JB 93-12

Entitled:

An Architectural Framework for Parallel Time-Recursive Computation

Authors:

(with Emmaneul Frantzeskakis and K.J. Ray Liu)

4th Conference on Informatics
organized by the Greeck Computer Society

October, 1993

Athens, Greece

An Architectural Framework for Parallel Time-Recursive Computation

Emmanuel Frantzeskakis John S. Baras and K. J. Ray Liu

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

Abstract

The time-recursive computation model has been proven particularly useful for the real-time evaluation of one and two-dimensional block transforms. Unlike the FFT based architectures, time-recursive ones 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 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 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. This routine is instructive for the design of a CAD tool that will facilitate the derivation of time-recursive architectures. The design of architectures for the Discrete Cosine Transform and the Modulated Lapped Transform based on this routine is reported.

1 Introduction

In this paper, we summarize a study towards a unifying methodology for deriving time-recursive realizations with focus on architectural implementations. We consider the implementation of the *mapping operator* $[h_0\ h_1\ \cdots\ h_{N-1}] : x(\cdot) \rightarrow X(\cdot)$, which operates on the semi-infinite sequence of scalar data $x(\cdot)$ and produces the sequence $X(\cdot)$ as follows:

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

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

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

For example, if we have $[h_n = 1, n = 0, 1, \cdots, N-1]$, then $X(t)$ will be the sum of the last N values in the input stream. The recursive algorithmic implementation of this operator will be simply the computation

$$X(t+1) = X(t) + x(t+1) - x(t-N+1).$$

There is a common infrastructure among the mapping operators that are involved in these diverse applications. The unifying feature is a *shift property* we discuss in the following Section, where we establish the framework for formulating time-recursive expressions. In Section 3, we derive a unifying procedure for architectural design. In Section 4, we report two design examples: the Discrete Cosine Transform and the Modulated Lapped Transform [9]. We conclude with Section 5.

2 Time-Recursive Computation: Basic Framework

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 $\mathbf{f}(\cdot)$. Furthermore, we will call *kernel group* a vector of kernel functions

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

Shift Property: *A kernel group* $\mathbf{f}(\cdot)$ *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 specified final condition $\mathbf{f}(N)$*, where* $\mathbf{R}$ *is a constant matrix of size* $M \times M$.

Lemma 2.1 *A recursive implementation of a kernel group* $\mathbf{f}(\cdot)$ *is feasible if this kernel group satisfies the shift property.*

Proof: (2) gives:

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

for $n = 1, 2, \cdots, N,\ p = 0, 1, \cdots, M-1$, where $r_{pq}, p, q = 0, 1, \cdots, M-1$ are the elements of the matrix $\mathbf{R}$. Let

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

for $p = 0, 1, \cdots, M-1$. 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:

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

and therefore we obtain:

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

$p = 0, 1, \cdots, M-1$. If we assume knowledge of the boundary values $\{f_q(0), f_q(N),\ q = 0, 1, \cdots, M-1\}$, (4) specifies the algorithm that performs the update computation we were after. 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 the shift property can be implemented 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 procedure belong in the initialization phase (off-line computation).

The issue of specifying a family of kernel groups that satisfy the shift property is addressed by lemma 2.2:

Lemma 2.2 *The shift property (SP) is satisfied by:*

1. *The singleton kernel group $[cb^n]$, where b and c are non-zero free parameters.*
2. *The kernel group $[c_{00}b^n + c_{01}b^{-n}, c_{10}b^n + c_{11}b^{-n}]^T$, 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 $\left[c_0, c_1 n, \cdots, c_Q n^Q\right]^T$, where Q is an arbitrary positive integer and the coefficients are non-zero parameters.*

The proof of this lemma can be found in [5].

In [5] and [4] we demonstrate how an arbitrary mapping operator can be expressed in an optimal manner as a linear combination of kernel functions that satisfy SP. Here we assume that such an expression can be obtained by inspection (and utilizing Lemma 2.2). For example, the kernel functions of the Short Time Fourier Transform, the Discrete Cosine Transform and the Modulated Lapped Transform [9] belong in this class of kernels. Consequently, we can obtain a time-recursive algorithm for a specified mapping operator as follows:

Design Procedure

Input : *$h_n = \sum_i c_i \phi_i(n)$, where $\{\phi_i(n)\}$ is a set of kernel functions that satisfy the shift property 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 2.2 / 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 $X_i(t)$, defined as $X_i(t) = \sum_{n=0}^{N-1} \phi_i(n)x(t+n-N+1)$.*
2. *Evaluate $X(t) = \sum_i c_i X_i(t)$.*

The kernel group associated to the mapping operator is the union $\bigcup_i \mathbf{f}_i(\cdot)$.

