Hardware-Efficient Realization of Prime-Length DCT Based on Distributed Arithmetic

Jiafeng Xie, Pramod Kumar Meher, Senior Member, IEEE, and Jianjun He

Abstract—This paper presents an efficient decomposition scheme for hardware-efficient realization of discrete cosine transform (DCT) based on distributed arithmetic (DA). We have proposed an efficient design for the implementation of cyclic convolution based on group distributed arithmetic (GDA) technique where the ROM size could be reduced over the existing GDA-based design. The proposed structure for DCT implementation, based on the new decomposition scheme and proposed design of GDA-based cyclic convolution, involves significantly less area complexity than the existing one. For example, to implement the DCT of transform length \( N = 17 \), the proposed design needs a lookup table of 128 words, while the existing design for \( N = 16 \) requires a lookup table of 256 words. From the synthesis results it is found that proposed design involves significantly less area, gives higher throughput, and consumes less power compared to the existing designs of nearly the same or lower lengths.

Index Terms—Distributed arithmetic (DA), cyclic convolution, discrete cosine transform (DCT), hardware-efficient.

1 INTRODUCTION

TRANSFORM computation is frequently encountered in various signal and image processing applications [1-3]. Therefore, many efforts have been made on this area. Some of those are on development of the fast algorithm and VLSI realization [4, 5], while some others focus on the automatic tools for fast generation of hardware processors [6]. The discrete cosine transform (DCT) is one such transform which is widely explored. Not only is it used in video processing, but also it is used in other applications, e.g., realization of a bandpass filter-bank for conversion between time-division and frequency-division voice multiplexing [7]. Several algorithms and architectures are suggested over the time for efficient computation of DCT. Some of those architectures are designed for power-of-two length DCT [3, 8-11], while some others are for prime-length [7, 12-14]. Although for video processing applications, power-of-two length DCT is used, we find that compared to power-of-two length DCT, prime-length DCT has scope to be implemented more efficiently by hardware. Besides, for some applications like voice multiplexing system, power-of-two length DCT would not be preferred because of limited choices and wide difference between two successive choices of transform length. Since the transform size in voice multiplexing system is dictated by the sampling frequencies, power-of-two length DCT is not suitable for this application [7]. Current video coding standards do not use prime-length DCT, but for exploiting the scope for more efficient hardware implementation of prime length DCTs, they could be used for non-standard video compressions, e.g., in personal storage systems, and digital video cameras etc.. Moreover, as different standards are evolving from time to time, considering the significant benefit of more efficient hardware implementation, prime-length DCT could be potential candidate to be included in future standards. To handle such situations, several hardware architectures for prime-length DCT have been suggested in the past years. Prime-factor decomposition approach and hardware solution for the implementation of DCT is presented in [7]. A new fast algorithm for prime-length DCT is proposed in [12]. In [13], the authors present a novel systolic structure for prime-length DCT. In this paper, we present an efficient DCT algorithm and its mapping to hardware architecture, where the transform-length is a prime number.

Distributed arithmetic (DA)-based technique is one of the most popular and hardware-efficient approaches for DCT implementation [8]. DA was introduced by Croisier [15] in 1973, and it was used by Peled and Liu [16] for the implementation of digital filters. The main hardware component of DA-based implementation is a look up table (LUT) which stores the precomputed results of partial inner-products to be read out when bit-slices of input vector are fed as address of the LUT. In past decades, a number of work have been explored on efficient implementation of DCT using DA [3, 8-11, 14]. The major disadvantage of DA approach is that the LUT size increases exponentially with the transform-length, since it requires a read only memory (ROM) of \( O(2^N) \) words, where \( N \) is the transform length. Some of the DA-based DCT architectures are mainly for the reduction of power-consumption while some others are for ROM-size-reduction. In [3], the authors present a low-power structure for two-dimensional (2-D) DCT. In [8], a hardware efficient structure for DA-based DCT realization is proposed. In [9], the authors present an efficient low-power design for DA-based DCT implementation. In [10], the
authors have suggested another low-power design for DCT based on New DA (NEDA) or adder-based DA. In [14], a group distributed arithmetic (GDA) technique is suggested for efficient realization of cyclic convolution and applied that to DCT, which is more efficient compared with the conventional design. A DA-based systolic-like design for DCT is proposed in [11], where the transform-length \( N=4L \). (\( L \) is any integer), and more efficient compared with the designs of [13] and [14].

