On Computing the Discrete Hartley Transform

HENRIK V. SORENSEN, STUDENT MEMBER, IEEE, DOUGLAS L. JONES, STUDENT MEMBER, IEEE, C. SIDNEY BURRUS, FELLOW, IEEE, AND MICHAEL T. HEIDEMAN, STUDENT MEMBER, IEEE

Abstract—**The discrete Hartley transform (DHT) is a real-valued transform closely related to the DFT of a real-valued sequence. Bracewell has recently demonstrated a radix-2 decimation-in-time fast Hartley transform (FHT) algorithm. In this paper a complete set of fast algorithms for computing the DHT is developed, including decimation-in-frequency, radix-4, split radix, prime factor, and Winograd transform algorithms. The philosophies of all common FFT algorithms are shown to be equally applicable to the computation of the DHT, and the FHT algorithms closely resemble their FFT counterparts. The operation counts for the FHT algorithms are determined and compared to the counts for corresponding real-valued FFT algorithms. The FHT algorithms are shown to always require the same number of multiplications, the same storage, and a few more additions than the real-valued FFT algorithms. Even though computation of the FHT takes more operations, in some situations the inherently real-valued nature of the discrete Hartley transform may justify this extra cost.**

Introduction

The discrete Fourier transform (DFT) maps a length-N complex sequence to a length-N complex spectrum. Although symmetries of the DFT of a real-valued sequence can be exploited to reduce both the storage and the computation required to produce the transform of real-valued data, a transform that directly maps a real-valued sequence to a real-valued spectrum while preserving some of the useful properties of the DFT is sometimes preferred. One such transform closely related to the DFT is the discrete Hartley transform (DHT) [1]. Bracewell has recently presented a radix-2 decimation-in-time fast Hartley transform (FHT) algorithm [2], [3], demonstrating that the DHT gives rise to at least one fast algorithm that exists for the DFT. In this paper a complete set of fast algorithms foı computing the discrete Hartley transform is developed, and it is shown that the philosophy of all of the common fast Fourier transform (FFT) algorithms can also be applied to the computation of the DHT. The DHT can be computed from the DFT of a real-valued sequence, so FFT algorithms optimized for real-valued sequences must also be considered viable fast algorithms for computation of the discrete Hartley transform. The operation counts for the FHT algorithms are determined and compared to the counts for corresponding real-valued FFT algorithms. A unified development of algorithms for the real-valued FFT and the DHT gives insight into both the discrete Hartley transform and fast algorithms.

Manuscript received December 18, 1984; revised March 22, 1985. This work was supported by the National Science Foundation (NSF) under grant ECS 83-14006 and an NSF graduate fellowship.

The authors are with the Department of Electrical and Computer Engineering, Rice University, Houston, TX 77251.

The Discrete Hartley Transform

The discrete Hartley transform pair is defined for a real-valued length-N sequence $x(n)$, $0 \le n \le N-1$, by the following equations:

$$H(k) = \sum_{n=0}^{N-1} x(n) \operatorname{cas}\left(\frac{2\pi}{N} kn\right) \qquad 0 \le k \le N-1 \quad (1a)$$

$$x(n) = \frac{1}{N} \sum_{k=0}^{N-1} H(k) \operatorname{cas}\left(\frac{2\pi}{N} kn\right) \qquad 0 \le n \le N-1 \quad (1b)$$

where $\operatorname{cas}(x) = \cos(x) + \sin(x)$. The symmetry of the transform pair is a valuable feature of the DHT. It should be observed that the forward transform differs from the DFT only by lack of the "$-j$" multiplying the sine term, which implies that the DHT is equivalent to simply subtracting the imaginary from the real portion of the DFT of a real-valued sequence.

$$\text{DHT}[x(n)] = \text{Re}\{\text{DFT}[x(n)]\} - \text{Im}\{\text{DFT}[x(n)]\}. \quad (2)$$

Since the real part of the DFT of a real-valued sequence is even and the imaginary part is odd, the DFT can easily be calculated from the DHT as follows:

$$\text{Re}\{\text{DFT}[x(n)]\} = 1/2\{\text{DHT}[x(N-n)] + \text{DHT}[x(n)]\} \quad (3a)$$

$$\text{Im}\{\text{DFT}[x(n)]\} = 1/2\{\text{DHT}[x(N-n)] - \text{DHT}[x(n)]\} \quad (3b)$$

where $x(N) = x(0)$. Properties of the DHT are easily proven by use of DFT properties applied to these relationships. Many properties are quite similar to the corresponding theorems for the DFT. One of the more useful properties is the shift theorem

$$\text{DHT}[x(n+M)](k) = H(k)\cos\left(\frac{2\pi}{N} Mk\right) - H(N-k)\sin\left(\frac{2\pi}{N} Mk\right). \quad (4)$$

The stretch theorem states that if a sequence is stretched by inserting zeros between the samples, the transform will repeat. This theorem is the same as for the DFT. The DHT has a convolution property as well, according to the following equation:

$$\text{DHT } [x_1(n)*x_2(n)](k) = \frac{1}{2}[H_1(k)H_2(k) + H_1(k)H_2(N-k) + H_1(N-k)H_2(k) - H_1(N-k)H_2(N-k)]. \qquad (5)$$

These properties allow both the development of fast Hartley transform algorithms [2] and the use of these transforms in important applications such as fast convolution.

