Composite-Length Real-Valued Cyclic Convolutions

Hideo Murakami, Member, IEEE

Abstract—Recently the Bruun algorithm, which uses real arithmetic only, is generalized and modified to be applicable to the case when the length is other than a power of two. This paper derives efficient algorithms for the generalized Bruun algorithm focusing on the case when the length is prime because a composite-length algorithm can be decomposed into prime-length transforms.

Index Terms—Cyclic convolution, Fast algorithm, Prime-length algorithm, Real-value algorithm

I. INTRODUCTION

Computing convolutions is a very important issue in digital signal processing, and a variety of real-valued fast algorithms have been proposed in literature. One class of efficient algorithms is based on a factorization of the polynomial 1-z^{-N} over the real numbers. Bruun introduced a recursive real polynomial factorization for 1-z^{-N} when N is a power of two, and derived an efficient algorithm for computing the DFT [1], [2]. This algorithm was formulated in a more solid mathematical framework, modified to be suitable for implementation by the conventional FFT technologies [3], and further generalized to the case for N not a power of two [4].

Generalized Bruun Algorithm: The generalized Bruun algorithm has been derived using the set of polynomials defined by

\[
\phi_d = 1 - 2\cos(2\pi f)z^{-d} + z^{-2d}, \quad 0 < f < 1
\]

for a positive integer d and real f in the range 0 < f < 1. If an integer m is a multiple of d, the polynomial \(\phi_m\) is factored as

\[
\phi_m = \prod_{k=0}^{d-1} \phi_{m/d}(z^{d/k}), \quad 0 < f < 0.5.
\]

With regard to these factorizations, the T-transform is defined by

\[
T_{m,d}(X(z)) = (X_0(z), \ldots, X_{d-1}(z))
\]

where

\[
X_k(z) \equiv X(z) \mod \phi_{m/d}(z^{d/k}), \quad 0 < f < 0.5
\]

for a real signal X(z) of length 2m in its z-transform.

Of particular importance is the case when \(d=m=N/2\) and \(f=0\) assuming that N is even, that is, \(T_{N/2,0,N/2}\). For this case, the transform outputs are given by

\[
X_k(z) \equiv X(z) \mod \phi_{1/2}(z^{1/2}),
\]

for a real signal X(z) of length N. These residues are referred to as real-valued DFT (RDFT) of X(z), which are polynomials of degree less than two.

Computing the RDFT directly using the congruencies (5) is generally not efficient. When N/2 is composite, it is possible to derive a recursive factorization starting with \(\phi_{0,2} = 1-z^N\) leading to its factors \(\phi_{k,0,2}, \ k=0, 1, \ldots, N/2-1\), by applying the factorizations in (2) recursively. Applying the T-transforms according to the recursive factorization increases computational efficiency. This method is shown in Fig. 1 when N=24. The cyclic convolution can be computed using RDFT by a method similar to the system by DFT as shown in Fig. 2.

As easily conjectured from the factorization of (2), the algorithm for a composite-length signal is composed of transforms of the form \(T_{m,d}\) where d is prime. Moreover, if the 2m-length input signal is decomposed into m/d polyphase components of length 2d, the transform \(T_{m,d}\) can be computed by applying \(T_{d,d}\) to each component. Therefore, the problem of finding a fast algorithm for a composite-length signal reduces to the problem of finding an efficient implementation of \(T_{d,d}\) when d is prime. The computational details for \(T_{d,d}\) have already been treated in [3]. However, as yet, no serious effort has been made for improving the computational efficiency of \(T_{d,d}\) when d is an odd prime. For simplifying the notation, in the remainder of this paper, the transform \(T_{d,d}\) will be denoted simply as \(T_d\).

The objective of this paper is to derive efficient algorithms for \(T_d\) when d is an odd prime and f is a real number in the range 0 < f < 0.5. It is always assumed that d is the odd prime in the remainder of this paper.
Fig. 1. The generalized Bruun algorithm when $N=24$.

\[
\begin{align*}
H_i(z) &\xrightarrow{\text{RDFT}} X_i(z) \\
&\xrightarrow{\text{mod}(\phi_{1,0})} Y_i(z) \\
&\xrightarrow{\text{RDFT}} Y_i(z)
\end{align*}
\]

Fig. 2. The $N$-point cyclic convolution system using RDFT.

