An Efficient Prime-Factor Algorithm for the Discrete Cosine Transform and Its Hardware Implementations*

PeiZong Lee† and Fang-Yu Huang

Institute of Information Science
Academia Sinica
#128, Sec. 2, Yen-Chiu-Yun Road
Nankang, Taipei, Taiwan, R. O. C.

December 3, 1993

Abstract

The prime-factor decomposition is a fast computational technique for many important digital signal processing operations, such as the convolution, the discrete Fourier transform, the discrete Hartley transform, and the discrete cosine transform (DCT). In this paper, we present a new prime-factor algorithm for the DCT. We also design a prime-factor algorithm for the discrete sine transform which is based on the prime-factor DCT algorithm.

Hardware implementations for the prime-factor DCT are also studied. We are especially interested in the hardware designs which are suitable for the VLSI implementations. We will show three hardware designs for the prime-factor DCT, including a VLSI circuit fabricated directly according to the signal-flow graph, a linear systolic array, and a mesh-connected systolic array. These three designs show the trade-off between cost and performance. The methodology, which deals with general $(N_1 \cdot N_2)$-point DCTs, where N_1 and N_2 are mutually prime, is illustrated by converting a 15-point DCT problem into a (3×5)-point 2-D DCT problem.

EDICS number: 4.1, 4.1.1, 7.1.4.

Index Terms: discrete cosine transform, inverse discrete cosine transform, prime-factor decomposition, signal-flow graph, systolic array.

*This work was partially supported by the NSC under Grants NSC 81-0408-E-001-505.

†PeiZong Lee: leepe@iis.sinica.edu.tw, TEL: +886 (2) 788-3799, FAX: +886 (2) 782-4814.

1 Introduction

This paper is concerned with designing a fast prime-factor algorithm for the discrete cosine transform (DCT). The DCT, which performs much like the theoretically optimal Karhunen-Loeve transform for the first-order Markov stationary random data, has found wide applications in speech and image processing as well as telecommunication signal processing for the purpose of data compression, feature extraction, and filtering. In order to compute the DCT efficiently, fast algorithms have been intensively studied [9][12][13][16][19][28][30]. A complete survey of the DCT algorithms can be seen elsewhere [14][21].

It is our goal in this paper to present a new prime-factor DCT algorithm. The prime-factor decomposition is a fast computational technique for many important digital signal processing operations, such as the convolution [20], the discrete Fourier transform (DFT) [1][2][3][4][10][22][24][25][26][27], the discrete Hartley transform (DHT) [5][18][23], and the DCT [5][6][7][13][35]. It has both theoretical and practical significance. Its main theoretical rationale is to convert a large-size one-dimension problem, by employing certain appropriate index mappings, into a multidimensional one. Then, we can deal with the resulting groups of small-size problems in each dimension.

For practical considerations, since in a typical DSP processor the memory for data storage is expensive and usually not large, it is more feasible to process small-size problems one at a time. In addition, when this approach is combined with efficient short-length algorithms, such as Rader's algorithm [20], or Winograd type minimum multiplication algorithms in DFT [31], or Heideman's small odd-length DCT modules [8], etc., it would be of practical interest in reducing the scalar multiplication complexity.

Although Chan and Ho [6] and Cho and Lee [7] derived prime-factor DCT algorithms based on variant DFT algorithms before, their algorithms required additional complex-number multiplications. Unlike them, Yang and Narasimha [35] proposed a prime-factor DCT algorithm which included only real-number multiplications; however, its index mapping was complicated. From our point of view, an index mapping not only should be easy to understand, but also should be efficient in running. Lee [13] has achieved this goal. However, his input index mapping is realized by constructing and combining two index tables, which occupy additional memory space and would be infeasible in variable-size applications.

Chakrabarti and JáJá [5] develop a systolic architecture for implementing Lee's algorithm. Because they want to compute the DCT from the DHT, they modify the index mappings, which are essentially the same as Lee's. However, they did not discuss the actual implementation for these index mappings.

In this paper, the input index mapping we adopt is the Ruritanian mapping, since its efficient realization can be based on the previous research efforts [17][32][33][34]. In addition, the resulting algorithm complexity is by no means increased. As for the output index mapping, we employ the same one as Lee's, and this may be the most natural one in view of the DCT transform kernel's structure.

We also consider hardware implementations for the prime-factor DCT. We are interested in the hardware designs which are suitable for the VLSI (Very Large Scale Integration) implementations. We will show three hardware designs for the prime-factor DCT, including a VLSI circuit fabricated directly according to the signal-flow graph, a linear systolic array, and a mesh-connected systolic array. These three designs show the trade-off between cost and performance. Finally, by generalizing Wang's method [29], we design a prime-factor discrete sine transform (DST) algorithm based on the proposed prime-factor DCT algorithm. Both the DCT and the DST discussed in this paper are of type II defined in [30], and their inverse transforms are of type III.

The rest of this paper is organized as follows. In Section 2, the DCT is introduced. In Section 3, we derive the prime-factor DCT algorithm. In Section 4, we propose three hardware implementations. In Section 5, we derive the prime-factor DST based on the prime-factor DCT. In Section 6, we give some concluding remarks. Finally, in the Appendix, we give the proof of two main theorems.

2 Background

For a given input data sequence x_n, $0 \leq n \leq N-1$, the DCT output data sequence X_k, $0 \leq k \leq N-1$, is defined by

$$X_k = \sqrt{\frac{2}{N}}\,\epsilon(k) \sum_{n=0}^{N-1} x_n \cos(\frac{\pi(2n+1)k}{2N})\;; \tag{1}$$

and the IDCT (inverse discrete cosine transform) is defined by

$$x_n = \sqrt{\frac{2}{N}} \sum_{k=0}^{N-1} \epsilon(k)\, X_k \cos(\frac{\pi(2n+1)k}{2N}) \; ; \tag{2}$$

where

$$\epsilon(k) = \begin{cases} \frac{1}{\sqrt{2}} \, , & \text{for } k = 0; \\ 1 \, , & \text{for } 1 \le k \le N-1 \, . \end{cases}$$

We assume throughout this paper that N is a product of two mutually prime integers N_1 and N_2. For convenience, we will ignore the scaling factor $\epsilon(k)$ and the normalization factor $\sqrt{\frac{2}{N}}$ in Equations (1) and (2), since they can be done in a separate step. In the following, we deal with the simplified version of Equation (2) :

$$x_n = \sum_{k=0}^{N-1} X_k \cos(\frac{\pi(2n+1)k}{2N}) \; . \tag{3}$$

We want to convert Equation (3) into the form in Equation (4) by taking appropriate input and output index mappings:

$$x_{(n_1,n_2)} = \sum_{k_1=0}^{N_1-1} \sum_{k_2=0}^{N_2-1} Y_{(k_1,k_2)} \cos(\frac{\pi(2n_1+1)k_1}{2N_1}) \cos(\frac{\pi(2n_2+1)k_2}{2N_2}) \; , \tag{4}$$

where $0 \le n_1 < N_1$, $0 \le n_2 < N_2$, and $Y_{(k_1,k_2)}$ are the result of certain modifications on $X_{(k_1,k_2)}$.

