# Time-Recursive Computation and Real-Time Parallel Architectures, Part I: Framework

*by E. Frantzeskakis, J.S. Baras, and K.J.R. Liu*

T.R. 93-17r1

*The Institute for Systems Research is supported by the National Science Foundation Engineering Research Center Program (NSFD CD 8803012), the University of Maryland, Harvard University, and Industry*

# Time-Recursive Computation and Real-Time Parallel Architectures, Part I: Framework[*]

Emmanuel Frantzeskakis John S. Baras[†] and K. J. Ray Liu[‡]

Electrical Engineering Department
Institute of Systems Research
University of Maryland
College Park, MD 20742

**Abstract**

The time-recursive computation has been proved as 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 high degree of parallelism. In this two part paper, we establish an architectural framework for parallel time-recursive computation. In part I, 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, an architecture design procedure is developed in part II [12] that can be used for routinely obtaining the time-recursive architecture of a given linear operator.

SP EDICS:
5.2. Algorithms and Application Mappings
5.1. Architectures and VLSI Hardware

[*]Research supported in part by grant NSFD CDR 8803012 through the Engineering Research Center's Program.
[†]Martin Marietta Chair in Systems Engineering
[‡]Research also supported in part by ONR grant N00014-93-1-0566.

## 1 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 [13] and later explored by other researchers [2, 3], can be used for implementing an $N$-point DFT in cases where only a small subset of the $N$ frequency components is desired [26]. During the last two decades, the running transforms have been used in frequency domain filtering [27] 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 like channel equalization, echo cancellation, adaptive line enhancing and others [4, 25, 32, 9, 24]. 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 rational applies for realizing the sliding transforms that are used in spectrum analysis, the DFT being the most popular among them [27, 4]. A non-sinusoidal transform used in this context was realized in a time-recursive way independently in [1] and [11]. Liu has demonstrated how non-rectangular data windowing can be embodied in the time-recursive implementation of the Short Time Fourier Transform (STFT) and he generalized the time-recursive design for multiple dimensions [18].

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 penalizes 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

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

area $O(N^2)$ [29, pp.216-219]. 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 constraint is satisfied for a real-time application the area minimization becomes the only concern in the design. 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 resulted designs, by virtue of the area optimality property and the communication locality property [8, 19, 20, 5]. Furthermore, the time-recursive architectures are very efficient for separable multi-dimensional data transforms. In particular, the implementation cost is linear in terms of operator counts and the communication requirement remains local. The induction procedure for designing multi-dimensional architectures based on the one-dimensional ones is described in [18, 20], while a detailed example is given in [8].

In this two part paper, 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, we develop a routine that produces a time-recursive architectural implementation for a given operator. We use this routine for introducing a realization for a lossless QMF bank [31], for modifying the realization of the cosine modulated QMF bank design in [30] and for deriving the time-recursive designs of the Modulated Lapped Transform (MLT) [21, 23] (oftentimes referred in the data coding community as Modified DCT - MDCT [15]) and an Extended Lapped Transform (ELT) [22, 23]. These applications demonstrate the potential of the proposed design procedure. At the same time, they introduce the use of time-recursive computation in multirate digital signal processing. Aiming at single chip implementations of the associated computations, they provide new results applicable to transform coding, subband coding and data sampling alteration with an impact to real-time video and audio data compression, adaptive filtering and spectrum analysis.

The rest of this part I is organized as follows. In Section 2, we introduce some terminology. In Section 3, we study the time-recursive algorithmic structures and their properties. In Section 4, we focus on the architectural implementation of time-recursive architectures. In Section 5, we briefly discuss the special features pertinent to block data transforms. We conclude with Section 6. In the

Appendix, we give the proofs of some lemmas that are stated in the course of the paper.

## 2 Preliminaries

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

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

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 $k^{\text{th}}$ frequency component of the $N$-point Discrete Fourier Transform (DFT) is obtained for $h_n = e^{-j\frac{2\pi}{N}kn}$.

We can specify a mapping operator $[h_0\ h_1\ \cdots\ h_{N-1}]$ 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 $f_0(\cdot), f_1(\cdot), \cdots, f_{M-1}(\cdot)$:

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

A *time-recursive implementation* of a mapping operator $[h_n\ h_1\ \cdots\ h_{N-1}]$ 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 $k^{\text{th}}$ frequency component of the $N$-point Discrete Fourier Transform (DFT) can be extracted as follows [27]

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

## 3 Design of Time-Recursive Algorithm

### 3.1 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, the implementation cost not always justifies the time-recursive computation.

