EURASIP Journal on Applied Signal Processing 2005:12, 1902–1909

Fast DCT-I, DCT-III, and DCT-IV via Moments

J. G. Liu
Key Laboratory of Education Ministry for Image Processing and Intelligence Control, Institute for Pattern Recognition and Artificial Intelligence, Huazhong University of Science & Technology, Wuhan 430074, China
Email: liujg11@126.com

Y. Z. Liu
Department of Computer Science and Engineering, China University of Geosciences, Wuhan 430074, China
Email: yogurt1981@msn.com

G. Y. Wang
Key Laboratory of Education Ministry for Image Processing and Intelligence Control, Institute for Pattern Recognition and Artificial Intelligence, Huazhong University of Science & Technology, Wuhan 430074, China
Email: gywang@mail.hust.edu.cn

Received 3 December 2003; Revised 16 October 2004; Recommended for Publication by Wan-Chi Siu

This paper presents a novel approach to compute DCT-I, DCT-III, and DCT-IV. By using a modular mapping and truncating, DCTs are approximated by linear sums of discrete moments computed fast only through additions. This enables us to use computational techniques developed for computing moments to compute DCTs efficiently. We demonstrate this by applying our earlier systolic solution to this problem. The method can also be applied to multidimensional DCTs as well as their inverses.

Keywords and phrases: discrete cosine transforms, moment, fast transform, systolic array.

1. INTRODUCTION

Discrete cosine transforms (DCTs) are widely used in speech coding and image compression. They resemble Karhunen-Loeve transform for first-order Markov stationary random data and are classified into four groups [1]. Finding fast computational algorithms for DCTs has been a rather active subject [2, 3, 4, 5, 6]. These methods all tried to reduce the amount of multiplications. It is very important to low-power implementations of DCTs on mobile devices that no floating multiplications or less multiplications are needed. At the same time, the parallel hardware methods also have been developed for designing fast DCT processors [7, 8, 9, 10, 11, 12, 13, 14, 15]. Among them the systolic array methods have been given more attentions due to their easy VLSI implementation [7, 8, 9, 10, 11, 12]. Chang and Wang [7] proposed a systolic array approach that only used $\log_2 N$ multipliers for N-point DCT (i.e., N transform samples). It is superior to other systolic array approaches used N or more multipliers [8, 9, 10, 11, 12]. Its disadvantage is that N should be a power of two. This limits the application of the method. We proposed a novel approach to DCT-II [16], which exploited the idea of moment-based all-adder approximation of DCT-II. It could be implemented by both software and systolic array and could be applied to any size N-point DCT-II. There are only $O(\log_2 N/\log_2 \log_2 N)$ multipliers in the systolic arrays. Now we extend our method to DCT-I, DCT-III, and DCT-IV.

In this paper, first the relationship between DCTs and discrete moments is set up and then a moment-based algorithm and a systolic array to perform DCT-III are presented, followed by a complexity analysis. Finally our conclusion is given.

2. RELATIONSHIP BETWEEN DCTs AND DISCRETE MOMENTS

The N-point DCT-III is defined by the equations

$$X(k) = \sum_{n=0}^{N-1} a_n x(n) \cos \frac{\pi(2k+1)n}{2N}, \quad 0 \le k \le N-1, \quad (1)$$

where

$$a_n = \begin{cases} \dfrac{1}{\sqrt{N}} & \text{for } n = 0, \\ \sqrt{\dfrac{2}{N}} & \text{otherwise.} \end{cases} \quad (2)$$

We transform (1) to look for a new method to compute it. For each pair of k and n, $0 \le k, n \le N-1$, by using the properties of cosine functions, there exists an integer i, $0 \le i \le N$, satisfying

$$\cos\frac{\pi(2k+1)n}{2N} = \cos\frac{i\pi}{2N} \quad \text{or} \quad -\cos\frac{i\pi}{2N}. \tag{3}$$

Define $C(k,i)$ and $c(k,i)$ by

$$\begin{aligned} C(k,i) &= \left\{ n \mid \cos\frac{\pi(2k+1)n}{2N} \right. \\ &\qquad \left. = \cos\frac{i\pi}{2N}, 0 \le n \le N-1 \right\}, \\ c(k,i) &= \left\{ n \mid \cos\frac{\pi(2k+1)n}{2N} \right. \\ &\qquad \left. = -\cos\frac{i\pi}{2N}, 0 \le n \le N-1 \right\}, \end{aligned} \tag{4}$$

then define $x_{k,i}$ ($i = 0,1,2,\ldots,N$, $k = 0,1,2,\ldots,N-1$) by

$$x_{k,i} = \begin{cases} \displaystyle\sum_{n\in C(k,i)} a_n x(n) - \sum_{n\in c(k,i)} a_n x(n) & \text{if } C(k,i)\cup c(k,i) \ne \phi, \\ 0 & \text{otherwise.} \end{cases} \tag{5}$$

Thus, by direct substitution, (1) can be rewritten as follows:

$$\begin{aligned} X(k) &= \sum_{n=0}^{N-1} a_n x(n) \cos\frac{\pi(2k+1)n}{2N} \\ &= \sum_{i=0}^{N} x_{k,i} \cos\frac{\pi i}{2N} \\ &= \sum_{i=0}^{N-1} x_{k,i} \cos\frac{\pi i}{2N}, \quad 0 \le k \le N-1. \end{aligned} \tag{6}$$

