Block diagonal structure in discrete transforms

Ja-Ling Wu

Indexing terms: Mathematical techniques, Matrix algebra, Signal processing

Abstract: The author investigates and summarises some of the computational tasks of discrete transforms in which block diagonal structure plays a dominant role. Walsh-Hadamard transform (WHT) based algorithm designs for various well known discrete transforms are presented; it can be proved that, owing to their block diagonal structure, the WHT based discrete transforms are more efficient than those of the conventional radix-r algorithms for transforms of length $N \leqslant 64$. It is proved that block diagonal structures exist in the running Walsh-Hadamard transform, the running discrete Hartley transform (DHT), and the running discrete cosine transform (OCT).

With regard to block diagonal structure in the transform conversions between DHT and DCT, some existing research results are summarised, and an efficient architecture for generating multiple discrete transform simultaneously is proposed. The two-stage DFT algorithm proposed by Ersoy is extended to that of the DHT and it is proved that two stage DHTs possess a somewhat more interesting 'balanced-block-diagonal' structure. In the context of VLSI system design, two factors are of particular importance: the regularity of processor cells and local communication between processors. The hardware implementation of the block diagonal algorithm, for moderate N, just meets the above requirements. An example of the WHT/DHT is also included.

1 Introduction

Fast transforms are playing an increasingly important role in many practical applications [1]. For example, the DFT, coupled with the existence of FFT algorithms, provides a powerful means for spectral analysis and synthesis [2]. Such transforms are also used for digital filter design and for fast convolution computations [3]. Analogously, the Walsh-Hadamard transform (WHT) has been widely used in computer engineering and digital signal processing for its simple operation and good performance [4]. In recent years there has been a growing interest in the use of the discrete Hartley transform (DHT) in spectral analysis [5]. The DHT has two advanatages over the DFT; namely (i) the forward and the inverse transforms are the same, and (ii) the Hartley transformed outputs are real-valued rather than complex, as with the DFT. Also, the Fourier spectrum can be calculated via the Hartley transform [6].

Paper 6714E (C2/C1), received 26th July 1988

The author is with the Department of Computer Science and Information Engineering, National Taiwan University, Taipei, Taiwan, 10764, Republic of China

Another extensively investigated branch in the field of fast transforms is the computation of discrete cosine transforms (DCTs), because it performs most favourably with the statistically optimal Karhunen-Loeve transform (KLT) of a large number of signal classes [7].

Generally speaking, the computation of discrete transforms can always be presented as a matrix-vector product. If the matrix is of the form shown in Fig. 1, the associated transform is said to possess 'block diagonal structure' (BDS). The sparseness and block independence of the BDS imply the existence of a fast algorithm and the possibility of computational parallelism. This, it is believed, produces the attractiveness of this subject to researchers in signal processing.

2 Block diagonal structure in WHT-based discrete transforms

An N-point discrete transform can be defined as

$$[\bar{X}(k)] = [T(N)][X(n)] \quad 0 \leqslant k, n \leqslant N-1 \tag{1}$$

where $[\bar{X}(k)]$ and $[X(n)]$ denote the input and the output vectors, respectively. For simplicity, the following notations are adopted:

$[WH(N)]$ = N-point transform matrix for WHT

$[DF(N)]$ = N-point transform matrix for DFT

$[DH(N)]$ = N-point transform matrix for DHT

$[DC(N)]$ = N-point transform matrix for DCT

The so-called WHT-based discrete transforms can be interpreted by the matrix decompositions given below:

$$[T(N)] = [A(N)][WH(N)] \tag{2}$$

In other words, the WHT coefficients are computed first, and then used to obtain the discrete transform, $[T(N)]$, coefficients. This is achieved by the transform matrix $[A(N)]$ which is orthonormal and has block-diagonal structure. The reason for the existence of BDS in $[A(N)]$ can be proved as follows.

With the aid of column-row permutations, $[T(N)]$ can always be formulated as

$$[\hat{T}(N)] = \begin{bmatrix} \hat{T}(N/2) & \hat{T}(N/2) \\ B(N/2) & -B(N/2) \end{bmatrix} \tag{3}$$

where '$\hat{}$' stands for an appropriate permutation (which will be detailed later), and $[B(N/2)]$ is a $(N/2) * (N/2)$ submatrix of $\hat{T}(N/2)$. (The decomposibility of $B(N)$ depends heavily on $T(N)$.)