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

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

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

*with a specified final condition* $\mathbf{f}(N)$*, 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:** (2) gives:

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

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

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

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:

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

$$= \sum_{n=1}^{N} x(t+n-N+1) \sum_{q=0}^{M-1} r_{pq} f_q(n) = \sum_{q=0}^{M-1} r_{pq} \left( \sum_{n=1}^{N} x(t+n-N+1) f_q(n) \right)$$

and therefore we obtain the algorithm:

$$X_p(t+1) = \sum_{q=0}^{M-1} r_{pq} \left[ X_q(t) - x(t-N+1)f_q(0) + x(t+1)f_q(N) \right], \tag{4}$$

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. (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 $\mathbf{f}(N)$. $\square$

**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).

## 3.2 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:*

1. *The singleton kernel group* $[cb^n]$*, where* $b$ *and* $c$ *are non-zero free parameters.*

2. *The kernel group*

$$\left[c_{00}b^n + c_{01}b^{-n}, c_{10}b^n + c_{11}b^{-n}\right]^T, \tag{5}$$

*where $b$ is a non-zero parameter and the coefficients are free parameters, such that $c_{00}c_{11} - c_{01}c_{10} \neq 0$.*

3. *The kernel group $\left[c_0, c_1 n, \cdots, c_{M-1}n^{M-1}\right]^T$, where the coefficients are non-zero parameters.*

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

$$\begin{aligned}
\mathbf{R}^{(1)} &= \frac{1}{b}, \\
\mathbf{R}^{(2)} &= \begin{bmatrix} c_{00} & c_{01} \\ c_{10} & c_{11} \end{bmatrix} \begin{bmatrix} b^{-1} & 0 \\ 0 & b \end{bmatrix} \begin{bmatrix} c_{00} & c_{01} \\ c_{10} & c_{11} \end{bmatrix}^{-1} \quad \text{and} \\
\mathbf{R}^{(3)} &= [r_{pq}]_{p,q=0,1,\cdots,M-1}, \quad r_{pq} = \begin{cases} \frac{c_p}{c_q} \begin{pmatrix} p \\ q \end{pmatrix} (-1)^{p-q}, & q \leq p \\ 0, & q > p \end{cases} . \quad \square
\end{aligned}$$

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

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

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 $[h_0 \; h_1 \; \cdots \; h_{N-1}]$. 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 1 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.

## 3.3 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 :*

$$h_n = \sum_i c_i \phi_i(n), \tag{6}$$

*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 $\mathbf{f}_i(n) = [1 \ n \ n^2]^T$.*

*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 $X_i(t) = \sum_{n=0}^{N-1} \phi_i(n) x(t + n - N + 1)$.*

2. *Evaluate*

$$X(t) = \sum_i c_i X_i(t). \tag{7}$$

Detailed examples along the lines of this procedure are given in part II.

## 3.4 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 of a close 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* $[h_0\ h_1\ \cdots\ h_{N-1}]$ *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 $[h_0\ h_1\ \cdots\ h_{N-1}]$, 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 [16, 17]. 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

[2]For the definition of the Markov parameters of an LTI system see [16, pp.92-93].

$\{\mathbf{A}, \mathbf{b}, \mathbf{c}\}$. □

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.

## 3.5 Systematic Design II

For the time-recursive implementation of an arbitrary mapping operator $[h_0\ h_1\ \cdots\ h_{N-1}]$ three steps need to be added at the beginning of the design procedure in Subsection 3.3:

**Design Procedure Supplement**

*Input :* *The mapping operator* $[h_0\ h_1\ \cdots\ h_{N-1}]$.

*Step 0.1:* *Compute the quantities* $\mathbf{A}, \mathbf{b}$ *and* $\mathbf{c}$ *in (8) [16, 17].*

*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 close 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 Subsection 3.3.

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 close 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 $2 \times 2$ plain rotation matrices. In Section 4 we will see that both of these features are very desirable for the architectural implementation.

[3]For the definition of similarity transforms and the canonical realization forms for LTI systems one may refer to [16].

## 3.6 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 [16]:

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

where $\alpha_p,\ p = 1, 2, \cdots, M$ are the constants specifying the system matrix $\mathbf{A}$ in the controller canonical form [16]. Equivalently, this can be written in a difference equation format as follows:

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

where

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

Let $\mathbf{e}_p$ be the row vector of length $M$, for which the $p^{\text{th}}$ 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 $p^{\text{th}}$ 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* $f_q(n), n = -1, -2, \cdots, -M$.

**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.

## 4 Design of Time-Recursive Architecture

### 4.1 Lattice Architecture Design for Mapping Operators

In Section 3, 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 [19]. 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 $M$ 2-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 $M(M-1) + 2M = M^2 + M$ 2-input adders. The weighted-sum circuit consists of $M$ multipliers and $M-1$ adders. The cost of the overall implementation is given on Table 1 (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 Subsections 3.3 and 3.5 the matrix $R$ will be block diagonal with blocks consisted of plain rotations. Consequently, we can implement the interconnection network very efficiently, with locally interconnected rotation circuits. The latter can be realized either with CORDIC processors [14] or with distributed arithmetic techniques [28]. The cost for implementing a mapping operator with this approach is shown on Table 1 (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 [30, 31]. The lossless QMF bank implementation presented in part II [11] takes advantage of this fact to reduce the number of multipliers to be implemented.

