92-23

PROCEEDINGS

SPIE—The International Society for Optical Engineering

Advanced Signal Processing Algorithms, Architectures, and Implementations IV

Franklin T. Luk
Chair/Editor

11–13 July 1993
San Diego, California

Volume 2027

Time-recursive computation and real-time parallel architectures, with application on the Modulated Lapped Transform

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

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

ABSTRACT

The time-recursive computation has been proved particularly useful for the real-time evaluation of one and two-dimensional block transforms. Unlike the FFT based ones, 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 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 architecture derivation.

Using this background, we design an architecture for the Modulated Lapped Transform (commonly called Modified Discrete Cosine Transform), which has linear cost in operator counts.

1. INTRODUCTION

Since the revolutionary publication by Cooley and Tukey[4] the fast Fourier transform (FFT) has been playing a key role in a wide range of applications, including transform coding, data filtering and spectral estimation[16,20,22]. Nevertheless, the increasing demand of processing huge volumes of data in real time, especially in audio, radar, sonar and video applications, suggests that the FFT-like algorithms may not be considered as the main building block in such applications. Three different reasons corroborate to this conjecture:

- *Global Communication:* The flow graphs of the FFT and the FFT-like fast algorithms exhibit a common structure composed of a series of $\log_2 N$ alternating butterfly interconnection and multiplier stages[16,20,25]. The butterfly communication scheme requires global communication links if a parallel implementation is to be considered.

- *Block Processing:* The FFT-like algorithms require buffering of blocks of data, and then block processing. This is not the natural way of processing for a number of applications such as audio, radar, sonar and video, where the the input data are supplied in a sequential manner.

- *Sliding Transform Computational Complexity:* The FFT has been widely used for the transformation of windowed data, where the frequency content of the data is extracted for each displacement of a sliding window. This phenomenon appears in the transform domain adaptive filtering problem[17,19]. The Discrete Fourier Transform used in such applications is referred to as *the sliding DFT*[22]. The FFT implementation of the sliding DFT requires the repetitive processing of neighboring data samples when the window slides.

0-8194-1276-7/93/$6.00

The time-recursive approach manages to overcome these disadvantages and provides efficient architectural implementations for block transforms[18,19,9,26,11,12,13,3].

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

$$X(t+1) = U(X(t), x(t+1), x(t+1-N))$$

(see Fig. 1). Within the time-recursive approach, the 8×8 Discrete Fourier Transform for example is viewed as eight linear operators that function independently from each other: given a common input sequence they produce eight coefficient sequences. The frequency coefficients are evaluated at time instant $t+1$ based on their values at time instant t and the input samples at the edges of the sliding window in Fig. 1. For the k^{th} frequency component we have $\left[h_n = e^{-j\frac{2\pi}{N}kn}, n = 0, 1, \cdots, N-1\right]$. The time-recursive update procedure for this case is[21,11]

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

There is a common infrastructure among the time-recursive computations in the above mentioned diverse areas. 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 a design example: the Modulated Lapped Transform (MLT)[14]. The time-recursive implementation of an Extended Lapped Transform[16] with basis length equal to $4N$ is also briefly discussed. 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 a specified final condition $\mathbf{f}(N)$, where $\mathbf{R}$ is a constant matrix of size $M \times M$.

Lemma 0.1 *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

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

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:

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

and therefore we obtain:

$$X_p(t+1) = \sum_{q=0}^{M-1} \left\{ r_{pq} \left[X_q(t) - x(t-N+1) f_q(0) + x(t+1) f_q(N) \right] \right\}, \tag{3}$$

$p = 0, 1, \cdots, M-1$. If we assume knowledge of the boundary values $\{f_q(0), f_q(N),\ q = 0, 1, \cdots, M-1\}$, (3) 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 (3).*

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 0.2:

Lemma 0.2 *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*

$$\begin{bmatrix} f_0(n) \\ f_1(n) \end{bmatrix} = \begin{bmatrix} c_{00} b^n + c_{01} b^{-n} \\ c_{10} b^n + c_{11} b^{-n} \end{bmatrix}, \tag{4}$$

   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.*
4. *The union of two kernel groups that satisfy SP.*
5. *The cartesian product of two kernel groups that satisfy SP.*

The proof of this lemma can be found in [5] and [7].

