Improved Fourier and Hartley Transform Algorithms: Application to Cyclic Convolution of Real Data

PIERRE DUHAMEL, MEMBER, IEEE, AND MARTIN VETTERLI, MEMBER, IEEE

Abstract—**This paper highlights the possible tradeoffs between arithmetic and structural complexity when computing cyclic convolution of real data in the transform domain.**

Both Fourier and Hartley-based schemes are first explained in their usual form and then improved, either from the structural point of view or in the number of operations involved.

Namely, we first present an algorithm for the in-place computation of the discrete Fourier transform on real data: a decimation-in-time split-radix algorithm, more compact than the previously published one. Second, we present a new fast Hartley transform algorithm with a reduced number of operations.

A more regular convolution scheme based on FFT's is also proposed.

Finally, we show that Hartley transforms belong to a larger class of algorithms characterized by their "generalized" convolution property.

I. Introduction

THE radix-2 fast Fourier transform algorithm was first explained by Cooley and Tukey in 1965 in a version on complex data [1]. But, as most of the data to be treated are real instead of complex, the need for a version of the FFT on real data removing extra calculations from the complex case became soon apparent, and such a program was quickly published by Bergland [3].

In fact, this program does not seem to have been widely used, and many people preferred either to compute a length 2^n real DFT either by the use of a length 2^{n-1} complex FFT plus some additional operations, or to compute two real FFT's at a time by using one length 2^n complex FFT.

Let us also notice that a version of the radix-2 FFT algorithm was also published in [4] for symmetric real-values series, and that the algorithm by Preuss [2] can be straightforwardly applied to real and real symmetric data.

More recently, the 2^n FFT algorithms with the minimum known number of both multiplications and additions were also proposed in a real-data version [6], [7], [10], [11].

At about the same time, Bracewell [8] proposed the use of fast discrete Hartley transforms (FDHT), which are real in nature, as a substitute to the FFT for computing cyclic convolutions of real data.

Manuscript received March 14, 1986; revised December 10, 1986.

P. Duhamel is with CNET/PAB/RPE, 92131 Issy-les-Moulineaux, France.

M. Vetterli was with the Ecole Polytechnique Fédérale de Lausanne, CH-1007 Lausanne, Switzerland. He is now with the Center for Telecommunications Research, Columbia University, New York, NY 10027.

IEEE Log Number 8613866.

But it was not clear at all what approach was best suited for cyclic convolutions of real sequences when taking into account the structure of the algorithm as well as its computational complexity.

Nevertheless, the well-known approach of computing real convolutions through FFT algorithms on complex data will not be considered here, and we shall only consider programs specially designed for real data since they obtain the lowest known number of arithmetic operations without an increase in program length.

In Section II, we shall first briefly present the two basic approaches considered here: convolution using FFT's and convolution using FDHT's. This comparison shows that the former has some advantages when considering arithmetic complexity, while the latter has a simpler structure since the DHT is self-inverse.

Arithmetic complexities for these cyclic convolution schemes are given when the FFT and FDHT are computed by the algorithms with the lowest known number of both multiplications and additions.

Sections III and IV describe improved FFT and FDHT programs, together with a more regular FFT-based convolution scheme.

First, a decimation-in-time split-radix FFT algorithm on real data is presented, with a more regular structure and a more compact code than the previously published DIF one (it has exactly the same arithmetic complexity).

Section IV presents an improved FDHT algorithm, allowing us to obtain any length -2^n FDHT using only two additions more than the best FFT algorithms known for real data (four additions more for the whole convolution). In this section, we also show how to derive improved FDHT algorithms from their FFT counterparts.

Finally, Section IV discusses a generalized convolution property, which includes the various approaches presented in this paper.

II. The Initial Schemes

A. Convolution Using FFT's

Let $\{X_k^f\}$ be the DFT of $\{x_n\}$:

$$X_k^f = \sum_{n=0}^{N-1} x_n \left(\cos \frac{2\pi nk}{N} - j \sin \frac{2\pi nk}{N} \right) = \sum_{n=0}^{N-1} x_n W_N^{nk}. \quad (1)$$

0096-3518/87/0600-0818$01.00 © 1987 IEEE

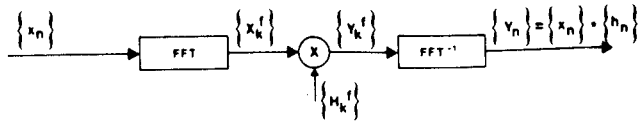

Fig. 1. Usual convolution scheme using FFT's.

