Transactions Briefs

A Novel CORDIC-Based Array Architecture for the Multidimensional Discrete Hartley Transform

Jiun-In Guo, Chi-Min Liu, and Chein-Wei Jen

***Abstract*— In this paper, a coordinate rotation digital computer (CORDIC)-based array architecture is presented for computing the multidimensional (M-D) discrete Hartley transform (DHT). Since the kernel of the M-D DHT is inseparable, the M-D DHT problems cannot be computed through the 1-D DHT's directly. Bracewell *et al.* have presented an algorithm for the M-D DHT through the 1-D DHT's. The existing hardware architectures have been designed using this algorithm. However, the postprocessing required in the algorithm leads to high hardware overhead. This paper presents a new algorithm to compute M-D DHT through a special 1-D transform which is derived through considering both the separable computation and the efficient implementation with CORDIC architectures. This algorithm provides a direct way to compute the M-D DHT separably through 1-D transforms with simpler postprocessing than that in Bracewell's approach. Also, the algorithm exploits the symmetry of the triangular functions to reduce the computational complexity. Using this algorithm, we design an array architecture for the M-D DHT. This architecture features a systolic computing style, PE's with a CORDIC structure, low I/O cost, and the encapsulated new M-D DHT algorithm.**

I. Introduction

The discrete Hartley transform (DHT) [1], [2] is considered to be a good alternative to the discrete Fourier transform (DFT) since it requires only real number computations, which are much simpler than the complex number computations required in the DFT. Therefore, the DHT has been widely applied in the field of digital signal processing. Though computing the DHT involves only real number computations, the M-D DHT is still computation-intensive. Thus, a dedicated VLSI implementation for the M-D DHT is necessary to provide a high-speed, low-cost means of computing the M-D DHT.

A. The Difficulty in Computing the M-D DHT

The M-D DHT of the input data $\{x(n_1,n_2,\cdots,n_M), n_1,n_2,\cdots, n_M = 0,1,\cdots,N-1\}$ is defined as

$$H(k_1,k_2,\cdots,k_M) = \sum_{n_M=0}^{N-1}\cdots\sum_{n_2=0}^{N-1}\sum_{n_1=0}^{N-1} x(n_1,n_2,\cdots,n_M) \cdot \mathrm{cas}(\varphi_{n_1k_1}+\varphi_{n_2k_2}+\cdots\varphi_{n_Mk_M}) \quad ; k_1,k_2,\cdots,k_M = 0,1,\cdots,N-1 \tag{1}$$

where $\varphi_{n_ik_i} = \frac{2\pi n_i k_i}{N}$ and $\mathrm{cas}(\theta) = \cos(\theta)+\sin(\theta)$. A major difficulty in computing (1) is that the kernel of the M-D DHT is inseparable, i.e., $\mathrm{cas}(\theta_1+\cdots+\theta_M) \neq \mathrm{cas}(\theta_1)\times\cdots\times\mathrm{cas}(\theta_M)$. This inseparability induces heavy computational load in realizing the M-D DHT problems. In the following, we discuss the 2-D transform as an example. A 2-D $N\times N$ separable transform, like the DFT, can be decomposed into $2N$ identical 1-D N-point transforms for reducing the computational complexity from $O(N^4)$ to $O(2N^3)$ [3]. The inseparability of the 2-D DHT, on the other hand, implies that its computational complexity should be $O(N^4)$. To reduce this complexity, Bracewell *et al.* [4] introduced a temporary variable $T(k_1,k_2)$, defined as

$$T(k_1,k_2) = \sum_{n_1=0}^{N-1}\sum_{n_2=0}^{N-1} x(n_1,n_2)\cdot\mathrm{cas}(\frac{2\pi n_1 k_1}{N})\cdot\mathrm{cas}(\frac{2\pi n_2 k_2}{N}) \quad ; k_1,k_2 = 0,1,\cdots,N-1. \tag{2}$$

The 2-D DHT, denoted as $H(k_1,k_2)$, can then be calculated from $T(k_1,k_2)$ through

$$H(k_1,k_2) = \frac{1}{2}\cdot[T(k_1,k_2)+T(N-k_1,k_2) + T(k_1,N-k_2) - T(N-k_1,N-k_2)]. \tag{3}$$

Note that the kernel in (2) has been separated, and hence the $T(k_1,k_2)$ can be computed through $2N$ 1-D N-point DHT's. The existing hardware designs for the 2-D DHT in the literature [5]–[8] are based on (2) and (3). However, these designs are used to realize only the $T(k_1,k_2)$ in (2), and they do not consider the issue of how to efficiently realize (3) to obtain $H(k_1,k_2)$. Note that the complexity of the postadditions indicated in (3) is $O(N^M)$ if a M-D DHT problem is considered. This complexity increases dramatically as M increases. From the perspective of VLSI implementation, the $O(N^2)$ additions and the poor data locality of $T(k_1,k_2)$ in (3) impose much overhead in hardware cost, computation time, and control circuits. Moreover, as higher dimensional DHT problems are considered, this overhead will increase exponentially.

B. The Design Concepts and Main Results of This Paper

Being different from the conventional approach which computes the M-D transforms using the same kind of 1-D transforms, our basic concept is that any kind of 1-D transforms can be used to compute a M-D transform if there are advantages in doing so. Following this concept, we intend to derive a kind of 1-D transform to compute the M-D DHT through considering both the separable computation and the efficient hardware implementation.

The implementation of triangular functions and multiplications is a key consideration in the design of VLSI architectures for the DHT. The coordinate rotation digital computer (CORDIC) structure is considered to be a good choice for these operations [9], [10]. Fig. 1 shows the basic CORDIC processor operating in the circular coordinate system, which realizes the following operations

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} \cos(\psi) & \sin(\psi) \\ -\sin(\psi) & \cos(\psi) \end{bmatrix} \cdot \begin{bmatrix} x \\ y \end{bmatrix}. \tag{4}$$

Manuscript received May 11, 1993; revised April 15, 1994. This work was supported by the NSC, Taiwan, R.O.C. under Grant NSC 82-0404-E009-225. This paper was recommended by Associate Editor P. Agathoklis.

