ELSEVIER

Signal Processing 48 (1996) 123–134

SIGNAL PROCESSING

In-place in-order mixed radix fast Hartley transforms

Soo-Chang Pei*, Sy-Been Jaw

Department of Electrical Engineering, National Taiwan University, Taipei, Taiwan, ROC

Received 2 September 1994; revised 10 March 1995 and 3 October 1995

Abstract

This paper presents two Fortran programs that calculate the mixed radix discrete Hartley transform (DHT) using a general odd length p-point DHT module and several short length Winograd DHT (WDHT) modules. Each program has its own advantages such as simplicity for implementation or minimum arithmetic complexity. New efficient radix-3, odd radix-p FHT algorithms and short WDHT modules have been developed to be incorporated into a general FHT algorithm. It allows a much wider selection of transform sizes, and calculates the DHT in order.

Zusammenfassung

Diese Arbeit stellt zwei Fortran-Programme vor, die die diskrete Hartley-Transformation (DHT) für gemischte Basiszahlen (mixed radix) berechnen. Dabei werden ein allgemeiner p-Punkte DHT Modul für ungerade Längen und einige Winograd DHT (WDHT) Module für kurze Längen benutzt. Jedes Programm hat seine eigenen Vorteile wie die Einfachheit einer Implementierung oder minimale arithmetische Komplexität. Es wurden neue Radix-3, sowie FHT Algorithmen für ungerade Basiszahlen p und kurze WDHT Module entwickelt, die in einen allgemeinen FHT-Algorithmus eingearbeitet werden können. Dies erlaubt eine viel größere Auswahl von Transformationslängen und berechnet die DHT in der richtigen Reihenfolge.

Résumé

Ce papier présente deux programmes en Fortran qui calculent la transformée de Hartley discrète à rayon mélangés DHT en utilisant un module DHT p-point général à longueur impaire et plusieurs modules DHT de Winograd à longueur courte. Chaque programme a ses propres avantages tels que la simplicité d'implémentation ou la complexité arithmétique minimum. De nouveaux algorithmes FHT p-rayon impair, rayon-3 et de courts modules WDHT ont été développé pour être incorporés dans un algorithme FHT général. Il permet une sélection bien plus large des tailles de transformée et calcule la DHT dans l'ordre.

Keywords: Hartley transform; Mixed radix; Fast algorithm

* Corresponding author.

SSDI 0165-1684(95)00129-8

1. Introduction

Recently, the discrete Hartley transform (DHT) has been considered as an interesting alternative to the Fourier transform for spectrum analysis and convolution of real signals [2]. Hence it is very desirable to have fast algorithms for its realization; several fast Hartley transform (FHT) algorithms have been developed in these years for efficient transformation of real data. Bracewell [3] demonstrated a radix-2 decimation-in-time FHT algorithm for transform lengths equal to a power of 2. Pei [8] developed a split-radix FHT algorithm to reduce its complexity. Sorensen et al. [9] further showed that most FHT algorithms closely resemble their FFT counterparts. In particular, they suggested a prime-factor mapping technique [5, 9] to construct a prime factor FHT using short length modules. Recently, Lun and Siu proposed a new prime factor FHT without extra additions [6]. These prime factor algorithms still limit the available transform sizes to products of the small DHT module sizes. This paper aims to expand the range of sizes available by developing two mixed radix FHT programs. Several efficient radix-2/3/4/p FHT algorithms and short Winograd DHT modules have been integrated to become a very general and useful software tool. It allows a much wider selection of transform sizes, and calculates the DHT in order. A special section on the Hartley transform and its applications has been recently reported in [7] for good overview.

2. Prime factor index mapping

The discrete Hartley transform of the N-point sequence $x(n)$ is given by

$$H(k) = \sum_{n=0}^{N-1} x(n)\operatorname{cas}(2\pi nk/N),$$

$$\operatorname{cas}\theta = \cos\theta + \sin\theta, \tag{1}$$

for $k = 0, 1, \ldots, N-1$. If the sequence length can be factored into two mutually prime factors $N = N_1 N_2$, then the following substitutions [5, 9] can be made:

$$n = N_2 n_1 + N_1 n_2 \pmod N, \tag{2}$$

$$k = N_2 N_2^{-1} k_1 + N_1 N_1^{-1} k_2 \pmod N, \tag{3}$$

with

$$N_2 N_2^{-1} = 1 \pmod{N_1}, \tag{4}$$

$$N_1 N_1^{-1} = 1 \pmod{N_2}, \tag{5}$$

for $n_1, k_1 = 0, 1, \ldots, N_1 - 1$ and $n_2, k_2 = 0, 1, \ldots, N_2 - 1$. This gives from (1)

$$H(N_2 N_2^{-1} k_1 + N_1 N_1^{-1} k_2) = \sum_{n_1}\sum_{n_2} x(N_2 n_1 + N_1 n_2) \times \operatorname{cas}\left[2\pi\left(\frac{n_1 k_1}{N_1} + \frac{n_2 k_2}{N_2}\right)\right]. \tag{6}$$

Defining the two-dimensional arrays $\hat{H}$ and $\hat{x}$ gives

$$\hat{H}(k_1, k_2) = H(N_2 N_2^{-1} k_1 + N_1 N_1^{-1} k_2), \tag{7a}$$

$$\hat{x}(n_1, n_2) = x(N_2 n_1 + N_1 n_2) \tag{7b}$$

and

$$\hat{H}(k_1, k_2) = \sum_{n_1=0}^{N_1-1}\sum_{n_2=0}^{N_2-1} \hat{x}(n_1, n_2)\operatorname{cas}\left[2\pi\left(\frac{n_1 k_1}{N_1} + \frac{n_2 k_2}{N_2}\right)\right]. \tag{7c}$$

This prime factor mapping is very unique that the original length N transform kernel can be decomposed into a pure two-dimensional $N_1 \times N_2$ DHT kernel without generating any coupling twiddle factors.

By direct analogy with the two-dimensional FFT, one takes the one-dimensional DHTs of the rows one by one, and then transform the columns. The temporary output $T(k_1, k_2)$ is of the form [4]

$$T(k_1, k_2) = \sum_{n_1=0}^{N_1-1}\sum_{n_2=0}^{N_2-1} \hat{x}(n_1, n_2)\operatorname{cas}(2\pi n_1 k_1/N_1) \times \operatorname{cas}(2\pi n_2 k_2/N_2). \tag{8}$$

Eq. (8) is not the two-dimensional Hartley transform; however, the result can be converted to the desired two-dimensional Hartley transform of Eq. (7c) as follows:

Since

$$2\operatorname{cas}(\alpha+\beta) = \operatorname{cas}\alpha\operatorname{cas}\beta + \operatorname{cas}\alpha\operatorname{cas}(-\beta) + \operatorname{cas}(-\alpha)\operatorname{cas}\beta - \operatorname{cas}(-\alpha)\operatorname{cas}(-\beta), \tag{9}$$

the desired Hartley transform $\hat{H}(k_1, k_2)$ can be expressed as a sum of four temporary transforms:

$$2\hat{H}(k_1, k_2) = T(k_1, k_2) + T(k_1, N_2 - k_2) + T(N_1 - k_1, k_2) - T(N_1 - k_1, N_2 - k_2). \quad (10)$$

Finally, we use the index mapping (Eqs. (2) and (3)) to recover $H(K)$ from $\hat{H}(K_1, K_2)$.

