfer function is determined from two different starting points: the matrix difference equation (2) and the scalar difference equation (13). In the sequel, we will consider the implementation of the associated mapping operator, which is the goal of our construction. As a direct consequence of (7), the desired transfer function $H(z)$ is

$$H(z) = \sum_{p=0}^{M-1} c_p H_p(z),$$

where $H_p(z), p = 0, 1, \cdots, M-1$ are the transfer functions of the members of the associated kernel group and $c_p, p = 0, 1, \cdots, M-1$ are specified by the algorithm design procedure. Based on Lemmas 4.3 and 4.4, one can show that

$$H(z) = \frac{1}{a(z)} \sum_{p=0}^{M-1} c_p b_p^0(z) - z^{-N} \frac{1}{a(z)} \sum_{p=0}^{M-1} c_p b_p^1(z) \tag{25}$$

where the expressions of $a(z), b_p^0(z)$, and $b_p^1(z)$ are described by Lemma 4.3 or by Lemma 4.5, depending on the specifications we are given. In a similar way, based on Corollaries 2 and 3, one can show that for the case where the associated kernel group satisfies the periodicity property the transfer function we were after is:

$$H(z) = (S - z^{-N}) \frac{1}{a(z)} \sum_{p=0}^{M-1} c_p b_p^1(z) \tag{26}$$

where the expressions of $a(z)$ and $b_p^1(z)$ are specified as above.

We conclude our discussion on IIR architectural implementations with some comments on the implementation cost.[4] For the denominator $a(z)$ in (25) we need M multipliers and M adders. For the two numerators of this expression we need $2M$ multipliers and $2(M-1)$ adders. An additional adder is needed for the addition in (25). If the periodicity property is satisfied, the implementation of the numerator in (26) requires M multipliers and $M-1$ adders. Note that no multiplier is needed for the factor S, since the constant S takes values in $\{1,-1\}$. The overall cost is shown in Table I (IIR architecture). A comparison of the lattice and the IIR architectures on the basis of the costs in Table I will yield the following conclusion: The IIR architecture is better if the periodicity property is satisfied by the underlying kernel group, while the lattice architecture is appropriate for the cases where the above property is not satisfied. Note that the implicit assumption we have made is that only one kernel function from the associated kernel group participated in the linear expression that specifies the mapping operator in consideration (cf. (6)).

V. Implementing Sliding and Block Transforms

An $N \times N$ data transform can be viewed as a bank of N mapping operators of length N. A time-recursive implementation of these operators yields a locally interconnected, modular, regular, and scalable with N design with linear cost $O(N)$ (in terms of operator counts). In particular, the constant term underlying the asymptotic cost expression can be made linear in terms of the associated kernel group size M, as manifested by the figures in Table I, resulting in the more accurate expression of $O(MN)$. In the introductory Section I, we distinguished between the *sliding* and the *block transforms*. We observe in Table I that such classification reflects different implementation costs. This is justified as follows.

The output of the operators that implement a block transform are sampled at the time instances $t = 0, N, 2N, \cdots$. Consequently, between two adjacent sampling instances we compute $N-1$ pieces of data that are neglected. The only purpose of this computation is to have a transition phase to computing the data output at the next time instance that is a multiple of N. Consider now the computation of the first valid output that is at time instant $t = N$. The scenario for producing this output amounts to initializing the memory elements of the time-recursive structure at $t = 0$ and feeding the N first input samples. If we reset (to 0) the memory elements periodically with period N, we can periodically imitate the computation of the initialization phase, while being able to produce all the useful output data. The consequence of this observation is a simplification of the time-recursive design for the operators in block transforms: the delay element z^{-N} will never deliver a nonzero quantity and therefore it should be replaced by 0 in (25) and (26) (as well as in (16), (18), (21), and (23)). The architecture designs need to be changed accordingly. For example, both IIR structures in Fig. 5 and 6 reduce to the one in Fig. 8.

[4]The IIR structure we consider throughout this paper is the well known type-1 realization and the cost analysis that follows is based on this fact. Nevertheless, any one of the known filter realizations can be used for implementing the transfer functions we specify in this subsection.

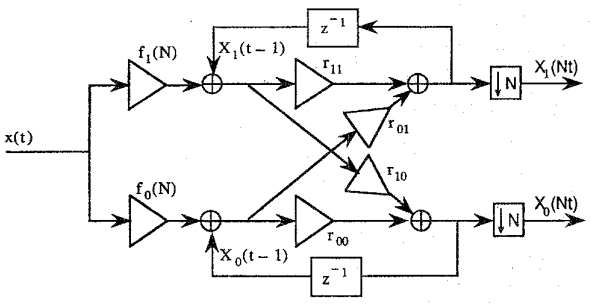


Fig. 7. Lattice architecture for $M = 2$ for an operator used in block transform.

