Scalable and modular memory-based systolic architectures for discrete Hartley transform

Meher, Pramod Kumar; Srikanthan, Thambipillai; Patra, Jagdish Chandra

2006

Meher, P. K., Srikanthan, T., & Patra, J. C. (2006). Scalable and modular memory-based systolic architectures for discrete Hartley transform. *IEEE Transactions on Circuits and Systems I: Regular Papers*, 53(5), 1065-1077.

<https://hdl.handle.net/10356/94247>

<https://doi.org/10.1109/TCSI.2006.870225>

© 2006 IEEE. Personal use of this material is permitted. Permission from IEEE must be obtained for all other uses, in any current or future media, including reprinting/republishing this material for advertising or promotional purposes, creating new collective works, for resale or redistribution to servers or lists, or reuse of any copyrighted component of this work in other works. The published version is available at: [DOI: <http://dx.doi.org/10.1109/TCSI.2006.870225>].

Downloaded on 26 Apr 2023 13:15:00 SGT

Scalable and Modular Memory-Based Systolic Architectures for Discrete Hartley Transform

Pramod Kumar Meher, Senior Member, IEEE, Thambipillai Srikanthan, Senior Member, IEEE, and Jagdish C. Patra, Member, IEEE

*School of Computer Engineering, Nanyang Technological University, Singapore 639798
(e-mail: aspkmeher@ntu.edu.sg; astsrikan@ntu.edu.sg; aspatra@ntu.edu.sg).*

Abstract—In this paper, we present a design framework for scalable memory-based implementation of the discrete Hartley transform (DHT) using simple and efficient systolic and systolic-like structures for short and prime transform lengths, as well as, for lengths 4 and 8. We have used the proposed short-length structures to construct highly modular architectures for higher transform lengths by a new prime-factor implementation approach. The structures proposed for the prime-factor DHT, interestingly, do not involve any transposition hardware/time. Besides, it is shown here that an N -point DHT can be computed efficiently from two $(N/2)$ -point DHTs of its even- and odd-indexed input subsequences in a recursive manner using a ROM-based multiplication stage. Apart from flexibility of implementation, the proposed structures offer significantly lower area-time complexity compared with the existing structures. The proposed schemes of computation of the DHT can conveniently be scaled not only for higher transform lengths but also according to the hardware constraint or the throughput requirement of the application.

Index Terms—Discrete Fourier transform (DFT), discrete Hartley transform (DHT), distributed arithmetic, ROM-based architecture, systolic arrays, VLSI.

I. INTRODUCTION

OVER the years, the discrete Hartley transform (DHT) [1] has been established as a potential tool for signal processing and communication applications, e.g., computation of circular convolution, and deconvolution [2], [3], interpolation of real-valued signals [4], image compression [5], [6], error control coding [7], adaptive filtering [8], multi-carrier modulation and many other applications [9]–[11]. The DHT is popular due to its real-valued and symmetric transform kernel that is identical to that of its inverse. Not only it is used as a real-valued alternative to the discrete Fourier transform (DFT), but also it is used for more efficient computation of the DFT, and the other widely used unitary transforms like the discrete cosine transform (DCT) and the discrete sine transform (DST) [11], [12]. Several algorithms have, therefore, been reported in the literature for fast computation of the DHT [12]–[14]. Nevertheless, there is a strong need of dedicated processors for high-speed computation of the transform coefficients to meet the requirements of real-time signal processing and digital multimedia communication

systems. For example, in a discrete multi-tone (DMT) modulation-based digital subscriber line (DSL) transceiver, it is necessary to compute the transform of the order as high as 4096 at sampling rate up to 44.16 MHz [11]. Similarly, in a video processor it is necessary to compute $O(10^6)$ of 8-point transform samples for encoding/decoding of an image of (512×512) pixels. Moreover, the computational demand has been increasing with time along with the widespread use of multimedia communication. At the same time, the embedded and portable applications continue to impose serious limitations on the amount of hardware involved and the rate of energy consumption. To meet the challenges of ever-growing computational demand with minimal of hardware and power, several attempts have been made for fast implementation of the DHT in bit-level as well as word-level VLSI structures [15]-[26].

Systolic architectures are now established as the most popular class of VLSI structures for computation intensive digital signal processing applications due to simplicity of their processing elements (PE), modularity of their structure, regular and nearest neighbor interconnections between the PEs, high-level of pipelinability, small chip-area and low-power consumption [27], [28]. In systolic arrays, the desired data is pumped rhythmically in a regular interval across the PEs to yield high throughput by fully pipelined processing. Chakrabarti and Jaja [16] have reported efficient modular bit-level systolic architecture for computation of the DHT by prime-factor decomposition. The structure of [16] however, involves significant memory and time for transposition of intermediate results. Besides, it does not support any variation for the hardware-throughput tradeoff. The bit-level architecture for the DHT and other orthogonal transforms of reference [17] also similarly does not support any hardware-throughput tradeoff, and involves high latency for large transform lengths. Several systolic architectures using the CORDIC circuits and Given rotors have been reported for computation of the DHT [18]–[21]. Due to high regularity, compactness, and high throughput rate, the memory-based structures are, however, gaining more popularity over those based on the CORDIC circuits and Given rotors [22], [23]. Growing interest is also observed in the recent years on field-programmable gate array (FPGA) implementation of the DHT using distributed arithmetic (DA) [24], [25]. The structures of [24], [25] are quite suitable for FPGA, but they are not attractive for pipelined VLSI implementation of long-length DHT due to recursive accumulation loop and global communication requirement. The ROM-based structure suggested in [22] involves ROM of $O(2^{(L+1)})$ words, where the hardware-cost increases exponentially with the word length L . It has a latency of N cycles, which is too long for large values of transform length N . The DA-based DHT structures of [23]–[25] require ROM of $O(2^N)$ words which is very high for large values of N . For $N = 32$ the ROM size of these structures exceed 1 GB. Therefore, it becomes practically unrealizable for large transform lengths. Keeping these on view, in this paper, we aim at both DA-based and direct-ROM-based design of modular systolic hardware, which is suitable for computation of the DHT of small, as well as, large transform lengths.

It is further observed that the existing structures offer rigid design solutions, and do not support flexibilities for simultaneous scaling of transform length, hardware and throughput according to the requirement of the applications, while such flexibilities are often necessary for modern day digital consumer products, and also important for settling the

power-speed-hardware tradeoff. Recently, some schemes using optimal pipelined modules have been suggested for flexible implementation of the DHT of composite and even transform lengths [26]. The convolution property, inherent simplicity, and real-valued nature of the DHT kernel are, however, still to be exploited fully so as to arrive at more efficient, modular and scalable architectures. In this paper, we have used these properties of the DHT to design efficient systolic and systolic-like architectures for scalable implementation of the DHT of prime-transform lengths, and the DHT for lengths $N = 4$ and $N = 8$. Furthermore, for scaling of transform length, we aim at developing efficient structures for prime-factor DHT that do not involve any transposition hardware/time, while it is otherwise usually associated with such decomposition techniques [16].

The organization of the paper is as follows. The mathematical background for implementation of the DHT of even and other composite transform lengths is discussed in the next Section. The scalable implementation of short-length DHTs is described in Section III. The derivation of the modular pipelined structures for composite length DHT and two-dimensional (2-D) DHT are provided in Section IV. The hardware and throughput performance of the proposed structures is discussed in Section V. Finally, the conclusions and scope for future work are presented in Section VI.

II. DEVELOPMENT OF ALGORITHMS FOR COMPOSITE-LENGTH DHT

In this section, we discuss the mathematical background, and derive the necessary algorithms for computation of the DHT of large transform length N , where N is either an even number, a power-of-2 number or $N = N_1 \times N_2$, where N_1 and N_2 are relatively prime.

A. Algorithm for Computation of DHT for Even Values of N and $N = 2^n$

The DHT of a real-valued N -point input vector, $\mathbf{x} = \{x(0), x(1), x(2), \dots, (N - 1)\}$, may be defined as [1]

$$X(k) = \sum_{n=0}^{N-1} x(n) C_N(k, n) \quad (1)$$

Where $C_N(k, n) = \text{cas}(2\pi kn/N) = \cos(2\pi kn/N) + \sin(2\pi kn/N)$ for $k, n = 0, 1, 2, \dots, N-1$.

