SRC TR 88-26

Optimal Systolic Designs for the Computation of the Discrete Hartley and the Discrete Cosine Transforms

by

Chaitali Chakrabarti
and Joseph Ja Ja

Optimal Systolic Designs for the Computation of the Discrete Hartley and the Discrete Cosine Transforms [1]

Chaitali Chakrabarti
Department of Electrical Engineering
Systems Research Center
University of Maryland
College Park, MD. 20742

Joseph JáJá
Department of Electrical Engineering
Institute for Advanced Computer Studies
Systems Research Center
University of Maryland
College Park, MD. 20742

Abstract

In this paper, we propose new algorithms for computing the Discrete Hartley and the Discrete Cosine Transform. The algorithms are based on iterative applications of the modified small n algorithms of DFT. The one dimensional transforms are mapped into two dimensions first and then implemented on two dimensional systolic arrays. Pipelined bit serial architectures operating on left to right LSB to MSB binary arithmetic is the basis of the hardware design. Different hardware schemes for implementing these transforms are studied. We show that our schemes achieve a substantial speed-up over existing schemes.

[1]Supported in part by NSA Contract No. MDA-904-85H-0015, NSF Grant No. DCR-86-00378 and by the Systems Research Center Contract No. OIR-85-00108

1 Introduction

The Discrete Cosine Transform (DCT) has emerged as the most popular transform for many coding schemes. This is due to the fact that, for a wide class of signals, it performs very closely to the statistically optimal Karnuhen-Loeve Transform (KLT). Its energy packing efficiency is also greater than any other transform. Since the DCT block is an integral part of any image processing scheme, a study of its implementation in VLSI is very important. Another popular scheme is the Discrete Hartley Transform (DHT) which may replace the Discrete Fourier Transform (DFT) in many spectral analysis and digital processing schemes. The advantage is that the computation of this transform does not require complex arithmetic and could be done much faster.

There exist many methods for computing the one-dimensional DCT. They are either based on direct factorisation of the DCT matrix [Lee], [CHF], or on a rotation of the output of a Fourier [NP], [VN] or Hartley [Mal] transform. There are many methods [Br], [PW], [SJBH], [Hou] for computing fast DHT in one dimension too. However, there are no known VLSI implementations of these algorithms. The straightforward approach would be to implement these algorithms on a linear array of inner product processors. The result is complicated control and arithmetic units.

In this paper we propose new algorithms to compute DHT and DCT along with their implementations on fully pipelined bit serial two dimensional systolic arrays. We map the one dimensional transforms into two dimensions such that N computations are roughly replaced by $\sqrt{N}$ subcomputations which can be done fully in parallel. In the computation of DCT(N), we suggest four schemes based on whether DFT($\sqrt{N}$) or DHT($\sqrt{N}$) is used for the subcomputations on the rows and on the columns. A comparison of these schemes lead us to claim that the scheme based on DHT-DHT is the best. Moreover, we have improved upon the method suggested by Sorensen et.al. to compute the two-dimensional DHT. The hardware

design of all these consists of a few types of regular and modular components. The compatibility of the input/output characteristics makes the interconnections between the components simple. The delay is small and the throughput is very high.

The rest of the paper is organized as follows. In Section 2, we discuss the index mappings for two-dimensional DFT, DHT and DCT and the various schemes for their computation. Sections 3 and 4 deal with the systolic array implementation of the proposed schemes. We introduce the word design in Section 3 and the bit serial design in Section 4. We conclude the paper in Section 5 with a comparison of the various schemes for computing DHT and DCT and an estimate of the sizes and delays of the various components.

2 New Two Dimensional Schemes

Any one dimensional linear transform $f(k) = \sum_n a(k,n)x(n)$,
$0 \leq n, k \leq N-1$, can be mapped into a two dimensional transform. Let $N = N_1 N_2$ and let

$$n = K_1 n_1 + K_2 n_2 \bmod N = K_1 n_1 \bmod N + K_2 n_2 \bmod N - a_{n_1 n_2} N \quad (1)$$

$$k = K_3 k_1 + K_4 k_2 \bmod N = K_3 k_1 \bmod N + K_4 k_2 \bmod N - b_{k_1 k_2} N \quad (2)$$

where K_1, K_2, K_3, K_4 are integer constants, $a_{n_1 n_2}, b_{k_1 k_2}$ are either 0 or 1,
$0 \leq n_1, k_1 \leq N_1 - 1$, $0 \leq n_2, k_2 \leq N_2 - 1$.
Then $f(k)$ can be written in the form

$$\hat{f}(k_1, k_2) = \sum_{n_2} \sum_{n_1} a(k_1, k_2, n_1, n_2) \hat{x}(n_1, n_2). \quad (3)$$

The main motive behind this transformation is to map the computation into a systolic two dimensional array of size $N_1 \times N_2$. The number of systolic steps will be drastically reduced from $O(N)$ to $O(N_1 + N_2)$. The success of this transformation depends on choosing K_1, K_2, K_3, K_4 in such a way that equation (3) can indeed

be computed by one or few $N_1 \times N_2$ systolic arrays. Our goal in this section is to establish such transformations for the one dimensional Discrete Fourier Transform (DFT), Discrete Hartley Transform (DHT) and Discrete Cosine Transform (DCT). We will also sketch the procedures to execute the two dimensional computations. The detailed calculations are included in Appendices 1,2,3 and 4. We have modified an existing scheme for DHT. The schemes for DCT have not appeared in the literature before.

2.1 Discrete Fourier Transform

The one dimensional Discrete Fourier Transform of N points, DFT(N), is defined by

$$F(k) = \frac{1}{\sqrt{N}} \sum_{n=0}^{N-1} x(n) W_N^{nk}, \qquad 0 \leq k \leq N, \tag{4}$$

where $W_N = exp(j\frac{2\pi}{N}nk)$. Equation (4) can then be reduced to

$$\hat{F}(k_1, k_2) = \frac{1}{\sqrt{N}} \sum_{n_1} \sum_{n_2} \hat{x}(n_1, n_2) W_N^{nk}, \tag{5}$$

where n and k are defined as in (1) and (2). The choice of $K_1 = N_2$, $K_2 = N_1$, $K_3 = (N_2^{-1} \bmod N_1)N_2$, $K_4 = (N_1^{-1} \bmod N_2)N_1$, where $N = N_1 N_2$ and N_1, N_2 are mutually prime, gives a two dimensional DFT of the form

$$\hat{F}(k_1, k_2) = \frac{1}{\sqrt{N}} \sum_{n_2} \left\{ \sum_{n_1} \hat{x}(n_1, n_2) W_{N_1}^{n_1 k_1} \right\} W_{N_2}^{n_2 k_2} \tag{6}$$

The derivation is given in [Bu]. It is clear that expression (6) consists of a set of column DFT's followed by a set of DFT's on the rows.

We now turn to the one dimensional DFT computation of columns or rows. The most popular method of computing DFT is by Fast Fourier Transform [CT] which uses $O(N \log N)$ operations as opposed to $O(N^2)$ operations required by algorithms for DFT(N). Recently, faster algorithms, based on the so called small n algorithms, have been derived by Winograd and others [AC], [Gd], [Sil], [Win]. For small values of N, these algorithms use the fewest number of multiplications.

Each such algorithm can be expressed as $Y = S_N C_N T_N \hat{x}$, where S_N is an $N \times J$ incidence matrix, T_N is a $J \times N$ incidence matrix, C_N is a $J \times J$ diagonal matrix with purely real or purely imaginary entries, $J \approx N$. Therefore, a two dimensional DFT of size $N_1 \times N_2$ can be expressed as

$$Y = S_{N_2} C_{N_2} T_{N_2} \left(S_{N_1} C_{N_1} T_{N_1} x\right)^T . \tag{7}$$

[OJ] have provided a systolic implementation based on three types of components, namely summation component ($Z = TX$), scaling component ($Z = CX$), and transpose component ($Z = X^T$). The interconnection is shown in Fig.1.

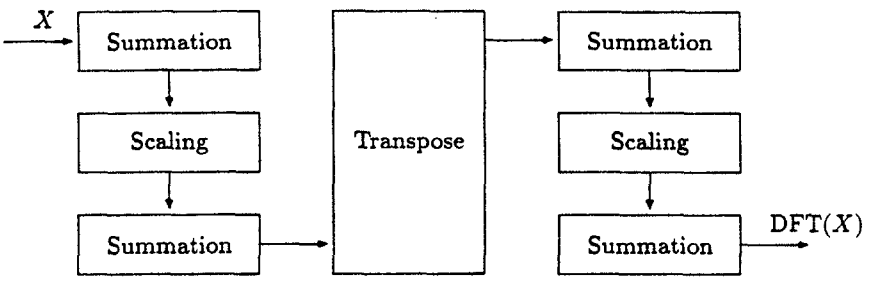


Fig.1 Interconnection of the various components for DFT

2.2 Discrete Hartley Transform

The one dimensional Discrete Hartley Transform of N points, DHT(N), is defined by

$$H(k) = \frac{1}{\sqrt{N}} \sum_{n=0}^{N-1} x(n) cas \frac{2\pi}{N} nk, \qquad 0 \le k \le N-1, \tag{8}$$

where $cas\,(x) = cos\,(x) + sin\,(x)$.

If we define two dimensional arrays $\hat{H}$ and $\hat{x}$, (8) can be reduced to

$$\hat{H}(k_1, k_2) = \frac{1}{\sqrt{N}} \sum_{n_1} \sum_{n_2} \hat{x}(n_1, n_2) cas \frac{2\pi}{N} nk, \tag{9}$$

where n and k are defined as in (1) and (2). If $N = N_1 N_2$ and N_1, N_2 are relatively prime, a choice of $K_1 = N_2$, $K_2 = N_1$, $K_3 = (N_2^{-1} \bmod N_1) N_2$,
$K_4 = (N_1^{-1} \bmod N_2) N_1$ gives a two dimensional DHT of the form

$$\hat{H}(k_1,k_2) = \frac{1}{\sqrt{N}}\sum_{n_2}\left\{\sum_{n_1}\hat{x}(n_1,n_2)cas\,\frac{2\pi}{N_1}n_1k_1\right\}cas\,\frac{2\pi}{N_2}n_2k_2$$
$$-\frac{2}{\sqrt{N}}\sum_{n_2}\left\{\sum_{n_1}\hat{x}(n_1,n_2)sin\,\frac{2\pi}{N_1}n_1k_1\right\}sin\,\frac{2\pi}{N_2}n_2k_2. \qquad (10)$$

The derivation is given in Appendix 1A. From expression (10) it is obvious, that this is not a simple DHT of the columns of $\hat{x}(n_1,n_2)$ followed by a DHT of the rows. Let $A(k_1,n_2) = \frac{1}{\sqrt{N}}\sum_{n_1}\hat{x}(n_1,n_2)cas\,\frac{2\pi}{N_1}n_1k_1$, and
$B(k_1,n_2) = \frac{1}{2}[A(k_1,n_2) + A(N_1-k_1,n_2) + A(k_1,N_2-n_2) - A(N_1-k_1,N_2-n_2)]$, then

$$\hat{H}(k_1,k_2) = \sum_{n_2}B(k_1,n_2)cas\,\frac{2\pi}{N_2}n_2k_2. \qquad (11)$$

Thus two dimensional DHT can be obtained [SJBH] by computing the DHT of the columns followed by some adjustments to produce $B(k_1,n_2)$ and then computing the DHT of the rows. For details see Appendix 1B.

One dimensional DHT can be produced by various schemes. We choose the Winograd-Hartley transform algorithm since the number of multiplications is minimum. The algorithm can be expressed as $H = S_N\hat{C}_NT_N\hat{x}$, where $\hat{C} = Re[C_N] - Im[C_N]$, S_N, T_N, C_N are the matrices appearing in DFT(N). Since C_N contains purely real or purely imaginary entries, $\hat{C}_N$ differs from C_N in that the imaginary entries are negated. Sorensen et.al. [SJBH] claim that they can compute this replacement in place with eight additions for four output points. In Appendix 4, we show how matrices S_N and T_N can be modified and how the rows and columns of $\hat{x}$ can be permuted so that the replacement can be computed with four additions for four output points. Two dimensional DHT can be computed using summation, scaling, transpose and adder components. The interconnection is shown in Fig.2A.

