New Efficient Algorithm for the Discrete Hartley Transform

Mounir T. Hamood

Electrical Engineering Department, College of Engineering, University of Tikrit, Salah Al-Deen Governorate, P O Box 42, Tikrit, Iraq
m.t.hamood@tu.edu.iq Tel: +9647714748928.

Abstract. In real-time signal processing applications, the discrete version of the transform introduced by Hartley (DHT) has been proved to be an efficient substitute to the discrete Fourier transform (DFT). A new algorithm for fast calculations of the DHT (FHT) based on radix−2/4/8 method is introduced in this paper. In comparison with the split radix FHT algorithm, the proposed algorithm has a comparable arithmetic complexity, but preserves the regularity and the simple butterfly structure of the radix−2 algorithm. The development of the algorithm is motivated by firstly deriving a new radix−2/8 FHT algorithm and then cascading it with the radix−4 and radix−2 FHT algorithms. The arithmetic complexity of the developed algorithm has been implemented and analyzed by calculating the number of real additions and multiplications for different transform lengths. Comparisons with the existing FHT algorithms have shown that this algorithm can be considered as a good compromise between structural and arithmetic complexity.

Keywords: Fast transform algorithms; Discrete Hartley Transform (DHT); Discrete Fourier Transform (DFT); radix-based algorithms;

1. Introduction

Over the decades, fast orthogonal transforms have been playing a major role in signal processing and communication fields. The result of the research carried out in this discipline over the last few years has been inspiring, producing in a wide range of fast algorithms and new transforms (Madisetti, 2017; Poularikas, 2009). In fact, transforms are continuously evolving and new ones are needed to solve many new or current challenges created by scientific developments or the request of new applications and tasks such as modern communication systems, signal/image processing and just to name a few. One the most important orthogonal transforms that was introduced by Bracewell (Bracewell, 1990; K. Jones, 2010), known as the discrete Hartley transform (DHT) become a progressively common as a substitute to the real valued discrete Fourier transform (RDFT). The DHT is particularly attractive in the applications involving real sequences, as it is likely that all the properties related with the DFT such the shifting and circular convolution theorems (Bi & Zeng, 2012; Oppenheim, Schafer, & Buck, 1999) can be applied to the DHT as well. As most spectrum analysis problems in real world involve real valued data only, therefore, substantial computational improvements can be gained using DHT. This is achieved
by the fact that the computation of $N$-point real valued data samples using the DHT results in $N$ real-valued samples. However, the DFT will produce $2N$ real valued sequence, where half of them are redundant. Thus, the arithmetic operations and the memory storage will be halved when using the DHT, as it requires real data arrays rather than complex as compared with the DFT. Furthermore, the DHT has the self-inverse property, i.e. there is no need to distinguish between the forward and inverse except from the scale factor (K. Jones, 2010). This feature is of most significance and in specific for the multidimensions (MD-DHTs) for large amounts of data is to be processed.

In (Lihong, Yonghong, Jouni, & Hannu, 1998), a VLSI oriented FFT algorithm was presented, for pipeline implementation of the discrete Fourier transform (DFT) by developing a radix-2/4/8 FFT algorithm. This algorithm is beneficial that it provides an improved efficiency over existing radix-8 and radix-4 algorithms and it offers versatility in that it is not restricted to transform lengths based on powers of 8 and 4 respectively, and that it is significantly easier to implement than the well-known split-radix algorithm. This is achieved by cascading radix-2, radix-2/8 and radix-2/4 butterflies together into a single butterfly.

A new decimation-in-frequency (DIF) algorithm for the fast calculation of the DHT is presented in this paper, by developing a new radix-2/8 FHT algorithm. This algorithm is further developed by incorporating the methodology given in (Lihong, Bingxin, Yonghong, & Tenhunen, 1998; Lihong, Yonghong, et al., 1998) and cascading it with radix-2 and radix-2/4 algorithms into a single algorithm. It provides a complexity that is comparable to the split radix FHT algorithm, but with the benefit of an implementation that is significantly easier. The real-time succeeded applications of the DHT depend mainly on the existent of the efficient algorithms, since the well-known DHT fast algorithms (FHTs) such as radix-2 (M. Hamood, 2016; Mounir Taha, 2013; Shah & Rathore, 2010), radix-4 (M. T. Hamood & Boussakta, 2011; K. J. Jones, 2006) and split-radix (Bi, 1994, 1997; Bouguetzel, Ahmad, & Swamy, 2004; Longyu, Huazhong, Jiaosong, Lu, & Lotfi, 2010) algorithms are developed. Consequently, the aim of this paper is to develop an efficient new algorithm using radix-2/4/8 method.

