High-throughput memory-based architecture for DHT using a new convolutional formulation

Swamy, M. N. S.; Meher, Pramod Kumar; Patra, Jagdish Chandra

2007

Meher, P. K., Patra, J. C., & Swamy, M. N. S. (2007). High-Throughput Memory-Based Architecture for DHT Using a New Convolutional Formulation. IEEE Transactions on Circuits and Systems II: Express Briefs, 54(7), 606-610.

<https://hdl.handle.net/10356/94342>

<https://doi.org/10.1109/TCSII.2007.894407>

© 2007 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/TCSII.2007.894407>].

Downloaded on 26 Apr 2023 19:02:46 SGT

High-Throughput Memory-Based Architecture for DHT Using a New Convolutional Formulation

Pramod K. Meher, *Senior Member, IEEE*, Jagdish C. Patra, *Member, IEEE*, and M. N. S. Swamy, *Fellow, IEEE*

Abstract—A new formulation is presented for the computation of an N -point discrete Hartley transform (DHT) from two pairs of $[(N/2 - 1)/2]$ -point cyclic convolutions, and further used to obtain modular structures consisting of simple and regular memory-based systolic arrays for concurrent pipelined realization of the DHT. The proposed structures for direct-memory-based implementation is found to involve nearly the same hardware complexity as those of the existing structures, but offers two to four times more throughput and two to four times less latency compared with others. The distributed-arithmetic (DA)-based implementation is also found to offer very less memory-complexity and considerably low area-delay complexity compared with the existing DA-based structures.

Index Terms—Discrete Hartley transform (DHT), systolic array, VLSI.

I. INTRODUCTION

THE discrete Hartley transform (DHT) [1] has been established as a potential tool for various signal processing applications, e.g., computation of circular convolution and interpolation, error control coding, adaptive filtering, multi-carrier modulation and many other applications [2], [3]. It is popular due to its real-valued and symmetric transform kernel that is identical to that of its inverse. Not only is it used as a real-valued alternative to the discrete Fourier transform (DFT), but also for more efficient computation of the DFT, and other widely used unitary transforms like discrete cosine and sine transforms. Since the DHT is highly computation-intensive, algorithms and architectures are suggested for its efficient implementation in dedicated hardware. Systolic designs represent a popular architectural paradigm for efficient VLSI implementation of computation-intensive digital signal processing (DSP) applications supported by several attractive features such as simplicity, regularity and modularity of structure. Many systolic array architectures have, therefore, been suggested for efficient computation of the DHT [4]–[6]. The multipliers in these structures, however, use a large portion of the chip-area, and consequently enforce limitation on the maximum possible number of processing elements (PEs) to be used and the maximum transform-size to be implemented. Multiplierless memory-based techniques, on the other hand, have gained substantial popularity, in the recent

years, for their increased regularity resulting in cost-effective and area-time (AT)-efficient VLSI structures [7]–[11].

There are two basic variants of memory-based computing techniques. One of the techniques is the direct-memory-based implementation [12] of multiplications, while the other is based on distributed arithmetic (DA) [13]. In the direct-memory-based implementations, the multiplications of input values with the fixed coefficients are performed by a look-up table (LUT) of size 2^W , (W is the word-length), where each of the LUTs contains the pre-computed product values corresponding to all possible values of input samples with a fixed coefficient. The DA principle is used primarily to compute the inner-products by repeated shift-add operations of partial products corresponding to the successive bit-vectors of one of the input vectors. When one of the vectors is invariant, it is possible to store all the partial product values in a memory. In discrete transform evaluation, one of the vectors is derived from the input samples while the other vector is derived from the fixed coefficients of the transform kernel. It is, thus, possible to compute the transform components by using an LUT and a shift-accumulator, according to the DA principle. To minimize the input-output (I/O) bandwidth and to have hardware-efficiency of memory-based implementation, the DHT kernel has been converted into cyclic convolution forms in some previous work [7], [8], [10]. Since the average computation time (ACT) and the latency of direct-memory-based implementation is high for large convolution-lengths, DHT has been converted further into multiple number of shorter circular convolutions [9].

