Bilinear Algorithms for Discrete Cosine Transforms of Prime Lengths

Venkatram Muddhasani and Meghanad D. Wagh[1]

Department of Electrical and Computer Engineering

Lehigh University, Bethlehem, PA 18015, USA

vem3@Lehigh.Edu, mdw0@Lehigh.Edu

Abstract: This paper presents a strategy to design bilinear discrete cosine transform (DCT) algorithms of prime lengths. We show that by using multiplicative groups of integers, one can identify and arrange the computation as a pair of convolutions. When the DCT length p is such that $(p-1)/2$ is odd, the computation uses two $(p-1)/2$ point cyclic convolutions. When $(p-1)/2 = 2^m q$ with $m > 0$ and q odd, the computation requires one $(p-1)/2$ point cyclic convolution and a combination of a q point cyclic convolution and a 2^m point Hankel product. Using bilinear algorithms for convolutions and Hankel products, one gets a bilinear DCT algorithm. We also show that the additions required beyond the convolutions can be minimized by a small modification to the convolution algorithms. This minimization exploits the fact that efficient bilinear convolution algorithms are almost always based on Chinese Remainder Theorem.

Keywords. Discrete cosine transform, Bilinear, Fast algorithm, Cyclic convolution, Group theory.

[1]Corresponding author. E-mail: mdw0@Lehigh.Edu, Tel: +1 (610) 758-4142, Fax: +1 (610) 758-6279.

1 Introduction

Discrete cosine transform (DCT) is one of the most popular transforms used in a variety of applications. It's close approximation to Karhunen-Loeve transform implies a high degree of energy compaction. It is therefore used in a wide range of data compression applications including the low bit rate video and speech coding [1] and JPEG and MPEG standards [2]. It is also used for linear filtering [1] and pattern recognition [3].

The ever increasing need for faster digital signal processing (DSP) applications has forced development of new high speed algorithms for core DSP computations such as discrete Fourier and discrete cosine transforms. Most algorithms follow the pioneering strategy outlined by Cooley and Tukey [4] that recursively decomposes the transform kernel into smaller kernels. The resultant algorithms for n point transforms generally have $O(\log n)$ sequential stages, each typically having $O(n)$ computational complexity [1, 5–7]. Even though this approach seems appealing computationally, it is applicable only to transforms lengths that are composite.

The other strategy, first used by Rader for obtaining a prime length DFT [8], identifies cyclic or skew-cyclic structures within the transform matrix. By using efficient algorithms for these known structures, one can obtain a fast algorithm for the transform. With the development of low complexity convolution algorithms [9, 10], this method of deriving transform algorithms looks even more appealing. It has been applied to the prime length discrete Fourier transform (DFT) [8, 10, 11], composite length DFT [10]. Since the cyclic convolution algorithms of [9, 10] are bilinear, the transform algorithms derived from them are also bilinear. They can be characterized by three sequential stages – the first and last stages of simple additions and a middle stage of multiplications. Further, all the multiplications are independent and can be done concurrently. Thus these algorithms have a further advantage if they are executed or implemented on architectures that can exploit this concurrency.

The DCT is also amenable to similar methods as demonstrated by Wagh and Ganesh [12] in case of arbitrary lengths. Their algorithm extracts convolution-like structures within the DCT transform kernel using group theoretic methots and then employs efficient cyclic convolution algorithms to derive the complete DCT algorithm. Chan and Siu [13] gave a method to convert a DCT computation of a prime length into two pure cyclic convolutions. Unfortunately, this conversion requires a number of extra multiplications and additions, thereby increasing the overall complexity of the procedure. Later they improved upon this by giving a new technique to convert the prime DCT computation into a cyclic and a skew-cyclic convolution [14]. This new technique uses only a few additions for the conversion, but forces the user to deal with a skew-cyclic convolution. This was further improved upon by Yin and Siu [15] by proving that the DCT of prime lengths p can be expressed as two cyclic convolutions if $(p-1)/2$ is also a prime.

In this paper we show that a p point DCT can be computed as two cyclic convolutions, each of length $(p-1)/2$, when $(p-1)/2$ is *any* odd number. For other primes, the DCT can be computed as a cyclic convolution and a skew cyclic convolution, each of length $(p-1)/2$. We also show that this $2^m q$ point skew-cyclic convolution can be converted to a multidimensional computation with a cyclic convolution of odd length q along one dimension and multiplication by a $2^m \times 2^m$ Hankel matrix along the other. Using bilinear algorithms for each of these algorithms, we obtain a bilinear algorithm for the DCT. Finally, the number of additions involved in the DCT computation is reduced by $(3p-7)/2$ through a small modification to the cyclic convolution algorithms used.

This paper is organized as follows. Section 2 deals with the design of bilinear DCT algorithms of prime lengths p; the case of $(p \bmod 4) = 3$ being treated in subsection 2.1 and the case of $(p \bmod 4) = 1$, in 2.2. Section 3 focuses on minimizing number of additions in DCT calculation. In Section 4 we discuss the complexity of the resultant DCT algorithm and compares our results with others. Finally, Section 5 presents our concluding remarks.

2 Design of the algorithms

As mentioned in Section 1, the DCT algorithms in this paper use cyclic and/or skew-cyclic convolutions. A cyclic convolution of N point sequences a and b is defined as

$$c(j) = \sum_{i=0}^{N-1} a(i)b((j-i) \bmod N), \qquad 0 \le j < N. \tag{1}$$

In this paper we encounter this same calculation, but in a slightly different form. By the mappings described later, a DCT can be split into smaller computations of the form

$$c(j) = \sum_{i=0}^{N-1} a(i)b((j+i) \bmod N), \qquad 0 \le j < N. \tag{2}$$

By reflecting the components of a to get $a' = [a_0, a_{N-1}, a_{N-2}, \ldots, a_1]$ and using it in (2) gives

$$c(j) = \sum_{i=0}^{N-1} a'(i)b((j-i) \bmod N), \qquad 0 \le j < N. \tag{3}$$

One can readily see that (3) is a cyclic convolution of sequences a' and b. Thus computation (2) can be carried out by reflecting a and cyclically convolving the resultant sequence with b. Clearly, the complexity and structure of this computation is identical to that of a cyclic convolution. In fact, following others [10], unless mentioned otherwise, cyclic convolution in this paper will refer to the computation (2).

Skew-cyclic convolution of N point sequences a and b is defined as

$$c(j) = \sum_{i=0}^{N-1} sign(j-i)a(i)b((j-i) \bmod N), \qquad 0 \le j < N, \tag{4}$$

where, $sign(k)$ equals -1 if $k < 0$ and $+1$ otherwise. Whereas a cyclic convolution corresponds to multiplication of polynomials modulo $(x^N - 1)$, a skew-cyclic convolution represents the same polynomial multiplication modulo $(x^N + 1)$. DCT algorithms presented later use the following computation

$$c(j) = \sum_{i=0}^{N-1} sign(N-(i+j))a(i)b((j+i) \bmod N), \qquad 0 \le j < N. \tag{5}$$

As before, one can show that by replacing a by $a' = [a_0, -a_{N-1}, -a_{N-2}, \ldots, -a_1]$ in (5), computation (5) reduces to the usual skew-cyclic convolution (4). Thus (5) has the same computational complexity and structure as (4) and will be referred to in this paper as the skew-cyclic convolution.

We now focus on DCT computation. The DCT of prime length p is defined as [1]

$$X(j) = \sum_{i=0}^{p-1} x(i) \cos((2i+1)j\pi \,/\, 2p), \qquad 0 \le j < p. \tag{6}$$

Equation (6) shows that $X(0)$ can be computed merely by adding all the x components. We discuss details of this calculation in Section 3. For the rest of the DCT, we compute even and odd components separately.

When j in (6) is even, $x(i)$ and $x(p-1-i)$ multiply with the same cosine term. Therefore, the computation of the DCT for these j's can be simplified to

$$X(j) = \sum_{i=0}^{(p-3)/2} y(i) \cos((2i+1)j\pi \,/\, 2p) \;+\; (-1)^{j/2} x((p-1)/2), \qquad 0 \le j < p, \quad j \text{ even}, \tag{7}$$

where

$$y(i) = x(i) + x(p-1-i). \tag{8}$$

Similarly, when j is odd, coefficients of $x(i)$ and $x(p-1-i)$ are negative of each other. Thus the DCT computation for these j's becomes

$$X(j) = \sum_{i=0}^{(p-3)/2} z(i) \cos((2i+1)j\pi \,/\, 2p), \qquad 0 \le j < p, \quad j \text{ odd}, \tag{9}$$

where

$$z(i) = x(i) - x(p-1-i) \tag{10}$$

Note that when $0 \le i \le (p-3)/2$, $(2i+1)$ represents integers less than $(p-1)$ that are relatively prime to $2p$. These integers are part of a well known set

$$A(2p) = \{n \in \mathbb{Z} \mid 1 \le n < 2p, \;\; \gcd(n, 2p) = 1\}$$

that forms a group under the operation of multiplication modulo $2p$. By applying permutations and negations obtained from the structure of this group, the DCT kernel can be turned into convolutions. However, the structure of $A(2p)$ can take two quite different forms depending upon p; thus forcing us to use two distinct design strategies described below.