The paper is organized as follows; section 2 presents the complete development of the proposed radix-2/4/8 FHT DIF algorithm. Section 3 describes analysis of the algorithm’s performance in contrast to existing FHT algorithms. Lastly, the conclusion is given in section 4.

2. Algorithm Derivation

This section develops and analyses a new radix-2/4/8 FHT algorithm using decimation-in-frequency (DIF) approach. This has been achieved firstly by developing the radix-2/8 FHT algorithm. The derivation has been accomplished by using radix-8 decompositions to the odd-indexed sequence and the radix-2 decompositions to the even-indexed sequence and of $X(k)$. The desired algorithm will then be obtained by further incorporating it with the radix-2/4 and radix-2 FHT algorithms.

2.1 Development of the new FHT Algorithm

Conventionally, the radix-2/4/8 algorithm is developed using a combination of different fixed-radix decompositions (Lihong, Yonghong, et al., 1998). It is therefore convenient to describe these decompositions by generalizing them as a radix-$q$ FHT algorithm, where $q$ is assumed to be a power-of-two. The DHT $X(k)$ of an real valued sample $x(n)$ with transform size $N$ is defined as

$$X(k) = \sum_{n=0}^{N-1} x(n) \cos(\theta nk) \quad 0 \leq k \leq N-1$$

(1)

where $\cos(\theta nk) = \cos(\theta nk) + \sin(\theta nk)$ and $\theta = 2\pi/N$.

The development begins by dividing (1) into $q$ inner summations and substituting $n$ with $(n + \lambda N/q)$ for $n = 0, 1, \ldots, N/q - 1$

$$x(k) = \sum_{n=0}^{N/q-1} \sum_{\lambda=0}^{q-1} x(n + \lambda \frac{N}{q}) \cos(\theta(n + \lambda \frac{N}{q} + k))$$

(2)
In Eq. (2), the input samples $x(n)$ is divided into $q = 2^s$ sets ($s$ positive integer) so that each inner sum denotes DHT of size $N/q$. The output samples $X(k)$ is calculated as $q$-distinct parts, and each of them represented by $X(qk + l)$ has $(N/q)$ successive components indexed by $k$ for $k = 0, 1, ..., N/q - 1$ and $l = 0, 1, ..., q - 1$. Therefore, (2) becomes

$$X(qk + l) = \sum_{n=0}^{N/q - 1} x(n + \frac{N}{2q} l) \cos(\theta n + \frac{\pi}{2q}(qk + l))$$

$$= \sum_{n=0}^{N/q - 1} x(n + \frac{N}{2q} l) \cos(\theta_{nq} + nl + \frac{\pi}{4})$$

(3)

Equation (3) is the general radix$-q$ formula for the DIF FHT algorithm; decomposing it gives the preferred output sequence. Accordingly, the radix$-2/8$ algorithm can be calculated using (3) by using both radix$-8$ decimation (considering $q = 8$ and $l = 0, 1, ..., 7$) and radix$-2$ decimation (considering $q = 2$ and $l = 0, 1$). Consequently, the radix$-2$ decimation for even-indexed sequences ($q = 2$ and $l = 0$), becomes

$$x(2k) = \sum_{n=0}^{N/2 - 1} \sum_{l=0}^{q-1} x(n + \frac{N}{4} l) \cos(2\theta nk)$$

$$= \sum_{n=0}^{N/2 - 1} x(2n) \cos(2\theta nk)$$

(4)

and radix$-8$ decimation for the first odd-indexed sequences ($q = 8$ and $l = 1$), is given by:

$$x(2k+1) = \sum_{n=0}^{N/8 - 1} \sum_{l=0}^{q-1} x(n + \frac{N}{8} l) \cos\left(\frac{n\pi}{4} + \frac{\pi}{2} + \frac{\pi}{8}\right)$$

(5)