3 Architecture Design

Let us consider the simple case of a mapping operator $h_n = \phi(n)$, where $\phi(n)$ is an element of the size-2 kernel group $\mathbf{f}(n)$ that satisfies SP. (4) dictates an architectural implementation that has the lattice structure in Fig. 1. In an abuse of terminology, we will use the name *lattice architecture* for the architectures that realize (4) regardless the size of the associated kernel group.

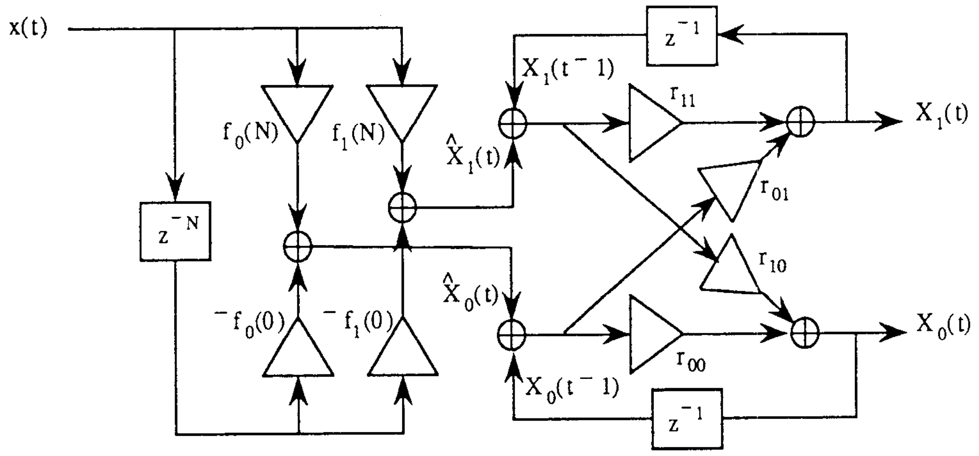


Figure 1: Lattice architecture for kernel group of size $M = 2$.

An alternative time-recursive architecture, obtained by using a transfer function approach, is shown in Fig. 2, where

$$
\begin{array}{lll}
d_1 = -r_{00} - r_{11} & n_{00} = f_0(N)r_{00} + f_1(N)r_{01} & n_{10} = f_0(0)r_{00} + f_1(0)r_{01} \\
d_2 = r_{00}r_{11} - r_{01}r_{10} & n_{01} = -f_0(N)d_2 & n_{11} = -f_0(0)d_2 \\
 & n_{02} = f_0(N)r_{10} + f_1(N)r_{11} & n_{12} = f_0(0)r_{10} + f_1(0)r_{11} \\
 & n_{03} = -f_1(N)d_2 & n_{13} = -f_1(0)d_2 \quad .
\end{array}
\tag{5}
$$

We use the name *IIR architecture* for the architecture derived by computing the transfer function. If a kernel group $\mathbf{f}(n)$ of size $M = 2$ satisfies the difference equation

$$\mathbf{f}(n) = \gamma_1 \mathbf{f}(n-1) + \gamma_2 \mathbf{f}(n-2) \qquad n = 1, 2, \cdots, N \tag{6}$$

for some constant scalars $\gamma_p, p = 1, 2$ and arbitrary initial conditions $\mathbf{f}(n), n = -1, -2$, then the parameters of the IIR implementation are specified by the expressions [5]

$$\begin{array}{lll} d_1 = -\gamma_1/\gamma_2 & n_{00} = f_0(N-1) & n_{10} = f_0(-1) \\ d_2 = -1/\gamma_2 & n_{01} = f_0(N)/\gamma_2 & n_{11} = f_0(0)/\gamma_2 \\ & n_{02} = f_1(N-1) & n_{12} = f_1(-1) \\ & n_{03} = f_1(N)/\gamma_2 & n_{13} = -f_1(0)/\gamma_2 \ . \end{array} \tag{7}$$

The latter are useful in cases where we know in advance that a mapping operator exhibits the property of satisfying such a difference equation (see for example [2]).