A simpler scheme would be to compute two dimensional DFT and then subtract the imaginary part from the real part to get two dimensional DHT. Since we already have good schemes to compute two dimensional DFT, the only hardware

extension necessary is a row of subtractors. The interconnection is shown in Fig.2B. The disadvantage of this method is that we are not utilising the fact that DHT is a real transform, instead, we are resorting to complex adders and multipliers and then with the help of another subtractor unit retrieving DHT. In the next two sections, we will see that this scheme fares poorly compared to the first scheme both in terms of hardware as well as delay.

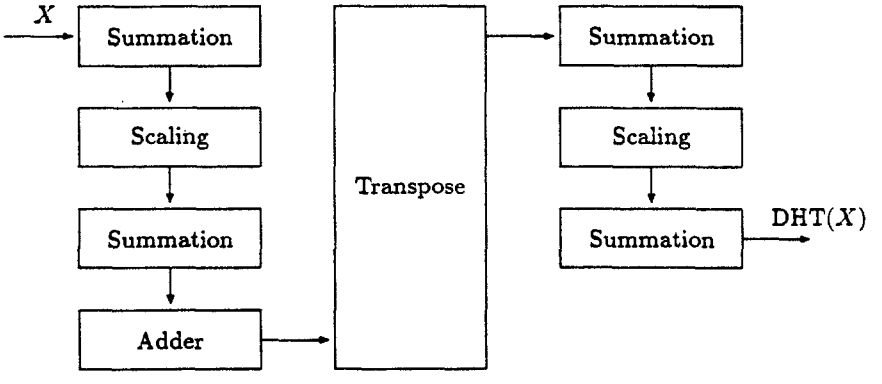


Fig.2A Interconnection of the various components for Scheme 1 of DHT

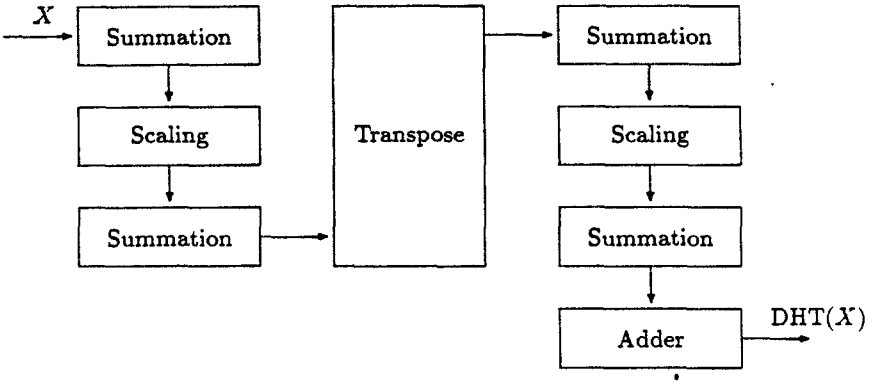


Fig.2B Interconnection of the various components for Scheme 2 of DHT

2.3 Discrete Cosine Transform

The one dimensional Discrete Cosine Transform of N points, DCT(N), is defined [ANR] by

$$\begin{aligned} C(k) &= \frac{2}{N}\sum_{n=0}^{N-1} x(n)cos\left[\frac{\pi}{2N}(2n+1)k\right], \quad 1 \le k \le N-1 \\ C(0) &= \frac{\sqrt{2}}{N}\sum_{n=0}^{N-1} x(n). \end{aligned} \tag{12}$$

We define a new series $y(n)$ by

$$\begin{aligned} y(n) &= x(2n) \qquad\qquad 0 \le n \le \frac{N}{2}-1 \\ &= x(2N-2n-1) \qquad \frac{N}{2} \le n \le N-1. \end{aligned} \tag{13}$$

With this substitution equation (12) reduces to

$$C(k) = \frac{2}{N}\sum_{n=0}^{N-1} y(n)cos\left[\frac{\pi}{2N}(4n+1)k\right] \tag{14}$$

$$C(0) = \frac{\sqrt{2}}{N}\sum_{n=0}^{N-1} y(n). \tag{15}$$

Applying the same procedure as in Sections 2.1 and 2.2, we define two dimensional arrays $\hat{C}(k_1,k_2)$ and $\hat{y}(n_1,n_2)$. Equation (14) then reduces to

$$\begin{aligned} \hat{C}(k_1,k_2) &= \frac{2}{N}\sum_{n_2}\sum_{n_1}\hat{y}(n_1,n_2)cos\left[\frac{\pi}{2N}(4n+1)k\right], \\ \hat{C}(0,0) &= \frac{\sqrt{2}}{N}\sum_{n_2}\sum_{n_1}\hat{y}(n_1,n_2), \end{aligned} \tag{16}$$

where n and k are defined as in (1) and (2). Since computation of $\hat{C}(0,0)$ is straightforward, we shall not treat it here. We define two functions $s(k_1)$ and $t(k_2)$ as follows $s(k_1) = (K_3k_1 \bmod N)/(2N_2k_1)$, $t(k_2) = (K_4k_2 \bmod N)/(2N_1k_2)$.
If N_1 and N_2 are mutually prime to each other, a choice of $K_1 = N_2, K_2 = N_1$, $K_3 = (N_2^{-1} \bmod N_1)N_2, K_4 = (N_1^{-1} \bmod N_2)N_1$ gives a two dimensional DCT

$$\hat{C}(k_1,k_2) = \frac{2}{N}\sum_{n_2}\sum_{n_1}\hat{y}(n_1,n_2)cos\left[\frac{\pi}{N_1}(2n_1+s(k_1))k_1 + \frac{\pi}{N_2}(2n_2+t(k_2))k_2 - \frac{\pi}{2}b_{k_1k_2}\right]. \tag{17}$$

The derivation is included in Appendix 2.

When $b_{k_1 k_2} = 0$

$$\begin{aligned}\hat{C}(k_1,k_2) &= \frac{2}{N}\sum_{n_2}\left\{\sum_{n_1}\hat{y}(n_1,n_2)cos\left[\frac{\pi}{N_1}(2n_1+s(k_1))k_1\right]\right\}cos\left[\frac{\pi}{N_2}(2n_2+t(k_2))k_2\right] \\ &- \frac{2}{N}\sum_{n_2}\left\{\sum_{n_1}\hat{y}(n_1,n_2)sin\left[\frac{\pi}{N_1}(2n_1+s(k_1))k_1\right]\right\}sin\left[\frac{\pi}{N_2}(2n_2+t(k_2))k_2\right].\end{aligned} \tag{18}$$

When $b_{k_1,k_2} = 1$

$$\begin{aligned}\hat{C}(k_1,k_2) &= \frac{2}{N}\sum_{n_2}\left\{\sum_{n_1}\hat{y}(n_1,n_2)cos\left[\frac{\pi}{N_1}(2n_1+s(k_1))k_1\right]\right\}sin\left[\frac{\pi}{N_2}(2n_2+t(k_2))k_2\right] \\ &+ \frac{2}{N}\sum_{n_2}\left\{\sum_{n_1}\hat{y}(n_1,n_2)sin\left[\frac{\pi}{N_1}(2n_1+s(k_1))k_1\right]\right\}cos\left[\frac{\pi}{N_2}(2n_2+t(k_2))k_2\right].\end{aligned} \tag{19}$$

Thus the two dimensional DCT can be obtained by applying one dimensional transforms on columns followed by one dimensional transforms on rows. Though the expressions for $b_{k_1,k_2} = 0$ or 1, seem very different, they are actually not so. We will show in Section 3, how a small manipulation in the last phase is enough to realise the two different expressions.

When $b_{k_1,k_2} = 0$

$$\hat{C}(k_1,k_2) = \frac{2}{N}\left\{DC\acute{T}_{n_2}\left[DC\acute{T}_{n_1}[\hat{y}(n_1,n_2)]\right] - DS\acute{T}_{n_2}\left[DS\acute{T}_{n_1}[\hat{y}(n_1,n_2)]\right]\right\} \tag{20}$$

When $b_{k_1,k_2} = 1$,

$$\hat{C}(k_1,k_2) = \frac{2}{N}\left\{DS\acute{T}_{n_2}\left[DC\acute{T}_{n_1}[\hat{y}(n_1,n_2)]\right] + DC\acute{T}_{n_2}\left[DS\acute{T}_{n_1}[\hat{y}(n_1,n_2)]\right]\right\}, \tag{21}$$

where $DC\acute{T}(DS\acute{T})$ differs from DCT(DST) in the cosine(sine) argument.

We will first discuss the popular methods of computing one dimensional DCT. These methods use either one dimensional DFT or one dimensional DHT on permuted data. Let

$$Y_F(k) = \frac{1}{\sqrt{N}}\sum_{n=0}^{N-1} y(n)W_N^{nk} \tag{22}$$

where $y(n)$ is defined as in equation (13). $C(k)$ can then be expressed as

$$C(k) = \frac{2}{\sqrt{N}} Re\left[e^{-\frac{j\pi k}{2N}} Y_F(k)\right]. \tag{23}$$

Thus a computation of one dimensional DFT followed by a complex multiplication gives one dimensional DCT. Let

$$Y_H(k) = \frac{1}{\sqrt{N}} \sum_{n=0}^{N-1} y(n) cas \frac{2\pi}{N} nk, \tag{24}$$

then

$$C(k) = \frac{1}{\sqrt{N}} \left[Y_H(k) cas \left(-\frac{k\pi}{2N}\right) + Y_H(N-k) cas \left(\frac{k\pi}{2N}\right)\right]. \tag{25}$$

Thus a computation of one dimensional DHT followed by real multiplications gives one dimensional DCT.

Let $Z_C(k) = DC\acute{T}_n = \sum_n y(n) cos \frac{\pi}{N}(2n + g(k))k$ and
$Z_S(k) = DS\acute{T}_n = \sum_n y(n) sin \frac{\pi}{N}(2n+g(k))k$, where $g(k)$ is a predetermined function. We need to know how to compute them in order to be able to compute the two dimensional DCT. Both DFT and DHT can be used for this computation.
If we use DFT then,

$$\begin{aligned} Z_C(k) &= Re\left[e^{-\frac{j\pi}{N} kg(k)} Y_F(k)\right] \\ Z_S(k) &= -Im\left[e^{-\frac{j\pi}{N} kg(k)} Y_F(k)\right]. \end{aligned} \tag{26}$$

If we use DHT then,

$$\begin{aligned} Z_C(k) &= \frac{1}{2}\left[Y_H(k) cas \left(-\frac{\pi}{N} kg(k)\right) + Y_H(N-k) cas \left(\frac{\pi}{N} kg(k)\right)\right] \\ Z_S(k) &= \frac{1}{2}\left[Y_H(k) cas \left(\frac{\pi}{N} kg(k)\right) - Y_H(N-k) cas \left(-\frac{\pi}{N} kg(k)\right)\right]. \end{aligned} \tag{27}$$

For systolic implementation, it is necessary for $Y_H(k)$ and $Y_H(N-k)$ to be adjacent so that $Z_C(k)$ and $Z_S(k)$ can be computed with constant time delay, irrespective of the value of k. In Appendix 3, we show how to modify matrix,S_N, so that this is possible.

Since two dimensional DCT is a function of $Z_C(k_1)$, $Z_S(k_1)$, $Z_C(k_2)$, $Z_S(k_2)$, we can use either one dimensional DFT or DHT on the rows as well as on the columns and then adjust using multipliers and adders.

We propose the following algorithm

- Compute DFT or DHT on columns.
- Adjust using complex or real multipliers and adders.
- Compute DFT or DHT on rows.
- Adjust using complex or real multipliers and adders.

The interconnection is shown in Fig.3. There are thus four possible schemes based on whether DFT or DHT is used on columns and on rows, DFT-DFT, DFT-DHT, DHT-DFT, DHT-DHT. We shall discuss them in the next section. Since both DCT and DHT are real transforms, we would expect to do better if we used DHT to compute DCT. In the subsequent sections we will show that this is true.