3. Radix-3 FHT algorithm

When the number of data samples is close to a power of 3, rather than a power of 2 or 4, the radix-3 FHT algorithms [1, 10] can be used effectively. Anupindi et al. [1] have developed an efficient radix-3 FHT algorithm, which is not in-place and needs some special "trit-reversed" input data ordering [1]. Zhao [10] proposes another novel radix-3 algorithm which requires less arithmetic complexity than the above algorithm; however, this new algorithm is actually partially in-place and needs extra data transfer load at each decomposition stage. This is due to the fact that the two $N/3$-point DHTs $H_1(k)$ and $H_{-1}(k)$ in Zhao's algorithm [10] need to be subtracted from each other and time-reversed to get $B(k) = H_1(\frac{1}{3}N - k) - H_{-1}(\frac{1}{3}N - k)$ [10]. These time consuming data transfer or time-reversal operations are carried out in the "Arrange" unit of Figs. 1 and 2 in [10]. For fast transform performance analysis, data transfer load is also a critical and important factor to be considered as well as multiplication and addition complexity. In this paper, a truly in-place in-order radix-3 algorithm is presented. This algorithm eliminates the time-consuming data transfer load, needs only conventional radix-3 digit-reversed input ordering, and keeps the same arithmetic complexity as Zhao's algorithm [10].

Assume the DHT of Eq. (1) for a real N-point sequence $x(n)$ needs to be computed with $N = 3^r$.

A radix-3 decimation-in-time (DIT) FHT algorithm is found by decomposing (1) as follows:

$$H(k) = \sum_{n=0}^{N/3-1} x(3n)\operatorname{cas}\left(\frac{2\pi nk}{N/3}\right) + \cos\frac{2\pi k}{N}\sum_{n=0}^{N/3-1} x(3n+1)\operatorname{cas}\left(\frac{2\pi nk}{N/3}\right) + \sin\frac{2\pi k}{N}\sum_{n=0}^{N/3-1} x(3n+1)\operatorname{cas}\left(\frac{-2\pi nk}{N/3}\right) + \cos\frac{4\pi k}{N}\sum_{n=0}^{N/3-1} x(3n+2)\operatorname{cas}\left(\frac{2\pi nk}{N/3}\right) + \sin\frac{4\pi k}{N}\sum_{n=0}^{N/3-1} x(3n+2)\operatorname{cas}\left(\frac{-2\pi nk}{N/3}\right). \quad (11)$$

Define

$$H_0(k) = \sum_{n=0}^{N/3-1} x(3n)\operatorname{cas}\left(\frac{2\pi nk}{N/3}\right), \quad (12a)$$

$$H_1(k) = \sum_{n=0}^{N/3-1} x(3n+1)\operatorname{cas}\left(\frac{2\pi nk}{N/3}\right), \quad (12b)$$

$$H_2(k) = \sum_{n=0}^{N/3-1} x(3n+2)\operatorname{cas}\left(\frac{2\pi nk}{N/3}\right), \quad (12c)$$

and

$$\theta = \frac{2\pi k}{N}, \quad (13)$$

then Eq. (11) becomes

$$H(k) = H_0(k) + H_1(k)\cos\theta + H_1(-k)\sin\theta + H_2(k)\cos 2\theta + H_2(-k)\sin 2\theta. \quad (14a)$$

$$H(-k) = H_0(-k) + H_1(-k)\cos\theta - H_1(k)\sin\theta + H_2(-k)\cos 2\theta - H_2(k)\sin 2\theta. \quad (14b)$$

where $H(-k) = H(N-k)$ and $H_i(-k) = H_i(\frac{1}{3}N - k)$ for $i = 0, 1, 2$.

$$H(k + \tfrac{1}{3}N) = H_0(k) - \tfrac{1}{2}H_1(k)\cos\theta - \tfrac{1}{2}\sqrt{3}H_1(k)\sin\theta - \tfrac{1}{2}H_1(-k)\sin\theta + \tfrac{1}{2}\sqrt{3}H_1(-k)\cos\theta - \tfrac{1}{2}H_2(k)\cos 2\theta + \tfrac{1}{2}\sqrt{3}H_2(k)\sin 2\theta - \tfrac{1}{2}H_1(-k)\sin 2\theta - \tfrac{1}{2}\sqrt{3}H_2(-k)\cos 2\theta, \quad (15a)$$

$$\begin{aligned}H(-k+\tfrac{1}{3}N) &= H_0(-k) - \tfrac{1}{2}H_1(-k)\cos\theta \\ &\quad + \tfrac{1}{2}\sqrt{3}H_1(-k)\sin\theta \\ &\quad + \tfrac{1}{2}H_1(k)\sin\theta + \tfrac{1}{2}\sqrt{3}H_1(k)\cos\theta \\ &\quad - \tfrac{1}{2}H_2(-k)\cos 2\theta \\ &\quad - \tfrac{1}{2}\sqrt{3}H_2(-k)\sin 2\theta \\ &\quad + \tfrac{1}{2}H_1(k)\sin 2\theta \\ &\quad - \tfrac{1}{2}\sqrt{3}H_2(k)\cos 2\theta, \qquad (15b)\end{aligned}$$

$$\begin{aligned}H(k+\tfrac{2}{3}N) &= H_0(k) - \tfrac{1}{2}H_1(k)\cos\theta \\ &\quad + \tfrac{1}{2}\sqrt{3}H_1(k)\sin\theta - \tfrac{1}{2}H_1(-k)\sin\theta \\ &\quad - \tfrac{1}{2}\sqrt{3}H_1(-k)\cos\theta \\ &\quad - \tfrac{1}{2}H_2(k)\cos 2\theta - \tfrac{1}{2}\sqrt{3}H_2(k)\sin 2\theta \\ &\quad - \tfrac{1}{2}H_2(-k)\sin 2\theta \\ &\quad + \tfrac{1}{2}\sqrt{3}H_2(-k)\cos 2\theta, \qquad (16a)\end{aligned}$$