2.1 DCT lengths: primes of type $\mathbf{4k+3}$

When DCT length p is of type $4k+3$, $(p-1)/2$ is odd and consequently,

$$A(2p) \equiv C_{p-1} \equiv C_2 \times C_{(p-1)/2}. \tag{11}$$

Clearly, in (11), subgroup $C_2 = \{1, (2p-1)\}$. Let g denote the generator of the subgroup $C_{(p-1)/2}$. A group has multiple generators. In the present application, we choose a generator g such that $4|(g-1)$. Even though we do not prove here that such a g always exists, it has been verified for primes (of type $4k+3$) less than 100,000.

To determine the signal and transform permutations, we define a function $\phi(\cdot) : \mathbb{Z} \to \mathbb{Z}$ as follows:

$$\phi(i) = \begin{cases} g^i \bmod 2p & \text{if} \quad (g^i \bmod 2p) < p \\ (-g^i) \bmod 2p & \text{otherwise.} \end{cases} \tag{12}$$

Similarly, to determine the negations, define a function $\sigma_c(\cdot) : \mathbb{Z} \to \{-1, 1\}$ as follows.

$$\sigma_c(i) = \begin{cases} -1 & \text{if} \quad p < g^i \bmod 4p < 3p \\ 1 & \text{otherwise.} \end{cases} \tag{13}$$

Function $\phi(\cdot)$ defined above has the following property.

Lemma 1. When $p \bmod 4 = 3$ and $g \in A(2p)$ is an element of order $(p-1)/2$, mapping $(\phi(\cdot)-1)/2$ is a permutation of integer set $\{0, 1, \ldots, (p-3)/2\}$.

Proof. The structure (11) of $A(2p)$ implies that all the $p-1$ elements of $A(2p)$ are given by

$$\{g^i \bmod 2p \mid 0 \le i \le (p-3)/2\} \cup \{(2p-1)g^i \bmod 2p \mid 0 \le i \le (p-3)/2\}. \tag{14}$$

Thus for each i, $i \in \{0, 1, \ldots, (p-3)/2\}$, $\phi(i)$ is distinct. Further, each $\phi(i) < p$ because from the definition (12), $\phi(i)$ equals $g^i \bmod 2p < p$ or $(-g^i) \bmod 2p = 2p - (g^i \bmod 2p) < p$. Therefore $0 \leq (\phi(i)-1)/2 \leq (p-3)/2$. ∎

Odd DCT components

Using Lemma 1, the odd transform components in (9) can be expressed as:

$$X(j) = \sum_{i=0}^{(p-3)/2} z((\phi(i)-1)/2) \cos(\phi(i) j\pi \,/\, 2p), \qquad 0 \leq j < p, \quad j \text{ odd}. \tag{15}$$

With the help of Table 1 one can replace the $\phi(i)$ in the argument of the cosine function in (15) by $g^i \bmod 4p$. By using the fact that j is odd and a function $\sigma_c(i)$ to take care of the proper sign, one can simplify (15) to:

$$X(j) = \sum_{i=0}^{(p-3)/2} \sigma_c(i) z((\phi(i)-1)/2) \cos((g^i \bmod 4p) j\pi \,/\, 2p), \qquad 0 \leq j < p, \quad j \text{ odd}. \tag{16}$$

Table 1: Values of $\phi(\cdot)$, $\sigma_c(\cdot)$ and $\sigma_s(\cdot)$ for various ranges of $g^i \bmod 4p$.

Range of $(g^i \bmod 4p)$	$\phi(i)$	$\sigma_c(i)$	$\sigma_s(i)$
$(0,\ p)$	$g^i \bmod 4p$	1	1
$(p,\ 2p)$	$2p - (g^i \bmod 4p)$	-1	1
$(2p,\ 3p)$	$(g^i \bmod 4p) - 2p$	-1	-1
$(3p,\ 4p)$	$4p - (g^i \bmod 4p)$	1	-1

An odd j in (16) can be replaced by $(2j'+1), 0 \leq j' \leq (p-3)/2$. Since the range of j' is the same as that of index i, it is also possible to permute it by $\phi(\cdot)$. With this permutation applied to the transform components, (16) becomes for $0 \leq j' \leq (p-3)/2$,

$$X(\phi(j')) = \sum_{i=0}^{(p-3)/2} \sigma_c(i)\sigma_c(j') z((\phi(i)-1)/2) \cos((g^i \bmod 4p)(g^{j'} \bmod 4p)\pi \,/\, 2p). \tag{17}$$

or,

$$X(\phi(j')) = \sum_{i=0}^{(p-3)/2} \sigma_c(i)\sigma_c(j')z((\phi(i)-1)/2)\cos(g^{i+j'}\pi\,/\,2p), \qquad 0 \le j' \le (p-3)/2. \tag{18}$$

Note that for the ranges of i and j', $(i+j') < p$. Thus,

$$g^{i+j'} = g^{(i+j') \bmod ((p-1)/2)} \cdot g^{(p-1)/2}. \tag{19}$$

However, order of g in $A(2p)$ is $(p-1)/2$. Let

$$g^{(p-1)/2} = 1 + k \cdot 2p \tag{20}$$

Thus

$$\begin{aligned} k \cdot 2p &= g^{(p-1)/2} - 1 \\ &= (g-1)(1 + g + g^2 + \cdots + g^{(p-3)/2}) \end{aligned} \tag{21}$$

Since $4 \mid (g-1)$, k in (21) is even. Using this fact and combining (18), (19) and (20) gives

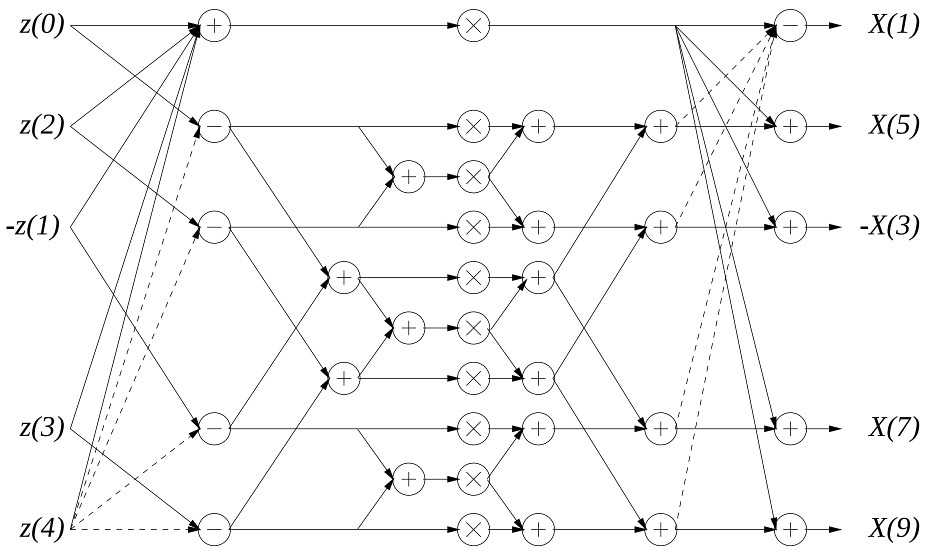
$$X(\phi(j')) = \sum_{i=0}^{(p-3)/2} \sigma_c(i)\sigma_c(j')z((\phi(i)-1)/2)\cos(g^{(i+j') \bmod ((p-1)/2)}\pi\,/\,2p), \qquad 0 \le j' \le (p-3)/2. \tag{22}$$

Equation (22) shows that the odd components of the DCT can be computed as a $(p-1)/2$ point cyclic convolution. In particular,

$$[\sigma_c(i)X(\phi(i))] = [\sigma_c(i)z((\phi(i)-1)/2)] \;*\; [\cos(g^i\pi\,/\,2p)], \qquad 0 \le i < (p-1)/2, \tag{23}$$

where $*$ denotes a cyclic convolution and the notation $[a(i)]$ stands for a sequence with components $a(0)$, $a(1)$, etc.