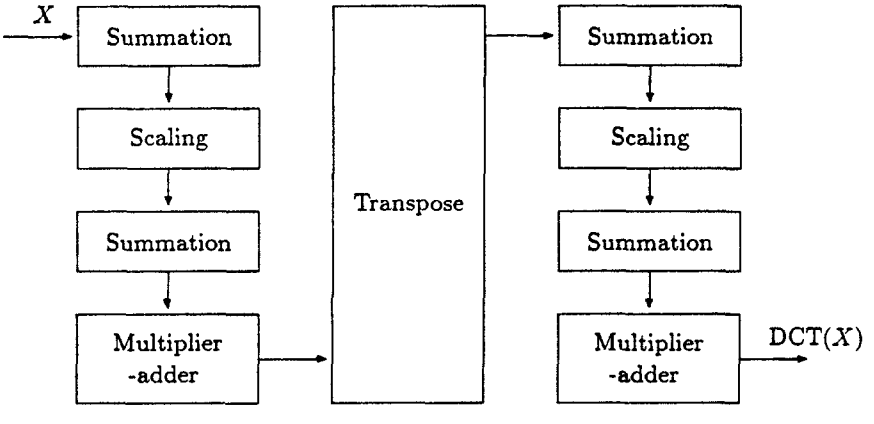


Fig.3 Interconnection of the various components for DCT

3 Systolic Implementations

In this section we discuss the systolic array implementations of the two dimensional schemes of DFT, DHT and DCT. The scheme [OJ] for DFT uses only three types of components, summation, scaling and transpose. Our schemes for DHT and DCT use these components along with special type of multipliers and adders. Here we present the functional definitions and the data flow through the various components.

3.1 DFT implementation

Summation Component: This component performs the operation $Z = SX$, where S is an incidence matrix of size $M \times N_1$, X is the input matrix of size $N_1 \times N_2$, and Z is the output matrix of size $M \times N_2$. Since S is known, the elements of S can be directly embedded. The nature of S makes it possible for the summation component to be constructed with three different types of subcomponents, addition ($s_{ij} = 1$), delay ($s_{ij} = 0$) and subtraction ($s_{ij} = -1$) subcomponents. The functional description of each subcomponent is given in Fig.4.

$$
\begin{array}{lll}
\textit{Addition} & : & X_{out}(i,j) = X_{in}(i,j) \\
\textit{Subcomponent} & & Z_{out}(i,j) = Z_{in}(i,j) + X_{in}(i,j) \\
 & & \\
\textit{Delay} & : & X_{out}(i,j) = X_{in}(i,j) \\
\textit{Subcomponent} & & Z_{out}(i,j) = Z_{in}(i,j) \\
 & & \\
\textit{Subtraction} & : & X_{out}(i,j) = X_{in}(i,j) \\
\textit{Subcomponent} & & Z_{out}(i,j) = Z_{in}(i,j) - X_{in}(i,j)
\end{array}
$$

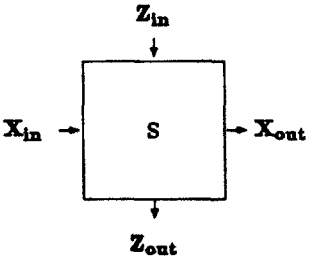


Fig.4 Function description of the various subcomponents

X_{in} and Z_{in} are the values prior to some clocking, while X_{out} and Z_{out} are the values generated after the clocking. The three different types of subcomponents are interconnected as shown in Fig.5.

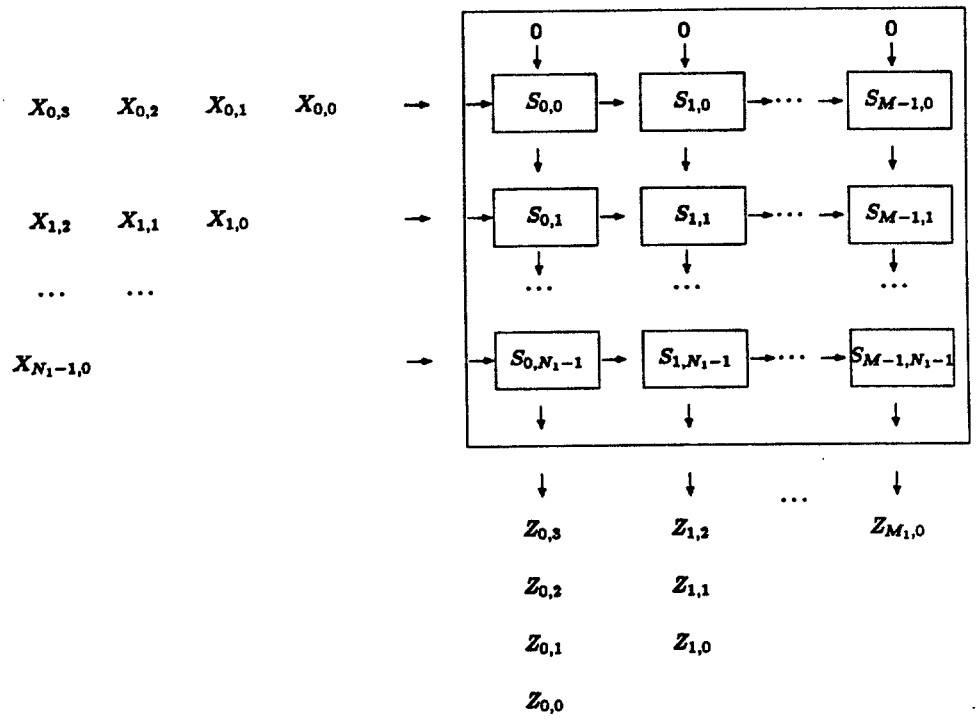


Fig.5 Data flow through the summation component

Scaling component : This component performs the operation $Z = CX$, where C is a diagonal matrix of size $N_1 \times N_1$, X and Z are the input and output matrices of size $N_1 \times N_2$. Since the elements C_{ij} are known beforehand, they can be built into this component. The function description of this subcomponent is $Z_{out}(i,j) = C(i,i)X_{in}(i,j)$.

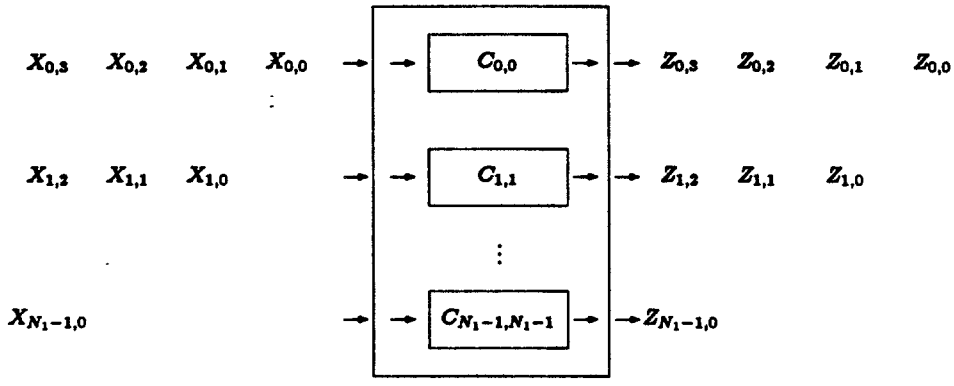


Fig.6 Data flow through the scaling component

A linear array of such subcomponents form this component. The skewed data through this component are shown in Fig.6. Notice that the data flow is consistent with that of the summation component.

Transpose component: This component performs the operation $Z = X^T$, where X and Z are the input and output matrices of sizes $N_1 \times N_2$ and $N_2 \times N_1$ respectively. The functional description of each subcomponent is given in Fig.7.

if $C(i,j) > 0$ then

$X_{out}(i,j) = Z_{out}(i,j) = Y_{out}(i,j) = X_{in}(i,j)$

else if $C(i,j) < 0$ then

$X_{out}(i,j) = Z_{out}(i,j) = Y_{out}(i,j) = Y_{in}(i,j)$

else

$X_{out}(i,j) = Z_{out}(i,j) = Y_{out}(i,j) = Z_{in}(i,j)$

endif

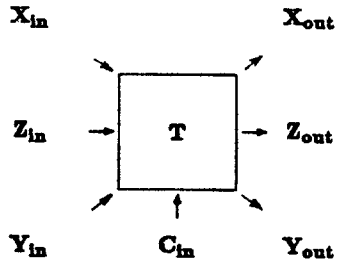


Fig. 7 Function description of transpose subcomponent

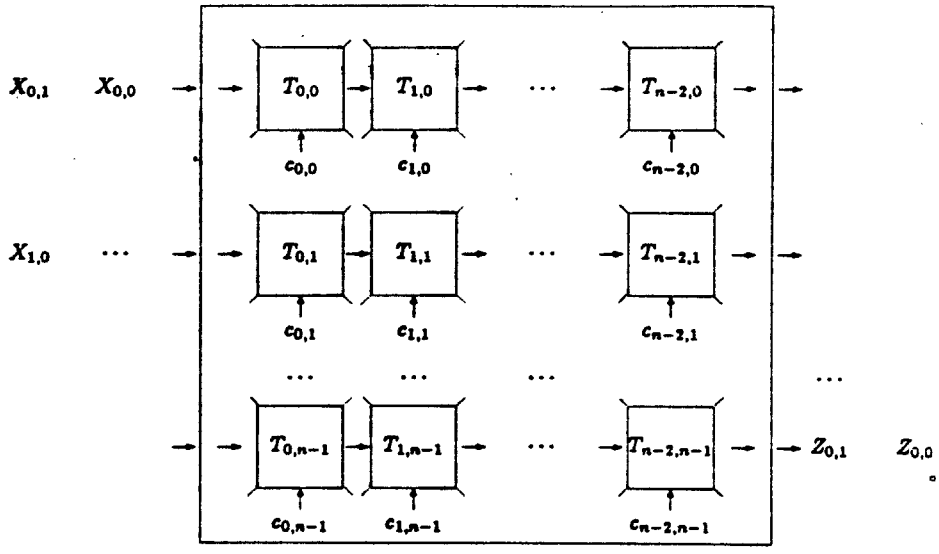


Fig.8 Data flow through the transpose component

The control flag $C_{i,j}$ are set according to

if $(k-j) < (n-1) - j$ then $C_{i,j} = 1$

else if $(k-j) > (n-1) + j$ then $C_{i,j} = -1$

else $C_{i,j} = 0$

where $n = max(N_1, N_2)$ and k is the clocking instant. The subcomponents are connected as shown in Fig.8.

3.2 DHT implementation

We have seen in Section 2.2 that two dimensional DHT can be computed using summation, scaling, transpose and adder units. The input $\hat{x}$ is formed by permuting the columns of x. The second summation component of Fig.2A does not have the elements of S_{N_1} embedded in it, instead, the elements are those of the modified S_{N_1}, $\acute{S}_{N_1}$, as explained in Appendix 4. Let $\acute{T}_{N_2}$ be obtained by permuting the columns of T_{N_2}, such that $\acute{T}_{N_2}(i,j)$ and $\acute{T}_{N_2}(i, N_2 - j)$ are adjacent to each other for $1 \leq j \leq N_2 - 1$. The third summation component of Fig.2A has the elements of $\acute{T}_{N_2}$ embedded instead of those of T_{N_2}. The adder component in Scheme 1 (Fig.2A) consists of N_1 subcomponents, each of which compute either addition or subtraction. The input to this component is the matrix $\hat{A}$ fed in a skewed manner and the output is the matrix B. They are related as follows.

$$
\begin{array}{lcl}
B(i,j) & = & \hat{A}(i,j) + \hat{A}(N_1 - i, j) \\
B(N_1 - i, j) & = & \hat{A}(i,j) - \hat{A}(N_1 - i, j) \\
B(i, N_2 - j) & = & \hat{A}(i, N_2 - j) + \hat{A}(N_1 - i, N_2 - j) \\
B(N_1 - i, N_2 - j) & = & \hat{A}(i, N_2 - j) - \hat{A}(N_1 - i, N_2 - j)
\end{array}
$$

for $1 \leq i \leq N_1 - 1$ and $1 \leq j \leq N_2 - 1$.