$$\begin{aligned}H(-k+\tfrac{2}{3}N) &= H_0(-k) - \tfrac{1}{2}H_1(-k)\cos\theta \\ &\quad - \tfrac{1}{2}\sqrt{3}H_1(-k)\sin\theta \\ &\quad + \tfrac{1}{2}H_1(k)\sin\theta - \tfrac{1}{2}\sqrt{3}H_1(k)\cos\theta \\ &\quad - \tfrac{1}{2}H_2(-k)\cos 2\theta \\ &\quad + \tfrac{1}{2}\sqrt{3}H_2(-k)\sin 2\theta \\ &\quad - \tfrac{1}{2}H_2(k)\sin 2\theta \\ &\quad + \tfrac{1}{2}\sqrt{3}H_2(k)\cos 2\theta,\end{aligned}$$

$$0 \leqslant k \leqslant \tfrac{1}{2}(\tfrac{1}{3}N - 1). \qquad (16b)$$

Notice that at each stage both the kth term and the $(\frac{1}{3}N - k)$th term from each length-$\frac{1}{3}N$ DHT (i.e. the six terms $H_i(k)$ and $H_i(-k)$, $i = 0, 1, 2$) are required for the computation of one output point $H(K)$. Thus the above six terms in the conventional DIT FHT two 3-point butterflies can be combined into a 6-point butterfly in Fig. 1(a) to avoid overwriting an element that will be needed later [9]. This results in saving of both data transfer load and the number of multiplications. Each 6-point butterfly requires 10 multiplications and 16 additions totally. The three outputs of $H(0)$, $H(N/3)$ and $H(2N/3)$ for $k = 0$ in Eqs. (14)–(16) are reduced into a 3-point butterfly in Fig. 1(b), also the twiddle factors are equal to 1 and 0, respectively. Then Eqs. (14)–(16) become

$$H(0) = H_0(0) + [H_1(0) + H_2(0)], \qquad (17a)$$

$$\begin{aligned}H(\tfrac{1}{3}N) &= H_0(0) - \tfrac{1}{2}[H_1(0) + H_2(0)] \\ &\quad + \tfrac{1}{2}\sqrt{3}[H_1(0) - H_2(0)], \qquad (17b)\end{aligned}$$

$$\begin{aligned}H(\tfrac{2}{3}N) &= H_0(0) - \tfrac{1}{2}[H_1(0) + H_2(0)] \\ &\quad - \tfrac{1}{2}\sqrt{3}[H_1(0) - H_2(0)]. \qquad (17c)\end{aligned}$$

This 3-point butterfly needs only one multiplication, six additions and one right shift (multiplied by $\frac{1}{2}$).

For our proposed algorithm, the above decomposition of an N-point DHT into three $\frac{1}{3}N$-point DHTs only requires $\frac{1}{2}(\frac{1}{3}N - 1)$ 6-point butterflies for $k \neq 0$ plus one additional 3-point butterfly for $k = 0$. Assume the number of multiplications and additions required for an N-point DHT are $M(r)$ and $A(r)$, respectively, where $N = 3^r$, then

$$\begin{aligned}M(r) &= 3M(r-1) + \tfrac{1}{2}(\tfrac{1}{3}N - 1)\times 10 + 1 \\ &= 3M(r-1) + \tfrac{5}{3}N - 4, \qquad (18a)\end{aligned}$$

$$\begin{aligned}A(r) &= 3A(r-1) + \tfrac{1}{2}(\tfrac{1}{3}N - 1)\times 16 + 6 \\ &= 3A(r-1) + \tfrac{8}{3}N - 2. \qquad (18b)\end{aligned}$$

For Zhao's algorithm in [10], the "trit" decomposition [1, 10] into three $\frac{1}{3}N$-point DHTs requires $(\frac{1}{3}N - 1)$ 3-point butterflies for $k \neq 0$ and one 3-point butterfly for $k = 0$ plus additional data transfer for $B(k)$. Each 3-point butterfly for $K \neq 0$ needs five multiplications, instead of four multiplications in Fig. 1 of [10], and eight additions. Multiplication by $\frac{1}{2}\sqrt{3}$ has been neglected carelessly and it needs be counted in Zhao's 3-point butterfly, then we obtain the multiplications as $M(r) = 3M(r-1) + (\frac{1}{3}N - 1)\times 5 + 1 = 3M(r-1) + \frac{5}{3}N - 4$ and the additions as $A(r) = 3A(r-1) + (\frac{1}{3}N - 1)\times 8 + 6 = 3A(r-1) + \frac{8}{3}N - 2$.

We get exactly the same arithmetic complexity in both algorithms, however, the proposed algorithm is unique in normal digit reversal decomposition for radix-3, and has eliminated the time consuming data transfer load at each decomposition stage.

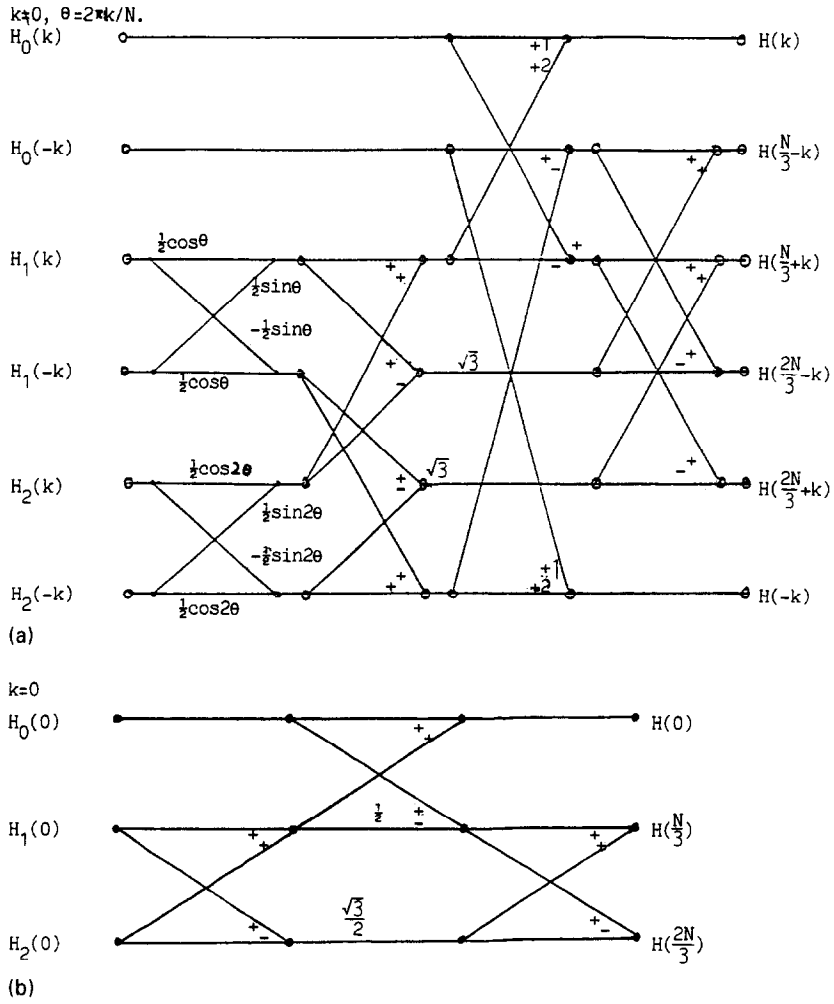

Fig. 1. (a) 6-point radix-3 FHT butterfly for $k \neq 0$. (b) 3-point radix-3 FHT butterfly for $k = 0$.