For $0 \le i \le N-1$,

$$\begin{aligned} \cos\frac{\pi i}{2N} &= 1 - \frac{(\pi i/2N)^2}{2!} \\ &\quad + \cdots + \frac{(-1)^p (\pi i/2N)^{2p}}{(2p)!} + R_i, \end{aligned} \tag{7}$$

where

$$R_i = \cos\frac{(\xi_i + (2p+1)\pi/2)(\pi i/2N)^{2p+1}}{(2p+1)!}, \quad 0 < \xi_i < \frac{\pi i}{2N}. \tag{8}$$

This follows immediately by applying the theorem of extended law [17] of the mean to $\cos(\pi i/2N)$, where R_i is the Taylor remainder term in the form of Lagrange. Substituting (7) into (6) yields

$$\begin{aligned} X(k) &= x_{k,0} + \sum_{i=1}^{N-1} x_{k,i} \sum_{r=0}^{p} \frac{(-1)^r (\pi i/2N)^{2r}}{(2r)!} + R_{k,p} \\ &= x_{k,0} + \sum_{i=1}^{N-1} \sum_{r=0}^{p} \frac{(-1)^r (\pi/2N)^{2r} x_{k,i} i^{2r}}{(2r)!} + R_{k,p} \\ &= x_{k,0} + \sum_{r=0}^{p} \left(\frac{(-1)^r \pi^{2r}}{(2N)^{2r}(2r)!} \right) \sum_{i=1}^{N-1} x_{k,i} i^{2r} + R_{k,p} \\ &= x_{k,0} + \sum_{r=0}^{p} a_r m_{k,2r} + R_{k,p}, \quad 0 \le k \le N-1, \end{aligned} \tag{9}$$

where

$$\begin{aligned} a_r &= \frac{(-1)^r \pi^{2r}}{(2N)^{2r}(2r)!}, \qquad m_{k,2r} = \sum_{i=1}^{N-1} x_{k,i} i^{2r}, \\ R_{k,p} &= \sum_{i=1}^{N-1} x_{k,i} \cos\frac{(\xi_i + (2p+1)\pi/2)(\pi i/2N)^{2p+1}}{(2p+1)!}, \\ &\qquad 0 < \xi_i < \frac{\pi i}{2N}. \end{aligned} \tag{10}$$

If $R_{k,p}$ is ignored, we have

$$X(k) = x_{k,0} + \sum_{r=0}^{p} a_r m_{k,2r}, \quad 0 \le k \le N-1. \tag{11}$$

The absolute value of the error introduced by overlooking $R_{k,p}$ is bounded by

$$\begin{aligned} |R_{k,p}| &= \left| \sum_{i=1}^{N-1} x_{k,i} \cos\frac{(\xi_i + (2p+1)\pi/2)(\pi i/2N)^{2p+1}}{(2p+1)!} \right| \\ &\le \sum_{i=1}^{N-1} \left| x_{k,i} \cos\frac{(\xi_i + (2p+1)\pi/2)(\pi i/2N)^{2p+1}}{(2p+1)!} \right| \\ &\le \max |a_n x(n)|_n \ \frac{(N-1)\pi^{2p+1}}{2^{2p+1}(2p+1)!} \\ &< \max |x(n)|_n \ \frac{\sqrt{2N}\pi^{2p+1}}{2^{2p+1}(2p+1)!}. \end{aligned} \tag{12}$$

From (12), it is clear that $R_{k,p}$ converges to zero rapidly and uniformly. For example, for $\max|x(n)| \le 256$, ($0 \le n \le N-1$), $N \le 2048$, $p = 9$, $|R_{k,p}| \le 7.17 \times 10^{-10}$. This error can satisfy the accuracy requirement of most applications by computing only ten terms. When deciding the value of p, we should compromise between computation accuracy and complexity [18]. Furthermore, we can prove that the least upper bound of p is not more than $O(\log_2 N/\log_2\log_2 N)$ as N tends to infinity [16, 19].

The DCT-I is defined by the equations

$$X(k) = c_k \sum_{n=0}^{N-1} a_n x(n) \cos\frac{nk\pi}{N}, \quad 0 \le k \le N-1, \tag{13}$$

where

$$c_k, a_n = \begin{cases} \dfrac{1}{\sqrt{N}} & \text{for } k, n = 0, \\ \sqrt{\dfrac{2}{N}} & \text{otherwise.} \end{cases} \tag{14}$$

By using the properties of cosine functions, we have

$$\begin{aligned} &\cos\frac{kn\pi}{N} = \cos\frac{i\pi}{N} \quad \text{or} \quad -\cos\frac{i\pi}{N}, \\ &0 \le i \le N-1,\ 0 \le n,\ k \le N-1. \end{aligned} \tag{15}$$