In [7] and [6] we demonstrate how an arbitrary mapping operator can be expressed in an optimal manner as a superposition of the basis functions specified by Lemma 0.2. Here we assume that such an expression can be obtained by inspection (and utilizing Lemma 0.2). For example, the kernel functions of the Short Time Fourier Transform, the Discrete Cosine Transform and the Modulated Lapped Transform 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 0.2 / statement 3, we get* $\mathbf{f}_i(n) = \left[1 \; n \; n^2\right]^T$.

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 (3) 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. (3) dictates an architectural implementation that has the lattice structure in Fig. 2. In an abuse of terminology, we will use the name *lattice architecture* for the architectures that realize (3) regardless the size of the associated kernel group.

An alternative time-recursive architecture, obtained by using a transfer function approach, is shown in Fig. 3, where[7]

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

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

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

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[7]

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

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]).

Fig. 2 and Fig. 3 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* (defined next) is satisfied by the kernel group in question.

The role of these factors becomes clearer with the example 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 (4). In [5] (as well as in [7]) we prove the following Lemma:

Lemma 0.3 *If the parameter* b *of the kernel group (4) is of the form* $b = e^{j\beta}$, *then (4) 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. 4 and Fig. 5 show how the architectures in Fig. 2 and Fig. 3 will be modified if the periodicity property holds.

Fig. 6 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 a 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.

4. ARCHITECTURE FOR MODULATED LAPPED TRANSFORM

In this Section, we consider the architectural implementation of the Modulated Lapped Transform (MLT)[14]. We also discuss the design of the time-recursive architecture for an Extended Lapped Transform (ELT) with basis length equal to $4N$ [15,16]. Both transforms are Lapped Orthogonal Transforms, thus they exhibit the desirable property of suppressing the blocking effects when used in data compression. The use of these transforms in a novel audio coding scheme has been reported recently[24]. Furthermore, if they are used in transform domain adaptive filtering they operate as Quadrature Mirror Filter (QMF) banks. In both cases the time-recursive implementation is particularly useful.

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[14]:

$$X_k(t) = c_k\sqrt{\frac{2}{N}}\sum_{n=0}^{2N-1}\sin\frac{\pi}{2N}\left(n+\frac{1}{2}\right)\cos\left[\frac{\pi}{N}\left(k+\frac{1}{2}\right)\left(n+\frac{1}{2}+\frac{N}{2}\right)\right]x(t+n-2N+1), \tag{9}$$

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

$$h_{k,n} = \left(c_k\sqrt{\frac{2}{N}}\right)\sin\frac{\pi}{2N}\left(n+\frac{1}{2}\right)\cos\left[\frac{\pi}{N}\left(k+\frac{1}{2}\right)\left(n+\frac{1}{2}+\frac{N}{2}\right)\right]. \tag{10}$$

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

$$h_{k,n} = -\left(c_k\sqrt{\frac{1}{2N}}\right)f_{k+1,0}(n) - \left(c_k\sqrt{\frac{1}{2N}}\right)f_{k,1}(n), \quad k = 0, 1, \cdots, N-1, \tag{11}$$

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. For this kernel group we have $\mathbf{f}_k(n-1) = \mathbf{R}_k\mathbf{f}_k(n)$, where

$$\mathbf{R}_k = \begin{bmatrix} \cos\frac{k\pi}{N} & \sin\frac{k\pi}{N} \\ -\sin\frac{k\pi}{N} & \cos\frac{k\pi}{N} \end{bmatrix}. \tag{12}$$

We also have

$$\begin{bmatrix} f_{k,0}(0) \\ f_{k,1}(0) \end{bmatrix} = S\begin{bmatrix} f_{k,0}(2N) \\ f_{k,1}(2N) \end{bmatrix} = \begin{bmatrix} \cos\left[\frac{k\pi}{2N}+\left(k+\frac{1}{2}\right)\frac{\pi}{2}\right] \\ \sin\left[\frac{k\pi}{2N}+\left(k+\frac{1}{2}\right)\frac{\pi}{2}\right] \end{bmatrix}, \tag{13}$$