Using the initial conditions $M(1) = 1$ and $A(1) = 6$, we obtain

$$M(r) = \tfrac{5}{3}rN - 2N + 2, \quad A(r) = \tfrac{8}{3}rN - N + 1,$$

$$\text{where} \quad r = \log_3 N. \tag{19}$$

The above algorithm can be effectively used by zero padding when the transform length is equal to or close to a power of 3. More importantly, it can be combined with radix-2 FHTs to form an efficient algorithm of composite length 2^k3^l. Note that the density of DHT lengths covered by length 2^k3^l FHTs is much higher than that of single radix algorithms, thus making such mixed radix FHTs quite flexible and useful.

4. Odd radix-p FHT algorithm

In this section, we will develop a general odd radix-p FHT algorithm although it is not optimal in minimum arithmetic complexity. However, it does have its simplicity advantage for software implementation, and it is suitable for any specific odd radix FHT algorithm. A radix-p DIT FHT algorithm is decomposed into several length-N/p

DHTs in a similar way as radix-3 FHT (i.e. $H_i(k)$ and $H_i(-k)$, $i = 0, 1, 2, \ldots, p$).

$$H(k) = H_0(k) + \sum_{j=1}^{p-1} [H_j(k)\cos(2\pi jk/N) + H_j(-k)\sin(2\pi jk/N)]. \quad (20)$$

$$\begin{aligned} H\left(k + \frac{mN}{p}\right) &= H_0(k) + \sum_{j=1}^{p-1} H_j(k)\cos(2\pi jk/N)\cos(2\pi jm/p) \\ &\quad - H_j(k)\sin(2\pi jk/N)\sin(2\pi jm/p) \\ &\quad + H_j(-k)\cos(2\pi jk/N)\sin(2\pi jm/p) \\ &\quad + H_j(-k)\sin(2\pi jk/N)\cos(2\pi jm/p). \end{aligned} \quad (21)$$

Let $\theta = 2\pi k/N$, assume p is odd; Eq. (21) then becomes

$$\begin{aligned} H\left(k + \frac{mN}{p}\right) &= H_0(k) + \sum_{j=1}^{(p-1)/2} \{[H_j(k)\cos j\theta + H_j(-k)\sin j\theta \\ &\quad + H_{p-j}(k)\cos(p-j)\theta \\ &\quad + H_{p-j}(-k)\sin(p-j)\theta]\cos(2\pi jm/p) \\ &\quad + [H_j(-k)\cos j\theta - H_j(k)\sin j\theta \\ &\quad - H_{p-j}(-k)\cos(p-j)\theta \\ &\quad + H_{p-j}(k)\sin(p-j)\theta]\sin(2\pi jm/p)\} \end{aligned} \quad (22)$$

and

$$\begin{aligned} H\left(k + \frac{(p-m)}{p}N\right) &= H_0(k) + \sum_{j=1}^{(p-1)/2} \{[H_j(k)\cos j\theta + H_j(-k)\sin j\theta \\ &\quad + H_{p-j}(k)\cos(p-j)\theta \\ &\quad + H_{p-j}(-k)\sin(p-j)\theta]\cos(2\pi m/p) \\ &\quad - [H_j(-k)\cos j\theta - H_j(k)\sin j\theta \\ &\quad - H_{p-j}(-k)\cos(p-j)\theta \\ &\quad + H_{p-j}(k)\sin(p-j)\theta]\sin(2\pi jm/p)\}, \end{aligned}$$

$k = 0, 1, \ldots, \frac{1}{2}(N/p - 1)$ and

$m = 0, 1, \ldots, \frac{1}{2}(p-1)$. (23)

The same redundant terms in Eqs. (21)–(23) can be used for computation saving. Similar expressions for $H(-k)$, $H(-k + \frac{mN}{p})$ and $H(-k + \frac{(p-m)}{p}N)$ are not listed here for saving space.

For $k = 0$, Eqs. (22) and (23) are reduced to

$$\begin{aligned} H\left(\frac{mN}{p}\right) &= H_0(0) \\ &\quad + \sum_{j=1}^{(p-1)/2} \{[H_j(0) + H_{p-j}(0)]\cos(2\pi jm/p) \\ &\quad + [H_j(0) - H_{p-j}(0)]\sin(2\pi jm/p)\}, \end{aligned} \quad (24)$$

$$\begin{aligned} H\left(\frac{(p-m)}{p}N\right) &= H_0(0) \\ &\quad + \sum_{j=1}^{(p-1)/2} \{[H_j(0) + H_{p-j}(0)]\cos(2\pi jm/p) \\ &\quad - [H_j(0) - H_{p-j}(0)]\sin(2\pi jm/p)\}. \end{aligned} \quad (25)$$

Eqs. (22)–(25) are general radix-p butterfly forms of Eqs. (14)–(17) in radix-3 case. Each radix-p butterfly requiring about $(p+1)^2 - 4$ multiplications/additions, the total number of multiplications/additions is approximately equal to

$$\begin{aligned} &(p^2 + 2p - 3) \text{ operations/butterfly} \\ &\quad \times \left(\frac{N}{p \times 2}\right) \text{ butterflies/stage} \\ &\quad \times (\log_p N) \text{ stages} \approx \frac{p^2 + 2p - 3}{2p} N \log_p N \\ &\quad \text{for } N = p^r. \end{aligned} \quad (26)$$

5. Winograd short-length DHT modules

It is well-known that the short-length Winograd DFT modules [5] can get the minimum number of multiplications for prime-length transforms. The algorithm can be expressed as

$$X(k) = \mathrm{DFT}_p[x(n)] = S_p C_p T_p[x(n)], \quad (27)$$

where T_p is a $J \times p$ pre-weave incidence matrix operator, S_p is a $p \times J$ post-weave incidence matrix operator and C_p is a $J \times J$ diagonal matrix with complex

entries, and J is the number of multiplications. The incidence matrices T_p and S_p are matrices, whose elements are -1, 0 or 1, leading only to additions and subtractions. The diagonal matrix C_p is decomposed into its real and imaginary parts by

$$X(k) = \mathrm{DFT}_p[x(n)] = S_p[C_p^{\mathrm{R}} + \mathrm{j}C_p^{\mathrm{I}}]\, T_p[x(n)]$$
$$= S_p C_p^{\mathrm{R}} T_p[x(n)] + \mathrm{j} S_p C_p^{\mathrm{I}} T_p[x(n)]. \quad (28)$$