J.-I. Guo is with the Computer & Communication Research Laboratories, Industrial Technology Research Institute, Chutung, Hsinchu, Taiwan, R.O.C.

C.-M. Liu is with the Department of Computer Science and Information Engineering, National Chiao Tung University, Hsinchu, Taiwan, R.O.C.

C.-W. Jen is with the Institute of Electronics, National Chiao Tung University, Hsinchu, Taiwan, R.O.C.

IEEE Log Number 9409261.

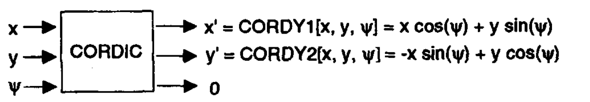

x
y
ψ
CORDIC
x' = CORDY1[x, y, ψ] = x cos(ψ) + y sin(ψ)
y' = CORDY2[x, y, ψ] = -x sin(ψ) + y cos(ψ)
0

Fig. 1. The CORDIC processor operating in the circular coordinate system (Ψ denotes the rotation angle).

The terms (x, y) and (x', y') denote the coordinates before and after rotation, respectively. ψ denotes the rotation angle. Note that in Fig. 1 x' and y' are expressed as combinations of the terms $x, y, \cos(\psi)$, and $\sin(\psi)$. The special property of CORDIC is that the same hardware can support different output functions by simply changing the assignments of the input data. For example, it supports the functions of $f \cdot [\cos(\psi) + \sin(\psi)]$ and $f \cdot [\cos(\psi) - \sin(\psi)]$ if we assign $(x, y) = (f, f)$ and supports the functions of $f \cdot cos(\psi)$ and $f \cdot sin(\psi)$ if we assign $(x, y) = (0, f)$. This property allows large flexibility in deriving the 1-D transform. Using this property, we derive a kind of 1-D transform to compute the M-D DHT separably. This algorithm only needs an addition as the postprocessing which is simpler than that in the Bracewell's approach. Besides, we exploit the symmetry of the triangular functions to reduce the complexity of the derived 1-D transform.

Based on this algorithm, we design an array architecture for the M-D DHT. This architecture consists of M linear systolic arrays and (M-1) transpose RAM's. Each array consists of $\frac{N}{2}$ processing elements (PE's), which are composed of CORDIC processors, ROM's, and adders. The throughput of the architecture is two samples per cycle. The cycle time is the consumed time for data multiplexing, a CORDIC rotation, and one addition. In this design, the input/output (I/O) cost, including the number of I/O channels and the I/O bandwidth, is independent of the transform length *N*. To sum up, the proposed architecture features a systolic computing style, PE's with the CORDIC structure, low I/O cost, and the encapsulated new M-D DHT algorithm.

C. Related Research

In the literature, various DHT algorithms for VLSI implementation have been considered [5]–[8]. In the following, we analyze these designs from four different points of view: algorithm, architecture, implementation, and application.

Encapsulated algorithm: The algorithms used in designs [5]–[8] can be categorized into two groups, the fast Hartley transform (FHT) algorithms [5], [6] and the DHT algorithms [7], [8]. The FHT algorithms [5], [6] primarily focus on reducing the number of multiplications. They require about $O(N \cdot \log N)$ or $O(N)$ multiplications to compute an *N*-point DHT, but suffer from complex data routing. The DHT algorithms [7], [8] provide simple and local communication but require $O(N^2)$ multiplications. Checking the analysis for the architectures of the DFT and fast Fourier transform (FFT) [11], we find that the high communication and control cost of fast algorithms would make the architectures realizing the FFT have approximately the same hardware cost as those realizing the DFT. A good review of such analysis [12] has been published. In addition, considering the commercial chips for the DCT [13], we find that most of the designs are based on the discrete cosine transform (DCT) algorithms instead of the fast cosine transform (FCT) algorithms. Following these analyses, we adopt the DHT algorithms instead of the FHT algorithms to design the VLSI architectures in this paper.

Architecture topology: Among various VLSI architectures, systolic arrays [14] are a type of architecture that provides both high processing speeds and convenient VLSI implementation. The designs in [5]–[8] are based on systolic arrays. Their architecture topology consists of linear arrays and 2-D arrays. Basically, a 2-D systolic array has an I/O cost depending on the transform length *N*. For linear arrays, we can restrict the I/O channels on the array boundaries so that the I/O cost can be kept independent of the transform length. The arrays designed in this paper are linear arrays. We utilize the *Tag control* scheme [15] to arrange all the I/O channels located at the two extreme ends of the arrays so as to minimize the I/O cost.

Implementation: Basically, the operations required in computing the DHT are triangular functions, multiplications, and additions. The triangular functions can be precomputed in a host computer and sent to the arrays for computation. However, such an approach would add overhead to the I/O cost [5]. Alternatively, the Givens Rotor [7] and CORDIC processor [6], [8] are two functional elements that have been used to realize the operations of the DHT. They both employ the shift-and-add computation as the basic computing style. Since the property of the CORDIC is beneficial in deriving the new M-D DHT algorithm, we use the CORDIC processor as the main computing element in the systolic arrays designed in this paper.

Application: Basically, the designs [5]–[7] are for 1-D DHT and the design [8] is for both the 1-D and 2-D DHT. Based on the Bracewell's approach, one can compute the 2-D DHT using the designs [5]–[7] accompanied with the transpose memory and complex postaddition circuits. Alternatively, one can use the design [8] to compute a 2-D DHT, which realizes the temporary variable $T(k_1, k_2)$ through a 2-D array and needs complex postaddition circuits to realize $H(k_1, k_2)$. The proposed design realizes a M-D DHT problem through M linear arrays and transpose memory accompanied with only two adders for postadditions. Using transpose memory and extra control circuits in the proposed design induces some overhead as compared with the design [8]. However, as higher dimensional DHT problems are considered, the hardware cost required in the [8] will increase tremendously.

The rest of this paper is organized as follows. The derivation of the new M-D DHT algorithm is presented in Section II. The hardware realization of the new algorithm is presented in Section III. A brief conclusion is given in Section IV. Detailed derivations of the new algorithm are given in Appendix A and Appendix B.