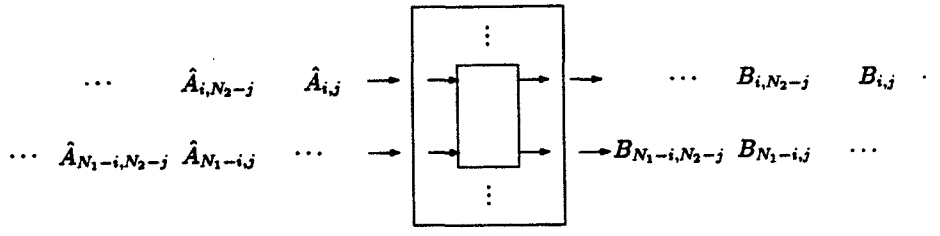
$B(0,j) = \hat{A}(0,j)$ for $0 \leq j \leq N_2 - 1$ and $B(i,0) = \hat{A}(i,0)$ for $0 \leq i \leq N_1 - 1$. The data flow is shown in Fig.9.

The components of Scheme 2 (Fig.2B) are identical to those of DFT, except for the adder in the last stage. The adder component consists of a linear array of N_2

subcomponents each of which perform the following operation

$$\hat{H}(i,j) = Y_{F_R}(i,j) - Y_{F_I}(i,j),$$

where $Y_{F_R}(i,j)$ and $Y_{F_I}(i,j)$ are the real and imaginary components of the DFT output $Y_F(i,j)$. The data flow is shown in Fig.10.



Data flow through the ith adder subcomponent of Scheme 1

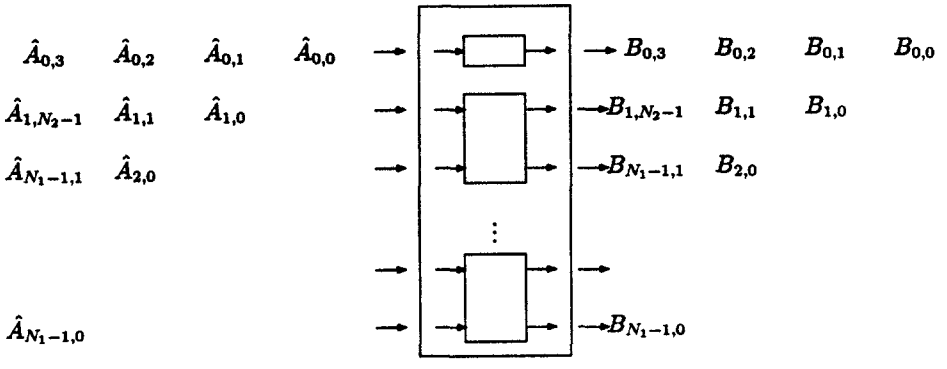


Fig.9 Data flow through the adder component of Scheme 1

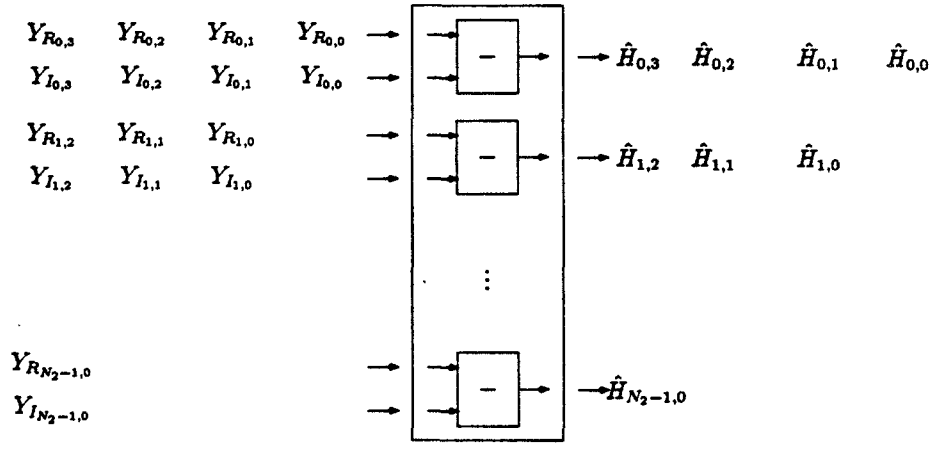


Fig.10 Data flow through the adder component of scheme 2

3.3 DCT implementation

The algorithm for the DCT implementation can be split into two phases, the first phase consists of the computation of DFT or DHT on columns followed by adjustment using multipliers and adders while the second phase consists of the computation of DFT or DHT on rows followed by adjustment using multipliers and adders. The four schemes are based on whether DFT or DHT is used to operate on the rows and on the columns. Since the computation of DFT and DHT over rows and columns have already been discussed in the previous sections, we discuss here only the designs of the various multiplier-adder units . We then proceed to give a short description of each of the four schemes.

Multiplier-adder type A :

The input to this component consists of complex data formed after computing DFT on columns. The function description of the ith component is as follows

$$Z_C(i,j) = X_R(i,j)cos\,\alpha - X_I(i,j)sin\,\alpha$$

$$Z_S(i,j) = X_R(i,j)sin\,\alpha + X_I(i,j)cos\,\alpha,$$

where $\alpha = \frac{\pi}{N_1} is(i)$ and X_R and X_I are the real and imaginary parts of the input data (see equation(26)). The skewed data flow through this component is shown in Fig.11.

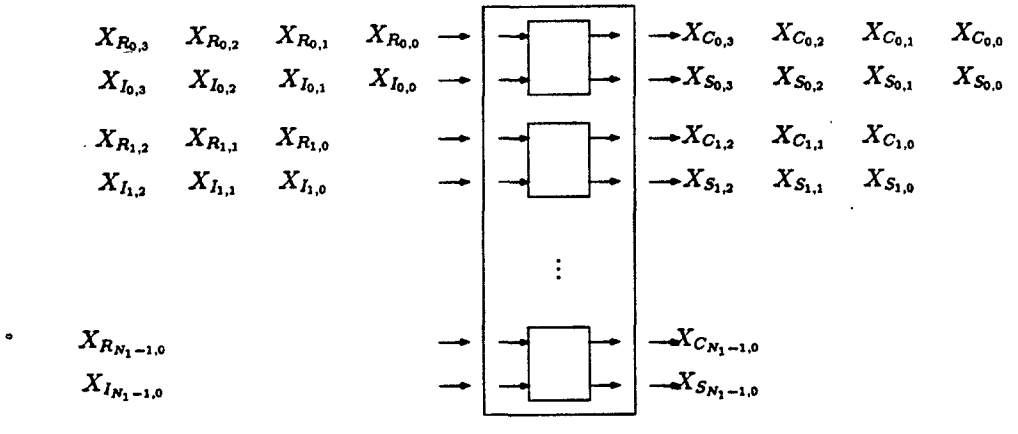


Fig.11 Data flow through multipler-adder A

Multiplier-adder type B :

The input to this component is $Y_H(i,j)$, the data after computing DHT on columns. The function description of the ith subcomponent is as follows

$$Z_C(i,j) = \tfrac{1}{2}[Y_H(i,j)cas(-\alpha) + Y_H(N-i,j)cas\,\alpha]$$

$$Z_S(i,j) = \tfrac{1}{2}[Y_H(i,j)cas\,\alpha - Y_H(N-i,j)cas(-\alpha)],$$

where $\alpha = \frac{\pi}{N_1} is(i)$ (see equation (27)). $Y_H(i,j)$ and $Y_H(N-i,j)$ are made adjacent by modifying S_{N_1} to $\hat{S}_{N_1}$ as explained in Appendix 3. Fig.12 shows the skewed data flow through this component.

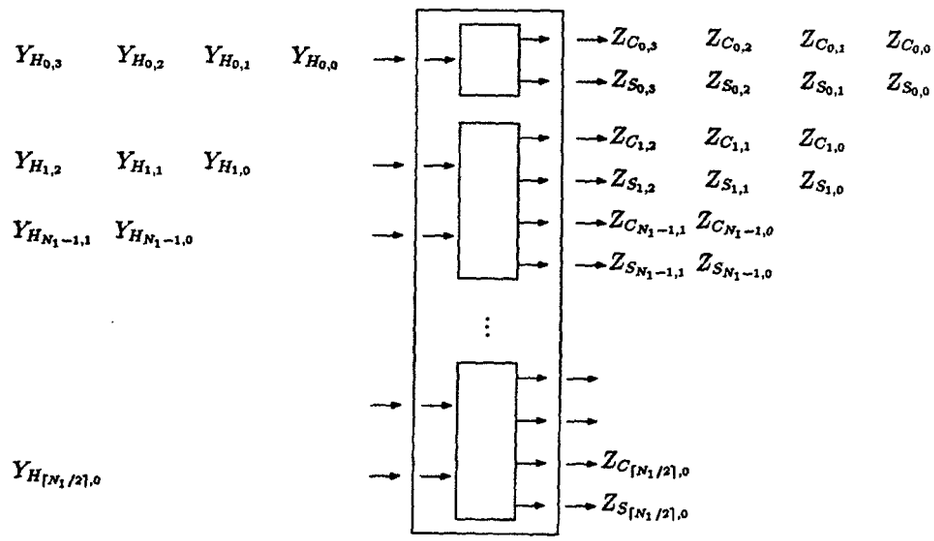


Fig.12 Data flow through multiplier-adder B

Multiplier-adder type C :

The input to this component is complex data formed after computing DFT on rows along with a control line, b_{ij}. The output is the real cosine transform $\hat{C}(i,j)$. The function description of the jth subcomponent is as follows

When $b_{ij} = 0$

$$\hat{C}(i,j) = X_R(i,j)cos\,\alpha - X_I(i,j)sin\,\alpha$$

When $b_{ij} = 1$

$$\hat{C}(i,j) = X_R(i,j)sin\,\alpha + X_I(i,j)cos\,\alpha,$$

where $\alpha = \frac{\pi}{N_2} jt(j)$ and $X_R(i,j)$ and $X_I(i,j)$ are the real and imaginary components of the input data (see equations 20, 21, 26). The data flow through this component is shown in Fig.13.

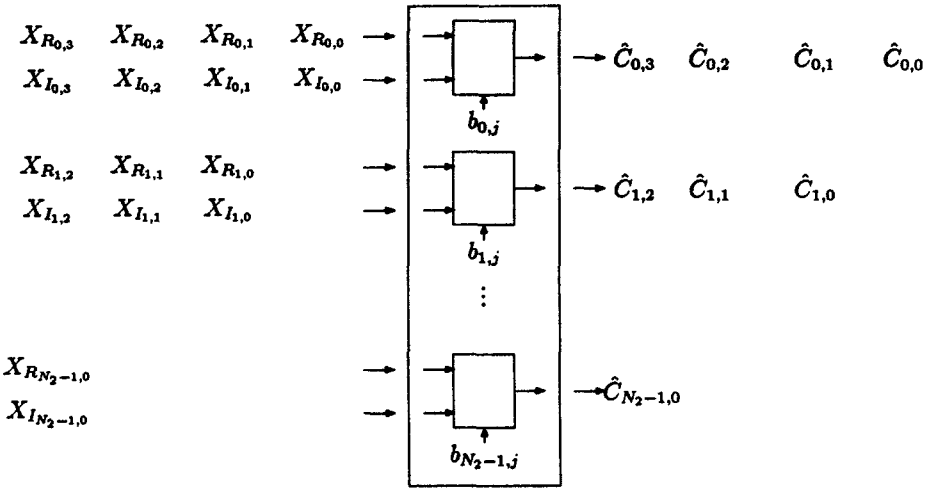


Fig.13 Data flow through multiplier-adder C

Multipler-adder type D :

The input data to this component consists of two parts, $Y_C(i,j)$, which was formed after computing DHT or DFT on the rows of $Z_C(i,j)$, and $Y_S(i,j)$, which was formed after computing DHT or DFT on the rows of $Y_S(i,j)$. The control line b_{ij} is also fed as an input. The function description of the jth subcomponent is as follows

When $b_{ij} = 0$

$$\hat{C}(i,j) = \tfrac{1}{2}[Y_C(i,j)cas(-\alpha) + Y_C(i,N_2-j)cas\,\alpha - Y_S(i,j)cas\,\alpha + Y_S(i,N_2-j)cas(-\alpha)]$$

When $b_{ij} = 1$

$$\hat{C}(i,j) = \tfrac{1}{2}[Y_C(i,j)cas\,\alpha - Y_C(i,N_2-j)cas(-\alpha) + Y_S(i,j)cas(-\alpha) + Y_S(i,N_2-j)cas\,\alpha],$$

where $\alpha = \frac{\pi}{N_2} jt(j)$ (see equations 20, 21, 27). $Y_C(i,j), (Y_S(i,j))$ and $Y_C(i,N_2-j), (Y_S(i,N_2-j))$ are made adjacent by modifying matrix S_{N_2} to $\hat{S}_{N_2}$. The data flow through this component is shown in Fig.14.