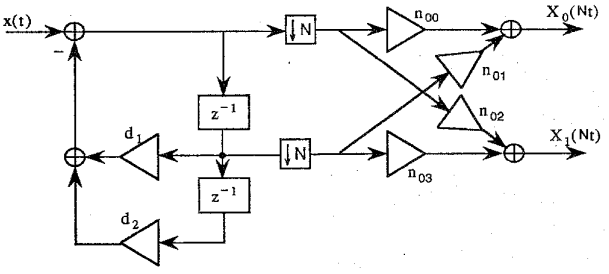


Fig. 8. IIR architecture for $M = 2$ for an operator used in block transform.

Similarly, the lattice structure in Fig. 2 reduces to the one in Fig. 7. A specific instance of this class of circuits, namely the DFT IIR structure, is the well known Goertzel filter [14], [2], [3].

Observe that the periodicity property has an interesting interpretation in this context: if the mapping operators that implement a data transform satisfy PP, the implementation cost of the block transform is almost identical (it differs by one adder) to the one of the sliding transform.

Note also that the decimation in Fig. 8 lets a substantial part of the circuitry operate at minimum rate (that is N times lower than the input data rate).

Finally, an important consequence of the periodical reseting we mentioned above is the elimination of the accumulated round-off error. In this way, limit cycles and other problems of numerical nature associated with the use of finite wordlength in the recursive structure are avoided [23], [7]. The algorithm design procedure suggested in Sections III-C and III-E, along with the cost figures in Table I, can be used as design guides. Based on this background, a time-recursive architecture of a given mapping operator can be routinely obtained. An example of this design procedure has been presented in [12] regarding the modulated lapped transform (MLT) [24], [26] and an extended lapped transform (ELT) [25], [26] with N basis vectors of length $4N$ each. Table II depicts cost metrics associated with time-recursive and feed-forward architectures for sliding and block MLT and ELT. We denote with u the time unit that is equal to the time that lapses between two consequent input samples. All operations acting on a single

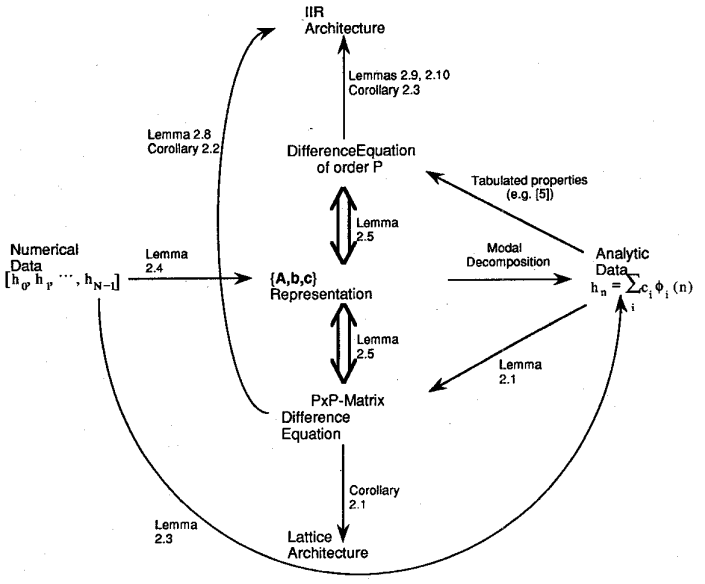


Fig. 9. Overview of the time-recursive architecture design principles.

input sample have to be performed in time no longer than u, or else the data processing rate cannot match the input data rate. This is instructive for the delays[5] allowed to the different operators, as shown in Table II, under the column labeled "basic pipeline rate." In case these conditions are not met, look ahead techniques [32], [33] should be used to improve the speed with the trade-off of complexity. An instance of this trade-off is discussed in [7]. In Table II, operator counts are also specified. All expressions corresponding to feed-forward implementations are based on the fast algorithms derived by Malvar [24]–[26]. The VLSI layout of the operators that can be used for implementing the two architectural schemes have to be taken into account before one is able to choose between a time-recursive and a feed-forward realization. Nevertheless, comparison tables such as Table II can be used as guides for the choice of VLSI operator instantiations, in the perspective of the trade-off between the two architectural schemes. Note that unlike feed-forward fast transform implementations, in time-recursive architectures the locality property allows not only bit-serial implementations, but also bit-parallel ones. Furthermore, the throughput of the overall implementation can be enhanced with the use of pipelining. More detailed information on operator instantiations that have been proven useful in time-recursive architectures and comparisons with competent techniques at this level can be found in [23], [8], [13], [19], and [5].