In this paper, we aim at presenting a more efficient hardware implementation of cyclic convolution based on DA and use that for DCT, where the transform-length \( N \) is a prime number of the form \( N=4L+1 \). Besides, we present here an efficient decomposition algorithm for the DCT. The main novelty of this paper is that an \( N \)-point DCT is converted into an \((N-1=4L)\)-point cyclic convolution, and the cyclic convolution is then decomposed into a pair of 2x2 block convolution with circulant matrices* of size \( 1xL \). Block convolutions are then computed efficiently using the fast algorithm of [17], so that the \( N \)-point DCT is computed finally by four \( L \)-point cyclic convolutions and a pair of vector-matrix multiplication with a sparse matrices of size \( 1xL \). Note that the \((4L)\)-point cyclic convolution has nearly 16 times arithmetic complexity of \( L \)-point cyclic convolution. Therefore, the proposed approach of decomposition provides significant saving of computation. We have also shown that the cyclic convolution can be implemented by the proposed modified GDA approach efficiently, where the ROM size is significantly reduced. The multiplications with sparse matrices can be implemented by traditional DA approach since they have only a few non-zero elements. The whole structure for two-dimensional (2-D) DCT involves significantly less area-time-power complexity than the existing designs.

The rest of this paper is organized as follows: The proposed algorithm for DCT is derived in Section 2. In Section 3, we present the proposed design for cyclic convolution and the DCT architecture. The comparison of proposed architecture with corresponding existing architectures are discussed in Section 4. Conclusion is presented in Section 5.

2 Algorithm

2.1 DCT

The DCT of length \( N \) can be defined as

\[
Y(k) = \sum_{i=0}^{N-1} y(i) \cos \left( \frac{2\pi(2i + 1)k}{4N} \right)
\]

for \( k=0, 1, ..., N-1 \), where \( y(i) \) and \( Y(k) \) are the input samples and DCT output, respectively. To derive the proposed circular matrix, the following identity can be used:

\[
\cos[(2n + 1)x] = \left[ (-1)^n + 2 \sum_{i=1}^{n} (-1)^{n-i} \cos(2xi) \right] \cos x
\]

Using (2), one can express (1) in a form given by

\[
Y(0) = \sum_{i=0}^{N-1} y(i)
\]

\[
Y(k) = \{2T(k) + x(0)\} \cos \left( \frac{k\pi}{2N} \right)
\]

where

\[
T(k) = \sum_{i=1}^{N-1} x(i) \cos \left( \frac{\pi i k}{N} \right), \quad k=0, 1, ..., N-1.
\]

*The multiplication of a vector by a circulant matrix generated by another vector results the cyclic convolution of the pair of sequences generated by the vectors.*
Then from (5) we can have
\[ A_n = \begin{bmatrix} a_1 & a_2 & \cdots & a_{n-1} & a_n \\ a_n & a_1 & \cdots & a_{n-2} & a_{n-1} \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ a_3 & a_4 & \cdots & a_1 & a_2 \\ a_2 & a_3 & \cdots & a_n & a_1 \end{bmatrix} \] \hspace{1cm} (7a)

where \( n = (N-1)/2 \).

If \( (N-1) = 4L \), where \( L \) is any positive integer, we can write the \( A_{(N-1)/2} \) as
\[ A_{(N-1)/2} = \begin{bmatrix} F_1 & F_2 \\ F_2 & F_1 \end{bmatrix} \] \hspace{1cm} (7b)

where \( F_1 \) and \( F_2 \) are both \([ (N-1)/4 ] \times [ (N-1)/4 ] \) matrices:
\[ F_1 = \begin{bmatrix} a_1 & \cdots & a_n \\ \vdots & \ddots & \vdots \\ a_n & \cdots & a_1 \end{bmatrix}, \quad F_2 = \begin{bmatrix} a_{n+1} & \cdots & a_n \\ \vdots & \ddots & \vdots \\ a_2 & \cdots & a_1 \end{bmatrix} \] \hspace{1cm} (7c)

Example: We illustrate here the transformation of DCT of \( N=7 \) into cyclic convolution form, we find \( g=3 \) from (4). Then from (5) we can have
\[
\begin{align*}
T(3) & = \begin{bmatrix} C3 & C1 & C2 & -C3 & -C1 & -C2 \\ T(2) & = \begin{bmatrix} C2 & -C3 & -C1 & C2 & -C3 & -C1 \\ T(6) & = \begin{bmatrix} -C1 & C2 & -C3 & -C1 & C2 & -C3 \\ T(4) & = \begin{bmatrix} -C3 & -C1 & C2 & -C3 & -C1 & C2 \\ T(5) & = \begin{bmatrix} -C2 & C3 & -C1 & C2 & -C3 & C1 \\ T(1) & = \begin{bmatrix} C1 & -C2 & C3 & -C1 & C2 & -C3 \\ x[1] & x(5) & x(4) & x(6) & x(2) & x(3) \end{bmatrix}^T
\end{align*}
\]

where \( C_i = \cos(\pi i/N) \), and \( C_i = -C(N-i) \).