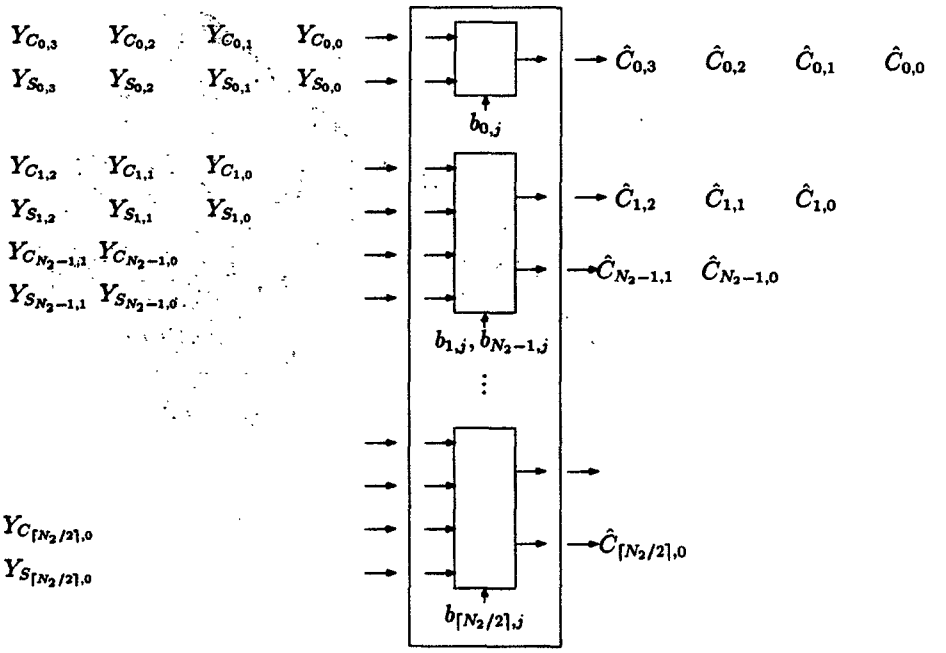


Fig.14 Data flow through multiplier-adder D

Scheme 1 DFT-DFT

First phase : The first summation unit has elements of T_{N_1} embedded, the second summation unit has elements of S_{N_1} embedded and the scaling unit has elements of C_{N_1} embedded in it. The multiplier- adder is of type A.

Second phase : The first summation unit has elements of T_{N_2} embedded, the second summation unit has elements of S_{N_2} embedded and the scaling unit has elements of C_{N_2} embedded in it. The multiplier- adder is of type C.

The interconnection of the various components is shown in Fig.15.

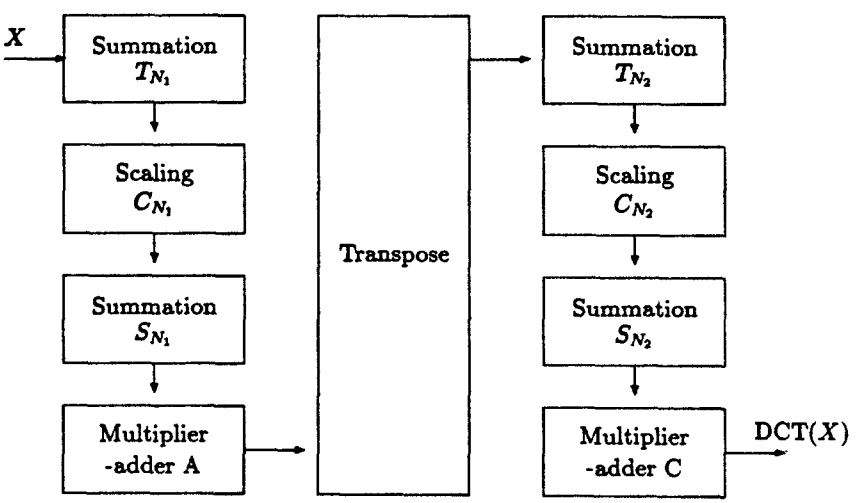


Fig.15 Interconnection for Scheme 1

Scheme 2 DFT-DHT

First phase : This is identical to the first phase of Scheme 1.

Second phase : The first summation unit has elements of T_{N_2} embedded, the second summation unit has elements of $\hat{S}_{N_2}$ embedded and the scaling unit has elements of $\hat{C}_{N_2}$ embedded in it. The multiplier-adder is of type D.

The interconnection of the various components is shown in Fig.16.

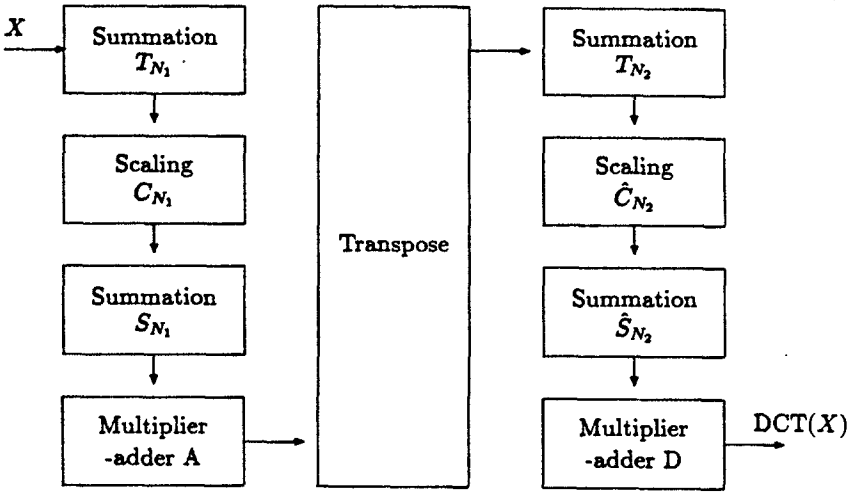


Fig.16 Interconnection for Scheme 2

Scheme 3 DHT-DFT

First phase : The first summation unit has elements of T_{N_1} embedded, the second summation unit has elements of $\hat{S}_{N_1}$ embedded and the scaling unit has elements of $\hat{C}_{N_1}$ embedded. The multiplier-adder is of type B.

Second phase : This is identical to the second phase of Scheme 1.

The interconnection is shown in Fig.17.

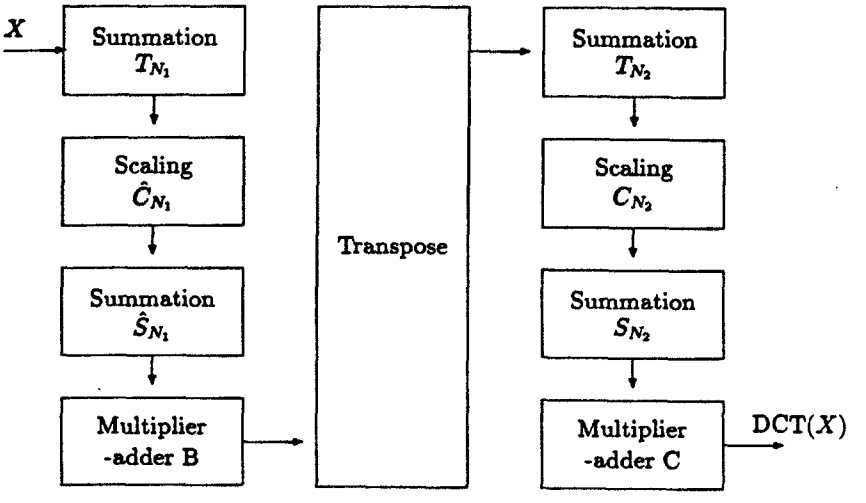


Fig.17 Interconnection for Scheme 3

Scheme 4 DHT-DHT

First phase : This is identical to the first phase of Scheme 3.

Second phase : This is identical to the second phase of Scheme 2.

The interconnection is shown in Fig.18.

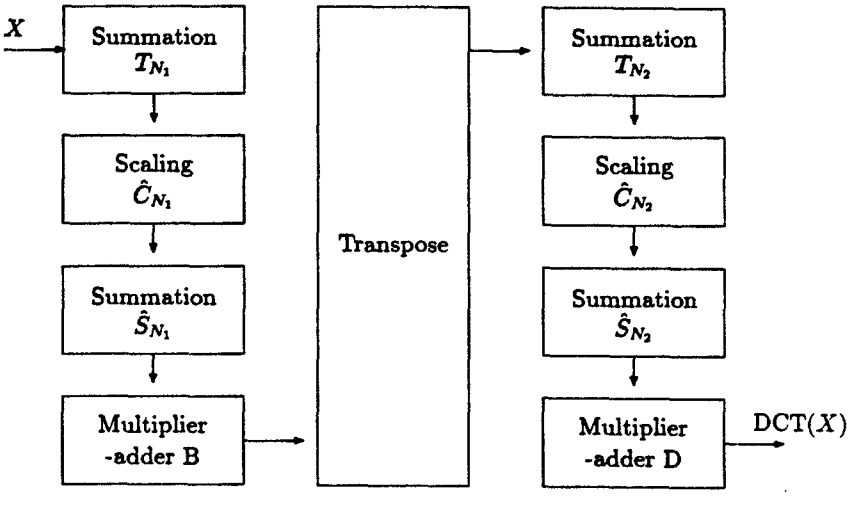


Fig.18 Interconnection for Scheme 4

The matrices $\hat{S}$, $\hat{C}$, T for computing the DCT for $N = 35$ ($N_1 = 7$, $N_2 = 5$) using Scheme 4 have been listed in Appendix 5.

Thus we find that the element skewed implementation of DFT(N), DHT(N), DCT(N) requires an area of $O(N)$ and a computation time of $O(max(N_1, N_2))$,

where $N = N_1N_2$ and N_1, N_2 are mutually prime. Our designs are optimal since the theoretical lower bound of $AT^2 = \Omega(N^2)$ have been achieved.

4 Bit serial implementations

In this section we discuss the synchronous bit serial implementations of the components discussed in Section 3. We chose to use this mode because for data of precision p, there is a drastic reduction in area from $O(pN_1N_2)$ to $O(N_1N_2)$ for most of the components. Another advantage is that the communication within and between VLSI chips is more efficient. The disadvantage of having to clock the bit serial version p times more can be taken care of by clocking at a faster rate, since the basic subcomponents are much simpler. We have chosen the binary left directed LSB to MSB technique [JK] as against the higher base right directed MSB to LSB technique [OJ], since the circuitry of the subcomponents is of lesser complexity . The larger delay in the scaling subcomponent is compensated by a smaller delay in each of the summation subcomponents and a higher clock rate.

4.1 DFT implementation

In this section we discuss briefly the bit serial versions of the summation, scaling and transpose components as in [OJ], [JK] but in the light of a different data flow format. Every element of data of precision p, is followed by a "0" word, which is a string of p zeros. This is necessary since $2p$ bits are generated by the bit serial multiplier and unless the data is appended by a "0" word, the results would be erroneous. We can thus think of every data to be a $2p$ bit word now. The control input r_{in} would indicate that the first bit of an element is being supplied. Thus $r_{in} = 1$ for the least significant bit of the $2p$ word and is 0 otherwise. We choose to propagate the real and the imaginary parts of the data along the same line inorder to further reduce the size of each subcomponent. The sequence is as follows, the $2p$ bit real part followed by the $2p$ bit imaginary part. The control input q_{in} indicates whether

the data is real or imaginary. $q_{in} = 0$ for the real part and is 1 for the imaginary part.