TABLE II
COST METRICS FOR THE ARCHITECTURAL IMPLEMENTATION OF BLOCK TRANSFORMS. M, A, AND R DENOTE THE TIME DELAYS ASSOCIATED WITH THE IMPLEMENTATION OF THE MULTIPLIER, THE ADDER, AND THE ROTATION CIRCUIT, RESPECTIVELY

		basic pipeline rate	implementation cost (mult, add, rotation)
MLT	time-rec.,sliding	$\max\{M+A, A+R\}=u$	$2N+3, 3N+3, N-1$
	time-rec.,block	$\max\{M+A, A+R\}=u$	$2N+3, 3N+2, N-1$
	fast algo.,sliding	$M+A=u$	$\frac{N}{2}(\log_2 N+5)$,
	fast algo.,block	$M+A=Nu$	$\frac{3N}{2}\log_2 N+\frac{5N}{2}$, 0
ELT	time-rec.,sliding	$\max\{M+2A, A+R\}=u$	$3N+4, 4N+4, N+2$
	time-rec.,block	$\max\{M+2A, A+R\}=u$	$3N+4, 4N+3, N+2$
	fast algo.,sliding	$M+A=u$	$\frac{N}{2}(\log_2 N+5)$,
	fast algo.,block	$M+A=Nu$	$\frac{3N}{2}\log_2 N+4N$, 0

VI. CONCLUSION

In this paper, a unifying architectural framework for parallel, time-recursive computation is established.

The structure of the realization of a given mapping operator is dictated by the decomposition of the latter with respect to proper basis functions. Three properties of these functions that are instructive for the architecture design are *the shift property (SP), the difference equation property (DEP),* and *the periodicity property (PP).* The design of a *lattice architecture* can be based on SP and the design of an *IIR architecture* can be based on either SP or DEP. PP yields a cost reduction and it should be involved in the decision-making for choosing between the two candidate architectural options. The time-recursive architectures associated to block transforms are simpler from the corresponding ones associated to sliding transforms.

A comprehensive overview of the above results is given in Fig. 9. With the help of this diagram, the most efficient among the time-recursive realizations can be easily determined and compared to competent feed-forward alternatives. For demonstrating this concept, realizations of the MLT and an ELT have been briefly commented.

Application areas for this framework include real-time data compression, adaptive filtering and spectrum analysis. Although focused on architectural implementations, the developments in this work are equally useful for uniprocessor algorithmic implementations of sliding transforms.

[5] We assume there is no further fine-grain pipelining for both time-recursive and feed-forward architectures.

APPENDIX

Proof of Lemma 3.4: We will proceed with the proof by showing that there are algorithms for the following computations:

1) Compute $\{\mathbf{A}, \mathbf{b}\}$ based on the knowledge of $\mathbf{R}$ and $\mathbf{f}(0)$.
2) Compute $\{\mathbf{R}, \mathbf{f}(0)\}$ based on $\{\mathbf{A}, \mathbf{b}\}$.
3) Compute $\{\mathbf{A}, \mathbf{b}\}$ based on $\{\mathbf{f}(-1), \mathbf{f}(-2), \cdots, \mathbf{f}(-M), \gamma_1, \gamma_2, \cdots, \gamma_M\}$.
4) Compute $\{\mathbf{f}(-1), \mathbf{f}(-2), \cdots, \mathbf{f}(-M), \gamma_1, \gamma_2, \cdots, \gamma_M\}$ based on $\{\mathbf{A}, \mathbf{b}\}$

The first two algorithms are straightforward implications of relation (10). Note the implicit nonsingularity assumption we have made for the matrix $\mathbf{R}$.