Supposing N to be an even number, we can derive two sub-sequences $\mathbf{x}_1$ and $\mathbf{x}_2$ of length $N/2$ each, from $\mathbf{x}$ such that $\mathbf{x}_1 = \{x(0), x(2) \dots x(N - 2)\}$ contains all the even-indexed values, and $\mathbf{x}_2 = \{x(1), x(3) \dots x(N - 1)\}$, contains all the odd-indexed values of $\mathbf{x}$. The DHT defined by (1) then can be expressed by two summation terms as given in the following [26]:

$$X(k) = \sum_{n=0}^{(\frac{N}{2})-1} x_1(n) \cos\left(\frac{2\pi k(2n)}{N}\right) + \sum_{n=0}^{(\frac{N}{2})-1} x_2(n) \cos\left(\frac{2\pi k(2n+1)}{N}\right). \quad (2)$$

Let $X_1(k)$ and $X_2(k)$, for $k = 0, 1, 2, \dots, (N/2) - 1$, represent the $(N/2)$ -point DHT coefficients of sequences x_1 and x_2 of length $(N/2)$, respectively. Using the symmetry properties of cosine and sine functions, the N -point DHT of (2) may then be expressed by the following set of equations:

$$\left. \begin{aligned} X(k) &= X_1(k) + E(k), \\ X(M+k) &= X_1(k) - E(k), \\ \text{and } E(k) &= X_2(k) \cos\left(\frac{\pi k}{M}\right) + X_2(M-k) \sin\left(\frac{\pi k}{M}\right) \end{aligned} \right\},$$

for $k = 1, 2, \dots, M-1$, where $M = \frac{N}{2}$ (3)

$$X(0) = X_1(0) + X_2(0) \quad (4a)$$

$$X(M) = X_1(0) - X_2(0). \quad (4b)$$

Equations (3) and (4), can be used to derive the scalable and pipelined hardware for computing the DHT of length N using two modules of $(N/2)$ -length DHT, along with $(N/4 - 1)$ ROMs, equal number of multiplexers and demultiplexers pairs, and $(3N - 4)/2$ parallel adders. The pipelined implementation of 8-point DHT, using two modules of 4-point DHT, is discussed in Section IV, which can be extended further to implement the transform for higher values of N .

B. Algorithm for Computation of DHT of Other Composite Transform Lengths

For transform length $N = N_1 \times N_2$, where N_1 and N_2 are relatively prime numbers, as shown in [13], the indexes k and n in (1) may, respectively, be mapped into pairs of indexes (k_1, k_2) and (n_1, n_2) , according to the index mapping scheme of [29] and the DHT of (1) may be expressed as

$$X(k_1, k_2) = \sum_{n_2=0}^{N_2-1} w(k_1, n_2) \cos\left(\frac{2\pi k_2 n_2}{N_2}\right) \quad (5)$$

where

$$w(k_1, n_2) = \sum_{n_1=0}^{N_1-1} \left[x(n_1, n_2) \cos\left(\frac{2\pi k_1 n_1}{N_1}\right) + x(n_1, (N_2 - n_2)_{N_2}) \sin\left(\frac{2\pi k_1 n_1}{N_1}\right) \right]. \quad (6)$$

For $k_1, n_1 = 0, 1, \dots, N_1 - 1$ and $k_2, n_2 = 0, 1, \dots, N_2 - 1$. The symbol, $(.)_r$ in (6) and in the rest of the paper denotes “ $(.)$ modulo r ” where r is a positive integer. Using the properties of sine and cosine functions, (6) can be further expressed as

$$w(k_1, n_2) = \sum_{n_1=0}^{N_1-1} y(n_1, n_2) \cdot \cos\left(\frac{2\pi k_1 n_1}{N_1}\right) \quad (7)$$

where

$$y(n_1, n_2) = [x(n_1, n_2) + x((N_1 - n_1)_{N_1}, n_2) + x(n_1, (N_2 - n_2)_{N_2}) - x((N_1 - n_1)_{N_1}, (N_2 - n_2)_{N_2})]. \quad (8)$$

We can form a matrix $\mathbf{Y}$ of size $(N_1 \times N_2)$ from the 2-D sequence $\{y(n_1, n_2)$, for $n_1 = 0, 1, 2, \dots, N_1 - 1$ and $n_2 = 0, 1, 2, \dots, N_2 - 1\}$, and we may then represent (5) and (7) in a matrix form as

$$\mathbf{X} = \mathbf{H}_2 [\mathbf{H}_1 \mathbf{Y}]^T \quad (9)$$

where the elements of matrix $\mathbf{H}_i$ (for $i = 1$ and 2) are given by

$$H_i(k_i, n_i) = \cos\left(\frac{2\pi k_i n_i}{N_i}\right), \quad \text{for } k_i, n_i = 0, 1, \dots, N_i - 1. \quad (10)$$

Using (9), one may compute the DHT, conventionally, in the following three stages:

- 1) compute $\mathbf{W} = \mathbf{H}_1 \mathbf{Y}$;
- 2) transpose $\mathbf{W}$;
- 3) compute $\mathbf{X} = \mathbf{H}_2 \mathbf{W}^T$.

To save hardware, as well as, time involved in the transposition of intermediate matrix in the VLSI implementation of the prime-factor DHT, we suggest here an alternative algorithm as follows.

- 1) Initialize $k_1 = 0$
- 2) Compute the $(k_1 + 1)$ th row of the intermediate matrix $\mathbf{W}$, i.e., $w(k_1, n_2)$ for $n_2 = 0, 1, 2, \dots, N_2 - 1$, where $w(k_1, n_2)$ is the $(k_1 + 1)$ th component of the N_1 -point DHT of the $(n_2 + 1)$ th column of $\mathbf{Y} = [y(n_1, n_2)]$.
- 3) Compute the N_2 -point DHT of the $(k_1 + 1)$ th row of the intermediate matrix $[w(k_1, n_2)]$.
- 4) Increment k_1 by one.
- 5) If k_1 is less than N_1 then go to Step-2, else stop.

To implement the prime-factor DHT by this approach, the architecture should be designed to generate the intermediate matrix row-by-row in the first stage, and then it should use them as input for the second stage.

III. SCALABLE IMPLEMENTATION OF SHORT-LENGTH DHTs

In the followings, we suggest a hardware/throughput-scalable architecture for realization of prime-length DHT. Besides, we have derived fully pipelined and scalable structures for 4-point DHT in this Section.

A. Computation of Prime-Length DHT via Circular Convolution

To convert the DHT into a circular convolution let us write (1) as

$$X(0) = \sum_{n=0}^{N-1} x(n) \quad (11a)$$

and

$$X(k) = x(0) + T(k) \quad (11b)$$

where

$$T(k) = \sum_{n=1}^{N-1} x(n)h_N(k, n), \quad \text{for } k = 1, 2, \dots, N-1. \quad (12)$$

When N is a prime number, as shown in [13], the indexes k and n in (12) are mapped into l and m , respectively, according to the index mapping of [30], to obtain a circular convolution form

$$T(\alpha^l) = \sum_{m=0}^{N-2} x((\alpha^{-m})_N) Q_N((\alpha^{l-m})_N) \quad (13a)$$

where

$$Q_N(t) = \cos\left(\frac{2\pi t}{N}\right), \quad (13b)$$

$$\text{for } t = (\alpha^{l-m})_N \text{ and } l, m = 0, 1, \dots, N-2. \quad (14)$$

$$k = (\alpha^l)_N \text{ and } n = (\alpha^{-m})_N$$

where $m, l = 0, 1, \dots, N-2$ and α is the $(N-1)$ th primitive root of unity, such that

$$(\alpha^{N-1})_N = 1 \text{ and } (\alpha^j)_N \neq 1, \quad \text{for } 0 < j < (N-1). \quad (15)$$

An example of such mapping of the indexes k and n into l and m , respectively, for $N = 5$ is illustrated in the following.

- Find the value of the fourth primitive root of unity to be “2” that satisfies the

conditions $(2^4)_5 = 1$ and $(2^j)_5 \neq 1$ for $0 < j < 4$ as given by (15).