0	...	0	$(x_{I_{ij}})_{p-1}$	...	$(x_{I_{ij}})_1$	$(x_{I_{ij}})_0$	0	...	0	$(x_{R_{ij}})_{p-1}$	...	$(x_{R_{ij}})_1$	$(x_{R_{ij}})_0$	X_{in}
0	...	0	0	...	0	1	0	...	0	0	...	0	1	r_{in}
1	...	1	1	...	1	1	0	...	0	0	...	0	0	q_{in}

Summation component : The functional definition of the bit serial summation subcomponent is as follows.

$x_{out} = x_{in}$

$r_{out} = r_{in}$

$q_{out} = \acute{q}_{out} = q_{in}$

if $q_{in} = \acute{q}_{in}$ then

  if $r_{in} \equiv 1$ then

    $z_{out} = z_{in} + x_{in} - 2c_{out}$

  else

    $z_{out} = z_{in} + x_{in} + c_{in} - 2c_{out}$

  endif

endif

Since the delay in the summation subcomponent is only one, the elements are fed in 1 bit skewed manner.

Scaling component : The scaling subcomponent is essentially a multiplier followed by a shift register so that only the higher order bits can be retained. There are two types of subcomponents depending on whether C_{ii} is real or imaginary. The subcomponent corresponding to an imaginary C_{ii} consists of additional circuitry so that the data in the real and the imaginary parts can be swapped. This is possible by having an additional $2p$ bit shift register. To maintain uniformity in the bit skewness of the data, the subcomponent corresponding to a real C_{ii} has to be equipped with a $2p$ bit shift register too. The functional definition of the scaling subcomponent is as follows.

$x_{out} = x_{in}$

$r_{out} = r_{in}$

$q_{out} = q_{in}$

if $r_{in} = 1$ then

$z_{out} = x_{in}y - 2c_{out}$

else

$z_{out} = x_{in}y + z_{in} + c_{in} - 2c_{out}$

endif

$c_{out} = c_{in}$

Transpose component : This component consists of an array of shift registers of length $2p - 2$. The area of this subcomponent is thus a function of p. The functional definition of the shift register SR of the ijth subcomponent is as follows.

$x_{out} = y_{out} = SR[0]$

for $i = 0, 1, \ldots, 2p - 3$

$SR[i] = SR[i + 1]$

if $c_{in} > 0$ then

$SR[2p - 2] = x_{in}$

else

$SR[2p - 2] = y_{in}$

endif

The control input at the kth clock unit is as follows

if $k \geq 2(i + j) - (2p - 2)N_1$

$c_{in}(i, j) = 1$

else

$c_{in}(i, j) = -1$

endif.

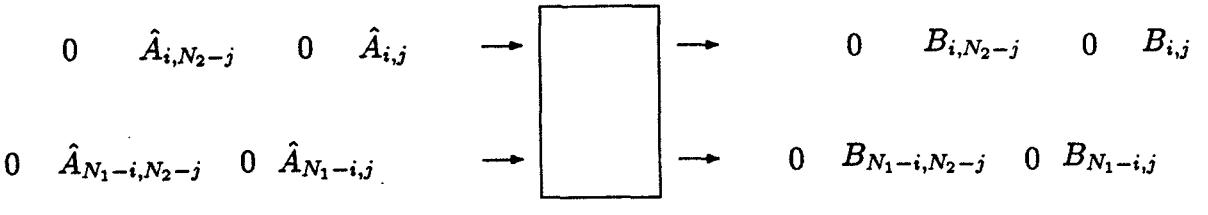
4.2 DHT implementation

In this section we discuss the data flow formats and the bit serial implementation of the adder units of Schemes 1 and 2 of DHT. The data in Scheme 1 is of the form as shown below.

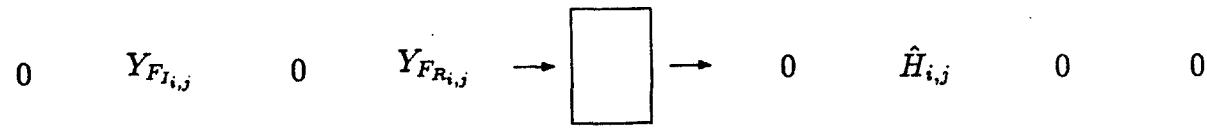
$$\begin{array}{cccccccccccccc} 0 & \ldots & 0 & (x_{i,j1})_{p-1} & \ldots & (x_{i,j1})_1 & (x_{i,j1})_0 & 0 & \ldots & 0 & (x_{i,j})_{p-1} & \ldots & (x_{i,j})_1 & (x_{i,j})_0 & X_{in} \\ 0 & \ldots & 0 & 0 & \ldots & 0 & 1 & 0 & \ldots & 0 & 0 & \ldots & 0 & 1 & r_{in} \end{array}$$

We do not need the control input q_{in} since the data is always real. The elements are 1 bit skewed. The bit serial summation and the transpose components are identical to those of DFT. Since both the data and C_{ii} are always real, the scaling component performs only real multiplications. No additional shift registers are required. The scaling component is not only smaller, but the delay for every element passing through it is $2p$ less.

The adder unit in Scheme 1 consists of $\lfloor N_1/2 \rfloor$ components, each of which consists of $2p$, $4p$ shift registers, an adder, a subtractor and a couple of delay units. The data flow through the ith component is as follows.



The data flow through the various components in Scheme 2 is identical to that of DFT. The adder unit consists of N_2 subcomponents, each of which consists of $2p$ bit shift registers and subtractors. The data flow through the jth subcomponent is as follows.



4.3 DCT implementation

In this section we discuss the data format and the bit serial implementation of the different multiplier-adder units of DCT. The format of the data is identical to that of DFT. After the first phase, the data consists of the cosine part X_C and the sine part X_S. We can think of X_C to be equivalent to the real part X_R in DFT and X_S to be equivalent to the imaginary part X_I in DFT. Here the control input q_{in} would indicate whether the data is the cosine part or the sine part, $q_{in} = 0$ for the cosine part and is 1 for the sine part. The input data X is always real. Thus in Schemes 1 and 2, which compute DFT first, every element of the input data is followed by three "0" words.

0	...	0	0	0	...	0	0	0	...	0	0	$(x_{ij})_{p-1}$	...	$(x_{ij})_1$	$(x_{ij})_0$	X_{in}
0	...	0	0	0	...	0	1	0	...	0	0	0	...	0	1	r_{in}
1	...	1	1	1	...	1	1	0	...	0	0	0	...	0	0	q_{in}

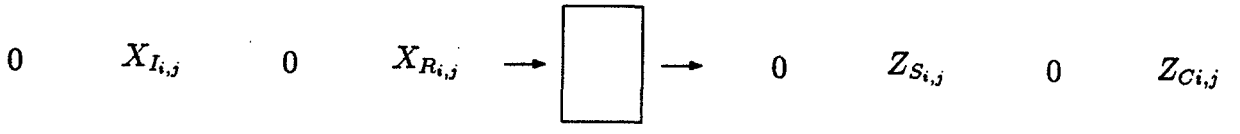
In Schemes 3 and 4, which compute DHT first, the input data occurs in both the sine and the cosine part. This guarantees simpler circuitry and less delay in the multiplier-adder B.

0	...	0	0	$(x_{ij})_{p-1}$	...	$(x_{ij})_1$	$(x_{ij})_0$	0	...	0	0	$(x_{ij})_{p-1}$	...	$(x_{ij})_1$	$(x_{ij})_0$	X_{in}
0	...	0	0	0	...	0	1	0	...	0	0	0	...	0	1	r_{in}
1	...	1	1	1	...	1	1	0	...	0	0	0	...	0	0	q_{in}

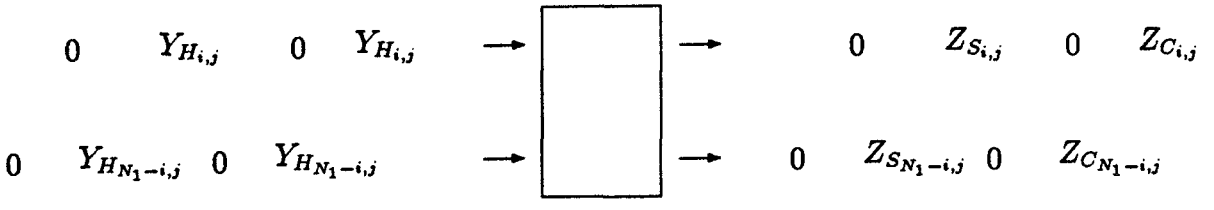
The multiplier-adder units are a combination of summation, scaling subcomponents timed properly with the help of shift registers and regulated with the control inputs. We shall not discuss the functional definition of the subunits here. Other than the multiplier-adder units, the rest of the components in the various schemes

are either identical to those used in DFT or to those used in DHT. Multiplier-adders C and D need a control input b_{ij}. Since N_1 and N_2 are fixed, the value of b_{ij} for $0 \leq i \leq N_1 - 1$ and $0 \leq j \leq N_2 - 1$ are known beforehand. These values can be stored in N_2 N_1-bit shift registers. The clock would be an AND of the system clock with $r_{in}\overline{q_{in}}$.

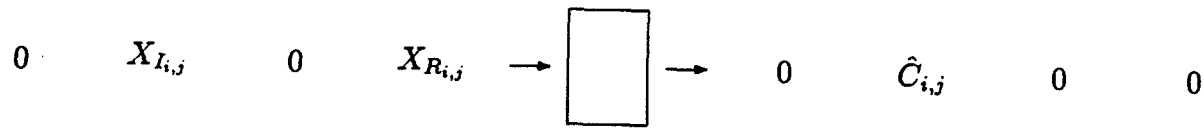
Multiplier-adder A : The ith subcomponent, $1 \leq i \leq N_1 - 1$, would consist of two fixed serial multipliers, $cos\ \frac{\pi}{N_1} is(i)$ and $sin\ \frac{\pi}{N_1} is(i)$, $2p$ bit shift registers, an adder, a subtractor and switches based on the control input q_{in}. The data flow through this component is as follows. Z_C, Z_S, X_R, X_I are defined as in Section 3.3.



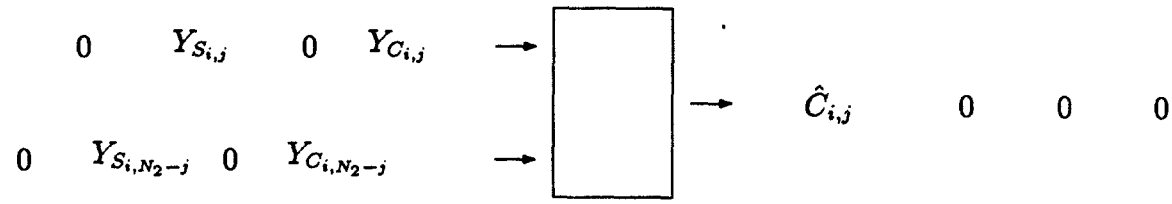
Multiplier-adder B : The ith subcomponent, $1 \leq i \leq \lfloor N_1/2 \rfloor$, would consist of four fixed serial multipliers, $cas\left(-\frac{\pi}{N_1} is(i)\right)$, $cas\left(\frac{\pi}{N_1} is(i)\right)$, $cas\left(-\pi s(i) + \frac{\pi}{N_1} is(i)\right)$, $cas\left(\pi s(i) - \frac{\pi}{N_1} is(i)\right)$, switches based on q_{in}, adders, subtractors and some simple logic circuitry. The data flow through this subcomponent is as follows. Z_C, Z_S, Y_H are defined as in Section 3.3.