### 4.2 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 3 2-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 1 (case b.). In Lemma 4.1, which follows, we state a condition on the 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* $\mathbf{f}(\cdot) = [f_0(\cdot)\ f_1(\cdot)\ \cdots\ f_{M-1}(\cdot)]^T$*, 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 non-zero 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 the 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 $\beta = j\frac{k\pi}{N}$, 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 $\mathbf{f}_k(n) = \left[\cos\frac{k\pi}{N}(n + \frac{1}{2})\ \sin\frac{k\pi}{N}(n + \frac{1}{2})\right]^T$.

## 4.3 IIR Architecture Based on Shift Property

The *lattice architecture* we have seen in Subsection 4.1 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) [20]. 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 $\mathbf{f}(\cdot) = [f_0(\cdot)\ f_1(\cdot)\ \cdots\ f_{M-1}(\cdot)]^T$ 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 $H_p(z)$*

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

*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: $a(z) = |\mathbf{A}(z)|$, $b_p^i(z) = \left|\mathbf{B}_p^i(z)\right|$, $i = 0, 1$, where*

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

*$\mathbf{B}_p^i(z)$ is an $M \times M$ matrix formed by substituting the $p^{th}$ column of $\mathbf{A}(z)$ with $\left[s_0^i\ s_1^i\ \cdots\ s_{M-1}^i\right]^T$, $i = 0, 1$, and*

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

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

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

**Corollary 2** *Let $\mathbf{f}(\cdot) = [f_0(\cdot)\ f_1(\cdot)\ \cdots\ f_{M-1}(\cdot)]^T$ 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:*

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

*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

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

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

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

Furthermore, we have

$$b_0^i(z) = \begin{vmatrix} s_0^i & r_{01}z^{-1} \\ s_1^i & -1 + r_{11}z^{-1} \end{vmatrix}, \quad b_1^i(z) = \begin{vmatrix} -1 + r_{00}z^{-1} & s_0^i \\ r_{10}z^{-1} & s_1^i \end{vmatrix},$$

where

$$s_p^0 = -r_{p0}f_0(0) - r_{p1}f_1(0) \quad \text{and} \quad s_p^1 = -r_{p0}f_0(N) - r_{p1}f_1(N), \quad p = 0, 1.$$

The architectural implementation resulted 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 following expressions:

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

## 4.4 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 3.6. 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 considered, the reason being both its importance for a number of practical applications [20] and its simplicity.

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

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

*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 [20]. In 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:

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

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 $M^{th}$ 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*

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

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:

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

1. *The $M^{th}$ 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 non-zero 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:

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

## 4.5 IIR Architecture Design for Mapping Operators

So 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 4.5 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:

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

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 1 (IIR architecture). A comparison of the lattice and the IIR architectures on the basis of the costs in Table 1 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*

[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.

*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)). A decision rule that encounters all the different factors affecting the proper choice of the architecture is provided in part II.

## 5 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 and 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 1, resulting in the more accurate expression of $O(MN)$. In the introductory Section 1, we have distinguished between the *sliding* and the *block transforms.* We observe in Table 1 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 non-zero 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.

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 [13, 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 finally 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).

## 6 Conclusion

In this first part of the two-part 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 rule 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. The algorithm design procedure suggested in Subsections 3.3 and 3.5, along with the cost figures in Table 1 can be used as design guides. Based on this background, an architecture design procedure is developed in part II that can be used for routinely obtaining the time-recursive architecture of a given mapping operator.

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 algorithmic implementations of sliding transforms.

## A 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 non-singularity assumption we have made for the matrix $\mathbf{R}$.

For the computation in 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 [16]

$$\mathcal{C} = \left[\mathbf{b}\ \mathbf{A}\mathbf{b}\ \cdots\ \mathbf{A}^{M-1}\mathbf{b}\right] = [\mathbf{f}(0)\ \mathbf{f}(1)\ \cdots\ \mathbf{f}(M-1)].$$

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:

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

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

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

by using the relation [16]

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

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

The computation in 4. is as follows: From the knowledge of $\{\mathbf{A}, \mathbf{b}\}$, we obtain the corresponding pair in controller canonical form $\{\mathbf{A}_c, \mathbf{b}_c\}$ [16]. 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

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

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

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

which in turn is equivalent to (15).