Index Mapping

The concept of index mapping [4], [5] provides a powerful approach to the development of fast algorithms, and most of the well-known FFT algorithms can be derived using this technique. Index mapping is a method of mapping a one-dimensional array of size $N = N_1N_2$ onto a two-dimensional array of size N_1 by N_2. If N_1 and N_2 are relatively prime, a prime factor map (PFM) can be derived, while if they are not relatively prime a common factor map (CFM) will result. The mapping is done by substituting

$$n \equiv K_1n_1 + K_2n_2 \pmod N \qquad (6a)$$

$$k \equiv K_3k_1 + K_4k_2 \pmod N \qquad (6b)$$

where n is the time index and k the frequency index. The factors K_1 to K_4 depend on which type of map is used, and the specific requirements are given in [4]. When the index mapping is used with DFT's, the term $e^{-j2\pi nk/N}$ factors into a product of four terms, of which one or two can be forced to unity by suitable choices of the constants K_1 through K_4. This reduction in terms makes it possible to save operations by breaking the DFT into smaller DFT's. Applying the general map equations (6) to the DHT results in a complicated expression which will not be reproduced here. Instead, the particular CFM map

$$n \equiv n_1 + Ln_2 \qquad (7a)$$

$$k \equiv k_1 + Kk_2 \qquad (7b)$$

where $KL = N$, will be examined. Substituting the mapping into (1) and applying the two identities

$$\text{cas } (x+y) = \text{cas } (x) \cos (y) + \text{cas } (-x) \sin (y) \qquad (8a)$$

$$\text{cas } (-x) = \text{cas } (x) - 2 \sin (x) \qquad (8b)$$

results in the following equation.

$$\begin{aligned} H(k_1 + Kk_2) &= \sum_{n_1=0}^{L-1}\sum_{n_2=0}^{K-1} x(n_1 + Ln_2) \\ &\cdot \Bigg[\text{cas}\left(\frac{2\pi}{N}n_1k_1\right)\text{cas}\left(\frac{2\pi}{K}n_2k_1\right)\text{cas}\left(\frac{2\pi}{L}n_1k_2\right) \\ &-2\sin\left(\frac{2\pi}{K}n_2k_1\right)\cos\left(\frac{2\pi}{L}n_1k_2\right)\sin\left(\frac{2\pi}{N}n_1k_1\right) \\ &-2\sin\left(\frac{2\pi}{K}n_2k_1\right)\sin\left(\frac{2\pi}{L}n_1k_2\right)\cos\left(\frac{2\pi}{N}n_1k_1\right) \\ &-2\sin\left(\frac{2\pi}{K}n_2k_1\right)\sin\left(\frac{2\pi}{L}n_1k_2\right)\sin\left(\frac{2\pi}{N}n_1k_1\right) \\ &-2\cos\left(\frac{2\pi}{K}n_2k_1\right)\sin\left(\frac{2\pi}{L}n_1k_2\right)\sin\left(\frac{2\pi}{N}n_1k_1\right)\Bigg]. \end{aligned} \qquad (9)$$

This expression is still complicated, but when $L = 2$ and $K = N/2$ the last three terms become zero. Trigonometric identities can then be applied to convert this expression into the radix-2 decimation-in-time decomposition [6]. Decimation-in-frequency and higher radix algorithms are derived in the same manner.

The Radix-2 Decimation-in-Time Fast Hartley Transform

Choosing $L = 2$ and $K = N/2$ as described in the previous section corresponds to dividing a length $N = 2^M$ DHT into two length-$N/2$ DHT's; one over the even-indexed samples H_{2n} and one over the odd-indexed samples H_{2n+1}. Equation (9) then becomes, with the indices of the half-length transforms evaluated modulo $N/2$,

$$H(k) = H_{2n}(k) + H_{2n+1}(k)\cos\left(\frac{2\pi}{N}k\right) + H_{2n+1}(N-k)\sin\left(\frac{2\pi}{N}k\right) \quad 0 \le k \le N-1 \qquad (10)$$

which corresponds to the decomposition used by Bracewell [2]. Each of the length-$N/2$ transforms can be decomposed into two length-$N/4$ transforms, and this decomposition can be applied recursively until length-2 transforms are obtained. Thus the structure of the fast algorithm resembles that of the FFT derived by Cooley and Tukey [6]-[8]. The permutation of the time indices will be exactly the same as for a Cooley-Tukey radix-2 FFT, because the same decomposition sequence is applied to both.

Since it is generally desired that the algorithm be in-place, it must be noted that at each stage both the kth term and the $(N/2 - k)$th term from each length-$N/2$ DHT are required for the computation of one output point. Thus four elements must be included in each butterfly to avoid overwriting an element that will be needed later. This also results in a savings in the number of multiplications, by use of the identities

$$\sin\left(\frac{2\pi}{N}k(N-n)\right) = -\sin\left(\frac{2\pi}{N}kn\right) \qquad (11a)$$

$$\cos\left(\frac{2\pi}{N}k(N-n)\right) = \cos\left(\frac{2\pi}{N}kn\right). \qquad (11b)$$

Application of these symmetries reduces the operation count to four real multiplications and six real additions per four point butterfly. Fig. 1 shows a pictorial representation of the Hartley butterfly. The four outputs indexed as $k = (N/4)i$, $i = 0, 1,$

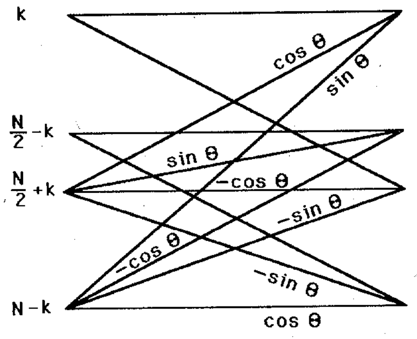


Fig. 1.

2, 3 should be computed separately because the twiddle factors are 0 to 1 and can be done without multiplications. Similarly, the first two stages require no multiplications and should be implemented separately.

The discrete Hartley transform can be computed from the discrete Fourier transform of a real-valued sequence by taking the difference of the real and imaginary frequency components, so it is of interest to compare the operation count required to obtain the DHT using a radix-2 decimation-in-time FHT and a corresponding radix-2 decimation-in-time FFT for real input data. Since the DFT of a real-valued sequence has an even real part and an odd imaginary part, an efficient way to store the transform is to maintain the coefficients in a length-N array in which the real coefficients are stored in the locations 0 through $N/2$, and the imaginary components are stored in locations $N/2 + 1$ through $N - 1$. Savings in computation are possible by realizing that only the first half of the DFT need be calculated due to the complex conjugate symmetry of the transform of a real sequence [9].