Since the DHT is equivalent to subtracting the imaginary part from the real part of the DFT $X(k)$, we get

$$H(k) = \mathrm{DHT}_p[x(n)]$$
$$= S_p C_p^{\mathrm{R}} T_p[x(n)] - S_p C_p^{\mathrm{I}} T_p[x(n)]$$
$$= S_p[C_p^{\mathrm{R}} - C_p^{\mathrm{I}}]\, T_p[x(n)] = S_p \hat{C}_p^{\mathrm{H}} T_p[x(n)], \quad (29)$$

where $\hat{C}_P^{\mathrm{H}} = C_p^{\mathrm{R}} - C_p^{\mathrm{I}}$ is the new diagonal matrix with real entries for the Winograd DHT module, and the matrices S_p and T_p are the same as the corresponding Winograd DFT modules. So the DHT via the Winograd approach requires exactly the same minimum number of multiplications, but a few more additions than the DFT. We give an example for computing the length-5 DHT as below

$$H(k) = \mathrm{DHT}_5[x(n)] = S_5 \hat{C}_5^{\mathrm{H}} T_5\, x(n), \quad (30a)$$

$$\begin{bmatrix} H(0) \\ H(1) \\ H(2) \\ H(3) \\ H(4) \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & -1 & 0 \\ 1 & 1 & -1 & 1 & 0 & 1 \\ 1 & 1 & -1 & -1 & 0 & -1 \\ 1 & 1 & 1 & -1 & 1 & 0 \end{bmatrix}$$
$$\times \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & -1.25 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0.56 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0.95 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1.54 & 0 \\ 0 & 0 & 0 & 0 & 0 & -0.36 \end{bmatrix}$$
$$\times \begin{bmatrix} 1 & 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 1 & 1 \\ 0 & 1 & -1 & -1 & 1 \\ 0 & 1 & -1 & 1 & -1 \\ 0 & 0 & -1 & 1 & 0 \\ 0 & 1 & 0 & 0 & -1 \end{bmatrix} \begin{bmatrix} x(0) \\ x(1) \\ x(2) \\ x(3) \\ x(4) \end{bmatrix}. \quad (30b)$$

This length-5 Winograd DHT only requires 5 real multiplications and 17 real additions. The other short Winograd DHT modules of length 7, 8, 9 and 16 can be easily obtained by modifying the DFT modules in [5].

During the middle stages of the decomposition, the general Winograd radix-p FFT butterfly can be expressed as

$$F(k) = SCTW f(n), \quad (31)$$

where $f(n)$ is a $p \times 1$ vector of previous stage output values, and W is a twiddle-factor diagonal matrix with the following form:

$$W = \begin{bmatrix} 1 & & & & 0 \\ & w^{-n} & & & \\ & & w^{-2n} & & \\ & & & \ddots & \\ 0 & & & & w^{-(p-1)n} \end{bmatrix}. \quad (32)$$

The main difference between Eqs. (32) and (27) is that the input $f(n)$ needs be premultiplied first by the twiddle factors, and then transformed by the p-point DFT module. Since in the middle stages of the transformation, $f(n)$ and $F(k)$ are generally complex valued, we get

$$[F_{\mathrm{R}}(k) + \mathrm{j}F_{\mathrm{I}}(k)] = S[C_{\mathrm{R}} + \mathrm{j}C_{\mathrm{I}}]\, T\, [W_{\mathrm{R}} + \mathrm{j}W_{\mathrm{I}}]$$
$$\times [f_{\mathrm{R}}(n) + \mathrm{j}f_{\mathrm{I}}(n)] \quad (33)$$

We separate the real-part and the imaginary part:

$$F_{\mathrm{R}}(k) = SC_{\mathrm{R}} T W_{\mathrm{R}} f_{\mathrm{R}}(n) - SC_{\mathrm{R}} T W_{\mathrm{I}} f_{\mathrm{I}}(n)$$
$$- SC_{\mathrm{I}} T W_{\mathrm{R}} f_{\mathrm{I}}(n) - SC_{\mathrm{I}} T W_{\mathrm{I}} f_{\mathrm{R}}(n), \quad (34a)$$

$$F_{\mathrm{I}}(k) = SC_{\mathrm{R}} T W_{\mathrm{R}} f_{\mathrm{I}}(n) + SC_{\mathrm{R}} T W_{\mathrm{I}} f_{\mathrm{R}}(n)$$
$$+ SC_{\mathrm{I}} T W_{\mathrm{R}} f_{\mathrm{R}}(n) - SC_{\mathrm{I}} T W_{\mathrm{I}} f_{\mathrm{I}}(n). \quad (34b)$$

The Hartley transform can be expressed as the real-part minus the imaginary-part:

$$\begin{aligned} H(k) &= F_R(k) - F_I(k) \\ &= SC_R T W_R[f_R(n) - f_I(n)] \\ &\quad - SC_R T W_I[f_R(n) + f_I(n)] \\ &\quad - SC_I T W_R[f_R(n) + f_I(n)] \\ &\quad - SC_I T W_I[f_R(n) - f_I(n)]. \end{aligned} \tag{35a}$$

If we add $F_R(k)$ and $F_I(k)$, we get $H(-k) = F_R(-k) - F_I(-k) = F_R(k) + F_I(k)$, since the real part is symmetric, the imaginary part is anti-symmetric with respect to k. We get

$$\begin{aligned} H(-k) &= F_R(k) + F_I(k) \\ &= SC_R T W_R[f_R(n) + f_I(n)] \\ &\quad + SC_R T W_I[f_R(n) - f_I(n)] \\ &\quad + SC_I T W_R[f_R(n) - f_I(n)] \\ &\quad - SC_I T W_I[f_R(n) + f_I(n)]. \end{aligned} \tag{35b}$$

Substituting $h(n) = f_R(n) - f_I(n)$ and $h(-n) = f_R(n) + f_I(n)$, we obtain

$$\begin{aligned} H(k) &= SC_R T W_R[h(n)] - SC_R T W_I[h(-n)] \\ &\quad - SC_I T W_R[h(-n)] - SC_I T W_I[h(n)], \end{aligned} \tag{36a}$$

$$\begin{aligned} H(-k) &= SC_R T W_R[h(-n)] + SC_R T W_I[h(n)] \\ &\quad + SC_I T W_R[h(n)] - SC_I T W_I[h(-n)]. \end{aligned} \tag{36b}$$