Figure 1 illustrates the computation of the odd components of an 11-point DCT obtained through a 5-point cyclic convolution. In this and later such figures, we also list the constants, m_0, m_1, ..., used in the multiplications from top to bottom. Note that a generator $g = 5$ is used to obtain the permutation $\phi(i) = 1, 5, 3, 7, 9$ and the sign $\sigma_c(i) = 1, 1, -1, 1, 1$ values for $i = 0$ through 4.



$m_0 = (c_0 + c_1 + c_2 + c_3 + c_4)/5$ $\quad m_1 = c_1 - c_2 - c_3 + c_4$ $\quad m_2 = c_2 - c_4$
$m_3 = -c_0 - c_2 + c_3 + c_4$ $\quad m_4 = c_3 - c_4$ $\quad m_5 = (-c_0 - c_1 - c_2 - c_3 + 4c_4)/5$
$m_6 = c_0 - c_4$ $\quad m_7 = c_0 - c_1 - c_3 + c_4$ $\quad m_8 = c_1 - c_4$
$m_9 = -c_0 - c_1 + c_2 + c_4$ $\quad c_i = \cos(5^i\pi/22)$

Figure 1: Odd components of the DCT of length 11.

Even DCT components

To compute even components of the DCT, one can similarly convert the summation term of (7) to a cyclic convolution. A simple addition of $(-1)^{j/2}x((p-1)/2)$ to every convolution output then gives all $X(j)$'s. Later in Section 3 we show how these additions may be accomplished efficiently. Here we focus on the evaluation of the summation term $\sum_{i=0}^{(p-3)/2} y(i)\cos((2i+1)j\pi\,/\,2p)$, denoted here as $X'(p-j)$. (Thus $X(j) = X'(p-j) + (-1)^{j/2}x((p-1)/2)$.) By splitting j in the cosine argument as $p-(p-j)$, one gets

$$X'(p-j) = \sum_{i=0}^{(p-3)/2} (-1)^i y(i)\sin((2i+1)(p-j)\pi\,/\,2p), \quad 0 \le (p-j) < p, \;\; (p-j) \text{ odd} \tag{24}$$

Note that (24) is similar to (15) except that it uses a sine function in place of a cosine and an odd $(p-j)$ in place of an odd j. The occurrence of sine requires use of a function $\sigma_s(\cdot) : \mathbb{Z} \to \{-1, 1\}$ defined as

$$\sigma_s(i) = \begin{cases} -1 & \text{if} \quad 2p < g^i \bmod 4p \\ 1 & \text{otherwise.} \end{cases} \tag{25}$$

The σ_s defined above has an interesting property stated in Lemma 2 below.

Lemma 2. Let $p \bmod 4 = 3$ and $g \bmod 4 = 1$. Then $\phi(\cdot)$ and $\sigma_s(\cdot)$ defined by (12) and (25) respectively, satisfy

$$\sigma_s(i)(-1)^{(\phi(i)-1)/2} \;=\; 1, \qquad \text{for all } \; 0 \le i < (p-1)/2.$$

Proof. Since $g \bmod 4 = 1$, $(g^i \bmod 4p) \bmod 4 = 1$. Then, $g^i \bmod 4p$ can be expressed as $4t+1$. Using this, and the values of functions $\phi(\cdot)$ and $\sigma_s(\cdot)$ from Table 1, the lemma can be easily verified. For example, if $0 < g^i \bmod 4p < p$, then $\phi(i) = 4t+1$ and $\sigma_s(i)(-1)^{(\phi(i)-1)/2} = (1)(-1)^{2t} = 1$. ∎

From Table 1, it is easy to verify that one now has the following expression similar to (16).

$$X'(p-j) = \sum_{i=0}^{(p-3)/2} \sigma_s(i)(-1)^{(\phi(i)-1)/2} y((\phi(i)-1)/2)\sin((g^i \bmod 4p)(p-j)\pi \,/\, 2p),$$

$$0 \le (p-j) < p, \quad (p-j) \text{ odd.}$$

Use of Lemma 2 now simplifies this to

$$X'(p-j) = \sum_{i=0}^{(p-3)/2} y((\phi(i)-1)/2)\sin((g^i \bmod 4p)(p-j)\pi \,/\, 2p),$$
$$0 \le (p-j) < p, \quad (p-j) \text{ odd.} \qquad (26)$$

Equation (26) is similar to (16) obtained in the odd DCT components case. One can therefore use a procedure similar to that following (16). Let $(p-j) = 2j'+1$. Then one can express the sequence $[\sigma_s(i)X'(\phi(i))]$ through a cyclic convolution as

$$[\sigma_s(i)X'(\phi(i))] = [y((\phi(i)-1)/2)] \; * \; [\sin(g^i\pi \,/\, 2p)], \qquad 0 \le i < (p-1)/2. \qquad (27)$$

Recall that to complete the computation of each $X(2j)$, one should add $(-1)^j x((p-1)/2)$ to the corresponding convolution output $X'(j)$.

2.2 DCT lengths: primes of type $\mathbf{4k+1}$

In this case, $(p-1)/2$ is even and $A(2p) = C_{p-1}$ cannot be split as before. Let g denote the generator of $A(2p)$. Using this g, define functions $\phi(\cdot) : \mathbb{Z} \to \mathbb{Z}$, $\sigma_c(\cdot) : \mathbb{Z} \to \{-1, 1\}$ and $\sigma_s(\cdot) : \mathbb{Z} \to \{-1, 1\}$ as before using (12), (13) and (25). We then have the following lemma.

Lemma 3. When $p \bmod 4 = 1$ and $g \in A(2p)$ is an element of order $(p-1)$, mapping $(\phi(\cdot)-1)/2$ is a permutation of integer set $\{0, 1, \ldots, (p-3)/2\}$.

Proof. We will prove that each $\phi(i)$ is distinct and $\phi(\cdot)$ is onto. Assume $i \neq j$. If $(g^i \bmod 2p)$ and $(g^j \bmod 2p)$ are both either less than or greater than p, then clearly $\phi(i)$ and $\phi(j)$ are distinct. Assume now that only one of them, say, $g^i \bmod 2p$ is less than p. If $\phi(i) = \phi(j)$, then

$$\begin{aligned} g^i + g^j &= 0 \mod 2p, \qquad \text{or} \\ g^{j-i} &= -1 \mod 2p. \end{aligned}$$

Since g is the generator of C_{p-1}, this implies that

$$(p-1)/2 \mid (j-i). \tag{28}$$

But (28) is impossible because $0 \leq j, i < (p-1)/2$. Therefore all $\phi(i)$'s are distinct for $i \in \{0, 1, \ldots, (p-3)/2\}$. Further, from the definition, $\phi(i) < p$, or $0 \leq (\phi(i)-1)/2 \leq (p-3)/2$. ∎

Odd DCT components

Using Lemma 3 (in place of Lemma 1) and following exactly the same procedure as in the case of $p = 4k+3$, one gets as before,

$$X(\phi(j')) = \sum_{i=0}^{(p-3)/2} \sigma_c(i)\sigma_c(j')z((\phi(i)-1)/2)\cos(g^{i+j'}\pi\,/\,2p), \qquad 0 \leq j' \leq (p-3)/2. \tag{29}$$

Note that in the present case, order of $g \in A(2p)$ is $(p-1)$ and not $(p-1)/2$ as in the previous case. Since $0 \le i, j' \le (p-3)/2$, one can express $g^{i+j'}$ for $(i+j') > (p-1)/2$ as

$$g^{i+j'} = g^{(i+j') \bmod ((p-1)/2)} \cdot g^{(p-1)/2}. \tag{30}$$

As the order of g in $A(2p)$ is $(p-1)$,

$$g^{(p-1)/2} = -1 + k \cdot 2p \tag{31}$$

$(p-1)/2$ being an even number, one may subtract 1 from both sides of (30) and factor the left side to get

$$(g^{(p-1)/4}+1)(g^{(p-1)/4}-1) = 2(-1+kp) \tag{32}$$

Each factor on the left side of (32) is even. Therefore k on the right side is odd.

