Computing Discrete Hartley Transform Using Algebraic Integers

Vassil Dimtrov and Ramin Baghaie

Helsinki University of Technology
Department of Electrical and Communications Engineering
P.O. BOX 3000, 02015 HUT, Finland
E-mail: vdimitro@wooster.hut.fi, ramin.baghaie@hut.fi

Abstract

An algorithm for computing the discrete Hartley transform is presented that is based on the algebraic integers encoding scheme. With the aid of this scheme, an error-free representation of the cas function becomes possible. In addition, for further complexity reduction an approximation scheme is proposed. Finally, for the implementation of the algorithm a fully pipelined systolic architecture with O(N) throughput is proposed.

1. Introduction

The discrete Hartley transform (DHT) is an attractive alternative to the discrete Fourier transform (DFT) because of its real-valued computation and properties similar to those of the DFT [1]. Another interesting property of the DHT is that the same kernel is used for both the transform and its inverse transform. Consequently, since its introduction the DHT has found its way to many digital signal processing applications [2]. The 1-D DHT of a N-point sequence $\{x_n, n = 0, \ldots, N - 1,$ and $N = 2^m\}$ is defined as:

$$X_k = \sum_{n=0}^{N-1} x_n \mathrm{cas}(2\pi \frac{kn}{N}), \quad 0 \le k \le N - 1 \qquad (1)$$

where $\mathrm{cas}\theta = \cos\theta + \sin\theta$.

Since the introduction of the DHT, a number of systolic architectures have been proposed, many of which are based on the direct implementation of algorithm [3,4]. In these implementations for the calculation of the *cas* function, different types of approximations have been introduced. Processing with algebraic integers, in which the signal sample is represented by a set of small integers, was introduced in [5].

Algebraic integers are roots of monic polynomials that have integer coefficients with leading coefficient equal to unity. The motivation for introducing this new mapping of real numbers is to drastically reduce the dynamic range of each of the independent computations.

In this paper, we illustrate how with the aid of the algebraic integers scheme, an efficient error-free systolic implementation of DHT can be obtained. This paper is organized as follows. In Section 2 algebraic integers interpretation of the discrete Hartley transform is presented. In Section 3, for the implementation of the proposed algorithm, a novel systolic array is presented. In Section 4, for further complexity reduction an approximation scheme is proposed. Certain hardware and throughput issues are discussed in Section 5. Concluding remarks are provided in Section 6.

2. Algebraic-Integer Interpretation of DHT

In [5], Cozzens and Finkelstein introduced a new approach for computing the DFT that uses the residue number system (RNS) in a ring of algebraic integers. Algebraic integers combine with RNS processing to add a second level of parallelism to integer RNS processing generalizing the quadratic residue number system (QRNS) concept. Algebraic integers are roots of monic polynomials that have integer coefficients with the leading coefficient equal to unity.

Consider the 16-point DHT. The kernel of this transformation is $\mathrm{cas}(2kn\pi/16)$ where $0 \le k, n \le N\text{-}1$. The classical method for calculating the *cas* function is to approximate the function in binary number system, which leads to rounding off errors. In this paper, we adopt the algebraic integers encoding scheme. Consider the first nonzero angle of the *cas* function that is $2\pi/16$. We can represent the *cos* and *sin* functions of this angle as:

$$\cos(2\pi/16) = \frac{\sqrt{2+\sqrt{2}}}{2} \quad (2)$$

$$\sin(2\pi/16) = \frac{\sqrt{2-\sqrt{2}}}{2} \quad (3)$$

Now without compromising the calculations, we omit the "2" from the denominator of Eqs. (2) and (3). Let us denote z as $z = 2\cos(2\pi/16) = \sqrt{2+\sqrt{2}}$, where z is a root of Eq. (4).

$$x^4 - 4x^2 + 2 = 0 \quad (4)$$

Consider now the polynomial of Eq. (5):

$$f(z) = \sum_{i=0}^{3} a_i z^i \quad (5)$$

where a_i are integers.