For the computation in Step 3, we follow four steps: first, compute the quantities $\mathbf{f}(n), n = 0, 1, \cdots, M-1$ based on $\mathbf{f}(n), n = -1, -2, \cdots, -M$ and (13). Since we have $\mathbf{f}(n) = \mathbf{A}^N\mathbf{b}$, the controllability matrix specified by the unknown quantities $\{\mathbf{A}, \mathbf{b}\}$ will be [18]

$$\begin{aligned}\mathcal{C} &= [\mathbf{b} \quad \mathbf{Ab} \quad \cdots \quad \mathbf{A}^{M-1}\mathbf{b}] \\ &= [\mathbf{f}(0) \quad \mathbf{f}(1) \quad \cdots \quad \mathbf{f}(M-1)].\end{aligned}$$

Second, by using relation (12), find the controller canonical-form system matrix $\mathbf{A}_c$ and output vector $\mathbf{b}_c$. So, the controllability matrix of the controller canonical form is obtained:

$$\mathcal{C}_c = [\mathbf{b}_c \quad \mathbf{A}_c\mathbf{b}_c \quad \cdots \quad \mathbf{A}_c^{M-1}\mathbf{b}_c].$$

Third, compute the matrix $\mathbf{T}$ that defines the similarity transform

$$\{\mathbf{A}_c, \mathbf{b}_c\} \rightarrow \{\mathbf{A} = \boldsymbol{T}^{-1}\mathbf{A}_c\mathbf{T}, \mathbf{b} = \mathbf{T}^{-1}\mathbf{b}_c\} \tag{27}$$

by using the relation [18]

$$\mathbf{T} = \mathcal{C}_c\mathcal{C}^{-1}.$$

Fourth, the quantities $\{\mathbf{A}, \mathbf{b}\}$ are computed by the relations specified in (27).

The computation in Step 4 is as follows: from the knowledge of $\{\mathbf{A}, \mathbf{b}\}$, we obtain the corresponding pair in controller canonical form $\{\mathbf{A}_c, \mathbf{b}_c\}$ [18]. The desired coefficients $\gamma_1, \gamma_2, \cdots, \gamma_M$ can be obtained from the elements of the first row of the matrix $\mathbf{A}_c$ by using (12). The initial values $\mathbf{f}(-1), \mathbf{f}(-2), \cdots, \mathbf{f}(-M)$ can be obtained by simply evaluating the expression $\mathbf{f}(n) = \mathbf{A}^n\mathbf{b}$ for $n = -1, -2, \cdots, -M$.

Proof of Lemma 4.1: We will consider here the special case of $M = 3$. The proof can be easily generalized for arbitrary values of M.

One can verify that the transfer functions from the input to the points $\hat{X}_0(t), \hat{X}_1(t)$, and $\hat{X}_2(t)$ in Fig. 3, respectively, are

$$-f_0(0)z^{-N} + f_0(N), \quad -f_1(0)z^{-N} + f_1(N) \quad \text{and} \quad -f_2(0)z^{-N} + f_2(N).$$

Consequently, from the $\mathcal{Z}$ transform of (14) we get

$$\begin{aligned}\hat{X}_p(z) &= D_p\hat{X}_0(z) \quad \text{or} \quad -f_p(0)z^{-N} + f_p(N) \\ &= D_p[-f_0(0)z^{-N} + f_0(N)], \qquad p = 1, 2.\end{aligned}$$

Since this is true for every z in some open interval, the latter implies

$$\begin{bmatrix} f_p(0) \\ f_p(N) \end{bmatrix} = D_p \begin{bmatrix} f_0(0) \\ f_0(N) \end{bmatrix}$$

for $p = 1, 2$, or equivalently

$$\frac{f_p(0)}{f_0(0)} = \frac{f_p(N)}{f_0(N)} \quad \text{or} \quad \frac{f_0(N)}{f_0(0)} = \frac{f_p(N)}{f_p(0)}, \qquad p = 1, 2$$

which in turn is equivalent to (15).

Proof of Lemma 4.2: If we have $b = e^{j(k\pi/N)}$ one can verify that (15) holds with ratio value $1/S = (-1)^k$, by simply substituting the above expression of b in (5).

On the other hand, suppose that (15) is satisfied by a kernel group specified by (5) with $b = e^{j\beta}$. If $1/S$ is the value of the ratio in (15), then the latter implies:

$$c_{0p}e^{j\beta N} + c_{1p}e^{-j\beta N} = \frac{1}{S}(c_{0p} + c_{1p}), \quad p = 0, 1.$$

