The Efficient Memory-Based VLSI Array Designs For DFT and DCT

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

Abstract—**In this paper, the efficient memory-based VLSI arrays and the accompanied new design approach for the discrete Fourier transform (DFT) and discrete cosine transform (DCT) are presented. The DFT and DCT are formulated as cyclic convolution forms and mapped into linear arrays which characterize small numbers of I / O channels and low I / O bandwidth. Since the multipliers consume much hardware area, the presented designs utilize small ROM's and adders to implement the multiplications, which is based on good data arrangements exploiting the number properties of the transform kernels. Moreover, the ROM size can be reduced effectively by arranging the data in our designs appropriately. Typically, to perform 1-D N-point DFT and DCT, the arrays need $N \times 2^L$ words of ROM only. Compared to the conventional distributed arithmetic architectures which should require $N \times 2^N$ words of ROM, much memory can be saved if N is greater than L, which occurs in most DFT applications. To summarize, the presented arrays outperform others in the architectural topology (local and regular connection), computing speeds, hardware complexity, the number of I / O channels, and I / O bandwidth. They take the advantages of both systolic arrays and the memory-based architectures.**

I. Introduction

THE DISCRETE Fourier transform (DFT) and discrete cosine transform (DCT) are the key functions widely used in many significant image and signal processing applications. Because of the high computational complexity, the derivations of efficient algorithms suitable for VLSI are inevitable in many real-time applications. In the literatures, a variety of algorithms have been proposed for computing the DFT and DCT. Since each algorithm has its own specific property and application field, not all the algorithms are well suited for VLSI implementation. The efficiency of an algorithm to be implemented in VLSI is based more on the degree of the communication complexity required among arithmetic elements rather than on the number of computations. Hence, the fact having been observed by many researchers [1]–[8] is that fast Fourier transform (FFT) like algorithms which have been used extensively for their low numbers of multiplications are not well suited for VLSI implementation.

Systolic arrays [1], [2] can meet the increasing requirements of processing speeds and be well suited for VLSI implementation. They attain high processing speeds through parallel and pipeline processing, and make the VLSI implementation feasible through modularity, structural regularity, and local interconnection. We refer to the paper in [3] for the motivations of systolic array architectures over others. When systolic arrays are used as the design vehicle, the speeds and implementation benefits are able to be pronounced only if a large number of low-cost processing elements (PE's) can be implemented in a VLSI chip. In the existing systolic arrays for DFT and DCT, multipliers are the fundamental computing elements in PE's. Since multipliers should consume a large silicon area, the limited chip size should put a severe limitation to the allowable number of PE's. Passively, such arrays [2]–[6], [8]–[14] should wait for the advent of VLSI technology such as wafer-scale integration to make their benefits visible. Constructively, systolic arrays and the encapsulated algorithms should be developed to simplify the structure and complexity of PE's. Based on this point, this paper presents a new approach to design the VLSI arrays for DFT and DCT. Since this approach derives algorithms based on the data permutations introduced in [8], [14], the designed arrays possess better performance in the computing parallelism, computational complexity, and I/O cost than the designs in [2]–[5], [9]–[11] do as analyzed in [8], [14]. Also, this approach considers the efficiency of hardware implementation and provides an efficient way to replace multipliers by small ROM's such that the designed arrays can attain high computing speeds at the expense of a small silicon area.

Owing to the regular and compact structure of ROM's, the methods to replace multipliers by ROM's have been studied by numerous researchers (see the references in [16]). Among them, distributed arithmetic (DA) has been successfully applied to implement a 16×16 DCT in a single chip [20] and widely adopted for commercial products [21]–[29]. DA is a technique that computes the multiplications involved in an inner-product by a series of memory access and accumulation operations. If the vector length and the wordlength of input data are assumed to be N and L, respectively. DA typically performs an inner-product by a ROM with size equal to 2^N words in L

Manuscript received February 18, 1992; revised July 24, 1992. This work was supported by the National Science Council of Taiwan under Contract NSC81-0404-E009-134, and by the Telecommunication Laboratory of the Ministry of communication. This paper was recommended by Associate Editor I. Shirakawa.

J. Guo and C. Jen are with the Department of Electronics Engineering and Institute of Electronics, National Chiao Tung University, Hsinchu, Taiwan, ROC.

C. Liu is with the Department of Computer Science and Information Engineering, National Chiao Tung University, Tsinchu, Taiwan, ROC.

IEEE Log Number 9205006.

steps in a bit-serial manner. When applying DA to DCT or DFT, the combination of bit-serial and bit-parallel operations renders the implemented chip requiring a large amount of shift registers or buffers [20]–[29]. Also these architectures [19]–[29] suffer from the imbalance between the wordlength L and the vector length N. Based on total ROM size of $N \times 2^N$ words, the number of operation steps required to perform a 1-D N-point DCT or DFT is determined by the larger value of N and L. In this paper, a new technique is presented to efficiently replace the multipliers by ROM's for DFT and DCT. This technique leads to an architecture which attains structural regularity and modularity among PE's like systolic arrays. The operations of PE's are performed simply by ROM's and adders, which is like the style of DA. The total ROM size and the number of operation steps for the presented architectures to perform 1-D N-point DCT and DFT are about $N \times 2^L$ words and N, respectively. If N is greater than L, which occurs in most DFT problems, the presented designs shall require lower hardware cost in ROM's than the DA architectures do. Moreover, the presented architectures operate in a bit-par allel manner which is different from that of the DA architectures. Thus, they are free from the large amount of shift registers or buffers. To sum up, the new approach presented in this paper can be used to design the VLSI architectures f or the DFT and DCT, which can take the advantages of both systolic arrays and t he architectures based on memory.

For the purpose of showing the essentials of this approach concretely, the characteristics of the approach are discussed in the following. In the first place, to attain high computing parallelism, low computational complexity, and the attractive feature of linear arrays that the I/O bandwidth as well as the number of I/O channels can be kept independent of the array length, the DFT and DCT are formulated as cyclic convolution forms [8], [14]. That is, the structure of *Galois Field* is utilized to permute the input and output data such that the DFT and DCT are formulated as cyclic convolution forms and mapped into linear arrays. Hence the designed arrays possess outstanding performance in the computing parallelism, computational complexity, the number of I/O channels and I/O bandwidth. As has been discussed in [15], the high I/O bandwidth required for most systolic arrays would limit the computing speeds. Therefore, reducing the high I/O bandwidth is capable of enhancing the computing speeds at the same time. Secondly, we modify the cyclic convolution forms in order to replace multipliers by ROM's efficiently. Fig. 1 illustrates the motivation of this modification. If a multiplier with two time-variant operands a and b is directly replaced by a ROM as shown in Fig. 1(a), the required ROM size equals to 2^{2L} words which are too large to be practical in hardware realizations, where L is the input data wordlength. Based on the modified forms, one of the operands in the multiplier can be fixed. Therefore, one multiplier can be replaced by 2^L words of ROM as shown in Fig. 1(b). Moreover, as has been discussed in [20], a technique named partial sums