II. POLYNOMIAL FACTORIZATION ALGORITHMS FOR $T_{d,0}$

This section treats with the problem of finding efficient algorithms for $T_{d,0}$ which is based on the factorization

\[
\phi_{d,0} = \prod_{k=0}^{d} \phi_{1,0} \lambda_d(z).
\]

(6)

We first find recursive polynomial factorizations of $\phi_{d,0}=1-z^{-2d}$ leading to the factors given in the right-hand side of (6), and then derive a polynomial factorization algorithm for the transform.

If $\lambda_d(z)$ is defined by

\[
\lambda_d(z) = \sum_{m=0}^{d} z^{-m},
\]

(7)

The direct division of $\phi_{d,0}$ by $\phi_{1,0}$ immediately yields another factorization of $\phi_{d,0}$ as

\[
\phi_{d,0} = \phi_{1,0} \lambda_d(z').
\]

(8)

This equation implies that $\lambda_d(z')$ is the product of $\phi_{1,k/2,d}$, $k=1, 2, ..., d-1$. Using (2a), the factor polynomials can be properly paired and multiplied to yield $\phi_{2,k,d}=\phi_{1,k/2,d}^2 \phi_{1,k/4,d}$, $k=1, 2, ..., (d-1)/2$, and hence

\[
\lambda_d(z') = \prod_{k=1}^{(d-1)/2} \phi_{2,k,d}.
\]

(9)

This factorization suggests three step polynomial factorization algorithm for $T_{d,0}$.

Step 1: For a given real signal $X(z)$ of length $2d$, compute the residues

\[
\begin{align*}
A_0(z) &= X(z) mod(\phi_{1,0}) \\
A_1(z) &= X(z) mod(\lambda_d(z'))
\end{align*}
\]

Step 2: From $A_1(z)$, compute the residues

\[
B_k(z) = A_1(z) mod(\phi_{2,k,d}), \quad k=1, ..., (d-1)/2
\]

Step 3: For each $k=1, 2, ..., (d-1)/2$, compute the two residues

\[
\begin{align*}
C_{k,0}(z) &= B_k(z) mod(\phi_{1,k/2,d}) \\
C_{k,1}(z) &= B_k(z) mod(\phi_{1,k/4,d})
\end{align*}
\]

This algorithm is called a polynomial factorization algorithm (PFA). The PFAs for $T_{d,0}$ are described in Fig. 3 when $d=3, 5$ and 7.

Fig. 3. PFA for $T_{d,0}$: (a) $T_{3,0}$, (b) $T_{5,0}$, (c) $T_{7,0}$.

III. POLYNOMIAL FACTORIZATION ALGORITHMS FOR $T_{d,1/4}$

The transform $T_{d,1/4}$ $(0<\xi<0.5)$ computes the residues on the factors of $\phi_{d,1/4}$ given by the factorization

\[
\phi_{d,1/4} = \prod_{k=0}^{d-1} \phi_{1,1/4+k,d}.
\]

(10)

In this section we derive a polynomial factorization algorithm for $T_{d,1/4}$, which will be needed later for deriving algorithms for $T_{d,0}$ with arbitrary $f$. To do this, we need factorizations of $\phi_{d,1/4}=1+z^{-2d}$. We have

\[
\phi_{d,1/4} = \phi_{1,1/4} \lambda_d(z^4)
\]

(11)

After some calculations, we can show the factorization,

\[
\lambda_d(z^4) = \prod_{k=0}^{(d-1)/2} \phi_{2,1/4+k,d/2}.
\]

(12)

The PFA for $T_{d,1/4}$ from the above factorizations is the
following:

Step 1: For a real signal $X(z)$ of length $2d$, compute the residues

$A_0(z) = X(z) \mod (\Phi_{1,1/4})$

$A_k(z) = X(z) \mod (\lambda_k(-z^2))$

Step 2: From $A_1(z)$, compute the residues

$B_k(z) = A_k(z) \mod (\Phi_{d,2,(d-1)/2})$, $k = 0, 1, \ldots, (d-1)/2 - 1$

Step 3: For each $k=0, 1, \ldots, (d-1)/2 - 1$, compute the two residues

$C_{k,0}(z) = B_k(z) \mod (\Phi_{12,4k/4})$

$C_{k,1}(z) = B_k(z) \mod (\Phi_{12,4k-1/4})$

The PFAAs for $T_{d,1/4}$ are described in Fig. 4 when $d=3$, 5, and 7. For $d=3$, the second step computation is not necessary because $\lambda_2(-z^2)=\Phi_{2,1/8}$.