The input index mapping connects the input index k, $0 \le k < N$, to (k_1, k_2), $0 \le k_1 < N_1$ and $0 \le k_2 < N_2$. The output index mapping connects the output index n, $0 \le n < N$, to (n_1, n_2), $0 \le n_1 < N_1$ and $0 \le n_2 < N_2$. Since the DCT is orthonormal, the forward transform also can be realized by taking the transpose of the inverse transform.

3 Derivation of the Prime-Factor Decomposition

The input index mapping we adopt is the Ruritanian mapping, which was also used in the prime-factor DFT algorithms [2] [3] [4] [10] [22], and several researchers have studied its efficient implementation methods [17] [32] [33] [34]. That is,

$$\forall\, k \in [0,\, N-1],\ \exists!\,(k_1,k_2) \in [0,\, N_1-1] \times [0,\, N_2-1],$$

such that

$$k = \psi(k_1,k_2) = (N_2 k_1 + N_1 k_2) \bmod N \; ,$$

where $N = N_1 N_2$, N_1 and N_2 are mutually prime.

In other words, based on the Ruritanian mapping, the integers k on the interval $[0,\ N-1]$ correspond one-to-one to the lattice points in the region $[0,\ N_1 - 1] \times [0,\ N_2 - 1]$. For the sake of our derivation, these lattice points are divided into five disjoint groups.

Let

$$\begin{aligned} f(k_1, k_2) &= N_2 k_1 + N_1 k_2 \ , \\ g(k_1, k_2) &= N_2 k_1 - N_1 k_2 \ . \end{aligned}$$

Define

$$\begin{aligned} E &= \{(k_1, k_2) \,|\, k_1 = 0 \text{ or } k_2 = 0\} \ , \\ A &= \{(k_1, k_2) \,|\, f(k_1, k_2) < N \text{ and } g(k_1, k_2) > 0;\ k_1 k_2 \neq 0\} \ , \\ B &= \{(k_1, k_2) \,|\, f(k_1, k_2) < N \text{ and } g(k_1, k_2) < 0;\ k_1 k_2 \neq 0\} \ , \\ C &= \{(k_1, k_2) \,|\, f(k_1, k_2) > N \text{ and } g(k_1, k_2) < 0;\ k_1 k_2 \neq 0\} \ , \\ D &= \{(k_1, k_2) \,|\, f(k_1, k_2) > N \text{ and } g(k_1, k_2) > 0;\ k_1 k_2 \neq 0\} \ . \end{aligned}$$

Let

$$\begin{aligned} W &= \{k \,|\, k = \psi(k_1, k_2),\ (k_1, k_2) \in E\} \ , \\ U &= \{k \,|\, k = f(k_1, k_2),\ (k_1, k_2) \in (A \cup B)\} \ , \\ V &= \{k \,|\, k = g(k_1, k_2),\ (k_1, k_2) \in (A \cup D)\} \ . \end{aligned}$$

Theorem 1 $\{\,k \,|\, k \in [0,\ N-1],\ \ k \textit{ is an integer}\} = W \cup U \cup V.$ $\square$

From Theorem 1, we can rewrite Equation (3) as follows:

$$x_n = \Big(\sum_{k \in W} + \sum_{k \in U} + \sum_{k \in V}\Big) X_k \cos\Big(\frac{\pi(2n+1)k}{2N}\Big) \ . \tag{5}$$

Let

$$\begin{aligned} X_{(k_1,k_2)} &= X_k, \text{ where } k = \psi(k_1, k_2) \ , \\ X^f_{(k_1,k_2)} &= X_k, \text{ where } k = f(k_1, k_2) \text{ and } (k_1, k_2) \in (A \cup B) \ , \\ X^g_{(k_1,k_2)} &= X_k, \text{ where } k = g(k_1, k_2) \text{ and } (k_1, k_2) \in (A \cup D) \ . \end{aligned}$$

Because, when (k_1, k_2) is in E, $f(k_1,k_2) = |\, g(k_1,k_2)\,| = \psi(k_1,k_2)$, we have $\cos(\frac{\pi(2n+1)f(k_1,k_2)}{2N}) = \cos(\frac{\pi(2n+1)g(k_1,k_2)}{2N}) = \cos(\frac{\pi(2n+1)\psi(k_1,k_2)}{2N})$. Therefore, Equation (5) becomes:

$$\begin{aligned} x_n \;=\; & \frac{1}{2}\{ \sum_{(k_1,k_2)\in E} X_{(k_1,k_2)} [\, \cos(\frac{\pi(2n+1)\, f(k_1,k_2)}{2N}) + \cos(\frac{\pi(2n+1)\, g(k_1,k_2)}{2N})] + \\ & 2 \sum_{(k_1,k_2)\in(A\cup B)} X^f_{(k_1,k_2)} \cos(\frac{\pi(2n+1)\, f(k_1,k_2)}{2N}) + \\ & 2 \sum_{(k_1,k_2)\in(A\cup D)} X^g_{(k_1,k_2)} \cos(\frac{\pi(2n+1)\, g(k_1,k_2)}{2N}) \} \,. \end{aligned} \tag{6}$$

Now, we define

$$Y_{(k_1,k_2)} = \begin{cases} X_{(k_1,k_2)}, & \text{if } (k_1,k_2) \in E \\ X_{(k_1,k_2)} + X_{(k_1,N_2-k_2)}, & \text{if } (k_1,k_2) \in A \\ X_{(k_1,k_2)} + X_{(N_1-k_1,k_2)}, & \text{if } (k_1,k_2) \in B \\ -X_{(N_1-k_1,N_2-k_2)} + X_{(N_1-k_1,k_2)}, & \text{if } (k_1,k_2) \in C \\ -X_{(N_1-k_1,N_2-k_2)} + X_{(k_1,N_2-k_2)}, & \text{if } (k_1,k_2) \in D \,. \end{cases} \tag{7}$$

Theorem 2

$$x_n = \sum_{k_1=0}^{N_1-1} \sum_{k_2=0}^{N_2-1} Y_{(k_1,k_2)} \cos(\frac{\pi(2n+1)k_1}{2N_1}) \cos(\frac{\pi(2n+1)k_2}{2N_2}) \,. \qquad \square$$

Next, we introduce the output index mapping, which was proposed by Lee [13]. For the completeness of this presentation, we list it here for easy reference.