II. Algorithm Derivation

By separating the arguments of the cosine and sine functions and using the ability of CORDIC to support different output functions, the M-D DHT defined in (1) can be reformulated as

$$H(k_1, k_2, \cdots, k_M) = C_M(k_1, k_2, \cdots, k_M) + S_M(k_1, k_2, \cdots, k_M) \tag{5}$$

where $C_M(k_1, k_2, \cdots, k_M)$ and $S_M(k_1, k_2, \cdots, k_M)$ can be computed through the iterative formula

$$C_i(k_i) = \sum_{n_i=0}^{N-1} CORDY2[S_{i-1}(n_i), C_{i-1}(n_i), \varphi_{n_i k_i}]$$

$$S_i(k_i) = \sum_{n_i=0}^{N-1} CORDY1[S_{i-1}(n_i), C_{i-1}(n_i), \varphi_{n_i k_i}]; \tag{6}$$

from $i = 1, \cdots, M$, where

$$C_0(n_1, n_2, \cdots, n_M) = x(n_1, n_2, \cdots, n_M)$$
$$S_0(n_1, n_2, \cdots, n_M) = 0.$$

Here, we use $A(k_i)$ and $B(n_i)$ to, respectively, denote $A(k_1, \cdots, k_i, n_{i+1}, \cdots, n_M)$ and $B(k_1, \cdots, k_{i-1}, n_i, \cdots, n_M)$ for simplifying the mathematical expressions. And $CORDY1[x, y, \psi]$ and $CORDY2[x, y, \psi]$ denote the two outputs of the basic CORDIC processor with inputs x, y, and rotation angle ψ, as shown in Fig. 1. The detailed derivation from (1) to (5)–(6) is shown in Appendix A. The $C_i(k_i)$ and $S_i(k_i)$ are the results of one iteration. One iteration computation in (6) is a kind of 1-D transform. This transform can be easily realized with the CORDIC processors by assigning $(x, y, \psi) = (S_{i-1}(n_i), C_{i-1}(n_i), \varphi_{n_ik_i})$. So, similar to other separable transforms, we can compute the M-D DHT through the decomposed 1-D transforms iteratively. Moreover, the new algorithm only needs a postaddition which is much simpler in hardware than the postprocessing indicated in (3).

Besides, exploiting the symmetry of the triangular functions can further reduce the complexity of (6). It is shown in Appendix B that (6) can be written as

$$C_i(k_i) = \sum_{n_i=0}^{\frac{N}{2}-1} CORDY2[S'_{i-1}(n_i), C'_{i-1}(n_i), \varphi_{n_ik_i}]$$

$$C_i(k_i + \frac{N}{2}) = \sum_{n_i=0}^{\frac{N}{2}-1} (-1)^{n_i} \cdot CORDY2[S'_{i-1}(n_i), C'_{i-1}(n_i), \varphi_{n_ik_i}]$$

$$S_i(k_i) = \sum_{n_i=0}^{\frac{N}{2}-1} CORDY1[S'_{i-1}(n_i), C'_{i-1}(n_i), \varphi_{n_ik_i}]$$

$$S_i(k_i + \frac{N}{2}) = \sum_{n_i=0}^{\frac{N}{2}-1} (-1)^{n_i} \cdot CORDY1[S'_{i-1}(n_i), C'_{i-1}(n_i), \varphi_{n_ik_i}];$$

$$i = 1, \cdots, M; k_i = 0, \cdots, \frac{N}{2} - 1 \qquad (7)$$

where

$$C'_{i-1}(n_i) = C_{i-1}(n_i) + (-1)^{k_i} \cdot C_{i-1}(n_i + \frac{N}{2})$$
$$S'_{i-1}(n_i) = S_{i-1}(n_i) + (-1)^{k_i} \cdot S_{i-1}(n_i + \frac{N}{2})$$

and

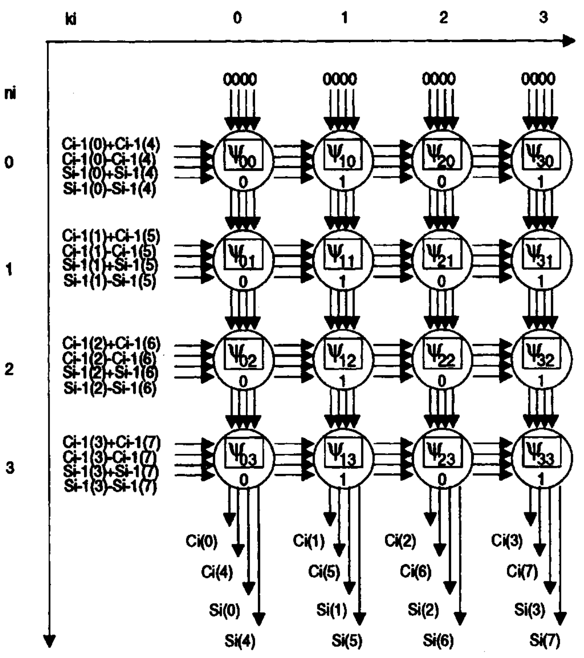
$$C_0(n_1, n_2, \cdots, n_M) = x(n_1, n_2, \cdots, n_M)$$
$$S_0(n_1, n_2, \cdots, n_M) = 0.$$

In the above equation, it is assumed that $N = 2^p$ and the integer $p \geq 2$. Note that (7) possesses properties similar to those of (6), but requires lighter computational load than (6) owing to two features. First, the numbers of CORDIC rotations required to compute $C_i(k_i)$ and $S_i(k_i)$ are reduced to one half of those required in (6). That is, the upper limit of the index n_i is reduced from N in (6) to $\frac{N}{2}$ in (7). This phenomenon results in the reduction of the computation time by a factor of 2. Second, the computations of $C_i(k_i)$ and $C_i(k_i + \frac{N}{2})$ share the same intermediate results obtained from the CORDIC processors, except that the results may have different signs. That is, we can use a CORDIC processor to compute two output results at a time, which reduces the number of the CORDIC processors by a factor of 2. The same phenomenon is also found in computing $S_i(k_i)$ and $S_i(k_i + \frac{N}{2})$. This saving in the number of the CORDIC processors will be even more apparent if pipelined CORDIC structures [16] are utilized.