The decimation-in-time algorithm decomposes the DFT into shorter transforms of subsets of the real sequence, so the savings in both memory and computation can be utilized at every stage of the real-valued FFT algorithm. A standard radix-2 FFT butterfly with a complex twiddle factor is applied over only half of the DFT length at each stage, returning two real and two imaginary coefficients. The cost of this operation is four real multiples, six real adds, and storage for four real data elements. This cost is exactly equal to that of the corresponding Hartley butterfly. A close inspection of the multiplies in the Hartley butterfly reveals that they form a complex multiply by exactly the same twiddle factor found in the corresponding butterfly in the standard FFT formulation. Only the method of compactly storing the Fourier transform of a real-valued sequence has changed; the underlying fast algorithm is essentially the same.

When computing the outputs with indices $k = (N/4)i$, $i = 0, 1, 2, 3$ for the real-valued FFT, a special butterfly will be used, since the input data has only real components and the twiddle factors require no multiplications. This butterfly requires only two additions, whereas the special butterfly in the DHT requires four additions. The first two stages in the FFT algorithm will be done separately since they require no multiplications, for a total of six additions per four outputs, as compared to eight for the first two stages of the FHT. The

TABLE I
OPERATION COUNTS FOR PROGRAMS IN APPENDIX

Length	Mults	Adds	Mult + Add
RADIX-2			
4	0	8	8
8	4	26	30
16	20	74	94
32	68	194	262
64	196	482	678
128	516	1154	1670
256	1284	2690	3974
512	3076	6146	9222
1024	7172	13826	20998
2048	16388	30722	47110
4096	36868	67586	104454
RADIX-4			
4	0	8	8
16	14	70	84
64	142	450	594
256	942	2498	3440
1024	5294	12802	18096
4096	27310	62466	89776
SPLIT-RADIX			
4	0	8	8
8	2	22	24
16	12	64	76
32	42	166	208
64	124	416	540
128	330	998	1328
256	828	2336	3164
512	1994	5350	7344
1024	4668	12064	16732
2048	10698	26854	37552
4096	24124	59168	83292

program FHT2T in Appendix implements the radix-2 decimation-in-time FHT algorithm described above, with an operation count of $N \log_2 N - 3N + 4$ multiplications and $(3N \log_2 N - 3N + 4)/2$ additions (as tabulated in Table I). The corresponding real-valued FFT requires the same number of multiplications and $(3N \log_2 N - 5N + 8)/2$ additions. The FHT algorithm takes $N - 2$ more additions than required by the real-valued FFT, exactly the operations needed to convert the real-valued DFT into a DHT. In essence, the FHT algorithm moves these additions from the last stage of the real-valued FFT algorithm and buries them within the FHT. Thus it takes exactly the same number of operations to compute the discrete Hartley transform using a real-valued DFT algorithm and adding the real and imaginary components as it does to compute the DHT directly with an FHT algorithm of the same sophistication. Computation of the real-valued DFT by means of the FHT requires $2N - 4$ more additions than the direct method. The storage requirements are the same; the DHT can be viewed as another way of compactly storing the DFT of a real-valued sequence, as at the expense of $N - 2$ additions.

Special techniques for saving operations that have been applied to the FFT can also be applied to the FHT. Since the four real multiplications in the Hartley butterfly correspond to a complex multiply, they can be implemented with three real multiplications and three additions. The use of special butterflies to remove trivial multiplications and lookup tables for the trigonometric coefficients are applicable to the FHT. An FHT program that incorporates these features can be developed by comparison with a corresponding FFT program such as those found in [5]. As will be shown later in this paper, larger radix algorithms can also be used to compute the DHT. The fundamental similarities of the DHT and the DFT make most special techniques equally applicable to fast algorithms for computing either transform.

Decimation-in-Frequency Fast Hartley Transform

The common-factor index map $n = n_1 + n_2N/2$ and $k = k_1 + 2k_2$ leads to a decimation-in-frequency Hartley transform. By using $K = 2$ and $L = N/2$ in (9) this mapping results in the decomposition

$$H(2k_2) = \sum_{n_1=0}^{L-1} [x(n_1) + x(n_1 + L)] \text{ cas } \left(\frac{2\pi}{L} n_1 k_2\right) \qquad 0 \le k_2 \le L-1 \tag{12a}$$

$$\begin{aligned} H(2k_2+1) = \sum_{n_1=0}^{L-1} &[x(n_1) - x(n_1+L)] \\ &\cdot \left\{\text{cas}\left(\frac{2\pi}{L} n_1k_2\right) \cos\left(\frac{2\pi}{N} n_1\right) \right. \\ &\left. + \text{cas}\left(\frac{2\pi}{L}(N-n_1)k_2\right) \sin\left(\frac{2\pi}{N} n_1\right)\right\} \\ &0 \le k_2 \le L-1. \end{aligned} \tag{12b}$$

The sum in (12a) is a discrete Hartley transform, but the sum in (12b) is not. By splitting (12b) into two sums and substituting n_1 for $N - n_1$ in the last sum, the following equation is obtained

$$\begin{aligned} H(2k_2+1) = \sum_{n_1=0}^{L-1} &\left[\{x(n_1) - x(n_1+L)\} \cos\left(\frac{2\pi}{N} n_1\right)\right. \\ &\left. + \{x(L-n_1) - x(N-n_1)\} \sin\left(\frac{2\pi}{N} n_1\right)\right] \\ &\cdot \text{cas}\left(\frac{2\pi}{L} n_1k_2\right) \qquad 0 \le k_2 \le L-1 \end{aligned} \tag{13}$$

which is a discrete Hartley transform. Each of the length-$N/2$ DHT's can be decomposed in a similar fashion to complete the decimation-in-frequency algorithm. The butterfly that results from this method is the mirror image of the butterfly in Fig. 1. This algorithm is especially interesting because no simple methods are known for implementing a decimation-in-frequecy FFT for real-valued input. The operation count is the same as for the radix-2 decimation-in-time algorithm discussed earlier.