It is well known that

$$[WH(N)] = \begin{bmatrix} WH(N/2) & WH(N/2) \\ WH(N/2) & -WH(N/2) \end{bmatrix} \tag{4}$$

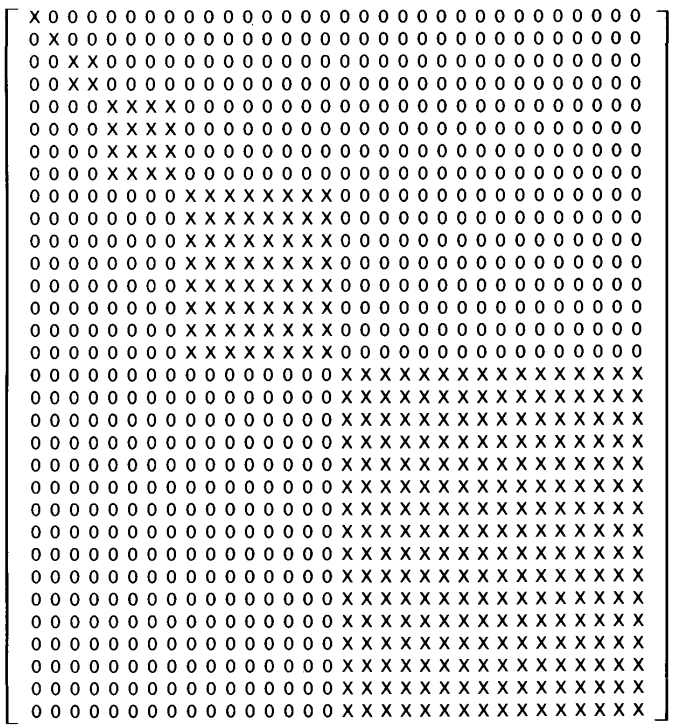

Fig. 1 *Block diagonal structure*

From eqns. 2–4, it follows that

$$[\hat{A}(N)] = (1/N)[\hat{T}(N)][WH(N)]$$

$$= (2/N)\begin{bmatrix} \hat{T}(N/2)WH(N/2) & 0 \\ 0 & B(N/2)WH(N/2) \end{bmatrix} \quad (5)$$

Iterate eqn. 5 ($\log_2 N - 1$) times. $[\hat{A}(N)]$ is of the form shown in Fig. 1, or equivalent to, say, $[\hat{A}(N)]$ in BDS form. The appropriate permutations, for various discrete transforms, required in eqn. 3 are summarised in the following.

(*a*) DFT: Rearranging the row ordering of $[DF(N)]$ in bit-reversed order, we obtain:

$$[\widehat{DF}(N)] = \begin{bmatrix} \widehat{DF}(N/2) & \widehat{DF}(N/2) \\ B(N/2) & -B(N/2) \end{bmatrix} \quad (6)$$

where $[B(N/2)]$ can be regarded as the odd-indexed-part of the DFT kernel. It is easy to verify, by trigonometric computations that

$$[B(N)] = \begin{bmatrix} D(N/2) & -jD(N/2) \\ \tilde{D}(N/2)^* & j\tilde{D}(N/2)^* \end{bmatrix} \quad (7)$$

where '~' stands for the time-inverse of the row ordering and * denotes the complex conjugate operator. With eqns. 6 and 7, after ($\log_2 N - 1$) iterations, it follows that $(1/N)[DF(N)][WH(N)] \triangleq [F(N)]$ (the so-called F-matrix transform)

$$\begin{bmatrix} \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} & & & & & & & \\ & \frac{1}{2}\cdot\begin{bmatrix} 1-j & 1+j \\ 1+j & 1-j \end{bmatrix} & & & & & & \\ & & \frac{1}{4}\cdot\begin{bmatrix} x & y & z & t \\ y & x & t & z \\ y^* & x^* & t^* & z^* \\ x^* & y^* & z^* & t^* \end{bmatrix} & & & & & \\ & & & \frac{1}{8}\cdot\begin{bmatrix} a & b & c & d & e & f & g & h \\ b & a & d & c & f & e & h & g \\ i & j & k & l & m & n & o & p \\ j & i & l & k & n & m & p & o \\ j^* & i^* & l^* & k^* & n^* & m^* & p^* & o^* \\ i^* & j^* & k^* & l^* & m^* & n^* & o^* & p^* \\ b^* & a^* & d^* & c^* & f^* & e^* & h^* & g^* \\ a^* & b^* & c^* & d^* & e^* & f^* & g^* & h^* \end{bmatrix} & \\ & & & & & & & \ddots \end{bmatrix} \quad (8)$$

(*b*) DHT: If we rearrange the row ordering of the DHT transform matrix in bit-reverse order, then the rearranged transform matrix is in the form of eqn. 3. This result has been proved by Hsu and Wu in Reference 8. Similarly, if the columns of the transform matrix are rearranged in bit-reversed order, the rearranged transform matrix is in the form of the transposition of eqn. 3 (eqn. 10 of Reference 9). Thus, it is clear that $(1/N)[DH(N)][WH(N)]$ is of BDS. For example,

$$(1/16)[DH(16)][WH(16)] \triangleq [H(16)]$$

(the H-matrix-transform for $N = 16$)

where

$$[C_1] = [C(4)] = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0.924 & 0.383 \\ 0 & 0 & -0.383 & 0.924 \end{bmatrix} \quad (14)$$

$$[C_2] = \begin{bmatrix} 0.906 & -0.075 & 0.375 & 0.180 \\ 0.213 & 0.768 & -0.513 & 0.318 \\ -0.318 & 0.513 & 0.768 & 0.213 \\ -0.180 & -0.375 & -0.075 & 0.906 \end{bmatrix} \quad (15)$$