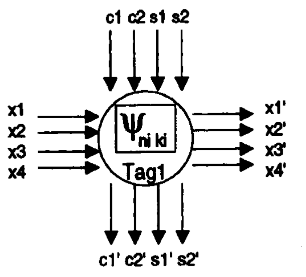
III. Hardware Implementation

To illustrate the array architecture for the M-D DHT, in this section we consider an example of a 3-D 8 × 8 × 8 DHT.

Dependence graph: Fig. 2 shows the dependence graph (DG) to compute $C_i(k_i)$ and $S_i(k_i)$ in (7). The DG clearly illustrates the data operations, the data dependency, and the control signals involved in this algorithm. In the DG, the nodes represent the operations to be



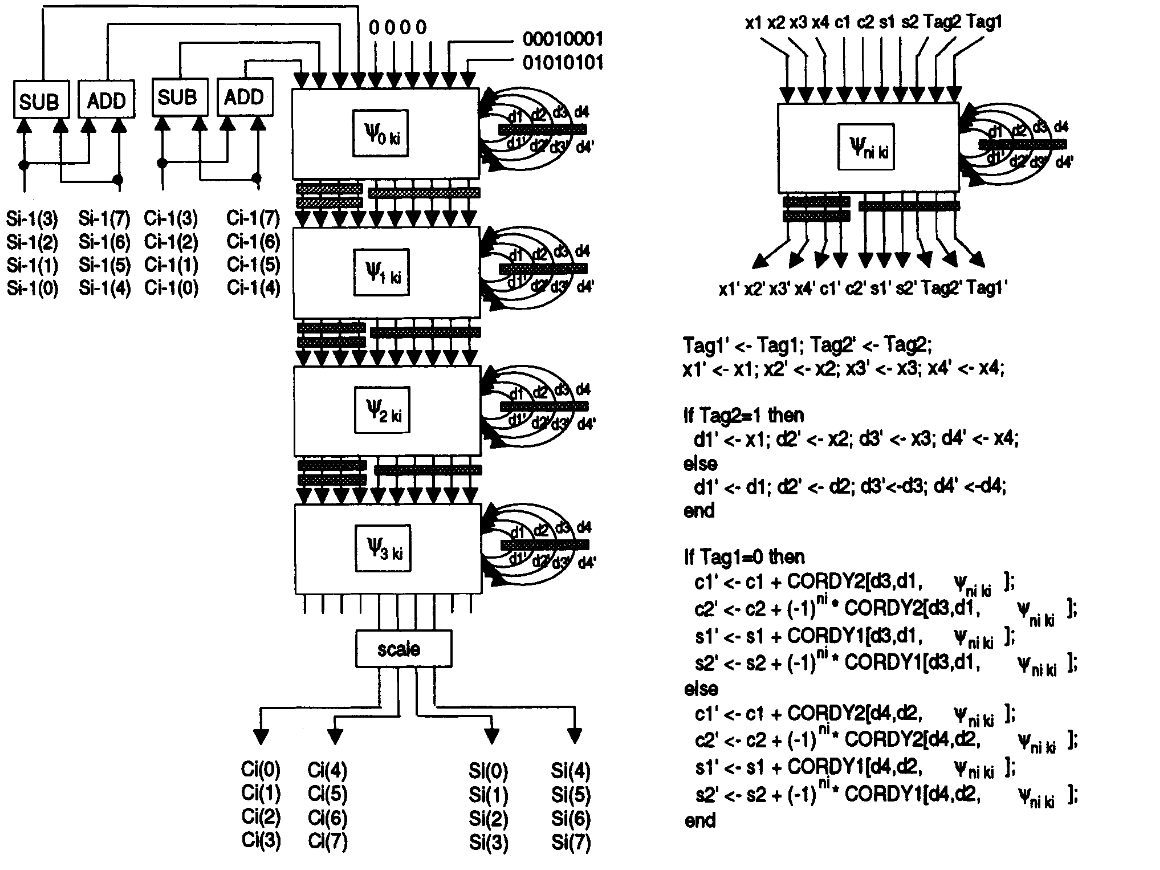
(a)



```
x1' <- x1; x2'<-x2; x3' <- x3; x4' <- x4;
If Tag1=0,
    c1' <- c1 + CORDY2[x3,x1, $\Psi_{niki}$];
    c2' <- c2 + $(-1)^{ni}$ * CORDY2[x3,x1, $\Psi_{niki}$];
    s1' <- s1 + CORDY1[x3,x1, $\Psi_{niki}$];
    s2' <- s2 + $(-1)^{ni}$ * CORDY1[x3,x1, $\Psi_{niki}$];
else
    c1' <- c1 + CORDY2[x4,x2, $\Psi_{niki}$];
    c2' <- c2 + $(-1)^{ni}$ * CORDY2[x4,x2, $\Psi_{niki}$];
    s1' <- s1 + CORDY1[x4,x2, $\Psi_{niki}$];
    s2' <- s2 + $(-1)^{ni}$ * CORDY1[x4,x2, $\Psi_{niki}$];
end
```

(b)

Fig. 2. (a) The dependence graph for computing $C_i(k_i)$ and $S_i(k_i)$. (b) The function of the nodes ($\Psi_{n_ik_i} = \frac{2\pi n_ik_i}{8}$, $CORDY1[x, y, z]$ and $CORDY2[x, y, z]$ denote the outputs of the CORDIC processor with inputs x, y, and z. $C_i(k_i)$ and $S_i(k_i)$, respectively, denote $C_i(k_1, \cdots, k_i, n_{i+1}, \cdots, n_M)$ and $S_i(k_1, \cdots, k_i, n_{i+1}, \cdots, n_M)$. $C_{i-1}(n_i)$ and $S_{i-1}(n_i)$, respectively, denote $C_{i-1}(k_1, \cdots, k_{i-1}, n_i, \cdots, n_M)$ and $S_{i-1}(k_1, \cdots, k_{i-1}, n_i, \cdots, n_M)$).