where $S = 1$. Therefore, the periodicity property is satisfied. Since both member functions of the kernel group appear in the decomposition (11), the lattice architecture is recommended (cf. Fig. 6). In Fig. 7, we provide the lattice architecture module that is used as the building block in the MLT architectural implementation. The latter is depicted on Fig. 8 for the case of $N = 8$. Note that the lattice is a rotation circuit that can be implemented very efficiently either with a CORDIC processor[10] or by a distributed arithmetic approach[23]. The lattice architecture we propose for the N-point MLT involves $2N+3$ multipliers, $3(N+1)$ adders, $N-1$ rotation circuits and local interconnection, as opposed to $\frac{N}{2}(\log N+1)$ multipliers, $\frac{3N}{2}(\log N+1)$ adders, $\frac{N}{2}$ rotation circuits and global interconnection of the implementation in [14]. The IIR implementation for building the MLT modules is also considered for the sake of completeness. By substituting the expressions (12) and (13) in (5) we obtain

$$\begin{array}{ll} d_1 = -2\cos\frac{k\pi}{N} & n_{00} = \cos\left[\left(k+\frac{1}{2}\right)\frac{\pi}{2}-\frac{k\pi}{2N}\right] \\ d_2 = 1 & n_{01} = -\cos\left[\left(k+\frac{1}{2}\right)\frac{\pi}{2}+\frac{k\pi}{2N}\right] \\ & n_{10} = \sin\left[\left(k+\frac{1}{2}\right)\frac{\pi}{2}-\frac{k\pi}{2N}\right] \\ & n_{11} = -\sin\left[\left(k+\frac{1}{2}\right)\frac{\pi}{2}+\frac{k\pi}{2N}\right]. \end{array}$$

The resulted IIR module is obtained by using the above expressions in Fig. 5.

One can view MLT as a cosine transform for which the input data are modulated by a *sin*-shaped window. The modulation in time domain is equivalent to a convolution operation in frequency domain. Under this light, we observe that the time-recursive design in Fig. 8 implements the data modulation in the frequency domain (more accurately, in a cosine transform domain). The length of the convolution that takes place in this domain is equal to 2, since the *sin*-shaped window contains two frequency components. Other Extended Lapped Transforms utilize more complex, sum-of-cosine type modulation windows[15,16]. One can easily observe that the shape of the modulation window (and in particular the number of the summation terms) relates with the locality of the communication links in the associated time-recursive architecture. This is the same phenomenon appears in the use of sum-of-cosine windows in the Short Term Fourier Transform[1].

For the sake of concreteness consider the following ELT for which the basis length is equal to $4N$. It operates on segments of data of length $4N$, $x(t+n-4N+1), n = 0,1,\cdots,4N-1$ and it produces N output coefficients $X_{ELT}(k,t), k = 0,1,\cdots,N-1$ [16]:

$$X_{ELT}(k,t) = \sqrt{\tfrac{2}{N}}\sum_{n=0}^{4N-1}\left[-\tfrac{1}{2\sqrt{2}}+\tfrac{1}{2}\cos\tfrac{\pi}{N}\left(n+\tfrac{1}{2}\right)\right]\cos\left[\tfrac{\pi}{N}\left(k+\tfrac{1}{2}\right)\left(n+\tfrac{1}{2}+\tfrac{N}{2}\right)\right] x(t+n-4N+1), \tag{14}$$

where $t = 0,1,\cdots$. The modulation window is

$$\left[-\frac{1}{2\sqrt{2}}+\frac{1}{2}\cos\frac{\pi}{N}\left(n+\frac{1}{2}\right),\quad n=0,1,\cdots,4N-1\right].$$

Obviously it is a sum-of-cosine type of window that contains three frequency components. One can show that in transform domain the modulation operation translates into a convolution operation with length equal to 3. The implementation cost of the associated time-recursive architecture is slightly higher from the cost of the MLT (see Table 1), while the locality of the design is maintained[5].

5. CONCLUSION

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 have developed 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 for the associated time-recursive architectures.

The time-recursive architecture of the Modulated Lapped Transform has been designed and the architecture for an Extended Lapped Transform is discussed. These designs have an important impact on real-time audio data coding. In particular, the real-time computation of the MLT has become very interesting, since it has been incorporated recently by the ISO-MPEG and ASPEC standards for audio data coding with the name Modified Discrete Cosine Transform (MDCT).