**Proof of Lemma 4.2:** If we have $b = e^{j\frac{k\pi}{N}}$ 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:

$$c_{p0}e^{j\beta N} + c_{p1}e^{-j\beta N} = \frac{1}{S}(c_{p0} + c_{p1}), \quad p = 0,1.$$

The left hand side expression can also be written as

$$c_{p0}(\cos\beta N + j\sin\beta N) + c_{p1}(\cos\beta N - j\sin\beta N) = \cos\beta N(c_{p0}+c_{p1}) + j\sin\beta N(c_{p0}-c_{p1}),$$

where $p = 0,1$. Therefore we have either $c_{p0} = c_{p1},\ p = 0,1$ or $\beta = j\frac{k\pi}{N}$. 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

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

From (4) we get

$$X_p(t) = \sum_{q=0}^{M-1} r_{pq}\left[X_q(t-1) + \hat{X}_q(t)\right], \quad p = 0,1,\cdots,M-1, \quad t = 1,2,\cdots \tag{28}$$

where

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

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

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

Since

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

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

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

where

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

From (31) we have

$$\sum_{q=0, q\neq p}^{M-1} r_{pq} z^{-1} X_q(z) + (-1 + r_{pp} z^{-1}) X_p(z) = -\sum_{q=0}^{M-1} r_{pq} \hat{X}_q(z)$$

$$= -X(z) \sum_{q=0}^{M-1} r_{pq} \left[-f_q(0) z^{-N} + f_q(N)\right], \quad p = 0, 1, \cdots, M-1.$$

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

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

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

**Proof of Lemma 4.4:** 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

$$\mathcal{Z}\{f_p(n-1)\} = z^{-1} F(z) + f_p(-1) - z^{-N} f_p(N-1) \quad \text{and}$$

$$\mathcal{Z}\{f_p(n-2)\} = z^{-2} F(z) + f_p(-2) + z^{-1} f_p(-1) - z^{-N} f_p(N-2) - z^{N-1} f_p(N-1). \tag{34}$$

Also, we have

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

where

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

By taking the $\mathcal{Z}_N$ transform of both sides of (20), using (34) and solving for $F(z)$, we obtain:

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

From (3) we have

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

or equivalently

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

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

$$Y(z) = \sum_{n=0}^{N-1} \widetilde{f}(n)\left[z^{-n}X(z)\right] = X(z)\sum_{n=0}^{N-1} \widetilde{f}(n)z^{-n} = X(z)\widetilde{F}(z).$$

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

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

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:

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

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 Transactions on Signal Processing*, 40(12):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. on CAS*, 36:95–100, 1989.

[3] J.A. Beraldin and W. Steenhart. Oveflow analysis of a fixed-point implementation of the Goertzel algorithm. *IEEE Trans. on Circuits and Systems*, 36:322–324, 1989.

[4] R.R. Bitmead and B.D.O. Anderson. Adaptive Frequency Sampling Filters. *IEEE Transactions on Circuits and Systems*, 28(6):524–534, June 1981.

[5] J. Canaris. A VLSI Architecture for the Real-Time Computation of Discrete Trigonometric Transforms. *Journal of VLSI Signal Processing*, 5(1):95–104, Jan. 1993.

[6] T.S. Chihara. *An Introduction to Orthogonal Polynomials.* Gordon and Breach Science Pub., New York, 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 Kung Yao et al., editor, *VLSI Signal Processing, V*, pages 101–110. IEEE Press, New York, 1992.

[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 Transactions on Circuits and Systems for Video Technology*, 2(1):25–37, March 1992.

[9] G.A. Clark, M.A. Soderstrand, and T.G. Johnson. Transform Domain Adaptive Filtering Using a Recursive DFT. In *Proc. IEEE ISCAS*, pages 1113–1116, June 1985.

[10] E. Frantzeskakis. *An Architectural Framework for VLSI Time-Recursive Computation with Applications.* PhD 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*, pages I.445–448, 1993.

[12] E. Frantzeskakis, J.S. Baras, and K.J.R. Liu. Time-Recursive Computation, Part II: Methodology and Application on QMF Banks and ELT. *submitted to IEEE Trans. on SP*, July 1993.

[13] G. Goertzel. An algorithm for the evaluation of finite trigonometric series. *Amer. Math. Monthly*, 65:34–35, 1958.

[14] Y.H. Hu. CORDIC-Based VLSI Architectures for Digital Signal Processing. *IEEE Signal Processing Magasine*, pages 16–35, July 1992.

[15] N. Jayant. Signal Compression: Technology Targets and Research Directions. *IEEE Journal on Selected Areas in Communications*, 10(5):796–818, June 1992.

[16] T. Kailath. *Linear Systems.* Prentice Hall, London, 1980.

[17] S.Y. Kung. *Multivariable and Multidimentional Systems: Analysis and Design.* PhD thesis, Stanford University, June 1977.

[18] K.J.R. Liu. Novel Parallel Architectures for Short Time Fourier Transform. *To appear in IEEE Transactions on Circuits and Systems II: Analog and Digital Signal Processing*, 1993.