To arrive at the desired cyclic convolution form, we write (8) as
\[
\begin{bmatrix} T_0 \\ T_1 \end{bmatrix} = \begin{bmatrix} A_3 & 0 \\ 0 & A_3 \end{bmatrix} \begin{bmatrix} R_3 (X_0 + X_1) \end{bmatrix} \] \hspace{1cm} (9a)

where
\[
\begin{align*}
A_3 & = \begin{bmatrix} -C3 & -C1 & C2 \\ C2 & -C3 & -C1 \\ -C1 & C2 & -C3 \end{bmatrix} \\
R_3 & = \begin{bmatrix} -1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \end{align*}
\] \hspace{1cm} (9b)

Thus, the DCT for \( N=7 \) is transformed into a pair of three-point cyclic convolutions with exactly the same structure \( A_3 \). The reordering of \( T(k) \) and \( x(i) \) facilitates the derivation of the proposed algorithm, as shown in following equations. For other prime transform lengths also, we can have similar reordering of \( T(k) \) and \( x(i) \) as shown in (8-9).

To reduce the complexity of (7b), we can use the 2-point cyclic convolution algorithm of [17] for computing the two point block cyclic convolution
\[
\begin{bmatrix} U_1 \\ U_2 \end{bmatrix} = \begin{bmatrix} F_1 & F_2 \\ F_2 & F_1 \end{bmatrix} \begin{bmatrix} V_1 \\ V_2 \end{bmatrix} \] \hspace{1cm} (10a)

by the following steps, where \( [U_1, U_2, V_1, V_2, F_1, F_2] \) are square matrices of equal sizes.

1) Pre-computation of \( E_1 \) and \( E_2 \):
\[ E_1 = (F_1 + F_2)/2, \quad E_2 = (F_1 - F_2)/2 \] \hspace{1cm} (10b)

2) Input additions:
\[ S_1 = (V_1 + V_2), \quad S_2 = (V_1 - V_2) \] \hspace{1cm} (10c)

3) Matrix-vector product:
\[ W_1 = E_1 \times S_1, \quad W_2 = E_2 \times S_2 \] \hspace{1cm} (10d)

4) Output additions:
\[ U_1 = (W_1 + W_2), \quad U_2 = (W_1 - W_2) \] \hspace{1cm} (10e)

where the \( F_1 \) and \( F_2 \) are defined by (7b), and \( E_1 \) and \( E_2 \) can be seen from (10f) and (10g)—at the top of next page, where it can be seen that \( E_1 \) is a circulant matrix, while \( E_2 \) is not.

To arrive at (6) at the top of next page, we can have \( A_3 \) and \( F_2 \) as
\[
\begin{align*}
T_0 & = \begin{bmatrix} T(4) & T(2) & T(6) \end{bmatrix}^T \\ T_1 & = \begin{bmatrix} T(3) & T(5) & T(1) \end{bmatrix}^T \\ X_0 & = \begin{bmatrix} x(1) & x(5) & x(4) \end{bmatrix}^T \\ X_1 & = \begin{bmatrix} x(6) & x(2) & x(3) \end{bmatrix}^T
\end{align*}
\] \hspace{1cm} (9d)

Thus, the former \([ (N-1)/2 ] \times [ (N-1)/2 ] \) size matrix of (7a) has been decomposed into smaller matrices, which facilitates the use of proposed DA-based approach.

For clarity of presentation, we derive here the matrix \( E_1 \) and \( E_2 \) for \( N = 13 \). Using \( C_i = \cos(\pi i/N) \) and \( C_i = -C(N-i) \), we can have \( F_1 \) and \( F_2 \) as
Then using the short cyclic convolution algorithm of [17] on (11), we can have \( E_1 \) and \( E_2 \) as shown in (12).

\[
E_1 = \frac{1}{2} \begin{bmatrix}
    a_1 + a_{n+1}^2 & a_2 + a_{n+2}^2 & a_3 + a_{n+3}^2 & \cdots & a_n + a_{2n}^2 \\
    a_n + a_{n+1}^2 & a_1 + a_{n+2}^2 & a_2 + a_{n+3}^2 & \cdots & a_{n-1} + a_{2n}^2 \\
    \vdots & \vdots & \ddots & \ddots & \vdots \\
    \vdots & \vdots & \cdots & a_1 + a_{n+2}^2 & a_2 + a_{n+3}^2 \\
    a_2 + a_{n+2}^2 & a_3 + a_{n+3}^2 & \cdots & a_n + a_{2n}^2 & a_1 + a_{2n+1}^2 \\
\end{bmatrix}
\]  
(10f)

\[
E_2 = \frac{1}{2} \begin{bmatrix}
    a_1 - a_{n+1}^2 & a_2 - a_{n+2}^2 & a_3 - a_{n+3}^2 & \cdots & a_n - a_{2n}^2 \\
    a_n - a_{n+1}^2 & a_1 - a_{n+2}^2 & a_2 - a_{n+3}^2 & \cdots & a_{n-1} - a_{2n}^2 \\
    \vdots & \vdots & \ddots & \ddots & \vdots \\
    \vdots & \vdots & \cdots & a_1 - a_{n+2}^2 & a_2 - a_{n+3}^2 \\
    a_2 - a_{n+2}^2 & a_3 - a_{n+3}^2 & \cdots & a_n - a_{2n}^2 & a_1 - a_{2n+1}^2 \\
\end{bmatrix}
\]  
(10g)