By assigning (0, 1, 0, 0) to a_i, we have an exact code for z. Similarly, by assigning (0, –3, 0, 1) to a_i we have an exact code for $2(\sin 2\pi/16)$. Consequently, (0, –2, 0, 1) is an exact representation of $2(\cos 2\pi/16)$. Proceeding in the same manner as in [6], we can obtain an *error-free* representation of the *cas* function necessary to evaluate the 16-point DHT. Tables 1, 2, and 3 present the corresponding coefficients of the *cos*, *sin* and *cas* functions for every required angle, correspondingly. Other remaining angles can be obtained by changing the signs of the given coefficients.

In order to demonstrate how Table 3 can be employed in the computation of the DHT, consider the following example. Assume that integer number x is needed to be multiplied by $2\text{cas}(4\cdot\pi/16)$. Since (–4, 0, 2, 0) is an error-free representation of $2\text{cas}(4\cdot\pi/16)$, this means that $x\cdot 2\text{cas}(4\cdot\pi/16)$ can be replaced by the following operation $x\cdot(-4, 0, 2x, 0)$.

Therefore, (–4x, 0, 2x, 0) is an error-free representation of $x\cdot 2\cos(4\cdot\pi/16)$. However, for the implementation of operations such as –4x and 2x only one shift operation is required. In other words, multiplication by the integers in Table 3 can be replaced by *only* one shift operation.

Up to this point, for the computation of the DHT, we have utilized an error-free format. However, the accuracy of the final reconstruction depends on the precision used to represent z. Therefore, one can estimate the precision that is needed to insure the required accuracy. As an example, if the word-length of data stream is 8-bit, then $\hat{z} = 2 - 2^{-3} - 2^{-5} + 2^{-8} = 1.84765625$ is a very good approximation of $z = 1.847759065\cdots$ with accuracy of 12 bits.

Table 1. Representation of the cos function needed for the 16-point DHT

	a_0	a_1	a_2	a_3	*Error*
$2\cos(0\cdot\pi/16)$	2	0	0	0	0
$2\cos(2\cdot\pi/16)$	0	1	0	0	0
$2\cos(4\cdot\pi/16)$	-2	0	1	0	0
$2\cos(6\cdot\pi/16)$	0	-1	0	1	0

Table 2. Representation of the sin function needed for the 16-point DHT

	a_0	a_1	a_2	a_3	*Error*
$2\sin(0\cdot\pi/16)$	0	0	0	0	0
$2\sin(2\cdot\pi/16)$	0	-3	0	1	0
$2\sin(4\cdot\pi/16)$	-2	0	1	0	0
$2\sin(6\cdot\pi/16)$	0	-1	0	0	0

Table 3. Representation of the cas function for 16-point DHT

	a_0	a_1	a_2	a_3	*Error*
$2\text{cas}(0\cdot\pi/16)$	2	0	0	0	0
$2\text{cas}(2\cdot\pi/16)$	0	-2	0	1	0
$2\text{cas}(4\cdot\pi/16)$	-4	0	2	0	0
$2\text{cas}(6\cdot\pi/16)$	0	-2	0	1	0

For the final reconstruction, in order to reduce the computation complexity of the polynomial of Eq. (5) Horner's rule is applied. As a result, Eq. (5) can be rewritten as:

$$f(z) = ((a_3 z + a_2)z + a_1)z + a_0 \quad (6)$$

There are however alternative methods to the Horner's rule. The interested reader may refer to [7] for more information on this topic. We have applied the Horner's rule due to its simplicity and straightforward applicability to systolic array [8].

3. Systolic Implementation

In general, there are two main approaches for the implementation of the DHT algorithm. The first method is to employ butterfly structures that lead to fast implementation of the algorithms. Although, fast transformations such as fast Hartley transform (FHT) require less computation as compared to the DHT, we are faced with some common problems.

Generally, the support of perfect shuffling between different stages requires global communication. This is considered as a drawback when very large scale integration (VLSI) implementation of such transformation is of interest. The second method is the direct implementation that utilizes architectures such as systolic arrays [8]. The systolic implementation of transforms such as the DHT enjoys simple communication and thus it is much more suitable for the VLSI implementation.

