Fast Algorithms for the Discrete Cosine Transform

Ephraim Feig, *Fellow, IEEE,* and Shmuel Winograd, *Fellow, IEEE*

***Abstract*—We introduce several fast algorithms for computing discrete cosine transforms (DCT's) and their inverses on multidimensional inputs of sizes which are powers of 2. Because the one-dimensional 8-point DCT and the two-dimensional 8 × 8-point DCT are so widely used, we discuss them in detail. We also present algorithms for computing scaled DCT's and their inverses; these have applications in compression of continuous tone image data, where the DCT is generally followed by scaling and quantization.**

I. Introduction

DISCRETE cosine transforms (DCT's) are probably the most widely used transforms in the signal processing of image data, especially in coding for compression [1], [20]. In these applications, a two-dimensional DCT on rather small input sizes (8 × 8 or 16 × 16) is used. Because of the wide-spread use of DCT's, research into fast algorithms for their implementation has been rather active [2], [5]–[12], [14], [16]–[18], [21]–[24]. A book devoted to DCT's and their applications [19] was recently published. The algebraic structure of DCT's has also been investigated [6], [10] in order to obtain bounds on the arithmetic complexity of their computations. The works in [7], [9], [10], [23], [24] are particularly relevant to the present work, as they address the two-dimensional case directly, rather than treating it naively as a row-column implementation of the one-dimensional case.

Section II gives a detailed description of the structure of the one-dimensional DCT on 8 points, and uses this to build various algorithms for its implementation. The main result here is that the DCT matrix on 8 points is readily converted (with additions and permutations only) to a direct sum of matrices corresponding to certain polynomial products modulo irreducible polynomials. Section III uses the constructions of the previous section to obtain algorithms for the two-dimensional DCT on 8 × 8 points. It uses the classical result that the tensor product of fields is isomorphic to a direct sum of fields. A particular instantiation of this result to the present case is worked out in detail. As an application, we present an algorithm for the DCT on 8 × 8 points which uses 94 multiplications and 454 additions. A similar algorithm which uses 96 multiplications and considerably more additions was presented in [7]. We also give an algorithm which uses only 86 multiplications but too many additions to be practical.

In most coding applications the DCT is postprocessed with scaling and quantization. Therefore one need not actually compute the DCT itself but rather a scaled version of it, and then compensate for the scale factors in the postprocessing. This observation leads to significant algorithmic savings [2], [9], [10], [24]. In Section IV we present an explicit factorization of the 8-point DCT matrix which takes advantage of this observation, and obtain from this an efficient algorithm for a scaled two-dimensional DCT on 8 × 8 points. This algorithm uses 54 multiplications, 462 additions, and 6 shifts (multiplications by 1/2). Various other factorizations for isolating scale factors are presented in Section VI.

The DCT matrix is orthogonal; its inverse is its transpose. Our factorizations are easily transformed into factorizations of the inverse DCT via either direct inversion or transposition. These lead to two different types of factorizations which, in turn, yield different algorithms. These are briefly discussed in Section V. The arithmetic complexity of the algorithms we obtain here are the same as the corresponding algorithms for the forward direction.

Finally, the general theory for the DCT and the scaled DCT algorithms on input sizes which are powers of 2 is presented in Section VII.

II. The 8-Point DCT

The N-point DCT is defined by for $0 \le n \le N-1$ the equations

$$y_n = c_n \sum_{k=0}^{N-1} \cos \frac{2\pi n(2k+1)}{4N} x_k \tag{1}$$

where $c_0 = 1/\sqrt{N}$ and $c_n = \sqrt{2/N}$ for $1 \le n \le N-1$. Its inverse (IDCT) is given by

$$x_k = \sum_{n=0}^{N-1} \cos \frac{2\pi n(2k+1)}{4N} c_n y_n. \tag{2}$$

In this section we develop explicitly algorithms for the DCT on 8 points. These are important in image processing applications, and also the ideas presented here extend to the general case of the DCT on any 2^m set of points.

Manuscript received September 2, 1990; revised September 27, 1991.
The authors are with the IBM Research Division, Thomas J. Watson Research Center, Yorktown Heights, NY 10598.
IEEE Log Number 9201588.