The left-hand side expression can also be written as

$$\begin{aligned}&c_{0p}(\cos\beta N + j\sin\beta N) + c_{1p}(\cos\beta N - j\sin\beta N) \\ &\quad = \cos\beta N(c_{0p} + c_{1p}) + j\sin\beta N(c_{0p} - c_{1p})\end{aligned}$$

where $p = 0, 1$. Therefore, we have either $c_{0p} = c_{1p}, p = 0, 1$ or $\beta = j(k\pi/N)$. Since the first condition yields $c_{00}c_{11} - c_{01}c_{10} = 0$, the alternative must be true. In turn, the above result implies

$$\frac{1}{S} = \cos\beta N = \cos k\pi = (-1)^k.$$

Proof of Lemma 4.3: Let $X_p(t), t = 0, 1, \cdots$ be the output data of the mapping operation defined by the operator

$$[f_p(0) \quad f_p(1) \quad \cdots \quad f_p(N-1)].$$

From (4) we get

$$\begin{aligned}X_p(t) = \sum_{q=0}^{M-1} r_{qp}[X_q(t-1) + \hat{X}_q(t)], \\ p = 0, 1, \cdots, M-1, t = 1, 2, \cdots\end{aligned} \tag{28}$$

where

$$\begin{aligned}\hat{X}_q(t) = -f_q(0)x(t-N) + f_q(N)x(t), \\ q = 0, 1, \cdots, M-1.\end{aligned} \tag{29}$$

Consider the unilateral $\mathcal{Z}_+$ transform, defined as

$$X(z) = Z_+\{x(t)\} = \sum_{t=0}^{\infty} x(t)z^{-t}.$$

$$F(z) = \frac{f_p(0) + \gamma_2 f_p(-1)z^{-1} - z^{-N}[f_p(N) + \gamma_2 f_p(N-1)z^{-1}]}{1 - \gamma_1 z^{-1} - \gamma_2 z^{-2}} \tag{36}$$

$$F(z) = \frac{\sum_{q=1}^{M} \gamma_q \left[\sum_{n=1}^{q} f_p(-n) z^{-q+n} - z^{-N} \sum_{n=1}^{q} f_p(N-n) z^{-N-q+n} \right]}{1 - \sum_{q=1}^{M} \gamma_q z^{-k}}$$

Since

$$Z_+\{x(t-m)\} = z^{-m}X(z), \quad \text{for every integer } m > 0 \tag{30}$$

the $\mathcal{Z}_+$ transform of (28) and (29) gives

$$X_p(z) = \sum_{q=0}^{M-1} r_{qp}[z^{-1}X_q(z) + \hat{X}_q(z)], \quad p = 0, 1, \cdots, M-1 \tag{31}$$

where

$$\hat{X}_q(z) = [-f_q(0)z^{-N} + f_q(N)]X(z), \quad q = 0, 1, \cdots, M-1. \tag{32}$$

From (31), we have

$$\begin{aligned} &\sum_{q=0, q\neq p}^{M-1} r_{qp} z^{-1} X_q(z) + (-1 + r_{pp}z^{-1})X_p(z) \\ &\quad = -\sum_{q=0}^{M-1} r_{qp}\hat{X}_q(z) \\ &\quad = -X(z) \sum_{q=0}^{M-1} r_{qp}[-f_q(0)z^{-N} + f_q(N)], \\ &\quad p = 0, 1, \cdots, M-1. \end{aligned}$$

By solving the above system of equations for $X_p(z), p = 0, 1, \cdots, M-1$, we obtain

$$X_p(z) = H_p(z)X(z), \qquad p = 0, 1, \cdots, M-1$$

where $H_p(z)$ can be brought into the form specified in Lemma 4.3 after a few algebraic manipulations.

Proof of Lemma 4.5: First, we define the $\mathcal{Z}_N$ transform of a discrete time function $f(n)$ over the time segment $\{0, \cdots, N-1\}$

$$\mathcal{Z}_N\{f(n)\} = \sum_{n=0}^{N-1} f(n)z^{-n}. \tag{33}$$

This variation of the $\mathcal{Z}$ transform is appropriate for the frequency domain representation of the kernel functions we consider here, since these functions are defined on a bounded segment of the time axis. On the other hand, we will use the unilateral $\mathcal{Z}_+$ transform as the frequency domain representation of the input signal $x(t)$ and the output signal $X(t)$, since these signals are defined on the semi-infinite sequence of time instances $t = 0, 1, \cdots$.