The output index mapping: $\varphi(n) = (n_1, n_2)$, where

$$\begin{aligned} \bar{n}_1 &= n \bmod 2N_1 \\ \bar{n}_2 &= n \bmod 2N_2 \\ n_1 &= \begin{cases} \bar{n}_1, & \text{if } \bar{n}_1 < N_1 \\ 2N_1 - 1 - \bar{n}_1, & \text{otherwise} \end{cases} \\ n_2 &= \begin{cases} \bar{n}_2, & \text{if } \bar{n}_2 < N_2 \\ 2N_2 - 1 - \bar{n}_2, & \text{otherwise.} \end{cases} \end{aligned} \tag{8}$$

It is true that $\varphi : [0,\, N-1] \longrightarrow [0,\, N_1-1] \times [0,\, N_2-1]$ is a one-to-one mapping function. In addition,

$$\cos(\frac{\pi(2n+1)k_1}{2N_1}) = \cos(\frac{\pi(2n_1+1)k_1}{2N_1}) \quad \text{and} \quad \cos(\frac{\pi(2n+1)k_2}{2N_2}) = \cos(\frac{\pi(2n_2+1)k_2}{2N_2}).$$

Therefore, the following theorem can be derived directly from Theorem 2.

Theorem 3

$$x_{(n_1,n_2)} = \sum_{k_1=0}^{N_1-1} \sum_{k_2=0}^{N_2-1} Y_{(k_1,k_2)} \cos(\frac{\pi(2n_1+1)k_1}{2N_1}) \cos(\frac{\pi(2n_2+1)k_2}{2N_2}) . \qquad \square$$

Prime-factor algorithm for the IDCT:

Input: An N-point data sequence X_k, $0 \leq k < N$, where $N = N_1 N_2$, N_1 and N_2 are mutually prime.

Output: An N-point data sequence x_n, $0 \leq n < N$.

Step 1: Apply the Ruritanian mapping on X_k to construct an $(N_1 \times N_2)$-point 2-D data matrix $X_{(k_1,k_2)}$, where $0 \leq k_1 < N_1$ and $0 \leq k_2 < N_2$.

Step 2: Modify the 2-D data matrix $X_{(k_1,k_2)}$ to get the 2-D data matrix $Y_{(k_1,k_2)}$ according to Equation (7).

/* This is proved to be correct in Theorem 2. */

Step 3: /* Row-column evaluation */

1. Execute the N_1-point IDCT for each of the N_2 columns of $Y_{(k_1,k_2)}$, and the result is $T_{(k_1,k_2)}$.
2. Execute the N_2-point IDCT for each of the N_1 rows of $T_{(k_1,k_2)}$, and the result is $x_{(n_1,n_2)}$, where $0 \leq n_1 < N_1$ and $0 \leq n_2 < N_2$.

/* This is proved to be correct in Theorem 3. */

Step 4: Apply the output index mapping in Equation (8) to get the final result x_n. $\square$

In the following, we use an example to demonstrate our method.

Example : In this example, $N = 15$, and $N_1 = 3$, $N_2 = 5$. Figure 1 shows the signal flow graph for implementing the 15-point 1-D IDCT, which can be converted into the (3×5)-point 2-D IDCT. First, when given a 15-point 1-D input data sequence X_k, $0 \leq k < 15$, we transform this 1-D data sequence by the Ruritanian mapping to a (3×5)-point 2-D data matrix $X_{(k_1,k_2)}$, $0 \leq k_1 < 3$ and $0 \leq k_2 < 5$. Table 1 shows the Ruritanian mapping for this case. The five disjoint sets of lattice

k_2 k k_1	0	1	2	3	4
0	0	3	6	9	12
1	5	8	11	14	2
2	10	13	1	4	7

Table 1: The Ruritanian mapping: $k = (5k_1 + 3k_2) \bmod 15$, when $N = 15$, and $N_1 = 3$, $N_2 = 5$.

points in the region $[0, 2] \times [0, 4]$, which are divided by $k_1 k_2 = 0$; $5k_1 + 3k_2 > 15$; $5k_1 + 3k_2 < 15$; $5k_1 - 3k_2 > 0$; and $5k_1 - 3k_2 < 0$; include

$$
\begin{aligned}
E &= \{(0,0), (0,1), (0,2), (0,3), (0,4), (1,0), (2,0)\}\,; \\
A &= \{(1,1), (2,1)\}\,; \\
B &= \{(1,2), (1,3)\}\,; \\
C &= \{(1,4), (2,4)\}\,; \\
D &= \{(2,2), (2,3)\}\,.
\end{aligned}
$$

Next, we modify the 2-D data matrix $X_{(k_1,k_2)}$ to get the 2-D data matrix $Y_{(k_1,k_2)}$ according to Equation (7). Then, we apply the row-column evaluation for the computation: first, we deal with five 3-point IDCTs; then, take the transpose of the result; and then, deal with three 5-point IDCTs. Finally, the output data sequence $x(n)$, $0 \le n < 15$, can be obtained by using the output index mapping in Equation (8). Table 2 shows the output index mapping for this case.

n_2 n n_1	0	1	2	3	4
0	0	11	12	6	5
1	10	1	7	13	4
2	9	8	2	3	14

Table 2: The output index mapping when $N = 15$, and $N_1 = 3$, $N_2 = 5$.

Note that the signal flow graph in Figure 1 performs the IDCT if the signals flow from left to right, and performs the DCT if the signals flow from right to left. $\square$

4 Hardware Implementations

In this section, we consider several hardware implementations for the prime-factor IDCT. We are especially interested in the hardware designs which are suitable for the VLSI implementations.

Fig. 1 shows the signal-flow graph for the implementation of a 15-point prime-factor IDCT. It is possible to fabricate such a 15-point prime-factor IDCT circuit directly into a single chip, which may contain millions of transistors, by using current VLSI technologies. The input data sequences can then be pipelined and enter into this chip, and an output data sequence is produced in every cycle time. This design, of course, has achieved the best performance and the maximum throughput. We will call this implementation, which is fabricated directly according to the signal-flow graph, *Design I*.

However, when the problem size grows, it is difficult to fabricate a large-size prime-factor IDCT circuit directly into a single chip. This is because it requires a network including several butterfly-like interconnections and a transpose network that are very costly to layout in most circuit technologies (including VLSI).

After carefully studying the prime-factor algorithm, we find that Step 1 and Step 4 only deal with $O(N)$ memory references. Step 2 only requires $O(N)$ addition or subtraction operations. However, Step 3, which implements the row-column evaluation, requires at least $O(N \log N)$ multiplication and addition operations by any fast IDCT algorithm [9] [12] [14]. It is clear that Step 3 is the bottleneck of the algorithm. Steps 1, 2, and 4 can be implemented by the host computer, or by a single processing element, or by a single processing element each. Therefore, in the following we concentrate our effort in solving Step 3 by using feasible VLSI algorithms.

In Fig. 1, we find that there are five 3-point IDCT components and three 5-point IDCT components. In general, there are N_2 N_1-point IDCT components and N_1 N_2-point IDCT components. In practice, in order to save hardware without slowing down the processing speed, we can use only one 3-point (N_1-point) IDCT component and one 5-point (N_2-point) IDCT component as shown in Fig. 2, provided the data sequences are carefully arranged.