TABLE I
NUMBER OF ARITHMETIC OPERATIONS FOR A CYCLIC CONVOLUTION ON REAL DATA

		Number of Additions			
N	Number of Mult.	Initial FFT-Based	Initial FDHT-Based	FFT-Based with 2 Forward FFT's	Improved FDHT-Based
8	15	49	53	55	53
16	43	141	153	155	145
32	115	373	393	403	377
64	291	933	977	995	937
128	707	2245	2329	2371	2249
256	1667	5253	5425	5507	5257
512	3843	12 037	12 377	12 547	12 041
1024	8707	27 141	27 825	28 163	27 145
2048	19 459	60 421	61 785	62 467	60 425

Since the DFT has the convolution property, the convolution scheme has the structure given in Fig. 1.

When the initial sequences $\{x_n\}$ and $\{h_n\}$ are real, $\{X_k\}$, $\{H_k\}$, and $\{Y_k\}$ have Hermitian symmetry: $Y_k = Y^*_{-k}$, and the forward DFT has to be performed on real data, while the inverse FFT has to be performed on data with Hermitian symmetry. Both kind of FFT's have been published [7], [11]. They have the same arithmetic complexity, and can be performed in place. (In fact, an FFT on data with Hermitian symmetry can be derived by reversing the flow graph of an FFT on real data.)

If the FFT's are performed with the lowest arithmetic complexity (lowest number of both additions and multiplications) using FFCT [6] or split-radix [7] algorithms, it leads to the following number of operations:

$$M_n^{\text{conv. FFT}} = 2M_n^{fr} + 2 + 3 \cdot (2^{n-1} - 1) = 2^{n-1}(2n - 3) + 3 \quad (2)$$

$$A_n^{\text{conv. FFT}} = 2A_n^{fr} + 3(2^{n-1} - 1) = 2^{n-1}(6n - 7) + 5 \quad (3)$$

where M_n^{fr} (respectively, A_n^{fr}) is the number of multiplications (respectively, additions) needed to compute a length 2^n DFT on real data.

In these operation counts, it has been taken into account that, due to the symmetry of both X_k^f and H_k^f, Y_k^f can be obtained with two real multiplications for $k = 0$ and $k = N/2$, and $N/2 - 1$ complex multiplications (three real mults + three real adds each) for $k = 1, \cdots, N/2 - 1$.

This arithmetic complexity is rather low, as shown in Table I, since it is based on a powerful FFT algorithm for real data.

Let us recall here that both split-radix and FFCT algorithms meet the minimum possible number of nontrivial real multiplications up to and including length $N = 16$, and the minimum possible number of nontrivial complex multiplications (i.e., $\neq j, \pm 1$) up to and including length $N = 64$, while possessing the lowest known number of additions needed to compute a length $N = 2^n$ DFT.

Nevertheless, a disadvantage of this scheme is that it needs both forward transform on real data and inverse transform on complex data with Hermitian symmetry. This results in an increase in program length for the cyclic convolution.

B. Convolution Using FDHT's

The discrete Hartley transform is defined as the sum of the real and imaginary part of the Fourier transform:

$$X_k^h = \sum_{n=0}^{N-1} x_n \left(\cos \frac{2\pi nk}{N} + \sin \frac{2\pi nk}{N} \right). \quad (4)$$

As can be seen from the definition, the DHT is its own inverse, and it is a real transform. Furthermore, the convolution in the time domain corresponds to the following operation in the Hartley domain:

$$Y_k^h = X_k^h \cdot H_e^h(k) + X_{-k}^h \cdot H_0^h(k) \quad (5)$$

where $H_e^h(k)$ [respectively, $H_0^h(k)$] is the even (respectively, odd) part of the Hartley transform of $\{h_n\}$.

So, the convolution scheme using Hartley transforms becomes as shown in Fig. 2.