Let $F(z) = \mathcal{Z}_N\{f_p(n)\}$. Based on (33), we can show that

$$\begin{aligned} &\mathcal{Z}\{f_p(n-1)\} = z^{-1}F(z) + f_p(-1) - z^{-N}f_p(N-1) \quad \text{and} \\ &\mathcal{Z}\{f_p(n-2)\} = z^{-2}F(z) + f_p(-2) + z^{-1}f_p(-1) \\ &\qquad - z^{-N}f_p(N-2) - z^{N-1}f_p(N-1). \end{aligned} \tag{34}$$

In addition, we have

$$\tilde{F}(z) = z^{-N+1}F(z^{-1}) \tag{35}$$

where

$$\tilde{F}(z) = \mathcal{Z}_N\{\tilde{f}_p(n)\} \quad \text{and}$$
$$\tilde{f}(n) = f_p(N-1-n), \qquad n = 0, 1, \cdots, N-1.$$

By taking the $\mathcal{Z}_n$ transform of both sides of (20), using (34), and solving for $F(z)$, we obtain (36), as given at the top of the page.

From (3), we have

$$X(t+N-1) = \sum_{n=0}^{N-1} f_p(n)x(t+n)$$

or equivalently

$$y(t) = \sum_{n=0}^{N-1} x(t-n)\tilde{f}(n) \tag{37}$$

where $y(t) = X(t+N-1)$. By taking the $\mathcal{Z}_+$ transform of both sides of (37) and using (30) we obtain:

$$\begin{aligned} Y(z) &= \sum_{n=0}^{N-1} \tilde{f}(n)[z^{-n}X(z)] = X(z)\sum_{n=0}^{N-1} \tilde{f}(n)z^{-n} \\ &= X(z)\tilde{F}(z). \end{aligned}$$

By substituting (35), we get

$$Y(z) = z^{-N+1}F(z^{-1})X(z)$$

and therefore, the transfer function we were after is

$$H(z) = z^{-N+1}F(z^{-1}). \tag{38}$$

If we substitute the expression (36) of $F(z)$ in the above we obtain the transfer function specified in (21).

Proof of Lemma 4.5: One can verify that

$$\mathcal{Z}_N\{f_p(n-q)\} = z^{-q}F(z) + \sum_{n=1}^{q} \cdot [f_p(-n)z^{-q+n}f_p(N-n)z^{-N-q+n}] \tag{39}$$

where the $\mathcal{Z}_N$ transform is defined by (33) and $F(z) = \mathcal{Z}_N\{f_p(n)\}$. By taking the $\mathcal{Z}_N$ transform of (13), using (39) and solving for $F(z)$ we obtain the second equation shown at the top of the previous page.

By substituting this expression in (38), we obtain (23).

REFERENCES

[1] L. A. Anderson, H. C. Yau, and M. T. Manry, "Recursive approximation of the energy spectral density," *IEEE Trans. Signal Processing,* vol. 40, no. 12, pp. 3059–3062, Dec. 1992.

[2] J. A. Beraldin, T. Aboulnasr, and W. Steenhart, "Efficient one-dimensional systolic array realization of the discrete Fourier transform," *IEEE Trans. Circuits Syst.,* vol. 36, pp. 95–100, 1989.

[3] J. A. Beraldin and W. Steenhart, "Overflow analysis of a fixed-point implementation of the Goertzel algorithm," *IEEE Trans. Circuits Syst.,* vol. 36, pp. 322–324, 1989.

[4] R. R. Bitmead and B. D. O. Anderson, "Adaptive frequency sampling filters," *IEEE Trans. Circuits Syst.,* vol. CAS-28, no. 6, pp. 524–534, June 1981.

[5] J. Canaris, "A VLSI architecture for the real-time computation of discrete trigonometric transforms," *J. VLSI Signal Processing,* vol. 5, no. 1, pp. 95–104, Jan. 1993.

[6] T. S. Chihara, *An Introduction to Orthogonal Polynomials.* New York: Gordon and Breach Science Pub., 1978.

[7] C. T. Chiu, R. K. Kolagolta, K. J. R. Liu, and J. F. Jaja, "VLSI implementation of real-time parallel DCT/DST lattice structures for video communications," in *VLSI Signal Processing, V,* Kung Yao *et al.,* Eds. New York: IEEE Pr., 1992, pp. 101–110.

[8] C. T. Chiu and K. J. R. Liu, "Real-time parallel and fully pipelined two-dimentional DCT lattice structures with application to HDTV systems," *IEEE Trans. Circuits Syst. [Video Technol.],* vol. 2, no. 1, pp. 25–37, Mar. 1992.

[9] G. A. Clark, M. A. Soderstrand, and T. G. Johnson, "Transform domain adaptive filtering using a recursive DFT," in *Proc. IEEE ISCAS,* June 1985, pp. 1113–1116.

[10] E. Frantzeskakis, "An architectural framework for vlsi time-recursive computation with applications," Ph.D. thesis, The University of Maryland at College Park, 1993.