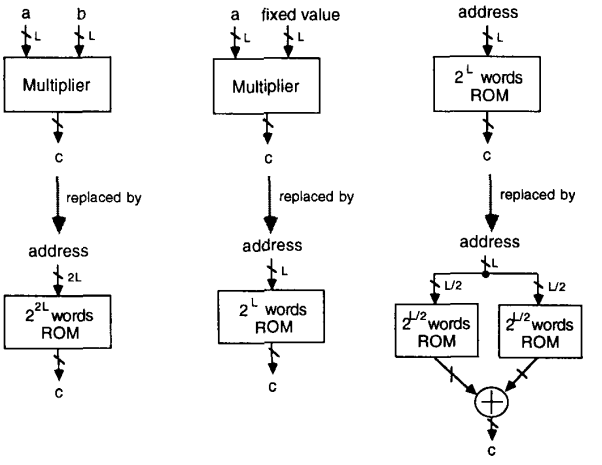


Fig. 1. (a) Direct replacement of multipliers by ROM's. (b) Replacement of multipliers with one fixed operand by ROM's. (c) Partitioning ROM's by using partial sums technique.

can be used to reduce the memory size in DA architectures. This technique can also be applied to the presented designs to reduce the ROM size from 2^L words to $2^{(L/2+1)}$ words as shown in Fig. 1(c). After using these two techniques illustrated in Fig. 1(b) and (c), a multiplier can be efficiently replaced by ROM's with size equal to $2^{(L/2+1)}$ words and an adder. Furthermore, owing to the small ROM size, short ROM access time can be attained to benefit the computing speeds. Considering for example the computation of a 17-point DFT of real inputs, only about 1K words of ROM are needed if the input wordlength is 8 bits. The rest of this paper is organized as follows. Section II presents the new systolic algorithm for 1-D DFT and DCT. Section III illustrates the hardware realizations of the presented algorithm. Section IV gives a conclusion.

II. Algorithm Derivation and Analysis

A. The Derivation for Cyclic Convolution

The 1-D DFT and DCT of the input sequence $\{y(i), i = 0, 1, \cdots, N-1\}$ can be generally expressed as

$$Y(k) = \sum_{i=0}^{N-1} y(i) \times H(i,k); \quad k = 0, 1, \cdots, N-1 \quad (1)$$

where $\{Y(k),\ k = 0, 1, \cdots, N-1\}$ is the output sequence and $\{H(i,k),\ i,k = 0, 1, \cdots, N-1\}$ denotes the kernels of transforms. Generalizing from our previous approaches [8], [14], we can formulate (1) as

$$Y(0) = \sum_{i=0}^{N-1} y(i) \quad (2a)$$

$$Y(g^k) = \{\alpha \times T(g^k) + x(0)\} \times \beta(g^k); \quad k = 1, \cdots, N-1 \quad (2b)$$

where

$$T(g^k) = \sum_{i=1}^{N-1} x(g^i) \times h(g^{i+k}); \quad k = 1,\cdots, N-1 \quad (2c)$$

$$\alpha = \begin{cases} 2; & \text{for 1-D DCT} \\ 1; & \text{for 1-D DFT} \end{cases}$$

$$\beta(g^k) = \begin{cases} \cos\left(\frac{\pi}{2N} \times g^k\right); & \text{for 1-D DCT} \\ 1; & \text{for 1-D DFT} \end{cases}$$

$$h(g^{i+k}) = \begin{cases} (-1)^m \times \cos\left(\frac{\pi}{N} \times g^{i+k}\right); & \text{for 1-D DCT} \\ \exp\left(-\frac{2j\pi}{N} \times g^{i+k}\right); & \text{for 1-D DFT} \end{cases}$$

and the sequence $\{x(i), i = 0, 1,\cdots, N-1\}$ is defined as

$$x(N-1) = y(N-1)$$

$$x(i) = \begin{cases} y(i) - x(i+1); & \text{for 1-D DCT} \\ y(i); & \text{for 1-D DFT} \end{cases}; \quad i = 0, 1,\cdots, N-2.$$

The value of m is determined by the following equation

$$g^i \times g^k = g^{i+k} + m \times N; \quad i, k = 1, 2,\cdots, N-1 \quad (2d)$$

where "g^i" denotes the result of "g^i modulo N operation" for short and "g" is a primitive element. The details of the derivation from (1) to (2) for the DFT and DCT can be found in Appendix A and Appendix B, respectively. From (2c), we know that the sequence $\{T(k), k = 1,\cdots, N-1\}$ is the cyclic convolution of the sequence $\{x(i), i = 1,\cdots, N-1\}$ and the kernels $\{h(g^{i+k}), i, k = 1,\cdots, N-1\}$. To illustrate the difference between (1) and (2) clearly, the matrix representations of 5-point DFT based on (1) and (2) are individually shown in the following

$$\begin{bmatrix} Y(0) \\ Y(1) \\ Y(2) \\ Y(3) \\ Y(4) \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 \\ 1 & W^1 & W^2 & W^3 & W^4 \\ 1 & W^2 & W^4 & W^6 & W^8 \\ 1 & W^3 & W^6 & W^9 & W^{12} \\ 1 & W^4 & W^8 & W^{12} & W^{16} \end{bmatrix} \begin{bmatrix} x(0) \\ x(1) \\ x(2) \\ x(3) \\ x(4) \end{bmatrix} \quad (3a)$$

$$\begin{bmatrix} Y(0) \\ Y(2) \\ Y(4) \\ Y(3) \\ Y(1) \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 \\ 1 & W^4 & W^3 & W^1 & W^2 \\ 1 & W^3 & W^1 & W^2 & W^4 \\ 1 & W^1 & W^2 & W^4 & W^3 \\ 1 & W^2 & W^4 & W^3 & W^1 \end{bmatrix} \begin{bmatrix} x(0) \\ x(2) \\ x(4) \\ x(3) \\ x(1) \end{bmatrix} \quad (3b)$$

where "W" and "g" are assumed to be $\exp(-j2\pi/5)$ and 2, respectively. Note that the elements in the matrix of (3b) have the same value in the same diagonal line exclusive the 1's in the first row and the first column while those of (3a) do not possess similar phenomenon. If (3a) is directly realized by using the linear arrays similar to that in [9], each PE should have one input channel to receive a kernel value, W^i, at each time step. Totally, the number of W^i's to be transmitted to the arrays is N^2. Such arrays require large numbers of I/O channels and high I/O bandwidth. On the other hand, if the special phenomenon of W^i's in the matrix of (3b) is efficiently utilized, the W^i's can be transmitted to the arrays only through one input channel at the array boundary and the number of W^i's to be transmitted to the arrays is only N. Hence, the number of I/O channels and the I/O bandwidth are reduced by a factor N. Moreover, (3b) should induce high computing parallelism and low computational complexity as analyzed in [8]. In the following subsection, we shall further modify the algorithm so that the multipliers in the arrays can be replaced by ROM's efficiently.

B. The Derivation for Efficient ROM Substitution

Fig. 1 illustrates the basic motivation for the further derivation of (2). As shown in Fig. 1(a), if a multiplier with two time-variant operands a and b is replaced by a ROM, the required ROM size equals to 2^{2L} words, where L is the input wordlength. If one of the operands in the multiplier is fixed, then the multiplier can be replaced by a ROM with size equal to 2^L words as shown in Fig. 1(b). Since the ROM is used to perform multiplications, it can be further replaced by two small ROM's and an adder as shown in Fig. 1(c). The size of each ROM equals to $2^{L/2}$ words and the total size of the ROM's equals to $2^{L/2+1}$ words. This partition scheme is similar to the partial sums technique used in [20]. As can be noted, the partition scheme induces an additional adder although the memory size is reduced. Further partitions of the ROM's are possible but the trade-off lies in the cost between memory and adders. It has been analyzed based on an implementation technology that the number of times to partition a ROM with 8-bit addresses is one [20]. In the rest of this section, (2) is further modified such that the multipliers used in an array can be replaced by ROM's based on the method illustrated in Fig. 1(b). Then, we shall partition the ROM's based on the method illustrated in Fig. 1(c). It is arbitrarily assumed in this paper that the number of times to partition the ROM's is one.