In this brief, we formulate a reduced-complexity concurrent algorithm to convert an N -point DHT into four identical cyclic convolutions of length $L = [(N/2 - 1)/2]$ in order to derive simple and regular AT-efficient linear systolic arrays for the DHT using the DA-based approach, as well as, the direct-memory-based approach.

II. MATHEMATICAL FORMULATION

We derive here a reduced-complexity algorithm for conversion of an N -point DHT into four matrix-vector products, where each matrix is of size $(L + 1) \times (L + 1)$, length of each vector is $(L + 1)$ and $L = [(N/2 - 1)/2]$. The matrix-vector products are converted thereafter to an L -point common cyclic convolutional representation.

A. Derivation of a Low-Complexity Computation Core

The DHT of a sequence $\{x(n)\}$ for $0 \leq n \leq N - 1$ may be defined in a common form as

$$X(k) = A(k) + B(k) \quad (1a)$$

P. K. Meher and J. C. Patra are with the School of Computer Engineering, Nanyang Technological University, Singapore 639798 (e-mail: aspkmehar@ntu.edu.sg; aspatra@ntu.edu.sg).

M. N. S. Swamy is with the Department of Electrical and Computer Engineering, Concordia University, Montreal, QC H3G 1M8, Canada (e-mail: swamy@ece.concordia.ca).

where

$$A(k) = \sum_{n=0}^{N-1} x(n) \cos \left[\frac{2\pi kn}{N} \right] \quad (1b)$$

$$B(k) = \sum_{n=0}^{N-1} x(n) \sin \left[\frac{2\pi kn}{N} \right] \quad (1c)$$

for $0 \leq k < N-1$. $A(k)$ and $B(k)$ for $0 \leq k < N-1$ are referred to as the even and the odd transforms in the rest of the brief. When N is even, the even and the odd transforms can be expressed as

$$A(k) = \sum_{n=0}^M a(n) \cos \left[\frac{\pi kn}{M} \right] \quad (2a)$$

$$B(k) = \sum_{n=0}^M b(n) \sin \left[\frac{\pi kn}{M} \right] \quad (2b)$$

for $0 \leq k < N-1$, and

$$a(n) = x(n) + x(N-n) \quad (2c)$$

$$b(n) = x(n) - x(N-n) \quad (2d)$$

for $1 \leq n \leq M-1$, $a(0) = x(0)$, $a(M) = x(M)$, $b(0) = b(M) = 0$, and $M = N/2$. For any odd value of M , the terms in (2a) and (2b) can be paired and combined together to have the even and the odd components of even and odd part transforms as

$$A(2k) = \sum_{n=0}^L a_1(n) \cos \left[\frac{2\pi k(2n+1)}{M} \right] \quad (3a)$$

$$A(M-2k) = \sum_{n=0}^L a_2(n) \cos \left[\frac{2\pi k(2n+1)}{M} \right] \quad (3b)$$

$$B(2k) = \sum_{n=0}^L b_1(n) \sin \left[\frac{2\pi k(2n+1)}{M} \right] \quad (3c)$$

$$B(M-2k) = \sum_{n=0}^L b_2(n) \sin \left[\frac{2\pi k(2n+1)}{M} \right] \quad (3d)$$

for $1 \leq k \leq L$ where $L = (M-1)/2$ and

$$a_1(n) = a(2n+1) + a(M-2n-1) \quad (3e)$$

$$a_2(n) = a(M-2n-1) - a(2n+1) \quad (3f)$$

$$b_1(n) = b(2n+1) - b(M-2n-1) \quad (3g)$$

$$b_2(n) = b(2n+1) + b(M-2n-1) \quad (3h)$$

for $0 \leq n \leq L$. Using the properties of sine and cosine functions, from (2) and (3), we can find further that $A(k) = A(N-k)$ and $B(k) = -B(N-k)$, for $1 \leq k \leq M-1$, $B(0) = B(M) = 0$, and