For every pair of k and i ($i = 0, 1, 2, \ldots, N-1$, $k = 0, 1, 2, \ldots, N-1$), define $C(k,i)$ and $c(k,i)$ by

$$\begin{aligned} C(k,i) &= \left\{ n \mid \cos\frac{kn\pi}{N} = \cos\frac{i\pi}{N}, 0 \le n \le N-1 \right\}, \\ c(k,i) &= \left\{ n \mid \cos\frac{kn\pi}{N} = -\cos\frac{i\pi}{N}, 0 \le n \le N-1 \right\}, \end{aligned} \tag{16}$$

then define $x_{k,i}$ ($i = 0, 1, 2, \ldots, N-1$; $k = 0, 1, 2, \ldots, N-1$) by

$$x_{k,i} = \begin{cases} \displaystyle\sum_{n \in C(k,i)} c_k a_n x(j) - \sum_{n \in c(k,i)} c_k a_n x(j) & \text{if } C(k,i) \cup c(k,i) \ne \phi, \\ 0 & \text{otherwise.} \end{cases} \tag{17}$$

Thus, by direct substitution, (13) can be rewritten as follows:

$$X(k) = \sum_{i=0}^{N-1} x_{k,i} \cos\frac{i\pi}{N}, \quad 0 \le k \le N-1. \tag{18}$$

The DCT-IV is defined by the equations

$$X(k) = \sum_{n=0}^{N-1} x(n) \cos\frac{(2k+1)(2n+1)}{4N}, \quad 0 \le k \le N-1. \tag{19}$$

We have

$$\begin{aligned} &\cos\frac{(2k+1)(2n+1)\pi}{4N} = \cos\frac{i\pi}{4N}, \\ &-\cos\frac{i\pi}{4N}, \quad \sin\frac{i\pi}{4N}, \quad \text{or} \\ &-\sin\frac{i\pi}{4N}, \quad 0 \le i \le N,\ 0 \le n,\ k \le N-1. \end{aligned} \tag{20}$$

For every pair of k and i ($i = 0, 1, \ldots, N$, $k = 0, 1, \ldots, N-1$), define $S(k,i)$, $s(k,i)$, $C(k,i)$, $c(k,i)$ by

$$\begin{aligned} S(k,i) &= \left\{ n \mid \cos\frac{(2k+1)(2n+1)\pi}{4N} = \sin\frac{i\pi}{4N},\ 0 \le n \le N-1 \right\}, \\ s(k,i) &= \left\{ n \mid \cos\frac{(2k+1)(2n+1)\pi}{4N} = -\sin\frac{i\pi}{4N},\ 0 \le n \le N-1 \right\}, \\ C(k,i) &= \left\{ n \mid \cos\frac{(2k+1)(2n+1)\pi}{4N} = \cos\frac{i\pi}{4N},\ 0 \le n \le N-1 \right\}, \\ c(k,i) &= \left\{ n \mid \cos\frac{(2k+1)(2n+1)\pi}{4N} = -\cos\frac{i\pi}{4N},\ 0 \le j \le N-1 \right\}, \end{aligned} \tag{21}$$

then define $x_{k,i}$ and $y_{k,i}$ ($i = 0, 1, \ldots, N$, $k = 0, 1, \ldots, N-1$) by

$$\begin{aligned} x_{k,i} &= \begin{cases} \displaystyle\sum_{n \in S(k,i)} x(n) - \sum_{n \in s(k,i)} x(n) & \text{if } S(k,i) \cup s(k,i) \ne \phi, \\ 0 & \text{otherwise,} \end{cases} \\ y_{k,i} &= \begin{cases} \displaystyle\sum_{n \in C(k,i)} x(n) - \sum_{n \in c(k,i)} x(n) & \text{if } C(k,i) \cup c(k,i) \ne \phi, \\ 0 & \text{otherwise.} \end{cases} \end{aligned} \tag{22}$$

Thus,

$$X(k) = \sum_{i=1}^{N} x_{k,i} \sin\frac{i\pi}{4N} + \sum_{i=0}^{N} y_{k,i} \cos\frac{i\pi}{4N}, \quad 0 \le k \le N-1. \tag{23}$$

Equations (18) and (23) can be transformed into the same forms of (11) as (6). In other words, other two forms of DCT could also be computed through computation of moments. They have the same error levels and the same convergence features. The discussions in the following sections only concern DCT-III, but they are also applicable to the others.

Equations (11) are the formulas of the relationship between DCTs and discrete moments. According to it, the computation of $X(k)$ requires the generation of the $2p$th-order moments of the transformed data sequence $x_{k,i}$, followed by computing the dot product between this and a constant vector (a_r) and an addition with $x_{k,0}$. The efficiency