From (31) the cosine term in (29) can be written as:

$$\cos(g^{i+j'}\pi \,/\, 2p) = \begin{cases} \cos(g^{(i+j') \bmod ((p-1)/2)}\pi \,/\, 2p) & \text{if } i+j' < (p-1)/2 \\ \cos(g^{(i+j') \bmod ((p-1)/2)}(-1+k2p)\pi \,/\, 2p) & \text{otherwise} \end{cases}$$

Since k is odd, this gives

$$\cos(g^{(i+j')}\pi \,/\, 2p) = sign((p-1)/2 - (i+j'))\cos(g^{(i+j') \bmod ((p-1)/2)}\pi \,/\, 2p). \tag{33}$$

Equations (29)and (33) show that the odd components of the DCT when p is of the form $4k+1$ can be obtained through a $(p-1)/2$ point skew-cyclic convolution. In particular,

$$[\sigma_c(i)X(\phi(i))] = [\sigma_c(i)z((\phi(i)-1)/2)] \underline{*} [\cos(g^i\pi \,/\, 2p)], \qquad 0 \le i < (p-1)/2, \tag{34}$$

where $\underline{*}$ denotes a skew-cyclic convolution. Let $(p-1)/2 = 2^m q$, where q is odd. In Section 3 we show that this skew-cyclic convolution can be computed using a q point cyclic convolution algorithm and a 2^m point Hankel matrix product algorithm.

Even DCT components

As in the case of p of type $4k+3$, one may denote $\sum_{i=0}^{(p-3)/2} y(i)\cos((2i+1)j\pi\,/\,2p)$ by $X'(p-j)$. Replacing j by $p-(p-j)$ as before, gives

$$X'(p-j) = \sum_{i=0}^{(p-3)/2} ((-1)^i y(i))\sin((2i+1)(p-j)\pi\,/\,2p), \qquad 0 \le (p-j) < p, \;\; (p-j) \text{ odd}$$

Using Lemma 3 and Table 1 and defining $p-j = 2j'+1$, one gets

$$X'(\phi(j')) = \sum_{i=0}^{(p-3)/2} \sigma_s(i)\sigma_s(j')y((\phi(i)-1)/2)\sin(g^{i+j'}\pi\,/\,2p), \qquad 0 \le j' \le (p-3)/2. \tag{35}$$

Using (31) one can write the sine term in (35) as:

$$\sin(g^{i+j'}\pi\,/\,2p) = \begin{cases} \sin(g^{(i+j') \bmod ((p-1)/2)}\pi\,/\,2p) & \text{if } i+j' < (p-1)/2 \\ \sin(g^{(i+j') \bmod ((p-1)/2)}(-1+k2p)\pi\,/\,2p) & \text{otherwise} \end{cases}$$

But since k is odd as shown above, this gives

$$\sin(g^{(i+j')}\pi\,/\,2p) = \sin(g^{(i+j') \bmod ((p-1)/2)}\pi\,/\,2p). \tag{36}$$

The two equations (35) and (36) show that the even components of the DCT are obtained through a $(p-1)/2$ point cyclic convolution. In particular,

$$[\sigma_s(i)X'(\phi(i))] = [\sigma_s(i)(-1)^{(\phi(i)-1)/2}y((\phi(i)-1)/2)] \,*\, [\sin(g^i\pi\,/\,2p)], \qquad 0 \le i < (p-1)/2. \tag{37}$$

However, the following lemma shows that by choosing an appropriate g, one can simplify (37).

Lemma 4. Let p mod $4 = 1$ and g mod $4 = 3$. Then $\phi(\cdot)$ and $\sigma_s(\cdot)$ defined by (12) and (25) respectively, satisfy

$$\sigma_s(i)(-1)^{(\phi(i)-1)/2} \;=\; (-1)^i, \qquad \text{for all } \; 0 \le i < (p-1)/2.$$

Proof. Since g mod $4 = 3$, (g^i mod $4p$) mod 4 equals 1 when i is even and -1 when it is odd. Therefore (g^i mod $4p$) can be represented by $4t+1$ for even i and by $4t+3$ for odd i.

Using this, and the values of functions $\phi(\cdot)$ and $\sigma_s(\cdot)$ from Table 1, the lemma can easily be verified. For example, if $0 < g^i \bmod 4p < p$, then $\phi(i)$ equals $4t+1$ for even i and $4t+3$ for odd i. Thus, $\sigma_s(i)(-1)^{(\phi(i)-1)/2} = (1)(-1)^i = (-1)^i$. ∎

We have verified that when $p < 100,000$ and $p \bmod 4 = 1$, a generator g characterized in Lemma 4, (i.e., $g \bmod 4 = 3$), always exists. In all the analysis that follows, we will use such a generator.

Lemma 4 simplifies (37) to

$$[\sigma_s(i)X'(\phi(i))] = [(-1)^i y((\phi(i)-1)/2)] \; * \; [\sin(g^i\pi \,/\, 2p)], \qquad 0 \le i < (p-1)/2. \tag{38}$$

Once $X'(p-j)$ is computed, it is added with $(-1)^{j/2}x((p-1)/2)$ to obtain $X(j)$. In Section 3 we show that these additions for all the even j's $0 < j < p$ are accomplished by only one addition.

3 Reducing Computational Complexity.

Equation (34) shows that the odd components of DCT of length p when p is of type $4k+1$ are obtained through a skew cyclic convolution of length $2^m q$ for $m \ge 1$ and odd q. We show in this section that this computation can be transformed into a computation involving a q point cyclic convolution and a 2^m point Hankel product.

In the latter part of this section we show that a simple modification to the cyclic convolution algorithm (without altering its bilinear nature) allows one to obtain $X(0)$ in only one extra addition. Further, one more addition completes the computation of each $X(2j)$ by adding $(-1)^{j/2}x((p-1)/2)$ to each convolution output. These modifications exploit the fact that efficient bilinear cyclic convolution algorithms are based on multiplication of polynomials modulo a third polynomial and this product is computed by using the *Chinese Remainder Theorem* [9–11].

Converting a skew-cyclic convolution into a cyclic convolution.

Equation (34) shows that the odd components of a p point DCT are obtained through a skew-cyclic convolution. We now show that this computation can be turned into one that uses only cyclic convolution and Hankel product algorithms. We illustrate this through an example using $p = 13$, but the procedure is quite general and can be applied to any skew-cyclic convolution.

For $p = 13$, one can choose a generator $g = 7$. Using definition (12) one gets $\phi(i) = 1, 7, 3, 5, 9, 11$ for $i = 0, 1, 2, 3, 4, 5$. Similarly, from (13), one sees that $\sigma_c(i)$ equals -1 only when $i = 3$ and equals 1 for all other values. Equation (34) can now be expressed as:

$$\begin{pmatrix} X(1) \\ X(7) \\ X(3) \\ -X(5) \\ X(9) \\ X(11) \end{pmatrix} = \begin{pmatrix} c_0 & c_1 & c_2 & c_3 & c_4 & c_5 \\ c_1 & c_2 & c_3 & c_4 & c_5 & -c_0 \\ c_2 & c_3 & c_4 & c_5 & -c_0 & -c_1 \\ c_3 & c_4 & c_5 & -c_0 & -c_1 & -c_2 \\ c_4 & c_5 & -c_0 & -c_1 & -c_2 & -c_3 \\ c_5 & -c_0 & -c_1 & -c_2 & -c_3 & -c_4 \end{pmatrix} \begin{pmatrix} z(0) \\ z(3) \\ z(1) \\ -z(2) \\ z(4) \\ z(5) \end{pmatrix}, \tag{39}$$

where $c_i = cos(7^i\pi/26)$. For a skew-cyclic convolution of length $2^m q$, we partition the columns of the matrix into groups of 2^m columns each. These groups are then permuted such that the new i-th group is the original $(2i \bmod q)$-th group. The same permutation is also applied to the rows. The computation of (39) will then take the following form.

$$\begin{pmatrix} X(1) \\ X(7) \\ X(9) \\ X(11) \\ X(3) \\ -X(5) \end{pmatrix} = \begin{pmatrix} c_0 & c_1 & c_4 & c_5 & c_2 & c_3 \\ c_1 & c_2 & c_5 & -c_0 & c_3 & c_4 \\ c_4 & c_5 & -c_2 & -c_3 & -c_0 & -c_1 \\ c_5 & -c_0 & -c_3 & -c_4 & -c_1 & -c_2 \\ c_2 & c_3 & -c_0 & -c_1 & c_4 & c_5 \\ c_3 & c_4 & -c_1 & -c_2 & c_5 & -c_0 \end{pmatrix} \begin{pmatrix} z(0) \\ z(3) \\ z(4) \\ z(5) \\ z(1) \\ -z(2) \end{pmatrix}. \tag{40}$$