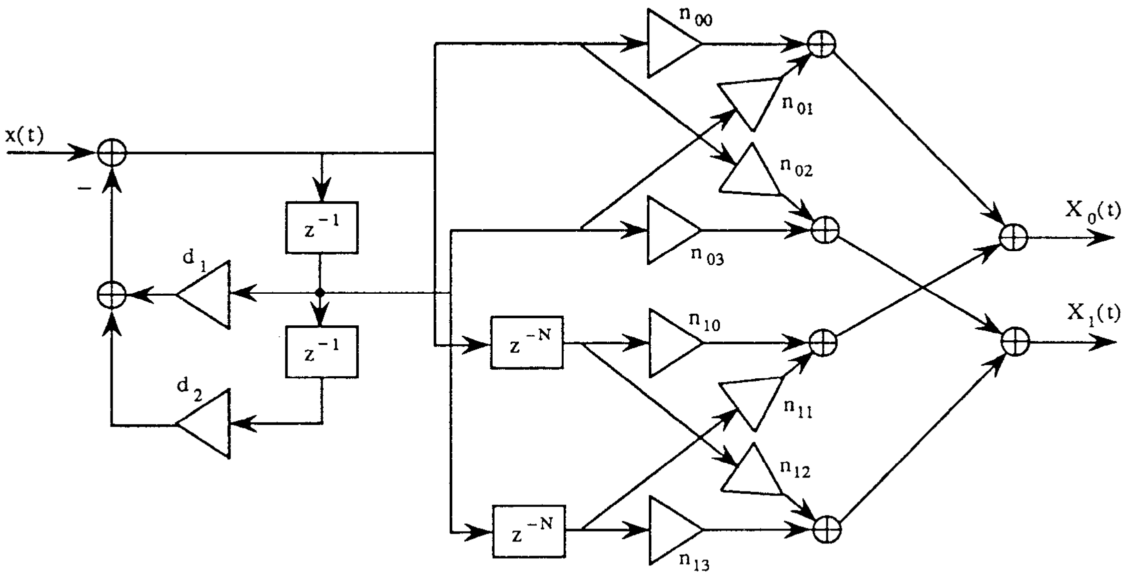


Figure 2: IIR architecture for $M = 2$.

Fig. 1 and Fig. 2 suggest that the IIR architectures imply higher implementation cost than their lattice counterparts. Nevertheless, this is not always true, since the implementation cost depends on the following two factors:

1. The number of the functions in the kernel group that participate in the linear expression of the mapping operator.

2. Whether the *periodicity property*, that is defined next, is satisfied by the kernel group in question.

The role of these factors becomes clearer in the examples in Section 4.

Periodicity Property: *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:*

$$\frac{f_0(N)}{f_0(0)} = \frac{f_1(N)}{f_1(0)} = \cdots = \frac{f_{M-1}(N)}{f_{M-1}(0)} = S \tag{8}$$

for some non-zero constant S.

The name of this property is justified by the following special case: Consider the kernel group specified by statement 2 in lemma 2.2, that is,

$$\begin{bmatrix} f_0(n) \\ f_1(n) \end{bmatrix} = \begin{bmatrix} c_{00}b^n + c_{01}b^{-n} \\ c_{10}b^n + c_{11}b^{-n} \end{bmatrix}, \tag{9}$$

where b is a non-zero parameter and the coefficients are free parameters, such that $c_{00}c_{11} - c_{01}c_{10} \neq 0$. In [5] we prove the following lemma:

Lemma 3.1 *If the parameter b of the kernel group (9) is of the form $b = e^{j\beta}$, then (9) 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 (8) is equal to $S = (-1)^k$.*

An example of kernel group that satisfies PP is $\mathbf{f}_k(n) = \left[\cos\frac{k\pi}{N}(n+\frac{1}{2}) \ \sin\frac{k\pi}{N}(n+\frac{1}{2})\right]^T$. Observe that these two kernel functions specify the Discrete Cosine Transform and the Discrete Sine Transform. Fig. 3 and Fig. 4 show how the architectures in Fig. 1 and Fig. 2 will be modified if the periodicity property holds.

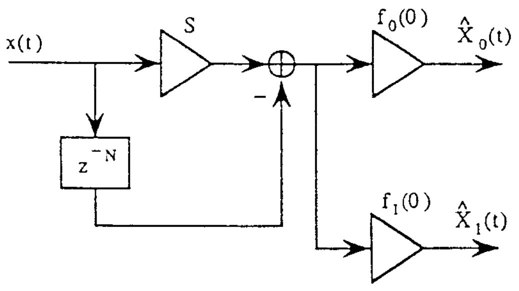


Figure 3: Part of lattice architecture for a size-2 kernel group if the periodicity property is satisfied.