Eqs. (36a) and (36b) can be combined into a $2p$-point FHT butterfly to avoid overwriting and to save both data transfer load and multiplications. If the index n of the previous stage is zero, then $h(0)$ and $h(-0)$ will be the same for each group; so there are only p points in the butterfly. In this case, the twiddle-factor matrix W of Eq. (32) for $n = 0$ is an identity matrix I_R. Then

$$\begin{aligned} &W_R = I_R, \qquad W_I = 0, \\ &H(k) = SC_R T[h(0)] - SC_I T[h(-0)] \\ &\quad = S(C_R - C_I) T[h(0)]. \end{aligned} \tag{37}$$

Eq. (37) will form a p-point FHT butterfly to be used for transformation. Since the size of the present stage $H(k)$ is N-point length, then $H(-k)$ in Eq. (36b) is equivalent to $H(N-K)$. However, if the size of $h(n)$ is (N/P)-point length, then $h(-n)$ is defined as $h[(N/P) - n]$ in Eq. (36a) and (36b).

Each radix has two major transformation procedures, one for p points, the other for $2p$ points. For example in our radix-5 Winograd DHT subroutine R5WDHT(M, x), the subprogram R5BTF1 does the 5-point butterfly procedure, and the routine R5BTF2 performs the 10-point butterfly operation.

6. Complexity analysis

Assume a composite length-N DHT is to be computed, where $N = M_1M_2M_3$, $M_1 = p_1^{r_1}$, $M_2 = p_2^{r_2}$, $M_3 = p_3^{r_3}$ and the factors M_1, M_2, M_3 are co-prime with respect to each other. Then this mixed-radix FHT needs M_2M_3 length-M_1 DHTs M_1M_3 length-M_2 DHTs and M_1M_2 length-M_3 DHT computations, totally. We have

♯Multiplies

$$= M_2M_3(\mu_1) + M_1M_3(\mu_2) + M_1M_2(\mu_3), \tag{38a}$$

♯Adds

$$= M_2M_3(\alpha_1) + M_1M_3(\alpha_2) + M_1M_2(\alpha_3). \tag{38b}$$

Here μ_i, α_i, $i = 1, 2, 3$, are the numbers of multiplications and additions required for each radix-p_i length-M_i DHT computation, respectively, in which μ_i and α_i can be approximately estimated as

$$\mu_i \simeq k_i^M \frac{M_i}{p_i} \log_{p_i} M_i = k_i^M \frac{M_i r_i}{p_i}, \tag{39a}$$

$$\alpha_i \simeq k_i^A \frac{M_i}{p_i} \log_{p_i} M_i = k_i^A \frac{M_i r_i}{p_i}. \tag{39b}$$

Here k_i^M and k_i^A are the numbers of multiplies and adds for a length-p_i Winograd DHT butterfly or module. Table 1 lists the multiplications and additions required for each short length WDHT module or butterfly.

There are $(p-1)$ twiddle factors to be premultiplied; the first twiddle factor is equal to 1, which can be eliminated. Each twiddle factor needs two multiplications and one addition. So the total number of multiplies/adds in each p-point WDHT butterfly

Table 1
Number of real Multiplies and Adds for short length p-real WDHT module (Butterfly)

p	M	(MB1)	(MB2)	A	(AB1)	(AB2)
2	0	(2)	(4)	2	(3)	(6)
3	2	(6)	(12)	6	(8)	(16)
4	0	(6)	(12)	8	(11)	(22)
5	5	(13)	(26)	17	(21)	(42)
6	8	(20)	(40)	36	(42)	(84)
7	2	(16)	(32)	26	(33)	(66)
8	10	(26)	(52)	44	(52)	(104)
9	10	(40)	(80)	74	(89)	(178)

MB1/AB1 are valid for a p-point butterfly, MB2/AB2 for a $2p$-point butterfly.

will be the sum of the twiddle-factor premultiply operations plus the WDHT transform module. It is calculated as below and shown in Table 1.

$$MB1 = M + 2(p-1), \qquad MB2 = 2MB1, \tag{40a}$$

$$AB1 = A + 1(p-1), \qquad AB2 = 2AB1. \tag{40b}$$

For the $2p$-point butterfly case, twice the number of the point butterfly operations are required (see MB2 and AB2 in Table 1). We give an example and calculate the complexity of a length-12 DHT: For $N = 12 = 3 \times 4$, we obtain

$$\sharp M = 3(\mu_4) + 4(\mu_3) = 3 \times 0 + 4 \times 2 = 8,$$

$$\sharp A = 3(\alpha_4) + 4(\alpha_3) = 3 \times 8 + 4 \times 6 = 36.$$

For $N = 12 = 3 \times 2^2$, we get

$$M = 3 \times (2 \times \tfrac{4}{2} \times 2) + 4 \times 2 = 32,$$

$$A = 3 \times (3 \times \tfrac{4}{2} \times 2) + 4 \times 6 = 60.$$

Since a length-4 WDHT module is much more efficient than a radix-2 DHT, we prefer the first factorization with $N = 3 \times 4$ instead of $N = 3 \times 2^2$. According to our experience, the large prime length of p and small power of r will usually be the better choice. Table 2 and Fig. 2 list the number of operation counts for several mixed radix DHT's for reference. Since the general odd radix-9 and radix-13

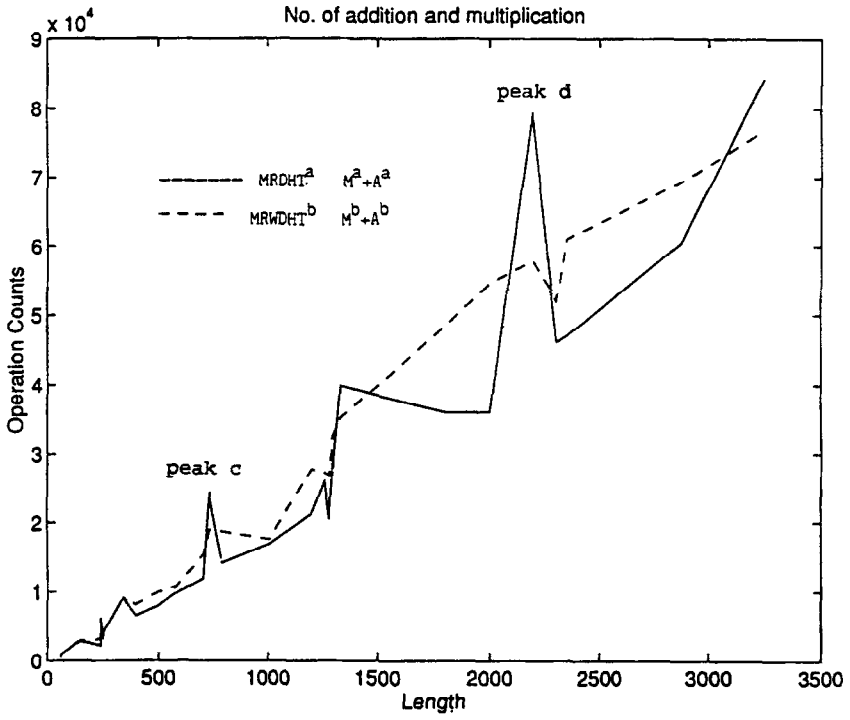

Fig. 2. The number of operation counts for the mixed radix DHTs.