$$[C_3] = \begin{bmatrix} 0.902 & -0.018 & -0.074 & -0.037 & 0.374 & -0.070 & 0.179 & 0.089 \\ 0.114 & 0.700 & 0.238 & -0.058 & 0.047 & 0.290 & -0.575 & 0.139 \\ 0.188 & -0.150 & 0.677 & 0.242 & -0.453 & 0.362 & 0.181 & 0.100 \\ 0.092 & 0.203 & -0.149 & 0.715 & -0.223 & -0.491 & -0.062 & 0.304 \\ -0.304 & 0.062 & 0.491 & 0.223 & 0.735 & -0.149 & 0.203 & 0.092 \\ -0.100 & -0.281 & -0.362 & 0.453 & 0.242 & 0.677 & -0.150 & 0.168 \\ -0.139 & 0.575 & -0.290 & -0.047 & -0.058 & 0.238 & 0.700 & 0.114 \\ -0.089 & -0.179 & 0.007 & -0.374 & -0.037 & -0.074 & -0.018 & 0.902 \end{bmatrix} \quad (16)$$

$$\begin{aligned} &= \text{diag}\,[H(8), H_3] \\ &= \text{diag}\,[H(4), H_2, H_3] \\ &= \text{diag}\,[H_1, H_2, H_3] \end{aligned} \quad (9)$$

where

$$[H_1] = [H(4)] = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (10)$$

$$[H_2] = \begin{bmatrix} 0.854 & 0.146 & 0.354 & -0.354 \\ 0.164 & 0.854 & -0.354 & 0.354 \\ 0.354 & -0.354 & 0.854 & 0.146 \\ -0.354 & 0.354 & 0.146 & 0.854 \end{bmatrix} \quad (11)$$

$$[H_3] = \begin{bmatrix} 0.753 & 0.100 & 0.209 & -0.062 & 0.503 & -0.150 & -0.312 & -0.041 \\ 0.100 & 0.753 & -0.062 & 0.209 & -0.150 & 0.503 & -0.041 & -0.312 \\ 0.209 & -0.062 & 0.100 & 0.753 & -0.041 & -0.312 & 0.503 & -0.150 \\ -0.062 & 0.209 & 0.753 & 0.100 & -0.312 & -0.041 & -0.150 & 0.503 \\ 0.312 & 0.041 & 0.150 & -0.503 & -0.062 & 0.209 & 0.753 & 0.100 \\ 0.041 & 0.312 & -0.503 & 0.150 & 0.209 & -0.062 & 0.100 & 0.753 \\ 0.150 & -0.503 & 0.041 & 0.312 & 0.100 & 0.753 & -0.062 & 0.209 \\ -0.503 & 0.150 & 0.312 & 0.041 & 0.753 & 0.100 & 0.209 & -0.062 \end{bmatrix} \quad (12)$$

(*c*) DCT: If the rows of the DCT transform matrix are rearranged in bit-reversed order, and the columns in dyadic order at the same time, then the rearranged transform matrix of DCT (the so-called C-matrix transform) is in the form of eqn. 3. The proof of this can be found in Reference 10. Also, in eqn. 14 of Reference 11, Hou suggested another permutation for DCT which also partitioned the DCT transform matrix into the form of eqn. 3. Thus, the existence of the BDS in C-matrix transform becomes clear. For example,

$$\begin{aligned} [C(16)] &= \text{diag}\,[C(8), C_3] \\ &= \text{diag}\,[C(4), C_2, C_3] \\ &= \text{diag}\,[C_1, C_2, C_3] \end{aligned} \quad (13)$$

3 Block diagonal structure in running dscrete transforms

Whenever the spectrum of sophisticated signals of long duration is desired, for example, the spectrum of a vocal recording, any numerical processing directly involving the discrete transform of the inputs is unrealistic because it is too complex to be recovered with sufficient accuracy from its samples. Furthermore, it is nearly impossible, from both software and hardware points of view, to deal with such a long transform. In order to overcome this difficulty, many algorithms have been proposed to perform the running or sliding DFTs [12–15].

Recently, Stuller [16] introduced the 'generalised running discrete transform' (GRDT) with respect to arbitrary bases. In his paper, Stuller has pointed out that the number of computations performed by the general recursive system in updating transform coefficients depends on the particular transform type in question. He has also shown that the running DFT is the only transform having coefficient update equations that are uncoupled.

In the rest of this section, we will present a unified treatment for the computation of running discrete transforms (RDTs) and show that BDSs exist in the computation of RDTs other than running DFT. Further, the upper-bound of the complexity for computing RDTs can be obtained from the analysis of BDS.