Radix-4 Fast Hartley Transform

By choosing $L = 4$ and $K = N/4$ in (9), a radix-4 decimation-in-time FHT can be derived. The decomposition becomes

$$\begin{aligned} H(k) = H_{4n}(k) &+ \cos\left(\frac{2\pi}{N} k\right) H_{4n+1}(k) \\ &+ \sin\left(\frac{2\pi}{N} k\right) H_{4n+1}(N-k) \\ &+ \cos\left(\frac{2\pi}{N} 2k\right) H_{4n+2}(k) \\ &+ \sin\left(\frac{2\pi}{N} 2k\right) H_{4n+2}(N-k) \\ &+ \cos\left(\frac{2\pi}{N} 3k\right) H_{4n+3}(k) \\ &+ \sin\left(\frac{2\pi}{N} 3k\right) H_{4n+3}(N-k) \\ 0 \le k \le N-1 & \end{aligned} \tag{14}$$

where H_i is a length-$N/4$ DHT. It is possible to perform the computation in place and save half the twiddle factor multiplications in (14) by doing two butterflies at a time, exactly as was done in the radix-2 algorithm. Hence only 12 multiplications and 22 additions are needed per 8 output points. The total multiplication count for a radix-4 FHT is $3N/2 \log_4(N) - 7N/3 + 10/3$ and the total number of additions is $11N/4 \log_4(N) - 5N/4 + 2$. Actual counts can be found in Table I. The number of real multiplications is exactly half the number required in a corresponding complex radix-4 FFT, and hence the same as in a real-valued radix-4 FFT. The number of additions is slightly larger than that required by a real-valued radix-4 FFT. Again, the multiplications take the form of the three complex multiplications found in the radix-4 FFT butterfly and can be computed with three real multiplications per complex multiply if desired. In the Appendix a radix-4 decimation-in-time FHT program, FHT4T, is given.

Split Radix Fast Hartley Transform

One of the most efficient algorithms for computing the DFT is the split radix FFT [10]. The split radix algorithm applies a radix-2 decomposition to the even indexed samples and a radix-4 decomposition to the odd indexed samples. In this way it is possible to obtain an algorithm for a length $N = 2^M$ sequence using even fewer multiplications and additions than a radix-4 FFT. The split radix algorithm follows indirectly from an index mapping approach. To derive a split radix FHT choose $K = 4$ and $L = N/4$, which gives a decimation in frequency radix-4 FHT. Now combine the two even indexed terms to get

$$H_{2k} = \sum_{n=0}^{N/2-1} [x_n + x_{n+N/2}] \text{ cas } \left(\frac{2\pi}{N} 2kn\right) \qquad 0 \le k \le \frac{N}{2} - 1 \tag{15a}$$

$$H_{4k+1}=\sum_{n=0}^{N/4-1}\Bigg[\{x_n-x_{n+N/2}+x_{N/4-n}-x_{3N/4-n}\}\cdot\cos\left(\frac{2\pi}{N}n\right)+\{x_{n+3N/4}-x_{n+N/4}+x_{N/2-n}-x_{N-n}\}\cdot\sin\left(\frac{2\pi}{N}n\right)\Bigg]\operatorname{cas}\left(\frac{2\pi}{N}4kn\right)$$

$$0\le k\le\frac{N}{4}-1 \tag{15b}$$

$$H_{4k+3}=\sum_{n=0}^{N/4-1}\Bigg[\{x_n-x_{n+N/2}+x_{3N/4-n}-x_{N/4-n}\}\cdot\cos\left(\frac{2\pi}{N}3n\right)+\{x_{n+3N/4}-x_{n+N/4}-x_{N/2-n}+x_{N-n}\}\cdot\sin\left(\frac{2\pi}{N}3n\right)\Bigg]\operatorname{cas}\left(\frac{2\pi}{N}4kn\right)$$

$$0\le k\le\frac{N}{4}-1. \tag{15c}$$

A program, FHTSR, implementing this algorithm is shown in the Appendix. Again two butterflies are done at the same time, and from the code it can be seen that 8 multiplications and 16 additions are required per 8 output points. This gives a total of $2N/3\log_2(N) - 19N/9 + 3 + (-1)^M/9$ multiplications and $4N/3\log_2(N) - 14N/9 + 3 + (-1)^M 5/9$ additions. The number of multiplications is exactly the same as for a similar real-valued split radix FFT, while the number of additions slightly exceeds that.

The Prime Factor Fast Hartley Transform

If the transform length N is a product of relatively prime factors, the transform can be expressed [5] in terms of a prime factor map $n \equiv K_1n_1 + K_2n_2 \pmod N$ and $k \equiv K_3k_1 + K_4k_2 \pmod N$, where $K_1K_4 = K_2K_3 = 0 \pmod N$, and the DHT becomes

$$H(k_1, k_2)=\sum_{n_1=0}^{L-1}\sum_{n_2=0}^{K-1}x(n_1, n_2)\cdot\left[\operatorname{cas}\left(\frac{2\pi}{L}n_1k_1\right)\operatorname{cas}\left(\frac{2\pi}{K}n_2k_2\right)-2\sin\left(\frac{2\pi}{L}n_1k_1\right)\sin\left(\frac{2\pi}{K}n_2k_2\right)\right]$$

$$0\le k_1\le L-1\quad 0\le k_2\le K-1 \tag{16}$$

where L and K are the lengths of the short transforms over n_1 and n_2, respectively. This expression is not a DHT of the rows of $x(n_1, n_2)$ followed by a transform of the columns, as it would be for the DFT [5]. However, this double sum can be computed using short length DHT's if, after transforming over the first time index, the elements in the $x(k_1, n_2)$ array are replaced by the values

$$\tilde{x}(k_1, n_2)=1/2[x(k_1, n_2)+x(L-k_1, n_2)+x(k_1, K-n_2)-x(L-k_1, K-n_2)]\quad 0\le k_1\le L-1\quad 0\le n_2\le K-1 \tag{17}$$

and followed with short transforms over the remaining time index. This replacement can be computed in-place with eight additions for four output points, or roughly $2N$ adds per stage. This computation can be verified by simply expressing the sequence of operations described above in the form of double sums and showing the equivalence to (16). Another way to arrive at this expression is to consider performing a prime factor DFT on real-valued data using short real-valued DFT blocks, and use linearity of the DFT and the DHT to move the final adds needed to compute the DHT from the real-valued DFT from the output stage to between the two stages. Since the short DHT's also require extra additions, the additions required between stages make the prime factor DHT algorithm less attractive than computation of the DHT from the result of a prime factor real-valued DFT algorithm.