In Fig. 2, Steps 1, 2, and 4 are each implemented by a single processing element. Step 3 requires a 3-point (N_1-point) IDCT component, a transpose buffer, and a 5-point (N_2-point)

IDCT component. The data matrix generated after Step 2 is pipelined and enters into the 3-point (N_1-point) IDCT component column by column. The output data matrix of the 3-point (N_1-point) IDCT component is stored in a transpose buffer. After this data matrix enters the transpose buffer, this data matrix in the transpose buffer is then pipelined and enters into the 5-point (N_2-point) IDCT component row by row. The resulting data matrix is then sent to a processing element for generating the output data sequence by using the output index mapping.

We now consider the fabrication of an arbitrary N-point IDCT component. It is true that there is no fast algorithm for the N-point IDCT, except if N has a value of 2 to a power. However, in practice, the value of N may be varied. Previously, if N did not have a value of 2 to a power, then either we should pad additional zeros so that we could use a fast IDCT algorithm, or we should use the conventional matrix-vector multiplication algorithm to solve the N-point IDCT.

Suppose that we base this on an $O(N \log N)$ fast sequential IDCT algorithm for laying out the IDCT chip. If $N_1 \neq N_2$, then the layout of the N_1-point IDCT component is quite different from the layout of the N_2-point IDCT component. In addition, as mentioned before, because this fast sequential algorithm requires a network with several butterfly-like interconnections, the circuit layout becomes expensive. Therefore, if a chip includes both the N_1-point IDCT component and the N_2-point IDCT component, then the circuit design cycle will certainly be lengthened.

Recently, Kung and Leiserson [11] used matrix-vector multiplication algorithm to solve the discrete Fourier transform (DFT). Although it requires $O(N^2)$ multiply-and-add operations for solving the N-point DFT in the sequential algorithm, the algorithm can be implemented by a linear systolic array with N processing elements in $O(N)$ time. This result is promising in comparison with any sequential fast Fourier transform algorithm with $O(N \log N)$ operation complexity.

Like the DFT, the IDCT also can be implemented by the conventional matrix-vector multiplication. In addition, we also can use a linear systolic array with N processing elements to implement the N-point IDCT in $O(N)$ time. We will call this linear systolic array implementation *Design II.*

However, as each processing element requires N registers or memory storage to store part of the N-point IDCT coefficient matrix, each processing element needs additional memory addressing and control hardware, which will increase the circuit complexity.

We now consider the time complexity and the area complexity of this linear systolic array implementation. It requires $O(N)$ units of time for solving the N-point IDCT with this linear systolic array with N processing elements. The area complexity of this linear systolic array which has the same order as the total number of registers or memory storage is $O(N^2)$. Therefore, the VLSI performance measure AT^2 of this linear systolic array implementation is $O(N^4)$, where A means the circuit area and T means the execution time. However, this performance can not be satisfied.

We now show that the computation required by the row-column evaluation can be represented by two matrix-matrix multiplications. Suppose that the $N_1 \times N_2$ data matrix Y is generated after Step 2. Then, by the row-column evaluation,

$$x^T_{N_1\times N_2} = C_{N_2\times N_2}\,(C_{N_1\times N_1}\,Y_{N_1\times N_2})^T,$$

where $C_{N_1\times N_1}$ and $C_{N_2\times N_2}$ are the N_1-point and the N_2-point IDCT coefficient matrices, respectively. That is,

$$T_{N_1\times N_2} = C_{N_1\times N_1}\,Y_{N_1\times N_2} \quad \text{and} \quad x^T_{N_1\times N_2} = C_{N_2\times N_2}\,T^T_{N_1\times N_2}.$$

Thus, the operation complexity of the prime-factor IDCT is $O(N(N_1+N_2))$ in the sequential algorithm. This is better than the operation complexity $O(N^2)$ of the IDCT implemented by the matrix-vector multiplication.

We now show how to use mesh-connected systolic arrays for implementing the row-column evaluation [15]. We will call this mesh-connected systolic array implementation *Design III.* In order to implement the row-column evaluation, three components are required: an $N_1 \times N_1$ mesh-connected systolic array for implementing the matrix-matrix multiplication $T_{N_1\times N_2} = C_{N_1\times N_1}\,Y_{N_1\times N_2}$; a transpose component for implementing $T^T_{N_1\times N_2}$ from $T_{N_1\times N_2}$; and an $N_2 \times N_2$ mesh-connected systolic array for implementing the matrix-matrix multiplication $x^T_{N_1\times N_2} = C_{N_2\times N_2}\,T^T_{N_1\times N_2}$.

Fig. 3 shows the complete data flow of the 15-point IDCT systolic array implementation. When given an input data sequence X_k, $0 \le k < 15$, we transform this 1-D data sequence by the Ruritanian mapping to a (3×5)-point 2-D data matrix $X_{(k_1,k_2)}$ by using only one processing element, for $0 \le k_1 < 3$ and $0 \le k_2 < 5$. Next, we modify the 2-D data matrix $X_{(k_1,k_2)}$ to the 2-D data matrix $Y_{(k_1,k_2)}$ according to Equation (7) also by using only one processing element.

Then, we use three systolic array components to implement the row-column evaluation. First, we deal with the matrix-matrix multiplication $T_{3\times 5} = C_{3\times 3} Y_{3\times 5}$ using a 3×3 mesh-connected systolic array. The data entries $C^3_{(i,j)} = \cos(\frac{\pi(2i+1)j}{2\cdot 3})$ of the 3-point IDCT coefficient matrix are stored in the 3×3 mesh-connected systolic array initially. The data matrix $Y_{(k_1,k_2)}$, which requires it to be aligned skewedly, flows from north to south. The intermediate data matrix $T_{(k_1,k_2)}$, which also requires it to be aligned skewedly and whose initial value $T^0_{(k_1,k_2)}$ is zero, flows from west to east. In each processing element of the 3×3 mesh-connected systolic array, it contains a multiply-and-add circuit which implements $T_{(k_1,k_2)} = T_{(k_1,k_2)} + C^3_{(k_1,i)} * Y_{(i,k_2)}$, $(O = I_2 + R * I_1)$, for some integers k_1, k_2, and i.

Second, the intermediate data matrix $T_{(k_1,k_2)}$ then passes through an adjustment buffer, which adds one unit of delay between two adjacent data entries of the data flow of $T_{(k_1,k_2)}$. This one unit of delay between two adjacent data entries is necessary when the data matrix $T_{(k_1,k_2)}$ enters into the transpose component to obtain the transposed skewed data matrix $T^T_{(k_1,k_2)}$.

The 3×5 transpose component computes $T^T_{3\times 5}$ from $T_{3\times 5}$, where T is the input matrix of size 3×5 $(N_1 \times N_2)$ and T^T is the output matrix of size 5×3 $(N_2\times N_1)$. The (k_1,k_2)-th entry of T $(T_{(k_1,k_2)})$ will enter into the transpose component from the k_1-th row, then will shift to the eastern buffer cell in each of the time units until it reaches the $(N_2 - 1 - k_2)$-th column of the transpose component, and then will shift to the northern buffer cell in each of the time units. The resulting skewed data matrix $T^T_{N_1\times N_2}$ can then be obtained.