- Map the index k to l and the index n to m in (12) according to the relations

$$n = (2^{-m})_5 \text{ and } k = (2^l)_5, \quad \text{for } m, l = 0, 1, 2 \text{ and } 3. \quad (16)$$

- Substituting (16) into (12), one can have

$$T((2^l)) = \sum_{n=1}^{N-1} x((2^{-m})_5) h((2^{l-m})_5),$$

for $m = 0, 1, 2 \text{ and } 3. \quad (17)$

The mapping of indexes k and n to l and m , respectively, is derivable from (16), and is given in Table I (A), while the values of $(kn)_5$ for different values of k and n are given in Table I(B).

- Using the values from Table I we can write the circular convolution of (17) as

$$\begin{bmatrix} T(1) \\ T(2) \\ T(4) \\ T(3) \end{bmatrix} = \begin{bmatrix} h(1) & h(3) & h(4) & h(2) \\ h(2) & h(1) & h(3) & h(4) \\ h(4) & h(2) & h(1) & h(3) \\ h(3) & h(4) & h(2) & h(1) \end{bmatrix} \begin{bmatrix} x(1) \\ x(3) \\ x(4) \\ x(2) \end{bmatrix}. \quad (18)$$

The circular convolution of (18), can be computed by a fully pipelined multiplier-based structure, as shown in Fig. 1, for high-throughput computation of the 5-point DHT given by (11)–(13) and (18). To perform the circular convolution of (18), we have used 16 multiplication cells (MC) and 16 adder cells (AC1 and AC2) arranged in four rows and four columns. Function of the multiplication cells is depicted in Fig. 1(b). Each of these cells performs a multiplication in every cycle time with a fixed multiplier. The multiplications in these cells may be implemented by a multiplier or by an adder-based structure or by CORDIC circuit. One may also implement each of these multiplications by a ROM table that stores 2^L possible product values corresponding one of the four coefficients $h(1)$, $h(3)$, $h(4)$, and $h(2)$, where L is the word length. The input words can be fed to the ROMs as addresses, and the product values thus read from the individual ROMs can be passed across an array of 3-pipelined adder cells (AC1). To fulfill the necessary data dependency requirement, the inputs to the third and the fourth multiplication cells are, respectively, staggered by duration Δ and 2Δ , where Δ denotes a cycle period. An accumulator and an adder cell (AC2) are used to compute $X(0)$ using (11a). The additions of (11b) are performed by four AC2 placed at the end of each of the four pipelined rows of AC1. The functions of AC1 and AC2 are depicted in Fig. 1(c) and (d), respectively.

B. Direct-ROM-Based Scalable Implementation

In this subsection, we derive a single-array structure using ROM-based multiplier for

computation of prime-length DHT. Besides, we show that one can use more than one array if hardware is not a constraint and/or high-throughput is required. To arrive at such scalable implementation of the DHT (using the commutative properties of circular convolution) we express (18) alternatively as

$$\begin{bmatrix} T(1) \\ T(2) \\ T(4) \\ T(3) \end{bmatrix} = \begin{bmatrix} x(1) & x(3) & x(4) & x(2) \\ x(3) & x(4) & x(2) & x(1) \\ x(4) & x(2) & x(1) & x(3) \\ x(2) & x(1) & x(3) & x(4) \end{bmatrix} \begin{bmatrix} h(1) \\ h(3) \\ h(4) \\ h(2) \end{bmatrix}. \quad (19)$$

The computation of (19) is then represented by a dependency graph (DG) in Fig. 2. Function of each node of the DG is depicted in Fig. 2(b). The DG in Fig. 2(a) can be projected vertically to a single array of computing nodes as given by Fig. 2(c), where the input samples are circularly left-shifted in every cycle. Using the circular convolution of (19), the 5-point DHT given by (11)–(13) can hence be implemented by a fully pipelined array of 4 multiplication cells and 4 adder cells as shown in Fig. 3.

The function of the multiplication cells and adder cells in Fig. 3 are the same as those for Fig. 1. For a direct-ROM based implementation, the input words are fed to the multiplication cells as addresses through a circular-left-shift buffer. Here also an accumulator and an AC2 are used to calculate the value of $X(0)$ according to (11a). 3-pipelined adder cells (AC1) are used to sum the output product values, and another AC1 is used for the addition of (11b). After the pipeline is filled in the first five cycles, $X(k)$, for $k = 1, 2, 3$, and 4 are obtained from the adder cells in successive cycles as shown in Fig. 3. In order to have a double-array structure, we can break the circular convolution of (19) in to two parts as follows:

$$\begin{aligned} \begin{bmatrix} T(1) \\ T(2) \end{bmatrix} &= \begin{bmatrix} x(1) & x(3) & x(4) & x(2) \\ x(3) & x(4) & x(2) & x(1) \end{bmatrix} \\ &\quad \times [h(1) \ h(3) \ h(4) \ h(2)]^T \quad (20a) \\ \text{and } \begin{bmatrix} T(4) \\ T(3) \end{bmatrix} &= \begin{bmatrix} x(1) & x(3) & x(4) & x(2) \\ x(3) & x(4) & x(2) & x(1) \end{bmatrix} \\ &\quad \times [h(4) \ h(2) \ h(1) \ h(3)]^T. \quad (20b) \end{aligned}$$

We may now partition the DG of Fig. 2 horizontally into two halves, to represent the computation of (20). The sub-graphs, thus obtained, may be projected vertically into two horizontal linear arrays, where each of the linear arrays consists of 4 computing nodes. The single-array structure of Fig. 3 has an average computation time (ACT) of four cycles, while the double-array structure will have ACT of two cycles. In general, for computing the DHT of transform length N , one can have a structure with P number of linear arrays to achieve the ACT of (N/P) cycles.

C. DA-Based Scalable Implementation

In the following, we develop the proposed scalable array architecture for computing the DHT of prime transform lengths using distributed arithmetic. It may be noted here that for simplicity of discussion we have assumed the signal samples to be unsigned words of size L , though the structures can also be used for 2's complement coding and offset binary coding by simple modifications. Let $\mathbf{c}$ and $\mathbf{y}$, respectively, be the K -point coefficient vector and K -point signal vector. The inner product of $\mathbf{c}$ and $\mathbf{y}$ can be given by

$$P = \sum_{l=1}^L P_l 2^{-l} \quad (21)$$

where

$$P_l = \sum_{k=1}^K c(k) \cdot b_{lk} \quad (22a)$$

and

$$y(k) = \sum_{l=1}^L b_{lk} 2^{-l}. \quad (22b)$$

The 2^K possible values of P_l corresponding to the 2^K permutations of bit sequence $\{b_{lk}$, for $k = 1, 2, \dots, K\}$ are stored in a ROM of 2^K words, which can be read out when the bit sequence is fed to the ROM as address. (Using offset binary coding the ROM size can be reduced to $2^{(K-1)}$ words [28], [31].) We may, thus, rewrite (21) as

$$P = \sum_{l=1}^L F(s_l) \cdot 2^{-l} \quad (23)$$

where $P_l = F(s_l)$ for $l = 0, 1, 2, \dots, L$, and $s_l = \{b_{lk}$, for $k = 1, 2, 3, \dots, K\}$ is the “address word” for the ROM look up table. The inner products of (19) can, therefore, be computed by repeated ROM read operations followed by shift-and-accumulation as given by (23). Further, if we make the following simple substitutions on (19):

$$\begin{aligned} Y(1) &\leftarrow T(1), Y(2) \leftarrow T(2), Y(3) \leftarrow T(4), Y(4) \leftarrow T(3) \\ y(1) &\leftarrow x(1), y(2) \leftarrow x(3), y(3) \leftarrow x(4), y(4) \leftarrow x(2) \\ c(1) &\leftarrow h(1), c(2) \leftarrow h(3), c(3) \leftarrow h(4) \text{ and } c(4) \leftarrow h(2) \end{aligned} \quad (24)$$

we can have

$$\begin{bmatrix} Y(1) \\ Y(2) \\ Y(3) \\ Y(4) \end{bmatrix} = \begin{bmatrix} y(1) & y(2) & y(3) & y(4) \\ y(2) & y(3) & y(4) & y(1) \\ y(3) & y(4) & y(1) & y(2) \\ y(4) & y(1) & y(2) & y(3) \end{bmatrix} \begin{bmatrix} c(1) \\ c(2) \\ c(3) \\ c(4) \end{bmatrix}. \quad (25)$$

Using (23), we can write (25) as

$$Y(i) = \sum_{l=1}^L F \left(R^{(i-1)}(s_l) \right) 2^{-l}, \quad \text{for } i=1, 2, 3 \text{ and } 4 \quad (26)$$

where the operator $R^{(l)}$ rotates the word s_l circularly left through l number of bit positions. The DG for computation of (26) is shown in Fig. 4, where the function of the nodes is depicted in Fig. 4(b).