[19] K.J.R. Liu and C.T. Chiu. Unified Parallel Lattice Structures for Time-Recursive Discrete Cosine/Sine/Hartley Transforms. *IEEE Trans. on SP*, 41(3):1357–1377, May 1993.

[20] K.J.R. Liu, C.T. Chiu, R.K. Kolagolta, and J.F. Jaja. Optimal Unified Architectures for the Real-Time Computation of Time-Recursive Discrete Sinusoidal Transforms. *Submitted to IEEE Transactions on Circuits and Systems for Video Technology*, 1992.

[21] H.S. Malvar. Lapped Transforms for Efficient Transform/Subband Coding. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 38(6):969–978, June 1990.

[22] H.S. Malvar. Extended Lapped Transforms: Properties, Applications, and Fast Algorithms. *IEEE Transactions on Signal Processing*, 40(11):2703–2714, Nov. 1992.

[23] H.S. Malvar. *Signal Processing with Lapped Transforms.* Artech House,Inc., Boston, 1992.

[24] N.R. Murthy and M.N.S. Swamy. On the Computation of Running Discrete Cosine and Sine Transforms. *IEEE Transactions on Signal Processing*, 40(6):1430–1437, June 1992.

[25] S.S. Narayan, A.M. Peterson, and M.J. Narasimha. Transform Domain LMS Algorithm. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, ASSP-31(3):609–615, June 1983.

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

[27] A. Papoulis. *Signal Analysis.* McGraw-Hill,Inc, New York, 1977.

[28] S.G. Smith and S.A. White. Hardware Approaches to Vector Plane Rotation. In *Proc. IEEE ICASSP*, pages 2128–2131, 1988.

[29] J. Ullman. *Computational Aspect of VLSI.* Computer Science Press, Rockville, MD, 1984.

[30] P.P. Vaidyanathan. *Multirate Filters and Filter Banks.* Signal Processing. Prentice Hall, Eglewood Cliffs, NJ, 1993.

[31] P.P. Vaidyanathan and Z. Doganata. The Role of Lossless Systems in Modern Digital Signal Processing: A Tutorial. *IEEE Transactions on Education*, 32(3):181–197, Aug. 1989.

[32] P. Yip and K.R. Rao. On the Shift Property of DCT's and DST's. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, ASSP-35(3):404–406, March 1987.

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

Table 1: 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.

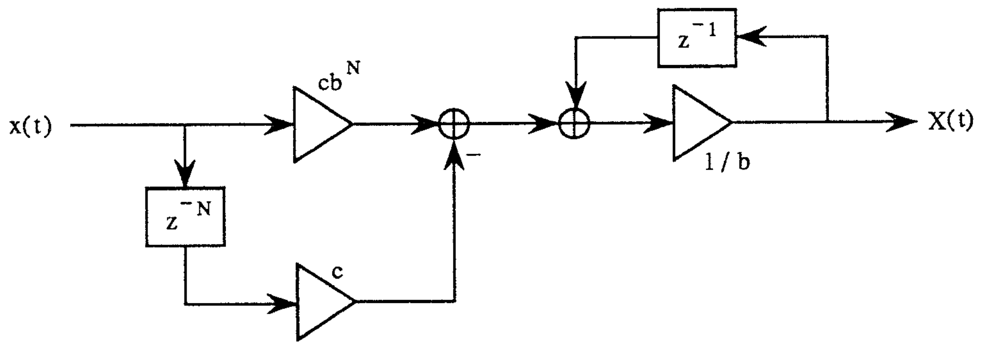

Figure 1: Architecture for kernel group of size $M = 1$.

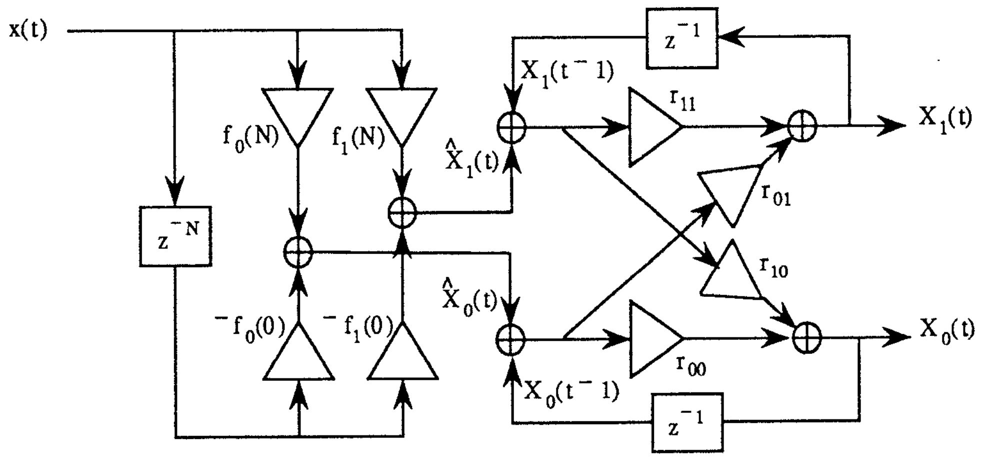

Figure 2: Lattice architecture for kernel group of size $M = 2$.