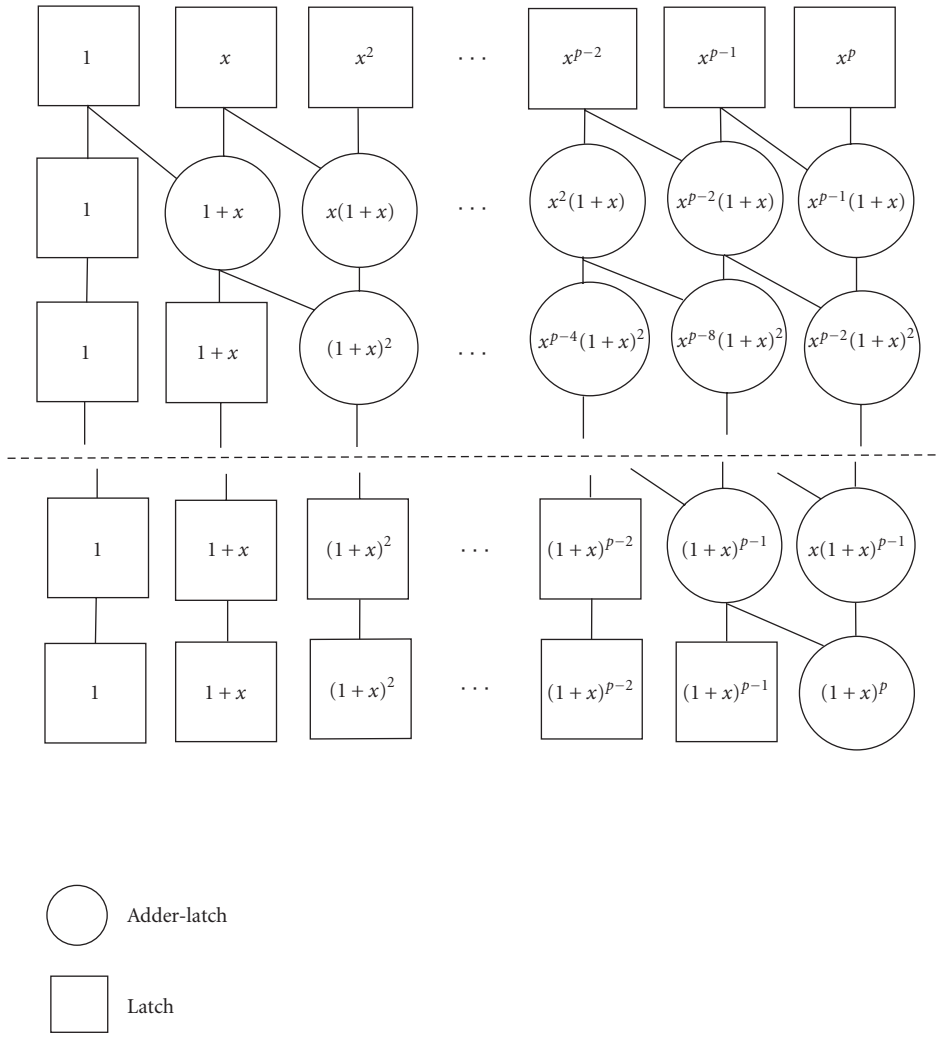

Figure 1: The p-network.

of the procedure largely depends on the efficiency in deriving the moments. Some approaches to moments of fast computation that involve just additions have been presented [20, 21, 22, 23].

3. ALGORITHMS FOR COMPUTING 1D MOMENTS

We defined the p-network in [20, 21]. The p-network shown in Figure 1 represents a map of transforming the vector $(1, x, x^2, \ldots, x^{p-1}, x^p)$ into $(1, (1+x), (1+x)^2, \ldots, (1+x)^{p-1}, (1+x)^p)$. It is denoted by F_p, that is,

$$F_p(1, x, x^2, \ldots, x^{p-1}, x^p) = (1, (1+x), (1+x)^2, \ldots, (1+x)^{p-1}, (1+x)^p). \tag{24}$$

The equations below follow immediately:

$$F_p(1, 1, 1, \ldots, 1, 1) = (1, 2, 4, \ldots, 2^{p-1}, 2^p),$$
$$F_p(a, a, a, \ldots, a, a) = (a, 2a, 4a, \ldots, 2^{p-1}a, 2^p a). \tag{25}$$

The equations below can then be verified by inputting data into the p-network:

$$F_p(a, ax, ax^2, \ldots, ax^{p-1}, ax^p) = (a, a(1+x), a(1+x)^2, \ldots, a(1+x)^{p-1}, a(1+x)^p), \tag{26}$$

$$F_p(a+b, a+b, a+b, \ldots, a+b, a+b) = F_p(a, a, a, \ldots, a, a) + F_p(b, b, b, \ldots, b, b), \tag{27}$$

$$\begin{aligned} F_p^2(1, x, x^2, \ldots, x^{p-1}, x^p) &= F_p(F_p(1, x, x^2, \ldots, x^{p-1}, x^p)) \\ &= F_p(1, (1+x), (1+x)^2, \ldots, (1+x)^{p-1}, (1+x)^p) \\ &= (1, (2+x), (2+x)^2, \ldots, (2+x)^{p-2}, (2+x)^p) \end{aligned} \tag{28}$$

and, in general,

$$\begin{aligned} &F_p^{n-1}(1, x, x^2, \ldots, x^{p-1}, x^p) \\ &\quad = F_p(\ldots F_p(1, x, x^2, \ldots, x^{p-1}, x^p)\ldots) \\ &\quad = (1, (n-1+x), (n-1+x)^2, \ldots, \\ &\qquad (n-1+x)^{p-2}, (n-1+x)^p). \end{aligned} \tag{29}$$

By substitution,

$$\begin{aligned} &F_p^{n-1}(1, 1, 1, \ldots, 1, 1) \\ &\quad = F_p(\ldots F_p(1, 1, 1, \ldots, 1, 1)\ldots) \\ &\quad = (1, n, n^2, \ldots, n^{p-2}, n^p), \\ &F_p^{n-1}(a, a, a, \ldots, a, a) = (a, na, n^2a, \ldots, n^{p-1}a, n^pa). \end{aligned} \tag{30}$$