Considering (3b), the sequence $\{x(i), i = 0, 1,\cdots, N-1\}$ is time-variant. The W^i's are transmitted from the PE at the array boundary through the internal PE's for proper computations. Hence, the two operands of multipliers in the PE's are both time-variant. As illustrated in Fig. 1(a), such multipliers cannot be replaced by ROM's efficiently. In order to efficiently replace multipliers by ROM's, (2c) can be reformulated as

$$T(g^k) = \sum_{i=1}^{N-1} h(g^i) \times x(g^{i-k}); \quad k = 1,\cdots, N-1 \quad (4)$$

based on the commutative property of cyclic convolution. To illustrate that (4) benefits the efficient ROM substitution, the matrix representation of 5-point DFT based on

(4) can be expressed as

$$\begin{bmatrix} T(2) \\ T(4) \\ T(3) \\ T(1) \end{bmatrix} = \begin{bmatrix} x(1) & x(2) & x(4) & x(3) \\ x(3) & x(1) & x(2) & x(4) \\ x(4) & x(3) & x(1) & x(2) \\ x(2) & x(4) & x(3) & x(1) \end{bmatrix} \begin{bmatrix} W^2 \\ W^4 \\ W^3 \\ W^1 \end{bmatrix} \quad (5)$$

where "W" and "g" are assumed to be $\exp(-j2\pi/5)$ and 2, respectively. It is noted from (5) that the input data $x(i)$'s are transmitted among PE's and are time-variant, but the $(N-1)$ W^i's are respectively allocated to $(N-1)$ PE's and are time-invariant. Hence, one operand in a multiplier is fixed and the multipliers used to implement the multiplications in (5) can be replaced by ROM's and adders based on the methods illustrated in Fig. 1(b) and (c). In the following section, the architectures that realize the DFT and DCT based on this algorithm are presented.

III. Array Realizations

A. The Hardware Architecture for 1-D 5-Point DFT

Based on the presented algorithm, the 1-D 5-point DFT of the input sequence $\{x(i),\ i = 0,1,\cdots,N-1\}$ can be formulated as

$$Y(0) = \sum_{i=0}^{4} x(i) \quad (6a)$$

$$Y(2^k) = x(0) + T(2^k); \quad k = 1,\cdots,4 \quad (6b)$$

$$T(2^k) = \sum_{i=1}^{4} W^{2^i} \times x(2^{i-k}); \quad k = 1,\cdots,4 \quad (6c)$$

where "W" and "g" are assumed to be $\exp(-j2\pi/5)$ and 2, respectively. Fig. 2 shows the memory-based systolic array for 5-point DFT where consecutive DFT calculations are assumed. The first input and output data bundles are denoted as $x1(i)$ and $Y1(k)$, the second input and output data bundles are denoted as $x2(i)$ and $Y2(k)$, and so on. The time instants for the input and output data bundles are indicated in the same row of each data. Analyzing the array shown in Fig. 2, the input data are piped in from the left-most PE while the output data are drained out from the right-most PE. Hence, the I/O channels are all located at the boundary PE's which makes the I/O cost independent of the array length N. The $(N-1)$ twiddle factors, W^i's are stored in $(N-1)$ PE's, respectively. As a result, each multiplier in the PE's can be efficiently replaced by two small ROM's and an adder as illustrated in Fig. 1(b) and (c). Fig. 2(b) and (c) illustrate the functions and structures of the PE's in the array. Fig. 2(d) illustrates the permutation stage of the array which performs the permutations and order arrangements of the input data. A RAM buffer and an address generation unit are used to implement the data permutations. To illustrate the advantages of the ROM substitution, the analysis model presented in [30] is used to analyze the hardware and time complexity. If a complex multiplication is implemented by ROM's with size equal to 2^5 words and two 8-bit adders as shown in Fig. 3(a), the cost function of hardware complexity and the time delay are respectively about 272 and $14t$ where t is the gate delay time. If a complex multiplication is implemented by two multipliers as shown in Fig. 3(b), the cost function of hardware complexity and the time delay are about 1920 and $32t$, respectively.

To help analyze the activity of the array shown in Fig. 2, the presented algorithm for 5-point DFT can be expressed as the recursive form

$$y_k^0 = 0; \qquad k = 1,\cdots,4 \quad (7a)$$

$$y_k^i = y_k^{i-1} + W^{2^i} \times x(2^{i-k}); \quad i,k = 1,\cdots,4 \quad (7b)$$

$$Y(2^k) = x(0) + y_k^4; \qquad k = 1,\cdots,4 \quad (7c)$$

$$z_0^i = z_0^{i-1} + x(i); \qquad i = 1,\cdots,4 \quad (7d)$$

$$z_0^0 = x(0) \quad (7e)$$

$$Y(0) = z_0^4. \quad (7f)$$

Fig. 4 depicts the activity of the array shown in Fig. 2 at successive six clocks from $t = 8$ to $t = 13$, where yp_k^j denotes the iterated result y_k^j in (7) of the pth data bundle. The right-most PE has a 2-bit control link named "Tag2" and all the PE's have the 1-bit control links named "Tag1." Link "Tag1" is used to indicate the PE's to select the appropriate input data, and link "Tag2" is used to indicate the right-most PE to perform the correct operations. Based on the control scheme named "*Tag control*" [18], the data in the local registers of each PE can be controlled from the input channels at the extreme ends of a linear array. The hardware overheads paid for this control scheme in each PE are about a 1-bit link and one multiplexer. The time overhead is $(N-1)T_{\text{cycle}}$, where T_{cycle} is the cycle time of the array. However, the time overhead can be skipped by overlapping the computation time of two consecutive DFT calculations. As depicted in Fig. 2, there is no extra time between the data bundle $x1(i)$'s and the data bundle $x2(i)$'s, or between the data bundle $Y1(k)$'s and the data bundle $Y2(k)$'s. In other words, this control scheme should give overhead to the latency time instead of the average computation time for a DFT problem. The latency time is defined as the consumption time from the input of the first datum to the output of the final datum for a DFT calculation. The average computation time is defined as the minimum execution time between the first datum of the current data bundle and the first datum of the next data bundle for consecutive DFT calculations. This phenomenon can also be checked from the array activity shown in Fig. 4. From $t = 8$ to $t = 13$, the array calculates the first DFT problem by using $x1(i)$'s and simultaneously fetches $x2(i)$'s for the second DFT problem. It is such a concurrent computing style that favors the average computation time of the presented DFT array.