$$A(0) = \sum_{n=0}^M a_1(n) \quad (4a)$$

$$A(M) = \sum_{n=0}^M a_2(n). \quad (4b)$$

It may be observed that an N -point DHT can be computed from the summations of (4) along with four matrix-vector products of (3), where each matrix is of size $(L+1) \times (L+1)$ and length of each vector is $(L+1)$. To convert (3a)–(3d) into a common form, and to obtain the desired reduced-complexity convolutional representation, we use the following pair of identities:

$$\cos[(2n+1)x] = \left[(-1)^n + 2 \sum_{i=1}^n (-1)^{n+i} \cos[2xi] \right] \cos x \quad (5a)$$

$$\sin[(2n+1)x] = \left[1 + 2 \sum_{i=1}^n \cos[2xi] \right] \sin x. \quad (5b)$$

Using (5a), one can express (3a) and (3b) in a common form given by

$$P_i(k) = (p_i(0) + 2S_i(k)) \cos \left[\frac{2\pi k}{M} \right] \quad (6a)$$

where

$$S_i(k) = \sum_{n=1}^L p_i(n) \cos \left[\frac{4\pi kn}{M} \right] \quad (6b)$$

for $k = 1, 2, \dots, L$, and the sequence $\{p_i(n)\}$ is generated as

$$p_i(L) = a_i(L) \quad (6c)$$

$$p_i(n) = a_i(n) - p_i(n+1) \quad (6d)$$

for $n = 0, 1, \dots, L-1$, $i = 1$ corresponds to (3a) and $i = 2$ corresponds to (3b).

Similarly, using (5b), one can express (3c) and (3d) in the form

$$Q_i(k) = [q_i(0) + 2T_i(k)] \sin \left[\frac{2\pi k}{M} \right] \quad (7a)$$

where

$$T_i(k) = \sum_{n=1}^L q_i(n) \cos \left[\frac{4\pi kn}{M} \right] \quad (7b)$$

for $k = 1, 2, \dots, L$, and the sequence $\{q_i(n)\}$ is generated by successive accumulation given by

$$q_i(L) = b_i(L) \quad (7c)$$

$$q_i(L-n) = b_i(L-n) + q_i(L-n+1) \quad (7d)$$

for $n = 0, 1, \dots, L-1$, $i = 1$ and 2 correspond to (3c) and (3d), respectively.

It may be noted that both (6c) and (7c) [which represent the core computation of the DHT given by (3)] are of identical form, and can be computed by the same computing structure or by identical computing structures.

B. Convolutional Representation of the Computation Core

Let us represent (6c) and (7c) by a common form

$$Y(k) = \sum_{n=1}^L y(n) \cos \left[\frac{2\pi\phi(k, n)}{M} \right] \quad (8a)$$

for $k, n = 1, 2, \dots, L$, and the argument of cosine function is given by

$$\phi(k, n) = \langle 2kn \rangle_M, \quad \text{if } \langle 2kn \rangle_M \leq L \quad (8b)$$

$$\phi(k, n) = M - \langle 2kn \rangle_M, \quad \text{otherwise.} \quad (8c)$$

The symbol $\langle \cdot \rangle_M$ in (8) denotes modulo (M) operation. When M is prime, (8a) can be converted into an L -point circular convolution by mapping the indexes k to l and n to m according to the following equations:

$$n = \begin{cases} \langle \eta^{-m} \rangle_M, & \text{if } \langle \eta^{-m} \rangle_M \leq L. \\ M - \langle \eta^{-m} \rangle_M, & \text{otherwise.} \end{cases} \quad (9a)$$