6. ACKNOWLEDGEMENTS

E. Frantzeskakis and J.S.Baras were partially supported by National Science Foundation grant NSFD CDR 8803012. K.J.R. Liu was partially supported by the ONR grant N00014-93-1-0566.

7. REFERENCES

1. H. Babic and G.C. Temes, "Optimum Low-Order Windows for Discrete Fourier Transform Systems",*IEEE Trans. on ASSP*, Vol. ASSP-24, pp. 512-517, Dec. 1976.

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

3. C.T. Chiu and K.J.R. Liu, "Real-Time Parallel and Fully Pipelined Two-Dimentional DCT Lattice Structures with Application to HDTV Sysytems", *IEEE Trans. on Circuits and Systems for Video Technology*, Vol. 2, pp. 25-37, March 1992.

4. J.W. Cooley and J.W. Tukey, "An algorithm for machine computation of complex Fourier series", *Math. Comput.*, Vol.19, pp. 297-301, 1965.

5. E. Frantzeskakis, An Architectural Framework for VLSI Time-Recursive Computation with Applications, PhD Thesis, The University of Maryland at College Park, 1993.

6. E. Frantzeskakis, J.S. Baras and K.J.R. Liu, "Time-Recursive Computation and Real-Time Parallel Architectures, Part I: Analysis",*Institute for Systems Research, University of Maryland at College Park, TR-93-17*, 1993.

7. E. Frantzeskakis, J.S. Baras and K.J.R. Liu, "Time-Recursive Computation and Real-Time Parallel Architectures, Part II: Methodology and Applications", *Institute for Systems Research, University of Maryland at College Park, TR-93-18*, 1993.

8. G. Goertzel, "An algorithm for the evaluation of finite trigonometric series", *Amer. Math. Monthly*, Vol. 65, pp. 34-35, 1958.

9. Y.H. Hu, "CORDIC-Based VLSI Architectures for Digital Signal Processing", *IEEE Signal Processing Magasine*, Vol. 9, pp. 16-35, July 1992.

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

11. K.J.R. Liu and C.T. Chiu, "Unified Parallel Lattice Structures for Time-Recursive Discrete Cosine/Sine/Hartley Transforms", *IEEE Trans. on ASSP*, Vol. ASSP-41, pp. 1357-1377, May 1993.

12. K.J.R. Liu, C.T. Chiu, R.K. Kolagotla and J.F. Jájá, "Optimal Unified Architectures for the Real-Time Computation of Time-Recursive Discrete Sinusoidal Transforms", *Submitted to IEEE Trans. on Circuits and Systems for Video Technology*, 1993.

13. H.S. Malvar, "Lapped Transforms for Efficient Transform/Subband Coding", *IEEE Trans. on ASSP*, Vol. ASSP-38, pp. 969-978, June 1990.

14. H.S. Malvar, "Extended Lapped Transforms: Properties, Applications, and Fast Algorithms", *IEEE Trans. on ASSP*, Vol. ASSP-40, pp. 2703-2714, Nov. 1992.

15. H.S. Malvar, Signal Processing with Lapped Transforms, Artech House,Inc., Boston, 1992.

16. D.F. Marshall, W.K. Jenkins and J.J. Murphy, "The Use of Orthogonal Transforms for Improving Performance of Adaptive Filters", *IEEE Trans. on Circuits and Systems*, Vol. 36, pp. 474-484, April 1989.

17. N.R. Murthy and M.N.S. Swamy, "On the Computation of Running Discrete Cosine and Sine Transforms", *IEEE Trans. on ASSP*, Vol. ASSP-40, pp. 1430-1437, June 1992.

18. S.S. Narayan, A.M. Peterson, and M.J. Narasimha, "Transform Domain LMS Algorithm", *IEEE Trans. on ASSP*, Vol. ASSP-31, pp. 609-615, June 1983.

19. H.J. Nussbaumer, Fast Fourier Transform and Convolution Algorithms, Springer, Berlin, 1981.

20. A. Papoulis, Signal Analysis, McGraw-Hill,Inc, New York, 1977.

21. J.J. Shynk, "Frequency-Domain and Multirate Adaptive Filtering", *IEEE Signal Processing Magasine*, Vol.9, pp. 14-37, Jan. 1992.