The DG can be projected vertically to derive a linear array consisting of L nodes [shown in Fig. 4(c)] to compute the convolution of (25). The resulting architecture for computing the DHT, using the DA-based convolution structure, is shown in Fig. 5. For reducing the ROM size to 2^{N-2} words for computing $(N-1)$ -point convolution here, we can use offset binary coding in the proposed structure by feeding the pre-computed constant term [28], [31] to the first PE from the left without modification inside the array structure. The structure then consists of L PEs, each consisting of a ROM of eight words. The input values $\{x(1), x(3), x(4), x(2)\}$ are fed to a bit-serial-word-parallel converter to generate the ROM addresses $\{s_1, s_2, \dots, s_L\}$ to be used as input for the L PEs in a staggered manner. The function of the PEs is shown in Fig. 5(b). Each PE here performs one ROM read operation, and one shift operation followed by one addition. The cycle time here may be considered as $T = T_{\text{ROM}} + T_{\text{SHIFT}} + T_{\text{ADD}}$. If the ROM access and shift operations for one pair of input values are performed simultaneously with the addition operation corresponding to its preceding pair of input values, then the computational cycle can be reduced to $T = T_{\text{ADD}}$. The single-array structure of Fig. 5 has latency of $(L+1)$ cycles and an ACT of four cycles. To achieve higher throughput over the single-array structure we may also partition the DG for computation of the circular convolution, horizontally into two halves, and the sub-graphs thus obtained, may be projected vertically to obtain two arrays. The number of ROM tables and the number of adders in the double-array structure will be double that of the single-array structure, but it will have ACT of two cycles.

For high throughput implementation, from the DG of Fig. 4, one can derive a fully pipelined structure with 4 linear arrays of L PEs each, as shown in Fig. 6. The 4-array structure of Fig. 6 has latency of $(L+1)$ cycles and an ACT of one cycle only.

D. Scalable Implementation of 4-Point DHT

It is interesting to observe that the DHT kernel for $N=4$ given by (27) in the following involves only 1 s and -1 s

$$\mathbf{C} = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & 1 & -1 \\ 1 & -1 & -1 & 1 \end{bmatrix}. \quad (27)$$

This feature of the DHT is used to design simple adder-based modules for computing the 4-point DHT shown in Figs. 7 and 8 in the following.

The single-array structure of Fig. 7(a) consists of 3 PEs, where each PE either performs an addition (for Tag – bit = 1) or a subtraction (for Tag – bit = 0). The Tag-bits may be fed to the individual PEs through 4-bit shift-registers (not shown in the figure). Each of the PEs is fed with the input samples in a staggered manner to maintain the necessary data dependencies. It will yield the first output after 3-cycle time, and thereafter it will produce a throughput of 1 output in every cycle (1 cycle time = T_{ADD}). The structure of Fig. 7(b) consists of 12 adder/subtractor cells, arranged in four rows and three columns, which can yield 4 DHT components during every cycle. Using the kernel values of (27) in (1), we can have a two-stage algorithm for computation of 4-point DHT as follows.

Step 1) Compute the following in parallel:

$$\begin{aligned} b1 &= x(0) + x(2) \\ b2 &= x(1) + x(3) \\ b3 &= x(0) - x(2) \\ b4 &= x(1) - x(3). \end{aligned} \quad (28a)$$

Step 2) Compute the following in parallel:

$$\begin{aligned} X(0) &= b1 + b2 \\ X(1) &= b3 + b4 \\ X(2) &= b1 - b2 \\ X(3) &= b3 - b4. \end{aligned} \quad (28b)$$

The two stages of computation of 4-point DHT given by (28) can be performed by two pipelined stages of adder units “ADDER-1” and “ADDER-2” as shown in Fig. 8, where each of the adder units consists of four adders to perform four additions in parallel. Functions of these adder units are depicted in Fig. 8(b) and (c). The structure has latency of two cycles, and it can yield a throughput of four DHT components in every cycle.

IV. IMPLEMENTATION OF COMPOSITE-LENGTH DHT

In this section, we describe the proposed structures for the DHT of power-of-two transform lengths and for transform length $N = N_1 \times N_2$, where N_1 and N_2 are

relatively prime.

A. Implementation of the DHT of Even and Power-of-2 Transform Lengths

In the following, we illustrate the derivation of the structure for computing the DHT for $N = 8$ as an example. For $N = 8$, we can substitute $M = 4$ in (3) and (4) to obtain the following set of equations:

$$\begin{aligned}
X(0) &= X_1(0) + X_2(0) \\
X(1) &= X_1(1) + \left(\frac{X_2(1)}{\sqrt{2}} + \frac{X_2(3)}{\sqrt{2}} \right) \\
X(2) &= X_1(2) + X_2(2) \\
X(3) &= X_1(3) + \left(\frac{X_2(1)}{\sqrt{2}} - \frac{X_2(3)}{\sqrt{2}} \right) \\
X(4) &= X_1(0) - X_2(0) \\
X(5) &= X_1(1) - \left(\frac{X_2(1)}{\sqrt{2}} + \frac{X_2(3)}{\sqrt{2}} \right) \\
X(6) &= X_1(2) - X_2(2) \\
X(7) &= X_1(3) - \left(\frac{X_2(1)}{\sqrt{2}} - \frac{X_2(3)}{\sqrt{2}} \right). \quad (29)
\end{aligned}$$

For computation of 8-point DHT according to (29), the multiplications of $x_2(1)$ and $x_2(3)$ by $(1/\sqrt{2})$ can be read out from a ROM, while a block of pipelined adders may perform the additions. The proposed structure for computation of 8-point DHT is shown in Fig. 9.

It computes the 8-point DHT in five pipelined stages. For computation of the first two stages, it consists of two 4-point DHT modules that receive the odd and even indexed subsequences $\mathbf{x}_1$ and $\mathbf{x}_2$ from the input buffer. In the third pipeline stage, the DHT components $X_2(1)$ and $X_2(3)$ corresponding to the odd-indexed subsequence $\mathbf{x}_2$ are multiplexed in two consecutive half-cycles and passed as addresses to the ROM that contains 2^L product values for multiplication by $(1/\sqrt{2})$. The two product terms are then read from the ROM and demultiplexed at the end of two corresponding consecutive half-cycles of the third pipelined stage. In the fourth pipeline stage, the demultiplexed outputs are fed to an add/subtract unit having one adder and one subtractor to perform the addition/subtraction operations corresponding to the terms within the brackets of (29). The remaining two DHT components $X_2(0)$ and $X_2(2)$ of the odd-indexed subsequence $\mathbf{x}_2$ along with DHT components of the even-indexed subsequence $\mathbf{x}_1$ are fed to unit delay cells in the pipeline stages 3 and 4. Finally, in the pipeline Stage 5, a block of eight adders is used to perform the remaining 8 additions of (29) in parallel to produce the desired 8-point DHT. It may be noted here that the pipeline stages-1 and -2 of the proposed structure can be obtained from Stage 2 and Stage 3 of the flow-graph of [2] (at page 311), but stages 3 to 5 of the proposed structure are not only different from the structure corresponding to flow-graph of [2] but also offer significant advantage leading to reduced-hardware pipelined implementation. A 16 -point DHT module may similarly be

derived from two 8-point DHT modules, and structures for higher power-of-2 transform lengths can be constructed, as well. This technique may also be used for even-length DHT (for $N = 2M$) using the optimal prime-length modules for $M = 3, 5, 7, 11$ etc. (discussed in Section III).

B. Implementation of the DHT of Composite Transform Length ($N = N_1 \times N_2$)

There can be several possible implementations of the prime-factor DHT for $N = (N_1 \times N_2)$ using the N_1 -point DHT modules in Stage 1, and N_2 -point DHT modules in Stage 2. One may have option for reduced-hardware implementation, high-throughput implementation, as well as, other scalable implementations. In the reduced-hardware implementation only a single-array for N_1 -point DHT may be used in Stage 1, and for computation of Stage 2, a single-array for N_2 -point DHT may be used. The intermediate result in this case would be required to be stored in a RAM of N words, before being used by Stage 2. For a scalable implementation, either a multi-array structure or multiple numbers of single-array structures for the N_1 - and N_2 -point DHTs may be used in both the stages. We have discussed here two representative examples of high-throughput implementation. In *Case 1*), we have taken N_2 to be a power-of-2 transform length, e.g., $N_2 = 4$ or 8 etc, while N_1 is a prime number. In *Case 2*), we have taken N_1 and N_2 to be two unequal prime numbers.