It is observed that \( E_1 \) is a circulant matrix, while \( E_2 \) is not. Here, we have proposed a new decomposition scheme of matrix \( E_2 \) as

\[
E_2 = \frac{1}{2} (G_1 + G_2)
\]

\[
= \frac{1}{2} \begin{bmatrix}
    C_2 + C3 & -C1 + C5 & C6 + C4 \\
    C4 + C6 & C2 + C3 & -C1 - C5 \\
    -C1 - C5 & C4 + C6 & C2 - C3 \\
\end{bmatrix}
\]  
(12a)

\[
= \frac{1}{2} \begin{bmatrix}
    C2 + C3 & -C1 + C5 & C4 - C6 \\
    C4 - C6 & C2 + C3 & -C1 + C5 \\
    -C1 + C5 & C4 - C6 & C2 + C3 \\
\end{bmatrix}
\]  
(12b)

It is seen that 3×3 matrix \( E_2 \) of (12) can be decomposed into two matrices: \( G_1 \) and \( G_2 \), where \( G_1 \) is a circulant matrix, which can be implemented by cyclic convolution, and \( G_2 \) is a sparse matrix and each row of it can be implemented by conventional DA-approach separately. Note that the direct implementation of (12) by the conventional DA-based design, the realization of \( E_1 \) and \( E_2 \) require 48 words to be stored in the ROM. But the decomposition of (13) can reduce the number of words to be stored in the ROM to 26.

We consider another example where \( E_2 \) is a 4×4 matrix of (14a). For simplicity of discussion, the scale factor 1/2 has been ignored in (14a). Using similar decomposition approach, we can have \( G_1 \) and \( G_2 \) as shown in (14b) and (14c), respectively.

\[
E_2 = \begin{bmatrix}
    t1 & t2 & t3 & t4 \\
    -t4 & t1 & t2 & t3 \\
    -t3 & -t4 & t1 & t2 \\
    -t2 & -t3 & -t4 & t1 \\
\end{bmatrix}
\]  
(14a)

\[
G_1 = \begin{bmatrix}
    t1 & t2 & -t3 & -t4 \\
    -t4 & t1 & t2 & -t3 \\
    -t3 & -t4 & t1 & t2 \\
    t2 & -t3 & -t4 & t1 \\
\end{bmatrix}
\]  
(14b)

\[
G_2 = \begin{bmatrix}
    0 & 0 & 2t3 & 2t4 \\
    0 & 0 & 0 & 2t3 \\
    0 & 0 & 0 & 0 \\
    -2t2 & 0 & 0 & 0 \\
\end{bmatrix}
\]  
(14c)

When \( E_2 \) is a 5×5 matrix, we can have a similar decom-
position:

\[
E_2 = \begin{bmatrix}
  t_1 & t_2 & t_3 & t_4 & t_5 \\
  -t_5 & t_1 & t_2 & t_3 & t_4 \\
  -t_4 & -t_5 & t_1 & t_2 & t_3 \\
  -t_3 & -t_4 & -t_5 & t_1 & t_2 \\
  -t_2 & -t_3 & -t_4 & -t_5 & t_1 \\
\end{bmatrix}
\]  

(15a)

\[
G_1 = \begin{bmatrix}
  t_1 & t_2 & t_3 & -t_4 & -t_5 \\
  -t_5 & t_1 & t_2 & t_3 & -t_4 \\
  -t_4 & -t_5 & t_1 & t_2 & t_3 \\
  t_3 & -t_4 & -t_5 & t_1 & t_2 \\
  t_2 & t_3 & -t_4 & -t_5 & t_1 \\
\end{bmatrix}
\]  

(15b)

\[
G_2 = \begin{bmatrix}
  0 & 0 & 0 & 2t_4 & 2t_5 \\
  0 & 0 & 0 & 0 & 2t_4 \\
  0 & 0 & 0 & 0 & 0 \\
  -2t_3 & 0 & 0 & 0 & 0 \\
  -2t_2 & -2t_3 & 0 & 0 & 0 \\
\end{bmatrix}
\]  

(15c)

It should be noted that for \( E_2 \) matrix of size \( d \times d \) the sparse matrix contains \((d^4/4)\) terms when \( d \) is even or \((d^4+1)(d-1)/4\) terms when \( d \) is odd.