Finally, we change the signs of the last $2^m \lfloor q/2 \rfloor$ rows and columns (and the corresponding X and z vectors to match). In the present case, we thus get

$$\left(\begin{array}{c} X(1) \\ X(7) \\ \hline X(9) \\ X(11) \\ \hline -X(3) \\ X(5) \end{array}\right) = \left(\begin{array}{cc|cc|cc} c_0 & c_1 & c_4 & c_5 & -c_2 & -c_3 \\ c_1 & c_2 & c_5 & -c_0 & -c_3 & -c_4 \\ \hline c_4 & c_5 & -c_2 & -c_3 & c_0 & c_1 \\ c_5 & -c_0 & -c_3 & -c_4 & c_1 & c_2 \\ \hline -c_2 & -c_3 & c_0 & c_1 & c_4 & c_5 \\ -c_3 & -c_4 & c_1 & c_2 & c_5 & -c_0 \end{array}\right) \left(\begin{array}{c} z(0) \\ z(3) \\ \hline z(4) \\ z(5) \\ \hline -z(1) \\ z(2) \end{array}\right). \tag{41}$$

The matrix in (41) partitioned as shown reveals a block cyclic structure made up of 2×2 sub-matrices each with a Hankel structure. The computation can be written as in (42), where A, B and C are 2×2 Hankel matrices and elements of $v(i)$ and $V(i)$ are appropriate sub-vectors of length 2.

$$\begin{pmatrix} V(0) \\ V(1) \\ V(2) \end{pmatrix} = \begin{pmatrix} A & B & C \\ B & C & A \\ C & A & B \end{pmatrix} \begin{pmatrix} v(0) \\ v(1) \\ v(2) \end{pmatrix}. \tag{42}$$

Since the matrix in (42) is cyclic, one can apply a 3-point cyclic convolution algorithm shown in Fig. 2(a) to vector $[v(0), v(1), v(2)]^T$. The additions in Fig. 2(a) simply translate to additions of 2-point vectors. The multiplications, on the other hand, become multiplications of 2×2 Hankel matrices and 2-point vectors; and can be implemented using the Hankel matrix multiplication algorithm shown in Fig. 2(b).

The resultant algorithm for computation (41) is shown in Fig. 3.

Computation of $X(0)$.

Recall that $X(0)$ is merely the sum of all the components of the input vector. We now show that while computing the cyclic convolution to get the even components of the DCT,

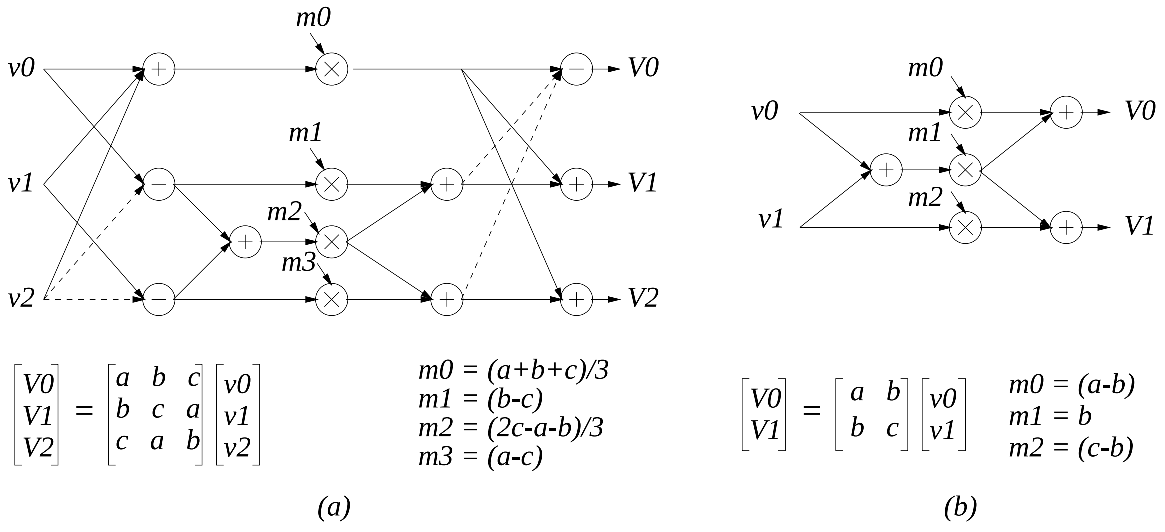

v0
v1
v2
m0
m1
m2
m3
V0
V1
V2
$\begin{bmatrix} V0 \\ V1 \\ V2 \end{bmatrix} = \begin{bmatrix} a & b & c \\ b & c & a \\ c & a & b \end{bmatrix} \begin{bmatrix} v0 \\ v1 \\ v2 \end{bmatrix}$
m0 = (a+b+c)/3
m1 = (b-c)
m2 = (2c-a-b)/3
m3 = (a-c)
(a)
v0
v1
m0
m1
m2
V0
V1
$\begin{bmatrix} V0 \\ V1 \end{bmatrix} = \begin{bmatrix} a & b \\ b & c \end{bmatrix} \begin{bmatrix} v0 \\ v1 \end{bmatrix}$
m0 = (a-b)
m1 = b
m2 = (c-b)
(b)

Figure 2: Bilinear algorithms for (a) 3-point cyclic convolution and (b) 2-point Hankel product.

one evaluates the sum of all but the $(p-1)/2$-th input vector components. Thus adding the one leftover component to this sum, one can get $X(0)$.

A t-point cyclic convolution can be viewed as a multiplication of two polynomials modulo $(s^t - 1)$. Let polynomials $a(s)$ and $b(s)$ be

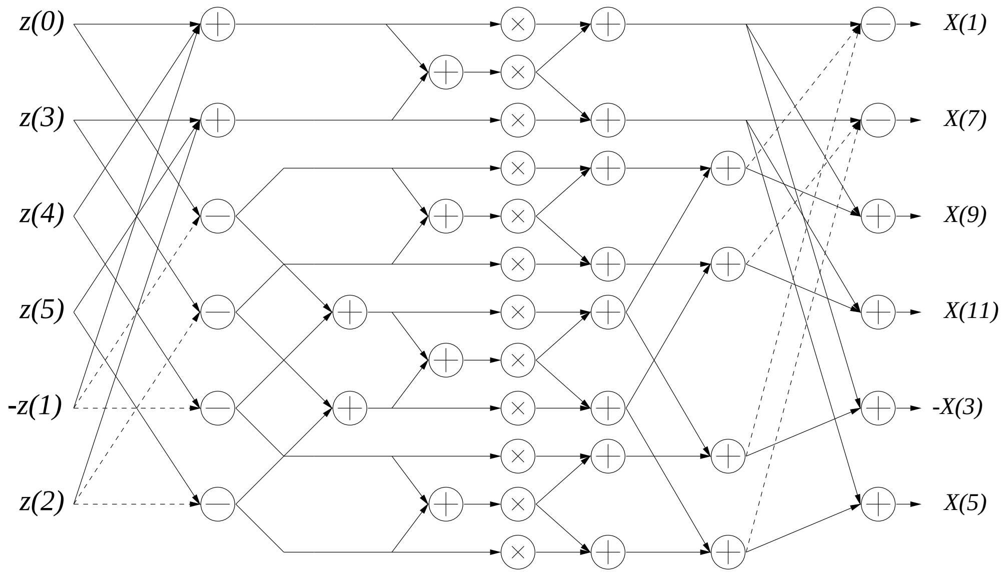
$$a(s) = \sum_{i=0}^{t-1} a_i \; s^i \quad \text{and} \quad b(s) = \sum_{i=0}^{t-1} b_i \; s^i. \tag{43}$$

Then the cyclic convolution of sequences a_i and b_i is merely the sequence made up of the coefficients of $c(s)$, where

$$c(s) = a(s) \; b(s) \bmod (s^t - 1). \tag{44}$$

Note that while computing the even DCT components through cyclic convolutions (see (27) and (38)), $t = (p-1)/2$, sequence a_i is made up of y components, sequence b_i has constant sine terms and the coefficients of $c(s)$ provide the $X'(i)$ values.