The Winograd–Hartley Transform Algorithm

One way of calculating the short length DFT's used by the prime factor algorithm is with Winograd DFT modules [11], [12]. These modules require the minimum number of multiplications for prime length transforms and can be written as

$$\text{DFT }[x(n)]=\text{CDA }x(n) \tag{18}$$

where C, D, and A are operators on the discrete sequence $x(n)$ [13]. A and C are the pre- and post-weave operators and consist only of additions. The D operator represents a set of multiplications and can be decomposed into its real and imaginary parts by

$$\text{DFT }[x(n)]=C\{D_r+jD_i\}Ax(n)=CD_rAx(n)+jCD_iAx(n). \tag{19}$$

By subtracting the imaginary from the real part of the output the discrete Hartley transform is obtained as discussed previously.

$$\text{DHT }[x(n)]=CD_rAx(n)-CD_iAx(n)=C\{D_r-D_i\}Ax(n) \tag{20}$$

which is exactly the same as (19) with the "$-j$" removed [13]. The DHT module in (20) requires exactly the same number of multiplications [5] but more additions than the DFT in (19), since, unlike $(x + jy)$, the operation $(x + y)$ is an addition.

As shown in [11], the short DFT modules can be nested to give the Winograd–Fourier transform algorithm (WFTA). After nesting, removal of "$-j$" from the imaginary constants yields the Winograd–Hartley transform algorithm (WHTA), again requiring the same number of multiplications and a few more additions than the real-valued WFTA. Programs for the WHTA are easily obtained by modifying the short modules in [5] or the general program in [14]. Operation counts are given in [5].

Comparison of FHT Algorithms

A comparison of operation counts can be used to determine the relative merits of different algorithms for computing the discrete Hartley transform. Table I shows the number of additions and multiplications for the programs in the Appendix. All these programs use 4 real multiplications and 2 real additions per complex multiply. As mentioned earlier, a complex multiplication can be done using 3 real multiplications and 3 real additions. For example, for a radix-2 length 256 FHT, it is possible to reduce the number of multiplications to 963, while the number of additions increases to 3011.

The split-radix uses less multiplications and less additions than the radix-2 and radix-4 and still has a fairly compact code (about the same size as the radix-4), so this will generally be the most advantageous algorithm for power-of-two length DHT's. If the operation counts are compared to counts for real-valued FFT's [5] it is noted that the FHT always requires exactly the same number of multiplications as the corresponding real-valued FFT (the radix-2 FHT and radix-4 FHT programs in the Appendix correspond to real-valued FFT's with 3 butterflies [5] and the split-radix FHT program corresponds to a split-radix real-valued FFT with 5 butterflies [5]). The addition count for the FHT is about N more than the addition count for the real-valued FFT. The FFT generally requires twice the storage of the FHT, unless special symmetries of the real-valued FFT are exploited. Hence the Hartley transform can be considered a convenient way of storing the DFT of a real-valued sequence, at the cost of at most N additions. If the FHT is used for performing convolution, it should be noted that the convolution property in equation (5) exactly corresponds to a complex multiplication where the real and imaginary parts are added to compute the first half, and subtracted to generate the second half. Hence the convolution takes roughly N more additions per transform and $N - 2$ more for the inner product in (5) when computed with the FHT instead of the real-valued FFT. For the special case of a symmetric filter, the extra additions required by the inner product disappear.

Conclusion

By examining fast algorithms for computation of the discrete Hartley transform, this paper showed the close relationship between this transform and the discrete Fourier transform of a real-valued sequence. This similarity was used to develop a corresponding fast Hartley algorithm for every well-known fast DFT algorithm. The radix-2 decimation-in-time fast Hartley transform algorithm, first derived by Bracewell in [2], was analyzed using the index mapping approach and compared to the real-valued radix-2 FFT. This approach also gave rise to a radix-2 decimation-in-frequency and radix-4 FHT's and is easily extended to higher radices. A split-radix algorithm for computation of the DHT was developed, as was a prime factor algorithm using short length Hartley transform modules. The fast Hartley transform algorithms closely resemble their FFT counterparts, and the operation counts for the fast Hartley algorithms exceed that of the real-valued FFT algorithms by only a few additions. The methods developed in this paper allow independent choice of the transform and the algorithm, and these results suggest that current algorithms can perhaps be extended to a wider family of real-valued transforms than previously expected. Use of the FHT instead of the real-valued FFT requires only a few more operations, and in some situations the greater regularity and the equivalence of the forward and inverse discrete Hartley transform may justify this cost.