For a prime number \( N \) of the form \( N = 4L + 1 \), where \( L \) is any positive integer, the proposed algorithm is described in general form in the following.

1) Represent the \( N \)-point DCT in terms of \((N-1)\)-point cyclic convolution.

2) Decompose the cyclic convolution to the form of (6) in terms of pair of matrix-vector products with a matrix, \( A_{(N-1)/2} \) of size \([(N-1)/2] \times [(N-1)/2] \).

3) Transform the matrix of \( A_{(N-1)/2} \) into the block cyclic convolution form

\[
A_{(N-1)/2} = \begin{bmatrix}
  F_{1,(N-1)/4} & F_{2,(N-1)/4} \\
  F_{2,(N-1)/4} & F_{1,(N-1)/4} \\
\end{bmatrix}
\]  

(16)

where \( F_{1,(N-1)/4} \) and \( F_{2,(N-1)/4} \) are both \([(N-1)/4] \times [(N-1)/4] \) matrices.

4) Compute the block cyclic convolution of (16) using the fast cyclic convolution algorithm of [17], which requires to compute the matrix-vector product with the following pair of matrices:

\[
E_{1,(N-1)/4} = \frac{1}{2}(F_{1,(N-1)/4} + F_{2,(N-1)/4})
\]  

(17)

and

\[
E_{2,(N-1)/4} = \frac{1}{2}(F_{1,(N-1)/4} - F_{2,(N-1)/4})
\]  

(18)

5) Using the decomposition approach presented in (13), (14) and (15), express the matrix \( E_2 \) by a circulant matrix and a sparse matrix.

6) Implement (17) and (18) by the structure presented in Section 3 after the decomposition. Note that the computation of \( E_1 \) and \( E_2 \) can proceed in parallel. Perform the input addition and output additions of the proposed algorithm in parallel (discussed in Section 3).

7) Compute the DCT output using (2b). This step requires addition and multiplication operations which all the convolution-based prime-length DCT algorithms need to perform without any variation. Therefore, we just skip this step.

Note that the step 1) to 4) of this algorithm is similar to that in [13], but our approach extends further to step 5) and 7).

### 2.2 Inverse DCT (IDCT)

The inverse DCT (IDCT) can also be computed by an algorithm similar to that of forward DCT, as discussed below.

The basic purpose of using IDCT is to recover back the time-domain sequence from the coefficients generated by the forward DCT. Firstly, we need to transform the prime-length IDCT computation into the cyclic convolution form. Then the circulant matrix is transformed into 2×2 block cyclic convolution of two matrices of nearly half sizes, where one of them is a circulant matrix while the other is not. After that, the non-circulant matrix is further decomposed into a circulant matrix plus a sparse matrix. These circulant matrices facilitate the use of proposed DA-based approach with significantly less ROM words.

The IDCT is defined as

\[
y(i) = \sum_{k=0}^{N-1} Y(k) \cos[(2i + 1)k\alpha] 
\]  

(19)

where \( \alpha = \pi / 2N \). Following the algorithm suggested in [19], we can have

\[
y(0) = Y(0) + P(0) + Q(0)
\]  

(20a)

\[
y(i) = Y(i) + P(i) + (-1)^i \cdot Q(i)
\]  

(20b)

\[
y(N - i) = Y(i) + P(i - 1) + (-1)^i \cdot Q(i - 1)
\]  

(20c)

for \( i = 1, \ldots, (N-1)/2 \), and

\[
P(0) = \sum_{j=0}^{(N-1)/2} Y_j(\phi(i))
\]  

(21a)

\[
P(i) = 2 \cdot T_n(i) - P(i - 1)
\]  

(22b)

\[
Q(0) = \sum_{i=1}^{(N-1)/2} Y_i(\phi(i))
\]  

(22c)

\[
Q(i) = 2 \cdot T_n(i) + Q(i - 1)
\]  

(22d)
where $g$ is the same as that in (4c). The sequences form as circular correlation structures of the same length and

$$
Y_c(k) = Y(\zeta(k)) \cdot \cos(2k\alpha), k = 1, \ldots, (N-1)/2
$$

$$
Y_s(k) = Y(\zeta(k)) \cdot \sin(2k\alpha), k = (N-1)/2, \ldots, N-1
$$

$$
\zeta(k) = \langle 2k \rangle_N
$$

Then we define

$$
\varphi(k) = \begin{cases} 
\langle g^k \rangle_N & \text{if } \langle g^k \rangle_N \leq (N-1)/2 \\
\langle g^{k+(N-1)/2} \rangle_N & \text{otherwise}
\end{cases}
$$

$$
\phi(k) = \begin{cases} 
\langle g^k \rangle_N & \text{if } \langle g^k \rangle_N > (N-1)/2 \\
\langle g^{k+(N-1)/2} \rangle_N & \text{otherwise}
\end{cases}
$$