Multiplier-adder C : The jth subcomponent, $1 \leq j \leq N_2 - 1$, consists of two fixed serial multipliers, $cos\ \frac{\pi}{N_2} jt(j)$ and $sin\ \frac{\pi}{N_2} jt(j)$, $2p$ bit shift registers, an adder, subtractor and some simple switches based on the control inputs q_{in} and b_{ij}. The data flow through this subcomponent is as follows. X_R, X_I, $\hat{C}$ are as defined as in Section 3.3.



Multiplier-adder D : The jth subcomponent, $1 \leq j \leq N_2 - 1$, consists of two fixed p bit serial multipliers, $cas\left(-\frac{\pi}{N_2} jt(j)\right)$ and $cas\left(\frac{\pi}{N_2} jt(j)\right)$, adders, subtractors and some simple switches based on q_{in} and b_{ij}. The data flow through this subcomponent is as follows. Y_C, Y_S, $\hat{C}$ are as defined as in Section 3.3.



Thus we find that the bit skewed implementation of DFT(N), DHT(N), DCT(N) requires an area of $O(pN)$ and a computation time of $O(max(N_1, N_2)p)$, where p is the precision, $N = N_1 N_2$ and N_1, N_2 are mutually prime. Note that these designs achieve the optimal time performance for bit serial systolic architectures.

5 Conclusion

In this report we have presented two dimensional schemes for computing DFT, DHT, DCT based on the so called small n algorithms, their systolic word and bit serial array implementations. The section on DFT exists in the literature. We included it here, since, we needed some of the components for our designs of DHT and DCT. Though the two dimensional formulation of DHT is not a new concept, detailed VLSI design to compute it does not exist in the literature. We have improved upon the method suggested by Sorensen et. al. [SJBH] and have given a fairly detailed description of the various components. We have formulated the mapping from one dimensional DCT into the two dimensional form. We have

proposed four schemes based on whether DFT or DHT is computed on the rows and on the columns.

We claim that Scheme 1 for the computation of DHT is superior to Scheme 2. This is because the data in Scheme 1 is always real and hence the associated circuitry is simpler. The delay in the scaling and adder units is significantly less. In the bit serial mode, $2p$ bits are necessary to describe each element as against $4p$ bits in Scheme 2. Thus the overall delay in scheme 1 is half of that of Scheme 2.

A comparison of the four schemes for DCT leads us to claim that scheme 4, that is, DHT-DHT is the best. The reasons are as follows. First of all, since the data is always real, its representation in terms of bits as well as control inputs is simpler. Secondly, the circuitry is less complex. Thirdly, the delay is much less. For instance, in the bit serial mode, the delay is $2p$ bits less for every element through each of the scaling components as well as the multiplier-adder units.

In the formulation for DCT, we have assumed that N_1 and N_2 are relatively prime. A practical case would be when N_1, $N_2 \approx \sqrt{N_1 N_2}$. We find that when $N_2 = N_1 + 1$ or when $N_1 = N_2 + 1$, there is a remarkable reduction in the hardware needed in the multiplier-adder units. In the case when $N_2 = N_1 + 1$, $s(k_1) = 1/2$ for all k_1. This implies that $X_C(N_1 - k_1) = X_S(k_1)$ and $X_S(N_1 - k_1) = X_C(k_1)$. The number of multipliers in multiplier-adder A and B is thus reduced by half.

An estimate of the sizes of the various components used to compute the DCT of 208 ($N_1 = 16$, $N_2 = 13$) points with 16 bit precision using Scheme 4 is as follows. The technology is CMOS. In the first phase, the summation components consist of 288 subcomponents each of size approximately $80\lambda \times 90\lambda$ with a delay of at most 5ns. The scaling component consists of 18 subcomponents each of size approximately $1200\lambda \times 1200\lambda$. The multiplier-adder subcomponent consists of 16 subcomponents of size $2500\lambda \times 2500\lambda$. Though each of these subcells work correctly for a clock period of 20ns, we choose 40ns for reliable operation. The throughput is 20,000 of such DCTs per second.

References

[AC] Agarwal,R.,Cooley,J.,"New algorithms for digital convolution", IEEE Trans. Acoust.,Speech,Signal Processing, vol.ASSP-25, pp.392-410, 1977.

[ANR] Ahmed,N.,Natarajan,T.,Rao,K.R.,"Discrete Cosine Transform", IEEE Trans.on Computer, C-23, pp.90-93, Jan'74.

[Br] Bracewell,R.N.,"Discrete Hartley Transform", J.Opt.Soc.Amer., vol.73, pp.1832-1835,Dec'83.

[Bu] Burrus,C.,S.,"Index Mappings for MUtidimensional Formulation of the DFT and Convolution", IEEE Trans. Acoust.,Speech,Signal Processing, vol. ASSP-25, pp.239-242, 1977.

[CHF] Chen,W.,Harrison,C.S.,Fralick,S.C.,"A fast computational algorithm for the discrete cosine transform", IEEE Trans. on Comm., COM-25, pp.1004-1008, 1977.

[CT] Cooley,J.,Tukey,J.,"An algorithm for the machine calculation of complex Fourier series", Math. Comput., vol.19, pp.297-301, 1985.

[Gd] Good,I.,"The interaction algorithm and practical Fourier analysis", J. Royal Stat.Soc., ser.B, vol.20, pp.361-372, 1958.

[Hou] Hou,H.S.,"The fast Hartley transform algorithm", IEEE Trans. on Computer, vol.C-36, pp.147-156, Feb'87.

[JK] JáJá,J.,Kapoor,A.,"Parallel and Pipelined VLSI Architectures based on decomposition", IEEE Workshop on VLSI Signal Processing (3rd Conf.) pp.177-187, Oct.'86.

[Lee] Lee,B.G.,"A new algorithm for the discrete cosine transform", IEEE Trans. Acoust.,Speech,Signal Processing, vol.ASSP-32, pp.1243-1245, 1984.

[Mal] Malvar,H.,"Fast Computation of Discrete Cosine Transform through Fast Hartley Transform", Electron.Lett., vol.22, pp.353-353, Mar 1986.

[NP] Narasimha,M.J.,Peterson,A.M.,"On the computation of the discrete cosine

transform", IEEE Trans.on Comm.,COM-26, pp.934-936, 1978.

[OJ] Owens,R.M.,JáJá,J.,"A VLSI chip for the Winograd/Prime Factor Algorithm to compute the Discrete Fourier Transform", IEEE Trans. Acoust.,Speech,Signal Processing, vol.ASSP-34, pp.979-989, Aug.'86.

[PW] Pei,S.C.,Wu,J.L.,"Split radix fast Hartley transform", Electron.Lett., vol.22, pp.26-27, Jan'86.

[Sil] Silverman,H.F.,"An introduction to programming the Winograd Fourier transform algorithm (WFTA)", IEEE Trans. Acoust.,Speech,Signal Processing, vol.ASSP-25, pp.152-165, 1977.

[SJBH] Sorensen,H.V.,Jones,D.L.,Burrus,C.S.,Heideman,M.T.,"On Computing the Discrete Hartley Transform", IEEE Trans. Acoust.,Speech,Signal Processing, vol.ASSP-33, no.4, pp.1231-1238, 1985.

[VN] Vetterli,M.,Nussbaumer,H.J.,"Simple FFT and DCT algorithms with reduced number of operations", Signal Process., pp.267-278, 1984.

[Win] Winograd,S.,"On computing the discrete Fourier transform", Math. Comput., vol.32, pp.175-195, 1978.

Appendix 1

Part A :

$$\hat{H}(k_1,k_2) = \frac{1}{\sqrt{N}}\sum_{n_2}\sum_{n_1}\hat{x}(n_1,n_2)cas\frac{2\pi}{N}nk \quad (1)$$

$$= \frac{1}{\sqrt{N}}\sum_{n_2}\sum_{n_1}\hat{x}(n_1,n_2)cas\frac{2\pi}{N}(K_1K_3n_1k_1 \bmod N + K_2K_4n_2k_2 \bmod N$$

$$+ \quad K_1K_4n_1k_2 \bmod N + K_2K_3n_2k_1 \bmod N - aN), \text{where a} = 0 \text{ or } 1 \quad (2)$$

In order for two dimensional nesting to be possible, the constants K_1, K_2, K_3, K_4 have to be chosen such that $K_1K_4n_1k_2 \bmod N$ and $K_2K_3n_2k_1 \bmod N$ are zero. $K_1 = \alpha N_2$ and $K_4 = \delta N_1$ guarantee the first term to be zero. Since N_1 and N_2 are relatively prime to each other, we can choose $K_2 = \beta N_1$ and $K_3 = \gamma N_2$ and make the second term zero. Expression (1) then reduces to

$$\hat{H}(k_1,k_2) = \frac{1}{\sqrt{N}}\sum_{n_2}\sum_{n_1}\hat{x}(n_1,n_2)cas\,\frac{2\pi}{N}\left(\alpha\gamma N_2^2n_1k_1 \bmod N + \beta\delta N_1^2n_2k_2 \bmod N\right) \quad (3)$$

By choosing $\alpha = \beta = 1$, $\gamma = N_2^{-1} \bmod N_1$, $\delta = N_1^{-1} \bmod N_2$, we get

$$\hat{H}(k_1,k_2) = \frac{1}{\sqrt{N}}\sum_{n_2}\sum_{n_1}\hat{x}(n_1,n_2)cas\,\frac{2\pi}{N}(N_2n_1k_1 \bmod N + N_1n_2k_2 \bmod N)$$

$$= \frac{1}{\sqrt{N}}\sum_{n_2}\sum_{n_1}\hat{x}(n_1,n_2)cas\,\left(\frac{2\pi}{N_1}n_1k_1 + \frac{2\pi}{N_2}n_2k_2\right)$$

$$= \frac{1}{\sqrt{N}}\sum_{n_2}\left\{\sum_{n_1}\hat{x}(n_1,n_2)cas\,\frac{2\pi}{N_1}n_1k_1\right\}cas\,\frac{2\pi}{N_2}n_2k_2$$

$$-\frac{2}{\sqrt{N}}\sum_{n_2}\left\{\sum_{n_1}\hat{x}(n_1,n_2)sin\,\frac{2\pi}{N_1}n_1k_1\right\}sin\,\frac{2\pi}{N_2}n_2k_2. \quad (4)$$

Part B :

$\hat{H}(k_1, k_2)$ can also be expressed as

$$\begin{aligned}\hat{H}(k_1,k_2) &= \frac{1}{\sqrt{N}}\sum_{n_2}\left\{\sum_{n_1}\hat{x}(n_1,n_2)cos\,\frac{2\pi}{N_1}n_1k_1\right\}cas\,\frac{2\pi}{N_2}n_2k_2 \\ &\quad +\frac{1}{\sqrt{N}}\sum_{n_2}\left\{\sum_{n_1}\hat{x}(n_1,n_2)sin\,\frac{2\pi}{N_1}n_1k_1\right\}cas\,\left(-\frac{2\pi}{N_2}n_2k_2\right) \qquad (5)\end{aligned}$$

$$\text{Let} A(k_1,n_2) = \frac{1}{\sqrt{N}}\sum_{n_1}\hat{x}(n_1,n_2)cas\,\frac{2\pi}{N_1}n_1k_1, \text{then}$$

$$\begin{aligned}\hat{H}(k_1,k_2) &= A(k_1,0)+\frac{1}{2}\sum_{n_2=1}^{N_2-1}\{A(k_1,n_2)+A(N_1-k_1,n_2)\}cas\,\frac{2\pi}{N_2}n_2k_2 \\ &\quad +\frac{1}{2}\sum_{n_2=1}^{N_2-1}\{A(k_1,n_2)-A(N_1-k_1,n_2)\}cas\,-\frac{2\pi}{N_2}n_2k_2 \\ &= A(k_1,0)+\frac{1}{2}\sum_{n_2=1}^{N_2-1}\{A(k_1,n_2)+A(N_1-k_1,n_2)\}cas\,\frac{2\pi}{N_2}n_2k_2 \\ &\quad +\frac{1}{2}\sum_{N_2-n_2=1}^{N_2-1}\{A(k_1,N_2-n_2)-A(N_1-k_1,N_2-n_2)\}cas\,\frac{2\pi}{N_2}(n_2-N_2)k_2 \\ &= A(k_1,0)+\frac{1}{2}\sum_{n_2=1}^{N_2-1}\{A(k_1,n_2)+A(N_1-k_1,n_2)+A(k_1,N_2-n_2) \\ &\quad -A(N_1-k_1,N_2-n_2)\}cas\,\frac{2\pi}{N_2}n_2k_2 \\ &= \sum_{n_2=0}^{N_2-1}B(k_1,n_2)cas\,\frac{2\pi}{N_2}n_2k_2, \qquad (6)\end{aligned}$$

where $B(0,n_2) = A(0,n_2)$ for $0 \le n_2 \le N_2 - 1$,

$B(k_1,0) = A(k_1,0)$ for $0 \le k_1 \le N_1 - 1$ and

$B(k_1,n_2) = \frac{1}{2}[A(k_1,n_2)+A(N_1-k_1,n_2)+A(k_1,N_2-n_2)-A(N_1-k_1,N_2-n_2)]$,

for $1 \le k_1 \le N_1 - 1$ and $1 \le n_2 \le N_2 - 1$.