Let $\mathbf{a_i} = (a_i, a_i, a_i, \ldots, a_i)$, $i = 1, 2, 3, \ldots, n$, and $\mathbf{a_i}$ is a $(p+1)$-dimensional vector; then

$$\begin{aligned} &F_p(F_p(\mathbf{a_n}) + \mathbf{a_{n-1}}) \\ &\quad = F_p(F_p(\mathbf{a_n})) + F_p(\mathbf{a_{n-1}}) \\ &\quad = F_p^2(\mathbf{a_n}) + F_p(\mathbf{a_{n-1}}) \quad \text{(from (27))}. \end{aligned} \tag{31}$$

Generally,

$$\begin{aligned} &F_p(F_p \ldots F_p(F_p(F_p(\mathbf{a_n}) + \mathbf{a_{n-1}}) + \mathbf{a_{n-2}}) + \cdots \mathbf{a_2}) + \mathbf{a_1} \\ &\quad = F_p^{n-1}(\mathbf{a_n}) + F_p^{n-2}(\mathbf{a_{n-1}}) + F_p^{n-3}(\mathbf{a_{n-1}}) \\ &\qquad + \cdots + F_p^2(\mathbf{a_3}) + F_p(\mathbf{a_2}) + \mathbf{a_1} \\ &\quad = \left(\sum_{i=1}^n a_i, \sum_{i=1}^n a_i i, \sum_{i=1}^n a_i i^2, \ldots, \sum_{i=1}^n a_i i^{p-1}, \sum_{i=1}^n a_i i^p\right). \end{aligned} \tag{32}$$

For emphasis, (32) is rewritten as

$$\begin{aligned} &\text{Moment}_p(a_n, a_{n-1}, a_{n-2}, \ldots, a_2, a_1) \\ &\quad = F_p(F_p \ldots F_p(F_p(F_p(\mathbf{a_n}) + \mathbf{a_{n-1}}) + \mathbf{a_{n-2}}) \\ &\qquad + \cdots \mathbf{a_2}) + \mathbf{a_1} \\ &\quad = \left(\sum_{i=1}^n a_i, \sum_{i=1}^n a_i i, \sum_{i=1}^n a_i i^2, \ldots, \sum_{i=1}^n a_i i^{p-1}, \sum_{i=1}^n a_i i^p\right). \end{aligned} \tag{33}$$

Equation (32) can be proved by mathematical induction. These components of the resultant vector are known as 1D moments. To compute these 1D moments, F_p is used $(n-1)$ times in the iteration procedure except for the $(n-1)$ additions of $(p+1)$-dimensional vectors. As F_p only involves $p(p+1)/2$ additions, Moment_p only requires $(n-1)p(p+1)/2 + (n-1)(p+1) = (p+1)(p+2)(n-1)/2$ additions. The algorithm for computing 1D moments is summarized in Algorithm 1.

```
Subroutine Moment_p(a_n, a_{n-1}, a_{n-2}, ..., a_2, a_1)
Input initial p, a_1, a_2, a_3, ..., a_{n-1}, a_n
Moment = a_n
for i = 1 to n − 1
  Perform F_p(Moment)
  Moment = F_p(Moment) + a_{n−i}
end for
```

Algorithm 1: The algorithm for computing 1D moments.

```
Input initial p, N, x(0), x(1), ..., x(N − 1)
Perform preprocessing
for k = 0 to N − 1
  Compute Moment_{2p}(x_{k,N−1}, x_{k,N−2}, ..., x_{k,1})
  X(k) = 0
  for r = 0 to p
    X(k) = X(k) + a_r m_{k,2r}
  end for
  X(k) = X(k) + x_{k,0}
end for
```

Algorithm 2: The algorithm for computing $X(k)$.

4. ALGORITHMS FOR COMPUTING 1D DCT-III

From Section 2, our fast discrete cosine transform includes three steps:

(1) computing $x_{k,i}$ $(i = 0, 1, 2, \ldots, N-1;\ k = 0, 1, 2, \ldots, N-1)$ and a_r $(r = 0, 1, 2, \ldots, p)$;
(2) computing $m_{k,2r}$ $(r = 0, 1, 2, \ldots, p;\ k = 0, 1, 2, \ldots, N-1)$;
(3) computing $X(k)$ $(k = 0, 1, 2, \ldots, N-1)$.

Step (1) constitutes a preprocessing step. The algorithm of computing $X(k)$ can be described in Algorithm 2.

Next we analyze the complexity of the algorithm. In preprocessing, the computation of a_r and the index set $C(k, i)$ and $c(k, i)$ can be performed once and for all after N and p are given. In a real-time system, these values can be precomputed and reused. The computation of $x_{k,i}$ is bounded by $N(N-1)$ additions/subtractions and $2N$ multiplications (in fact, the number of addition/subtraction required is much less than this value because only a few $x_{k,i}$'s are equal to $x(0)/\sqrt{N} + \sqrt{2}(\pm x(1) \pm x(2) \pm \cdots \pm x(N-1))/\sqrt{N}$). Computing Moment_p N times requires $(2p+1)(2p+2)(N-2)N/2$ additions. Finally computing $X(k)$ $(k = 0, 1, 2, \ldots, N-1)$ involves $(p+1)N$ additions and pN multiplications ($a_0 = 1$, there is no multiplication when computing $a_0 m_{k,0}$). So there are altogether fewer than $N(N-1) + (2p+1)(2p+2)(N-2)N/2 + (p+1)N$ additions/subtractions and $(p+2)N$ multiplications in the algorithms.