For the sake of consistency, the notations used in Reference 16 are preserved throughout this section. Let $f(n)$ be an arbitrary sequence with long duration and $w(n)$ be

a sliding window of length-N, where we assume $[w(n)] = 1$ and N is a power of two. We define the RDT of $f(n)$ with respect to a nonsingular but otherwise arbitrary $N \times N$ transform matrix, say $T = [t_{pk}]$, as

$$F_k(p) = T[f(n+k)w(n)], \quad n = 0, 1, \ldots, N-1 \tag{17}$$

where t_{pk} is the pkth element of T and k is a positive integer. From eqn. 17, it follows that

$$t_{pk} = \exp\left(-j\frac{2\pi}{N}pk\right) \quad \text{for running DFT (RDFT)}$$

$$= \text{cas}\left(\frac{2\pi}{N}pk\right) \quad \text{for running DHT (RDHT)}$$

$$= \text{wal}\,(p, k) \quad \text{for running WHT (RWHT)}$$

$$= \cos\left(\frac{\pi}{2N}(2k+1)p\right) \quad \text{for running DCT (RDCT)}$$

From References 16 and 17, the general recursion for RDT can be expressed as

$$F_{k+1}(p) = AF_k(p) + [f(t+N) - f(t)]t_{N-1} \tag{18}$$

where A is the so-called 'circular advance matrix' (CAM) associated with T which can be computed as

$$A = TET^{-1} \tag{19}$$

and in general

$$E = \begin{bmatrix} 0 & & \\ \vdots & & I \\ 0 & & \\ \hline 1 & 0 \cdots & 0 \end{bmatrix} \tag{20}$$

where I is the identity matrix and

$$t_k = [t_{0k}, t_{1k}, \ldots, t_{(N-1)k}]' \tag{21}$$

(the prime stands for matrix or vector transposition). Direct evaluation of eqn. 17 requires N^2 multiplications and $N(N-1)$ additions, and direct evaluation of eqn. 18 requires $N^2 + N$ multiplications and $N^2 + 1$ additions. Therefore, there is no benefit in using the recursive algorithm, such as eqn. 18, whenever A is not sufficiently sparse. It will be shown later that, after the same appropriate permutations used in Section 2, A is in BDS form, and therefore the computation of GRDT is bounded by

$$\left(\frac{N^2}{3}\right) \cdot \left(1 - \frac{1}{N^2}\right) - 1 \tag{22}$$

Since WHT, DHT, and DCT are real unitary transforms, it then follows that, for the above four transforms,

$$T^{-1} = \left(\frac{1}{N}\right)T' \tag{23}$$

As in Section 2, after some appropriate permutations, $[\hat{T}(N)]$ is in the form of eqn. 3 and from eqn. 19 we obtain

$$[\hat{A}(N)] = \frac{1}{N}[\hat{T}(N)][E(N)][\hat{T}(N)]' \tag{24}$$

It is interesting to note that

$$[E(N)][\hat{T}(N)]' = [t_1^+, t_2^+, \ldots, t_{N-1}^+, t_0^+]' \tag{25}$$

where $t_k^+ = [t_{k0}, t_{k1}, \ldots, t_{k(N-1)}]$. It is easy to verify that the resultant matrix of eqn. 25 can be expressed by the following generating formula

$$[E(N)][\hat{T}(N)]' = [G(N)] = \begin{bmatrix} G\left(\frac{N}{2}\right) & S\left(\frac{N}{2}\right) \\ G\left(\frac{N}{2}\right) & -S\left(\frac{N}{2}\right) \end{bmatrix} \tag{26}$$

then

$$\begin{aligned}[\hat{A}(N)] &= \frac{1}{N}[\hat{T}(N)][E(N)][\hat{T}(N)]' \\ &= \frac{1}{N}[\hat{T}(N)][G(N)] \\ &= \frac{1}{N}\begin{bmatrix} \hat{T}\left(\frac{N}{2}\right) & \hat{T}\left(\frac{N}{2}\right) \\ B\left(\frac{N}{2}\right) & -B\left(\frac{N}{2}\right) \end{bmatrix} \cdot \begin{bmatrix} G\left(\frac{N}{2}\right) & S\left(\frac{N}{2}\right) \\ G\left(\frac{N}{2}\right) & -S\left(\frac{N}{2}\right) \end{bmatrix} \\ &= \frac{2}{N}\begin{bmatrix} \hat{T}\left(\frac{N}{2}\right) G\left(\frac{N}{2}\right) & 0 \\ 0 & B\left(\frac{N}{2}\right)S\left(\frac{N}{2}\right) \end{bmatrix}\end{aligned} \tag{27}$$

1.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
0.000	−1.000	0.000	0.000	0.000	0.000	0.000	0.000
0.000	0.000	0.000	1.000	0.000	0.000	0.000	0.000
0.000	0.000	−1.000	0.000	0.000	0.000	0.000	0.000
0.000	0.000	0.000	0.000	0.500	0.500	0.500	−0.500
0.000	0.000	0.000	0.000	−0.500	−0.500	0.500	−0.500
0.000	0.000	0.000	0.000	−0.500	−0.500	0.500	0.500
0.000	0.000	0.000	0.000	−0.500	0.500	−0.500	−0.500

Fig. 2 *8 × 8 circular advance matrix of WHT*

1.003	0.000	0.000	0.000	0.000	0.000	0.000	0.000
0.000	−1.003	0.000	0.000	0.000	0.000	0.000	0.000
0.000	0.000	0.000	1.000	0.000	0.000	0.000	0.000
0.000	0.000	−1.000	0.000	0.000	0.000	0.000	0.000
0.000	0.000	0.000	0.000	0.271	0.271	0.653	0.653
0.000	0.000	0.000	0.000	−0.269	−0.653	−0.271	−0.655
0.000	0.000	0.000	0.000	−0.655	−0.271	0.652	0.269
0.000	0.000	0.000	0.000	−0.653	0.653	−0.271	−0.271