When considering (5), the "multiplications" in the Hartley domain seem to need two multiplications per point, but it it easy to see that by grouping the computation of Y_k^h and Y_{-k}^h, this number can be reduced to $3/2$ multiplications per point:

$$\begin{bmatrix} Y_k^h \\ Y_{-k}^h \end{bmatrix} = \begin{bmatrix} H_e(k) & H_0(k) \\ -H_0(k) & H_e(k) \end{bmatrix} \begin{bmatrix} X_k^h \\ X_{-k}^h \end{bmatrix} \quad (6)$$

and (6) is easily recognized as a complex multiplication, needing only three real multiplications and three additions.

If we consider also that for $k = 0$ and $N/2$, (6) reduces

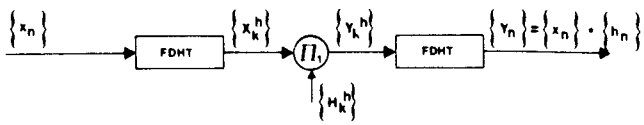

Fig. 2. Convolution scheme based on FDHT's.

to

$$Y_0^h = X_0^h \quad H_e(0)$$
$$Y_{N/2}^h = X_{N/2}^h \quad H_e(N/2) \tag{7}$$

one can see that the set of "multiplications" in the Hartley domain needs exactly the same number of operations as in the Fourier domain.

The remaining part of the scheme given in Fig. 2 to be evaluated consists of the FDHT.

FDHT was first introduced by Bracewell [8] in a radix-2, decimation-in-time version, and Sorensen *et al.* [9] recently proposed different types of FDHT algorithms. (See also [15].)

Inspection of these algorithms allows the following conclusions.

A radix-2 FDHT algorithm needs $N - 2$ additions more than the corresponding FFT algorithm on real data and the same number of multiplications.

The split-radix algorithm, which seems already to be one of the best compromises for real-data FFT's (lowest known number of operations, in-place computation, compact program), also gives the lowest known number of operations for the FDHT, but requires $2(2^{n-1} - (-1)^{n-1}/3)$ additions more than the corresponding split-radix FFT algorithm on real data.

If the FDHT in Fig. 2 is computed through a split-radix algorithm, this gives the following arithmetic complexity for the cyclic convolution:

$$M_n^{\text{conv. DHT}} = 2^{n-1}(2n - 3) + 3 \tag{8}$$

$$A_n^{\text{conv. DHT}} = 2^{n-1}(6n - 7) + 5 + 4\,\frac{2^{n-1} - (-1)^{n-1}}{3}$$
$$= 2^{n-1}\left(6n - \frac{17}{3}\right) + 5 - \frac{4}{3}(-1)^{n-1}. \tag{9}$$

One can see that convolution using FDHT's will have a simpler structure, even if the FDHT programs have the same code length as FFT programs, due to the self-inverse property of the Hartley transform. This simplicity will be obtained at the cost of a small increase of the number of additions.

III. Improved FFT-Based Convolution Schemes

These initial schemes described in Section II can be improved in two ways: either by an improvement of the basic algorithm (FFT on real data, FDHT) or by an improvement of the structure of the convolution scheme. The FFT-based convolution scheme is relevant to both improvements.

A. A DIT Split-Radix FFT Algorithm on Real Data

The decimation-in-frequency (DIF) split-radix decomposition applies a radix-2 decomposition to the even-indexed samples, and a radix-4 decomposition to the odd-indexed samples of the transform $\{X_k\}$. The decimation-in-time (DIT) split-radix decomposition of the DFT, being the dual algorithm of the DIF SRFFT, considers separately the even-indexed samples $\{x_{2n}\}$, the samples $\equiv 1 \pmod 4$, $\{x_{4n+1}\}$, and the samples $\equiv 3 \pmod 4$: $\{x_{4n+3}\}$. This DIT decomposition is given in (10).

$$X_k^f = \sum_{k=0}^{N/2-1} x_{2n} W_N^{2nk} + W_N^k \sum_{n=0}^{N/4-1} x_{4n+1} W_N^{4nk} + W_N^{3k} \sum_{n=0}^{N/4-1} x_{4n+3} W_N^{4nk}. \tag{10}$$

When the initial sequence $\{x_n\}$ is real, (10) involves DFT's of real data only. Furthermore, since these DFT's have Hermitian symmetry, it is possible to work "in place" by using, for a DFT of length L, the $L/2 + 1$ first locations for the real part of the corresponding samples of the transform, and the $L/2 - 1$ remaining ones for the imaginary part.