$$k = \begin{cases} \langle \eta^l \rangle_M, & \text{if } \langle \eta^l \rangle_M \leq L. \\ M - \langle \eta^l \rangle_M, & \text{otherwise.} \end{cases} \quad (9b)$$

$$k = \begin{cases} \langle \eta^l \rangle_M, & \text{if } \langle \eta^l \rangle_M \leq L. \\ M - \langle \eta^l \rangle_M, & \text{otherwise.} \end{cases} \quad (10a)$$

$$k = \begin{cases} \langle \eta^l \rangle_M, & \text{if } \langle \eta^l \rangle_M \leq L. \\ M - \langle \eta^l \rangle_M, & \text{otherwise.} \end{cases} \quad (10b)$$

η is the $(M-1)$ -th primitive root of unity, such that $\eta^{M-1} \bmod M = 1$, and $\eta^j \bmod M \neq 1$, for $0 < j < (M-1)$. Using the mapping given by (9) and (10), each of the four (4a)-(4d) can be converted into L -point cyclic convolution form. An N -point DHT, therefore, can be computed from two pairs of L -point cyclic convolutions, where $L = (M-1)/2$ and $N = 2M$.

III. EXAMPLE OF CONVERSION OF DHT INTO CIRCULAR CONVOLUTION

For simple illustration of the proposed circular-convolution formulation, we show here the conversion of a 14-point DHT into two pairs 3-point circular convolutions. For transform length $N = 14$, we have $M = 7$ and $L = 3$. As discussed in Section II, and given by (8a), a 14-point DHT can be expressed as four matrix-vector products of the form

$$\begin{bmatrix} Y(1) \\ Y(2) \\ Y(3) \end{bmatrix} = \begin{bmatrix} C_2 & C_3 & C_1 \\ C_3 & C_1 & C_2 \\ C_1 & C_2 & C_3 \end{bmatrix} \begin{bmatrix} y(1) \\ y(2) \\ y(3) \end{bmatrix} \quad (11)$$

where $C_i = \cos(i\beta)$, for $i = 1, 2$ and 3 , and $\beta = (2\pi/7)$.

To convert (11) into the desired circular convolution, from (11) we can find the primitive root of unity η to be 3 for $M = 7$, and map the indexes according to (9) and (10) as shown in Table I. Mapping the indexes (k, n) to (l, m) according to Table I, (12) can be written as a cyclic convolution

$$\begin{bmatrix} Y(1) \\ Y(3) \\ Y(2) \end{bmatrix} = \begin{bmatrix} C_2 & C_3 & C_1 \\ C_1 & C_2 & C_3 \\ C_3 & C_1 & C_2 \end{bmatrix} \begin{bmatrix} y(1) \\ y(2) \\ y(3) \end{bmatrix}. \quad (12)$$

TABLE I
MAPPING OF INDEXES k AND n TO l AND m

m or l	0	1	2
n	1	2	3
k	1	3	2

Fig. 1. Proposed structure for computing a 14-point DHT.

Fig. 2. Structure of each computing module for direct-memory-based implementation. (a) Structure of the module. (b) Function of each PE.

A 14-point DHT may thus be computed from two pairs of 3-point circular convolutions of the form as shown in (12).

IV. PROPOSED STRUCTURE

The proposed structure for memory-based implementation of the DHT of transform length $N = 14$ is shown in Fig. 1. It consists of four computing modules for the computation of the four matrix-vector products of (3). An input adder unit is used to perform the additions and subtractions of (2c), (2d), (3e), (6d), and (7d) to feed the computing modules with the desired input values. Each of the computing modules consists of a convolution unit and an output cell, where the convolution operation can either be implemented by direct-memory-based approach [12] or DA-based approach [13]. The proposed structure of a computing module using direct-memory-based convolution is shown in Fig. 2. The convolution unit in this case is a simple and regular locally connected linear systolic array of three PEs for implementation of a 3-point cyclic convolution. The input values are fed to the individual PEs through a circularly extended input interface, such that the input values to a PE are staggered by one cycle-period with respect to the preceding PE to maintain the data dependency requirement. Function of each of the PEs of the structure is shown in Fig. 2(b). Each of the PEs performs a multiplication and an addition in each cycle period, where the multiplications in a PE are always performed