Fig. 3 *8 × 8 circular advance matrix of DCT*

1.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
0.000	-1.000	0.000	0.000	0.000	0.000	0.000	0.000
0.000	0.000	0.000	1.000	0.000	0.000	0.000	0.000
0.000	0.000	-1.000	0.000	0.000	0.000	0.000	0.000
0.000	0.000	0.000	0.000	0.707	0.000	0.000	0.707
0.000	0.000	0.000	0.000	0.000	-0.707	-0.707	0.000
0.000	0.000	0.000	0.000	0.000	0.707	0.707	0.000
0.000	0.000	0.000	0.000	-0.707	0.000	0.000	0.707

Fig. 4 *8 × 8 circular advance matrix of DHT*

Since eqn. 27 is a recursive formula, it is obvious that after the iterations of $(\log_2 N - 1)$, $[\hat{A}(N)]$ is a block-diagonal matrix. This completes the proof of our statement of this Section. Examples of CAMs for RWHT, RDCT and RDHT are given in Figs. 2, 3 and 4, respectively.

As mentioned before, eqn. 22 only gives the upper bound of the computation of GRDT. In other words, the decompossibility of $B(N/2)$ and $S(N/2)$ may reduce the complexity further; it is, however, transform dependent.

4 Block diagonal structure in transform conversions

Fig. 5 summarises present progress in the development of the four above-mentioned best known and important fast transforms. The direction of each arrow indicates the direction of transform conversion, FDCT denotes the Fourier-to-cosine transform conversion as summarised by John Markhoul in Reference 18; HDCT stands for the transform conversion from the DHT to the DCT, as suggested by Malvar in Reference 19; FHT/HFT represents the transform conversions between DFT-to-DHT and DHT-to-DFT, respectively, as proposed by Bracewell [20]; $F(N)$, $C(N)$ and $H(N)$ are the transform conversions of WHT-to-DFT, WHT-to-DCT, and WHT-to-DHT, respectively (or equivalently, the WHT-based discrete transforms as discussed in Section 2); CMT denotes the integer-valued approximation of $C(N)$ as given in Reference 21.

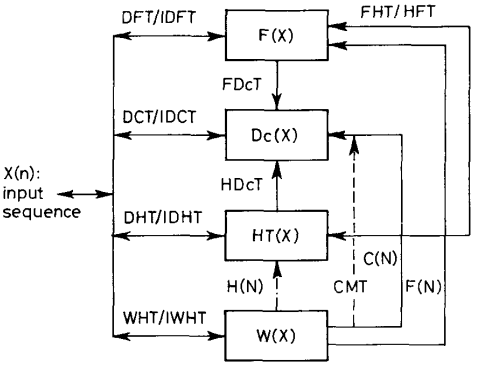

Fig. 5 *Current progress in development of DFTs, DCTs, WHTs and DHTs*

From Fig. 5, one can find the following facts:

Fact 1: If $F(x)$ is known (or has been calculated), the $DC(x)$ and $HT(x)$ can be obtained through FDCT and FHT, respectively.

Since DFT is one of the most fundamental operations in DSP, efficient software and hardware are generally available. Thus, based on the concept of transform conversions, no special investments are required for computing $DC(x)$ and $HT(x)$. However, in this approach, complex operations are required.

Fact 2: If $DC(x)$ is known first, only the DHT can be directly computed by the inverse-HDCT, which can easily be derived by modifying the Malvar's algorithm. $F(x)$ can then be obtained through HFT using only real arithmetics, but the degree of complexity is high.

Fact 3: If $HT(x)$ is known, then $F(x)$ and $DC(x)$ can be obtained through the use of HFT and HDCT respectively. Again, only real operations are required in this approach. It is obvious, from Reference 20, that the FHT involves only additions and subtractions. Based on HDCT and with the aid of bit-reversed ordering, Hou has shown that the transform matrix for converting $HT(x)$ to $DC(x)$ is in block-diagonal form (Reference 9, eqn. 27).

Fact 4: If $W(x)$ is first computed, after some simple additions and subtractions, then all the others can be obtained directly from the transform conversions given in Fig. 5. Section 2 of this paper has shown that BDSs exist for all transform conversions.

From the above four facts and Fig. 5, it follows that BDSs do play an important role in the problem of transform conversions.