Several comments need to be made on the implementation of the polynomial factorization algorithms for $T_{d,0}$ and $T_{d,1/4}$. The final step residue computations of PFAAs for both $T_{d,0}$ and $T_{d,1/4}$ compute the residues $C_{d}(z)=B(z)\mod(\Phi_{1,0/2})$ and $C_{d}(z)=B(z)\mod(\Phi_{1,7/8})$ from a signal $B(z)$ on $\mod(\Phi_{2,2})$. If the signals are represented in the forms, $B(z)=b(-2)z^2+b(-1)z+b(0)+b(1)z^1 \mod(\Phi_{2,2})$, $C_{d}(z)=c(-1)z+c(0) \mod(\Phi_{1,0/2})$, and $C_{d}(z)=c(-1)z+c(0) \mod(\Phi_{1,7/8})$, the computation can be written as

\[
\begin{bmatrix}
   c_0(-1) \\
   c_0(0) \\
   c_0(-1) \\
   c_0(0)
\end{bmatrix} =
\begin{bmatrix}
   2\cos(\pi f_1) & 1 & 0 & -1 \\
   -1 & 0 & 1 & 2\cos(\pi f_1) \\
   -2\cos(\pi f_1) & 1 & 0 & -1 \\
   -1 & 0 & 1 & -2\cos(\pi f_1)
\end{bmatrix}
\begin{bmatrix}
   b(-2) \\
   b(-1) \\
   b(0) \\
   b(1)
\end{bmatrix}
\]  

(13)

because $z^2=2\cos(\pi f)z-1 \mod(\Phi_{1,0/2})$ and $z^2=2\cos(\pi f)z-1 \mod(\Phi_{1,7/8})$. The matrix multiplication by (13) involves two multiplications and six additions. A signal $A(z)$, if its length is $2m$, will be represented in the form

$A(z) = \sum_{n=0}^{2m} a(n) z^{-n}$

throughout the residue computations.

Residue computations of the form $B(z)=A(z) \mod(\gamma(z))$, appear in the first and second steps of the PFAAs for $T_{d,0}$ and $T_{d,1/4}$. If $A(z)$ is polyphase-decomposed in the form $A(z)=A^{(m)}(z)+z^{-1}A^{(m)}(z)$, and if $B^{(m)}(z)$, $m=0, 1$, are computed by $B^{(m)}(z)=A^{(m)}(z) \mod(\gamma(z))$, then $B(z)$ is given by $B(z)=B^{(m)}(z)+z^{-1}B^{(m)}(z)$. That is, the residue computation $B(z)=A(z) \mod(\gamma(z))$ can be completed by performing two residue computations on $\mod(\gamma(z))$ for a signal of a half-length to $A(z)$.

The resulting computational complexities for the implementations of $T_{d,0}$ and $T_{d,1/4}$ are summarized in Table I for $d=3$, 5, and 7. The Table also lists the complexities when the outputs of the transforms are implemented separately. As can be seen from Table I, the proposed algorithms perform better than the separate approaches.

IV. USEFUL IDENTITIES

It is extremely difficult to find factors of $\Phi_{d,f}$ for arbitrary $f$, and hence to derive the polynomial factorization algorithm for $T_{d,f}$. We need to take a different approach for deriving efficient algorithms. In this section, we derive some identities needed for deriving algorithms for $T_{d,f}$ in the next section.

A set of complex polynomials that possesses a factorization property similar to (10) was introduced in [5]. The polynomials are defined as

$\Phi_{d,f} = 1 - e^{i\pi f} z^{-d}$,  

(15)

where $d$ is an integer and $f$ is a real number. These polynomials possess the factorization property

$\Phi_{d,f} = \prod_{k=0}^{d} \Phi_{1,1/4}(\lambda_k) z^{-d}$.

(16)
This factorization is easily proved by substituting $z = e^{j2\pi(n-d)}$, $k=0, 1, ..., d-1$, into both sides of (16). It is important to note that this factorization is identical to the factorization (10) if $\phi$ is replaced by $\Phi$. With respect to this factorization, the $\tau$-transform is defined by

$$
\tau_{df}(U(z)) = \begin{cases} \left( U(0) \ldots U(d-1) \right) \\
\end{cases}
$$

(17)

for a polynomial $U(z)$ of degree less than $d$.