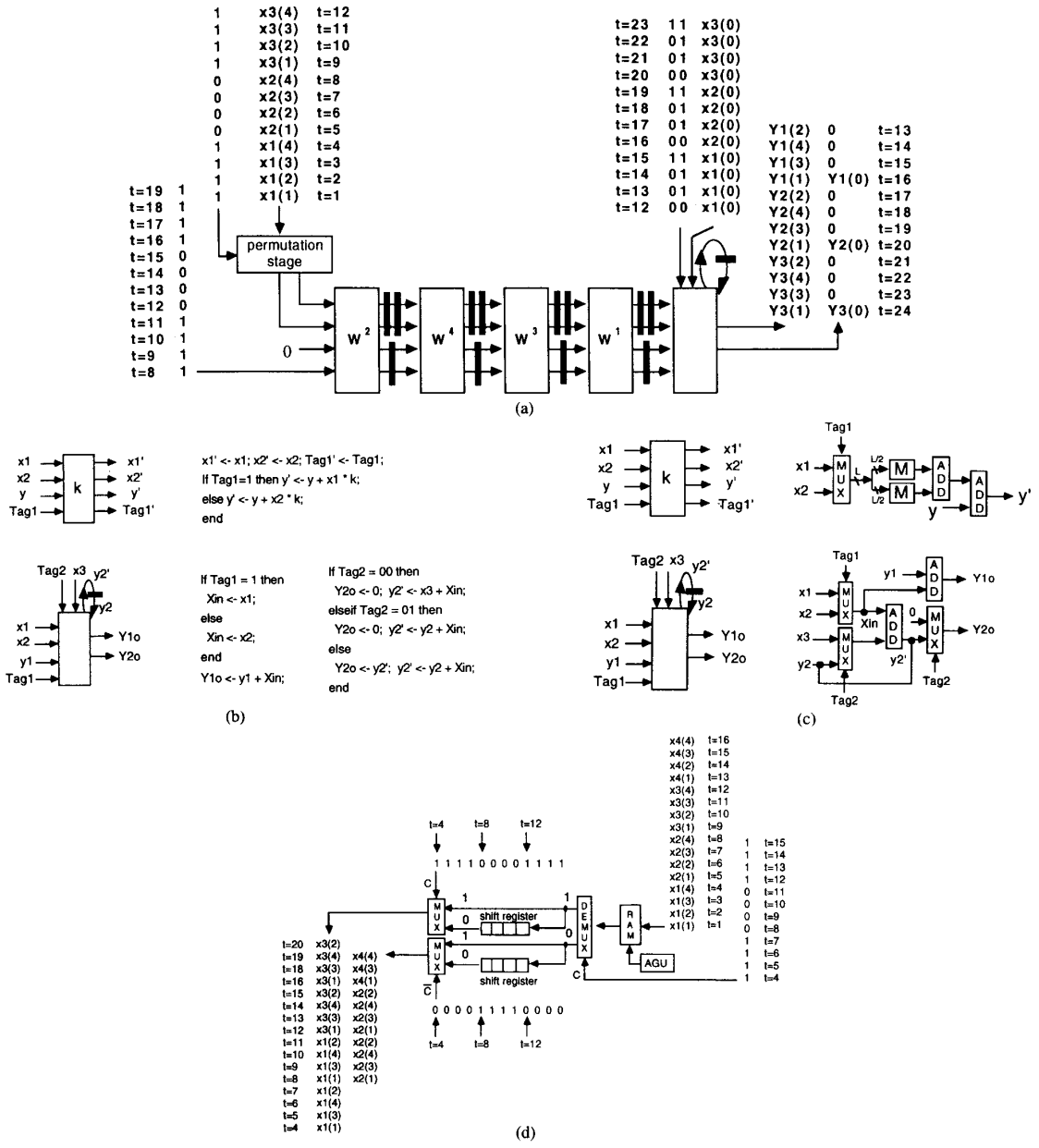


Fig. 2. (a) The memory-based systolic array for 5-point DFT where *xi*( ) and *Yi*( ) denote the I/O values of the *i*th problem. (b) The functions of the PE's used in the array. (c) The architectures of the PE's used in the array where L is the wordlength. (d) The structure of the permutation stage where AGU denotes address generation unit.

Due to the modularity of the presented array, it is very easy to extend the 5-point DFT array to the long length one based on the same topology. It is noted from Fig. 2 that the overall hardware cost of the DFT array is linearly proportional to N, and the number of I/O channels is independent of N. If N becomes large enough to induce unacceptable hardware cost, the efficient partition techniques which have been investigated in our previous paper [8] can be used to realize the presented array with a reasonable number of PE's.

Including the cost of permutation stage, the overall hardware cost of the designed array for N-point DFT consists of $(N-1) \times 2^{L/2+1}$ words of ROM, $2N$ adders, $N+5$ multiplexers, one RAM module with size equal to $2N-2$ words, an address generation unit, and $2N-2$ shift registers, where the number of ROM partition time is assumed to be one. It is seen that the hardware cost of the presented array is only proportional to N. However, the architectures using DA approach should require $N \times 2^{N/2+1}$ words of ROM, $2N$ adders, and $2N$ shift registers

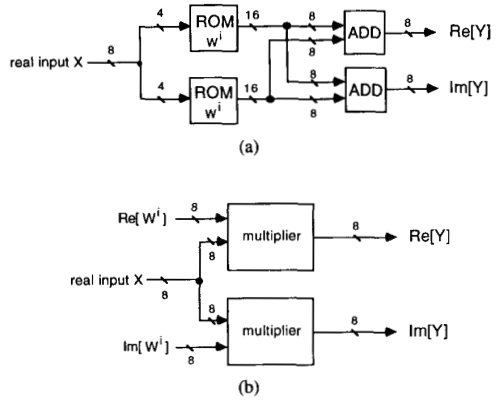


Fig. 3. (a) The complex multiplication $Y = X^*W^i$ which is implemented by ROM's and adders. (b) The complex multiplication $Y = X^*W^i$ which is implemented by two multipliers, where Re[Z] and Im[Z] denote the real part and imaginary part of Z, respectively.

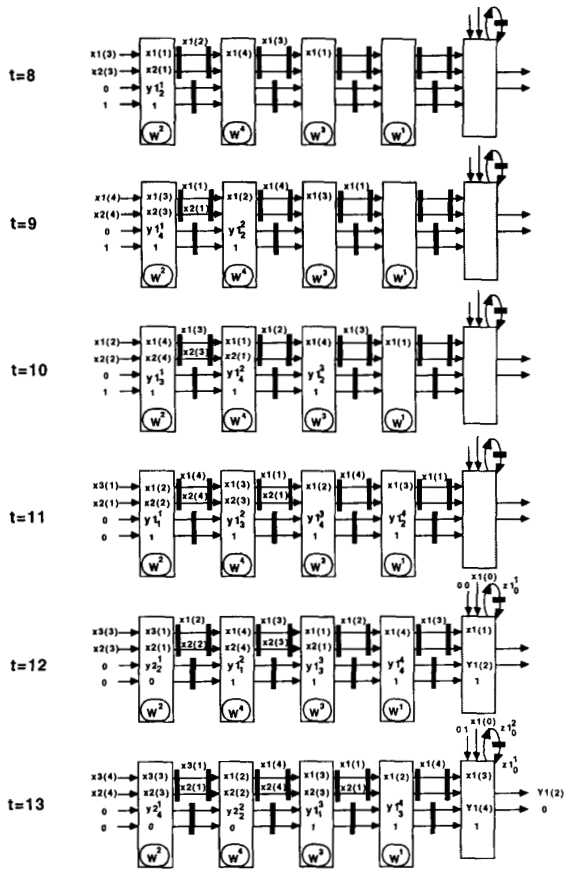


Fig. 4. The systolic array in Fig. 2 at six successive time instants.

for computing an N-point transform, where the number of ROM partition time is also assumed to be one. It is noted that the hardware cost of the architectures using DA approach increases exponentially as N increases. Therefore, the presented approach requires less hardware cost than the DA approach does when of $N > L$, which occurs in most DFT applications. In the following, the analysis model presented in [30] is used to evaluate the cost of the designed arrays. A cost function derived in [30] is adopted for objective cost evaluation. According to this model, the cost function of the permutation stage over that of the whole DFT array is about $(37N + 73)/(336N + 135)$. As N becomes large, the cost of the permutation stage is about 11% of the overall cost of the N-point DFT array, where the wordlength is assumed to be 8 bits. This percentage is not affected by the values of N.