22. S.G. Smith and S.A. White, "Hardware Approaches to Vector Plane Rotation", Proc. IEEE ICASSP, pp. 2128-2131, 1988.

23. L.F.C. Vargas and H.S. Malvar, "ELT-Based Wavelet Coding of High-Fidelity Audio Signals", Proc. IEEE ISCAS, 1993.

24. Z. Wang, "Fast Algorithms for the Discrete W Transform and for the Discrete Fourier Transform", *IEEE Trans. on ASSP*, Vol. ASSP-32, pp. 803-816, Aug. 1984.

25. P. Yip and K.R. Rao, "On the Shift Property of DCT's and DST's", *IEEE Trans. on ASSP*, Vol. ASSP-35, pp. 404-406, March 1987.

		implementation cost (mult, add, rotation)
MLT	time-recursive	$2N+3, 3N+3, N-1$
	fast algorithm	$\frac{N}{2}(\log_2 N+1)$, $\frac{3N}{2}(\log_2 N+1)$, $\frac{N}{2}$
ELT	time-recursive	$3N+4, 4N+4, N+2$
	fast algorithm	$\frac{N}{2}(\log_2 N+1)$, $\frac{3N}{2}(\log_2 N+2)$, N

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

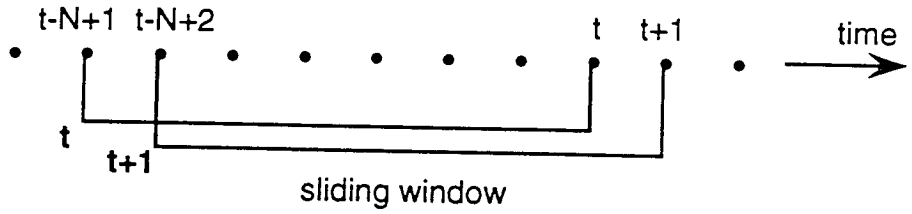

Figure 1: Pertain to the time-recursive computation.

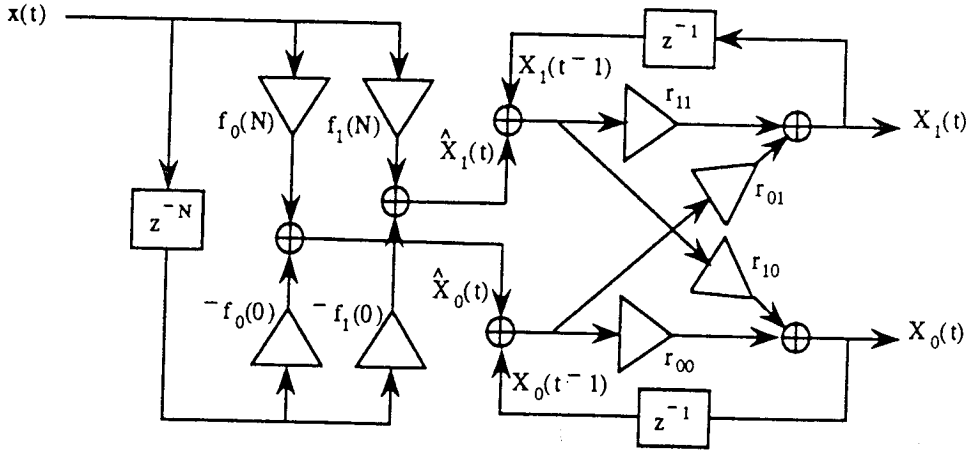

Figure 2: Lattice architecture for kernel group of size $M = 2$.