5. SYSTOLIC ARRAY FOR COMPUTING 1D DCT-III

The systolic array for fast DCT is shown in Figure 2. The system is constructed by inserting p multipliers and $p+1$ adders into the array for computing 1D moments constituting concatenation of $N-1$ $2p$-networks [20] and by supplementing $N-1$ preprocessor arrays constituting a special array to

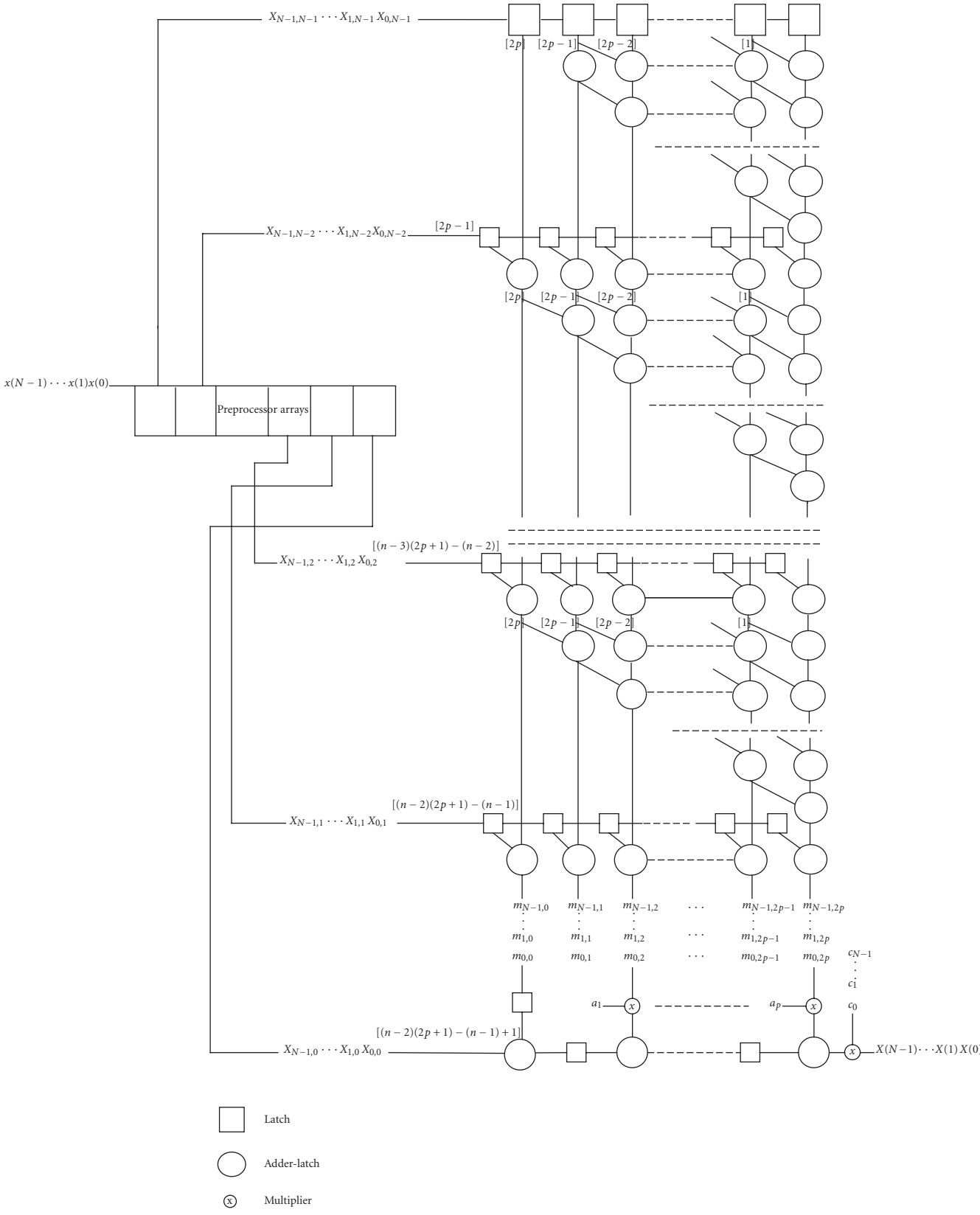

Figure 2: The systolic array for 1D DCT-III.

Table 1: Comparison of some systolic arrays for N-point DCT.

Operator types	Y.-T. Chang [7]	Totzek [8]	Cho [9]	L.-W. Chang [10]	Wang [11]	Proposed
Number of multipliers	$\log_2 N$	N	$4(N+1)$	$2N$	$2N$	p $11(N \le 2^{50})$
Number of adders	$3\log_2 N$	N	$4(N+1)$	$2N$	$2N$	$2p^2+2p+1$ $265(N \le 2^{50})$

implement $x_{k,i}$ $(i,k = 0,1,\ldots,N-1)$ [16]. The numbers in the square brackets in Figure 2 denote time delays to keep operation synchronization. The data are input into and output from the array every clock cycle. So the total execution time of the computing procedure is equal to

$$N + (2p+1)(N-1) + 2 + N - 1 = (2p+3)N - 2p \quad (34)$$

clock cycles. In the above equation, N is for preprocessing, the term $(2p+1)(N-1)$ for computing moments, 2 for the dot product, and finally $N-1$ for collecting all $X(k)$ $(k = 1,2,\ldots,N-1)$.