Appendix

```
CC---------------------------------------------------------------CC
CC                                                               CC
CC      SUBROUTINE FHT2T                                         CC
CC      ----------------                                         CC
CC      Radix-2 decimation in time fast Hartley transform        CC
CC                                                               CC
CC      Input   X       Sequence to be transformed               CC
CC              N,M     Length of sequence N = 2**M              CC
CC      Output  X       Hartley transform                        CC
CC                                                               CC
CC      Authors: D.L. Jones and H.V. Sorensen                    CC
CC               Rice University,   August 5, 1984               CC
CC                                                               CC
CC---------------------------------------------------------------CC
        SUBROUTINE FHT2T(X,N,M)
C
        REAL X(1)
C
C-------Digit reverse counter-----------------------------------C
C
 100    J = 1
        N1 = N - 1
        DO  104  I = 1,N1
            IF (I .GE. J)  GOTO 101
               XT   = X(J)
               X(J) = X(I)
               X(I) = XT
 101        K = N/2
 102        IF (K .GE. J)  GOTO 103
               J = J - K
               K = K/2
               GOTO 102
 103        J = J + K
 104    CONTINUE
C
C-------Main FHT loops------------------------------------------C
C
        DO  10  I = 1,N,2
            XT     = X(I)
            X(I)   = XT + X(I+1)
            X(I+1) = XT - X(I+1)
 10     CONTINUE
C
        N2 = 1
        DO  20  K = 2,M
            N4 = N2
            N2 = N4 + N4
            N1 = N2 + N2
            E = 6.283185307179586/N1
C
            DO  30  J = 1,N,N1
                L2 = J  + N2
                L3 = J  + N4
                L4 = L2 + N4
                XT    = X(J)
                X(J)  = XT + X(L2)
                X(L2) = XT - X(L2)
                XT    = X(L3)
                X(L3) = XT + X(L4)
                X(L4) = XT - X(L4)
                A = E
C
                DO  40  I = 2,N4
                    L1 = J + I - 1
                    L2 = J - I + 1 + N2
                    L3 = L1 + N2
                    L4 = L2 + N2
                    CC1 = COS(A)
                    SS1 = SIN(A)
                    T1 = X(L3)*CC1 + X(L4)*SS1
                    T2 = X(L3)*SS1 - X(L4)*CC1
                    A = I * E
                    XT    = X(L1)
                    X(L1) = XT + T1
                    X(L3) = XT - T1
                    XT    = X(L2)
                    X(L2) = XT + T2
                    X(L4) = XT - T2
 40             CONTINUE
 30         CONTINUE
 20     CONTINUE
C
        RETURN
        END
```

```
CC--------------------------------------------------------------CC
CC                                                              CC
CC      SUBROUTINE FHT2F                                        CC
CC      ----------------                                        CC
CC      Radix-2, decimation in frequency fast Hartley transform CC
CC                                                              CC
CC      Input   X       Sequence to be transformed              CC
CC              N,M     Length of transform N = 2**M            CC
CC      Output  X       Hartley transform                       CC
CC                                                              CC
CC      Authors: D.L. Jones and H.V. Sorensen                   CC
CC               Rice University,   November 13, 1984           CC
CC                                                              CC
CC--------------------------------------------------------------CC
        SUBROUTINE FHT2F(X,N,M)
        REAL X(1)
C
C-------Main FHT loop--------------------------------------------C
C
        N4 = N/2
        M1 = M-1
        DO  10  K = 1,M1
            N4 = N4/2
            N2 = N4 + N4
            N1 = N2 + N2
            E = 6.283185307179586/N1
            DO  20  J = 1,N,N1
                L2 = J  + N2
                L3 = J  + N4
                L4 = L2 + N4
                T1    = X(J)
                X(J)  = T1 + X(L2)
                X(L2) = T1 - X(L2)
                T1    = X(L3)
                X(L3) = T1 + X(L4)
                X(L4) = T1 - X(L4)
                A = E
                DO  30  I = 2,N4
                    L1 = J + I - 1
                    L2 = J - I + 1 + N2
                    L3 = L1 + N2
                    L4 = L2 + N2
                    T1 = X(L1) - X(L3)
                    T2 = X(L2) - X(L4)
                    CC1 = COS(A)
                    SS1 = SIN(A)
                    T3 = T1*CC1 + T2*SS1
                    T4 = T1*SS1 - T2*CC1
                    A = I * E
                    X(L1) = X(L1) + X(L3)
                    X(L2) = X(L2) + X(L4)
                    X(L3) = T3
                    X(L4) = T4
 30             CONTINUE
 20         CONTINUE
 10     CONTINUE
C
        DO  40  I = 1,N,2
            T1     = X(I)
            X(I)   = T1 + X(I+1)
            X(I+1) = T1 - X(I+1)
 40     CONTINUE
C
C-------Digit reverse counter------------------------------------C
C
        Same 14 lines of code as for FHT2T
C
        RETURN
        END
```

```
CC--------------------------------------------------------------CC
CC                                                              CC
CC      Subroutine FHT4T                                        CC
CC      ----------------                                        CC
CC      Radix-4 decimation in time fast Hartley transform       CC
CC                                                              CC
CC      Input   X       Sequence to be transformed              CC
CC              N,M     Length of sequence is N = 4**M          CC
CC      Output  X       Hartley transform                       CC
CC                                                              CC
CC      Authors  H.V. Sorensen and D.L. Jones                   CC
CC               Rice University,  November 10, 1984            CC
CC                                                              CC
CC--------------------------------------------------------------CC
        SUBROUTINE FHT4T(X,N,M)
        REAL X(1)
C
C-------Radix-4 unscrambler--------------------------------------C
C
 100    J = 1
        N1 = N - 1
        DO  104  I = 1,N1
            IF (I.GE.J) GOTO 101
            XT   = X(J)
            X(J) = X(I)
            X(I) = XT
 101        K = N/4
 102        IF (K*3.GE.J) GOTO 103
                J = J - K*3
                K = K/4
                GOTO 102
 103            J = J + K
 104    CONTINUE
C
C-------First stage with length 4 transforms---------------------C
C
        DO  10  I = 1,N,4
            T1 = X(I)   + X(I+1)