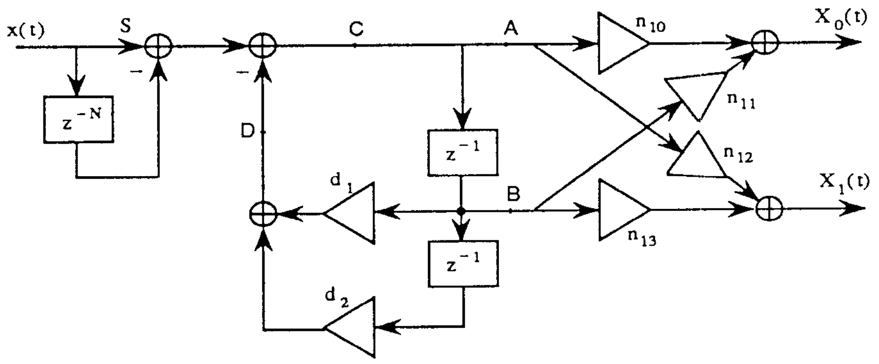


Figure 4: IIR architecture for a size-2 kernel group if the periodicity property is satisfied.

Fig. 5 depicts a flow graph diagram that summarizes the architecture design procedure for a given mapping operator. For the sake of simplicity here we have restricted ourselves to the special case where kernel groups of size $M = 2$ are sufficient for implementing the given mapping operator. Nevertheless, the generalization to arbitrary values values of M is straightforward. This design procedure provides the guidelines for developing a CAD tool appropriate for time-recursive architecture design.