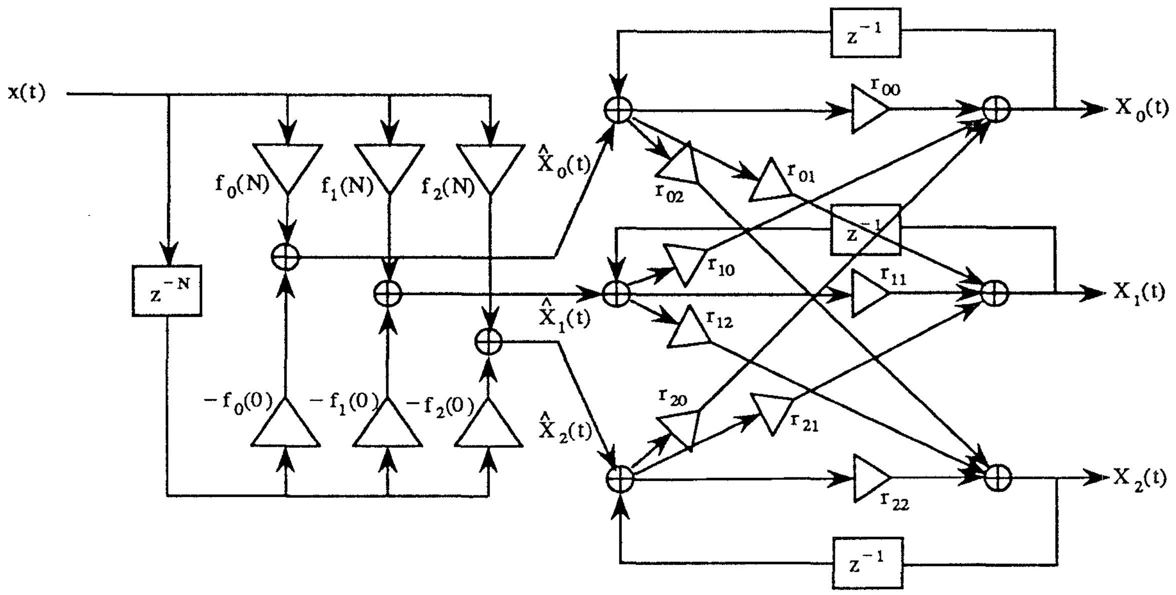

Figure 3: Lattice architecture for kernel group of size $M = 3$.

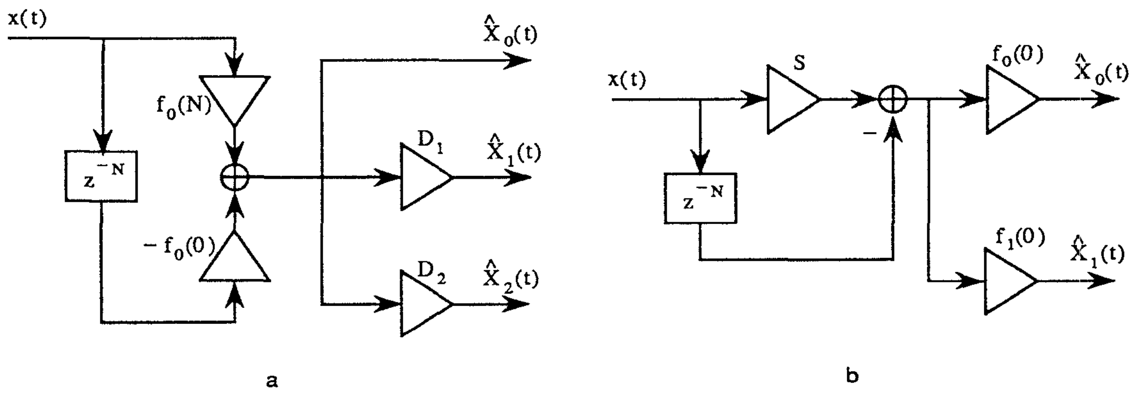

Figure 4: Part of lattice architecture if the periodicity property is satisfied.