Third, we deal with the matrix-matrix multiplication $x^T_{3\times 5} = C_{5\times 5} T^T_{3\times 5}$ using a 5×5 mesh-connected systolic array. The data entries $C^5_{(i,j)} = \cos(\frac{\pi(2i+1)j}{2\cdot 5})$ of the 5-point IDCT coefficient matrix are stored in the 5×5 mesh-connected systolic array initially. The skewed data matrix $T^T_{(k_1,k_2)}$ flows from south to north. The data matrix $x_{(n_1,n_2)}$, which requires it to be aligned skewedly and whose initial value $x^0_{(n_1,n_2)}$ is zero, flows from east to west. The operations performed in each processing element of the 5×5 mesh-connected systolic array are the same as the case in the 3×3 mesh-connected systolic array.

The resulting data matrix $x_{(n_1,n_2)}$ can then undergo the output index mapping according to Equation (8), and the output data sequence x_n, $0 \le n < 15$, can then be obtained. This step, however, can be implemented by using only one processing element.

We now analyze the time complexity and the area complexity of the mesh-connected systolic array implementation which implements the row-column evaluation. It requires $O(N_1 + N_2)$ units of time for solving two matrix-matrix multiplications and one transpose operation. Because each processing element only contains a constant number of registers, the area complexity of this implementation is $O(N_1^2 + N_1 N_2 + N_2^2)$. Thus, the VLSI performance measure AT^2 of this implementation is $O((\max\{N_1, N_2\})^4)$, which is much better than the case in *Design II.*

5 Computation of the IDST by using the IDCT

In this section, we show how to compute the IDST (inverse discrete sine transform) by using the IDCT. Wang [29] showed that an N-point DST could be computed by an N-point DCT. In the following, we generalize his method to compute the IDST by using the IDCT. Therefore, the prime-factor IDST can be computed by the prime-factor IDCT. Since the DST is orthonormal, the forward transform also can be realized by taking the transpose of the inverse transform.

The IDST output data sequence x_m^s of X_h^s, $0 \le m,\ h \le N-1$, is defined as

$$x_m^s = \sqrt{\frac{2}{N}} \sum_{h=0}^{N-1} \epsilon^s(h)\, X_h^s \sin(\frac{\pi(2m+1)(h+1)}{2N}) \,, \tag{9}$$

where

$$\epsilon^s(h) = \begin{cases} 1\,, & \text{for } 0 \le h \le N-2; \\ \frac{1}{\sqrt{2}}\,, & \text{for } h = N-1. \end{cases}$$

Note that, for the sake of distinguishing the formula of the IDST in Equation (9) from the formula of the IDCT in Equation (2), we use the superscript $'s'$ for denoting the sine transform.

We now state how to compute the IDST by using the IDCT. First, by substituting

$$k = N - 1 - h\,, \quad \text{for } 0 \le h \le N-1\,,$$

into Equation (9), we have

$$x_m^s = \sqrt{\frac{2}{N}} \sum_{k=0}^{N-1} \epsilon^s(N-1-k)\, X_{N-1-k}^s \sin(\frac{\pi(2m+1)(N-k)}{2N}) \,.$$

Since

$$\sin(\frac{\pi(2m+1)(N-k)}{2N}) = (-1)^m \cos(\frac{\pi(2m+1)k}{2N}) \ \text{ and } \ \epsilon^s(N-1-k) = \epsilon(k),$$

we obtain that

$$(-1)^m \, x_m^s \quad = \quad \sqrt{\frac{2}{N}} \sum_{k=0}^{N-1} \epsilon(k) \, X_{N-1-k}^s \cos(\frac{\pi(2m+1)k}{2N}) \; .$$

That is, the data sequence $\{y_m = (-1)^m \, x_m^s\}$, for $0 \leq m \leq N-1$, can be obtained by performing the IDCT on the data sequence $\{Y_k = X_{N-1-k}^s\}$, for $0 \leq k \leq N-1$.

Suppose that N is a product of two mutually prime integers. In the following, we list the procedure for computing the prime-factor IDST by using the prime-factor IDCT.

Prime-factor algorithm for the IDST:

Step 1: Compute the data sequence $\{Y_k = X_{N-1-k}^s\}$, for $0 \leq k \leq N-1$;

Step 2: compute the IDCT output data sequence $\{y_m\}$, $0 \leq m \leq N-1$, for the data sequence $\{Y_k\}$, $0 \leq k \leq N-1$, by using the prime-factor IDCT algorithm;

Step 3: compute $x_m^s = (-1)^m \, y_m$, for $0 \leq m \leq N-1$. □

6 Conclusion

We have presented in this paper a new prime-factor DCT algorithm, which consists of four steps: the input index mapping; the modification; the row-column evaluation; and the output index mapping. The input index mapping we adopted is the Ruritanian mapping method, which is much simpler than the ones designed by Yang and Narasimha [35], and Lee [13]. The output index mapping we employed is the same one as Lee's [13]. We also designed a prime-factor DST algorithm, which is based on the prime-factor DCT algorithm.

The operation count of our prime-factor DCT algorithm includes $(N_1-1)(N_2-1)$ real-number additions for modifying intermediate data and $N(N_1+N_2)$ real-number multiplications and additions each for the row-column evaluation. It is better than the operation count by [6] [7], which includes N complex-number multiplications for modifying intermediate data and $N(N_1+N_2)$ complex-number multiplications and additions each for the row-column evaluation.

We then considered hardware implementations for the prime-factor DCT. We have shown three hardware designs for the prime-factor DCT. The first one, which is a VLSI circuit fabricated directly according to the signal-flow graph, might be not easy to implement when the problem size becomes large. The second one is a linear systolic array implementation, which contains N processing elements and can solve an N-point DCT problem in $O(N)$ systolic steps. The third one is a mesh-connected systolic array implementation, which contains $O(N_1^2 + N_1 N_2 + N_2^2)$ processing elements and can solve the row-column evaluation in $O(N_1 + N_2)$ systolic steps, where $N = N_1 N_2$, and N_1 and N_2 are mutually prime. If we consider the VLSI performance measure AT^2 on the second and the third designs, then the latter one, which is $O((\max\{N_1, N_2\})^4)$, is better than the former one, which is $O(N^4)$.

Acknowledgements

The authors would like to thank Professor H. V. Sorensen and the anonymous referees for their valuable comments on this paper.

Appendix

Lemma 1 (1) *If* (k_1, k_2) *is in* $(A \cup B)$, *then* $\psi(k_1, k_2) = f(k_1, k_2)$.

(2) *If* (k_1, k_2) *is in* $(C \cup D)$, *then* $\psi(k_1, k_2) = g(k_1, N_2 - k_2)$.

Proof : (1) When (k_1, k_2) is in $(A \cup B)$, $N > f(k_1, k_2) = N_2 k_1 + N_1 k_2 = \psi(k_1, k_2)$.