Appendix 2

$$\begin{aligned}
\hat{C}(k_1,k_2) &= \frac{2}{N}\sum_{n_2}\sum_{n_1}\hat{y}(n_1,n_2)cos\left[\frac{2\pi}{N}nk+\frac{\pi}{2N}k\right] \\
&= \frac{2}{N}\sum_{n_2}\sum_{n_1}\hat{y}(n_1,n_2)cos\left[\frac{2\pi}{N}(K_1K_3n_1k_1 \bmod N + K_2K_4n_2k_2 \bmod N\right. \\
&\quad + K_1K_4n_1k_2 \bmod N + K_2K_3n_2k_1 \bmod N - aN) \\
&\quad \left.+\frac{\pi}{2N}(K_3k_1 \bmod N + K_4k_2 \bmod N - b_{k_1k_2}N)\right] \qquad (7)
\end{aligned}$$

where a is a positive integer constant. Two dimensional nesting is possible only if the cross terms, namely, $K_1K_4n_1k_2 \bmod N$ and $K_2K_3n_2k_1 \bmod N$ are zero. A choice of $K_1 = \alpha N_2$ and $K_4 = \delta N_1$, where α and δ are positive integer constants guarantee the first term to be zero. Since N_1 and N_2 are mutually prime, the second term can be made zero by choosing $K_2 = \beta N_1$ and $K_3 = \gamma N_2$, where β and $\gamma$are positive integer constants. Equation (7) then reduces to

$$\begin{aligned}
\hat{C}(k_1,k_2) &= \frac{2}{N}\sum_{n_2}\sum_{n_1}\hat{y}(n_1,n_2)cos\left[\frac{2\pi}{N}(\alpha\gamma N_2^2n_1k_1 \bmod N + \beta\delta N_1^2n_2k_2 \bmod N - aN)\right. \\
&\quad \left.+\frac{\pi}{2N}(\gamma N_2k_1 \bmod N + \delta N_1k_2 \bmod N - b_{k_1k_2}N)\right] \\
&= \frac{2}{N}\sum_{n_2}\sum_{n_1}\hat{y}(n_1,n_2)cos\left[\frac{2\pi}{N}(\alpha\gamma N_2^2n_1k_1 + \beta\delta N_1^2n_2k_2 - \acute{a}N)\right. \\
&\quad \left.+\frac{\pi}{2N}(\gamma N_2k_1 \bmod N + \delta N_1k_2 \bmod N - b_{k_1k_2}N)\right], \qquad (8)
\end{aligned}$$

where $\acute{a}$ is an integer constant. We define two functions $s(k_1)$ and $t(k_2)$ such that $s(k_1) = (\gamma N_2k_1 \bmod N)/(2N_2k_1)$ and $t(k_2) = (\delta N_1k_2 \bmod N)/(2N_1k_2)$. It is obvious that $s(k_1)$ and $t(k_2)$ can be predetermined. We choose $\alpha = \beta = 1$, $\gamma = N_2^{-1} \bmod N_1, \delta = N_1^{-1} \bmod N_2$. On substituting these values into equation (8) we get

$$\begin{aligned}
\hat{C}(k_1,k_2) &= \frac{2}{N}\sum_{n_2}\sum_{n_1}\hat{y}(n_1,n_2)cos\left[2\pi\left(\frac{n_1k_1}{N_1}+\frac{n_2k_2}{N_2}\right)\right. \\
&\quad \left.+\frac{\pi}{2N}(2N_2k_1s(k_1) + 2N_1k_2t(k_2) - b_{k_1k_2}N)\right] \\
&= \frac{2}{N}\sum_{n_2}\sum_{n_1}\hat{y}(n_1,n_2)cos\left[\frac{\pi}{N_1}(2n_1 + s(k_1))k_1 + \frac{\pi}{N_2}(2n_2 + t(k_2))k_2 - \frac{\pi}{2}b_{k_1k_2}\right]. \\
&\qquad (9)
\end{aligned}$$

Appendix 3

One dimensional DHT can be expressed by

$$H = S_N \hat{C}_N T_N x, \tag{10}$$

where S_N, $\hat{C}_N$, T_N are defined as in section 2.2. Let $\hat{H}$ be a permutation of H, such that $\hat{H} = (H(0)H(1)H(N-1)\cdots H(k)H(N-k)\cdots)^T$, $1 \leq k \leq \lfloor N/2 \rfloor$, and N is odd. $\hat{H}$ is thus equal to PH, where P is a permutation matrix with the following characteristics :

For odd i, $\quad P(i, \lceil i/2 \rceil) = 1$

For even i, and $i \neq 0$, $P(i, N - i/2) = 1$

$$P(0,0) = 1.$$

$\hat{H}$ can thus be expressed as

$$\hat{H} = \hat{S}_N \hat{C}_N T_N x, \tag{11}$$

where $\hat{S}_N = PS_N$. To get the output in the form of $\hat{H}$, all we need is to modify the matrix S_N to $\hat{S}_N$. Since both P and S_N are known beforehand, $\hat{S}_N$ can be precomputed. This modification results in an entire row getting permuted without changing the order of the elements in a row. So, we do not need to modify T_N in the computation of DHT of rows.

Example :

For $N = 5$, P takes the following form $\begin{pmatrix} 1 & . & . & . & . \\ . & 1 & . & . & . \\ . & . & . & . & 1 \\ . & . & 1 & . & . \\ . & . & . & 1 & . \end{pmatrix}$.

Appendix 4

Let $A = S_{N_1}\hat{C}_{N_1}T_{N_1}\hat{x}$.

In the first step we have to ensure that $A(k_1, N_2)$, $A(N_1 - k_1, n_2)$, $A(k_1, N_2 - n_2)$, $A(N_1 - k_1, N_2 - n_2)$ will be adjacent to each other. This can be made possible by

1. permuting the columns of x such that $x(k_1, n_2)$ and $x(k_1, N_2 - n_2)$ are adjacent to each other, for $1 \leq n_2 \leq N_2 - 1$.

2. replacing S_N by $\hat{S}_N$ as explained in Appendix 3.

Let $\hat{A}(k_1, n_2) = A(k_1, n_2) + A(N_1 - k_1, n_2)$ and

$\hat{A}(N_1 - k_1, n_2) = A(k_1, n_2) - A(N_1 - k_1, n_2)$, for $1 \leq k_1 \leq \lfloor N_1/2 \rfloor$. This is for odd N_1. For even N_1, $\hat{A}(N_1/2, n_2) = A(N_1/2, n_2)$. $\hat{A}$ can thus be expressed as $\hat{A} = PA$, where $\hat{A}$ and A are the two matrices defined above. The matrix P has the following characteristics :

For odd i, $\quad P(i,i) = 1;\ P(i,i+1) = 1.$

For even i and $i \neq 0$, $P(i,i) = -1;\ P(i,i-1) = 1.$

$$P(0,0) = 1.$$

Thus $\hat{A} = PS_{N_1}\hat{C}_{N_1}T_{N_1}x = \acute{S}_{N_1}\hat{C}_{N_1}T_{N_1}\hat{x}$, where $\acute{S}_{N_1} = P\hat{S}_{N_1}$. Since P is known beforehand, $\acute{S}_{N_1}$ can be precomputed. Thus $A(k_1, n_2) + A(N_1 - k_1, n_2)$, $A(k_1, n_2) - A(N_1 - k_1, n_2)$, $A(k_1, N_2 - n_2) + A(N_1 - k_1, N_2 - n_2)$ and $A(k_1, N_2 - n_2) - A(N_1 - k_1, N_2 - n_2)$ are adjacent to each other.

Example :

For $N = 5$, P takes the following form $\begin{pmatrix} 1 & . & . & . & . \\ . & 1 & 1 & . & . \\ . & 1 & -1 & . & . \\ . & . & . & 1 & 1 \\ . & . & . & 1 & -1 \end{pmatrix}$.

Appendix 5

We illustrate Scheme 4 of DCT (DHT-DHT) with the help of an example. Let $N = 35$ with $N_1 = 7$ and $N_2 = 5$. The matrices S_7, C_7, T_7, S_5, C_5, T_5 can be derived from the small n algorithms ($n = 5, 7$). $\hat{C}$ is obtained by negating the imaginary entries of C, $\hat{S}$ is obtained by permuting the rows of S as explained in Appendix 3.

$$\hat{S}_7 = \begin{pmatrix} 1 & . & . & . & . & . & . & . & . \\ 1 & 1 & 1 & 1 & . & 1 & 1 & 1 & . \\ 1 & 1 & 1 & 1 & . & -1 & -1 & -1 & . \\ 1 & 1 & -1 & . & -1 & 1 & -1 & . & -1 \\ 1 & 1 & -1 & . & . & -1 & -1 & 1 & . & 1 \\ 1 & 1 & . & -1 & 1 & -1 & . & 1 & -1 \\ 1 & 1 & . & -1 & 1 & 1 & . & -1 & 1 \end{pmatrix}, \qquad T_7 = \begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ . & 1 & 1 & 1 & 1 & 1 & 1 \\ . & 1 & . & -1 & -1 & . & 1 \\ . & . & -1 & 1 & 1 & -1 & . \\ . & -1 & 1 & . & . & 1 & -1 \\ . & 1 & 1 & -1 & 1 & -1 & -1 \\ . & -1 & . & -1 & 1 & . & 1 \\ . & . & 1 & 1 & -1 & -1 & . \\ . & 1 & -1 & . & . & 1 & -1 \end{pmatrix},$$

$$\hat{C}_7 = \begin{pmatrix} 1.00 & . & . & . & . & . & . & . & . \\ . & -1.17 & . & . & . & . & . & . & . \\ . & . & 0.79 & . & . & . & . & . & . \\ . & . & . & 0.06 & . & . & . & . & . \\ . & . & . & . & 0.73 & . & . & . & . \\ . & . & . & . & . & 0.44 & . & . & . \\ . & . & . & . & . & . & -0.34 & . & . \\ . & . & . & . & . & . & . & 0.53 & . \\ . & . & . & . & . & . & . & . & -0.87 \end{pmatrix},$$

$$\hat{S}_5 = \begin{pmatrix} 1 & . & . & . & . & . \\ 1 & 1 & 1 & 1 & -1 & . \\ 1 & 1 & 1 & -1 & 1 & . \\ 1 & 1 & -1 & 1 & . & 1 \\ 1 & 1 & -1 & -1 & . & -1 \end{pmatrix}, \qquad T_5 = \begin{pmatrix} 1 & 1 & 1 & 1 & 1 \\ . & 1 & 1 & 1 & 1 \\ . & 1 & -1 & -1 & 1 \\ . & 1 & -1 & 1 & -1 \\ . & . & -1 & 1 & . \\ . & 1 & . & . & -1 \end{pmatrix}.$$

$$\hat{C}_5 = \begin{pmatrix} 1.00 & . & . & . & . & . \\ . & -1.25 & . & . & . & . \\ . & . & 0.56 & . & . & . \\ . & . & . & 0.95 & . & . \\ . & . & . & . & 1.54 & . \\ . & . & . & . & . & -0.36 \end{pmatrix},$$