PARALLEL-RECURSIVE FILTER STRUCTURES FOR THE COMPUTATION OF DISCRETE TRANSFORMS

Richard J. Kozick and Maurice F. Aburdene

Electrical Engineering Department
Bucknell University, Lewisburg, PA 17837
kozick@bucknell.edu

ABSTRACT

A general approach is presented for implementing discrete transforms as a set of first-order or second-order recursive digital filters. Clenshaw's recurrence formulae are used to formulate the second-order filters. The resulting structure is suitable for efficient implementation of discrete transforms in VLSI or FPGA circuits. The general approach is applied to the discrete Legendre transform as an illustration.

1. INTRODUCTION

Parallel-recursive algorithms for the computation of forward and inverse discrete transforms have been studied recently due to their suitability for efficient VLSI implementation. Earlier papers have addressed the implementation of the discrete cosine transform (DCT) and related trigonometric transforms such as the discrete sine transform, discrete Hartley transform, and discrete Fourier transform [1]-[7]. In this paper, a general framework is developed for the design of parallel-recursive filter structures for the computation of discrete transforms. Previously-known implementations for the DCT [5] are time-invariant digital filters and follow as special cases of the general structure presented here. In addition, new parallel and efficient time-varying filter structures are presented for the discrete Legendre transform in which the number of time-varying filter coefficients is minimized.

The *forward transform* of a set of $N+1$ real-valued data points $y(0), y(1), \ldots, y(N)$ is defined by

$$Y(n) = \frac{1}{C_n^2} \sum_{k=0}^{N} y(k)\, P_n(k), \quad n = 0, 1, \ldots, N, \qquad (1)$$

and the *inverse transform* is defined by

$$y(k) = \sum_{n=0}^{N} Y(n)\, P_n(k), \quad k = 0, 1, \ldots, N. \qquad (2)$$

In (1) and (2), the $P_n(k)$ are the orthogonal basis vectors for the discrete transform with $\sum_{k=0}^{N} P_n(k) P_m(k) = \delta_{mn} C_n^2$. The basis vectors are assumed to be orthogonal to facilitate the application to discrete polynomial transforms in Section 3. However, the methods described in Section 4 are applicable to orthogonal as well as non-orthogonal transforms.

Section 2 describes first-order and second-order filter bank structures for computation of discrete transforms. The first-order and second-order filter implementations are compared, and Clenshaw's recurrence formulae are introduced as a tool for designing the second-order filters. Section 3 presents an analytical approach for designing a parallel-recursive implementation of the inverse transform when the basis vectors $P_n(k)$ are orthogonal polynomials. Section 4 describes a numerical procedure for designing parallel-recursive structures for forward and inverse discrete transforms for a wide class of basis vectors.

2. GENERAL FILTER STRUCTURES

Any transform can be implemented with first-order filters, but in many cases the second-order filter implementation is more efficient.

2.1. First-Order Filters

A first-order difference equation that computes the forward transform $Y(n)$ in (1) for a particular n is

$$\begin{aligned} \psi_{-1} &= 0 \\ \psi_k &= \psi_{k-1} + P_n(k) y(k), \quad k = 0, \ldots, N \qquad (3) \\ Y(n) &= (1/C_n^2)\psi_N. \end{aligned}$$

A parallel bank of $N+1$ first-order filters of the form (3) are needed to compute the transform values $Y(0)$, ..., $Y(N)$ from the data values $y(0), \ldots, y(N)$. Note that the filter coefficients $P_n(k)$ vary with the time index k. Hence implementation of the first-order filter bank

requires $(N+1)^2$ memory locations to store the time-varying coefficients $P_n(k)$, as well as a total of $N+1$ adders, $N+1$ multipliers, and $N+1$ delay elements. The memory requirement $(N+1)^2$ grows rapidly as N increases. Using a bank of second-order filters requires twice as many adders, multipliers, and delay elements, but in many cases the amount of memory required to store the time-varying filter coefficients is *linearly* proportional to N. Further, the second-order filters are *time-invariant* for many discrete trigonometric transforms [1]-[7], which greatly simplifies the implementation compared with the time-varying filters in (3). The remainder of the paper presents an approach for designing second-order filter banks that are more efficient than first-order filter banks for the computation of discrete transforms.

2.2. Second-Order Filters

Clenshaw's recurrence formulae [8] provide a general approach to deriving a second-order filter bank for discrete transform computation. As with the first-order filters in (3), the forward transform is computed by applying the data points $y(0), \ldots, y(N)$ serially to the bank of $N+1$ recursive digital filters. The transformed data $Y(0), Y(1), \ldots, Y(N)$ are then available in parallel after $N+1$ time steps. The structure of each recursive filter is identical and will be specified below. The inverse transform in (2) is computed with a similar parallel-recursive architecture, except that the input sequence $Y(0), \ldots, Y(N)$ is applied serially to the filter bank, and and the outputs $y(0), \ldots, y(N)$ appear in parallel after $N+1$ time steps.