0000
00010001
01010101
SUB
ADD
SUB
ADD
Si-1(3) Si-1(7) Ci-1(3) Ci-1(7)
Si-1(2) Si-1(6) Ci-1(2) Ci-1(6)
Si-1(1) Si-1(5) Ci-1(1) Ci-1(5)
Si-1(0) Si-1(4) Ci-1(0) Ci-1(4)
Ψ0 ki
Ψ1 ki
Ψ2 ki
Ψ3 ki
d1 d2 d3 d4
d1' d2' d3' d4'
scale
Ci(0) Ci(4) Si(0) Si(4)
Ci(1) Ci(5) Si(1) Si(5)
Ci(2) Ci(6) Si(2) Si(6)
Ci(3) Ci(7) Si(3) Si(7)
(a)
x1 x2 x3 x4 c1 c2 s1 s2 Tag2 Tag1
Ψni ki
x1' x2' x3' x4' c1' c2' s1' s2' Tag2' Tag1'
(b)

```
Tag1' <- Tag1; Tag2' <- Tag2;
x1' <- x1; x2' <- x2; x3' <- x3; x4' <- x4;

If Tag2=1 then
  d1' <- x1; d2' <- x2; d3' <- x3; d4' <- x4;
else
  d1' <- d1; d2' <- d2; d3'<-d3; d4' <-d4;
end

If Tag1=0 then
  c1' <- c1 + CORDY2[d3,d1, $\Psi_{ni\,ki}$ ];
  c2' <- c2 + $(-1)^{ni}$ * CORDY2[d3,d1, $\Psi_{ni\,ki}$ ];
  s1' <- s1 + CORDY1[d3,d1, $\Psi_{ni\,ki}$ ];
  s2' <- s2 + $(-1)^{ni}$ * CORDY1[d3,d1, $\Psi_{ni\,ki}$ ];
else
  c1' <- c1 + CORDY2[d4,d2, $\Psi_{ni\,ki}$ ];
  c2' <- c2 + $(-1)^{ni}$ * CORDY2[d4,d2, $\Psi_{ni\,ki}$ ];
  s1' <- s1 + CORDY1[d4,d2, $\Psi_{ni\,ki}$ ];
  s2' <- s2 + $(-1)^{ni}$ * CORDY1[d4,d2, $\Psi_{ni\,ki}$ ];
end
```

Fig. 3. (a) The array architecture for computing $C_i(k_i)$ and $S_i(k_i)$. (b) The function of the PE's ($\Psi_{n_i k_i} = \frac{2\pi n_i k_i}{8}$).

executed, including the CORDIC rotations and additions as described in Fig. 2(b). The directed arcs indicate a data dependency between two neighboring nodes; that is, the computed result from one node should be sent along an arc to be operated on the other node. Based on the DG-based array synthesis procedure [15], [17], we can obtain a linear array by suitably projecting the DG. If the DG shown in Fig. 2 is projected vertically, a PE should perform all the operations in the nodes along a certain column of the DG. Since there are addition paths along the column, e.g., $c1' = c1 + CORDY2[x3, x1, \psi_{n_i k_i}]$, the PE's would have accumulation loops. On the other hand, if the DG is projected horizontally, there will be data transmission loops in the PE's. Since the consumed time for the data transmission loops is shorter than that for the accumulation loops, using the two-level pipelining technique [18], [19] can further reduce the cycle time of the array from the addition time to the data transmission time. This fact facilitates the speed improvement of the designed array.

Array architecture for the 1-D transform: Fig. 3 illustrates the linear array obtained from projecting the DG shown in Fig. 2 horizontally. In this array, the input data are first preprocessed and piped in from the topmost PE and then transmitted to the neighboring PE's rhythmically. *Tag1* and *Tag2* are 1-b control signals to decide which data operands the PE's select. Since the rotation angles used in the array are known in advance, the rotation directions $\{\xi_{ij}, i, j = 0, 1, \cdots, \frac{N}{2} - 1\}$ instead of the rotation angles $\{\varphi_{ij}, i, j = 0, 1, \cdots, \frac{N}{2} - 1\}$ are precalculated and stored in ROM's to reduce the hardware and I/O cost. The output results are accumulated and drained out from the bottommost PE. Unlike the design in [8], we use the *Tag control* scheme [15] to locate all the I/O channels at the boundary PE's, which makes the I/O cost of the array independent of the transform length *N*. Note that there is a scaling operator in the end of the linear array to perform the scaling operations required in the CORDIC processors. This operator can be easily realized by using memory look-up tables or a CORDIC processor operated in linear rotation mode [16].

Exploiting the symmetry of the triangular functions, the linear array requires only $\frac{N}{2}$ PE's, where each PE consists of a CORDIC processor, a ROM with *N* words, several data multiplexing circuits, and four adders, as illustrated in Fig. 4. Additional two adders and substractors are required in the array for data preprocessing. The throughput of the array is two samples per cycle. The average computation time to compute a 1-D *N*-point transform is $\frac{N}{2} \cdot T_{cycle}$. The term T_{cycle}, which denotes the cycle time of the array, includes the consumed time for the data multiplexing, a CORDIC rotation, and one addition. The speed of the array can be further enhanced for high-speed DHT applications by utilizing the techniques presented in [20]–[22], such as the pipelined CORDIC, the forward angle recording CORDIC algorithm, and the CORDIC algorithms with redundant arithmetic and reduced iterations.

Array architecture for the 3-D DHT: By cascading three arrays shown in Fig. 3 and two transpose RAM's together, we construct