Expanding the partial summation of (5) yields:

$$X(2k+1) = \sum_{n=0}^{N/8 - 1} \left[x(n + \frac{N}{8} l) \cos\left(\frac{n\pi}{8} + \frac{\pi}{2} + \frac{\pi}{8}\right) + x(n + \frac{N}{8} l) \cos\left(\frac{n\pi}{8} + \frac{\pi}{2} + \frac{3\pi}{8}\right) + x(n + \frac{N}{8} l) \cos\left(\frac{n\pi}{8} + \frac{\pi}{2} + \frac{5\pi}{8}\right) + x(n + \frac{N}{8} l) \cos\left(\frac{n\pi}{8} + \frac{\pi}{2} + \frac{7\pi}{8}\right)ight]$$

(6)

Using the following cas identity

$$\text{cas}(a + b) = \text{cas}(a)\cos(b) + \text{cas}(-a)\sin(b)$$

(7)

the following relationship can be obtained:

$$\text{cas}\left(\theta n + \frac{\pi}{8}\right) = \text{cas}(\theta n)\cos\left(\frac{\pi}{8}\right) + \text{cas}(-\theta n)\sin\left(\frac{\pi}{8}\right) = -\text{cas}(\theta n)$$

(8)

$$\text{cas}\left(\theta n + \frac{\pi}{4}\right) = \text{cas}(\theta n)\cos\left(\frac{\pi}{4}\right) + \text{cas}(-\theta n)\sin\left(\frac{\pi}{4}\right) = \text{cas}(\theta n)$$

(9)

$$\text{cas}\left(\theta n + \frac{\pi}{2}\right) = \text{cas}(\theta n)\cos\left(\frac{\pi}{2}\right) + \text{cas}(-\theta n)\sin\left(\frac{\pi}{2}\right) = -\text{cas}(\theta n)$$

(10)

$$\text{cas}\left(\theta n + \frac{3\pi}{4}\right) = \text{cas}(\theta n)\cos\left(\frac{3\pi}{4}\right) + \text{cas}(-\theta n)\sin\left(\frac{3\pi}{4}\right) = \frac{1}{\sqrt{2}}[\text{cas}(\theta n) + \text{cas}(-\theta n)]$$

(11)
Substituting (8)-(13) into (6), we get:

\[
X(8k + 1) = \sum_{n=1}^{N-1} \left[ \frac{x(n) - x(n + \frac{N}{2})}{2} \cos(\theta n (8k + 1)) - \frac{x(n + \frac{N}{2}) - x(n)}{2} \cos(-\theta n (8k + 1)) \right]
\]

Replacing \([x(n) - x(n + N/2)]\) by \(u(n)\) into (14) we get,

\[
X(8k + 1) = \sum_{n=0}^{N-1} \left[ \frac{u(n)}{2} \left( u(n + \frac{N}{2}) - u(n + \frac{N}{2}) \right) \cos(\theta n (8k + 1)) \right]
\]

Using the relation:

\[
\sum_{n=0}^{N-1} u(n) \cos(-\theta n k - \theta n) = \sum_{n=0}^{N-1} u(n) \cos(\theta n k + \theta n)
\]

Substituting (16) into (15) and after some manipulations, the expression for \(X(8k + 1)\) can be written as:

\[
X(8k + 1) = \sum_{n=0}^{N-1} \left[ \frac{u(n)}{2} \left( u(n + \frac{N}{2}) - u(n + \frac{N}{2}) \right) \cos(\theta n (8k + 1)) \right]
\]

Succeeding the same procedure given by (5)-(17), other odd-indexed sequences \(X(8k + 3), X(8k + 5)\) and \(X(8k + 7)\) can be written as:

\[
X(8k + 2) = \sum_{n=0}^{N-1} \left[ \frac{u(n)}{2} \left( u(n + \frac{N}{2}) - u(n + \frac{N}{2}) \right) \cos(\theta n (8k + 2)) \right]
\]

\[
X(8k + 5) = \sum_{n=0}^{N-1} \left[ \frac{u(n)}{2} \left( u(n + \frac{N}{2}) - u(n + \frac{N}{2}) \right) \cos(\theta n (8k + 5)) \right]
\]

\[
X(8k + 7) = \sum_{n=0}^{N-1} \left[ \frac{u(n)}{2} \left( u(n + \frac{N}{2}) - u(n + \frac{N}{2}) \right) \cos(\theta n (8k + 7)) \right]
\]
Therefore, radix\(^{-2/8}\) FHT algorithm divides an \(N\)-point DHT given by (1) into four \(N/8\)-point DHTs defined by (17)-(20) and one \(N/2\)-point DHT defined by (4). The next step in deriving the algorithm, is by applying further the radix\(^{-4}\) and radix\(^{-2}\) FHT algorithms to the odd- and even-indexed samples of (4) respectively. Therefore, we get the following decompositions