Clenshaw's recurrence formulae provide a parallel-recursive algorithm for computing sums of the form (1) and (2). Clenshaw's recurrence formulae were used in [5] to develop parallel-recursive implementations for the discrete cosine transform (DCT). The approach in [5] relied on trigonometric identities that are available for the DCT kernel, so the approach in [5] does not extend in a straightforward manner to other discrete transforms. A contribution of this paper is the application of the Clenshaw recurrence formulae to a large class of transforms, and in particular to the discrete Legendre transform (DLT) [9].

Clenshaw's recurrence formulae are applied to the forward transform in (1) as follows. First, a recurrence relation is required for the basis vectors of the form

$$\begin{aligned} P_n(k+1) &= \alpha(k,n)P_n(k) + \beta(k,n)P_n(k-1), \\ & \qquad n = 0, 1, \ldots, N, \\ & \qquad k = 1, 2, \ldots, N-1. \end{aligned} \tag{4}$$

The coefficients $\alpha(k,n)$ and $\beta(k,n)$ that satisfy (4) for a given set of basis vectors $\{P_n(k)\}$ are generally not unique and can be chosen in many ways, as will be described in Sections 3 and 4. Once the $\alpha(k,n)$ and $\beta(k,n)$ coefficients are fixed, then a recursive algorithm for computing $Y(n)$ in (1) is given by [8]

$$\begin{aligned} \psi_{-2} &= \psi_{-1} = 0 \\ \psi_k &= \frac{1}{\beta(k+1,n)} \left[\psi_{k-2} - \alpha(k,n)\psi_{k-1} - y(k)\right], \\ & \qquad k = 0, 1, \ldots, N. \end{aligned} \tag{5}$$

The transform value $Y(n) = \psi_N$ is computed with the following definitions:

$$\beta(N,n) = \text{Any nonzero value, e.g. 1} \tag{6}$$

$$\alpha(N,n) = -\beta(N,n)\frac{P_n(N-1)}{P_n(N)} \tag{7}$$

$$\beta(N+1,n) = -\frac{C_n^2}{P_n(N)}. \tag{8}$$

Figure 1 contains a signal flow graph for the filter. In the event that $P_n(N) = 0$, then (7) and (8) are not used, but instead $Y(n)$ is taken at time $k = N$ from the point in Figure 1 following the $-\alpha(N,n)$ multiplier, with $\alpha(N,n) = \beta(N,n)P_n(N-1)/C_n^2$.

The filter in Figure 1 has time-varying coefficients when $\alpha(k,n)$ or $\beta(k,n)$ change with the time index k. If the α and β coefficients vary independently with k and n, then $2(N+1)^2$ memory locations are needed for storage and there is no advantage relative to the first-order filters in (3). As will be shown in Sections 3 and 4, in many cases the α, β values can be chosen to satisfy the recursion (4) while only requiring memory of size proportional to N. Note that a bank of $N+1$ second-order filters with the structure in Figure 1 requires $2(N+1)$ adders, $2(N+1)$ multipliers, and $2(N+1)$ delay elements.

A recursive algorithm for the *inverse* transform in (2) is formulated using Clenshaw's recurrence formulae in a similar way. Instead of beginning with the recursion over k as in (4), the inverse transform begins with a recursion over n of the form

$$\begin{aligned} P_{n+1}(k) &= \alpha(n,k)P_n(k) + \beta(n,k)P_{n-1}(k), \\ & \qquad k = 0, 1, \ldots, N, \\ & \qquad n = 1, 2, \ldots, N-1. \end{aligned} \tag{9}$$

Although we have used the same symbols, the coefficients α, β are generally different in the k-recursion (4) and the n-recursion (9). Methods for finding α and β in (9) are considered in Sections 3 and 4. The recursive algorithm for the inverse transform is identical to the forward transform, except that the roles of k and n are interchanged. For example, the algorithm for

computing $y(k)$ with transform values $Y(0), \ldots, Y(N)$ entering serially is as follows.

$$\begin{aligned} \psi_{-2} &= \psi_{-1} = 0 \\ \psi_n &= \frac{1}{\beta(n+1,k)} \left[\psi_{n-2} - \alpha(n,k)\psi_{n-1} - Y(n)\right], \\ & \qquad n = 0, 1, \ldots, N. \end{aligned} \tag{10}$$

Then $y(k) = \psi_N$ with the following definitions:

$$\beta(N,k) = \text{Any nonzero value, e.g. 1} \tag{11}$$

$$\alpha(N,k) = -\beta(N,k)\frac{P_{N-1}(k)}{P_N(k)} \tag{12}$$

$$\beta(N+1,k) = -\frac{1}{P_N(k)}. \tag{13}$$

In the event that $P_N(k) = 0$, then (12) and (13) are not used, but instead $y(k)$ is taken at time $n = N$ from the point in the filter following the $-\alpha(N,k)$ multiplier, with $\alpha(N,k) = \beta(N,k)P_{N-1}(k)$.