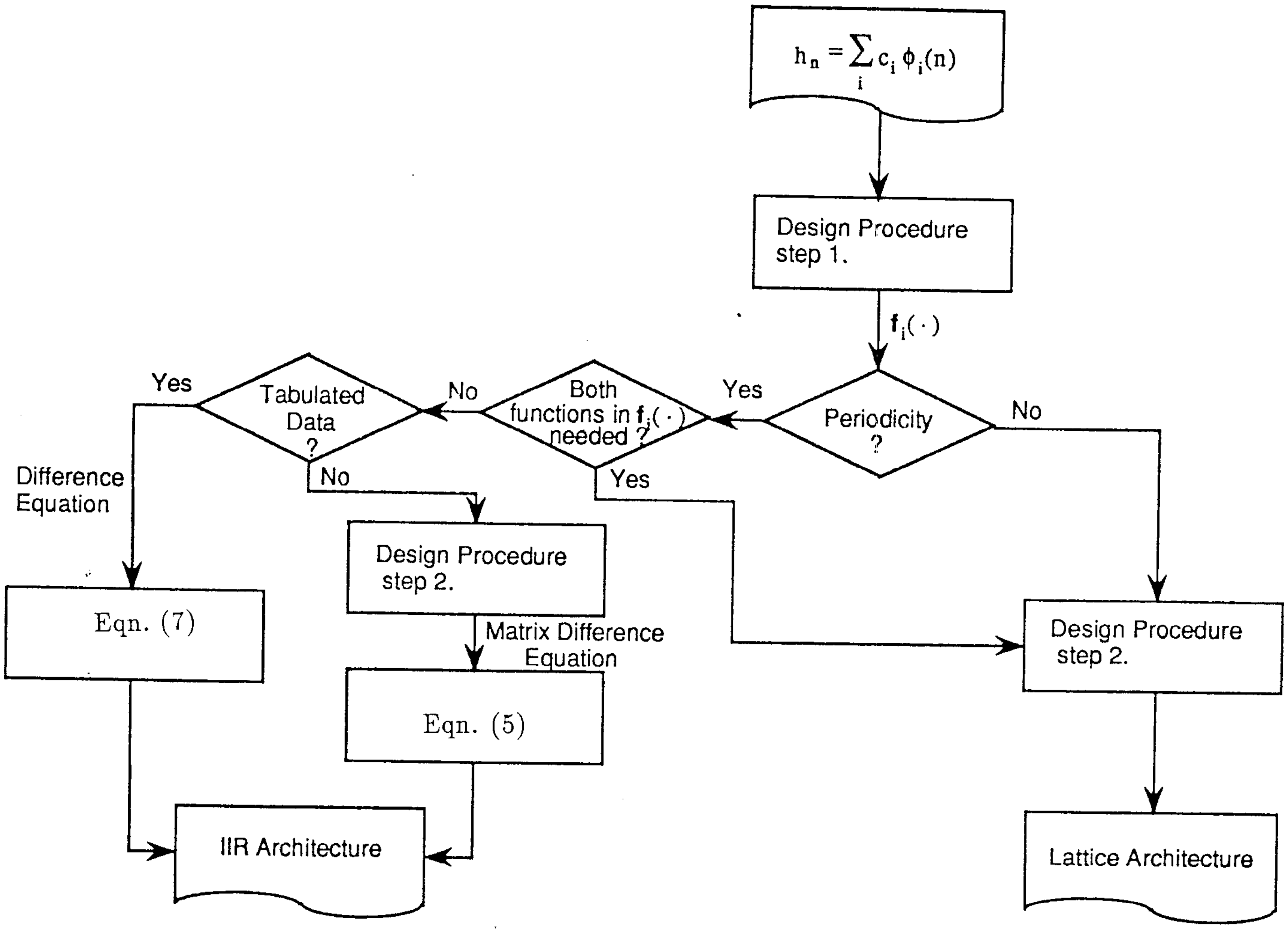


Figure 5: Architecture design procedure.

4 Design Examples

In this Section, we consider the architectural implementation of the Discrete Cosine Transform (DCT) and the Modulated Lapped Transform (MLT). For the first one an IIR architecture is used, while for the second one we will see that the lattice architecture is more appropriate.

The N-point Discrete Cosine Transform of a semi-infinite sequence of (real, scalar) data $x(\cdot)$ consists of N semi-infinite sequences $X_k(\cdot), k = 0, 1, \cdots, N-1$ defined as follows:

$$X_k(t) = c_k \sum_{n=0}^{N-1} \cos \frac{k\pi}{N} \left(n + \frac{1}{2}\right) x(t+n-N+1), t = 0, 1, \cdots \tag{10}$$

where $c_0 = \sqrt{\frac{1}{N}}$ and $c_k = \sqrt{\frac{2}{N}}, k = 1, 2, \cdots, N-1$. Consequently, the k^{th} frequency component of the DCT is specified by the mapping operator $\left[h_n = c_k \cos \frac{k\pi}{N}(n+\frac{1}{2}), n = 0, 1, \cdots, N-1\right]$. The latter can be expressed as

$$h_n = c_k f_{k,0}(n), \tag{11}$$

where $f_{k,0}(n)$ is an element of the kernel group

$$\mathbf{f}_k(n) = \begin{bmatrix} f_{k,0}(n) \\ f_{k,1}(n) \end{bmatrix} = \begin{bmatrix} \cos \frac{k\pi}{N}(n+\frac{1}{2}) \\ \sin \frac{k\pi}{N}(n+\frac{1}{2}) \end{bmatrix}. \tag{12}$$

$\mathbf{f}_k(n)$ satisfies the shift property with

$$\mathbf{R} = \begin{bmatrix} \cos \frac{k\pi}{N} & \sin \frac{k\pi}{N} \\ -\sin \frac{k\pi}{N} & \cos \frac{k\pi}{N} \end{bmatrix}. \tag{13}$$