$c(s)$ is evaluated from the *Chinese Remainder Theorem* by factoring $(s^t - 1)$ in pairwise



$m_0 = (c_0 - c_1 + c_4 - c_5 - c_2 + c_3)/3$ $m_1 = (c_1 + c_5 - c_3)/3$ $m_2 = (-c_1 + c_2 - c_5 - c_0 + c_3 - c_4)/3$
$m_3 = c_4 - c_5 + c_2 - c_3$ $m_4 = c_5 + c_3$ $m_5 = -c_5 - c_0 - c_3 + c_4$
$m_6 = (-c_0 + c_1 - c_4 + c_5 - 2c_2 + 2c_3)/3$ $m_7 = (-c_1 - c_5 - 2c_3)/3$ $m_8 = (c_1 - c_2 + c_5 + c_0 + 2c_3 - 2c_4)/3$
$m_9 = c_0 - c_1 + c_2 - c_3$ $m_{10} = c_1 + c_3$ $m_{11} = -c_1 + c_2 - c_3 + c_4$
$c_i = \cos(7^i\pi/26)$

Figure 3: Computation of the odd components of a 13-point DCT by combining a 3-point cyclic convolution and 2-point Hankel product.

relatively prime factors $f_i(s)$ such that

$$s^t - 1 = \prod_i f_i(s).$$

Then $c(s)$ is given by

$$c(s) = \sum_i r_i(s)e_i(s) \mod (s^t - 1), \tag{45}$$

where,

$$r_i(s) = c(s) \mod f_i(s), \tag{46}$$

and

$$e_i(s) = \frac{(s^t - 1)}{f_i(s)} \left(\left(\frac{(s^t - 1)}{f_i(s)} \right)^{-1} \mod f_i(s) \right). \tag{47}$$

Let DCT length p be of type $4k + 3$. In this case, (27) shows that the convolution input $a_i = y((\phi(i) - 1)/2)$ and $t = (p - 1)/2$. Further, recall from (8) that $y(i) = x(i) + x(p - 1 - i)$.

Let $f_1(s) = (s-1)$, one of the simplest factors of $s^t - 1$. Then

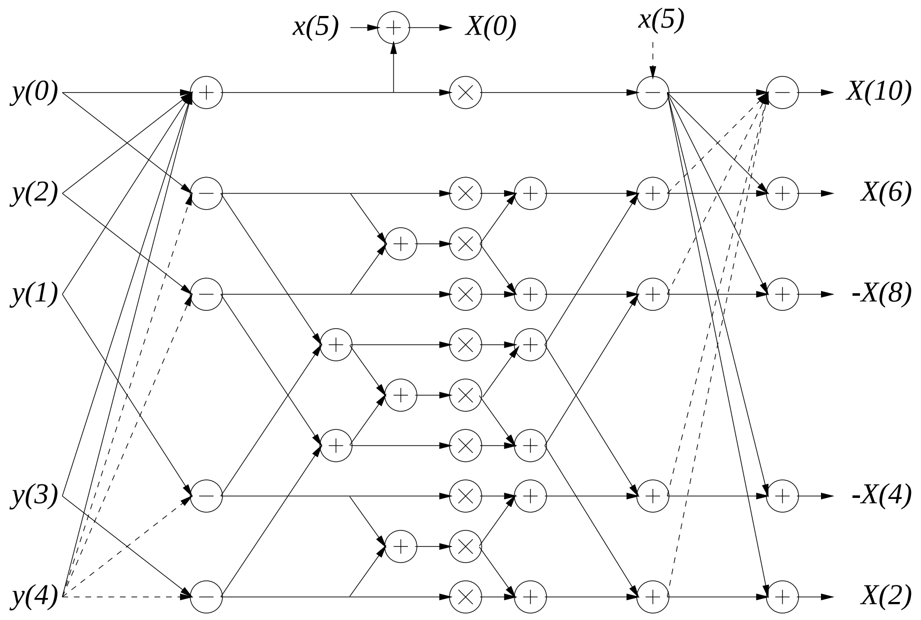
$$r_1(s) = (\sum_{i=0}^{t-1} a_i)\ (\sum_{i=0}^{t-1} b_i) \tag{48}$$

Now, $X(0)$ is given by

$$X(0) = \sum_{i=0}^{p-1} x(i) = x((p-1)/2) + \sum_{i=0}^{(p-3)/2} y((\phi(i)-1)/2) = x((p-1)/2) + \sum_{i=0}^{t-1} a(i). \tag{49}$$

While obtaining $r_1(s)$, one computes the sum of a_i's as shown by (48). $X(0)$ can be evaluated by merely adding $x((p-1)/2)$ to this sum.

Figure 4 shows the evaluation of the even components of the DCT of length $p = 11$ using a generator $g = 5$.



$m_0 = (s_0 + s_1 + s_2 + s_3 + s_4)/5$ $\quad m_1 = s_1 - s_2 - s_3 + s_4$ $\quad m_2 = s_2 - s_4$
$m_3 = -s_0 - s_2 + s_3 + s_4$ $\quad m_4 = s_3 - s_4$ $\quad m_5 = (-s_0 - s_1 - s_2 - s_3 + 4s_4)/5$
$m_6 = s_0 - s_4$ $\quad m_7 = s_0 - s_1 - s_3 + s_4$ $\quad m_8 = s_1 - s_4$
$m_9 = -s_0 - s_1 + s_2 + s_4$ $\quad s_i = \sin(5^i\pi/22)$

Figure 4: Even components of the DCT of length 11.

When DCT length p be of type $4k+1$, (38) shows that the convolution input $a_i = (-1)^i y((\phi(i)-1)/2)$ and $t = (p-1)/2$. Further, as defined in (8), $y(i) = x(i) + x(p-1-i)$.

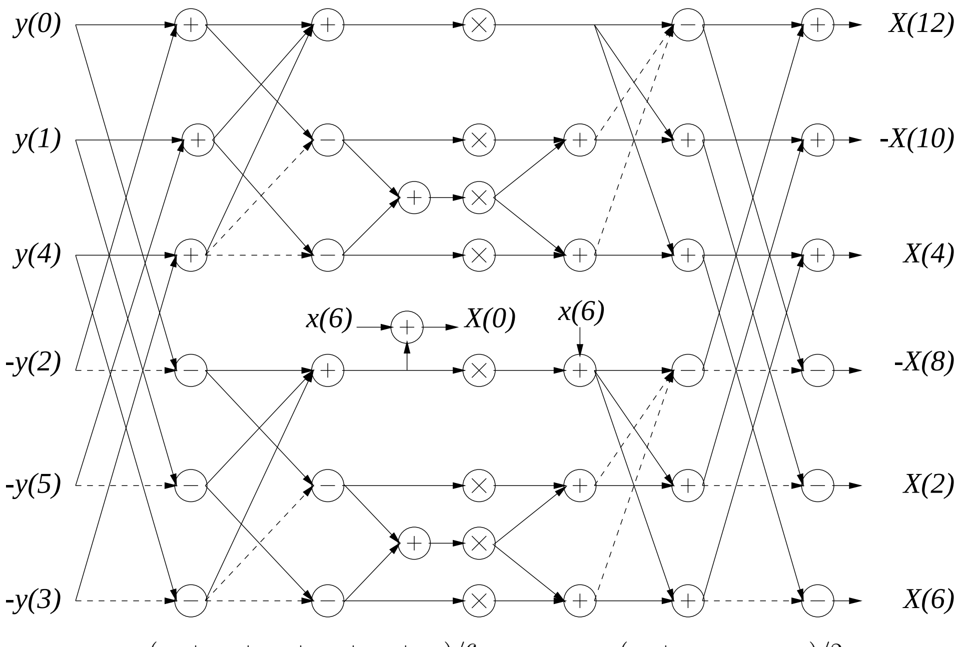
Since t is now even, $(s+1) \mid (s^t - 1)$. Let $f_1(s) = (s+1)$. Then

$$r_1(s) = (\sum_{i=0}^{t-1}(-1)^i a_i)\ (\sum_{i=0}^{t-1}(-1)^i b_i) \tag{50}$$

In this case $X(0)$ is given by

$$X(0) = \sum_{i=0}^{p-1} x(i) = x((p-1)/2) + \sum_{i=0}^{(p-3)/2} y((\phi(i)-1)/2) = x((p-1)/2) + \sum_{i=0}^{t-1}(-1)^i a(i). \tag{51}$$

Note that while obtaining $r_1(s)$, one computes the sum of $(-1)^i a_i$'s as shown by (50). $X(0)$ can be evaluated by merely adding $x((p-1)/2)$ to this sum. Figure 5 shows the evaluation of the even components of the DCT of length $p = 13$ using a generator $g = 7$.



$m_0 = (s_0 + s_1 + s_2 + s_3 + s_4 + s_5)/6$ $\quad m_1 = (s_1 + s_2 - s_3 - s_4)/2$
$m_2 = (-s_0 - s_1 - s_2 + 2s_3 + 2s_4 - s_5)/6$ $\quad m_3 = (s_0 - s_3 - s_4 + s_5)/2$
$m_4 = (s_0 - s_1 + s_2 - s_3 + s_4 - s_5)/6$ $\quad m_5 = (-s_1 + s_2 + s_3 - s_4)/2$
$m_6 = (-s_0 + s_1 - s_2 - 2s_3 + 2s_4 + s_5)/6$ $\quad m_7 = (s_0 + s_3 - s_4 - s_5)/2$
$s_i = \sin(7^i\pi/26)$

Figure 5: Even components of the DCT of length 13.