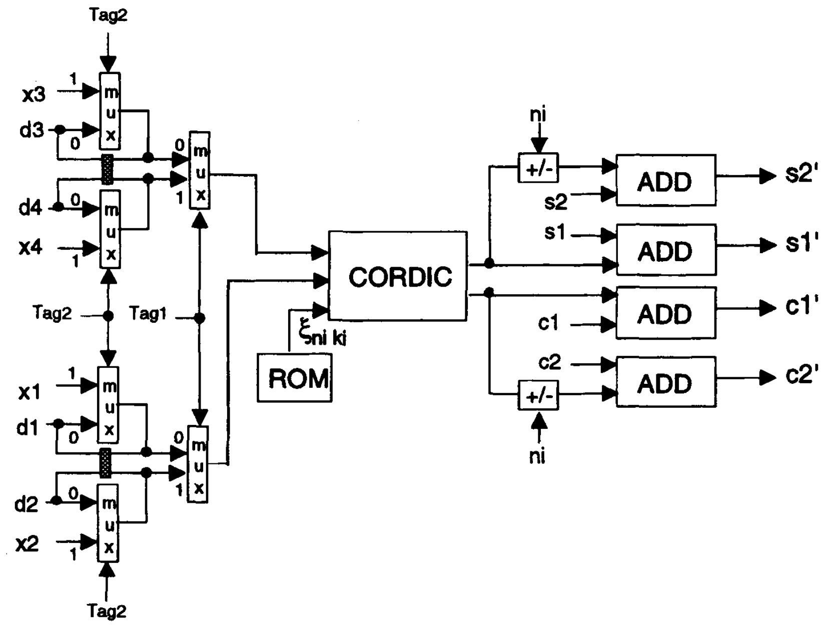


Fig. 4. The structure of the PE's in the designed array ($\xi_{n_ik_i}$ denotes the rotation direction corresponding to the rotation angle $\Psi_{n_ik_i}$).

the array architecture for the 3-D DHT as shown in Fig. 5. Note that the proposed architecture only requires two adders to perform the addition operations indicated in (5). These two adders are much simpler in hardware than the postaddition stage required to perform the operations indicated in (3). Since this architecture outputs two samples per cycle, it requires $\frac{N^3}{2}$ cycles to compute a 3-D $N \times N \times N$ DHT problem. If the hardware area is the main concern, we can use the array shown in Fig. 3 iteratively to compute all the decomposed 1-D transforms. This kind of architecture outputs two samples every three cycles. So, it requires $\frac{3N^3}{2}$ cycles to compute a 3-D $N \times N \times N$ DHT problem.

The proposed array architecture is designed based on the algorithm expressed by (7), which exploits the symmetry of the triangular functions for reducing the computational complexity. It gains the saving in the hardware cost and computation time, but needs a little overhead, such as data multiplexing circuits, simple control, some adders, and extra time consumption. If this overhead costs more hardware price than the saving of the proposed architecture gains according to a certain technology, the algorithm expressed by (6), which does not exploit the symmetry of the triangular functions, can be used to design the array architecture for the M-D DHT.

IV. Conclusion

In this paper, a new M-D DHT algorithm using CORDIC has been presented with simpler postprocessing as compared to the Bracewell's approach. A kind of 1-D transform different from 1-D DHT has been derived to compute the M-D DHT separably. To reduce the complexity of the 1-D transforms, we have exploited the symmetry of the triangular functions. Using this algorithm, we have designed an array architecture for the 3-D $N \times N \times N$ DHT. This architecture consists of three linear systolic arrays and two transpose RAM's. Each array consists of $\frac{N}{2}$ PE's, which are composed of CORDIC processors, ROM's, and adders. The throughput of the architecture is two samples per cycle, and the cycle time is the consumed time for data multiplexing, a CORDIC rotation, and one addition. In this design the I/O cost, including the number of I/O channels and the I/O bandwidth, is independent of the transform length N. In summary, the proposed design features a systolic computing style, PE's with a CORDIC structure, low I/O cost, and the encapsulated new M-D DHT algorithm.

Appendix A

This appendix illustrates the derivation from (1) to (5)–(6). First, we can rewrite (1) as

$$H(k_1,k_2,\cdots,k_M) = C_M(k_1,k_2,\cdots,k_M) + S_M(k_1,k_2,\cdots,k_M) \tag{8}$$

where

$$C_M(k_1,k_2,\cdots,k_M) = \sum_{n_M=0}^{N-1}\cdots\sum_{n_2=0}^{N-1}\sum_{n_1=0}^{N-1} x(n_1,n_2,\cdots,n_M) \cdot \cos(\varphi_{n_1k_1}+\varphi_{n_2k_2}+\cdots+\varphi_{n_Mk_M})$$

$$S_M(k_1,k_2,\cdots,k_M) = \sum_{n_M=0}^{N-1}\cdots\sum_{n_2=0}^{N-1}\sum_{n_1=0}^{N-1} x(n_1,n_2,\cdots,n_M) \cdot \sin(\varphi_{n_1k_1}+\varphi_{n_2k_2}+\cdots+\varphi_{n_Mk_M}). \tag{9}$$

With a view to simplify the mathematical expressions, we use $A(k_i)$ and $B(n_i)$ to, respectively, denote $A(k_1,\cdots,k_i,n_{i+1},\cdots,n_M)$ and $B(k_1,\cdots,k_{i-1},n_i,\cdots,n_M)$. By splitting the arguments of the cosine and sine functions in $C_M(k_M)$ and $S_M(k_M)$, we obtain

$$C_M(k_M) = \sum_{n_M=0}^{N-1}[C_{M-1}(n_M)\cdot\cos(\varphi_{n_Mk_M}) - S_{M-1}(n_M)\cdot\sin(\varphi_{n_Mk_M})] \tag{10}$$