The outputs $U[k], k=0, 1, ..., d-1$, of the $\tau$-transform are computed by substituting $z = e^{j2\pi(n-d)/d}$ into $U(z)$, that is, the outputs can be given by

$$
U[k] = \sum_{n=0}^{d-1} u(n) \frac{1}{2 \pi} \int_{-\pi}^{\pi} \frac{e^{j2\pi f \pi k/d}}{e^{j2\pi f \pi n/d}}
$$

(18)

which can be rewritten as

$$
U[k] = \sum_{n=0}^{d-1} u(n) e^{-j2\pi f n k/d}
$$

(19)

for an arbitrary real $\alpha$. This expression implies that the outputs are computed by the congruence

$$
U[k] = U(e^{j2\pi n f /d}) \mod \{1 - e^{j2\pi n f /d}, z = 1\}
$$

(20)

for $k=0, 1, ..., d-1$. This means that $U[k]$ is computed first by multiplying $e^{j2\pi n f /d}$ for the coefficients $u(n), n=0, 1, ..., d-1$, of $U(z)$, and then taking $\tau_{df}$ of the resulting polynomial. We introduce the rotation matrix

$$
\Omega_f = \begin{bmatrix} \cos(2\pi f) & \sin(2\pi f) \\ -\sin(2\pi f) & \cos(2\pi f) \end{bmatrix}
$$

(21)

and represent the coefficient $u(n)$ in the form $u(n) = u^R(n) + ju^I(n)$ where $u^R(n)$ and $u^I(n)$ are real. Then the multiplication by $e^{j2\pi f n /d}$ for $u(n)$ can be written in the matrix form as

$$
\begin{bmatrix} \nu^R(n) \\ \nu^I(n) \end{bmatrix} = \Omega_f \begin{bmatrix} u^R(n) \\ u^I(n) \end{bmatrix}
$$

(22)

where $\nu(n) = \nu^R(n) + j\nu^I(n)$ is the value after the multiplication. The implication of (20) is described in Fig. 5 using the matrix multiplication notation, where the real and imaginary parts of the input as well as the outputs of the transform $\tau_{df}$ are represented by two separate lines. Such convention will be used in the following Figures as well.

The next theorem establishes an identity between the $\tau$-transform and the $T$-transform.

**Theorem 1:** Assume $U(z)$ and $U[k], k=0, 1, ..., d-1$, are related by (17), and real polynomials $X(z)$ and $X_e(z), k=0, 1, ..., d-1$, are related by (5) with $N=2d$. Then, if $f \neq 0$,

$$
U(z) = X(z) \mod (\Phi_{df})
$$

(23)

if and only if

$$
U[k] = X_k \mod (\Phi_{df})
$$

(24)

In order to explicitly describe the implication of this theorem, we will represent the complex signal $U(z)$ in the form $U(z) = U^R(z) + jU^I(z)$ and $X(z)$ in the form $X(z) = X^R(z) + X^I(z)$,

$$
X(z) = z^{2\pi} X(z) + X^R(z)
$$

(25)

where $X(z)$ is assumed to be represented according to the convention given by (14). Accordingly, the coefficients of $X(z)$ and $X^R(z)$ are related to the coefficients of $X(z)$ by $x(n) = x(n-d)$, and $x^R(n) = x(n)$, $n=0, 1, ..., d-1$, respectively. Since the residue $X(z) \mod (\Phi_{df})$ is computed by substituting $z = e^{j2\pi f}$ in $X(z)$, $U(z)$ is given by $U(z) = e^{j2\pi f} X(z) + X^R(z)$ under the relation (23). This relation between $X(z)$ and $U(z)$ is written in the matrix form as

$$
\begin{bmatrix} U^R(z) \\ U^I(z) \end{bmatrix} = \sum_{k=0}^{d-1} C_k \begin{bmatrix} X^R(z) \\ X^I(z) \end{bmatrix}
$$

(26)

where

$$
C_k = \begin{bmatrix} 
\cos(2\pi f k) & 1 \\
-\sin(2\pi f k) & 1 
\end{bmatrix}
$$