(2) When (k_1, k_2) is in $(C \cup D)$, $N < f(k_1, k_2) = N_2 k_1 + N_1 k_2 < 2N$. Therefore,

$\psi(k_1, k_2) = f(k_1, k_2) - N = N_2 k_1 - N_1(N_2 - k_2) = g(k_1, N_2 - k_2)$. □

Lemma 2 (1) $f(k_1, k_2) = 2N - f(N_1 - k_1, N_2 - k_2)$.

(2) $g(k_1, k_2) = -g(N_1 - k_1, N_2 - k_2)$.

Proof : (1) $f(N_1 - k_1, N_2 - k_2) = N_2(N_1 - k_1) + N_1(N_2 - k_2) = 2N - f(k_1, k_2)$.

(2) $g(N_1 - k_1, N_2 - k_2) = N_2(N_1 - k_1) - N_1(N_2 - k_2) = -g(k_1, k_2)$. □

Lemma 3 (1) (k_1, k_2) *is in* $(A \cup B)$ *if and only if* $(N_1 - k_1, N_2 - k_2)$ *is in* $(C \cup D)$.

(2) (k_1, k_2) *is in* $(A \cup D)$ *if and only if* $(N_1 - k_1, N_2 - k_2)$ *is in* $(B \cup C)$.

(3) (k_1, k_2) *is in* $(C \cup D)$ *if and only if* $(k_1, N_2 - k_2)$ *is in* $(A \cup D)$.

(4) (k_1, k_2) *is in* $(B \cup C)$ *if and only if* $(N_1 - k_1, k_2)$ *is in* $(C \cup D)$.

Proof : We show the only-if part of (1), and other parts can be proved similarly. First, since (k_1, k_2) is in $(A \cup B)$, $f(k_1, k_2) < N$. Then, from Lemma 2-(1), $f(N_1 - k_1, N_2 - k_2) = 2N - f(k_1, k_2) > N$. Finally, because (k_1, k_2) is in $[1, N_1 - 1] \times [1, N_2 - 1]$, so is $(N_1 - k_1, N_2 - k_2)$. Therefore, by definition, $(N_1 - k_1, N_2 - k_2)$ is in $(C \cup D)$. $\square$

Proof of Theorem 1 :

First, for each (k_1, k_2) in $[1, N_1 - 1] \times [1, N_2 - 1]$, neither $f(k_1, k_2) = N$ nor $g(k_1, k_2) = 0$. Thus, the lattice points on the region $[0, N_1 - 1] \times [0, N_2 - 1]$ can be divided into five disjoint groups: E; A; B; C; and D. From the definition of the Ruritanian mapping,

$$
\begin{aligned}
& \{\, k \mid k \in [0, N-1],\ \ k \text{ is an integer} \,\} \\
= \ & \{\, k \mid k = \psi(k_1, k_2),\ (k_1, k_2) \text{ in } E \,\} \cup \\
& \{\, k \mid k = \psi(k_1, k_2),\ (k_1, k_2) \text{ in } (A \cup B) \,\} \cup \\
& \{\, k \mid k = \psi(k_1, k_2),\ (k_1, k_2) \text{ in } (C \cup D) \,\} \\
= \ & \{\, k \mid k = \psi(k_1, k_2),\ (k_1, k_2) \text{ in } E \,\} \cup \\
& \{\, k \mid k = f(k_1, k_2),\ (k_1, k_2) \text{ in } (A \cup B) \,\} \cup \qquad \text{/* Lemma 1-(1) */} \\
& \{\, k \mid k = g(k_1, N_2 - k_2),\ (k_1, k_2) \text{ in } (C \cup D) \,\} \quad \text{/* Lemma 1-(2) */} \\
= \ & \{\, k \mid k = \psi(k_1, k_2),\ (k_1, k_2) \text{ in } E \,\} \cup \\
& \{\, k \mid k = f(k_1, k_2),\ (k_1, k_2) \text{ in } (A \cup B) \,\} \cup \\
& \{\, k \mid k = g(k_1, k_2),\ (k_1, k_2) \text{ in } (A \cup D) \,\} \qquad \text{/* Lemma 3-(3) */} \\
= \ & W \cup U \cup V\,. \qquad \square
\end{aligned}
$$

Lemma 4

$$
x_n = \frac{1}{2} \sum_{k_1=0}^{N_1-1} \sum_{k_2=0}^{N_2-1} Y_{(k_1,k_2)} \left[\cos\left(\frac{\pi(2n+1)\, f(k_1, k_2)}{2N}\right) + \cos\left(\frac{\pi(2n+1)\, g(k_1, k_2)}{2N}\right) \right].
$$

Proof :

$$
\begin{aligned}
& \text{(right-hand side)} \times 2 \\
= \ & \textstyle\sum_{(k_1,k_2)\in E}^{[1]} X_{(k_1,k_2)} \left[\cos\left(\frac{\pi(2n+1) f(k_1,k_2)}{2N}\right) + \cos\left(\frac{\pi(2n+1) g(k_1,k_2)}{2N}\right) \right] + \\
& \textstyle\sum_{(k_1,k_2)\in(A\cup B)}^{[2]} X_{(k_1,k_2)} \cos\left(\frac{\pi(2n+1) f(k_1,k_2)}{2N}\right) + \\
& \textstyle\sum_{(k_1,k_2)\in(C\cup D)}^{[3]} \left(-X_{(N_1-k_1, N_2-k_2)}\right) \cos\left(\frac{\pi(2n+1) f(k_1,k_2)}{2N}\right) + \\
& \textstyle\sum_{(k_1,k_2)\in(A\cup B)}^{[4]} X_{(k_1,k_2)} \cos\left(\frac{\pi(2n+1) g(k_1,k_2)}{2N}\right) +
\end{aligned}
$$