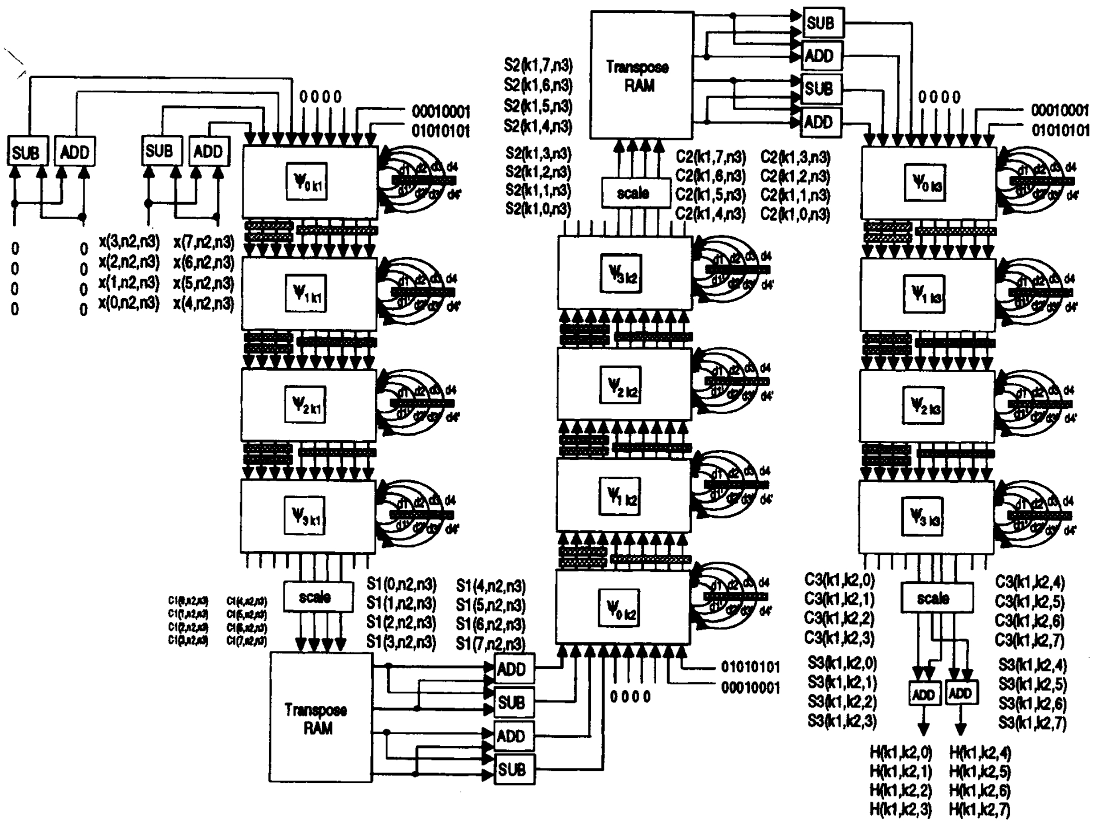


Fig. 5. The array architecture for the 3-D 8 × 8 × 8 DHT ($\Psi_{n_ik_i}$ denotes $\frac{2\pi n_i k_i}{8}$).

and

$$S_M(k_M) = \sum_{n_M=0}^{N-1} [S_{M-1}(n_M)\cdot\cos(\varphi_{n_Mk_M}) + C_{M-1}(n_M)\cdot\sin(\varphi_{n_Mk_M})] \quad (11)$$

where

$$C_{M-1}(n_M) = \sum_{n_{M-1}=0}^{N-1}\cdots\sum_{n_1=0}^{N-1} x(n_1,\cdots,n_M)\cdot\cos(\varphi_{n_1k_1}+\cdots+\varphi_{n_{M-1}k_{M-1}})$$

$$S_{M-1}(n_M) = \sum_{n_{M-1}=0}^{N-1}\cdots\sum_{n_1=0}^{N-1} x(n_1,\cdots,n_M)\cdot\sin(\varphi_{n_1k_1}+\cdots+\varphi_{n_{M-1}k_{M-1}}).$$

Using the property of CORDIC, we can express (10) and (11) as

$$C_M(k_M) = \sum_{n_M=0}^{N-1} CORDY2[S_{M-1}(n_M), C_{M-1}(n_M), \varphi_{n_Mk_M}]$$

$$S_M(k_M) = \sum_{n_M=0}^{N-1} CORDY1[S_{M-1}(n_M), C_{M-1}(n_M), \varphi_{n_Mk_M}] \quad (12)$$

where $CORDY1[x,y,\psi]$ and $CORDY2[x,y,\psi]$ denote the two outputs of the basic CORDIC processor with inputs $x, y,$ and rotation angle ψ, as shown in Fig. 1. (x,y,ψ) is assigned to be $(S_{M-1}(n_M), C_{M-1}(n_M), \varphi_{n_Mk_M})$. Repeating the same procedures, we can compute (9) iteratively through

$$C_i(k_i) = \sum_{n_i=0}^{N-1} CORDY2[S_{i-1}(n_i), C_{i-1}(n_i), \varphi_{n_ik_i}]$$

$$S_i(k_i) = \sum_{n_i=0}^{N-1} CORDY1[S_{i-1}(n_i), C_{i-1}(n_i), \varphi_{n_ik_i}]; \quad (13)$$

from $i = 1,\cdots,M$, where

$$C_0(n_1,n_2,\cdots,n_M) = x(n_1,n_2,\cdots,n_M)$$
$$S_0(n_1,n_2,\cdots,n_M) = 0.$$

Appendix B

This appendix illustrates the derivation from (6) to (7). In the following, it is assumed that $N = 2^p$ and the integer $p \geq 2$. First, considering the $C_i(k_i)$ in (6) and using the symmetry of the cosine function, we obtain

$$C_i(k_i) = \sum_{n_i=0}^{N-1}[C_{i-1}(n_i)\cdot\cos(\varphi_{n_ik_i}) - S_{i-1}(n_i)\cdot\sin(\varphi_{n_ik_i})]$$
$$= \sum_{n_i=0}^{\frac{N}{2}-1}[C'_{i-1}(n_i)\cdot\cos(\varphi_{n_ik_i}) - S'_{i-1}(n_i)\cdot\sin(\varphi_{n_ik_i})] \quad (14)$$

where

$$C'_{i-1}(n_i) = C_{i-1}(n_i) + (-1)^{k_i} \cdot C_{i-1}(n_i + \frac{N}{2})$$
$$S'_{i-1}(n_i) = S_{i-1}(n_i) + (-1)^{k_i} \cdot S_{i-1}(n_i + \frac{N}{2})$$

and $\varphi_{n_i k_i} = \frac{2\pi n_i k_i}{N}$. Now we split the index k_i in (14) into two parts, i.e., $k_i = 0, 1, \cdots, \frac{N}{2} - 1$ and $k_i = \frac{N}{2}, \cdots, N-1$. Thus, we can write $\{C_i(k_i), k_i = 0, 1, \cdots, \frac{N}{2} - 1\}$ as

$$\begin{aligned} C_i(k_i) &= \sum_{n_i=0}^{\frac{N}{2}-1} [C'_{i-1}(n_i) \cdot \cos(\varphi_{n_i k_i}) - S'_{i-1}(n_i) \cdot \sin(\varphi_{n_i k_i})] \\ &= \sum_{n_i=0}^{\frac{N}{2}-1} CORDY2[S'_{i-1}(n_i), C'_{i-1}(n_i), \varphi_{n_i k_i}]. \end{aligned} \quad (15)$$