Case 1) The structure for $N_1 = 5$ and $N_2 = 4$ is shown in Fig. 10, where the DHT for $N = 5 \times 4$ is computed in two distinct stages. Stage 1 of the structure consists of 4 modules of single-array for computation of 5-point DHT. Stage 2 of the structure uses a 4-point DHT module consisting of pipelined adder units (shown in Fig. 8). The input matrix $\mathbf{X}$ is fed to an input preprocessor comprised of four parallel adders to perform the additions of (8) so as to generate the elements of matrix $\mathbf{Y}$. The elements of all the four columns of $\mathbf{Y}$ are fed in parallel to the four modules of single-array structure for 5-point DHT (Stage 1). It is interesting to note that the output of Stage 1 is fed directly to the Stage 2 of the structure for computing the desired DHT components (without using any transposition unit). In case of the proposed multiplier-based implementation, the first row of intermediate matrix is obtained in five cycles ($L + 1$ cycles in case of DA-based structure) and the output of the second stage of computation becomes available after two more cycles. The first set of four transform components is thus computed in seven cycles ($L + 3$ cycles in case of DA-based structure); while in every subsequent cycle the structure yields four DHT components. The complete set of DHT components, therefore, can be obtained in every 5-cycle time.

Case 2) The two-stage DHT-structure for $N_1 = 5$ and $N_2 = 3$ is shown in Fig. 11. Stage 1 of the structure involves three linear arrays for computing the 5-point DHT. Stage 2 of the structure uses a 2-array module for computing the 3-point DHT. The input matrix $\mathbf{X}$ is fed to an input preprocessor comprised of four parallel adders to perform the additions of (10b) so as to generate the elements of matrix $\mathbf{Y}$. Three columns of the matrix $\mathbf{Y}$ are fed in parallel to the three

single-arrays for 5-point DHT (Stage 1).

As in the case of the structure for *Case 1*), here also the rows of the intermediate matrix (i.e., the output of Stage 1) are fed directly to Stage 2 of the structure without using any transposition network. In case of multiplier-based implementation, the first set of output of the intermediate matrix is obtained five cycles after the input arrives at Stage 1 (or after $(L + 1)$ cycles in case of DA-based implementation). The latency of Stage 2 is three cycles for the multiplier-based implementation or $(L + 1)$ cycles in case of DA-based implementation.

C. Implementation of 2-D DHT

The 2-D DHT of an input matrix $[x(m,n)]$ of size $N \times N$ is defined as [32]

$$X(k,l) = \sum_{m=0}^{N-1} \sum_{n=0}^{N-1} x(m,n) \cdot \left[\cos 2\pi \left(\frac{km}{N} + \frac{ln}{N} \right) + \sin 2\pi \left(\frac{km}{N} + \frac{ln}{N} \right) \right] \quad (30)$$

For $k, l, m, n = 0, 1, 2, \dots, N - 1$. Following the same procedure as that of the prime-factor DHT (discussed in Section II-B) one can write (30) as

$$\mathbf{X} = \mathbf{C}[\mathbf{C}\mathbf{Y}]^T. \quad (31a)$$

where the elements of matrices $\mathbf{C}$ and $\mathbf{Y}$ are, respectively, given by

$$C_N(i,j) = \left[\cos \left(\frac{2\pi ij}{N} \right) + \sin \left(\frac{2\pi ij}{N} \right) \right], \quad \text{for } i, j = 0, 1, \dots, N-1, \text{ and} \quad (31b)$$

$$y(m,n) = [x(m,n) + x((N-m)_N, n) + x(m, (N-n)_N) - x((N-m)_N, (N-n)_N)], \quad \text{for } m, n = 0, 1, 2, \dots, N-1. \quad (31c)$$

The computation of 2-D DHT of (31) is similar to that of the prime-factor DHT and, therefore, the 2-D DHT can be implemented by a structure similar to that of prime-factor DHT of *CASE 2*) discussed in the Section IV-B. For computing the 2-D DHT of size (5×5) we can replace the Stage 2 of the structure of Fig. 11 by a 4-array structure for computation of 5-point high-throughput DHT (as shown in Fig. 1 for multiplier-based structure or Fig. 6 for DA-based structure).

V. HARDWARE AND THROUGHPUT CONSIDERATIONS

In this section, we discuss the details of hardware requirement and throughput of the proposed structures, and compare them with those of the corresponding existing structures.

A. Scalable Prime-Length DHTs

In Section III, we have discussed both DA-based, as well as, multiplier-based implementations of prime-length DHTs. The hardware requirements, throughput and the ACT of the proposed scalable implementations of the prime-length DHT are listed in Table II. The multiplier-based single-array structure for N -point DHT (Fig. 3) has ACT of $(N - 1)$ cycles and requires $(N - 1)$ multipliers and N adders. In the multiplier-based implementation, one may wish to perform the necessary multiplications (with the fixed coefficients of the DHT kernel) by CORDIC circuits, or adder-based structures or hardwired multipliers. ROM-based look up tables can also be used to implement the multiplications where the structure will involve $(N - 1)$ ROM of 2^L words. The ROM-based structure of [22] has nearly the same ACT as the proposed direct-ROM-based single-array structure, but it involves more than double the number of adders and significantly more hardware than the proposed structure due to its preprocessing and post-processing requirements. Using the proposed double-array structure, the ACT can be reduced to $(N - 1)/2$ cycles, and using $(N - 1)$ such arrays (as in case of the structure in Fig. 1) the ACT can be reduced to one cycle. For DA-based implementation, the proposed single-array structure (Fig. 5) for computing N -point DHT involves L PEs where each PE consists of a ROM of 2^{N-2} words and an adder. It delivers one output in every cycle once the pipeline is filled during the first $(L + 1)$ cycles. The ACT of this structure is $(N - 1)$ cycles. As shown in Table II, the existing DA-based structures of [23]–[25] require more than double the normalized ROM (number of words to be stored in the ROM per unit throughput) compared with the proposed structure. For reduced-hardware realization one can use a single-array structure.

To increase the throughput at the same clock speed, more than one array can be used. Maximum throughput of N transform components per cycle can be obtained by using $(N - 1)$ number of arrays. In general, when P such arrays are used, the number of ROMs, as well as, the adder circuits are increased by P times, but at the same time the ACT is reduced by P times.

B. DHT of Power-of-2 Points and Other Composite Transform Lengths

The hardware requirements, throughput and the ACT of the proposed structure for the DHT of power-of-two length are listed in Table II along with those of prime-length DHT structures and the radix-2 DHT structure of [33]. It is observed that the proposed structure for power-of-two length DHT requires nearly $(N/16)$ times more ROM space and $(N/2)$ times more number adders compared with the equivalent memory-based implementation of the radix-2 structure of [33], but it offers $(N/2)$ times more throughput compared with the other (where each complex multiplication is implemented by four real multiplications and two real additions).

For reduced-hardware implementation, the prime-factor DHT for $N = N_1 \times N_2$ requires a

single-array structure for N_1 -point DHT and a single-array structure for N_2 -point DHT. The direct-ROM-based structure, thus, requires ROM for storing $(N_1 + N_2 - 2) \cdot 2^L$ words and $(N_1 + N_2 + 2)$ number of adders, while the DA-based structure will involve ROM of $L \cdot (2^{N_1-2} + 2^{N_2-2})$ words and $2(L + 2)$ adders. The computational complexities of the prime-factor DHT using single-array and double-array structures are listed in Table III. If we consider $(N_1 \approx N_2 \approx \sqrt{N})$, the proposed reduced-hardware prime-factor DHT structure in case of direct-ROM-based implementation, requires a ROM of $O(N^{1/2} \cdot 2^L)$ words and $O(N^{1/2})$ number of adders, where as, the DA-based implementation requires $O(L2^{N^{1/2}})$ ROM space and $O(L)$ number of adders. The proposed reduced-hardware structures, thus, require only a small fraction (nearly $N^{-1/2}$ times in case of ROM-based structure and $2^{-N^{1/2}}$ times in case of DA-based structures) of hardware compared to the existing structures of [22], [23], but yield nearly the same throughput as that of the others. For increasing the throughput further, one may have scaling options in the proposed designs to use more number of arrays in each of the stages. All these reduced-hardware and scalable implementation will require a RAM for storing N words of the intermediate results.