(24a, 24b)

where $g$ is the same as that in (4c). The sequences $T_c(i)$ and $T_d(i)$ can be efficiently computed in parallel using two circular correlation structures of the same length and form as

$$
T_c(\varphi(k)) = \sum_{i=1}^{(N-1)/2} Y_c(\varphi(i)) \cdot \cos(\varphi(i + k) \cdot 4\alpha)
$$

(25a)

$$
T_d(\varphi(k)) = \sum_{i=1}^{(N-1)/2} Y_s(\phi(i)) \cdot \cos(\phi(i + k) \cdot 4\alpha)
$$

(25b)

Alternatively, (20) can be expressed as

$$
\begin{bmatrix}
T_c \\
T_d
\end{bmatrix} =
\begin{bmatrix}
B & 0 \\
0 & B
\end{bmatrix}
\begin{bmatrix}
Y_c \\
Y_s
\end{bmatrix}
$$

(26)

where the matrices $T_c, T_d, B, Y_c, Y_s$ are derived from (25), respectively.

For a prime number $N$ of the form $N = 4L + 1$, where $L$ is any positive integer. The matrix $B$ is a $[(N-1)/2] \times [(N-1)/2]$ circulant matrix, which can be expressed in the form of

$$
B = \begin{bmatrix}
F_1 & F_2 \\
F_2 & F_1
\end{bmatrix}
$$

(27)

where $F_1$ and $F_2$ are both $[(N-1)/4] \times [(N-1)/4]$ matrices. Therefore, matrix $B$ is still suitable for step 4) of the algorithm presented in Subsection 2.1, and we can use similar approach as forward DCT suggested in Section 3 for the implementation of IDCT.

3 Proposed Structure

In this section, we derive the proposed GDA-based structure for cyclic convolution and apply that for DCT computation.

3.1 Proposed Structure for Cyclic Convolution

Let us consider the example of implementation of a 3-point cyclic convolution given by

$$
\begin{bmatrix}
u_1 \\
v_2 \\
v_3
\end{bmatrix} =
\begin{bmatrix}
a \\
b \\
c
\end{bmatrix}
$$

(28)

where $\{u_1, u_2, u_3\}$ are output data, $\{a, b, c\}$ are coefficients, and $\{v_1, v_2, v_3\}$ are input data. As suggested in [14] using a group ROM where the ROM contents are shared, only 12 words are needed instead of 24 words needed by the conventional DA. Besides, a barrel-shifter is used for shifting the successive LUT output for shift accumulation, and an address decoder is used for selecting the content of LUT location referenced by the address word.

Although the approach suggested in [14] provides an efficient implementation using GDA, it can be improved further. In Fig. 1, the proposed design is presented, where the original address and group address are the same as those in Table III in [14]. It is observed that in the existing memory unit, as shown in Table III in [14], when the group address is 3, the LUT produces three “0”s. Besides, the barrel-shifter does not make any shifts when the group address is 3. Therefore, we need not store “0”, and the ROM size can accordingly, be reduced. To generate the “0” output, we pass the LUT output through AND cells as shown in Fig. 1. The structure of AND cell is described as shown in Fig. 2 (a). It consists of an array of 2-input AND gates. One of the inputs of all the AND gates are fed with a control signal and the other input of the AND gates are fed with bits of LUT output. When the control signal is “1”, namely the group address is not 3, the memory unit yields its output according to the existing design. But when the group address is 3, the control signal becomes “0”, the AND cell produces ‘0’ as output. Each of the AND cells consists of $H$ number of AND gates working in
parallel, and Fig. 2(a) shows an example of \( H=2 \), where \( H \) is word-length of the input sequence. Fig. 2(b) shows the detail of the control unit which produces the control signal. Note that when the input-length of the memory unit becomes large, the reduction of memory size is more.

### 3.2 Proposed Design for DCT

The proposed structure for realization of \((N-1)\)-point cyclic convolution of (6) is shown in Fig. 3. It consists of an input buffer, two units for realization of \( T_0 \) and \( T_1 \) named as T units (TU) and an output buffer. For a low-cost design, we may share one TU to reduce the area requirement if high-speed computation is not required.

Each TU contains two sub-units for implementation of \( E_1 \) and \( E_2 \) of (16). Each TU involves two length-\((N-1)/4\) cyclic convolution units (CCU)s along with circuits of input and output additions. Fig. 3 shows the design detail for the computation of (16). The sub-unit for \( E_1 \) contains a CCU similar to the one shown in Fig. 1 since \( E_1 \) is a circulant matrix. The detailed structure of the sub-unit for the computation of \( E_2 \) depends on the detailed decomposition of \( E_2 \). Matrix \( E_2 \) can be decomposed into two matrices, where one is \([(N-1)/4] \times [(N-1)/4] \) circulant matrix and the other is a \([(N-1)/4] \times [(N-1)/4] \) sparse matrix. The circulant matrix is implemented by a \((N-1)/4\)-length CCU similar to the one shown in Fig. 1, while each of the inner-products with non-zero rows of the sparse matrix are implemented by DA-based inner-product units. For example, if a given row has \( n \) non-zero terms, the DA unit requires a ROM of \( 2^n \) words along with an accumulation cell (AC). After \( l \) cycles (\( l \) refers to the word-length of the input sequence), both the non-CCU and CCU yield their output to the output buffer for final addition.