When two butterflies are done at the same time, this algorithm requires exactly the same number of operations as the DIF split-radix FFT on real data [7] and the FFCT algorithm [6], [11], but requires only four different types of butterflies to meet the minimum number of operations, thus resulting in a more compact program. This has to be compared to the eight different butterflies needed by the DIF SRFFT [7] and the absence of compact looped programs for implementing FFCT [6].

The Fortran code corresponding to this algorithm is given in Appendix I. This program uses the indexing scheme given by Sorensen *et al.* in [12], which gives the usual three-loop structures of radix-2 or radix-4 programs.

It should be remarked that this program was written to be as compact as possible, and that a much faster one could be obtained by following the approach that was used in the complex case [7].

Since run-time comparisons on general-purpose machines are very machine-dependent (and even compiler-dependent), timings will not be given here. Nevertheless, precise timings are more important on DSP's: we could obtain run times as low as 235 μs for a length-64 real data FFT on a TMS 32010 (TI's benchmarks are 400.2 μs) and 572 μs for a length-128 real-data FFT (versus 955 ms for TI's benchmarks).

B. A More Regular Convolution Scheme Using FFT's

Even if the FFT program is compact and effective, the convolution scheme of Fig. 1 still requires two different FFT programs: a forward FFT on real data, and an inverse FFT on data with Hermitian symmetry.

We shall now propose a new scheme using two forward

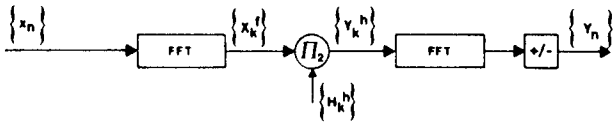

Fig. 3. Modified FFT-based convolution scheme using only forward FFT's on real data.

FFT's on real data, at the cost of a few more additions: this improvement is based on some of the remarks made when studying FDHT's.

The convolution property in the Hartley domain, as given in (5), can be rewritten as follows, by reversing the roles of X_k and H_k:

$$Y_k^h = X_e^h(k) \cdot H_0^h(k) H_{-k}^h. \tag{11}$$

But $X_e^h(k)$, being the even part of the Hartley transform of $\{x_n\}$, is also the real part of the DFT of $\{x_n\}$, and $X_0^h(k)$ is the corresponding imaginary part.

We can then rewrite (6) as follows:

$$\begin{bmatrix} Y_k^h \\ Y_{-k}^h \end{bmatrix} = \begin{bmatrix} H_k^h & H_{-k}^h \\ H_{-k}^h & -H_k^h \end{bmatrix} \begin{bmatrix} \text{Re } X_k^f \\ \text{Im } X_k^f \end{bmatrix} \tag{12}$$

which is also equivalent to a complex product.

Furthermore, since a DHT is the sum of the real and imaginary part of a DFT, the last FDHT in Fig. 2 can be replaced by an FFT followed by the sum of the real and imaginary part of each DFT component, thus resulting in the diagram of Fig. 3.

This scheme requires exactly the same number of operations as the usual convolution scheme based on FFT's, plus $N - 2$ additions due to the $+/-$ box:

$$A_n^{\text{conv. fft}} = 2^{n-1}(6n - 5) + 3. \tag{13}$$

This structure is now as regular as the one using FDHT's, to the price of about $2^n/3$ extra additions compared to the FDHT scheme. It has the advantage of using only usual FFT algorithms on real data, which should become widely used due to their simplicity and compactness.

IV. Improved FDHT Algorithm

As can be seen from Section II-B, DHT's and DFT's are very simply related. In fact, FDHT algorithms, as found in [8] and [9], can be improved by making full use of the close relationship between Fourier and Hartley transforms. In fact, nearly all the extra additions needed to compute FDHT's instead of FFT's can be nested inside the twiddle factors of the FFT, thus resulting in hybrid FDHT/FFT algorithms.

By multiplying the DFT of $\{x_n\}$ by $(1 + j)$, we obtain (14):

$$\begin{aligned}(1 + j)X_k^f &= (\text{Re } X_k^f - \text{Im } X_k^f) + j(\text{Re } X_k^f + \text{Im } X_k^f) \\ &= X_k^h + jX_{-k}^h \end{aligned} \tag{14}$$

which means that multiplication of X_k^f by $(1 + j)$ gives us the Hartley transform of $\{x_n\}$.