A special linear array to implement $x_{k,i}$ $(i,k = 0,1,\ldots, N-1)$ is designed as a preprocessor. Each original element $x(r)$ is tagged with a rank i (if $r \in C(k,i)$ or $c(k,i)$) and an operated sign (+, if $r \in C(k,i)$; −, otherwise), before it is sent into the array. These are sent into the linear array one element at a time. An element percolates from left to right until it reaches a cell in the array whose position is identical to its rank. When this happens, the value is accumulated in that cell (+, ⇒ addition; −, ⇒ subtraction). A cell in a linear array contains an adder/subtractor only if the corresponding partition $C_{k,i} \cup c_{k,i}$ has a size greater than 1.

The systolic arrays possess perfect properties of modularity, regularity, and scalability so that the size of the systolic array can be reduced flexibly to meet practical requirements. For example, when N is rather large, the systolic array can be composed by only one $2p$-network that produces moments [20], p multipliers and adder-latches. The preprocessing of data could be done by software or other processors. So there are altogether $2p^2+2p+1$ adder-latches and p multipliers in the systolic array. For most practical cases $(N \le 2^{50})$, p can get value of 11, so only 265 adder-latches and 11 multipliers are needed in the array.

A comparison of the proposed array with other systolic arrays described in [7, 8, 9, 10, 11] is shown in Table 1. The table lists the numbers of multipliers and adders used in the methods. It shows clearly our method and Chang's [7] require much fewer multipliers and adders than others, but Chang's method is suitable just for N of a power of two. When N is larger than 2^{11}, Chang's array should use more multipliers than ours. However, in most practical cases $(N \le 2^{50})$, ours uses more adders than Chang's.

Due to the separability of the DCT kernel, the 2D DCT-I, DCT-III, and DCT-IV can be computed using the row-column method. The algorithm computes a 2D DCT by taking 1D DCT rowwise and columnwise. The algorithms and the systolic arrays described above can be applied to 2D DCTs. The total execution time of the computing 2D DCT using the systolic array is $2N^2+2(p+1)N(2p-1)$ clock cycles [16]. The method can also be extended to multidimensional DCT (more than two dimensions). The higher the dimension, the more obvious the advantages of the method are. It is provable by mathematical induction that the execution time of the systolic array is $O(N^k)$ for kD $(k \ge 2)$ DCTs.

The approach is also applicable to DCTs inverse.

6. CONCLUSION

We have been able to establish for the first time the link between fast algorithms for DCTs and moments computations. Such novel approach enabled the computation of DCT-I, DCT-III, and DCT-IV more efficiently. By using a modular mapping, DCTs can be approximated by linear sums of discrete moments, and earlier computational efficient techniques developed for computing moments can hence be used to compute DCTs. The new method has several noteworthy advantages. Most of multiplications for the transform are replaced by simple additions, and trigonometric functions implemented by simple polynomial functions. The amount of multiplication for DCTs using our new method is $O(N\log_2 N/\log_2\log_2 N)$ *and is superior to that of* $O(N\log_2 N)$ needed in conventional fast cosine transforms. Furthermore, the algorithm can be directly realized in the form of a systolic array consisting of only latches, adder-latches and a few multipliers for real-time applications. This new method can be easily extended to the computation of multidimensional DCTs and their inverses. The computational complexity for k-dimensional DCTs is $O(n^k)$. The execution time of the systolic array is only $O(N\log_2 N/\log_2\log_2 N)$ for 1-dimensional DCTs and $O(N^k)$ for k-dimensional DCTs $(k \ge 2)$. The issues of convergence and errors of the new approach have also been critically examined in this paper.

With the evolution of electronic technology, it is or will soon be viable to implement discrete transforms directly in chips or embedded systems. Our solution strategy is to develop cost-effective designs for discrete moment computation as a basic building block implementable in both hardware [14] and software means. Then these building blocks can be extended to produce other useful discrete transforms for various applications.

ACKNOWLEDGMENT

This project is supported in part by the National Science Foundation of China under Grants 69775011 and 60372066.

REFERENCES

[1] Z. Wang, "A prime factor fast W transform algorithm," *IEEE Trans. Signal Processing*, vol. 40, no. 9, pp. 2361–2368, 1992.

[2] B. G. Lee, "A new algorithm to compute the discrete cosine transform," *IEEE Trans. Acoust., Speech, Signal Processing*, vol. 32, no. 6, pp. 1243–1245, 1984.

[3] W. Li, "A new algorithm to compute the DCT and its inverse," *IEEE Trans. Signal Processing*, vol. 39, no. 6, pp. 1305–1313, 1991.

[4] E. Feig and S. Winograd, "Fast algorithms for the discrete cosine transform," *IEEE Trans. Signal Processing*, vol. 40, no. 9, pp. 2174–2193, 1992.

[5] H. S. Hou, "A fast recursive algorithm for computing the discrete cosine transform," *IEEE Trans. Acoust., Speech, Signal Processing*, vol. 35, no. 10, pp. 1455–1461, 1987.