According to the data presented in Table IV in [14], for a \([(N-1)/4]\)-point cyclic convolution (such as \( E_1 \) of (18a)), the existing GDA approach requires ROM of size as

\[
[(N-1)/4] \times 2^{(6/7)\times[(N-1)/4]} \text{ words} \quad (30a)
\]

Using the proposed GDA approach, the ROM requirement is reduced to

\[
[(N-1)/4] \times 2^{(6/7)\times[(N-1)/4]} - [(N-1)/4] \text{ words} \quad (29b)
\]

### 4 COMPARISON AND DISCUSSION

The hardware and time complexities of the proposed structure are evaluated and listed in Table 1, along with those of the existing DA-based structure of [11]. The number of ROM words of proposed structure is estimated as follows:

1. The implementation of \( E_1 \):

\[
[(N-1)/4] \times 2^{(6/7)\times[(N-1)/4]} - [(N-1)/4] \text{ words} \quad (30a)
\]

2. The implementation of \( E_2 \):

\[
[(N-1)/4] \times 2^{(6/7)\times[(N-1)/4]} - [(N-1)/4] + R_S \text{ words} \quad (30b)
\]

where \( R_s \) is defined as the ROM words for implementation of sparse matrix, which is different for even and odd values of \((N-1)/4\).

If \((N-1)/4\) is an even number:

\[
R_s = 2^{[(N-1)/8]^2} + 2 \times 2^{[(N-1)/8-1]^2} + 2^{[(N-1)/8-2]^2} + \cdots + 2^1 \quad (30c)
\]

If \((N-1)/4\) is an odd number:

\[
R_s = 2 \times 2^{[(N-5)/8]^2} + 2^{[(N-5)/8-1]^2} + \cdots + 2^1 \quad (30d)
\]

3. Therefore, the total number of ROM words of the proposed design is

\[
(N-1) \times 2^{(3/14)\times[(N-1)]} - (N-1) + 2R_S \quad (30e)
\]

and the number of adders is \((2N-6)\).
The proposed structure and the existing design of [11] have nearly the same number of adders, while the proposed design has less number of ROM words than the existing design. For example, for $N=17$, the proposed design needs only 128 words to be stored in the ROM, while in the existing design requires 256 ROM words. The cycle period of the proposed design is slightly larger than the existing one, but the number of ROM-words of the proposed design is significantly less than that of the existing one. Besides, for low-speed applications, we can share one TU further reduce the area requirement.

The proposed two-dimensional (2-D) DCT is designed using two 1-D DCT structures and a transposition buffer, as shown in Fig. 4. To meet the peak-signal-to-noise-ratio (PSNR) requirements, the input and output precision are chosen to be 9-bit and 12-bit, respectively. We have used a test image “Lena” with $845 \times 845$ pixels to confirms the bit-length of the proposed 2-D DCT. The transform-length of the proposed 1-D DCT is chosen as 13. After inputting the original test image pixels to the proposed 2-D DCT core, the output data is obtained and fed to MATLAB 7.1 to compute the IDCT. The PSNR is close to 44.9 dB, which ensures the accuracy of the proposed design.

We have implemented the proposed 2-D DCT using Synopsys Design Compiler with 1.8-V TSMC 0.18$\mu$m library. The proposed design can operate at 129MHz with power consumption of 36.6 mW. Its gate count is 18.7 k and the area of the chip is 635$\times$636$\mu$m. We have also implemented the design of [11] for transform-length $N=12$ for the same input and output word-lengths as well as the proposed design.

The results of implementation of the design of [9], [10], [11] and the proposed design are shown in Table 2. Note that the minimum length of the proposed design is 13. Therefore, for the design of [11], the transform length is taken to be 12, for a fair comparison. While the other two designs [9] and [10] the authors have chosen 8 as the transform length. It can be seen that the proposed design outperforms the existing designs. Compared with the design of [9] and [10], the proposed design involves significantly less gate counts, less area and higher throughput rate. The proposed design consumes less than $(2/3)$rd the power, and nearly $(3/4)$th the area than the design of [11].