[11] E. Frantzeskakis, J. S. Baras, and K. J. R. Liu, "Time-recursive architectures and wavelet transform," in *Proc. IEEE ICASSP,* 1993, pp. I.445–I.448.

[12] ——, "Time-recursive computation and real-time parallel architectures, with application on the modulated lapped transform," in *Proc. SPIE, Int. Symp. Opt. Appl. Sci. Eng., Advanced Signal Processing Algorithms, Architectures, Implementations,* 1993, pp. 100–111.

[13] E. Frantzeskakis and H. Karathansis, "On computing the 2-D modulated lapped transform in real-time," in *VLSI Signal Processing, VI,*Eggermond *et al.,* Eds. New York: IEEE Pr., 1993, pp. 361–369.

[14] G. Goertzel, "An algorithm for the evaluation of finite trigonometric series," *Am. Math. Monthly,* vol. 65, pp. 34–35, 1958.

[15] R. Hartley and K. Welles, "Resursive computation of the Fourier transform," in *Proc. IEEE ISCAS,* 1990, pp. 1792–1795.

[16] Y. H. Hu, "CORDIC-based VLSI architectures for digital signal processing," *IEEE Signal Processing Mag.,* July 1992, pp. 16–35.

[17] N. Jayant, "Signal compression: Technology targets and research directions," *IEEE J. Select. Areas Commun.,* vol. 10, no. 5, pp. 796–818, June 1992.

[18] T. Kailath, *Linear Systems.* London: Prentice-Hall, 1980.

[19] H. C. Karathanasis, E. Frantzeskakis, I. C. Karathanasis, and A. N. Birbas, "An efficient rotation circut and its applications in VLSI transform processors," in *Proc. Euromicro,* Liverpool, England, 1994.

[20] S. Y. Yung, "Multivariable and multidimentional systems: analysis and design," Ph.D. thesis, Stanford Univ., Stanford, CA, June 1977.

[21] K. J. R. Liu, "Novel parallel architectures for short time Fourier transform," *IEEE Trans. Circuits Syst. II: Analog Digital Signal Processing,* vol. 40, no. 12, pp. 786–790, Dec. 1993.

[22] K. J. R. Liu and C. T. Chiu, "Unified parallel lattice structures for time-recursive discrete cosine/sine/Hartley transforms," *IEEE Trans. Signal Processing,* vol. 41, no. 3, pp. 1357–1377, May 1993.

[23] K. J. R. Liu, C. T. Chiu, R. K. Kolagolta, and J. F. Jaga, "Optimal unified architectures for the real-time computation of time-recursive discrete sinusoidal transforms," *IEEE Trans. Circuits Syst. Video Technol.,* vol. 4, no. 2, pp. 168–180, Apr. 1994.

[24] H. S. Malvar, "Lapped transforms for efficient transform/subband coding," *IEEE Trans. Acoust., Speech, Signal Processing,* vol. 38, no. 6, pp. 969–978, June 1990.

[25] ——, "Extended lapped transforms: Properties, applications, and fast algorithms," *IEEE Trans. Signal Processing,* vol. 40, no. 11, pp. 2703–2714, Nov. 1992.

[26] ——, *Signal Processing with Lapped Transforms.* Boston: Artech House, 1992.

[27] N. R. Murthy and M. N. S. Swamy, "On the computation of running discrete cosine and sine transforms," *IEEE Trans. Signal Processing,* vol. 40, no. 6, pp. 1430–1437, June 1992.

[28] S. S. Narayan, A. M. Peterson, and M. J. Narasimha, "Transform domain LMS algorithm," *IEEE Trans. Acoust., Speech, Signal Processing,* vol. ASSP-31, no. 3, pp. 609–615, June 1983.

[29] A. V. Oppenheim and R. W. Schafer, *Discrete-Time Signal Processing.* Englewood Cliffs, NJ: Prentice-Hall, 1989.

[30] M. Padmanabhan and K. Martin, "Filter banks for time-recursive implementation of transforms," *IEEE Trans. Circuits Syst. II: Analog and Digital Signal Processing,* vol. 40, no. 1, pp. 41–50, Jan. 1993.

[31] A. Papoulis, *Signal Analysis.* New York: McGraw-Hill, 1977.

[32] K. K. Parhi, "Algorithm transformation techniques for concurrent processors," *Proc. IEEE,* vol. 77, no. 12, pp. 1879–1895, 1989.