TABLE II
HARDWARE-AND TIME-COMPLEXITIES OF THE STRUCTURES FOR DIRECT-MEMORY-BASED IMPLEMENTATION OF THE DHT

structures	multipliers	adders	memory size	latency (cycles)	ACT (cycles)	area-time
Guo <i>et al.</i> [7]	3	$4N + 3$	$2N \cdot 2^{W/2}$	$N + 5$	N	$\approx 2N^2 \cdot 2^{W/2}$
Meher <i>et al.</i> [8]	0	$2N$	$(N - 1) \cdot 2^{W/2}$	N	$N - 1$	$\approx (N - 1)^2 \cdot 2^{W/2}$
Chiper <i>et al.</i> [9]	2	$2N + 7$	$(N - 1) \cdot 2^{W/2}$	$[(N - 1)/2] + 4$	$(N - 1)/2$	$\approx [(1/2)(N - 1)^2 \cdot 2^{W/2}]$
Proposed	4	$2(N + 9)$	$(N - 2) \cdot 2^{W/2}$	$[(N + 2)/4] + 4$	$(N - 2)/4$	$\approx (1/4)(N - 2)^2 \cdot 2^{W/2}$

Fig. 3. Structure of a processing element for the direct-memory-based implementation.

with a fixed coefficient. This feature of the PEs can be utilized to implement the multiplications by a ROM LUT that stores the product values for all possible input values for the given multiplying coefficient of the PE.

The proposed structure of the direct-memory-based PEs is shown in Fig. 3. It consists of a dual-port ROM of size $2^{W/2}$ words, where W is the word length. The dual-port ROM serves as LUT for all the possible values of the product. As shown in Fig. 3, the bits of each of the input word U_{in} are separated into two equal halves of $W/2$ bits each, and the two halves of the input word are fed in parallel to the pair of address ports of a dual-port ROM. The output of the ROM corresponding to the more significant half of the input word is left-shifted by $(W/2)$ -bit positions and added with the ROM output corresponding to the other half to generate the desired product value by an adder. The output of the adder is added further with input X_{in} , which is available to the PE from its left, to generate the output of the PE. It can be seen from Fig. 3 that the function of the direct-memory-based PEs can be implemented in three pipelined stages. The duration of a cycle period, therefore, can be $T = \max(T_{Mem}, T_A)$, where T_{Mem} and T_A are, respectively, the times required for a memory-read operation and the time required for an addition operation in the PEs. The right-most PE of the structure yields its first output three cycles after the first input arrives at its left-most PE, and produces its subsequent output in every cycle thereafter. It delivers a convolved output sequence in every three cycles, once the pipeline is filled in the first three cycles. The output cell consists of a shift-add circuit and a hard-wired-multiplier for the computations of (6a) or (7a). $A(0)$ and $A(M)$, given by (4a) and (4b), respectively, are computed by two accumulators in the input adder unit. Four DHT components can be calculated during a cycle period by the output adder unit of the proposed structure (Fig. 1). For DA-based implementation, the convolution units of the computing modules can be implemented by a DA-based architecture consisting of W PEs like the one suggested in [8]. Each PE of the DA-based architecture has a ROM of 2^L words and an adder. The details of the AT complexities of the direct-memory-based implementation and DA-based implementation of the proposed algorithm are discussed and compared in the following section.