In this section, for the implementation of the proposed algorithm, a systolic architecture is presented. The proposed systolic architecture has the advantages of simplicity, modularity, and regularity [8].

Let us now consider the 16-point DHT that is based on the integers in Table 3. Figure 1 illustrates the case where $N = 16$. In Figure 2, the cell function of each Processor Element (PE) is presented. This systolic array is fully pipelined and its PEs only communicate with the neighboring cells, thus it is very suitable for the VLSI implementation.

Parameter a_{ij}^{t} $\{i = I, \ldots, 0$ and $j, t = 0, \ldots, (N - 1)\}$ corresponds to the integers in Table 3 and they are stored in the local registers of PE1. Thus, each PE1 requires N local registers. In a_{ij}^{t}, indices j and i reflect the position of PE1 in the array. Note that $(i, j) = (I, 0)$ corresponds to the top left PE1, where I is the degree of the polynomial of Eq. (5). Index t denotes time.

In Figure 1, the input data is first pipelined into the array of PE1s through an output-buffered N-way demultiplexer. The pipelined data x_{in} should then be multiplied by the integers in Table 3. However, it is easy to see that the required multiplication can be replaced with *only* one shift operation as discussed in Section 2. In PE1, the $shift_{a_{ij}^{t}}(x_{in})$ operation means that x_{in} is shifted once and the type of shift is determined by the a_{ij}^{t} coefficients. The shift operation can be implemented with a shifter such as a barrel shifter.

Finally, for the implementation of Eq. (6), a linear array consisting of PE2 and PE3 processors is utilized.

It is important to note that in the proposed method, we only require multiplications in PE2s. Furthermore, in these multiplications one of the multiplicands is z, which is constant. Thus, based on the word-length of the input data, one can estimate the precision that is needed to insure the required accuracy. Consequently, for the required multiplications, simple special-purpose multipliers can be designed.

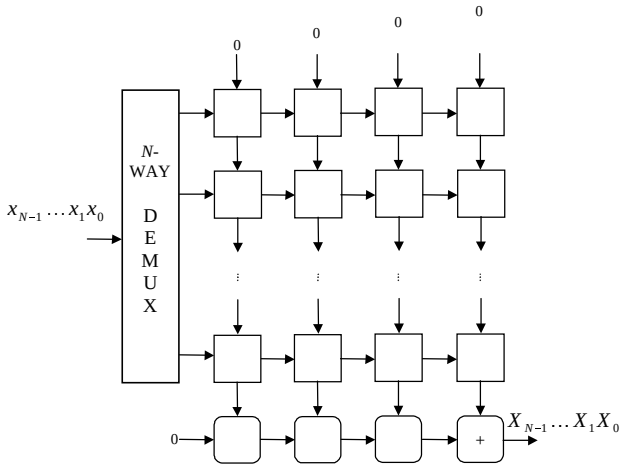

Figure 1. Systolic implementation of 16-point DHT algorithm

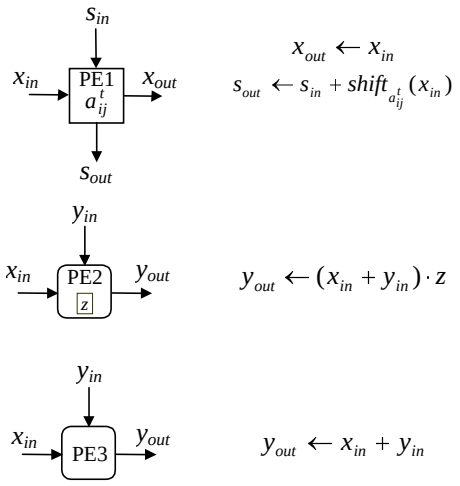

Figure 2. Input-output ports of the PEs and their cell functions

4. An Approximation Method

In the previous section, we illustrated that for an error-free presentation of the *cas* function that is required in the computation of the N-point DHT, I the degree of the polynomial of Eq. (5) is of length $(N / 4–1)$. It is important to note that in order to further reduce the degree of the polynomial, the proposed approach can be combined with *any* other DHT algorithm.