[33] K. K. Parhi and D. G. Messerschmitt, "Pipeline interleaving and parallelism in recursive digital filters—Part I: Pipelining using scattered look-ahead and decomposition," *IEEE Trans. Acoust., Speech, Signal Processing,* vol. 37, no. 7, pp. 1099–1117, July 1989.

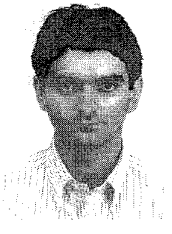
[34] S. G. Smith and S. A. White, "Hardware approaches to vector plane rotation," in *Proc. IEEE ICASSP,* 1988, pp. 2128–2131.

[35] J. Ullman, *Computational Aspect of VLSI.* Rockville, MD: Computer Science Pr., 1984.

[36] P. P. Vaidyanathan, "Multirate filters and filter banks," *Signal Processing.* Englewood Cliffs, NJ: Prentice-Hall, 1993.

[37] P. P. Vaidyanathan and Z. Doganata, "The role of lossless systems in modern digital signal processing: A tutorial," *IEEE Trans. Educ.,* vol. 32, no. 3, pp. 181–197, Aug. 1989.

[38] P. Yip and K. R. Rao, "On the shift property of DCT's and DST's," *IEEE Trans. Acoust., Speech, Signal Processing,* vol. ASSP-35, no. 3, pp. 404–406, Mar. 1987.



Emmanuel Frantzeskakis was born in Athens, Greece, in 1965. He received the B.S. degree in computer engineering and information science from the University of Patras, Greece, in 1988 and the M.S. and Ph.D. degrees in electrical engineering from the University of Maryland at College Park in 1990 and 1993, respectively. Currently, he is with INTRACOM, S.A., Athens, Greece.

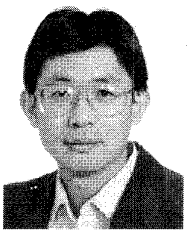
His research interests include VLSI architectures for real-time digital signal processing, HDTV, and adaptive filtering.

Dr. Frantzeskakis is a member of the Technical Chamber of Greece.

John S. Baras (S'73–M'73–SM'83–F'84) received the B.S. in electrical engineering from the National Technical University of Athens, Greece, in 1970 and the M.S. and Ph.D. degrees in applied mathematics from Harvard University, Cambridge, MA, in 1971 and 1973, respectively.

Since 1973, he has been with the Department of Electrical Engineering, University of Maryland at College Park, where he is currently Profesor and member of the Applied Mathematics Faculty. From 1985–1991, he was the Founding Director of the Systems Research Center, now the Institute for Systems Research. In February 1990, he was appointed to the Martin Marietta Chair in Systems Engineering. Since 1991, he has been the Director of the Center for Satellite and Hybrid Communication Networks, a NASA Center for the Commercial Development of Space, which he co-founded. He has numerous publications in control and communication systems, and is the co-editor of *Recent Progress in Stochastic Calculus* (Springer-Verlag, 1990). His current research interests include stochastic systems and signal processing and understanding with emphasis on speech and image signals, real-time architectures, symbolic computation, intelligent control systems, robust nonlinear control, distributed parameter systems, hybrid communication network simulation and management.

Among his awards are a 1978 Naval Research Laboratory Research Publication Award, the 1980 Outstanding Paper Award of the IEEE Control Systems Society, and the 1983 and 1993 Alan Berman Research Publication Award from the Naval Research Laboratory. He is a member of Sigma Xi, the American Mathematical Society, and the Society for Industrial and Applied Mathematics.



K. J. Ray Liu (S'86–M'86–SM'93) received the B.S. degree from the National Taiwan University in 1983 and the Ph.D. degree from the University of California, Los Angeles, in 1990, all in electrical engineering.

Since 1990, he has been with the Electrical Engineering Department and Institute for Systems Research of the University of Maryland at College Park, where he is an Associate Professor. His research interests span all aspects of high performance computational signal processing, including parallel and distributed processing, fast algorithm, VLSI, and concurrent architecture, with application to image/video, radar/sonar, communications, and medical and biomedical technology.

Dr. Liu was the recipient of the National Science Foundation Young Investigator Award in 1994. He received the IEEE Signal Processing Society's 1993 Senior Award and was awarded the George Corcoran Award for outstanding contributions to electrical engineering education at the University of Maryland. Dr. Liu is an Associate Editor of IEEE TRANSACTIONS ON SIGNAL PROCESSING and is a member of VLSI Signal Processing Technical Committee of the IEEE Signal Processing Society.