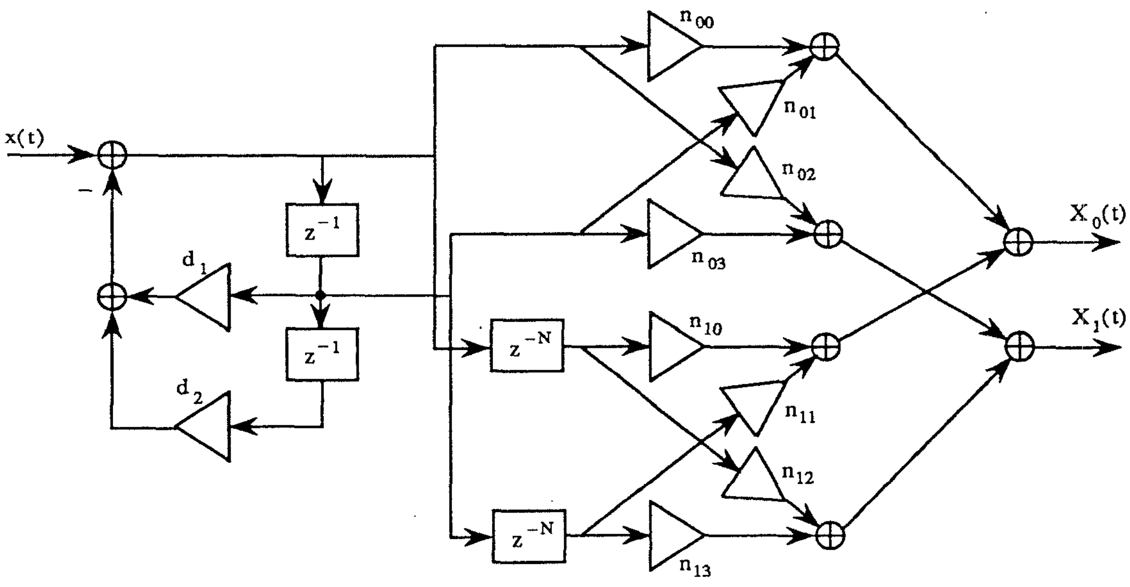

Figure 5: IIR architecture for $M = 2$.

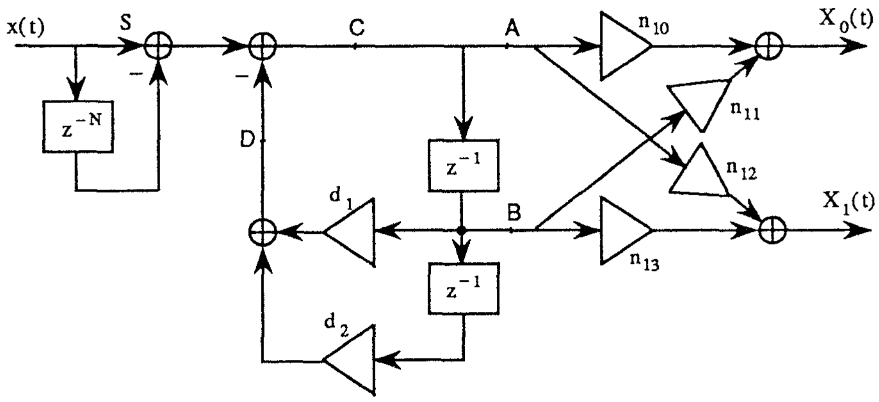

Figure 6: IIR architecture for $M = 2$ if the periodicity property is satisfied.

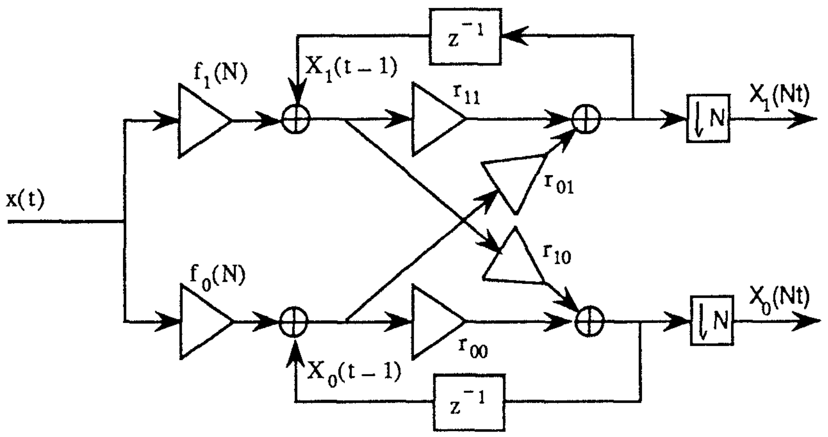

Figure 7: Lattice architecture for $M = 2$ for an operator used in block transform.