[6] J. Makhoul, "A fast cosine transform in one and two dimensions," *IEEE Trans. Acoust., Speech, Signal Processing*, vol. 28, no. 1, pp. 27–34, 1980.

[7] Y.-T. Chang and C.-L. Wang, "A new fast DCT algorithm and its systolic VLSI implementation," *IEEE Trans. Circuits Syst. II*, vol. 44, no. 11, pp. 959–962, 1997.

[8] U. Totzek and F. Mattiesen, "Two-dimensional discrete cosine transform with linear arrays," in *Proc. 3rd International Conference on Systolic Arrays (Systolic Array Processors)*, J. McCanny, J. McWhirter, and E. Swartzlander Jr., Eds., pp. 388–397, Killarney, Kerry, Ireland, May 1989.

[9] N. I. Cho and S. U. Lee, "DCT algorithms for VLSI parallel implementations," *IEEE Trans. Acoust., Speech, Signal Processing*, vol. 38, no. 1, pp. 121–127, 1990.

[10] L.-W. Chang and M.-C. Wu, "A unified systolic array for discrete cosine and sine transforms," *IEEE Trans. Signal Processing*, vol. 39, no. 1, pp. 192–194, 1991.

[11] C.-L. Wang and C.-Y. Chen, "High-throughput VLSI architectures for the 1-D and 2-D discrete cosine transforms," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 5, no. 1, pp. 31–40, 1995.

[12] N. Rama Murthy and M. N. S. Swamy, "On the real-time computation of DFT and DCT through systolic architectures," *IEEE Trans. Signal Processing*, vol. 42, no. 4, pp. 988–991, 1994.

[13] S. Yu and E. E. Swartziander Jr., "DCT implementation with distributed arithmetic," *IEEE Trans. Comput.*, vol. 50, no. 9, pp. 985–991, 2001.

[14] M. P. Leong and P. H. W. Leong, "A variable-radix digit-serial design methodology and its application to the discrete cosine transform," *IEEE Trans. VLSI Syst.*, vol. 11, no. 1, pp. 90–104, 2003.

[15] Y. H. Hu and Z. Wu, "An efficient CORDIC array structure for the implementation of discrete cosine transform," *IEEE Trans. Signal Processing*, vol. 43, no. 1, pp. 331–336, 1995.

[16] J. G. Liu, H. F. Li, F. H. Y. Chan, and F. K. Lam, "Fast discrete cosine transform via computation of moments," *Journal of VLSI Signal Processing*, vol. 19, no. 3, pp. 257–268, 1998.

[17] J. M. H. Olmsted, *Real Variables*, Appleton-Century-Crofts, New York, NY, USA, 1956.

[18] L. P. Yaroslavsky and M. Eden, *Fundamentals of Digital Optics*, Springer-Verlag, New York, NY, USA, 1996.

[19] J. G. Liu, H. F. Li, F. H. Y. Chan, and F. K. Lam, "A novel approach to fast discrete Fourier transform," *Journal of Parallel and Distributed Computing*, vol. 54, no. 1, pp. 48–58, 1998.

[20] F. H. Y. Chan, F. K. Lam, H. F. Li, and J. G. Liu, "An all adder systolic structure for fast computation of moments," *Journal of VLSI Signal Processing*, vol. 12, no. 2, pp. 159–175, 1996.

[21] J. G. Liu, F. H. Y. Chan, F. K. Lam, and H. F. Li, "A new approach to fast calculation of moments of 3-D gray level images," *Parallel Computing*, vol. 26, no. 6, pp. 805–815, 2000.

[22] M. Hatamian, "A real-time two-dimensional moment generating algorithm and its single chip implementation," *IEEE Trans. Acoust., Speech, Signal Processing*, vol. 34, no. 3, pp. 546–553, 1986.

[23] K. Chen, "Efficient parallel algorithms for the computation of two-dimensional image moments," *Pattern Recognition*, vol. 23, no. 1-2, pp. 109–119, 1990.

J. G. Liu received his B.S. degree in mathematics from the Wuhan University of Technology in 1982, his M.S. degree in computer science from the Huazhong University of Science and Technology in 1984, and his Ph.D. degree in electrical and electronic engineering from the University of Hong Kong in 1996, respectively. He was a Visiting Scholar in the Medical Image Processing Group, Department of Radiology, University of Pennsylvania, from December 1998 to July 2004. He is currently a Professor and Deputy Director of the Key Laboratory of State Education Ministry for Image Processing and Intelligent Control, Huazhong University of Science and Technology, China. His interests include signal processing, image processing, parallel algorithm and structure, and pattern recognition.

Y. Z. Liu received his B. Eng. degree in computer science and engineering and his B.A. degree in English literature in 2003 and 2004, respectively, from China University of Geosciences, Wuhan, China. He is interested in computer vision, computer algorithm and architecture, neural network, and artificial intelligence.

G. Y. Wang received the B.S. and M.S. degrees in electrical engineering and pattern recognition from Huazhong University of Science and Technology, China, in 1988 and 1992, respectively. He joined the Institute of Pattern Recognition and Artificial Intelligence, Huazhong University of Science and Technology, China, in 1992. Since 1999, he has been a Full Professor at the Institute of Pattern Recognition and Artificial Intelligence, Huazhong University of Science and Technology, China. His research interests include automatic target detection, image and video compression, biometrics security, image parallel computing, and integrated testing techniques.