Table 2
Time in milliseconds and operation counts for two FHT implementations

Length N	Factors	MRD-HT[a]	MRWD-HT[b]	M^a	A^a	$M^a + A^a$	M^b	A^b	$M^b + A^b$
60	$3\cdot4\cdot5$	66	57	180	360	540	100	444	544
63	$7\cdot9$	65	44	441	441	882	142	544	686
63	$7\cdot3^2$	59	56	273	273	546	156	436	592
64	2^6	59	59	384	576	960	384	576	960
125	5^3	153	95	1275	1275	2500	975	1575	2550
147	$3\cdot7^2$	169	143	1407	1407	2814	938	2058	2996
210	$2\cdot3\cdot5\cdot7$	315	280	1260	1260	2520	590	2424	3014
240	$3\cdot5\cdot16$	None	228	None	None	None	550	2406	2956
240	$3\cdot5\cdot4^2$	264	290	1120	1120	2240	1120	2616	3736
243	3^5	250	217	2430	2430	4860	2430	3240	5670
252	$4\cdot7\cdot9$	308	241	1764	1764	3528	568	3032	3600
252	$4\cdot7\cdot3^2$	275	288	1764	1764	3528	1296	3144	4440
256	4^4	245	210	1536	2816	4352	1536	2816	4352
343	7^3	467	294	4557	4557	9114	2940	6174	9114
400	$16\cdot5^2$	None	360	None	None	None	2330	5210	7540
400	$4^2\cdot5^2$	374	465	3280	3280	6560	3280	5560	8840
500	$4\cdot5^3$	557	527	4000	4000	8000	3900	6100	10000
576	$9\cdot8^2$	None	548	None	None	None	2944	7568	10512
576	$9\cdot4^3$	557	593	4896	4896	9792	3232	7568	10800
700	$4\cdot7\cdot5^2$	881	838	5740	5740	11840	4440	10880	15320
729	9^3	1084	623	11907	11907	23814	6318	12636	18954
784	$16\cdot7^2$	None	766	None	None	None	4970	13034	18004
784	$4^2\cdot7^2$	794	950	7056	7056	14112	5656	13720	19376
1008	$7\cdot9\cdot16$	None	975	None	None	None	2662	13694	16356
1008	$7\cdot3^2\cdot4^2$	1125	1404	7434	7434	14868	7968	11832	19800
1008	$7\cdot9\cdot4^2$	1256	1232	9450	9450	18900	5056	11384	16440
1200	$3\cdot16\cdot5^2$	None	1428	None	None	None	7790	18030	25820
1200	$3\cdot4^2\cdot5^2$	1428	1724	10640	10640	21200	10640	19080	29720
1260	$4\cdot5\cdot7\cdot9$	1960	1560	13230	13230	26460	4100	19444	23544
1260	$5\cdot7\cdot2^2\cdot3^2$	1834	2207	13860	13860	27720	9252	21264	30516
1260	$5\cdot7\cdot9\cdot2^2$	2006	1941	13860	13860	27720	6620	20704	27324
1260	$4\cdot5\cdot7\cdot3^2$	1799	1795	11025	11025	22050	7740	20004	27744
1280	$5\cdot16^2$	None	1271	None	None	None	7680	18592	26272
1280	$5\cdot4^4$	1286	1396	10240	10240	20480	8960	18432	27392
1296	$16\cdot9^2$	None	1232	None	None	None	8295	20970	29265
1296	$4^2\cdot9^2$	1410	1558	14256	14256	28512	11376	22104	33480
1296	$4^2\cdot3^4$	1050	1718	14248	14248	28496	14248	20982	35200
1331	11^3	2154	None	19965	19965	39930	None	None	None
1800	$8\cdot9\cdot5^2$	None	2142	None	None	None	11810	29860	41670
1800	$9\cdot2^3\cdot5^2$	2419	2845	19800	19800	39600	16760	32020	48780
1800	$2^3\cdot3^2\cdot5^2$	2180	3158	16200	16200	32400	21960	32820	54780
2000	$16\cdot5^3$	None	2106	None	None	None	16850	34450	51300
2000	$4^2\cdot5^3$	2133	2604	18000	18000	36000	21600	36200	57800
2197	13^3	3862	None	39546	39546	79092	None	None	None
2304	$9\cdot16^2$	None	2319	None	None	None	14080	36896	50976
2304	$9\cdot4^4$	2259	2553	23040	23040	46080	16384	36608	52992
2352	$3\cdot16\cdot7^2$	None	2935	None	None	None	16478	40806	57284
2352	$3\cdot4^2\cdot7^2$	3008	3514	23520	23520	47040	22064	42728	64792
2880	$5\cdot9\cdot8^2$	None	3516	None	None	None	17600	47632	65232
2880	$5\cdot3^2\cdot4^3$	3211	4263	30240	30240	60480	27360	48912	76272
2880	$5\cdot9\cdot4^3$	3590	3743	30240	30240	60480	19040	47632	66672
3240	$5\cdot8\cdot9^2$	None	3956	None	None	None	11520	53226	64746
3240	$5\cdot2^3\cdot9^2$	4617	5221	42120	42120	84240	31680	57276	88956
3375	15^3	6324	None	70785	70785	141750	None	None	None

MRDHT routines are less efficient than the others, notice that there are two sharp peaks c and d occurring at length 729(9^3) and 2197(13^3), respectively, in Fig. 2. For length 1260 with four different choices of factors in Table 2, these number of operation counts are averaged and plotted in Fig. 2.