The normalized ROM size of reduced-hardwired ROM-based prime-factor structure and the structure for power-of-2 transform lengths are plotted in Fig. 12 along with that of the existing structure [22]. It is observed that both the proposed structures involve considerably less ROM compared with the existing ROM-based structure. Also, it is observed that the structure for power-of-two transform lengths involves ROM of minimum size. Not only it offers N times more throughput compared with the ROM-based structure of [22] but also involves considerably less hardware than the other for $N < 128$. These structures would, therefore, be useful for high-speed applications. Moreover, when high throughput is not desired, the clock speed in structure can be reduced by N times for low-power realization. The ROM size of reduced-hardwired DA-based prime-factor structure for various transform lengths and that of existing structures of [23]–[25] for $L = 32$ in logarithmic scale of base 10 are plotted in Fig. 13. It is observed from the figure that the ROM requirement of the proposed DA-based structure is less than that of the existing structures [23]–[25] by several orders of magnitude. Also it can be observed that the ROM used in the structure of [24], [25] is comparable to that of [23], but the structure of [24], [25] involves less ROM than that of [23] for $N < L$.

In the high throughput structures, as the one discussed in *CASE 1*), N_2 is taken to be 4, 8 or other power-of-2 numbers. So the Stage 1 of the structure, in general, involves N_2 -number of single-array structure for N_1 -point DHT followed by a pipelined block structure for N_2 -point DHT of Stage 2. In *CASE 2*), on the other hand, N_2 may be a prime number other than N_1 . The structure of *CASE 2*) thus involves N_2 -number of single-array structures for N_1 -point DHT followed by an $(N_2 - 1)$ -array structure for N_2 -point DHT. The details of hardware requirement and throughput/ latency of the proposed *CASE 1*) and *CASE 2*) for ROM-based and DA-based implementations are listed in Table IV.

The AT^2 (Area – Time²) VLSI complexity measure of the proposed ROM-based prime-factor structures for *CASE 1*) and *CASE 2*) along with that of the structure of [22] in

logarithmic scale of base 10 are plotted in Fig. 14. The proposed structure for *CASE 1*) (for $N_2 = 8$) offers less than $(1/64)$ times AT^2 of the existing structure [22]. It is also observed that the ROM-based prime-factor structure of *CASE 2*) has better cost-performance than *CASE 1*) for N higher than 100. Similar plot for the proposed high-throughput DA-based prime-factor structure and the structures of [23]–[25] for $L = 32$ are given in Fig. 15. It can easily be seen from Fig. 15, that AT^2 complexities of the proposed DA-based structures for both *CASE 1*) and *CASE 2*) are less compared with the existing structures of [23]–[25] by several orders of magnitude. For large transform length (more than 90), the structure of *CASE 2*) has significantly lower AT^2 cost compared with that of *CASE 1*).

C. Comparison of DA-Based Structure and Conventional-Multiplier Based Structure

In the multiplier-based implementation, one may also opt to use hardwired multipliers for the multiplication with the fixed DHT kernel coefficients. The hardware complexities of the multipliers, however, vary widely as one can implement a multiplier by one adder and one shifter for very-low-hardware implementation, and one may also like to use several adders and shifters for fast multiplication. The hardware and time complexities of the DHT structure of hardwired multiplier-based structure will vary according to the implementation of the multipliers. If we use parallel carry-ripple array multiplier [28] which involves L^2 full-adders, $3L^2$ D-flip-flops, and $3L^2$ AND gates; the reduced-hardware multiplier-based prime-factor DHT structure will involve $[33L^2(N + N_2^2 - 3N_2 + 1).2^L]$ NAND gates for its multipliers. Besides, it will require $[9L(N + N_2^2 - N_2 + 7)]$ NAND gates for adders. (We have taken the gate counts of each 1-bit full-adder, D flip-flop, and AND gate, respectively, to be 9, 6, and 2 NAND gates.) Assuming 1-bit memory to cost 4 NAND gates, we find that the gate count of the proposed reduced-hardware DA-based prime-factor structure amounts to $4 * L^2(N_2.2^{(N_1-2)} + (N_2 - 1).2^{(N_2-2)})$ NAND gates. The AT^2 cost measure of the structure using hardwired-multiplier and the DA-based structure for $L = 32$ is accordingly evaluated for various transform lengths and are plotted as shown in Fig. 16. When the transform length N is less than nearly 1000, DA-based implementation has significantly lower AT^2 cost compared to the structure based on conventional arithmetic. However, for higher transform lengths and lower values of word length L the multiplier-based structure using conventional arithmetic has better performance than the DA-based structure.

VI. CONCLUSION

We have presented a design framework for scalable and modular memory-based implementation of the DHT in systolic hardware. The proposed structures for prime-length DHT have nearly the same time-complexity as those of the corresponding existing structures of [22]–[25], but involve significantly less hardware than the others. Besides, unlike the existing memory-based DHT architectures, the proposed structures do not involve control-tags, data-broadcast and can be conveniently scalable to meet the area-delay-power tradeoff. We have also proposed an algorithm for computing N -point DHT recursively from two $(N/2)$ -point DHTs, which is used to obtain an area-time efficient fully-pipelined structures for even or power-of-2 transform lengths. The

proposed structures for prime-length and power-of-2 lengths are used further to develop modular and scalable prime-factor structures. It is interesting to note that the proposed prime-factor structures do not involve any transposition unit which otherwise imposes a substantial overhead on hardware and time [16]. Moreover, these structures require only a small fraction (nearly $N^{-1/2}$ times in case of ROM-based structure and $2^{-N^{1/2}}$ times in case of DA-based structures for $N_1 \approx N_2$) of normalized ROM compared to the existing structures [22]–[25]. It is also found that the proposed structures for prime-factor DHT involve significantly less AT^2 cost over the existing structures by several orders of magnitude. Since the short-length modules are scalable for speed and hardware, the proposed prime-factor structures would provide flexible options for transform lengths and suitable tradeoff between hardware and time/power. The DA-based structure presented here can also be used for FPGA implementation. Since the memory requirement grows up with the transform-size, the maximum transform-size to be implemented in a given FPGA would of course depend on the memory available in the device [34]. A detailed study on FPGA implementation of the DHT can be taken up as a future work. It may also be noted that the look-up-tables in the proposed ROM-based structures can be replaced by CORDIC-circuits or hardwired multipliers whenever desired.

REFERENCES

- [1] R. N. Bracewell, "Discrete Hartley transform," *J. Opt. Soc. Amer.*, vol. 73, no. 12, pp. 1832–1835, 1983.
- [2] ———, *The Fourier Transform and its Applications*, 3rd ed. New York: McGraw-Hill, 2000.
- [3] L.-Z. Cheng, L. Tong, and Z.-R. Jiang, "Real transform algorithm for computing discrete circular deconvolution," in *Proc. 3rd IEEE Int. Conf. Signal Process.*, vol. 1, Oct. 1996, pp. 166–169.
- [4] G. S. Maharana and P. K. Meher, "Algorithm for efficient interpolation of real-valued signals using discrete Hartley transform," *Comp. Elect. Eng.*, vol. 23, no. 2, pp. 129–134, Mar. 1997.
- [5] K. H. Tzou and T. R. Hsing, "A study of discrete Hartley transform for image compression application," in *Proc. SPIE Int. Soc. Opt. Eng.*, vol. 534, Jan. 1985.
- [6] C. H. Paik and M. D. Fox, "Fast Hartley transform for image processing," *IEEE Trans. Med. Imag.*, vol. 7, no. 6, pp. 149–153, Jun. 1988.
- [7] J.-L. Wu and J. Shiu, "Discrete Hartley transform in error control coding," *IEEE Trans. Signal Process.*, vol. 39, no. 10, pp. 2356–2359, Oct. 1991.
- [8] P. K. Meher and G. Panda, "Unconstrained Hartley-domain least mean square adaptive filter," *IEEE Trans. Circuits Syst. II, Analog Digit. Signal Process.*, vol. 40, no. 9, pp. 582–585, Sep. 1993.
- [9] C.-L. Wang, C.-H. Chang, J. L. Fan, and J. M. Cioffi, "Discrete Hartley transform based multicarrier modulation," in *Proc. IEEE Int. Conf. Acoust., Speech, Signal Process. (ICASSP '00)*, vol. 5, Jun. 2000, pp. 2513–2516.
- [10] R. N. Bracewell, "Aspects of the Hartley transform," *Proc. IEEE*, vol. 82, no. 3, pp. 381–387, Mar. 1994.
- [11] C.-L. Wang and C.-H. Chang, "A DHT-based FFT/IFFT processor for VDSL transceivers," in *Proc. IEEE Int. Conf. Acoust., Speech, Signal Process. (ICASSP '01)*, vol. 2, May 2001, pp. 1213–1216.
- [12] H. S. Hou, "The fast Hartley transform algorithm," *IEEE Trans. Comput.*, vol. C-36, no. 2, pp. 147–156, Feb. 1987.
- [13] P. K. Meher, J. K. Satapathy, and G. Panda, "New high-speed prime-factor algorithm for discrete Hartley transform," *Proc. Inst. Elect. Eng. F, Radar Signal*