5 Block diagonal structure in two-stage DHT

Since Bracewell introduced the DHT as a new member of the family of discrete transforms, much work has been dedicated to the development of fast and better algorithms for DHT. It has been shown that all common FFT algorithms can be equally applicable to the computation of DHT [22–26]. The fast Hartley transform (FHT) algorithms are shown to require the same number of multiplications as, the same storage as, and a few more additions than the real-valued FFT algorithms [24]. However, all these FHT algorithms achieve the complexity reduction at the cost of FFT-liked butterflies which require communications between data points physically far from each other. Thus, they are unsuitable for parallel implementation techniques, such as the systolic and wavefront arrays, which require nearest neighbour interconnections. The WHT-based DHT algorithm, developed in Section 2, can somewhat alleviate this problem by implementing the WHT and the H-matrix transform successively in parallel structures. However, in this approach, the complexity is still in the order of $O(N^2)$ (the actual bound is given in eqn. 22) since the computa-

tional load of a WHT-based DHT depends heavily on the right-bottom block which is of size $N/2 \times N/2$.

To overcome this drawback, an $O(N \log_2 N)$ modified WHT-based DHT algorithm has been proposed by Hsu and Wu [27]. The theme of their approach is to derive a recursive algorithm for computing the high-order DHT coefficients from their lower-order counterparts. However, long range data communication cannot be avoided by using this approach.

Recently, Ersoy [28] proposed a two-stage algorithm for DFT which is based on the vector transformation of sines and cosines into new basis functions using Mobius's inversion of number theory. In this approach, the DFT matrix is factorised into the preprocessing and postprocessing matrices. The preprocessing matrix has elements that are samples of a bipolar rectangular wave function, make up of values 1, −1 and 0. More interestingly, the postprocessing matrix is in a balanced-block-diagonal form (which will be explained later), each block being a circular correlation matrix. These features seem to have the potential of simultaneously satisfying the requirements of very fast speed, low cost of implementation, suitability to VLSI, and high accuracy.

From Reference 28 and Fig. 5, it follows that, if we replace the preprocessing equation, $h(1)$ defined in eqn. 2.3 of Reference 28, by

$$h(1) = \sum_{k=0}^{N-1} x(k)\left[u\left(\frac{1k}{N} + \frac{1}{4}\right) + u\left(\frac{1k}{N}\right)\right] \tag{28}$$

then a very efficient two-stage DHT algorithm can be obtained. For example, the postprocessing matrix for $N = 16$ becomes

$$\begin{bmatrix} H(0) \\ H(8) \\ H(4) \\ H(2) \\ H(10) \\ H(1) \\ H(13) \\ H(9) \\ H(5) \\ H(12) \\ H(14) \\ H(6) \\ H(15) \\ H(3) \\ H(7) \\ H(11) \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & \multicolumn{2}{c}{\multirow{2}{*}{$CC(2)$}} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & & & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & \multicolumn{4}{c}{\multirow{4}{*}{$CC(4)$}} & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & & & & & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & & & & & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & & & & & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \multicolumn{2}{c}{\multirow{2}{*}{$CC(2)$}} & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & & & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \multicolumn{2}{c}{\multirow{2}{*}{$CC(4)$}} \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & & \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \multicolumn{2}{c}{\multirow{2}{*}{$CC(4)$}} \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & & \end{bmatrix} \begin{bmatrix} T(0) \\ T(8) \\ T(4) \\ T(2) \\ T(10) \\ T(1) \\ T(13) \\ T(9) \\ T(5) \\ T(12) \\ T(14) \\ T(6) \\ T(15) \\ T(3) \\ T(7) \\ T(11) \end{bmatrix} \tag{29}$$

where $$[CC(2)] = \begin{bmatrix} b(1, 8) & b(5, 8) \\ b(5, 8) & b(1, 8) \end{bmatrix} \tag{30}$$

and $$[CC(4)] = \begin{bmatrix} b(1, 16) & b(13, 16) & b(9, 16) & b(5, 16) \\ b(5, 16) & b(1, 16) & b(13, 16) & b(9, 16) \\ b(9, 16) & b(5, 16) & b(1, 16) & b(13, 16) \\ b(13, 16) & b(9, 16) & b(5, 16) & b(1, 16) \end{bmatrix} \tag{31}$$