Let us apply this remark to the basic split-radix decimation-in-time decomposition of the DFT [10] as given in (10):

$$\begin{aligned}(1 + j) \cdot X_k^f = X_k^h + jX_{-k}^h = (1 + j) &\sum_{n=0}^{N/2-1} x_{2n} W_N^{2nk} \\ &+ (1 + j) W_N^k \sum_{n=0}^{N/4-1} x_{4n+1} W_N^{4nk} \\ &+ (1 + j) W_N^{3k} \sum_{n=0}^{N/4-1} x_{4n+3} W_N^{4nk} \end{aligned} \tag{15}$$

and with the following notations

$$\begin{aligned} \{X_k^{2h}\} &\triangleq \text{DHT of } \{x_{2n}\} \\ \{X_k^{1f}\} &\triangleq \text{DFT of } \{x_{4n+1}\} \\ \{X_k^{3f}\} &\triangleq \text{DFT of } \{x_{4n+3}\} \end{aligned}$$

(15) can now be rewritten as follows;

$$\begin{aligned} X_k^h + jX_{-k}^h = X_k^{2h} &+ jX_{-k}^{2h} \\ &+ [(1 + j) \cdot W_N^k] X_k^{1f} \\ &+ [(1 + j) W_N^{3k}] X_k^{3f}. \end{aligned} \tag{16}$$

Equation (16) is now the basic recursion of the improved FDHT algorithm: it transforms the length N DHT into a length $N/2$ DHT plus two length $N/4$ DFT's, and some multiplications by twiddle factors of the form $(1 + j) W_N^k$ or $(1 + j) W_N^{3k}$.

As in the case of FFT's on real data, the butterflies with twiddle factors W_N^k have to be computed together with the ones with twiddle factors W_N^{-k}. This needs $(N/8 - 1)$ general 6 mult–18 adds butterflies, plus one 0 mult–6 adds butterfly (corresponding to $k = 0$) and one 2 mult–4 adds butterfly (corresponding to $k = N/8$).

Iteration of these addition counts gives exactly the same number as that obtained for FFT's on real data, except for the FDHT of length 4 which needs two extra additions.

This leads to

$$A_n^{\text{hart}} = 2^{n-1}(3 \cdot n - 5) + 6 \tag{17}$$

and the whole convolution with this algorithm will need

$$A_n^{\text{conv hart.}} = 2^{n-1}(6n - 7) + 9. \tag{18}$$

The number of multiplications is the same as for the other schemes.

The convolution scheme using this new fast Hartley transform algorithm now has the same number of multiplications and four additions more than the best known scheme, with the advantage of using only a single DHT program to be used twice.

This result is interesting theoretically since it is another example, besides [6], that uses a decomposition of a transform into smaller transforms of another type to achieve a reduction in arithmetic complexity.

Furthermore, the process used for deriving this new algorithm has interesting characteristics: it shows how the

additional operations used to convert transforms of one type (FFT's) into transforms of another type (FDHT's) can be nested inside existing algorithm, and hence can almost disappear. This may be useful for other transforms as well.

Note also that a reverse procedure is also of interest since the same procedure can be used to convert FDHT algorithms into FFT algorithms; this shows that it is hopeless to find FDHT algorithms requiring fewer arithmetic operations than the corresponding FFT algorithms. In fact, if there was a possibility of finding improved FDHT algorithms, they could be used as a starting point for deriving improved FFT algorithms using the technique described above. We have thus demonstrated that the only improvement that can be expected from FDHT's is a more regular structure (self-inverse property), a fact which was becoming more and more apparent from the recent literature.

From a practical point of view, it should be emphasized that this new algorithm brings only a slight improvement ($N/3$ additions) over the split-radix FDHT algorithm proposed in [9]. Nevertheless, a Fortran implementation is given in Appendix II for the purpose of providing the precise computation of the "butterflies" converting the FFT into an FDHT. It should be clear from what has been stated above that the improvement in the number of additions would produce an increase in the computational speed only when in-line code is used. This happens when programming DSP's or when using autogen techniques [16]. Otherwise, the program given in Appendix II will certainly be slower than the one in [9].

V. The "Generalized" Convolution Property

In fact, the existence of FDHT, which is a real-valued transform possessing some kind of convolution property, states a theoretical problem since it has been shown in [13] that real transforms having a convolution property could not exist.