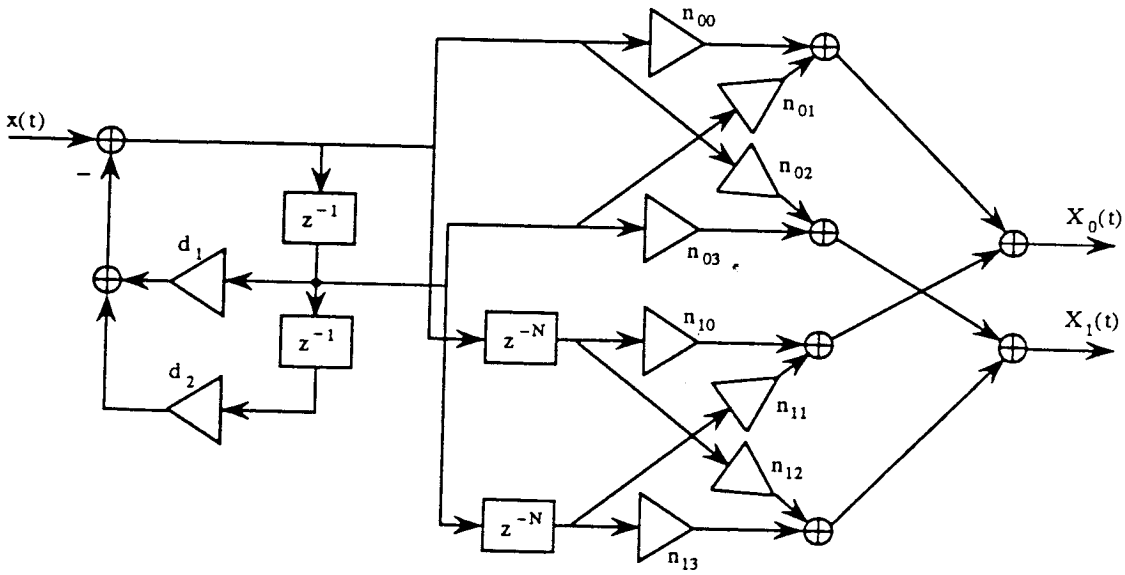

Figure 3: IIR architecture for $M = 2$.

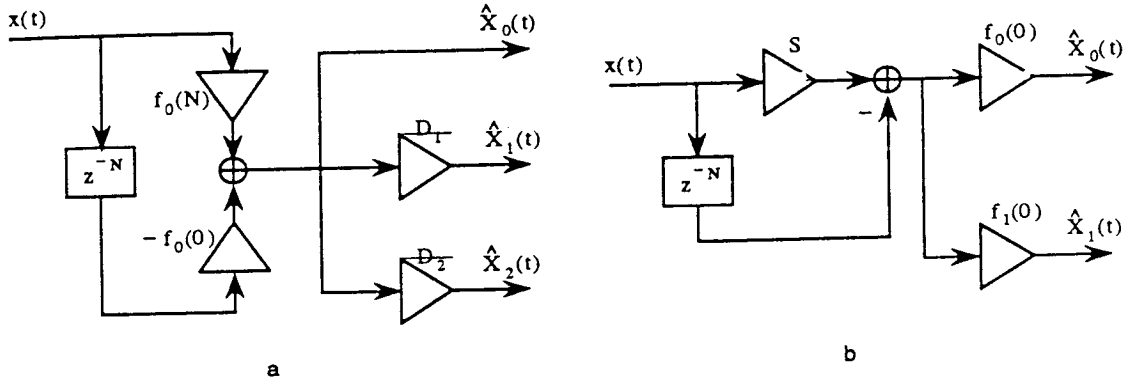

Figure 4: Part of lattice architecture if the periodicity property is satisfied.

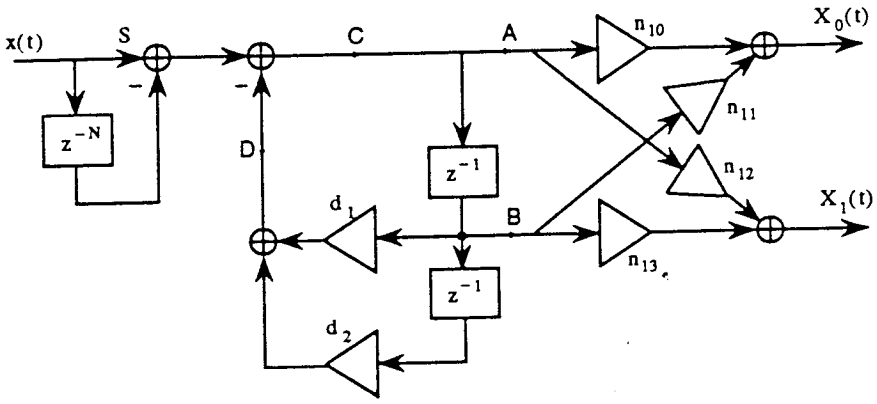

Figure 5: IIR architecture for $M = 2$ if the periodicity property is satisfied.