To sum up, the designed array has several distinctive features. In the first place, the input data and the computed results are piped in and drained out from the I/O channels at the extreme ends of a linear array. Hence, low I/O bandwidth and a small number of I/O channels can be achieved. Secondly, all the multiplications are efficiently realized by ROM's and adders to attain the benefits in hardware realization and computing speeds. Thirdly, the presented architecture takes the advantages of systolic arrays such as locality, modularity, pipelinability, and parallelism among PE's. Also, it utilizes the memory-based implementation to attain low hardware cost and high computing speeds inside PE's.

B. The Hardware Architecture for 1-D 7-Point DCT

Based on the presented algorithm, the 1-D 7-point DCT can be formulated as

$$Y(0) = \sum_{i=0}^{6} y(i) \tag{8a}$$

$$Y(3^k) = \{2 \times T(3^k) + x(0)\} \cos\left[\frac{\pi}{14} \times 3^k\right]; \quad k = 1,\cdots,6 \tag{8b}$$

$$T(3^k) = \sum_{i=1}^{6} x_s(i-k) \times \cos\left(\frac{\pi}{7} \times 3^i\right); \quad k = 1,2,\cdots,6 \tag{8c}$$

where

$$x_s(i) = (-1)^m \times x'(i)$$

and $x'(i)$ is defined as

$$x'(i) = \begin{cases} x(3^i); & \text{if } i > 0 \\ x'(6+i); & \text{if } i \le 0 \end{cases}.$$

The value of m used above is determined by the following equation

$$3^i + m \times 7 = 3^{i-k} \times 3^k; \quad i,k = 1,2,\cdots,6 \tag{9}$$

where "3^j" denotes the result of "3^j modulo 7" operation for short. For the purpose of showing the presented

algorithm clearly, (8c) can be written in a matrix-vector form as

$$\begin{bmatrix} T(3) \\ T(2) \\ T(6) \\ T(4) \\ T(5) \\ T(1) \end{bmatrix} = \begin{bmatrix} -x(3) & x(2) & x(6) & -x(4) & x(5) & x(1) \\ x(1) & x(3) & x(2) & -x(6) & -x(4) & -x(5) \\ x(5) & x(1) & x(3) & -x(2) & -x(6) & -x(4) \\ x(4) & x(5) & x(1) & -x(3) & -x(2) & -x(6) \\ x(6) & x(4) & -x(5) & x(1) & x(3) & -x(2) \\ x(2) & x(6) & x(4) & x(5) & x(1) & x(3) \end{bmatrix} \cdot \begin{bmatrix} \cos(2a) \\ \cos(6a) \\ \cos(4a) \\ \cos(5a) \\ \cos(1a) \\ \cos(3a) \end{bmatrix} \tag{10}$$

where "a" denotes $\pi/7$. Using the symmetry property of the cosine kernels [14], (10) can be written as

$$\begin{bmatrix} T(3) \\ T(2) \\ T(6) \\ T(4) \\ T(5) \\ T(1) \end{bmatrix} = \begin{bmatrix} -x(3) & x(2) & x(6) & -x(4) & x(5) & x(1) \\ x(1) & x(3) & x(2) & -x(6) & -x(4) & -x(5) \\ x(5) & x(1) & x(3) & -x(2) & -x(6) & -x(4) \\ x(4) & x(5) & x(1) & -x(3) & -x(2) & -x(6) \\ x(6) & x(4) & -x(5) & x(1) & x(3) & -x(2) \\ x(2) & x(6) & x(4) & x(5) & x(1) & x(3) \end{bmatrix} \cdot \begin{bmatrix} \cos(2a) \\ \cos(6a) \\ \cos(4a) \\ -\cos(2a) \\ -\cos(6a) \\ -\cos(4a) \end{bmatrix}. \tag{11}$$

That is,

$$\begin{bmatrix} T(3) \\ T(2) \\ T(6) \\ T(4) \\ T(5) \\ T(1) \end{bmatrix} = \begin{bmatrix} -\{x(3)-x(4)\} & \{x(2)-x(5)\} & \{x(6)-x(1)\} \\ \{x(6)+x(1)\} & \{x(3)+x(4)\} & \{x(2)+x(5)\} \\ \{x(2)+x(5)\} & \{x(6)+x(1)\} & \{x(3)+x(4)\} \\ \{x(3)+x(4)\} & \{x(2)+x(5)\} & \{x(6)+x(1)\} \\ \{x(6)-x(1)\} & -\{x(3)-x(4)\} & \{x(2)-x(5)\} \\ \{x(2)-x(5)\} & \{x(6)-x(1)\} & -\{x(3)-x(4)\} \end{bmatrix} \cdot \begin{bmatrix} \cos(2a) \\ \cos(6a) \\ \cos(4a) \end{bmatrix}. \tag{12}$$

Similar to (5), the elements in the matrix of (12) located in the same diagonal line consist the same data operands $x(i)$'s. But they are obtained by performing different kinds of operations on the data $x(i)$'s. All the required combinations of the operands are precalculated and the appropriate one is selected to be the ROM address by using the "*Tag control*" technique discussed in [18]. Fig. 5 shows the memory-based systolic array for 7-point DCT where consecutive DCT calculations are assumed. The notations of the I/O data bundles used in Fig. 5 are the same as those in Fig. 2. Fig. 5(a) shows the presented array architecture for 7-point DCT. The functions and architectures of the PE's in the array are shown in Fig. 5(b) and (c), respectively. The activity of the DCT array shown in Fig. 5(a) is very similar to that of the DFT array in Fig. 2(a) except that some control problems and preprocessing stages are introduced. The recursive generations of the control signals required in the array can be done by using a linear ROM buffer accompanied with a mod-6 counter generating the required ROM addresses. The preprocessing stages consist of the permutation stage and the add/sub stage which are shown in Fig. 5(d) and (e), respectively. The permutation stage is used to perform the input data permutations and to generate the required data sequences. It is composed of a subtractor, a RAM buffer, multiplexing circuits, and delay elements. The add/sub stage is to generate the required combinations of data operands $x(i)$'s in the array.