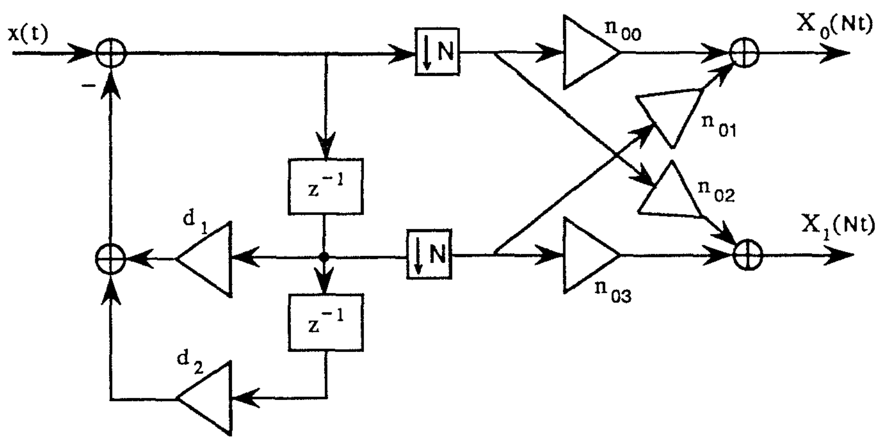

Figure 8: IIR architecture for $M = 2$ for an operator used in block transform.

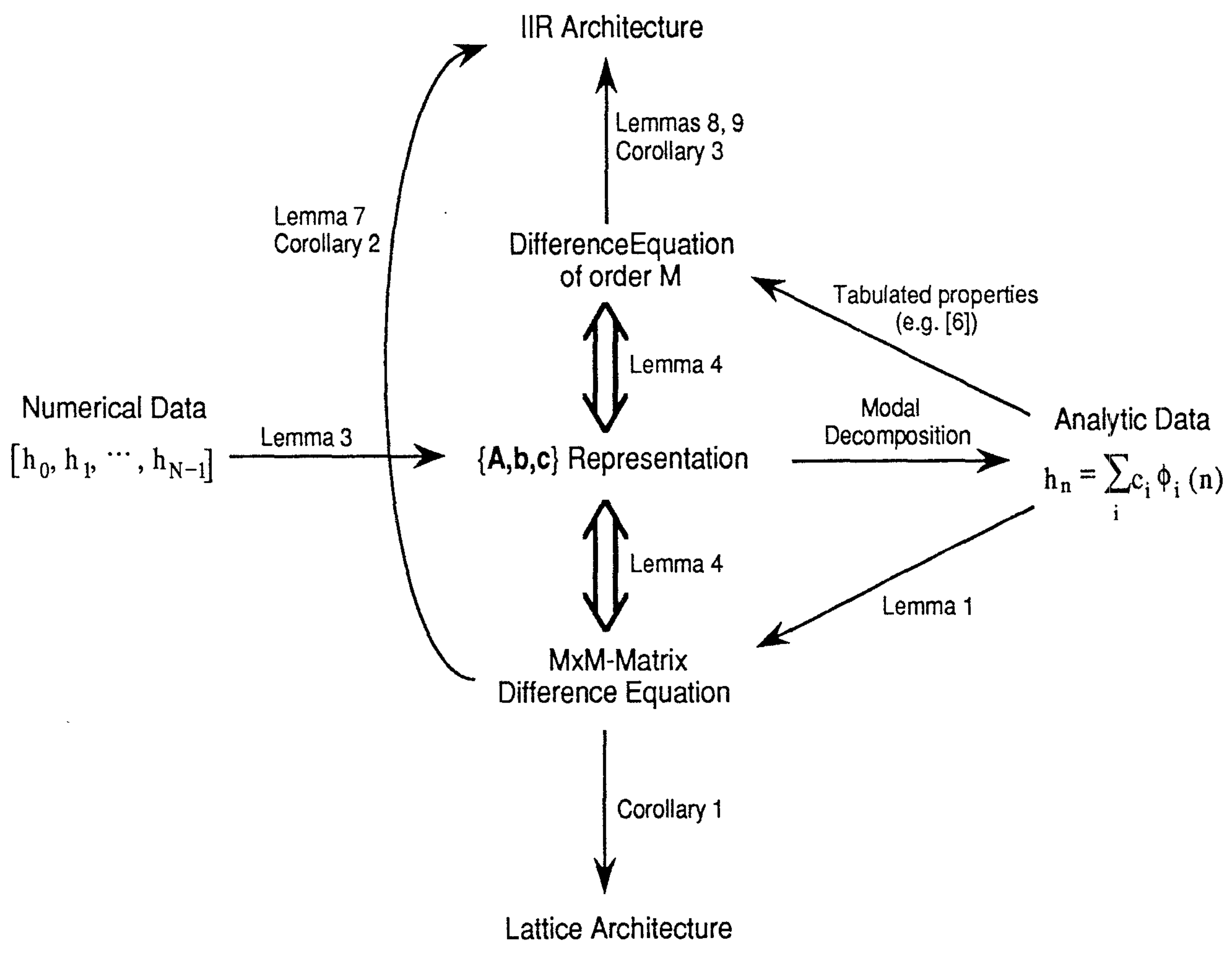

Figure 9: Overview of the time-recursive architecture design principles.