The corresponding preprocessing matrix becomes

$$\begin{bmatrix} T(0) \\ T(8) \\ T(4) \\ T(2) \\ T(10) \\ T(1) \\ T(13) \\ T(9) \\ T(5) \\ T(12) \\ T(14) \\ T(6) \\ T(15) \\ T(3) \\ T(7) \\ T(11) \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & -1 & 1 & -1 & 1 & -1 & 1 & -1 & 1 & -1 & 1 & -1 & 1 & -1 \\ 1 & 1 & -1 & -1 & 1 & 1 & -1 & -1 & 1 & 1 & -1 & -1 & 1 & 1 & -1 & -1 \\ 1 & 2 & 1 & 0 & -1 & -2 & -1 & 0 & 1 & 2 & 1 & 0 & -1 & -2 & -1 & 0 \\ 1 & -2 & 1 & 0 & -1 & 2 & -1 & 0 & 1 & -2 & 1 & 0 & -1 & 2 & -1 & 0 \\ 1 & 2 & 2 & 2 & 1 & 0 & 0 & 0 & -1 & -2 & -2 & -2 & -1 & 0 & 0 & 0 \\ 1 & 0 & -2 & 0 & 1 & 2 & 0 & -2 & -1 & 0 & 2 & 0 & -1 & -2 & 0 & 2 \\ 1 & -2 & 2 & -2 & 1 & 0 & 0 & 0 & -1 & 2 & -2 & 2 & -1 & 0 & 0 & 0 \\ 1 & 0 & -2 & 0 & 1 & -2 & 0 & 2 & -1 & 0 & 2 & 0 & -1 & 2 & 0 & -2 \\ 1 & -1 & -1 & 1 & 1 & -1 & -1 & 1 & 1 & -1 & -1 & 1 & 1 & -1 & -1 & 1 \\ 1 & 0 & -1 & -2 & -1 & 0 & 1 & 2 & 1 & 0 & -1 & -2 & -1 & 0 & 1 & 2 \\ 1 & 0 & -1 & 2 & -1 & 0 & 1 & -2 & 1 & 0 & -1 & 2 & -1 & 0 & 1 & -2 \\ 1 & 0 & 0 & 0 & -1 & -2 & -2 & -2 & -1 & 0 & 0 & 0 & 1 & 2 & 2 & 2 \\ 1 & 2 & 0 & -2 & -1 & 0 & 2 & 0 & -1 & -2 & 0 & 2 & 1 & 0 & -2 & 0 \\ 1 & 0 & 0 & 0 & -1 & 2 & -2 & 2 & -1 & 0 & 0 & 0 & 1 & -2 & 2 & -2 \\ 1 & -2 & 0 & 2 & -1 & 0 & 2 & 0 & -1 & 2 & 0 & -2 & 1 & 0 & -2 & 0 \end{bmatrix} \begin{bmatrix} x(0) \\ x(1) \\ x(2) \\ x(3) \\ x(4) \\ x(5) \\ x(6) \\ x(7) \\ x(8) \\ x(9) \\ x(10) \\ x(11) \\ x(12) \\ x(13) \\ x(14) \\ x(15) \end{bmatrix} \tag{32}$$

It is obvious, from eqn. 32, that the only operations needed in the preprocessing stage are adds and bit shifts, thus this stage can be implemented, easily and efficiently, as a matrix-vector product by using the systolic/wavefront arrays. The postprocessing stage has a balanced-block-diagonal structure; the left-top block is the same as the right-bottom one (refer to eqn. 29). Further, the circular correlation structure of each block of size N, can be realised, with only $O(N)$ multiplications, through the use of number theoretic transforms. In other words, the proposed two-stage approach can achieve an $O(N)$ algorithm for DHT which is the most efficient one among all existing DHT algorithms. Thus, the BDS also plays a dominant role in two-stage representation for discrete transforms.

6 Discussion and conclusions

By using the fast WHT structure for the preprocessing, we can get a fast algorithm for the WHT-based discrete transforms. Table 1 gives a comparison of the computation of $N \leqslant 64$ point DHT via WHT (WHT-based DHT), the radix-2 fast DHT [20] and the split-radix DHT [24]. Generally, for $N \leqslant 64$, the complexities of the WHT-based transform algorithms are almost equal to those of the split-radix algorithm, yet the regular block-diagonal structure and local communication are more suitable for parallel and VLSI implementation. However, the complexity of the block-diagonal $[T]$-transform matrix is proportional to N^2. Thus, it is only benificial for short length ($N \leqslant 64$) computation. In order to alleviate this shortage, a recursive algorithm for computing the high-order transform coefficients from their lower-order ones can be adopted. With the aid of precomputation, the complexity can be reduced to $O(N \log_2 N)$. In other words, the WHT-based algorithm can be viewed as a basic unit of implementation for both software and hardware.

The running discrete transforms are not so popular in the field of DSP because their heavy computational load inhibits the possibility of real-time applications. In order to reduce this shortcoming, we can take the advantage of BDS to reduce the computational load. Better results can be obtained through some specific properties of the discrete transforms, e.g. by the use of shifting properties, order N RDFT and RDHT algorithms can be obtained; furthermore, with the aid of BDS the upper bound of the GRDT algorithm can be easily derived.

From the discussions given in Section 4, one can observe that the BDSs play a major role in the transform conversions. Further, the simple and regular BDS parallelises the whole task. In other words, the BDS is not only a major part of the transform conversions, but also forms the basis of VLSI design for discrete transforms.

The balanced-block-diagonal structure of the two-stage representation of DFT and DHT makes this approach much more appropriate for VLSI and parallel implementation. Furthermore,the two-stage approach with the aid of number theoretic transforms, can achieve an algorithm of order N for DFT or FHT, which is the most efficient among all existing DFT or DHT algorithms.

Most recently, the BDS in the DFT spectrum filtering problem has been discussed in detail by Zarowski *et al.* [29 and 30]. This topic is somewhat similar to the WHT-based transform algorithm design presented in Section 2 and their works re-emphasise that the block diagonal structure in discrete transforms (even in digital signal processing) is an attractive and promising research topic.