### Table 1

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of ROM words</td>
<td>$(N-1) \times 2^{(3/14)(N-1)} + R_e - (N-1)$</td>
<td>$N \times 2^{N/4}$</td>
</tr>
<tr>
<td>Number of adders</td>
<td>$2N-6$</td>
<td>$2N+3$</td>
</tr>
<tr>
<td>Cycle period</td>
<td>$T_{mem} + T_{add} + T_b + T_A$</td>
<td>$T_{mem} + T_{add}$</td>
</tr>
<tr>
<td>Latency</td>
<td>$L+4$</td>
<td>$L+1$</td>
</tr>
<tr>
<td>Transform length</td>
<td>$(N-1)=4 \times L$</td>
<td>$N=4 \times L$</td>
</tr>
</tbody>
</table>

$T_{mem}$: time required for ROM read. $T_{add}$: time required for addition. $T_b$: time required for barrel-shifter. $T_A$: time required for an AND gate-access. $L$: word-length for the values stored in the ROM.

### 5 Conclusion

We have proposed an efficient decomposition scheme for hardware-efficient realization of DCT. Besides, we have proposed an efficient design for implementation of cyclic convolution based on our modified GDA technique, and applied to the proposed DCT design. From the synthesis results we find that the proposed design involves significantly less area complexity, yields higher throughput with lower power consumption than the existing ones. The structure proposed here is highly regular, modular and therefore, suitable for VLSI realization.

### Acknowledgment

The authors would like to thank the Editors, and the reviewers for their constructive comments. This work is supported by the National Natural Science Foundation of China under Project 60843002, 61174132 and the Natural Science Foundation of Hunan Province under Project 09JJ6098.

### References


![Fig. 4. Proposed 2-D DCT block diagram.](image-url)
TABLE 2


<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Technology</td>
<td>0.18μm</td>
<td>0.6μm</td>
<td>0.18μm</td>
<td>0.18μm</td>
</tr>
<tr>
<td>Multipliers or ROMs</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>Power consumption (mW)</td>
<td>36.6mW@129MHz (1.8V)</td>
<td>4.38mW@14MHz (1.56V)</td>
<td>—</td>
<td>44.6mW@103MHz (1.8V)</td>
</tr>
<tr>
<td>Gate counts (NAND2)</td>
<td>18.7k</td>
<td>30k*</td>
<td>22.5k</td>
<td>23.2k</td>
</tr>
<tr>
<td>Normalized power&amp;</td>
<td>0.31</td>
<td>0.55</td>
<td>—</td>
<td>0.51</td>
</tr>
<tr>
<td>Normalized area#</td>
<td>1.44</td>
<td>3.75</td>
<td>2.81</td>
<td>1.93</td>
</tr>
</tbody>
</table>

*4 transistors per NAND2 gate for different technology.
#The normalized area=(gate counts)/(transform length), the smaller, the better.
&The normalized power=(power consumed@14MHz)/(transform length), the smaller, the better.


Jiafeng Xie received the B.E. and M.E. from Yan- shan University and Central South University in 2006 and 2010, respectively. His research interest includes VLSI signal processing systems design, cryptography circuits design, low power circuits design and other algorithm-to-architecture design like the wireless communication circuits design.

Pramod Kumar Meher(SM’03) received the B.Sc. and M.Sc. degrees in Physics and the Ph.D. degree in science from Sambalpur University, Sambalpur, India, in 1976, 1978, and 1996, respectively. He has a wide scientific and technical background covering physics, electronics, and computer engineering. He is a Senior Scientist with the Institute for Infocomm Research, Singapore from 2009. Prior to this assignment, he was a visiting faculty member with the School of Computer Engineering, Nanyang Technological University, Singapore. Previously, he was a Professor of Computer Applications with Utkal University, Bhubaneswar (India), from 1997 to 2002, a Reader in Electronics with Berhampur University, Berhampur (India) from 1993 to 1997, and a Lecturer in Physics with various government colleges in India from 1981 to 1993. His research interest includes design of dedicated and reconfigurable architectures for computation-intensive algorithms pertaining to signal, image and video processing, communication, bioinformatics, and intelligent computing. He has published more than 170 technical papers in various reputed journals and conference proceedings. Dr. Meher is a Fellow of the Institution of Electronics and Telecommunication Engineers, India and a Fellow of the Institution of Engineering and Technology, U.K. He has served as an Associate Editor for the IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS-II: EXPRESS BRIEFS during 2008-2011. Currently he is serving as a speaker for the Distinguished Lecturer Program (DLP) of IEEE Circuits and Systems Society, and an Associate Editor for the IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS-I: REGULAR PAPERS, IEEE TRANSACTIONS ON VERY LARGE SCALE INTEGRATION (VLSI) SYSTEMS, and Journal of Circuits, Systems, and Signal Processing. He was the recipient of the Samanta Chandrasekhar Award for excellence in research in engineering and technology for 1999.

Jianjun He received Ph.D. from Central South University. His main research interest including control engineering system design, and VLSI signal processing system design.