i. For even-indexed samples

\[
\mathcal{X}(d_k) = \sum_{n=0}^{N-1} \left[ x(n/N + 2n/4 + n/8) + x(n/N + 2n/4 + n/8) \right] e^{j2\pi d_k n/N} 
\]  
(21)

ii. For odd-indexed samples

\[
\mathcal{X}(g_k + 2) = \sum_{n=0}^{N-1} \left[ x(n/N + 2n/4 + n/8) + x(n/N + 2n/4 + n/8) \right] e^{j2\pi g_k n/N} 
\]  
(22)

\[
\mathcal{X}(g_k + 0) = \sum_{n=0}^{N-1} \left[ x(n/N + 2n/4 + n/8) + x(n/N + 2n/4 + n/8) \right] e^{j2\pi g_k n/N} 
\]  
(23)

Finally, the preferred algorithm (radix\(^{-2/4/8}\) FHT) can be obtained by applying radix\(^{-2}\) FHT algorithm into (21), resulting in the remaining decompositions:

\[
\mathcal{X}(d_k) = \sum_{n=0}^{N-1} \left[ x(n/N + 2n/4 + n/8) + x(n/N + 2n/4 + n/8) \right] e^{j2\pi d_k n/N} 
\]  
(24)

\[
\mathcal{X}(g_k + 0) = \sum_{n=0}^{N-1} \left[ x(n/N + 2n/4 + n/8) + x(n/N + 2n/4 + n/8) \right] e^{j2\pi g_k n/N} 
\]  
(25)

Performing the computation in-place using double butterfly with 16-points provides an in-place butterfly of the proposed radix\(^{-2/4/8}\) DIF algorithm as depicted in Figure 1.
Figure 1. Radix-2/4/8 butterfly for the FHT algorithm, where dotted and solid lines stand for subtractions and additions respectively.

3. Computational Complexity

Equations (17)-(20) and (22)-(25) represent the radix-2/4/8 DIF FHT algorithm. By merging two butterflies together will remove the trivial arithmetic operations (multiplying by (-1, 0, 1) as shown in Figure 1. The developed in-place butterfly computes sixteen points and needs 66 additions and 32 multiplications. The whole transform fulfills the following relations.

\[ M(N) = 8M\left(\frac{N}{8}\right) + 2N - 22 \]

\[ A(N) = 8A\left(\frac{N}{8}\right) + \frac{31N}{8} - 10 \]

(26)

where \( A(N) \) and \( M(N) \) represents the total number of real additions and multiplications respectively. The computational complexity given by (26) is iterative. To get the closed form complexity at various transform size, the fundamental complexities values are required, which are the number of real additions and real multiplications that are required by size two and size four DHTs, that are given as \( A(2) = \)
2, \( A(4) = 8 \) \( A(8) = 24 \) and \( M(2) = 0, M(4) = 0, M(8) = 2 \). The comparison in total number of operations for different FHT algorithms is depicted in Table 1.

| Transform Size \( N \) | Radix–2/4/8 Algorithm | Radix–4 Algorithm | Radix–2 Algorithm |
|-------------------------|------------------------|-------------------|-------------------|
| 8                      | 24                     | 24                | 30                |
| 16                     | 74                     | 78                | 94                |
| 32                     | 210                    | 212               | 262               |
| 64                     | 520                    | 562               | 678               |
| 128                    | 1312                   | 1362              | 1670              |
| 256                    | 3088                   | 3314              | 3974              |
| 512                    | 7344                   | 7602              | 9222              |
| 1024                   | 16624                  | 17586             | 20998             |

An assessment has been done between radix–2, radix–4 and the developed radix–2/4/8 FHT algorithms, in terms of the total number of additions and multiplications as depicted in Figures. 2 and 3 respectively. As shown from these figures, a substantial saving can be achieved by using the newly developed radix–2/4/8 algorithm over other radix based FHT algorithms such as radix–4 and radix–2.

Furthermore, in order to verify the results based on arithmetic complexity for the developed radix–2/4/8 FHT algorithm. This algorithm and the corresponding radix–2 and radix–4 algorithms are implemented using MATLAB, and tested based on computer run-times on a Core i5 computer with speed of 1.8 GHz and 4 GB RAM. The results of the run-times are shown in Figure 4. The results of the computer run-time comparison are better than those obtained for the arithmetic operations comparison. This confirms the fact that the butterfly of the proposed algorithm computes triple stages concurrently as compared with the computation of single and double stages for radix–2 and radix–4 butterflies respectively.