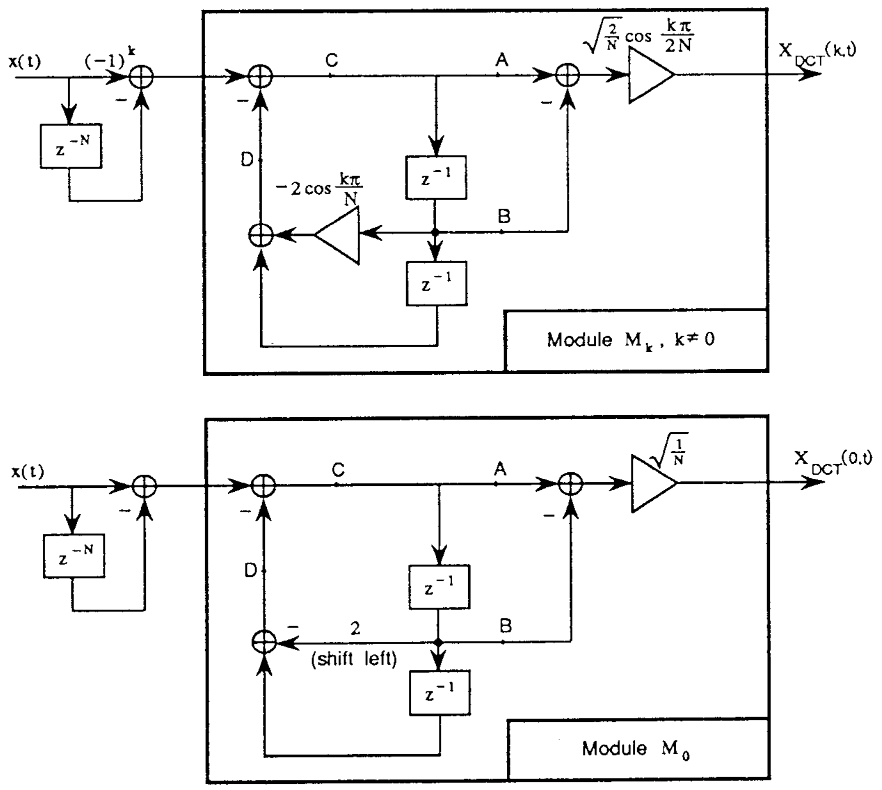


Figure 6: IIR architecture for the DCT kernel function.

Furthermore, it satisfies the periodicity property with $S = (-1)^k$ (cf. Lemma 3.1). Note finally that only one kernel function is used in the linear expression (11). Based on the above pieces of information the flow graph in Fig. 5 dictates that the IIR architecture should be used and the architecture parameters are specified based on (5). The resulted architecture for the k^{th} frequency component is shown in Fig. 6, while the architecture for the DCT with $N = 8$ is shown in Fig. 7. This architecture was first used in the context of sliding transforms in adaptive filtering [1]. It was recently rediscovered and used for a block transform VLSI architecture in data coding [8]. In [6], we prove that the inverse DCT can be implemented by using building blocks identical to the ones of the direct DCT.

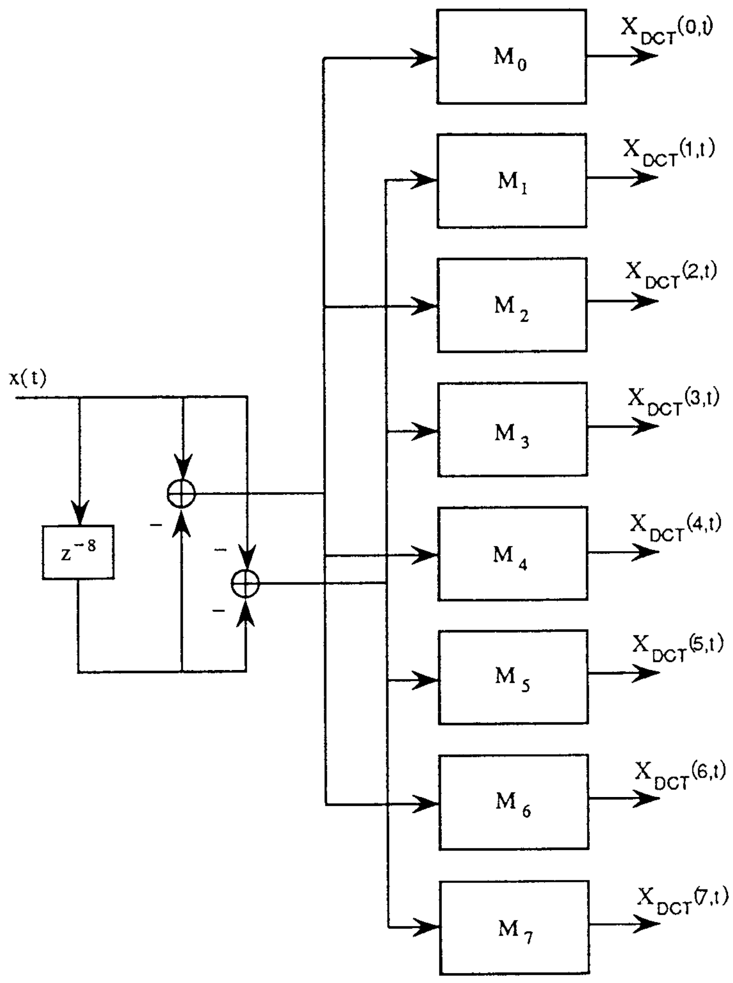


Figure 7: Recursive architecture for the DCT.