Process., vol. 140, no. 1, pp. 63–70, Feb. 1993.

- [14] A. M. Gregorian, “A novel algorithm for computing the 1-D discrete Hartley transform,” *IEEE Signal Process. Lett.*, vol. 11, no. 2, pp. 156–159, Feb. 2004.
- [15] J. I. Guo, “An efficient design for one-dimensional discrete Hartley transform using parallel additions,” *IEEE Trans. Signal Process.*, vol. 48, no. 10, pp. 2806–2813, Oct. 2000.
- [16] C. Chakrabarti and J. Jaja, “Systolic architectures for the computation of the discrete Hartley and discrete cosine transforms based on prime factor decomposition,” *IEEE Trans. Comp.*, vol. 39, no. 11, pp. 1359–1368, Nov. 1990.
- [17] S. S. Nayak and P. K. Meher, “High throughput VLSI implementation of discrete orthogonal transforms using bit-level vector-matrix multiplier,” *IEEE Trans. Circuits Syst. II, Analog Digit. Signal Process.*, vol. 46, no. 5, pp. 655–658, May 1999.
- [18] 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, Sep. 1991.
- [19] 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.
- [20] J. H. Hsiao, L. G. Chen, T. D. Chiueh, and C. T. Chen, “Novel systolic array design for the discrete Hartley transform with high throughput rate,” in *Proc. IEEE Int. Conf. Circuits Syst.*, 1993, pp. 1567–1570.
- [21] P. K. Meher, J. K. Satapathy, and G. Panda, “Efficient systolic solution for a new prime factor discrete Hartley transform algorithm,” *Proc. Inst. Elect. Eng. G, Circuits, Dev. Syst.*, vol. 140, no. 2, pp. 135–139, Apr. 1993.
- [22] J. I. Guo, C. M. Liu, and C. W. Jen, “A novel VLSI array design for the discrete Hartley transform using cyclic convolution,” in *Proc. IEEE Int. Conf. Acoust., Speech, Signal Process. (ICASSP '94)*, 1994, pp. II-501–II-504.
- [23] J. I. Guo, “A new distributed arithmetic algorithm for its hardware architecture for the discrete Hartley transform,” *Pattern Recogn. Image Anal.*, vol. 10, no. 3, pp. 368–378, 2000.
- [24] A. Amira and A. Bouridane, “An FPGA implementation of discrete Hartley transforms,” in *Proc. 7th Int. Symp. Signal Process. Appl.*, vol. 1, Jul. 2003, pp. 625–628.
- [25] A. Amira, “An FPGA-based parameterisable system for discrete Hartley transforms implementation,” in *Proc. Int. Conf. Image Process. (ICIP'03)*, vol. 2,

Sep. 2003, pp. 567–570.

- [26] P. K. Meher and T. Srikanthan, “A scalable and multiplier-less fully-pipelined architecture for VLSI implementation of discrete Hartley transform,” in *Proc. Int. Symp. Signals, Circuits Syst. (SCS’03)*, vol. 2, Jul. 10–11, 2003, pp. 393–396.
- [27] H. T. Kung, “Why systolic architectures?,” *Computer*, vol. 15, no. 1, pp. 37–46, Jan. 1982.
- [28] K. K. Parhi, *VLSI Digital Signal Processing Systems: Design and Implementation*. New York: Wiley-Interscience, 1999.
- [29] C. S. Burrus, “Index mappings for multidimensional formulation of the DFT and convolution,” *IEEE Trans. Acoust. Speech Signal Process.*, vol. ASSP-25, pp. 239–242, 1977.
- [30] C. M. Rader, “Discrete Fourier transforms when the number of data samples is prime,” *Proc. IEEE*, vol. 56, pp. 1107–1108, Jun. 1968.
- [31] S. A. White, “Applications of distributed arithmetic to digital signal processing: a tutorial review,” *ASSP Mag.*, vol. 6, no. 3, pp. 4–19, Jul. 1989.
- [32] R. N. Bracewell, O. Buneman, H. Hao, and J. Villasenor, “Fast two-dimensional Hartley transform,” *Proc. IEEE*, vol. 74, no. 9, pp. 1282–1283, Sep. 1986.
- [33] S.-F. Hsiao and W.-R. Shiue, “Low-cost unified architectures for the computation of discrete trigonometric transforms,” in *Proc. IEEE Int. Conf. Acoust., Speech, Signal Process. (ICASSP ’00)*, vol. 6, Jun. 2000, pp. 3299–3302.
- [34] <http://www.xilinx.com/> [Online]

LIST OF FIGURES

- Fig. 1 The 4-array pipelined structure for high-throughput computation of 5-point DHT using ROM-based multiplication cells. (a) The structure. (b) Function of the multiplier cells (MC). (c) Function of the adder cells AC1. (d) Function of the adder cells AC2. (Δ stands for delay of one cycle).
- Fig. 2 The DG for computation of 4-point circular convolution. (a) The DG. (b) Function of each node of the graph. (c) The vertical projection of DG.
- Fig. 3 The pipelined structure for reduced-hardware computation of 5-point DHT using ROM-based multiplication cells.
- Fig. 4 The DG for 4-point circular convolution by DA-technique. (a) The DG. (b) Function of each node of the DG. (c) The vertical projection of DG.
- Fig. 5 The DA-based pipelined single-array structure for reduced-hardware computation of 5-point DHT. (a) The structure. (b) Function of the PEs.
- Fig. 6 The DA-based fully pipelined 4-array structure for high-throughput computation of 5-point DHT. (a) The structure. (b) Function of the PEs.
- Fig. 7 The pipelined systolic structures for computation of 4-point DHT. (a) Structure and function of a linear array for the 4-point DHT. (b) 4-array structure for high-throughput computation of 4-point DHT. (Δ stands for unit delay).
- Fig. 8 The structure of a 4-point DHT module using pipelined adder units. (a) The 4-point DHT module. (b) Function of ADDER-1. (c) Function of ADDER-2.
- Fig. 9 The proposed structure for 8-point DHT.
- Fig. 10 The structure of prime-factor DHT for $N_1 = 5$ and $N_2 = 4$ (CASE-1). $\mathbf{Y}_i = \{y(4,i), y(3,i), y(2,i), y(1,i), y(0,i)\}$ and $\mathbf{X}_i = \{X(4,i), X(3,i), X(2,i), X(1,i), X(0,i)\}$ for $i = 0, 1, 2$, and 3 .
- Fig. 11 The structure of prime-factor DHT for $N_1 = 5$ and $N_2 = 3$ (CASE 2). $\mathbf{Y}_i = \{y(4,i), y(3,i), y(2,i), y(1,i), y(0,i)\}$ for $i = 0, 1$, and 2 .
- Fig. 12 Plot of the normalized ROM size in logarithmic scale of base 10 for various transform lengths of the proposed reduced-hardware direct-ROM-based prime-factor structure, proposed structure for power-of-2 transform lengths, and the existing structure of [22].
- Fig. 13 Plot of the ROM size in logarithmic scale of base 10 for various transform lengths of the proposed reduced-hardware DA-based structure and the

existing structures [23]–[25].