From (8b), it is seen that some multiplications are required to compute $Y(3^k)$ from $T(3^k)$. And, to obtain the final DCT coefficients, some scaling operations should be performed on $Y(3^k)$. These multiplications and the scaling operations can be simultaneously performed in the right-most PE of the array. To consider the hardware cost and the computing speeds at the same time, a pipelined multiplier is utilized to compute the multiplications and scaling operations involved in the right-most PE. The number of pipeline stages in the multiplier should be appropriately chosen such that the cycle time of the presented array can

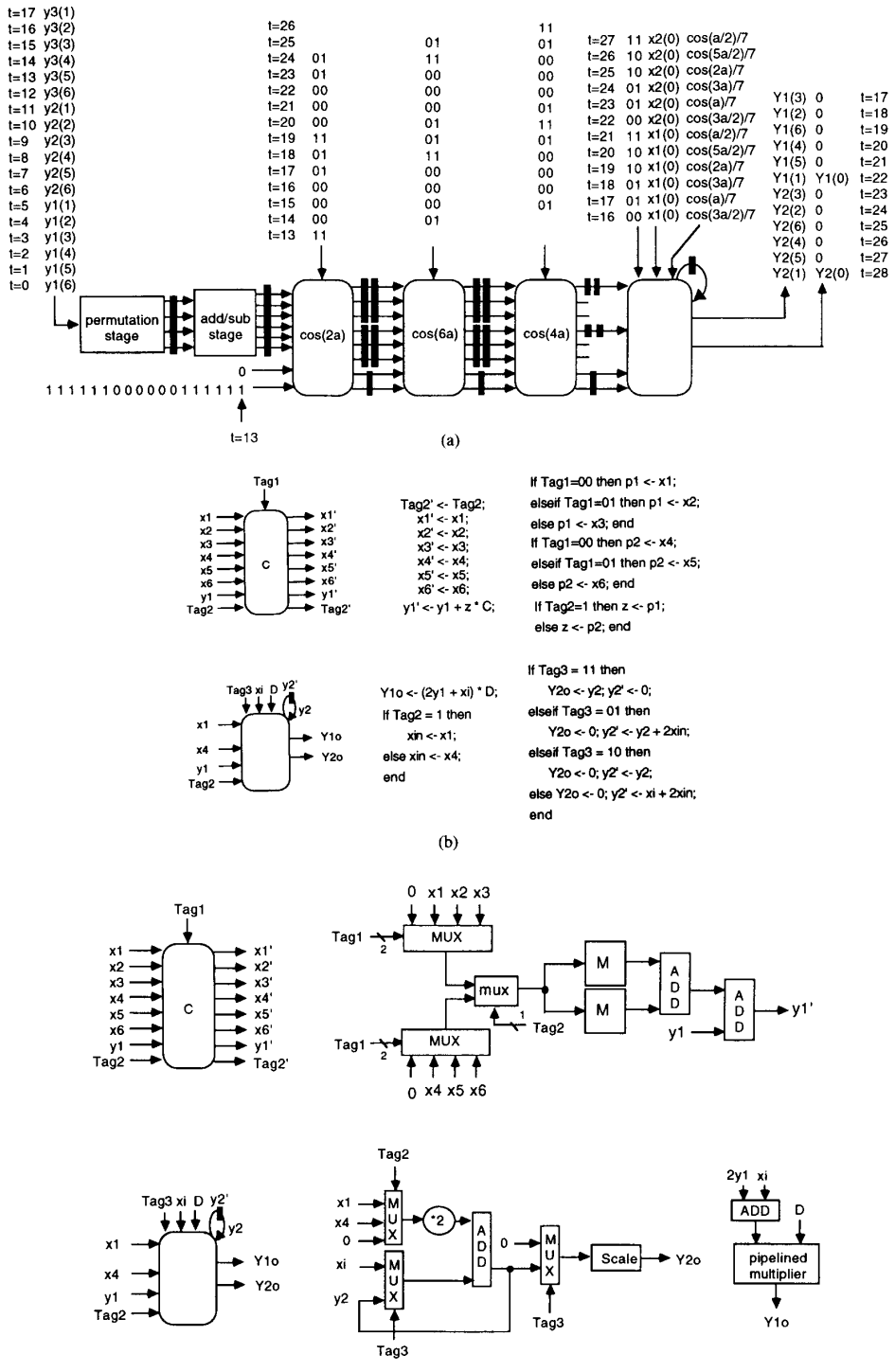


Fig. 5. (a) The memory-based systolic array for 7-point DCT with symmetry property, where $a = \pi/7$, $xi(\ )$ and $Yi(\ )$ denote the I/O values of the ith problem. (b) The functions of PE's used in the array. (c) The architectures of the PE's used in the array where L is the word-length. (d) The structure of the permutation stage where AGU denotes address generation unit. (e) The structure of the add/sub stage.

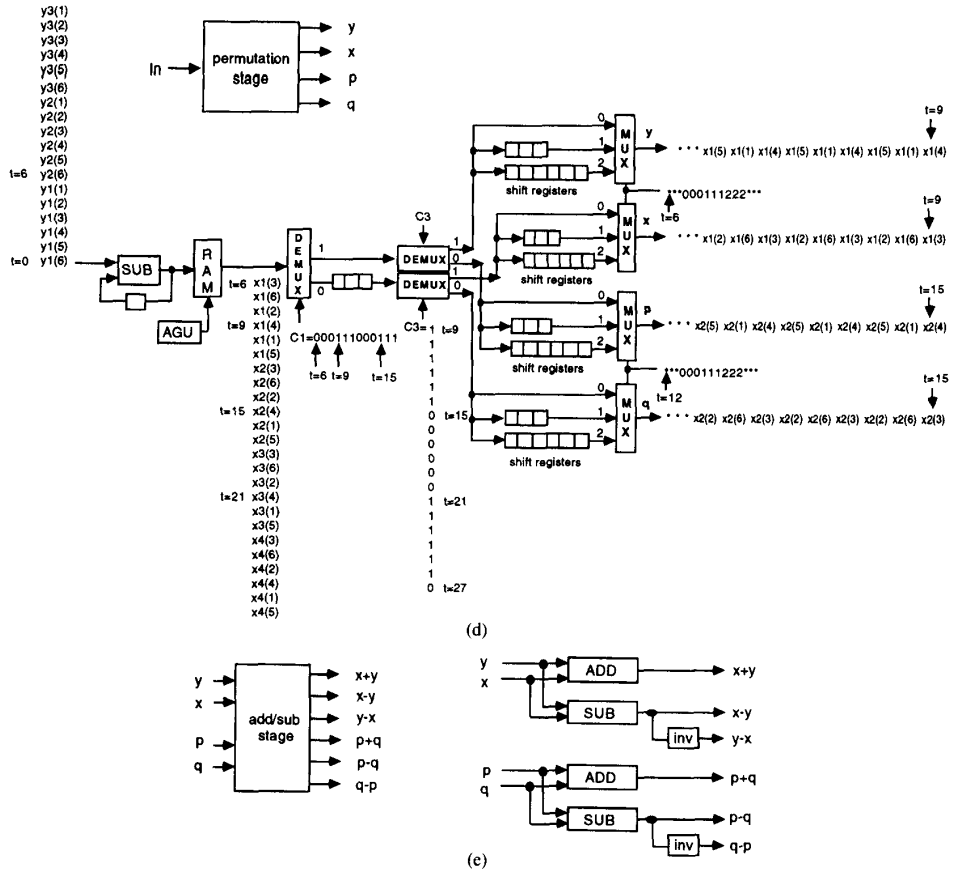


Fig. 5. (*Continued*)

be reduced as most as possible. In addition, scaling the $Y(0)$ component can be done by table look-up. Totally, besides a pipelined multiplier, the designed array requires only $N \times 2^{(L/2+1)}$ words of ROM to compute an N-point DCT in $(N-1)$ cycles.