Table 4. Approximation of cas(2πk/32) for the 32-point DHT

k	*4-bit dynamic range*					*5-bit dynamic range*					*6-bit dynamic range*				
	a_0	a_1	a_2	a_3	*Error*	a_0	a_1	a_2	a_3	*Error*	a_0	a_1	a_2	a_3	*Error*
0, 8	2	0	0	0	0	2	0	0	0	0	2	0	0	0	0
1, 7	–4	4	–4	2	0.0002810746	3	–12	10	–2	0.0000124810	3	–12	10	2	0.0000124810
2, 6	0	–2	0	1	0	0	–2	0	1	0	0	–2	0	1	0
3, 5	–7	3	–8	5	0.0012906956	–1	11	–3	–1	0.0000157223	–20	–18	–15	17	0.0000027199
4	–4	0	2	0	0	–4	0	2	0	0	–4	0	2	0	0
9, 15	–1	–6	4	0	0.0010900574	14	–5	12	–7	0.0001309120	5	–8	–17	11	0.0000083340
10, 14	0	–4	0	1	0	0	–4	0	1	0	0	–4	0	1	0
11, 13	–7	–7	6	0	0.0008308395	8	–16	12	10	0.0000659494	–25	17	26	–15	0.0000029285
12	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

k	*7-bit dynamic range*					*8-bit dynamic range*				
	a_0	a_1	a_2	a_3	*Error*	a_0	a_1	a_2	a_3	*Error*
0, 8	2	0	0	0	0	2	0	0	0	0
1, 7	23	–30	–50	–15	0.0000035242	–115	48	86	–42	0.0000002312
2, 6	0	–2	0	1	0	0	–2	0	1	0
3, 5	–18	21	52	–32	0.0000000645	85	10	98	–69	0.0000000539
4	–4	0	2	0	0	–4	0	2	0	0
9, 15	47	–28	0	1	0.0000003224	47	–28	0	1	0.0000003224
10, 14	0	–4	0	1	0	0	–4	0	1	0
11, 13	–23	–30	50	–15	0.0000002730	–103	–125	122	–13	0.0000000542
12	0	0	0	0	0	0	0	0	0	0

Nevertheless, for an error-free presentations of the *cas* function, the length of a_i ($0 \le i \le I$) and correspondingly, the size of Table 3 is linearly dependent on *N*. Therefore, for a large *N*, *I* is relatively large. Thus, direct implementation of these tables may not be practical in an application. However, by employing different approximation techniques tremendous hardware reductions can be achieved, and yet a very good estimate of the aforementioned functions can be obtained.

In these approximations, we assign a fixed length to *I* that is independent of *N*. As a result, the degree of the polynomial only depends on the dynamic range of the input data that accordingly dictates the accuracy of the computations.

Let us demonstrate this by means of an example. Tables 4 illustrates the case for the 32-point DHT, in which *I* is fixed and equal to three. Consequently, some error is introduced. As an example, (3, –12, 10, 2) is a good approximation of $2\cos(2\pi/32)$ with an error of 0.0000124810. This means that by having $I = 3$, and assigning *only* 6 bits to the dynamic range of the coefficients, 15-bit accuracy is achieved.

A better understanding of several issues associated to the problem analyzed can lead to substantial improvements and result in more efficient algorithms. To name a few possible directions for future research:

a) By observing the integers in Table 4, one can see that in some cases more than two shift operations maybe required. One of the most attractive ways to obtain efficient multiplierless structure in the field of DSP is to design the coefficients used as a sum of very small number of powers of two. Thus, by applying some constraints we can reduce the computation cost of each multiplication to maximum two shifts and one addition.

b) Error analysis especially in the case of multiplierless architectures is quite essential. The main theoretical hurdle is the non-uniform distribution of the algebraic integers in the multidimensional lattice generated by them. In [9], Cozzens and Finkelstein have posed several conjectures in order to obtain a realistic picture of the error distribution and propagation in the case of the FFT.

To the best of our knowledge, these conjectures are still unsolved. For the computation of the DHT, we are faced with similar problems.

In this paper, in order to give the reader some impression about the error analysis associated with the technique proposed Table 4 was presented. This table realistically demonstrate the relationship between the dynamic range of the data and the approximation errors.