Then, we can express $\{C_i(k_i), k_i = \frac{N}{2}, \cdots, N-1\}$ in terms of $\{C_i(k_i + \frac{N}{2}), k_i = 0, 1, \cdots, \frac{N}{2} - 1\}$. That is,

$$\begin{aligned} C_i(k_i + \frac{N}{2}) &= \sum_{n_i=0}^{\frac{N}{2}-1} [C'_{i-1}(n_i) \cdot \cos(\varphi_{n_i(k_i+\frac{N}{2})}) \\ &\quad - S'_{i-1}(n_i) \cdot \sin(\varphi_{n_i(k_i+\frac{N}{2})})] \\ &= \sum_{n_i=0}^{\frac{N}{2}-1} (-1)^{n_i} \cdot [C'_{i-1}(n_i) \cdot \cos(\varphi_{n_i k_i}) \\ &\quad - S'_{i-1}(n_i) \cdot \sin(\varphi_{n_i k_i})] \\ &= \sum_{n_i=0}^{\frac{N}{2}-1} (-1)^{n_i} \\ &\quad \cdot CORDY2[S'_{i-1}(n_i), C'_{i-1}(n_i), \varphi_{n_i k_i}]. \end{aligned} \quad (16)$$

Based on the same procedures, we can rewrite the $S_i(k_i)$ in (6) as

$$\begin{aligned} S_i(k_i) &= \sum_{n_i=0}^{\frac{N}{2}-1} CORDY1[S'_{i-1}(n_i), C'_{i-1}(n_i), \varphi_{n_i k_i}] \\ S_i(k_i + \frac{N}{2}) &= \sum_{n_i=0}^{\frac{N}{2}-1} (-1)^{n_i} CORDY1[S'_{i-1}(n_i), C'_{i-1}(n_i), \varphi_{n_i k_i}]; \\ & k_i = 0, 1, \cdots, \frac{N}{2} - 1. \end{aligned} \quad (17)$$

References

[1] R. N. Bracewell, "Discrete Hartley transform," *J. Opt. Soc. Amer.*, vol. 73, pp. 1832–1835, Dec. 1983.

[2] ——, "The fast Hartley transform," *Proc. IEEE*, vol. 72, pp. 1010–1018, Aug. 1984.

[3] D. E. Dudgeon and R. M. Mersereau, *Multidimensional Digital Signal Processing*. Englewood Cliffs, NJ: Prentice-Hall, 1984, ch. 2, pp. 75–76.

[4] R. N. Bracewell, O. Buneman, H. Hao, and J. Villasenor, "Fast two-dimensional Hartley transform," *Proc. IEEE*, vol. 74, no. 9, pp. 1282–1283, Sept. 1986.

[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.*, vol. 39, no. 11, pp. 1359–1368, Nov. 1990.

[6] M. Marchesi, G. Orlandi, and F. Piazza, "A systolic circuit for fast Hartley transform," in *Proc. ISCAS'88*, 1988, pp. 2685–2688.

[7] A. S. Dhar and S. Banerjee, "An array architecture for fast computation of discrete Hartley transform," *IEEE Trans. Circuits Syst.*, vol. 38, no. 9, pp. 1095–1098, Sept. 1991.

[8] L. W. Chang, and S. W. Lee, "Systolic arrays for the discrete Hartley transform," *IEEE Trans. Signal Process.*, vol. 39, no. 11, pp. 2411–2418, Nov. 1991.

[9] J. E. Volder, "The CORDIC trigometric computing technique," *IRE Trans. Electron. Comput.*, vol. EC-8, pp. 330–334, Sept. 1959.

[10] J. S. Walther, "A unified algorithm for elementary functions," in *AFIPS Spring Joint Comput. Conf.*, 1971, pp. 379–385.

[11] C. D. Thompson, "Fourier transforms in VLSI," *IEEE Trans. Comput.*, vol. C-32, no. 1, pp. 1047–1057, Nov. 1983.

[12] J. A. Beraldin, T. Aboulnasr, and W. Steenaart, "Efficient one-dimensional systolic array realization of discrete Fourier transform," *IEEE Trans. Circuits Syst.*, vol. 36, pp. 95–100, Jan. 1989.

[13] K. R. Rao and P. Yip, *Discrete Cosine Transform: Algorithm, Advantages, Applications*. New York: Academic, 1990.

[14] H. T. Kung, "Why systolic architectures?" *Comput. Mag.*, vol. 15, pp. 37–46, Jan. 1982.

[15] C. W. Jen and H. Y. Hsu "The design of a systolic array with tags input," in *Proc. ISCAS'88, Finland*, 1988, pp. 2263–2266.

[16] Y. H. Hu, "CORDIC-based VLSI architectures for digital signal processing," *IEEE Signal Process. Mag.*, vol. 9, no. 3, pp. 16–35, July 1992.

[17] S. Y. Kung, *VLSI Array Processors*. Englewood Cliffs, NJ: Prentice-Hall, 1988, ch. 3 and 4, pp. 110–282.

[18] K. K. Parhi and D. G. Messerschmitt, "Pipeline interleaving and parallelism in recursive digital filters—Part I: Pipelining using scattered look-ahead and decomposition," *IEEE Trans. Acoust., Speech, Signal Process.*, vol. 37, no. 7, pp. 1099–1117, 1989.

[19] C. M. Liu and C. W. Jen, "On the design of VLSI array for discrete Fourier transform," *IEE Proc.-G*, vol. 139, no. 4, pp. 541–552, Aug. 1992.

[20] Y. H. Hu, "A forward angle recoding cordic algorithm and pipelined processor array structure for digital signal processing," *Digital Signal Process.*, vol. 3, pp. 2–15, 1993.

[21] H. Dawid and H. Meyr, "VLSI implementation of the CORDIC algorithm using redundant arithmetic," in *Proc. ISCAS'92*, May 1992, pp. 1089–1092.

[22] D. Timmermann, H. Hahn, and B. Hosticka, "Modified CORDIC algorithm with reduced iterations," *Electron. Lett.*, vol. 25, no. 15, pp. 950–951, July 1989.