7. Brief program description and speed measurements

Two in-place, in-order mixed radix FHT programs have been implemented in Fortran on our Micro-VAX 3600 computer. The complete programs, mixed radix FHT subroutine "MRDHT" using the general radix-p algorithm, and a mixed-radix Winograd FHT subroutine "MRWDHT" with small DHT modules are available on request from the authors. The programs take real input data in array X, and calculate a length-N DHT in-place and in-order (the output being written over the input X). The length N must be such that it can be written as a product of M factors, which are powers of relatively prime numbers stored in the integer array PO. The even locations of array PO such as PO(0), PO(2) and PO(4), etc. store the prime numbers, and the odd locations such as PO(1), PO(3), PO(5), etc. are their corresponding powers. This gives $N = [\mathrm{PO}(0) ** \mathrm{PO}(1)] * [\mathrm{PO}(2) ** \mathrm{PO}(3)] * \cdots * [\mathrm{PO}(2M) ** \mathrm{PO}(2M+1)]$. If $\mathrm{PO}(2n) = 0$ for some n, this means the end of the parameter list. The maximum number of factors is $M = 3$. However, the routines can be extended easily to more than three factors by slightly modifying the main program. The maximum transform length of the DHTs is limited to $N = 5000$. Since the DHT is its own inverse, the two main subroutines MRDHT and MRWDHT can be used for both forward and inverse transformation.

The MRDHT subroutine includes radix-2, radix-3, radix-4 and general odd radix-p FHT transform routines, in which p must be any odd number; even radix-8 and 16 FHTs, etc. are not allowed in our MRDHT routines. The Winograd subroutine MRWDHT provides several efficient radix-2, 3, 5, 7, 8, 9 and 16 Winograd transform subroutines.

Times were measured on a Micro-VAX model 3600 computer, each specific length transform has been run 50 times to measure the speed, and the results are averaged and shown in Table 2 and Fig. 3. For the MRDHT subroutine, since general odd radix-p FHT routines are not as efficient as the Winograd short-length transform routines, it turns out to be better to use short radix-3 and 4 fast routines instead of high-power radix-2 and long radix-9 FHTs. However, for the MRWDHT subroutine, it is better to use the long-length Winograd routines as frequently as possible. The results show that the speed is much faster, for single or low powers of prime length, than for many or high powers of prime factors. The rule of thumb for factoring the length N is to let the radix for power-of-prime factors be as large as possible, and the power of each radix be as small as possible. Also we can use 4, 8, 16 and 9-point short length Winograd DHT modules to replace the radix-2 and radix-3 FHTs for reducing the number of multiplications.

Our two mixed-radix FHT programs show comparable performance in terms of speed measurements. For lengths less than 1260, the Winograd program MRWDHT is faster than MRDHT; but for length larger than 1260, the MRDHT becomes faster than the Winograd routine MRWDHT. Each program has its own advantages, such as simplicity of software code or minimum arithmetric complexity required.

8. Conclusions

New efficient radix-3 and odd radix-p FHT algorithms as well as short Winograd DHT modules have been developed, to be both incorporated into a general FHT algorithm. Two mixed radix FHT programs have been implemented as a very general and useful software tool. It allows a much wider selection of transform sizes, and calculates the DHT in-place and in-order. Extensive computer simulations have been run on a Micro-VAX computer to measure the transform speed.

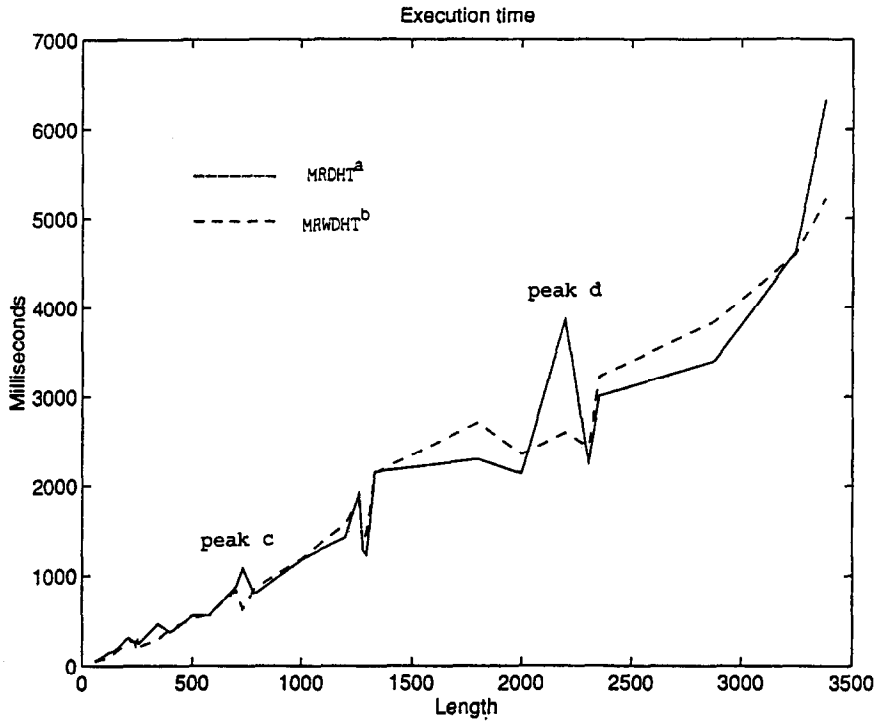

Fig. 3. The execution time for the mixed radix DHTs.

References

[1] N. Anupindi, S.B. Narayanan and K.M.M. Prabhu, "New radix-3 FHT algorithm", *Electronics Lett.*, Vol. 26, 1990, pp. 1537–1538.

[2] R.N. Bracewell, "Discrete Hartley transform", *J. Opt. Soc. Amer.*, Vol. 73, December 1983, pp. 1832–1835.

[3] R.N. Bracewell, "The fast Hartley transform", *Proc. IEEE*, Vol. 72, August 1984, pp. 1010–1018.

[4] R.N. Bracewell, O. Buneman, H. Hao and J. Villasenor, "Fast two-dimensional Hartley transform", *Proc. IEEE*, Vol. 74, 1986, pp. 1282–1283.

[5] C.S. Burrus and P.W. Eschenbacher, "An in-place, in-order prime factor FFT algorithm", *IEEE Trans. Acoust. Speech Signal Process.*, Vol. ASSP-29, August 1981, pp. 806–817.

[6] D.P.K. Lun and W.C. Siu, "On prime factor mapping for the discrete Hartley transform", *IEEE Trans. Acoust. Speech Signal Process.*, Vol. ASSP-40, No. 6, June 1992, pp. 1399–1411.

[7] K.J. Olejniczak and G.T. Heydt, "Scanning the special section on the Hartley transform", *Proc. IEEE*, Vol. 82, March 1994, pp. 372–380.

[8] S.C. Pei and J.L. Wu, "Split-radix fast Hartley transform", *Electronics Lett.*, Vol. 22, January 1986, pp. 26–27.

[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 Process.*, Vol. ASSP-33, October 1985, pp. 1231–1238.

[10] Z.J. Zhao, "In-place radix-3 fast Hartley transform algorithm", *Electronics Lett.*, Vol. 28, 1992, pp. 319–321.