The Modulated Lapped Transform (MLT) belongs in the family of the Lapped Orthogonal Transforms that intend to suppress the blocking effect in transform coding [9, 10]. The MLT operates on segments of data of length $2N$, $x(t+n-2N+1), n = 0, 1, \cdots, 2N-1$ and it produces N output coefficients $X_k(t), k = 0, 1, \cdots, N-1$ as follows [9]:

$$X_k(t) = c_k\sqrt{\frac{2}{N}}\sum_{n=0}^{2N-1} \sin\frac{\pi}{2N}\left(n+\frac{1}{2}\right)\cos\left[\frac{\pi}{N}\left(k+\frac{1}{2}\right)\left(n+\frac{1}{2}+\frac{N}{2}\right)\right]x(t+n-2N+1), \quad (14)$$

where $t = 0, 1, \cdots$ and $c_k = (-1)^{(k+2)/2}$ if k is even and $c_k = (-1)^{(k-1)/2}$ if k is odd. The sequence of the k^{th} output coefficients $X_k(t), t = 0, 1, \cdots$ can be thought of as the output of the mapping operator

$$h_{k,n} = \left(c_k\sqrt{\frac{2}{N}}\right) \sin\frac{\pi}{2N}\left(n+\frac{1}{2}\right)\cos\left[\frac{\pi}{N}\left(k+\frac{1}{2}\right)\left(n+\frac{1}{2}+\frac{N}{2}\right)\right]. \tag{15}$$

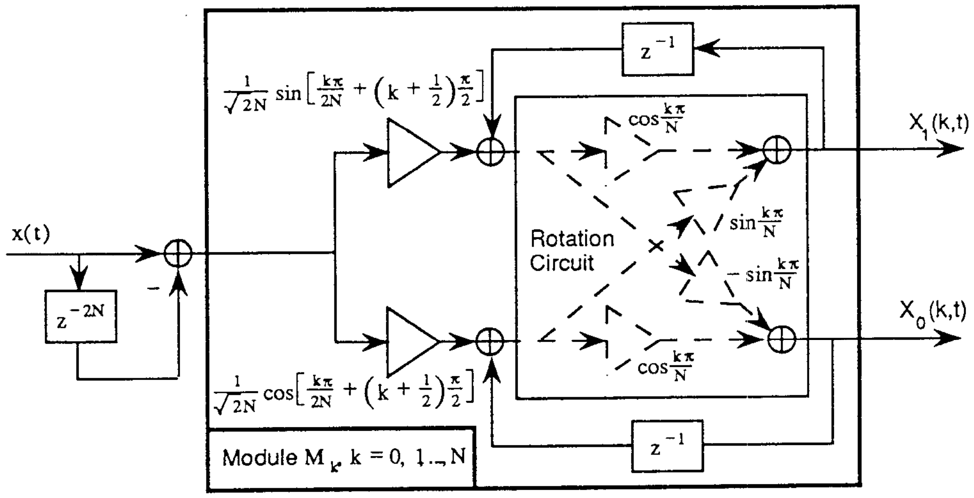
After a few algebraic manipulations, we derive the following decomposition of the mapping operator:

$$h_{k,n} = -\left(c_k\sqrt{\frac{1}{2N}}\right) f_{k+1,0}(n) - \left(c_k\sqrt{\frac{1}{2N}}\right) f_{k,1}(n), \quad k = 0, 1, \cdots, N-1, \tag{16}$$

where

$$\begin{bmatrix} f_{k,0}(n) \\ f_{k,1}(n) \end{bmatrix} = \mathbf{f}_k(n) = \begin{bmatrix} \cos\left[\frac{\pi}{N}k\left(n+\frac{1}{2}\right)+\left(k+\frac{1}{2}\right)\frac{\pi}{2}\right] \\ \sin\left[\frac{\pi}{N}k\left(n+\frac{1}{2}\right)+\left(k+\frac{1}{2}\right)\frac{\pi}{2}\right] \end{bmatrix}$$

is the associated kernel group. One can verify that the latter satisfies both the shift property and the periodicity property. Observe also that both kernel functions in $\mathbf{f}_k(n)$ participate in the linear expression (16). Consequently, the flow graph in Fig. 5 dictates that the lattice architecture should be used. The resulting design for the case $N = 8$ is given in Fig. 8 and Fig. 9. Note that the lattice is a rotation circuit that can be implemented with a CORDIC processor [7].



$\frac{1}{\sqrt{2N}}\sin\left[\frac{k\pi}{2N}+\left(k+\frac{1}{2}\right)\frac{\pi}{2}\right]$
z^{-1}
$\cos\frac{k\pi}{N}$
$X_1(k,t)$
x(t)
$\sin\frac{k\pi}{N}$
Rotation Circuit
$-\sin\frac{k\pi}{N}$
z^{-2N}
-
$X_0(k,t)$
$\cos\frac{k\pi}{N}$
$\frac{1}{\sqrt{2N}}\cos\left[\frac{k\pi}{2N}+\left(k+\frac{1}{2}\right)\frac{\pi}{2}\right]$
z^{-1}
Module M_k, k = 0, 1..., N

Figure 8: Lattice architecture for the MLT module.