V. HARDWARE AND TIME COMPLEXITIES

In Section II, it was shown that an N -point DHT can be computed via four number of L -point cyclic convolutions, where $L = (N/2 - 1)/2$. Each of the L -point cyclic convolutions is computed by the proposed direct-memory-based structure using a linear systolic array of L PEs. The structure would yield its first pair of convolution output after a latency of $(L + 2)$ cycles, and it would provide four outputs in every cycle after the latency period. A complete set of convolved output can be computed in every L cycles, where duration of a cycle period $T = \max(T_{Mem}, T_A)$ for the memory-based implementation. The ACT of the proposed structure thus amounts to $L = (N/2 - 1)/2$ cycles. The hardware- and time-complexities of the direct-memory-based realization of the proposed structure are listed in Table II, along with those of the corresponding existing structures [7]–[9]. For a fair comparison, all the structures are assumed to have dual-port ROMs for implementation of the multiplications. The proposed structure as well as the existing structures of [9] and [8] are found to involve nearly the same ROM size and similar adder-complexities. The structure of [7] has nearly twice the number of adders and double the memory complexities of the other structures. The proposed structure, on the other hand, requires half the number of cycles of ACT and half the latency as those of [9], and one-fourth of that of the structures of [7] and [8]. Since the memory elements contribute to most of the area of all these structures, memory size is taken as the area complexity, and AT complexity is approximated to be the product of memory size and the ACT as shown in Table II. The AT complexity of the proposed structure is found to be nearly half of that of the best of the existing structures.

The area of memory modules, multipliers and adders along with the addition-time and access-time for memory of various sizes are determined using the Synopsys DesignWare 0.18- μ m TSMC library for 16-bit data-width [14]. Addition-time is taken to be the cycle period for each of the structures, since it is found to be higher than the memory-access time. Accordingly, we have estimated the area complexity and the ACT of proposed structure and the existing structures of [7]–[9]. The AT complexities of the structures for different transform sizes thus obtained are plotted in Fig. 4. As shown in the figure, the proposed structure involves significantly less AT complexity compared to the existing structures, in accordance with the estimates in Table II. Since the area complexity of the proposed structure is nearly the same as that of the existing structures of [8], [9], lower AT complexity of the proposed structure is mainly due to its lower computation time. It may be noted that, although the word-length of the operands grow inherently in systolic computation during each stage of pipeline processing, the input at different stages may be suitably scaled to maintain a specific operand width.

TABLE III
HARDWARE- AND TIME-COMPLEXITIES OF THE STRUCTURES FOR DA-BASED IMPLEMENTATION OF THE DHT

structures	multipliers	adders	memory size	latency (cycles)	ACT (cycles)	area-time
Amira [11]	0	N	$N \cdot 2^{N-1}$	W	W	$\approx NW \cdot 2^{N-1}$
Guo [10]	2	$2W + 5$	$W \cdot 2^{N-1}$	$W + 5$	N	$\approx NW \cdot 2^{N-1}$
Meher <i>et al.</i> [8]	0	$W + 2$	$W \cdot 2^{N-2}$	$W + 1$	$N - 1$	$\approx (N - 1)W \cdot 2^{N-2}$
Proposed	4	$4(W + 5)$	$W \cdot 2^{(N/4+3/2)}$	$W + 5$	$(N - 2)/4$	$\approx (N - 2)W \cdot 2^{(N-2)/4}$

Fig. 4. AT complexities of the direct-memory-based implementation.

Accordingly, we have assumed the operands to be 16-bit wide for the proposed structure and the existing structures as well.

The hardware and time complexities of the proposed DA-based realization are listed in Table III, along with those of the existing DA-based DHT structures [8], [10], [11]. While the memory components constitute most part of all these structure, the proposed one involves only a small part of the memory-size of the existing structures. [It is $O(2^{N/4})$ while that of the existing designs is $O(2^N)$]. Throughput of the proposed structure is four times more compared with that of the structure of [10] and [8], and more than that of [11] for transform-size, $N < (4W + 2)$. Accordingly, the area-delay complexity of the proposed structure is also a small part of that of the existing DA-based structures. It may also be noted that the proposed DA-based implementation and direct-memory-based implementation have the same throughput, but differ in terms of latencies and hardware complexities. For $N > 2(W - 1)$, the direct-memory-based implementation will have less area-delay complexity compared with that of the DA-based realization of the proposed algorithm.