Sections 3 and 4 describe methods for finding the α, β coefficients that satisfy the recurrence relations (4) and (9) for the basis vectors. Once the α, β coefficients are known, then (5) and (10) are the second-order recursive digital filters that compute the transform.

3. ORTHOGONAL POLYNOMIAL TRANSFORMS

An analytical approach is available for finding the coefficients for the n-recursion in (9) when the basis vectors $P_n(0), \ldots, P_n(N)$ are discrete, orthogonal, n^{th}-order polynomials. The polynomials can be computed using the recursive equation

$$(k-n)P_n(k) - a_n P_{n+1}(k) = b_n P_n(k) + c_n P_{n-1}(k) \tag{14}$$

where a_n is chosen so that the left side of (14) has degree n and then b_n, c_n are given by

$$\begin{aligned} b_n &= \sum_{k=0}^{N} (k-n) P_n(k) P_n(k) \\ c_n &= \sum_{k=0}^{N} (k-n) P_n(k) P_{n-1}(k). \end{aligned}$$

Comparing (14) with (9), α, β are identified as

$$\begin{aligned} \alpha(n,k) &= \frac{(k-n) - b_n}{a_n} \\ \beta(n,k) = \beta(n) &= -\frac{c_n}{a_n}. \end{aligned}$$

Note that β is independent of k, so it is denoted by $\beta(n)$.[1] This approach provides closed-form expressions for $\alpha(n,k)$ and $\beta(n)$ whenever the $P_n(k)$ are orthogonal polynomials.

For the discrete Legendre polynomials defined in [9], the results are

$$\begin{aligned} \alpha(n,k) &= \frac{2n+1}{(n+1)(N-n)}(N-2k) \qquad (15) \\ &= \alpha_1(n)\alpha_2(k) \\ \beta(n) &= -\frac{n(N+n+1)}{(n+1)(N-n)} \qquad (16) \end{aligned}$$

for $k = 0, 1, \ldots, N$ and $n = 1, 2, \ldots, N-1$, with $\alpha(N,k)$, $\beta(N,k)$, and $\beta(N+1,k)$ defined by (11)-(13). Note that $\alpha(n,k) = \alpha_1(n)\alpha_2(k)$ is separable in n and k, and the k-dependence is relatively simple with $\alpha_2(k) = N, N-2, \ldots, -N+2, -N$ for $k = 0, 1, \ldots, N$. Each filter for the inverse DLT computation has the form shown in Figure 2. The set of $N+1$ filters requires $2(N+1)$ adders, $3(N+1)$ multipliers, and $2(N+1)$ delays. Further, only $4N+2$ memory locations are required to store the filter coefficients, which for large N is more efficient than the first-order filter structure.

The forward transform requires a k-recursion of the form (4). An analytical approach does not seem to be available that exploits properties of orthogonal polynomials for this case. A general numerical procedure that is applicable to a wide class of forward and inverse discrete transforms is developed in the following section.

4. GENERAL NUMERICAL PROCEDURE

The α, β coefficients that satisfy the recursions (4) for a given set of basis vectors $\{P_n(k)\}$ can be chosen in many ways. Indeed, (4) contains $(N-1)(N+1)$ linear equations with $2(N-1)(N+1)$ variables, so a non-unique solution exists as long as the condition $P_n(k+1) \neq 0$ with $P_n(k) = P_n(k-1) = 0$ is avoided. However, the second-order implementation is efficient only if the time-varying filter coefficients can be stored in less than $2(N-1)(N+1)$ memory locations.

One approach for obtaining efficient second-order implementations is to constrain the $\alpha(k,n), \beta(k,n)$ in some manner, and then solve (4) subject to the constraints. One interesting and useful solution of this type occurs when the $P_n(k)$ are the DCT basis vectors, and the constraints are $\beta(k,n) = -1$. Then the numerical solution of (4) for $\alpha(k,n)$ corresponds exactly with the *time-invariant* filter implementation presented in [5]. As another application of this procedure, the inverse DLT implementation (15),(16) results when the β are constrained to vary only with n and not with k. Thus in these two instances, known solutions that

[1] At the final time step $n = N+1$, β generally does vary with k according to (13).