Figure 2. Total number of multiplications comparison for different FHT algorithms
Figure 3. Total number of additions comparison for different FHT algorithms

Figure 4. The run-time for different FHT algorithms using MATLAB runs on Core i5 computer with speed of 1.8 GHz

4. Conclusion
The paper has been presented a newly developed radix–2/4/8 FHT algorithm using the decimation-in-frequency method. The efficiency of the proposed algorithm is comparable with the well-known split radix FHT algorithm in terms of arithmetical complexity, but it out-performs the split-radix by means of simple structure and regular indexing scheme which is offers a significant ease in both hardware and software implementations, in retrospect to the split-radix FHT algorithm. The new algorithm has been developed by introducing the radix–2/8 FHT DIF algorithm. The proposed radix–2/4/8 algorithm has been developed further by incorporating both radix–2 and radix–2/4 decimations. The resulting algorithm contains a number of radix-based FHT algorithms in a cascading configuration, by generalizing them as radix–q FHT algorithms, where q is assumed to be a power-of-two. The algorithm has been examined in respect to its arithmetical complexities of numerous transform sizes, in comparison to existing FHT algorithms based on radix–2 and radix–4 approaches.
References

[1] Bi, G. (1994). Split radix algorithm for the discrete Hartley transform. *Electronics Letters*, 30 (22), 1833-1835.

[2] Bi, G. (1997). New split-radix algorithm for the discrete Hartley transform. *IEEE Transactions on Signal Processing*, 45(2), 297-302.

[3] Bi, G., & Zeng, Y. (2012). *Transforms and Fast Algorithms for Signal Analysis and Representations*: Birkhäuser Boston.

[4] Bouguezel, S., Ahmad, M. O., & Swamy, M. N. S. (2004). *An efficient split-radix FHT algorithm*. Paper presented at the Proceedings of the International Symposium on Circuits and Systems, ISCAS '04.

[5] Bracewell, R. N. (1990). Assessing the Hartley transform. *IEEE Transactions on Acoustics, Speech and Signal Processing*, 38(12), 2174-2176.

[6] Hamood, M. (2016). New Decimation-In-Time Fast Hartley Transform Algorithm. *International Journal of Electrical and Computer Engineering* (IJECE), 6, 1654-1661.

[7] Hamood, M. T., & Boussakta, S. (2011). *New radix-based FHT algorithm for computing the discrete Hartley transform*. Paper presented at the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP).

[8] Jones, K. (2010). *The Regularized Fast Hartley Transform*: Springer Netherlands.

[9] Jones, K. J. (2006). Design and parallel computation of regularised fast Hartley transform. *IEEE Proceedings -Vision, Image and Signal Processing.*, 153(1), 70-78.

[10] Lihong, J., Bingxin, L., Yonghong, G., & Tenhunen, H. (1998). *Implementation of a low power 128-point FFT*. Paper presented at the International Conference on Solid-State and Integrated Circuit Technology.

[11] Lihong, J., Yonghong, G., Jouni, I., & Hannu, T. (1998). *A new VLSI-oriented FFT algorithm and implementation*. Paper presented at the IEEE International ASIC Conference.

[12] Longyu, J., Huazhong, S., Jiasong, W., Lu, W., & Lotfi, S. (2010). A novel split-radix fast algorithm for 2-D discrete Hartley transform. *Trans. Circuits and Systems Part I*, 57(4), 911-924.

[13] Madisetti, V. (2017). *Digital Signal Processing Fundamentals*: CRC Press.

[14] Mounir Taha, H. (2013). Improved FHT Algorithms for Fast Computation of the Discrete Hartley Transform. *Tikrit Journal of Engineering Sciences* 20(1), 62-69.

[15] Oppenheim, A. V., Schafer, R. W., & Buck, J. R. (1999). *Discrete-time signal processing*: Prentice Hall.

[16] Poularikas, A. D. (2009). *Transforms and applications primer for engineers with examples and MATLAB*: Taylor and Francis.

[17] Shah, G. A., & Rathore, T. S. (2010). *A New Architecture for Radix-2 DHT*. Paper presented at the International Conference on Computational Intelligence and Communication Networks.