            T2 = X(I)   - X(I+1)
            T3 = X(I+2) + X(I+3)
            T4 = X(I+2) - X(I+3)
            X(I)   = T1 + T3
            X(I+1) = T1 - T3
            X(I+2) = T2 + T4
            X(I+3) = T2 - T4
 10     CONTINUE
C
C-------Last M-1 stages------------------------------------------C
C
        N2 = 1
        DO  20  K = 2,M
            N4 = 2 * N2
            N2 = 4 * N2
            N1 = 4 * N2
            E = 6.283185307179586/N1
C
            DO  30  J = 1,N,N1
                L12 = J   + N2
                L13 = L12 + N2
                L14 = L13 + N2
                T1 = X(J)   + X(L12)
                T2 = X(J)   - X(L12)
                T3 = X(L13) + X(L14)
                T4 = X(L13) - X(L14)
                X(J)   = T1 + T3
                X(L12) = T1 - T3
                X(L13) = T2 + T4
                X(L14) = T2 - T4
                L21 = J   + N4
                L22 = L21 + N2
                L23 = L22 + N2
                L24 = L23 + N2
                T1 = X(L21)
                T2 = X(L22)*SQRT(2.0)
                T3 = X(L23)
                T4 = X(L24)*SQRT(2.0)
                X(L21) = T1 + T2 + T3
                X(L22) = T1 - T3 + T4
                X(L23) = T1 - T2 + T3
                X(L24) = T1 - T3 - T4
                A = E
C
                DO  40  I = 2,N4
                    L11 = J + I - 1
                    L12 = L11 + N2
                    L13 = L12 + N2
                    L14 = L13 + N2
                    L21 = J + N2 - I + 1
                    L22 = L21 + N2
                    L23 = L22 + N2
                    L24 = L23 + N2
                    CC1 = COS(A)
                    SS1 = SIN(A)
                    CC2 = COS(2*A)
                    SS2 = SIN(2*A)
                    CC3 = COS(3*A)
                    SS3 = SIN(3*A)
                    A = I * E
                    T12 = X(L12)*CC1 + X(L22)*SS1
                    T13 = X(L13)*CC2 + X(L23)*SS2
                    T14 = X(L14)*CC3 + X(L24)*SS3
                    T22 = X(L12)*SS1 - X(L22)*CC1
                    T23 = X(L13)*SS2 - X(L23)*CC2
                    T24 = X(L14)*SS3 - X(L24)*CC3
                    T1 = X(L21) + T23
                    T2 = X(L21) - T23
                    T3 = T22    + T24
                    T4 = T12    - T14
                    X(L24) = T2 - T3
                    X(L23) = T1 - T4
                    X(L22) = T2 + T3
                    X(L21) = T1 + T4
                    T1 = X(L11) + T13
                    T2 = X(L11) - T13
                    T3 = T24    - T22
                    T4 = T12    + T14
                    X(L14) = T2 - T3
                    X(L13) = T1 - T4
                    X(L12) = T2 + T3
                    X(L11) = T1 + T4
 40             CONTINUE
 30         CONTINUE
 20     CONTINUE
C
        RETURN
        END
```

```
CC--------------------------------------------------------------CC
CC                                                              CC
CC      SUBROUTINE FHTSR                                        CC
CC      ----------------                                        CC
CC      Split radix fast Hartley transform                      CC
CC                                                              CC
CC      Input   X       Sequence to be transformed              CC
CC              N,M     Length of sequense is N = 2 ** M        CC
CC      Output  X       Hartley transform                       CC
CC                                                              CC
CC      Authors: H.V. Sorensen  and  D.L. Jones                 CC
CC               Rice University,    November 24, 1984          CC
CC                                                              CC
CC--------------------------------------------------------------CC
        SUBROUTINE FHTSR(X,N,M)
C
        REAL X(1)
C
C-------First M-1 stages of transform----------------------------C
C
        N2 = 2*N
        DO  10  K = 1,M-1
            IS = 1
            ID = N2
            N2 = N2/2
            N4 = N2/4
            N8 = N2/8
            E = 6.283185307179586/N2
 40         DO  20  I = IS,N,ID
                L12 = I   + N4
                L13 = L12 + N4
                L14 = L13 + N4
                T1 = X(L12) - X(L14)
                T2 = X(I) + X(L13)
                T3 = X(L12) + X(L14)
                T4 = X(I) - X(L13)
                X(L14) = T4 - T1
                X(L13) = T4 + T1
```

```
                    X(L12) = T3
                    X(I)   = T2
                    IF (N4 .EQ. 1) GOTO 20
                    L21 = I + N8
                    L22 = L21 + N4
                    L23 = L22 + N4
                    L24 = L23 + N4
                    T1 = X(L22)
                    T2 = X(L23)
                    T3 = X(L24)
                    X(L24) = ( T1     - T3 )*SQRT(2.0)
                    X(L23) = ( X(L21) - T2 )*SQRT(2.0)
                    X(L22) =   T1     + T3
                    X(L21) =   X(L21) + T2
                    A = E

                DO  30  J = 2,N8
                    L11 = I   + J - 1
                    L12 = L11 + N4
                    L13 = L12 + N4
                    L14 = L13 + N4
                    L21 = I + N4 - J + 1
                    L22 = L21 + N4
                    L23 = L22 + N4
                    L24 = L23 + N4
                    A3 = 3*A
                    CC1 = COS(A)
                    SS1 = SIN(A)
                    CC3 = COS(A3)
                    SS3 = SIN(A3)
                    A = J * E
                    T5 = X(L21) - X(L23)
                    T2 = X(L11) - X(L13)
                    T1 = T2 + T5
                    T2 = T2 - T5
                    T5 = X(L22) - X(L24)
                    T4 = X(L14) - X(L12)
                    T3 = T4 + T5
                    T4 = T4 - T5
                    X(L11) = X(L11) + X(L13)
                    X(L12) = X(L12) + X(L14)
                    X(L21) = X(L21) + X(L23)
                    X(L22) = X(L22) + X(L24)
                    X(L13) = T1 * CC1 + T3 * SS1
                    X(L14) = T2 * CC3 - T4 * SS3
                    X(L23) = T1 * SS1 - T3 * CC1
                    X(L24) = T2 * SS3 + T4 * CC3
 30             CONTINUE
 20         CONTINUE
                IS = 2*ID - N2 + 1
                ID = 4*ID
            IF (IS .LT. N) GOTO 40
 10     CONTINUE
C
C-------Last stage of transform---------------------------------C
C
        IS = 1
        ID = 4
 50     DO  60  I = IS,N,ID
            T1 = X(I)
            X(I)   = T1 + X(I+1)
            X(I+1) = T1 - X(I+1)
 60     CONTINUE
            IS = 2*ID - 1
            ID = 4*ID
        IF (IS .LT. N) GOTO 50
C
C-------Digit reverse counter-----------------------------------C
C
        Same 14 lines of code as for the FHT2T
C
        RETURN
        END
```