The usual convolution property is defined as follows:

$$\text{let } y_n = \{x_n\} * \{h_n\} = \sum_{i=0}^{N-1} x_i \otimes h_{n-i}. \qquad (19)$$

The considered transform T will have the so-called convolution property if

$$T\{y_n\} = T\{x_n\} \otimes T\{h_n\} \qquad (20)$$

where $\otimes$ is the usual term-by-term multiplication. In other words, the convolution product in the time domain becomes a usual multiplication in the transform domain.

But it was shown in [13] that the existence of "square" transforms having the convolution property depends only on the existence of an α that is a root of unity of order N and on the existence of N^{-1}. In the case of the DFT, $\alpha = W_N$, as defined in (1), but the Hartley transform is not such a transform, although it has some kind of convolution property. (In fact, it was also shown that a real transform having the convolution property could not exist for $N > 2$.)

Nevertheless, this contradiction is only apparent since the "multiplication" in the Hartley domain is not the same one as is defined in (19).

The FDHT belongs then to the larger class of transforms having the "generalized" convolution property

$$\text{let } y_n = x_n * h_n = \sum_{i=0}^{N-1} x_i \otimes h_{n-i}; \qquad (21)$$

then

$$T'\{y_n\} = T'\{x_n\} \circledast T'\{h_n\} \qquad (22)$$

where $\circledast$ is allowed to be different from the usual term-by-term multiplication.

In fact, Ansari [14] has already shown that the discrete Hartley transform is not the only combination of the DFT coefficients having the generalized cyclic convolution property.

Further research of other members of this class of transforms should be of interest.

VI. Conclusion

In this paper, we have first presented two usual schemes for cyclic convolution of real signals via FFT or FDHT. Operation counts are given for both of them using the fastest known algorithms, showing that the FFT-based convolution has a lower arithmetic complexity, but a more complex structure.

Improvements are then made to both schemes since we propose

- an improved FFT algorithm on real data with increased regularity and compactness
- an improved FDHT algorithm which uses less operations than the previously proposed algorithms
- an FFT-based scheme using two forward transforms on real data
- an FDHT-based algorithm with reduced number of additions.

All operation counts are summarized in Table I, allowing one to choose the best tradeoff between low arithmetic complexity and structural complexity.

As can be seen from Table I, the difference between the arithmetic complexities involved are so small that they will not have a strong influence on the timings of the different algorithms. Furthermore, the FDHT-based schemes require the use of a special-purpose program for computing the FDHT of the input signal that will be used only for the computation of cyclic convolutions since the Hartley transform has no physical significance in itself.

These are the reasons why we believe that, in many circumstances, the scheme using two forward FFT's should be used since it provides a good compromise between structural complexity, arithmetic complexity, and is based on an FFT program that should become widely used, due to its low arithmetic complexity and regular structure.

APPENDIX I
SPLIT-RADIX, DECIMATION-IN-TIME FAST FOURIER TRANSFORM

$n = 2^m$: length of the transform.
x must be in natural order on input.

On output, x_k will contain the real part of X_k, $k = 1, \cdots, n/2 + 1$ and the imaginary part of X_k, $k = n/2, \cdots, n$.