Similar to the DFT array, the cost of the preprocessing stages as well as the overall cost of the DCT array are both linearly proportional to N. This fact reveals that the percentage of the cost occupied by the preprocessing stages over the overall cost of the DCT array is finite and not affected by the values of N. Moreover, all the features of the DFT array are also possessed in the DCT array, which include high computing speeds, low hardware cost, low hardware complexity of PE's, a small number of I/O channels and low I/O bandwidth. As a whole, the presented DFT and DCT arrays not only provide good performance in the hardware complexity of PE's, I/O cost, and throughput rate, but also possess the feasible VLSI structures inside and among PE's.

IV. Conclusion

The efficient memory-based VLSI array designs for the DFT and DCT have been presented. A new design approach for the designed arrays has also been presented in this paper. This approach has been shown to provide the method to derive systolic algorithms for linear arrays and give an efficient technique to replace multipliers by ROM's. Two linear systolic arrays have been designed for the DFT and DCT individually. The designed arrays have been shown to have good performance in the architectural topology (local and regular connection), computing speeds, hardware complexity, the number of I/O channels, and I/O bandwidth.

In a few words, the presented approach formulates the DFT and DCT as cyclic convolutions, maps the convolutions into VLSI arrays, and considers the issue of efficient hardware implementation such as using small ROM's for multipliers. This approach can also be applied to other applications. For example, the transform kernels of the discrete sine transform (DST) have similar properties to those of the DCT [10]. Besides, the kernels of the discrete Hartley transform (DHT) and the DFT have similar forms to each other. Therefore, based on the presented approach, the DST and DHT can also be formulated as cyclic convolutions, mapped into VLSI arrays, and implemented by using small ROM's instead of multipliers. If long length DFT and DHT are considered, the efficient partition techniques which have been discussed in our

previous paper [8] can be used to effectively reduce the hardware cost in the designed arrays.

A restriction for the presented approach is that the transform length N should be prime. This restriction should not put a severe limitation for the DFT because a non-prime length input sequence can be appended by zeros to attain prime length. The appending operation affects the energy of the output sequence but gives no influence to the shape of it. The restriction of prime length give a more severe limitation to the DCT. However, we may utilize some application properties to avoid this restriction. For example, DCT is most widely used in image coding. One problem facing the DCT coding is the blocking effect [31]. The overlap method is one of the remedies for this problem [31]. We may utilize the overlap method to append pixels from non-prime length to attain prime length, and hence avoid the restriction of prime length by solving the blocking effect.

Appendix A

The 1-D N-point DFT of an input sequence $\{y(i), i = 0,\cdots, N-1\}$ is defined as

$$Y(k) = \sum_{i=0}^{N-1} y(i) \times W^{ik}; \quad k = 0,1,\cdots, N-1 \quad \text{(A1)}$$

where "W" is assumed to be $\exp(-j2\pi/N)$. To formulate (A1) as a cyclic convolution [17], the periodic property of "$W^N = 1$" and the I/O data permutations based on the structure of *Galois Field* are utilized.

If N is a prime number, there exists some number "g," not necessarily unique, such that there is a one-to-one mapping from the integers $\{i, i = 1,2,\cdots, N-1\}$ to the integers $\{j, j = 1,2,\cdots, N-1\}$ given by

$$j = g^i_{\text{modulo N}}. \quad \text{(A2)}$$

In the following, "g^i" denotes the result of "g^i modulo N operation" for short. The DFT in (A1) will be rewritten with i and k as the powers of a primitive element "g." Because i and k take on the value zero which is not a power of "g," the zero frequency component must be treated specifically, i.e.,

$$Y(0) = \sum_{i=0}^{N-1} y(i) \quad \text{(A3)}$$

$$Y(k) = y(0) + \sum_{i=1}^{N-1} y(i) \times W^{ik}; \quad k = 1,\cdots, N-1. \quad \text{(A4)}$$

To replace i, k by "g^i", "g^k" and introduce a sequence $x(i)$ which equals the input sequence $y(i)$ for $i = 0,1,\cdots, N-1$, (A4) is rewritten as

$$Y(g^k) = x(0) + \sum_{i=1}^{N-1} x(g^i) \times W^{g^{i+k}}; \quad k = 1,\cdots, N-1. \quad \text{(A5)}$$

Introducing the sequence $x(i)$ is to unify the representations for DFT and DCT. Thus (A5) shows the cyclic convolution representation of (A1).

Appendix B

The 1-D N-point DCT is defined as

$$Y(k) = \sum_{i=0}^{N-1} y(i) \cos\left[\frac{2\pi(2i+1)k}{4N}\right]; \quad k = 0,1,\cdots, N-1 \quad \text{(B1)}$$

where $\{y(i), i = 0,1,\cdots, N-1\}$ is the input sequence and $\{Y(k), k = 0,1,\cdots, N-1\}$ is the output sequence. It can be shown that the DCT defined in (B1) can be formulated as

$$Y(k) = \{2T(k) + x(0)\} \cos\left[\frac{k\pi}{2N}\right]; \quad k = 0,1,\cdots, N-1 \quad \text{(B2)}$$

where

$$T(k) = \sum_{i=1}^{N-1} x(i) \cos\left[\frac{\pi ik}{N}\right]; \quad k = 0,1,\cdots, N-1 \quad \text{(B3)}$$

and $x(i)$ is another sequence defined as

$$x(N-1) = y(N-1)$$
$$x(i) = y(i) - x(i+1); \quad i = 0,1,\cdots, N-2. \quad \text{(B4)}$$

Similarly, if N is a prime number, the mapping relationship defined in (A2) can also be applied to reformulate (B2) and (B3). Therefore, (B2) and (B3) can be written with i and k as powers of the primitive element "g." Because i and k take on the value zero which is not a power of "g," the zero frequency component must be treated specially, i.e.,

$$Y(0) = \sum_{i=0}^{N-1} y(i) \quad \text{(B5)}$$

$$Y(k) = \{2T(k) + x(0)\} \cos\left[\frac{k\pi}{2N}\right]; \quad k = 1,\cdots, N-1 \quad \text{(B6)}$$

where

$$T(k) = \sum_{i=1}^{N-1} x(i) \cos\left[\frac{\pi ik}{N}\right]; \quad k = 1,\cdots, N-1 \quad \text{(B7)}$$

To replace i and k by "g^i" and "g^k", (B7) can be finally written as [14]

$$T(g^k) = \sum_{i=1}^{N-1} x(g^i) \times C^i_k; \quad k = 1,2,\cdots, N-1 \quad \text{(B8)}$$

where

$$C_k^i = \begin{cases} \cos\left(\frac{\pi}{N} \times g^{i+k}\right) & \text{if } m \text{ is even.} \\ -\cos\left(\frac{\pi}{N} \times g^{i+k}\right) & \text{if } m \text{ is odd.} \end{cases}$$

and "m" is an integer determined by the following equation:

$$g^i \times g^k = g^{i+k} + m \times N; \quad k = 1, 2, \cdots, N-1. \quad \text{(B9)}$$

Now (B5), (B6), and (B8) are the computational equations for 1-D DCT, and (B8) is the primary one among them. It is seen that (B8) is a cyclic convolution representation [14].

REFERENCES

[1] H. T. Kung, "Why systolic architectures?" *Comput. Mag.*, vol. 15, pp. 37–45, Jan. 1982.

[2] ——, "Special purpose devices for signal and image processing: An opportunity in very large scale integration (VLSI)," *Proc. SPIEL*, vol. 241, pp. 76–84, 1980.

[3] 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.