$$
\begin{array}{lll}
& \sum^{[5]}_{(k_1,k_2)\in(C\cup D)} (-X_{(N_1-k_1,N_2-k_2)}) \cos(\frac{\pi(2n+1)\,g(k_1,k_2)}{2N}) + & \\
& \sum^{[6]}_{(k_1,k_2)\in(A\cup D)} X_{(k_1,N_2-k_2)} \cos(\frac{\pi(2n+1)\,f(k_1,k_2)}{2N}) + & \\
& \sum^{[7]}_{(k_1,k_2)\in(B\cup C)} X_{(N_1-k_1,k_2)} \cos(\frac{\pi(2n+1)\,f(k_1,k_2)}{2N}) + & \\
& \sum^{[8]}_{(k_1,k_2)\in(A\cup D)} X_{(k_1,N_2-k_2)} \cos(\frac{\pi(2n+1)\,g(k_1,k_2)}{2N}) + & \\
& \sum^{[9]}_{(k_1,k_2)\in(B\cup C)} X_{(N_1-k_1,k_2)} \cos(\frac{\pi(2n+1)\,g(k_1,k_2)}{2N}) & \\
= & \sum^{[1]}_{(k_1,k_2)\in E} X_{(k_1,k_2)} [\cos(\frac{\pi(2n+1)\,f(k_1,k_2)}{2N}) + \cos(\frac{\pi(2n+1)\,g(k_1,k_2)}{2N})] + & \\
& \sum^{[2]}_{(k_1,k_2)\in(A\cup B)} X^f_{(k_1,k_2)} \cos(\frac{\pi(2n+1)\,f(k_1,k_2)}{2N}) + & \text{/* Lemma 1-(1) */} \\
& \sum^{[3]}_{(k_1,k_2)\in(C\cup D)} (-X^f_{(N_1-k_1,N_2-k_2)}) \cos(\frac{\pi(2n+1)(2N-f(N_1-k_1,N_2-k_2))}{2N}) + & \text{/* Lemmas 3-(1), 1-(1), 2-(1) */} \\
& \sum^{[4]}_{(k_1,k_2)\in(A\cup B)} X^f_{(k_1,k_2)} \cos(\frac{\pi(2n+1)g(k_1,k_2)}{2N}) + & \text{/* Lemma 1-(1) */} \\
& \sum^{[5]}_{(k_1,k_2)\in(C\cup D)} (-X^f_{(N_1-k_1,N_2-k_2)}) \cos(\frac{\pi(2n+1)(-g(N_1-k_1,N_2-k_2))}{2N}) + & \text{/* Lemmas 3-(1), 1-(1), 2-(2) */} \\
& \sum^{[6]}_{(k_1,k_2)\in(A\cup D)} X^g_{(k_1,k_2)} \cos(\frac{\pi(2n+1)f(k_1,k_2)}{2N}) + & \text{/* Lemmas 3-(3), 1-(2) */} \\
& \sum^{[7]}_{(k_1,k_2)\in(B\cup C)} X^g_{(N_1-k_1,N_2-k_2)} \cos(\frac{\pi(2n+1)(2N-f(N_1-k_1,N_2-k_2))}{2N}) + & \text{/* Lemmas 3-(4), 1-(2), 2-(1) */} \\
& \sum^{[8]}_{(k_1,k_2)\in(A\cup D)} X^g_{(k_1,k_2)} \cos(\frac{\pi(2n+1)g(k_1,k_2)}{2N}) + & \text{/* Lemmas 3-(3), 1-(2) */} \\
& \sum^{[9]}_{(k_1,k_2)\in(B\cup C)} X^g_{(N_1-k_1,N_2-k_2)} \cos(\frac{\pi(2n+1)(-g(N_1-k_1,N_2-k_2))}{2N}) & \text{/* Lemmas 3-(4), 1-(2), 2-(2) */} \\
& \text{/* Applying Lemmas 3-(1) and 3-(2), [2] = [3], [8] = [9], [4] + [5] = 0, [6] + [7] = 0. */} & \\
= & \text{(right-hand side of Equation (6))} \times 2 \ . \qquad \square &
\end{array}
$$

Proof of Theorem 2 :

Since

$$
\begin{aligned}
& \cos(\frac{\pi(2n+1)\,f(k_1,k_2)}{2N}) + \cos(\frac{\pi(2n+1)\,g(k_1,k_2)}{2N}) \\
= \; & \cos(\frac{\pi(2n+1)\,(N_2k_1+N_1k_2)}{2N}) + \cos(\frac{\pi(2n+1)\,(N_2k_1-N_1k_2)}{2N}) \\
= \; & 2\,\cos(\frac{\pi(2n+1)N_2k_1}{2N})\,\cos(\frac{\pi(2n+1)N_1k_2}{2N}) \\
= \; & 2\,\cos(\frac{\pi(2n+1)k_1}{2N_1})\,\cos(\frac{\pi(2n+1)k_2}{2N_2})
\end{aligned}
$$

from Lemma 4, we have

$$
x_n = \sum_{k_1=0}^{N_1-1}\sum_{k_2=0}^{N_2-1} Y_{(k_1,k_2)} \cos(\frac{\pi(2n+1)k_1}{2N_1}) \cos(\frac{\pi(2n+1)k_2}{2N_2}) \, . \qquad \square
$$

References

[1] C. S. Burrus. Index mappings for multidimensional formulation of the DFT and convolution. *IEEE Trans. Acoust., Speech, and Signal Processing*, ASSP-25(3):239–242, June 1977.

[2] C. S. Burrus. Bit reverse unscrambling for a radix-2^M FFT. In *Proc. IEEE Int. Conf. Acoust., Speech, and Signal Processing*, pages 1809–1810. IEEE, 1987.

[3] C. S. Burrus. Unscrambling for fast DFT algorithms. *IEEE Trans. Acoust., Speech, and Signal Processing*, 36(7):1086–1087, July 1988.

[4] C. S. Burrus and P. W. Eschenbacher. An in-place, in-order prime factor FFT algorithm. *IEEE Trans. Acoust., Speech, and Signal Processing*, ASSP-29(4):806–817, August 1981.

[5] C. Chakrabarti and J. JáJá. Systolic architectures for the computation of the discrete Hartley and the discrete Cosine transforms based on prime factor decomposition. *IEEE Trans. Comput.*, C-39(11):1359–1368, November 1990.

[6] S. C. Chan and K. L. Ho. Prime factor real-valued Fourier, Cosine, and Hartley transforms. In *Sixth European Signal Processing Conference*, pages 1045–1048, Brussels, Belgium, August 1992.

[7] N. I. Cho and S. U. Lee. DCT algorithms for VLSI parallel implementations. *IEEE Trans. Acoust., Speech, and Signal Processing*, 38(1):121–127, January 1990.

[8] M. T. Heideman. Computation of an odd-length DCT from a real-valued DFT of the same length. *IEEE Trans. Signal Processing*, 40(1):54–61, January 1992.

[9] H. S. Hou. A fast recursive algorithm for computing the discrete Cosine transform. *IEEE Trans. Acoust., Speech, and Signal Processing*, ASSP-35(10):1455–1461, October 1987.

[10] D. P. Kolba and T. W. Parks. A prime factor FFT algorithm using high-speed convolution. *IEEE Trans. Acoust., Speech, and Signal Processing*, ASSP-25(4):281–294, August 1977.

[11] H. T. Kung and C. E. Leiserson. *Introduction to VLSI Systems*, chapter 8.3 Algorithms for VLSI Processor Arrays. Addison-Wesley, Reading, MA, C. Mead and L. Conway edition, 1980.

[12] B. G. Lee. A new algorithm to compute the discrete Cosine transform. *IEEE Trans. Acoust., Speech, and Signal Processing*, ASSP-32(6):1243–1245, December 1984.

[13] B. G. Lee. Input and ouput index mappings for a prime-factor decomposed computation of discrete Cosine transform. *IEEE Trans. Acoust., Speech, and Signal Processing*, 37(2):237–244, February 1989.

[14] P.-Z. Lee and F. Y. Huang. Restructured recursive DCT and DST algorithms. *IEEE Trans. Signal Processing*, accepted, also Technical Report TR-93-001, Institute of Information Science, Academia Sinica, 1993.

[15] P.-Z. Lee and Z. M. Kedem. Mapping nested loop algorithms into multi-dimensional systolic arrays. *IEEE Trans. Parallel Distributed Syst.*, 1:64–76, January 1990.