Acknowledgment

The authors would like to thank H. W. Johnson for initiating our interest in this area and Prof. Bracewell for his careful review.

References

[1] R. N. Bracewell, "Discrete Hartley transform," *J. Opt. Soc. Amer.*, vol. 73, no. 12, pp. 1832–1835, Dec. 1983.

[2] ——, "The fast Hartley transform," *Proc. IEEE*, vol. 72, no. 8, pp. 1010–1018, Aug. 1984.

[3] ——, *The Hartley Transforms*. England: Oxford University Press, 1985.

[4] C. S. Burrus, "Index mappings for multidimensional formulation of the DFT and convolution," *IEEE Trans. Acoust., Speech, Signal Processing*, vol. ASSP-25, pp. 239–242, June 1977.

[5] C. S. Burrus and T. W. Parks, *DFT/FFT and Convolution Algorithms*. New York: Wiley, 1985.

[6] A. V. Oppenheim and R. W. Schafer, *Digital Signal Processing*. Englewood Cliffs, NJ: Prentice-Hall, 1975.

[7] J. W. Cooley and J. W. Tukey, "An algorithm for the machine calculation of complex Fourier series," *Math. Comput.*, vol. 19, no. 2, pp. 297–301, Apr. 1965.

[8] O. Buneman, "Conversion of FFT's to fast Hartley transforms," to appear in *SIAM J. Sci. Stat. Comput.*

[9] G. D. Bergland, "A fast Fourier transform algorithm for real-valued series," *Commun. ACM*, vol. 11, no. 10, pp. 703–710, Oct. 1968.

[10] P. Duhamel and H. Hollmann, "'Split radix' FFT algorithm," *Electron. Lett.*, vol. 20, no. 1, pp. 14–16, Jan. 5, 1984.

[11] S. Winograd, "On computing the discrete Fourier transform," *Mathematics of Computation*, vol. 32, no. 141, pp. 175–199, (reprinted in McClellan and Rader: Number Theory in DPS), Jan. 1978.

[12] H. J. Nussbaumer, *Fast Fourier Transform and Convolution Algorithms*. Berlin, Heidelberg, and New York: Springer-Verlag, 1981.

[13] M. T. Heideman, C. S. Burrus, and H. W. Johnson, "Prime factor FFT algorithms for real-valued series," in *Proc. 1984 IEEE Int. Conf. Acoust., Speech, Signal Processing*, San Diego, CA, Mar. 19–21, 1984, pp. 28A.7.1-28A.7.4.

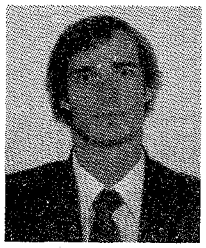
[14] J. H . McClellan and C. M. Rader, *Number Theory in Digital Signal Processing*. Englewood Cliffs, NJ: Prentice-Hall, 1979.

[15] R. Ansari, "An extension of the discrete Fourier transforms," *IEEE Trans. Circuits Syst.*, vol. 32, pp. 618–619, June 1985.

Henrik V. Sorensen (S'84) was born in Skanderborg, Denmark, on January 17, 1959. He received the B.S. and M.S. degrees in electrical engineering from Aalborg University Center, Aalborg, Denmark, in 1981 and 1983, respectively.

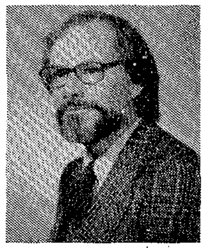
He is presently working as a doctoral student in electrical engineering at Rice University, Houston, TX. His professional interests are in the area of digital signal processing.

Mr. Sorensen is a member of DIF, the Danish Engineering Society.



Douglas L. Jones (S'81) was born in Dallas, TX, on January 17, 1961. He received the B.S.E.E. degree (summa cum laude) from Rice University Houston, TX, in 1983.

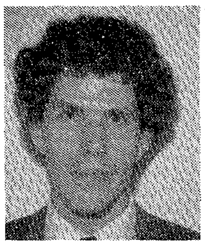
He is currently engaged in graduate study at Rice University under a National Science Foundation Graduate Fellowship. His research interests are in digital signal processing and time-varying signal analysis.



C. Sidney Burrus (S'55–M'61–SM'75–F'81) was born in Abilene, TX, on October 9, 1934. He received the B.A., B.S.E.E., and M.S. degrees from Rice University, Houston, TX, in 1957, 1958, and 1960 respectively, and the Ph.D. degree from Stanford University, Stanford, CA, in 1965.

From 1960 to 1962, he served in the Navy Reserve teaching at the Nuclear Power School in New London, CT, and during the summers of 1964 and 1965, he was a Lecturer in Electrical Engineering at Stanford University. In 1965, he joined Rice University, where he is now Professor and Chairman of Electrical and Computer Engineering. From 1968 to 1975, he was a visiting faculty member at Baylor College of Medicine and a Consultant at the VA Hospital Resarch Center, Houston, TX. From 1972 to 1978, he was Master of Lovett College at Rice University and in 1975, and again in 1979, he was a Visiting Professor at the Institut für Nachrichtentechnik, Universität Erlangen-Nürnberg, West Germany. During the summer of 1984 he was a Visiting Fellow at Trinity College, Cambridge University.

Dr. Burrus is a member of Tau Beta Pi and Sigma Xi. He has received teaching awards from Rice in 1969, 1974, 1975, 1976, and 1980, received an IEEE S-ASSP Senior Award in 1974, a Senior Alexander von Humboldt Award in 1975, and a Senior Fulbright Fellowship in 1979.



Michael T. Heideman (S'85) was born in Medford, OR, on November 13, 1956. He received the B.S. and M.S. degrees in electrical engineering from Stanford University in 1978 and 1980, respectively.

From 1978 to 1981, he was a Member of the Technical Staff of Lockheed Missiles and Space Company, Palo Alto, CA, conducting research in image processing. From 1981 to the present, he has been with Lockheed Engineering and Management Services Company, Houston, TX, where he is involved in the development of radar cross-section models for the space shuttle program. He is completing work on a Ph.D. degree in electrical and computer engineering at Rice University, Houston, TX, specializing in fast signal processing algorithms.