VI. CONCLUSION

We have presented a new formulation for the computation of an N -point DHT from two pairs of $[(N/2 - 1)/2]$ -point cyclic convolutions; and used that to obtain modular structures consisting of simple and regular systolic arrays for concurrent memory-based realization of the DHT. The proposed structure for direct-memory-based implementation is found to involve nearly the same hardware complexity as those of the existing structures, but offers two to four times more throughput and two to four times less latency compared with others. The DA-based implementation using the proposed algorithm is also found to offer very less memory-complexity and considerably low area-

delay complexity compared with the existing DA-based structures. Unlike the radix-2 and radix-4 algorithms, the proposed scheme offers close choices of transform-lengths. It can, therefore, be used efficiently in most applications without significant overhead of computation pertaining to extension of input-size to next-available length. The proposed convolutional formulation can be directly used for the implementation of the DFT, and can be extended further for efficient systolization of other sinusoidal transforms, as well.

REFERENCES

- [1] R. N. Bracewell, "Discrete Hartley transform," *J. Opt. Soc. Amer.*, vol. 73, no. 12, pp. 1832–1835, 1983.
- [2] S. Bouguezel, M. O. Ahmad, and M. N. S. Swamy, "A split vector-radix algorithm for the 3-D discrete Hartley transform," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 53, no. 9, pp. 1966–1976, Sep. 2006.
- [3] 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.
- [4] 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.
- [5] S. B. Pan and R. H. Park, "Unified systolic array for computation of DCT/DST/DHT," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 7, no. 2, pp. 413–419, Apr. 1997.
- [6] W. H. Fang and M. L. Wu, "Unified fully pipelined implementations of one- and two-dimensional real discrete trigonometric transforms," *IEICE Trans. Fund. Electron., Commun. Comput. Sci.*, vol. E82-A, no. 10, pp. 2219–2230, Oct. 1999.
- [7] 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'1994)*, Apr. 1994, vol. 2, pp. II/501–II/504.
- [8] P. K. Meher, T. Srikanthan, and J. C. Patra, "Scalable and modular memory-based systolic architectures for discrete Hartley transform," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 53, no. 5, pp. 1065–1077, May 2006.
- [9] D. F. Chiper, M. N. S. Swamy, and M. O. Ahmad, "An efficient systolic array algorithm for the VLSI implementation of a prime-length DHT," in *Proc. Int. Symp. Signals, Circuits Syst. (ISSCS 2005)*, Jul. 2005, vol. 1, pp. 167–169.
- [10] J. I. Guo, "A new distributed arithmetic algorithm and its hardware architecture for the discrete Hartley transform," *Pattern Recogn. Image Anal.*, vol. 10, no. 3, pp. 368–378, 2000.
- [11] A. Amira, "An FPGA based parameterisable system for discrete Hartley transforms implementation," in *Proc. 2003 Int. Conf. Image Processing (ICIP'03)*, Sep. 2003, vol. 2, pp. II-567–II-570.
- [12] H.-R. Lee, C.-W. Jen, and C.-M. Liu, "On the design automation of the memory-based VLSI architectures for FIR filters," *IEEE Trans. Consum. Electron.*, vol. 39, no. 3, pp. 619–629, Aug. 1993.
- [13] S. A. White, "Applications of the distributed arithmetic to digital signal processing: A tutorial review," *IEEE Acoust., Speech, Signal Process. Mag.*, vol. 6, no. 3, pp. 5–19, Jul. 1989.
- [14] "DesignWare. Foundry Libraries," Synopsys, Mountain View, CA [Online]. Available: <http://www.synopsys.com>