This gives $\phi(i) = 1, 7, 3, 5, 9, 11$ and $\sigma_s(i) = 1, 1, -1, -1, 1, 1$ for $i = 0, 1, 2, 3, 4$ and 5. The convolution input based on $\phi(i)$ is given by $\{y(0), -y(3), y(1), -y(2), y(4), -y(5)\}$. However, the 6-point cyclic convolution in Fig. 5 is obtained by combining a 3-point algorithm with a

2-point algorithm. This calls for permuting the input and the output to the order shown in the figure.

Adding $(-1)^{j/2}x((p-1)/2)$ to each $X'(p-j)$ for even j.

Recall from the earlier discussion that the even DCT components X and the convolution output X' are related by

$$X(p-\phi(i)) = X'(\phi(i)) + (-1)^{(p-\phi(i))/2}x((p-1)/2), \qquad 0 \le i < (p-1)/2. \tag{52}$$

Thus one may require $(p-1)/2$ extra additions beyond the convolution to get all the even components of X. We now focus on evaluating (52) with minimum complexity.

As shown by (45), the convolution result $c(s)$ is merely a sum of polynomials $e_i(s)$ weighed by corresponding $r_i(s)$. Recall that the coefficients of $c(s)$ give $X'(\phi(i))$. We will now show that one of the $r_i(s)$ in the algorithm may be altered in such a way that the coefficients of $c(s)$ give $X(p-\phi(i))$ instead of $X'(\phi(i))$.

When prime p mod $4 = 3$, one can use $f_1(s) = s-1$ as discussed earlier. In this case, $r_1(s)$ is the result of a single multiplication and is independent of s (see (48)). $e_1(s)$ can be shown to be

$$e_1(s) = (1/t)(1+s+s^2+\cdots+s^{t-1}). \tag{53}$$

To develop an efficient convolution algorithm, factor $(1/t)$ is absorbed in r_1. This is easily done by dividing the multiplication constant used in r_1 by t. However, this implies that r_1 takes a new value $r_1' = r_1/t$ thus simplifying $e_1(s)$ to $e_1'(s) = (1+s+s^2+\cdots+s^{t-1})$. (Note that since $r_1e_1(s) = r_1'e_1'(s)$, this computational convenience does not affect the convolution result.) If one subtracts $x((p-1)/2)$ from r_1', then the product $r_1'e_1'(s)$, and therefore the output $c(s)$ changes by

$$-x((p-1)/2)(1+s+s^2+\cdots+s^{t-1}).$$

Since the coefficient of s^i in $c(s)$ earlier gave $\sigma_s(i)X'(\phi(i))$, the alteration of r'_1 will transform this coefficient to

$$\sigma_s(i)X'(\phi(i)) - x((p-1)/2).$$

This term can be rewritten as

$$\begin{aligned}
& \sigma_s(i)(X'(\phi(i)) - \sigma_s(i)x((p-1)/2)) \\
= \; & \sigma_s(i)(X'(\phi(i)) - (-1)^{(\phi(i)-1)/2}x((p-1)/2)) && \text{using Lemma 2} \\
= \; & \sigma_s(i)(X'(\phi(i)) + (-1)^{(p-\phi(i))/2}x((p-1)/2)) \\
= \; & \sigma_s(i)X(p-\phi(i)) && \text{from (52)}
\end{aligned}$$

Thus through a single addition of $-x((p-1)/2)$ to r'_1, one can effectively implement all post-convolution additions and directly obtain every even DCT component. Figure 4 illustrates this modification to the convolution while computing a 11-point DCT.

Similarly when $p \bmod 4 = 1$, $f_1(s)$ can be chosen as $s+1$. In this case also, $r(s)$ is independent of s and is given by (50). $e(s)$ now evaluates to

$$e_1(s) = (1/t)(1 - s + s^2 - \cdots - s^{t-1}). \tag{54}$$

As before, the factor $(1/t)$ in $e_1(s)$ can be absorbed in r_1. If $x((p-1)/2)$ is added to $r'_1 = r_1/q$, then $r'_1 e'_1(s)$ and therefore the output $c(s)$ changes by

$$x((p-1)/2)(1 - s + s^2 - \cdots - s^{t-1}).$$

The alteration of r'_1 changes the coefficient $\sigma_s(i)X'(\phi(i))$ of s^i in $c(s)$ to a new value of

$$\sigma_s(i)X'(\phi(i)) + (-1)^i x((p-1)/2).$$

This can be rewritten as

$$\begin{aligned}
& \sigma_s(i)(X'(\phi(i)) + \sigma_s(i)(-1)^i x((p-1)/2)) \\
= \; & \sigma_s(i)(X'(\phi(i)) + (-1)^{(\phi(i)-1)/2}x((p-1)/2)) && \text{using Lemma 4} \\
= \; & \sigma_s(i)(X'(\phi(i)) + (-1)^{(p-\phi(i))/2}x((p-1)/2)) \\
= \; & \sigma_s(i)X(p-\phi(i)) && \text{from (52)}
\end{aligned}$$

Thus even in this case, a single addition of $x((p-1)/2)$ to r'_1 is equivalent to as many as $(p-1)/2$ post-convolution additions required to obtain all the even DCT components. Figure 5 illustrates this modification to the convolution while computing a 13-point DCT.

4 Complexity of the algorithms

This section evaluates the complexity of the DCT algorithm proposed in Section 2 with the modifications in Section 3.

Recall that all the even DCT components are computed by cyclically convolving a $(p-1)/2$ point sequence y with a pre-computed sequence, and then doing two more additions as suggested in Section 3. Creating y requires $(p-1)/2$ additions. Computing odd DCT components involves first getting a sequence z of length $(p-1)/2$ at the cost of $(p-1)/2$ additions. When $p \bmod 4 = 3$, z is cyclically convolved with a pre-computed sequence to obtain the required DCT components. On the other hand, when $p \bmod 4 = 1$, this cyclic convolution is replaced by an algorithm obtained by combining a 2^m point Hankel product and a q point cyclic convolution.

The cyclic convolution and Hankel product algorithms are obtained by combining appropriate algorithms from Table 2. Two cyclic convolution algorithms of relatively prime lengths may be combined to get a cyclic convolution algorithm of a larger length. A Hankel product algorithm may be combined with any cyclic convolution algorithm as needed. If the factor algorithms are bilinear, then so are the composite algorithms. To estimate the complexity of a composite algorithm, characterize a bilinear algorithm by a triple (n, a, m) made up of the length of its input vector, and its additive and multiplicative complexities. Algorithms (n_1, a_1, m_1) and (n_2, a_2, m_2) can be combined [9, 10] by partitioning the n_1n_2 point input vector into sub-vectors of lengths n_1 or n_2. The two resultant algorithms have the same functionality, but are characterized by complexities $(n_1n_2, n_1a_2 + a_1m_2, m_1m_2)$

and $(n_1n_2, n_2a_1 + a_2m_1, m_1m_2)$ respectively. Clearly, one should use a choice with a lower complexity.

To illustrate the complexity calculation, consider $p = 37$. To compute the even compo-

Table 2: Complexities of bilinear algorithms for
cyclic convolution (CC) and Hankel product (HP) [9,10,16].

Algorithm	Multiplications	Additions
2^n-point HP	3^n	$3(3^n - 2^n)$
3^n-point HP	5^n	$8(5^n - 3^n)$
5-point HP	14	27
2^n-point CC	$(3^n + 1)/2$	$(3^{n+1} + 2^{n+1} - 5)/2$
3^n-point CC	$(3 \cdot 6^n + 2)/5$	$(9 \cdot 6^n + 5 \cdot 3^n - 14)/5$
5-point CC	10	31
7-point CC	16	69
13-point CC	46	183

nents, one needs $(p-1)/2 = 18$ pre-convolution and 2 post-convolution additions. A length 18 convolution is obtained by combining cyclic convolution algorithms of length 2 and 9 with characteristics $(2, 4, 2)$ and $(9, 71, 22)$. The required cyclic convolution algorithm will then have characteristics $(18, 178, 44)$. Thus the even DCT components require 44 multiplications and 198 additions. Computation of odd components requires 18 additions to create z. The rest of the computation uses an algorithm combining a 2-point Hankel product with a length 9 cyclic convolution with characteristics $(2, 3, 3)$ and $(9, 71, 22)$ respectively. Their combination is characterized by $(18, 208, 66)$. Thus the even components of the DCT are computed using 66 multiplications and 226 additions. The total complexity of a bilinear 37-point DCT is 110 multiplications and 424 additions.

Table 3 provides the complexities of most of the primes less than 100. We have also listed a generator useful in each case.

Table 3: Prime DCT complexities and generator g.