5 Conclusion

The time-recursive computation has been successfully used in a number of diverse areas, such as real-time data coding and transform domain adaptive filtering. The *shift property* dictates the common infrastructure of the time-recursive computation in a variety of applications. The time-recursive approach yields architectural implementations that are modular, regular and require local communication, thus they are very appropriate for VLSI implementation. We develop a routine that can be used for designing time-recursive architectures in a systematic way. This routine specifies the guidelines for a CAD tool that could be specified high level description of mapping operators and produce VLSI layout. A more detailed presentation of this routine and its derivation is made in [5]. The time-recursive architectures of the Discrete Cosine Transform and the Modulated Lapped Transform are discussed. On table 1, we provide the relevant cost metrics. More examples of time-recursive architectures can be found in [6].

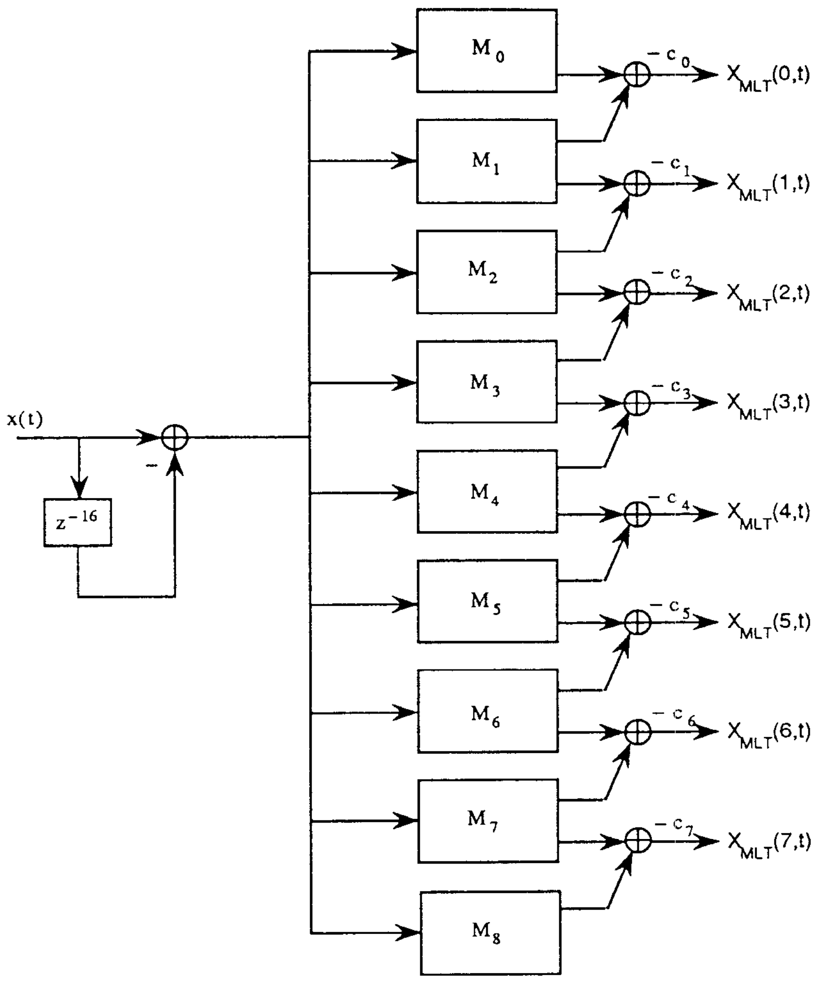


Figure 9: Recursive architecture for the MLT.

	multiplications	additions	rotations
DCT	$2N-1$	$3N+2$	-
MLT	$2N+3$	$3N+3$	$N-1$

Table 1: Operator counts for time-recursive architectures of some N-point block transforms.

References

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

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

[3] 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.

[4] 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.

[5] E. Frantzeskakis, J.S. Baras, and K.J.R. Liu. Time-Recursive Computation and Real-Time Parallel Architectures, Part I: Framework. *submitted to IEEE Trans. on SP*, July 1993.

[6] 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.

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

[8] 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.

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

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

[11] 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.

[12] 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.