[4] L. W. Chang and M. Y. Chen, "A new systolic array for discrete Fourier transform, "*IEEE Trans. Acoust. Speech, Signal Processing*, vol. 36, pp. 1665–1667, Oct. 1988.

[5] M. A. Bayoumi, G. A. Jullien, and W. C. Miller, "A VLSI array for computing the DFT based on RNS," *Proc. ICASSP*, pp. 2147–2150, 1986.

[6] T. E. Curtis and J. T. Wickenden, "Hardware-based Fourier transforms: Algorithms and architectures," *Proc. Inst. Elec. Eng.*, vol. 130, Pt.F, pp. 423–432, Aug. 1983.

[7] C. D. Thompson, "Fourier transforms in VLSI," *IEEE Trans. Computers*, vol. C-32, pp. 1047–1057, Nov. 1983.

[8] C. M. Liu and C. W. Jen, "A new systolic array algorithm for discrete Fourier transform," in *Proc. ISCAS*, pp. 2212–2215, 1991.

[9] U. Totzek and F. Matthiesen, "Two-dimensional discrete cosine transform with linear systolic arrays," in *Proc. Int. Conf. Systolic Arrays*, pp. 388–397, 1989.

[10] N. I. Cho and S. U. Lee, "DCT algorithms for VLSI parallel implementations," *IEEE Trans. Acoust., Speech, Signal Processing*, vol. 38, pp. 121–127, Jan. 1990.

[11] L. W. Chang and M. C. Wu, "A unified systolic array for discrete cosine and sine transforms," *IEEE Trans. Signal Processing*, vol. 39, pp. 192–194, Jan. 1991.

[12] C. Chakrabarti and J. Ja'Ja', "Systolic architectures for the computation of the discrete Hartley and the discrete cosine transforms based on prime factor decomposition," *IEEE Trans. Computers*, vol. 39, pp. 1359–1368, Nov. 1990.

[13] M. H. Lee, "On computing 2-D systolic algorithm for Discrete Cosine Transform," *IEEE Trans. Circuits Syst.*, vol. 37, pp. 1321–1323, Oct. 1990.

[14] J. I. Guo, C. M. Liu, and C. W. Jen, "New systolic arrays for prime length discrete cosine transform," *IEEE Workshop on Visual Signal Processing and Communications*, pp. 67–70, June 1991.

[15] A. L. Fisher and H. T. Kung, "Special-purpose VLSI architectures: General discussions and a case study," in *VLSI and Modern Signal Processing* Ed. by S. Y. Kung *et al.* Englewood Cliffs, NJ: Prentice-Hall, 1985, ch. 8, pp. 154–169.

[16] G.-K. Ma and F. J. Taylor, "Multiplier policies for digital signal processing," *IEEE ASSP Mag.*, pp. 6–20, Jan. 1990.

[17] C. M. Rader, "Discrete Fourier transforms when the number of data samples is prime," *Proc. IEEE*, vol. 56, pp. 1107–1108, 1968.

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

[19] N. Demassieux, G. Concordel, J.-P. Durandeau, and F. Jutand, "An optimized VLSI architecture for a multiformat discrete cosine transform," in *Proc. ICASSP*, pp. 547–550, 1987.

[20] M. T. Sun, T. C. Chen, and A. M. Gottlieb, "VLSI implementation of a 16 × 16 discrete cosine transform," *IEEE Trans. Circuits Syst.*, vol. 36, pp. 610–617, Apr. 1989.

[21] K. K. Chau, I.-F. Wang, and C. L. Eldridge, "VLSI implementation of a 2-D DCT in a compiler," *Proc. ICASSP*, pp. 1233–1236, 1991.

[22] F. Jutand, Z. J. Mou, and N. Demassieux, "DCT architectures for HDTV," in *Proc. ISCAS*, pp. 196–199, 1991.

[23] B. Sikstrom, L. Wanhammar, M. Afghahi, and J. Pencz, "A high speed 2-D discrete cosine transform chip," *Integration: The VLSI Journal*, vol. 5, pp. 159–169, June 1987.

[24] M. T. Sun, T. C. Chen, A. Gottlieb, L. Wu, and M. L. Liou, "A 16 × 16 discrete cosine transform chip," *Visual Commun. and Image Process. II, SPIE*, vol. 845, pp. 13–18, Oct. 1987.

[25] M. T. Sun, L. Wu, and M. L. Liou, "A concurrent architecture for VLSI implementation of discrete cosine transform," *IEEE Trans. Circuits System*, vol. CAS-34, pp. 992–994, Aug. 1987.

[26] T. C. Chen, A. Gottlieb, and M. T. Sun, "VLSI implementation of a 16 × 16 DCT," in *Proc. ICASSP*, pp. 1973–1976, Apr. 1988.

[27] A. M. Gottlieb, M. T. Sun, and T. C. Chen, "A video rate 16 × 16 discrete cosine transform IC," in *Proc. IEEE 1988 Custom Integrated Circuits Conf.*, pp. 8.3.1–8.3.4, May 1988.

[28] J. C. Carlach, P. Penard, and J. L. Sicre, "TCAD: A 27 MHz 8 × 8 discrete cosine transform chip, in *Proc. ICASSP*, pp. 2429–2432, May 1989.

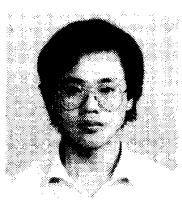
[29] I. Defilippis, U. Sjostrom, M. Ansorge, and F. Pellandini, "A two dimensional 16 point discrete cosine transform chip for real-time video applications," *Douzieme colloque sur le traitement du signal et des images*, pp. 813–816, June 1989.

[30] J. S. Ward *et al.*, "Figures of merit for VLSI implementation of digital signal processing algorithms," *Proc. Inst. Elec. Eng.*, vol. 131, pt.F, pp. 64–70, Feb. 1984.

[31] H. C. Reeve, III, and J. R. Lim, "Reduction of blocking effect in image coding, in *Proc. ICASSP*, pp. 1212–1215, 1983.

Jiun-In Guo received the B.S. degree in electronics engineering from National Chiao Tung University, Hsinchu, Taiwan, in 1989.

He is currently working on his Ph.D. degree in electronics at National Chiao Tung University. His research interests include parallel processing algorithms and architectures, image signal processing, VLSI architecture design, and implementation.



Chi-Min Liu received the B.S. degree in electrical engineering from Tatung Institute of Technology, Taiwan, ROC in 1985, and the M.S. degree and Ph.D. degree in electronics from National Chiao Tung University, Hsinchu, Taiwan, in 1987 and 1991, respectively.

He is currently an Associate Professor of the Department of Computer Science and Information Engineering, National Chiao Tung University, Hsinchu, Taiwan. His research interests include parallel processing algorithms, parallel architectures, adaptive signal processing, image signal processing, radar signal processing, neural networks, fast algorithms, and design automation for DSP VLSI architectures.

Chein-Wei Jen received the B.S. degree from National Chiao Tung University in 1970, the M.S. degree from Stanford University in 1977, and the Ph.D. degree from National Chiao Tung University in 1983.

He is a Professor in the Department of Electronics Engineering and the Institute of Electronics, National Chiao Tung University, Hsinchu, Taiwan. From 1985 to 1986 he was a visiting researcher in University of Southern California. Now he is a director of the Institute of Electronics National Chiao Tung University. His current research interests include VLSI signal processing, VLSI architecture design, design automation and fault tolerant computing.

Dr. Jen is a member of Phi Tau Phi.