This program uses the usual 4 mults–2 adds complex multiplication algorithm.

```
      subroutine splitdit(x,n,m)
      dimension x(1)
      data rac2s2/0.707106778/
c
c-----------------digit reverse counter----------
c
 1    j = 1
      n1 = n-1
      do 5 i=1,n1
      if(i.ge.j) go to 2
      xt = x(j)
      x(j) = x(i)
      x(i) = xt
 2    k = n/2
 3    if(k.ge.j) go to 4
      j = j-k
      k = k/2
      go to 3
 4    j = j+k
 5    continue
c
c-----------------length two transforms-----------
c
      is = 1
      id = 4
 10   do 20 i0=is,n,id
      i1 = i0+1
      r1 = x(i0)
      x(i0) = r1+x(i1)
      x(i1) = r1-x(i1)
 20   continue
      is = 2*id-1
      id = 4*id
      if(is.lt.n) go to 10
c
c-----------------other butterflies-------------
c
      n2 = 2
      do 100 k=2,m
      n2 = n2*2
      n4 = n2/4
c---------------without mult-------------
      is = 1
      id = 2*n2
 30   do 40 i0=is,n-1,id
      i1 = i0+n4
      i2 = i1+n4
      i3 = i2+n4
      t0 = x(i2)+x(i3)
      x(i3) = x(i2)-x(i3)
      x(i2) = x(i0)-t0
      x(i0) = x(i0)+t0
 40   continue
      is = 2*id-n2+1
      id = 4*id
      if(is.lt.n) go to 30
      if(n4.lt.2) go to 100
c----------------with 2 real mult
      is = n4/2+1
      id = 2*n2
 50   do 60 i0=is,n-1,id
      i1 = i0+n4
      i2 = i1+n4
      i3 = i2+n4
      t1 = (x(i2)-x(i3))*rac2s2
      t2 = (x(i2)+x(i3))*rac2s2
      x(i2) = t2-x(i1)
      x(i3) = t2+x(i1)
      x(i1) = x(i0)-t1
      x(i0) = x(i0)+t1
 60   continue
      is = 2*id-n2+n4/2+1
      id = 4*id
      if(is.lt.n) go to 50
      e = 6.283185307179586/n2
      a = e
      if(n4.lt.4) go to 100
      do 90 j=2,n4/2
      a3 = 3.*a
      cc1 = cos(a)
      ss1 = sin(a)
      cc3 = cos(a3)
      ss3 = sin(a3)
      a = j*e
      is = j
      id = 2*n2
 70   do 80 i0=is,n-1,id
c-----------------with 6 real mult-------------
      i1 = i0+n4
      i2 = i1-j-j+2
      i0 = i1+n4
      i2 = i0+n4
      i3 = i2+n4
      i0 = i0+n4
      i3 = i3+n4
      c2 = x(i2)*cc1-x(i3)*ss1
      d2 = -(x(i2)*ss1+x(i3)*cc1)
      c3 = x(i3)*cc3-x(i3)*ss3
      d3 = -(x(i3)*ss3+x(i3)*cc3)
      t1 = c2+c3
      c3 = c2-c3
      t2 = d2-d3
      d3 = d2+d3
      x(i2) = -x(i0)-d3
      x(i3) = -x(i1)+c3
      x(i3) = x(i1)+c3
      x(i3) = x(i0)-d3
      x(i1) = x(i1)+t2
      x(i0) = x(i0)-t1
      x(i0) = x(i0)+t1
      x(i1) = x(i1)-t2
c
c
 80   continue
      is = 2*id-n2+j
      id = 4*id
      if(is.lt.n) go to 70
 90   continue
 100  continue
c
      return
      end
```

APPENDIX II
PARTS OF PROGRAM TO BE INSERTED IN THE DIT SPLIT-RADIX FFT PROGRAM TO OBTAIN AN FDHT PROGRAM

1) To be inserted after line 48.

```
      if(i0.ne.1) go to 35
      t0 = x(i2)+x(i3)
      t1 = x(i2)-x(i3)
      x(i2) = x(i0)-t0
      x(i0) = x(i0)+t0
      x(i3) = x(i1)-t1
      x(i1) = x(i1)+t1
      go to 40
 35   continue
```

2) To be inserted after line 64.

```
      if(i0.ne.(n4/2+1)) go to 55
      t1 = x(i2)*rac2
      t2 = x(i3)*rac2
      x(i3) = x(i1)-t2
      x(i1) = x(i1)+t2
      x(i2) = x(i0)-t1
      x(i0) = x(i0)+t1
      go to 60
 55   continue
```

3) To be inserted after line 83.

```
      cps1 = cc1+ss1
      cps3 = cc3+ss3
      cms1 = cc1-ss1
      cms3 = cc3-ss3
```

4) To be inserted after line 95.

```
      if(ia0.ne.j) go to 75
      c2 = x(ia2)*cps1+x(ib2)*cms1
      d2 = x(ia2)*cms1-x(ib2)*cps1
      c3 = x(ia3)*cps3+x(ib3)*cms3
      d3 = x(ia3)*cms3-x(ib3)*cps3
      t1 = c2+c3
      c3 = c2-c3
      t2 = d2-d3
      d3 = d2+d3
      x(ib2) = x(ia1)-c3
      x(ia1) = x(ia1)+c3
      x(ia3) = x(ib1)-t2
      x(ib1) = x(ib1)+t2
      x(ib3) = x(ib0)+d3
      x(ib0) = x(ib0)-d3
      x(ia2) = x(ia0)-t1
      x(ia0) = x(ia0)+t1
      go to 80
 75   continue
```

ACKNOWLEDGMENT

The authors would like to thank the reviewers for their constructive criticisms.