		Multiplications			Additions		
Length	g	Odd	Even	Total	Odd	Even	Total
5	3	3	2	5	5	8	13
7	9	4	4	8	14	16	30
11	5	10	10	20	36	38	74
13	7	12	8	20	40	42	82
17	3	27	14	41	65	56	121
19	5	22	22	44	80	82	162
29	3	32	48	80	200	182	382
31	9	40	40	80	194	196	390
37	15	66	44	110	226	198	424
41	7	90	50	140	294	252	501
53	3	138	92	230	530	446	976
61	7	120	80	200	508	450	958
71	9	160	160	320	876	878	1754
73	11	198	110	308	650	528	1178
79	5	184	184	368	914	916	1830
97	7	324	164	488	1004	766	1770

Wagh and Ganesh have previously given a DCT algorithm that converts the DCT computation into cyclic convolutions [12]. Even though they employ the same number of multiplications as our algorithm[2], their number of additions tends to be quite high. For example, a 31 point DCT needs 422 additions in [12] compared with our 390 additions.

Table 4: Comparison of the computational complexity of the new algorithms with that of Chan and Siu [13] and Yin and Siu [15]. (The computational blocks involved, cyclic convolution and Hankel product of i points, are denoted by C_i and H_i respectively.)

DCT length	Algorithm	Even transform components	Odd transform components	Multiplications	Additionss
13	[13]	C_6	C_6	34	133
	[15]	C_6	H_6	23	105
	New	C_6	$C_3 \times H_2$	20	82
19	[13]	C_9	C_9	71	240
	[15]	C_9	H_9	47	235
	New	C_9	C_9	44	162
31	[13]	C_{15}	C_{15}	125	522
	[15]	C_{15}	H_{15}	124	491
	New	C_{15}	C_{15}	80	390
37	[13]	C_{18}	C_{18}	142	553
	[15]	C_{18}	H_{18}	119	581
	New	C_{18}	$C_9 \times H_2$	110	424

Table 4 compares the proposed algorithms with two of the more recent algorithms, namely those by Chan and Siu [13] and by Yin and Siu [15]. This table also shows the blocks employed in the computation of both the even and the odd transform components. Note that even though [13] uses only cyclic convolutions, an extra $3(p-1)/2$ multiplications and

[2]They use one extra multiplication to compute $X(0)$ because of their definition of DCT.

$11(p-1)/2-1$ additions are required to express the computation into cyclic convolutions. Our complexity is substantially lower than [15] also, because of two reasons. Firstly, when $(p-1)/2$ is a composite number, [15] has to use a skew-cyclic convolution which is typically computed as a Hankel product requiring large complexity. Instead, we use a cyclic convolution (e.g., $p = 31$) or a cyclic convolution combined with a small Hankel product (e.g., $p = 37$). Secondly, in all cases, we can obtain $X(0)$ and accomplish the addition (or subtraction) of $x((p-1)/2)$ to all even transform components at an extra cost of a mere 2 additions. On the other hand, for arbitrary lengths p, [15] has to use $(p-1)/2$ additions to compute $X(0)$ and another $(p-1)/2$ additions to add (or subtract) $x((p-1)/2)$.

The bilinear character of our algorithm lends itself to high speed hardware implementations. The independendance of the multiplications involved implies that they may be done concurrently. Thus the critical path delay of the hardware implementation of our algorithm is determined by a single multiplier and a few adders. For example, from Fig.s 1, 3, 4, and 5, one can see that 11 or 13 point DCTs can be completed within the time required for 8 additions and 1 multiplication. Further, the 8 additions along the critical path are grouped into two sets of four consecutive additions. In fixed point number system, delay of four consecutive n bit additions equals the delay of one n bit adder and that of three 1-bit adders. Thus the hardware realizations of these algorithms is expected to be very fast. In contrast, DCT hardware implemented as a systolic array [17–19] requires at least $(p-1)/2$ clock cycles for a p point DCT, where the clock period is at least equal to the delay of 1 multiplication and 1 addition.

5 Conclusion

This paper presents a complete methodology of deriving bilinear algorithms for prime length discrete cosine transforms. We show that the use of appropriate multiplicative groups of integers allows one to identify cyclic structures within a DCT matrix.

We show that a p point (with prime p) DCT is equivalent to two $(p-1)/2$ point cyclic convolutions whenever $(p-1)/2$ is an odd integer. This is a substantial improvement over [15] where the cyclic convolutions are guaranteed only if $(p-1)/2$ is a prime. Further, for other primes, we show that the skew-cyclic convolution involved can be computed using a multidimensional technique from a smaller cyclic convolution and a Hankel matrix product algorithm.

We also show that the 0-th component of the DCT can be computed with just one extra addition. Further, all the post-additions required to obtain the other even DCT components can be accomplished through a single addition. Both these enhancements result from small modifications to the cyclic convolution algorithm. It should be mentioned here that [15] had observed this for the specific case of a 7-point DCT that uses a cyclic convolution of length 3. We show that these enhancements are possible whenever the cyclic convolution algorithms are based on Chinese Remainder Theorem and prove them for an *arbitrary* prime length. The paper gives complete bilinear algorithms for DCT lengths 11 and 13, illustrating all aspects of the method.

The bilinearity of the algorithms obtained here implies that all the multiplications involved in the computation are independent of each other and can be performed concurrently. Because of this concurrency and their low complexity, these DCT algorithms are well suited for software implementations as well as dedicated hardware solutions using application specific integrated circuits (ASICs) or FPGAs.

References

[1] K. R. Rao, P. Yip, Discrete Cosine transform: Algorithms, Advantages, Applications, Academic Press, Boston, 1990.

[2] G. Mandyam, N. Ahmed, N. Magotra, Lossless image compression using the discrete cosine transform, J. Visual Comm. and Image Representation 8 (1) (1997) 21–26.

[3] Z. M. Hafed, M. D. Levine, Face recognition using the discrete cosine transform, Int. J. of Computer Vision 43 (3) (2001) 167–188.

[4] J. W. Cooley, J. W. Tukey, An algorithm for the machine computation of complex Fourier series, Math. Comput. 19 (1965) 297–301.

[5] G. Bi, L. W. Yu, DCT algorithms for composite sequence lengths, IEEE Trans. on Signal Processing 46 (3) (1998) 554–562.

[6] E. Feig, S. Winograd, Fast algorithms for the discrete cosine transform, IEEE Trans. on Signal Processing 40 (9) (1992) 2174–2193.

[7] Z. Cvetkovic, M. V. Popovic, New fast recursive algorithms for the computation of discrete cosine and sine transforms, IEEE Trans. Signal Process. SP-40 (8) (1992) 2083–2086.

[8] C. M. Rader, Discrete Fourier transforms when the number of data samples is prime, Proc. IEEE 56 (1968) 104–105.

[9] R. Agarwal, J. Cooley, New algorithms for digital convolution, IEEE Trans. Acoust., Speech, Signal Proc. 25 (1977) 392–410.

[10] S. Winograd, On computing the discrete Fourier transform, Math. Comp. 32 (141) (1978) 175–199.

[11] I. Selesnick, C. Burrus, Extending Winograd's small convolution algorithm to larger lengths, in: Proc. IEEE Intl. Symposium on Circuits and Systems, London, 1994, pp. 449–452.

[12] M. D. Wagh, H. Ganesh, A new algorithm for the discrete cosine transform of arbitrary number of points, IEEE Trans. on Computers C-29 (4) (1980) 269–277.

[13] Y. Chan, W. Siu, Algorithm for prime length discrete cosine transforms., Electronic Letters 26 (1990) 206–208.

[14] Y. H. Chan, W. C. Siu, Generalized approach for the realization of discrete cosine transform using cyclic convolutions, in: Intl. Conf. on Acoustics, Speech and Signal Processing, Vol. 3, 1993, pp. 277 – 280.

[15] R. Yin, W. Siu, A new fast algorithm for computing prime-length DCT through cyclic convolutions, Signal Processing 81 (5) (2001) 895–906.

[16] M. D. Wagh, Modular algorithms for cyclic convolutions, unpublished manuscript.

[17] J. Guo, C. M. Liu, C. W. Jen, A new array architecture for prime length discrete cosine transform, IEEE Trans. on Signal Processing 41 (1) (1993) 436–442.

[18] D. F. Chiper, Novel systolic array design for discrete cosine transform with high throughput rate, in: Proc. IEEE Int. Symp. Circuits and Systems, Vol. 2, 1996, pp. 746–749.

[19] C. Cheng, K. K. Parhi, A novel systolic array structure for DCT, IEEE Trans. on Circuits and Systems - II: Express Briefs 52 (7) (2005) 366–369.