Fig. 14 Plot of the VLSI cost measure in logarithmic scale of base 10 for various transform lengths of the proposed direct-ROM-based high-throughput prime-factor structures and the existing structure of [22].

Fig. 15 Plot of the VLSI cost measure in logarithmic scale of base 10 for various transform lengths of the proposed high-throughput DA-based prime-factor structures and the existing structure of [23]–[25].

Fig. 16 The VLSI cost measures of the conventional multiplier-based structure and the DA-based structure.

LIST OF TABLES

- Table. I INDEX MAPPING A. INDEX MAPPING OF k TO l AND n TO m . B. VALUES OF $(kn)_5$ FOR DIFFERENT VALUES OF k AND n
- Table. II COMPARISON OF HARDWARE AND TIME-COMPLEXITIES OF PROPOSED PRIME-LENGTH DHT STRUCTURE, PROPOSED POWER-OF-TWO LENGTH DHT STRUCTURE AND THE EXISTING STRUCTURES
- Table. III COMPARISON OF HARDWARE AND TIME-COMPLEXITIES OF REDUCED-HARDWARE PRIME-FACTOR DHT STRUCTURES AND EXISTING STRUCTURES
- Table. IV HARDWARE AND TIME-COMPLEXITIES OF THE PROPOSED HIGH-THROUGHPUT PRIME-FACTOR DHT STRUCTURES (CASE 1 AND CASE 2)

Fig. 1

Fig. 2

Fig. 3

Fig. 4

(a)

For every new input sequence :

$Y \leftarrow Y_{in}$;

During every cycle time :

$X_{out} \leftarrow (X_{in} / 2) + Read(Y)$;

$Y \leftarrow Rotate_left(Y)$.

(b)

Fig. 5

(a)

During every cycle time :
 $X_{out} \leftarrow (X_{in} / 2) + Read(Y_{in})$;
 $Y_{out} \leftarrow Rotate_left(Y_{in})$.

(b)

Fig. 6

Fig. 7

Fig. 8

Fig. 9

Fig. 10

Fig. 11

Fig.12

Fig. 13

Fig. 14

Fig. 15

Fig. 16

(A)

$m \text{ or } l$	0	1	2	3
n	1	3	4	2
k	1	2	4	3

(B)

$\begin{array}{c} n= \\ k= \end{array}$	1	3	4	2
1	1	3	4	2
2	2	1	3	4
4	4	2	1	3
3	3	4	2	1

Table I

Structure	ROM (Words)	Adders	Throughput Per Cycle	ACT (Cycles)
ROM-based Structure of [22]	$N.2^L$	$2N+3$	1	N
Proposed ROM-based Single-Array	$(N-1).2^L$	$N+1$	1	$(N-1)$
Structure of [33] for $N=2^P$	$4(\log_2 N).2^L$	$3\log_2 N + 2$	2	$N/2$
Proposed Structure for $N=2^P$	$\{N(\log_2 N - 3)/4 + 1\}.2^L$	$3N(\log_2 N - 1)/2 + 2$	N	1
DA-based Structure of [23]	$L.2^{(N-1)}$	$2L + 5$	1	N
DA-based Structure of [24,25]	$N.2^{(N-1)}$	N	N/L	L
Proposed DA-based Single-Array	$L.2^{(N-2)}$	$(L+2)$	1	$(N-1)$

Table II

Structure	ROM (Words)	Adders	Throughput Per Cycle	ACT (Cycles)
ROM-based Structure of [22]	$N.2^L$	$2N+3$	1	N
Proposed ROM-based Single-Array	$(N_1 + N_2 - 2).2^L$	$N_1 + N_2 + 2$	1	$N - N_1$
Proposed ROM-based Double-Array	$2(N_1 + N_2 - 2).2^L$	$2(N_1 + N_2)$	2	$(N - N_1)/2$
DA-based Structure of [23]	$L.2^{(N-1)}$	$2L + 5$	1	N
DA-based Structure of [24,25]	$N.2^{(N-1)}$	N	N/L	L
Proposed DA-based Single-Array	$L(2^{N_1-2} + 2^{N_2-2})$	$2L+4$	1	$N - N_1$
Proposed DA-based Double-Array	$2L(2^{N_1-2} + 2^{N_2-2})$	$4L+6$	2	$(N - N_1)/2$

Table III

Structure	ROM (Words)	Adder	Throughput Per Cycle	ACT (Cycles)
ROM-based Structure (CASE-1: For $N_2=4$)	$(N - 4). 2^L$	$N+16$	4	$N/4$
ROM-based Structure (CASE-1: For $N_2=8$)	$(N - 7). 2^L$	$N+38$	8	$N/8$
ROM-based Structure (CASE-2)	$(N+N_2^2 - 3N_2+1). 2^L$	$N+N_2^2 - N_2+7$	N_2	N_1
DA-based Structure CASE-1: For $N_2=4$)	$4L.2^{(N/4)-2}$	$4L+20$	4	$N/4$
DA-based Structure (CASE-1: For $N_2=8$)	$8L.2^{(N/8)-2}$	$8L+46$	8	$N/8$
DA-based Structure (CASE-2)	$L(N_2.2^{(N_1-2)} + (N_2 - 1).2^{(N_2-2)})$	$2N_2(L+3/2) - L + 4$	N_2	N_1

Table IV

Pramod Kumar Meher (SM'03) received the first class degrees of B.Sc. (Honors) in physics, M.Sc. in physics (with electronics specials), and the Ph.D. degree for his research in the field of VLSI for digital signal processing, all from Sambalpur University, Sambalpur, India in 1976, 1978, and 1996, respectively.

He has a wide scientific and technical background covering physics, electronics and computer engineering. Currently, he is a Senior Fellow in the School of Computer Engineering, Nanyang Technological University, Singapore. He was a Professor at Utkal University, Bhubaneswar, India, during 1997–2002, a Reader in Electronics at Berhampur University, Berhampur, India during 1993–1997, and a Lecturer in Physics in various Government Colleges (in India) during 1981–1993. His research interest includes design of dedicated and reconfigurable architectures for computation-intensive algorithms pertaining to signal processing, image processing, secured communication and bioinformatics. He has published nearly 70 technical papers.

Dr. Meher was awarded the Samanta Chandrasekhar Award for excellence in research in Engineering and Technology for the year 1999. His name appeared in Marquis *Who's Who in the World* in 1998, 1999, and 2001. He is a Fellow of IEE, Chartered Engineer of the Engineering Council of U.K., and a Fellow of The Institution of Electronics and Telecommunication Engineers of India.

Thambipillai Srikanthan (SM'92) received the B.Sc. degree (Hons) in computer and control systems and the Ph.D. degree in system modeling and information systems engineering from Coventry University, Coventry, U.K.

He joined Nanyang Technological University (NTU) in June 1991, where he now holds a joint appointment as a Professor and as the Director of the Centre for High Performance Embedded Systems (CHiPES), which he founded in 1998. His research interests include system integration methodologies for embedded systems, architectural translations of compute intensive algorithms, high-speed techniques for image processing, and dynamic-routing. He has successfully launched a number of research initiatives, which are funded by either the local industry or external funding agencies such as the SERC of Singapore.

Dr. Srikanthan is a corporate member of the IEE, senior member of the IEEE, and an executive member of the IEE system-on-chip professional network.

Jagdish C. Patra (M'96) received the B.Sc. (Eng.) and M.Sc. (Eng.) degrees, both in electronics and telecommunication engineering from Sambalpur University, Sambalpur, India, in 1978 and 1989, respectively, and the Ph.D. degree in electronics and communication engineering from the Indian Institute of Technology, Kharagpur, India, in 1996.

After receiving the Bachelor's degree, he worked for various R&D, teaching and government organizations for about eight years. In 1987, he joined Regional Engineering College, Rourkela, India, as a Lecturer, where he was promoted to an Assistant Professor in 1990. In April 1999, he went to Technical University, Delft, The Netherlands as a Guest Teacher (Gasdoscent) for six months. Subsequently, in October 1999, he joined the School of Electrical and Electronic Engineering, Nanyang Technological University (NTU), Singapore as a research fellow. Currently he is serving as an Assistant Professor in the School of Computer Engineering, NTU. His research interests include intelligent signal processing using neural networks, fuzzy sets and genetic algorithms in the areas of data security, sensor networks, image processing and bioinformatics.

Dr. Patra is a member of the Institution of Engineers (India).