5. Hardware and Throughput Consideration

The proposed architecture requires $N(I + 1)$ PE1s, I PE2s and one adder. For evaluation of the throughput, we first assume that one time step of the global clock corresponds to one operation in PE2. For the computation of the first set of 1-D DHT $(2N + I)$ time steps are required. The successive sets are computed in an interval of N steps. Therefore, $O(N)$ time complexity is achieved.

In the proposed method, only I multipliers with a constant multiplicand are required. Furthermore, from Table 3 it is obvious that 9/16 of operations in PE1s are actually simple data transfer operations $s_{out} \leftarrow s_{in}$.

In Table 5, different systolic implementations of the 1-D DHT are summarized and compared, respectively. As an alternative to the conventional multipliers, in some of the implementations such as [3], COordinate Rotation DIgital Computer (CORDIC) processors [10] have been employed. In order to further reduce the complexity, the proposed method can be combined with *any* other DHT algorithms.

Table 5. Comparison of various systolic implementations for 1-D DHT

length	**Total number of multipliers**			
	Proposed *exact Method*	Proposed *approximation Method*	Chang & Lee [3]	Pan & Park [4]
8	1	1	8	8
16	3	2	16	16
32	7	3	32	32
N	$(N / 4 - 1)$	I (fixed)	N CORDICs	N

6. Conclusions

In this paper, we proposed a novel approach that is aimed at efficient implementation of the DHT algorithm. One of the advantages of the proposed algorithm is an *error-free* implementation of the DHT computation up until the final reconstruction. The proposed method can be combined with *any* DHT algorithms to achieve further hardware reduction. Furthermore, by introducing an approximation method hardware complexity was drastically reduced. For the implementation of the algorithm, a fully pipelined systolic architecture with $O(N)$ throughput was proposed.

Acknowledgements

The authors would like to thank Dr. Graham A. Jullien for his valuable comments. This work is part of a research project of the Institute of Radio Communication (IRC) funded by Technology Development Center (TEKES), NOKIA Research Center, Sonera Ltd. and the Helsinki Telephone Company.

References

[1] R. N. Bracewell, "Discrete Hartley transform," *Journal of Optical Society of America*, vol. 73, no. 12, pp. 1832-1835, Dec. 1983.

[2] R. N. Bracewell, "Aspects of Hartley transform," *Proceedings of the IEEE*, vol. 82, no. 3, pp. 381-387, Mar. 1994.

[3] L. W. Chang and S. W. Lee, "Systolic arrays for the discrete Hartley transform," *IEEE Transactions on Signal Processing*, vol. 39, no. 11, pp. 2411-2418, Nov. 1991.

[4] S. B. Pan and R. H. Park, "Unified systolic arrays for computation of DCT/DST/DHT," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 7, no. 2, pp. 413-419, Apr. 1997.

[5] J. H. Cozzens and L. A. Finkelstein, "Computing the discrete Fourier transform using residue number systems in a ring of algebraic integers," *IEEE Transactions on Information Theory*, vol. IT-31, no. 5, pp. 580-588, Sept. 1985.

[6] V. Dimitrov, G. A. Jullien, and W. C. Miller, "A new DCT algorithm based on encoding algebraic integers," in *Proceedings IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP'98*, Seattle, Washington, USA, vol. 3, pp. 1377-1380, May 1998.

[7] D. E. Knuth, *The Art of Computer Programming, Volume 2: Seminumerical Algorithms.* Addison-Wesley Publishing Company, 1981.

[8] S. Y. Kung, *VLSI Array Processors.* Englewood Cliffs, New Jersey: Prentice Hall, 1988.

[9] J. H. Cozzens and L. A. Finkelstein, "Range and error analysis for a fast Fourier transform computed over $Z[\omega]$," *IEEE Transactions on Information Theory*, vol. IT-33, no. 4, pp. 582-590, July 1987.

[10] J. E. Volder, "The CORDIC trigonometric computing technique," *IRE Trans. on Electronic Computers*, vol. EC8, no. 3, pp. 330-334, Sept. 1959.