(27)

The multiplication by $C_k$ in (26) is applied to each pair $(x(n), x^R(n))$ of the coefficients for $n=0, 1, ..., d-1$, where superscript $T$ denotes the matrix transposition. The matrix $C_k$ is nonsingular as long as $f$ is not an integer, and the inverse of $C_k$ is given by

$$
C_k^{-1} = \begin{bmatrix} 0 & -1 \\
-\sin(2\pi f k) & 1 
\end{bmatrix}
$$

(28)

Realization of $T_{df}$ in terms of $\tau_{df}$ as implied by Theorem 1 is shown in Fig. 6, where the components $X(z)$ and $X^R(z)$ of the input as well as the components $x(-1)$ and $x^R(0)$ of the outputs $X^R(z)$ are represented by separate lines. By applying $C_k^{-1}$ from the left and $C_{df}$ from the right in Fig. 6, the realization of $\tau_{df}$ in terms of $T_{df}$ is obtained as shown in Fig. 7.
V. SYSTEMS FOR IMPLEMENTING $T_{d/f}$ WITH ARBITRARY $f$

The identities obtained in the previous section are now applied to derive efficient implementations for $T_{d/f}$ with arbitrary $f$. The combination of the identities in Figs. 6 and 5 immediately provides a means of converting $T_{d/f}$ into $\tau_{d/a}$ in which $\alpha$ can be chosen arbitrarily as shown in Fig. 8.

Implementation of $T_{d/f}$ via $T_{\tau_{a/f}}$ with an arbitrary real $\alpha$ in the range $0<\alpha<0.5$ can be obtained by converting $\tau_{d/a}$ in the system of Fig. 8 back into the transform $T_{d/f}$ using the identity in Fig. 7. This system provides a way of implementing $T_{d/f}$ via $T_{\tau_{a/f}}$, in which $\alpha$ can be chosen arbitrarily, as described in Fig. 9. In this system however, we cannot choose $\alpha=0$ because the identity in Fig. 7 is not applicable when $f=0$. A suitable choice would be to use $\alpha=1/4$ because a fast algorithm exists for $T_{d,1/4}$ as obtained in Section III. For this choice of $\alpha$, the pre-multiplication matrix is given by

$$C^{-1}_{1/4} = \begin{bmatrix} \sin(2\pi(f - 1/4) n/d) & \sin(2\pi(f - 1/4) n/d) \\ \cos(2\pi(f + 1/4) n/d) & \sin(2\pi(f + 1/4) n/d) \end{bmatrix}$$

and the post-multiplication matrix is given by

$$C_{1/4} = \begin{bmatrix} \sin(2\pi(1/4 + k)/d) \\ \sin(2\pi(1/4 + k)/d) \end{bmatrix}$$

A detailed diagram of the system for $d=3$ is shown in Fig. 10.

The overall computational complexities of the systems for $T_{d/f}$ when $d=3$, 5, and 7, are summarized in Table II. When $d=3$, no gain is obtained in using the proposed implementation. However, for $d=5$, the proposed implementations are more efficient than the separate computation.

VI. CONCLUSIONS

In this paper, real-valued fast algorithms for computing $T_{d/f}$ when $d$ is an odd prime were derived. Polynomial factorization algorithms were first derived for $T_{d,0}$ and $T_{d,1/4}$ by finding...
recursive factorizations of $\phi_{d,0}$ and $\phi_{d,1/4}$ based on the factorization property of $\phi$-polynomials.

The polynomial factorization approach is not feasible when $f$ is arbitrary. The complex transform $\tau_d$ introduced in [5] has the property that the frequency parameter $f$ can be shifted by pre-multiplications for coefficients of the input polynomial. In order to incorporate this property into the implementation of $T_d,f$, the identities between $\tau_d$ and $T_d,f$ were established. By combining the frequency shifting property of the $\tau$-transform and the identity between $\tau_d$ and $T_d,f$, the transform $T_d,f$ was shown to be implemented via $T_{d,1/4}$ for which the polynomial factorization algorithm can be applied. Comparisons show that, when $d \geq 5$, these implementations perform better than computing each output of $T_d,f$ separately. The algorithms for $T_d,f$ with prime $d$ have applications in improving the computational efficiency a cyclic convolution of real sequences with lengths other than a power of two.

REFERENCES