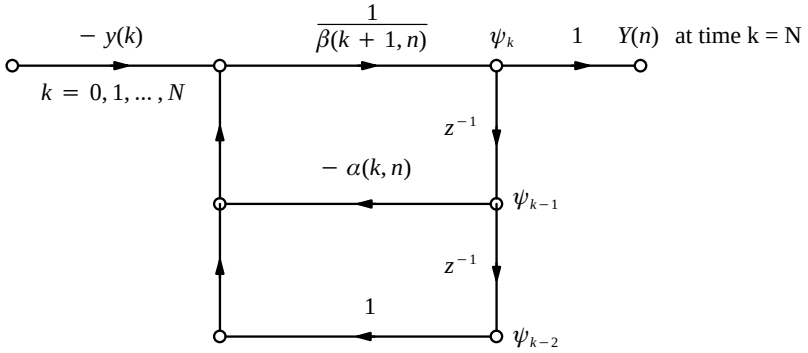

Figure 1: General second–order filter for computation of forward transform.

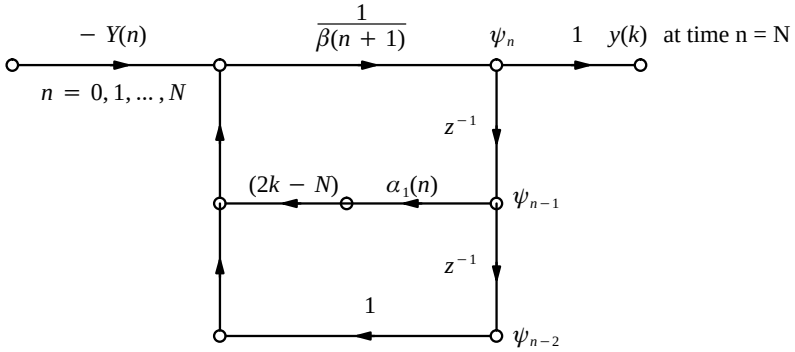

Figure 2: Filter for computation of inverse discrete Legendre transform.

were derived analytically are obtained from the numerical procedure with simple linear constraints.

A more general approach is to constrain the time-varying coefficients to be separable in k and n, i.e. $\alpha(k,n) = \alpha_1(k)\alpha_2(n)$ and $\beta(k,n) = \beta_1(k)\beta_2(n)$. The resulting implementation requires $6N$ memory locations to store the filter coefficients, but the equations are nonlinear and a solution is not guaranteed.

5. SUMMARY

Clenshaw's recurrence formulae were used to formulate filter structures for computation of discrete transforms. The approach provides an efficient structure for VLSI and/or FPGAs implementation of discrete transforms by reducing the number of time-varying coefficients. Known time-invariant filter structures for the discrete cosine transform (DCT) are special cases of the general procedure. A new and parallel time-varying structure to compute the discrete Legendre transform was presented.

6. REFERENCES

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

[2] J. Canaris, "A VLSI architecture for the real time computation of discrete trigonometric transforms," *J. VLSI Signal Processing*, vol. 5, pp. 95-104, 1993.

[3] L.-P. Chau and W.-C. Siu, "Recursive algorithm for the discrete cosine transform with general length," *Elect. Lett.*, vol. 30, pp. 197-198, 1994.

[4] Z. Wang, G.A. Jullien, and W.C. Miller, "Recursive algorithms for the forward and inverse discrete cosine transform with arbitrary length," *IEEE Sig. Proc. Lett.*, vol. 1, no. 7, pp. 101-102, July 1994.

[5] M.F. Aburdene, J. Zheng, R. Kozick, "Computation of Discrete Cosine Transform Using Clenshaw's Recurrence Formula," *IEEE Sig. Proc. Lett.*, vol. 2, no. 8, pp. 155-156, August 1995.

[6] K.J. Ray Liu and C.-T. Chiu, "Unified parallel lattice structures for time-recursive discrete cosine/sine/Hartley transforms," *IEEE Trans. Sig. Proc.*, vol. 41, no. 3, pp. 1357-1363, March 1993.

[7] E. Frantzeskakis, J.S. Baras, K.J. Ray Liu, "Time-recursive computation and real-time parallel architectures: a framework," *IEEE Trans. Sig. Proc.*, vol. 43, no. 11, pp. 2763-2770, Nov. 1995.

[8] W. Press, S.A. Teukolsky, W.T. Vetterling, and B.P. Flannery, *Numerical Recipes in C: The Art of Scientific Computing*, Cambridge, UK: Cambridge University Press, 1992, 2nd ed.

[9] N. Morrison, *Introduction to Sequential Smoothing and Prediction*, New York: McGraw-Hill, 1969.