7 References

1 ELLIOTT, D.P., and RAO, K.R.: 'Algorithms, analysis and applications' (Academic Press, New York, 1983)
2 GECKINLI, N.C., and YAVUZ, D.: 'Discrete Fourier transformation and its applications to power spectral estimation', *Stud. Electr. Electron. Eng.*, 1983, **8**, pp. 000–000
3 NUSSBAUMER, H.J.: 'Fast Fourier transform and convolution algorithms' (Springer-Verlag, West Germany, 1981)
4 BEAUCHAMP, K.G.: 'Applications of Walsh and related functions' (Academic Press, 1984)
5 BRACEWELL, R.N.: 'The Hartley transform' (Oxford University Press, 1986)
6 BRACEWELL, R.N.: 'Discrete Hartley transform', *J. Opt. Soc. Am.*, 1983, **73**, pp. 1832–1835
7 JAIN, A.K.: 'A sinusoidal family of unitary transforms', *IEEE Trans.*, 1979, **PAMI-1**, pp. 356–365
8 HSU, C.Y., and WU, J.L.: 'Fast computation of discrete Hartley transform via Walsh-Hadamard transform', *Electron. Lett.*, 1987, **23**, pp. 466–468
9 HOU, H.S.: 'The fast Hartley transform', *Proc. IEEE*, 1984, **72**, (8), pp. 1010–1018
10 HSU, C.Y., and WU, J.L.: 'Block-diagonal structure of Walsh-Hadamard/discrete cosine transform', *Electron. Lett.*, 1987, **23**, pp. 1123–1124
11 HOU, H.S.: 'A fast recursive algorithm for computation discrete cosine transform', *IEEE Trans.*, **ASSP-35**, pp. 1455–1461
12 RABINER, L.R., and GOLD, B.: 'Theory and applications of digital signal processing' (Prentice-Hall, NH., 1975)
13 PAPOULIS, A.: 'Signal analysis (McGraw-Hill, New York, 1977)
14 RORTNOFF, M.R.: 'Implementation of the digital phase vocorder using the fast Fourier transform', *IEEE Trans.*, 1976, **ASSP-24**, pp. 243–248
15 ALLEN, J.B., and RABINER, L.R.: 'A unified approach to short-time Fourier analysis and synthesis', *Proc. IEEE*, 1977, **65**, pp. 1158–1164
16 STULLER, J.A.: 'Feneralized running discrete transforms', *IEEE Trans.*, 1982, **ASSP-30**, pp. 60–68
17 WU, J.-L., HSU, C.Y., and CHAO, S.L.: 'On computing the running discrete transforms'. Proceedings of the National Computer Symposium, Taipei, pp. 851–856
18 MAKHOUL, J.: 'A fast cosine transform in one and two dimensions', *IEEE Trans.*, 1980, **ASSP-28**, pp. 27–34
19 MALVAR, H.: 'Fast computation of discrete cosine transform through fast Hartley transform', *Electron. Lett.*, 1986, **20**, pp. 352–353
20 BRACEWELL, R.N.: 'The fast Hartley transform', *Proc. IEEE*, 1984, **72**, pp. 1010–1018
21 KWAK, H.S. *et al.*: 'C-matrix transform', *IEEE Trans.*, 1983, **ASSP-32**, pp. 1304–1307
22 MECKELBURG, H.J., and LIPKA, D.: 'Fast Hartley transform algorithm', *Electron. Lett.*, 1985, **21**, pp. 341–343

Table 1: Comparison of the number of operations for DHT via WHT with split-radix and Bracewell's radix-2

Algorithm *N* operation	Radix-2		Split-radix		DHT via WHT	
	multiplications	additions	multiplications	additions	multiplications	additions
8	4	26	2	22	2	32
16	20	74	12	64	12	126
32	68	194	42	166	52	336
64	196	482	124	416	170	816

23 BUNEMAN, O.: 'Conversion of FFT's to fast Hartley transform', *SIAM J. Sci. Stat. Comput.*, 1986, **7,** (2), pp. 624–638
24 SORENSEN, H.V. *et al.*: 'On computing the discrete Hartley transform', *IEEE Trans.*, 1985, **ASSP-33,** (4), pp. 1231–1238
25 EVANS, D.M.W.: 'An improved digital-reversal permutation algorithm for the fast Fourier and Hartley transforms', *IEEE Trans.*, 1987, **ASSP-35,** (8), pp. 1120–1125
26 DUHAMEL, P., and VETTERLI, M.: 'Improved Fourier and Hartley algorithms: application to cyclic convolution of real data', *IEEE Trans.*, 1987, **ASSP-35,** (6), pp. 818–824
27 HSU, C.Y., and WU, J.L.: 'An order $N \log_2 N$ WHT/DHT algorithm', *Electron. Lett.*, 1988, **24,** pp. 315–316
28 ERSOY, O.K.: 'A two stage representation and its applications', *IEEE Trans.*, 1987, **ASSP-35,** (6), pp. 825–831
29 ZAROWSKI, C.J., and YUNIK, M.: 'Spectral filtering using the fast Walsh transform', *IEEE Trans.*, 1985, **ASSP-33,** pp. 1264–1252
30 ZAROWSKI, C.J.: *et al.*: 'DFT spectrum filtering', *IEEE Trans.*, 1988, **ASSP-36,** pp. 461–470