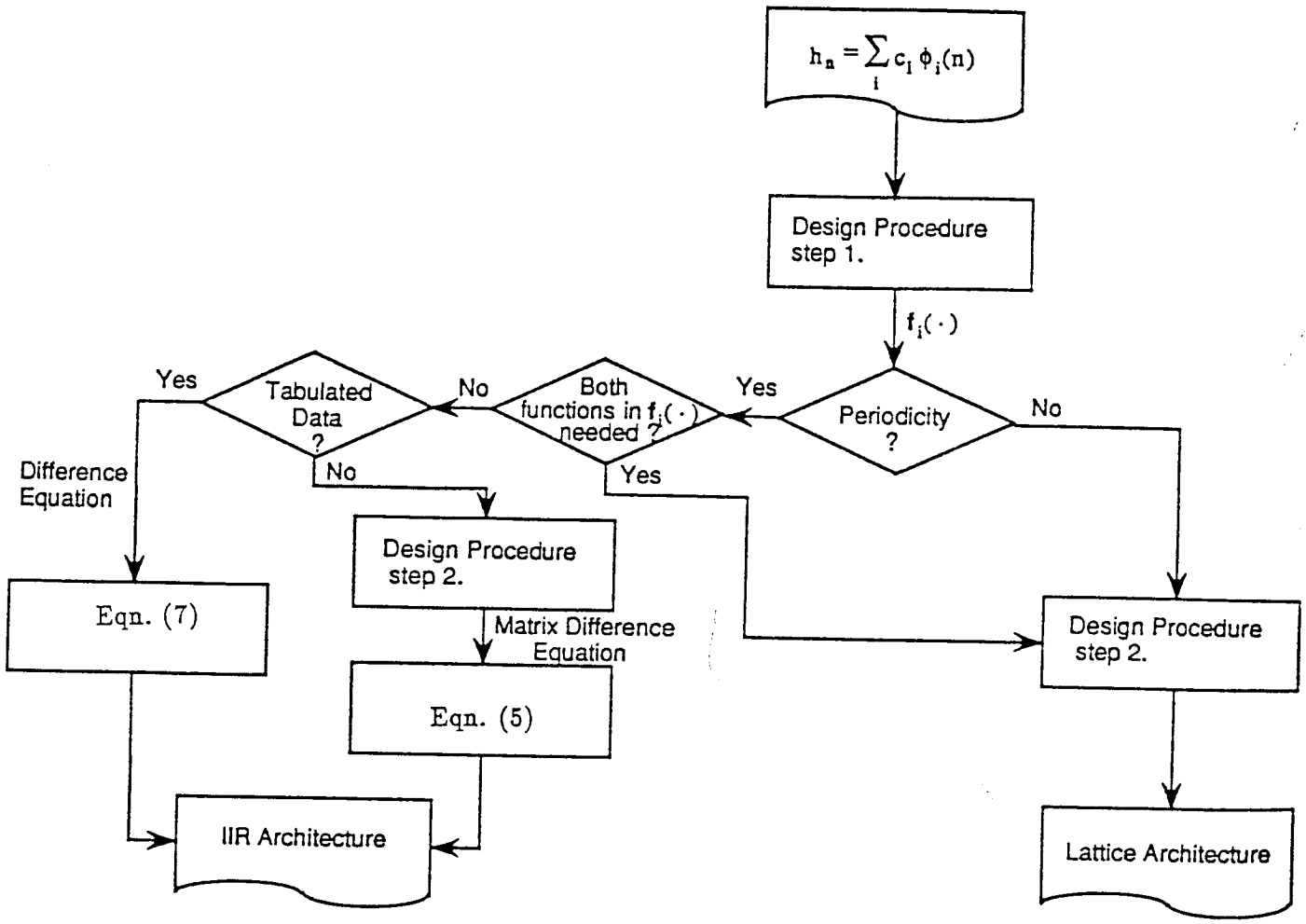

Figure 6: Architecture design procedure.