REFERENCES

[1] J. W. Cooley and J. W. Tukey, "An algorithm for the machine calculation of complex Fourier series," *Math. Comput.*, vol. 19, pp. 297-301, 1965.

[2] R. D. Preuss, "Very fast computation of the radix 2 discrete Fourier transform," *IEEE Trans. Acoust., Speech, Signal Processing*, vol. ASSP-30, pp. 595-607, 1982.

[3] G. D. Bergland, "A fast Fourier transform algorithm for real-valued series," *Commun. ACM*, vol. 11, pp. 703-710, Oct. 1968.

[4] H. Ziegler, "A fast Fourier transform algorithm for symmetric real-valued series," *IEEE Trans. Audio Electroacoust.*, vol. AU-20, pp. 353-356, Dec. 1972.

[5] J. B. Martens, "Discrete Fourier transform algorithms for real-valued sequences," *IEEE Trans. Acoust., Speech, Signal Processing*, vol. ASSP-32, pp. 390-396, Apr. 1984.

[6] M. Vetterli and H. J. Nussbaumer, "Simple FFT and DCT algorithms with reduced number of operations," *Signal Processing*, vol. 6, pp. 267-278, July 1984.

[7] P. Duhamel, "Implementation of split-radix FFT algorithms for complex, real, and real-symmetric data," *IEEE Trans. Acoust., Speech, Signal Processing*, vol. ASSP-34, pp. 285-295, Apr. 1986.

[8] R. N. Bracewell, "The fast Hartley transform," *Proc. IEEE*, vol. 22, pp. 1010-1018, Aug. 1984.

[9] H. V. Sorensen, D. L. Jones, C. S. Burrus, and M. T. Heideman, "On computing the discrete Hartley transform," *IEEE Trans. Acoust., Speech, Signal Processing*, vol. ASSP-33, pp. 1231-1238, Oct. 1985.

[10] P. Duhamel, "Un algorithme de transformation de Fourier rapide à double base," *Ann. Télécommun.*, vol. 40, pp. 481-494, Sept.-Oct. 1985.

[11] M. Vetterli and H. J. Nussbaumer, "Algorithmes de transformation de Fourier et en cosinus mono et bi-dimensionnels," *Ann. Télécommun.*, vol. 40, pp. 466-476, Sept.-Oct. 1985.

[12] H. V. Sorensen, M. T. Heideman, and C. S. Burrus, "On computing the split-radix FFT," *IEEE Trans. Acoust., Speech, Signal Processing*, vol. ASSP-34, pp. 152-156, Feb. 1986.

[13] R. C. Agarwal and C. S. Burrus, "Fast convolution using Fermat number transform with application to digital filtering," *IEEE Trans. Acoust., Speech, Signal Processing*, vol. ASSP-22, pp. 87-97, Apr. 1974.

[14] R. Ansari, "An extension of the discrete Fourier transform," *IEEE Trans. Circuits Syst.*, vol. CAS-32, pp. 618-619, June 1985.

[15] S.-C. Pei and J.-L. Wu, "Split-radix fast Hartley transform," *Electron. Lett.*, vol. 22, pp. 26-27, Jan. 1986.

[16] L. R. Morris, "Automatic generation of time efficient digital signal processing software," *IEEE Trans. Acoust., Speech, Signal Processing*, vol. ASSP-25, pp. 74-79, Feb. 1977.

Pierre Duhamel (M'87) was born in France in 1953. He received the Ingenieur degree in electrical engineering from the National Institute for Applied Sciences, Rennes, France, in 1975, the Dr. Ing. degree in 1978, and the Doctorat es Sciences in 1986 from Orsay University, France.

From 1975 to 1980 he was with Thomson-CSF, Paris, France, where his research interests were in circuit theory and signal processing, including digital filtering and automatic analog fault diagnosis. In 1980 he joined the National Research Center in Telecommunications (CNET), Issy-les-Moulineaux, France, where his activities were first concerned with the design of recursive CCD filters. He is now working on fast convolution algorithms, including number theoretic transforms and fast Fourier transforms.

Martin Vetterli (M'87) for a photograph and biography, see p. 372 of the March 1987 issue of this TRANSACTIONS.