[16] W. Li. A new algorithm to compute the DCT and its inverse. *IEEE Trans. Signal Processing*, 39(6):1305–1313, June 1991.

[17] D. P. K. Lun, R. Chan, and W. C. Siu. Yet a fast address generation scheme for the computation of prime factor algorithms. In *Proc. IEEE Int. Conf. Acoust., Speech, and Signal Processing*, pages 1499–1502. IEEE, 1990.

[18] D. P. K. Lun and W. C. Siu. On prime factor mapping for the discrete Hartley transform. *IEEE Trans. Signal Processing*, 40(6), June 1992.

[19] M. J. Narasimha and A. M. Peterson. On the computation of the discrete Cosine transform. *IEEE Trans. Commun.*, COM-26:934–936, June 1978.

[20] H. J. Nussbaumer. *Fast Fourier Transform and Convolution Algorithms*, volume 2 of *Springer Series in Information Sciences*. Springer-Verlag, Berlin, Heidelberg, and New York, 1981.

[21] R. K. Rao and P. Yip. *Discrete Cosine Transform: Algorithms, Advantages, and Applications*. Academic Press, Inc., 1990.

[22] J. H. Rothweiler. Implementation of the in-order prime factor transform for variable sizes. *IEEE Trans. Acoust., Speech, and Signal Processing*, ASSP-30(1):105–107, February 1982.

[23] H. V. Sorensen, D. L. Jones, C. S. Burrus, and M. T. Heideman. On computing the discrete Hartley transform. *IEEE Trans. Acoust., Speech, and Signal Processing*, 33(4):1231–1238, October 1985.

[24] H. V. Sorensen, D. L. Jones, M. T. Heideman, and C. S. Burrus. Real-valued fast Fourier transform algorithms. *IEEE Trans. Acoust., Speech, and Signal Processing*, ASSP-35(6):849–863, June 1987.

[25] C. Temperton. A note on prime factor FFT algorithms. *Journal of Computational Physics*, 52:198–204, 1983.

[26] C. Temperton. Implementation of a self-sorting in-place prime factor FFT algorithm. *Journal of Computational Physics*, 58:283–299, 1985.

[27] C. Temperton. Implementation of a prime factor FFT algorithm on CRAY-1. *Parallel Computing*, 6:99–108, 1988.

[28] M. Vetterli and H. J. Nussbaumer. Simple FFT and DCT algorithm with reduced number of operations. *Signal Processing*, 6(4):267–278, 1984.

[29] Z. Wang. A fast algorithm for the discrete Sine transform implemented by the fast Cosine transform. *IEEE Trans. Acoust., Speech, and Signal Processing*, ASSP-30:814–815, 1982.

[30] Z. Wang. Fast algorithms for the discrete W transform and for the discrete Fourier transform. *IEEE Trans. Acoust., Speech, and Signal Processing*, ASSP-32(4):803–816, August 1984.

[31] S. Winograd. On computing the discrete Fourier transform. *Mathematics of Computation*, 32(141):175–199, January 1978.

[32] K. L. Wong, R. Chan, D. P. K. Lun, and W. C. Siu. Efficient address generation for prime factor algorithms. *IEEE Trans. Acoust., Speech, and Signal Processing*, 38(9):1518–1528, September 1990.

[33] K. L. Wong and W. C. Siu. Fast address generation for the computation of prime factor algorithms. In *Proc. IEEE Int. Conf. Acoust., Speech, and Signal Processing*, pages 1091–1094. IEEE, 1989.

[34] K. L. Wong and W. C. Siu. Data routing networks for systolic/pipeline realization of prime factor mapping. *IEEE Trans. Comput.*, 40(9):1072–1074, September 1991.

[35] P. P. N. Yang and M. J. Narasimha. Prime factor decomposition of the discrete Cosine transform. In *Proc. IEEE Int. Conf. Acoust., Speech, and Signal Processing*, pages 772–775. IEEE, 1985.

k_2 k k_1	0	1	2	3	4
0	0	3	6	9	12
1	5	8	11	14	2
2	10	13	1	4	7

Table 1: The Ruritanian mapping: $k = (5k_1 + 3k_2) \bmod 15$, when $N = 15$, and $N_1 = 3$, $N_2 = 5$.

.

n_2 n n_1	0	1	2	3	4
0	0	11	12	6	5
1	10	1	7	13	4
2	9	8	2	3	14

Table 2: The output index mapping when $N = 15$, and $N_1 = 3$, $N_2 = 5$.

Table 1: The Ruritanian mapping: $k = (5k_1 + 3k_2) \bmod 15$, when $N = 15$, and $N_1 = 3$, $N_2 = 5$.

Table 2: The output index mapping when $N = 15$, and $N_1 = 3$, $N_2 = 5$.

Figure 1: The signal-flow graph for implementing the 15-point IDCT, which can be decomposed into the (3×5)-point 2-D IDCT. If signals flow from left to right, it performs the IDCT; if signals flow from right to left, it performs the DCT. Solid lines represent transfer factor 1, while dashed lines represent transfer factor -1. Circles $\circ$ represent adders.

Figure 2: One 3-point IDCT component, one transpose buffer, and one 5-point IDCT component are necessary for computing the row-column evaluation for the 15-point prime-factor IDCT.

Figure 3: The complete data-flow diagram of the mesh-connected systolic array implementation for computing the 15-point prime-factor IDCT. In the lower right corner, each processing element of the mesh-connected systolic array contains a multiply-and-add circuit, which implements $O = I_2 + R * I_1$.

Affiliation

This work was partially supported by the NSC under Grants NSC 81-0408-E-001-505 and NSC 82-0408-E-001-016.

PeiZong Lee is with the Institute of Information Science, Academia Sinica, Taipei, Taiwan, R.O.C.
Fang-Yu Huang is with the National Bureau of Standard, Taipei, Taiwan.

Biography

PeiZong Lee (M'85–S'86–M'90) received the B.A. degree in mathematics from the National Taiwan University, Taiwan, R.O.C., in 1979, the M.S. degree in computer and decision sciences from the National Tsing Hua University, Taiwan, in 1984, and the Ph.D. degree in computer science, at the Courant Institute of Mathematical Sciences, New York University, NY, in 1989.

Between 1984 and 1986, he was a research engineer at Institute of Information Industry, Taiwan, where he joined an artificial intelligence project. Currently, he is an Associate Research Fellow in the Institute of Information Science, Academia Sinica, Taiwan. His research interests are in parallel algorithm design, computer architectures, compilers, and digital signal processing. Dr. Lee is a member of SIAM, the IEEE Computer Society and Circuits & Systems Society.

Fang-Yu Huang is a technical staff in the National Bureau of Standard, Taipei, Taiwan. She received the B.A. degree and the M.S. degree in mathematics from the National Tsing Hua University, Taiwan, in 1987 and 1989, respectively. Between 1989 and 1993, she was an assistant research staff in the Institute of Information Science, Academia Sinica. Her research interests are in functional analysis, game theory, and software engineering.