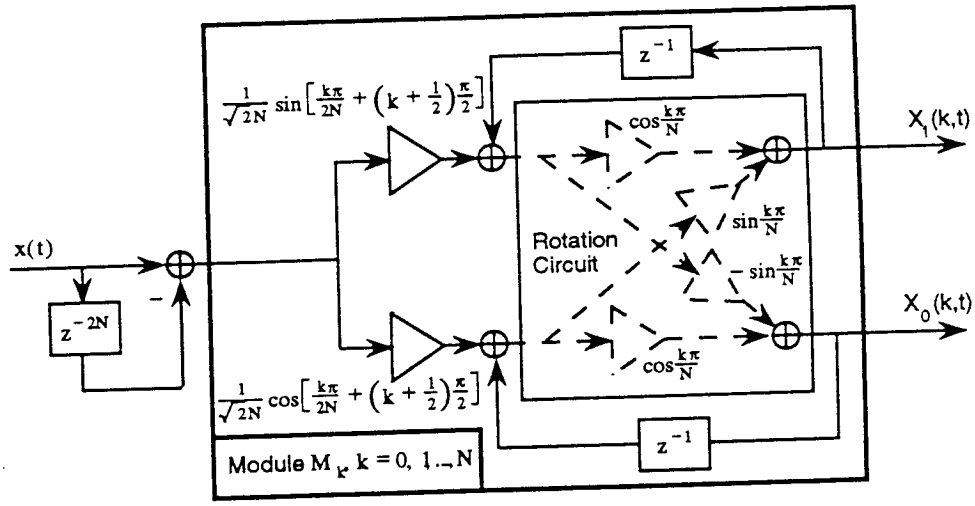

Figure 7: Lattice architecture for the MLT module.

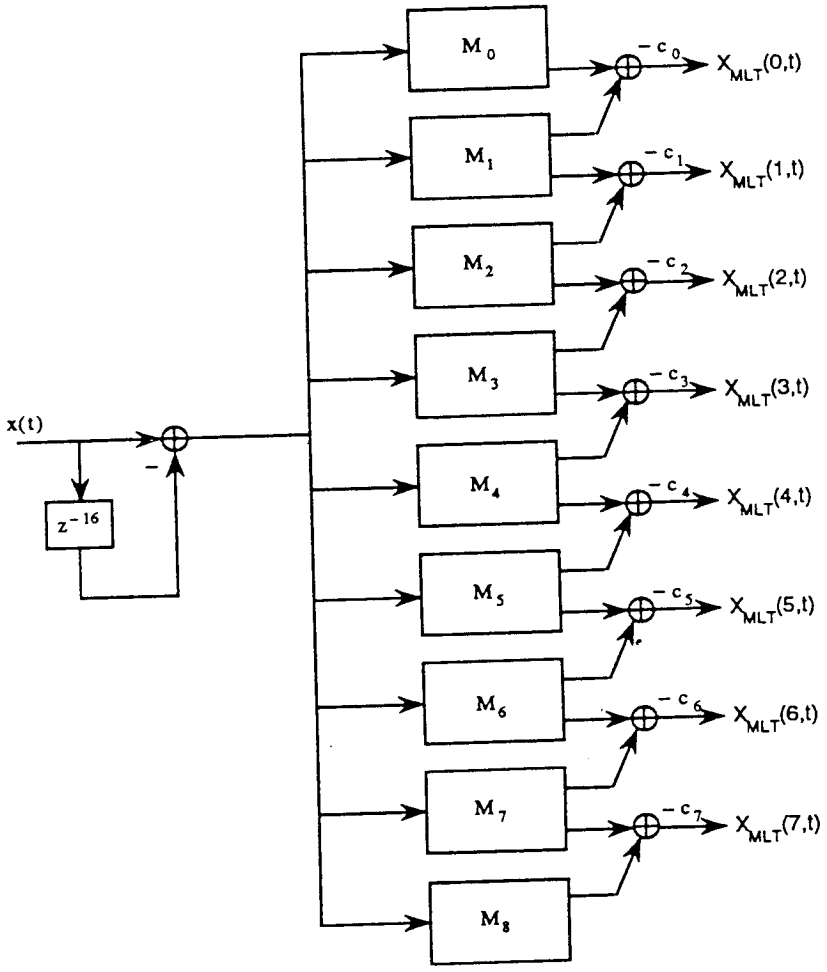

Figure 8: Time-recursive architecture for the MLT.