Setting $\gamma(k) = \cos(2\pi k/32)$, the 8-point DCT matrix is

$$C_8 = \frac{1}{2}\begin{bmatrix} \gamma(4) & \gamma(4) & \gamma(4) & \gamma(4) & \gamma(4) & \gamma(4) & \gamma(4) & \gamma(4) \\ \gamma(1) & \gamma(3) & \gamma(5) & \gamma(7) & -\gamma(7) & -\gamma(5) & -\gamma(3) & -\gamma(1) \\ \gamma(2) & \gamma(6) & -\gamma(6) & -\gamma(2) & -\gamma(2) & -\gamma(6) & \gamma(6) & \gamma(2) \\ \gamma(3) & -\gamma(7) & -\gamma(1) & -\gamma(5) & \gamma(5) & \gamma(1) & \gamma(7) & -\gamma(3) \\ \gamma(4) & -\gamma(4) & -\gamma(4) & \gamma(4) & \gamma(4) & -\gamma(4) & -\gamma(4) & \gamma(4) \\ \gamma(5) & -\gamma(1) & \gamma(7) & \gamma(3) & -\gamma(3) & -\gamma(7) & \gamma(1) & -\gamma(5) \\ \gamma(6) & -\gamma(2) & \gamma(2) & -\gamma(6) & -\gamma(6) & \gamma(2) & -\gamma(2) & \gamma(6) \\ \gamma(7) & -\gamma(5) & \gamma(3) & -\gamma(1) & \gamma(1) & -\gamma(3) & \gamma(5) & -\gamma(7) \end{bmatrix}. \tag{3}$$

We will index the rows of C_8 from 0 to 7. Let $P_{(8,1)}$ be the permutation matrix acting on the rows of C_8 reordering them as follows: 0, 2, 4, 6, 1, 3, 5, 7. Let $P_{(8,2)}$ be the permutation matrix acting on the columns of C_8 by keeping the first four columns fixed and reversing the order of the last four columns. Let I_n denote the $n \times n$ identity matrix,

$$F = \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}$$

and define $R_8 = F \otimes I_4$. (Recall, in general, if A is the $m \times n$ matrix $(a_{i,j})$ and B is the $p \times q$ matrix $(b_{i,j})$, then the tensor product $A \otimes B$ is the $mp \times nq$ matrix composed of the $m \times n$ blocks $(a_{i,j}B)$.) Then

$$P_{(8,1)}K_8P_{(8,2)}R_8 = \begin{bmatrix} \gamma(4) & \gamma(4) & \gamma(4) & \gamma(4) & & & & \\ \gamma(2) & \gamma(6) & -\gamma(6) & -\gamma(2) & & & & \\ \gamma(4) & -\gamma(4) & -\gamma(4) & \gamma(4) & & & & \\ \gamma(6) & -\gamma(2) & \gamma(2) & -\gamma(6) & & & & \\ & & & & \gamma(1) & \gamma(3) & \gamma(5) & \gamma(7) \\ & & & & \gamma(3) & -\gamma(7) & -\gamma(1) & -\gamma(5) \\ & & & & \gamma(5) & -\gamma(1) & \gamma(7) & \gamma(3) \\ & & & & \gamma(7) & -\gamma(5) & \gamma(3) & -\gamma(1) \end{bmatrix}. \tag{4}$$

In order to understand the structure of the bottom right 4×4 block, which we shall label $\tilde{G}_4$, we digress and look at the ring structure of $\mathbb{Z}_{32}$, the integers between 0 and 31 with addition and multiplication modulo 32. The odd integers in $\mathbb{Z}_{32}$ form a group U_{32} under multiplication modulo 32, called the group of units of $\mathbb{Z}_{32}$. It is well known that U_{32} is isomorphic to $\mathbb{Z}_2 \oplus \mathbb{Z}_8$. Three can be taken as a generator for the $\mathbb{Z}_8$ summand in the direct sum representation for U_{32}, and we compute the cyclic subgroup of U_{32} of order 8 to be the powers of 3: {1, 3, 9, 27, 17, 19, 25, 11}. Thus, the bottom right 4×4 block can be written as

$$\begin{pmatrix} \gamma(3^0) & \gamma(3^1) & \gamma(3^3) & -\gamma(3^2) \\ \gamma(3^1) & \gamma(3^2) & -\gamma(3^0) & -\gamma(3^3) \\ \gamma(3^3) & -\gamma(3^0) & -\gamma(3^2) & \gamma(3^1) \\ -\gamma(3^2) & -\gamma(3^3) & \gamma(3^1) & -\gamma(3^0) \end{pmatrix}. \tag{5}$$

Observe that $\gamma(3^{4+j}) = -\gamma(3^j)$, $j = 0, 1, 2, 3$. Also $\gamma(9) = -\gamma(7)$ and $\gamma(27) = \gamma(5)$. Define

$$P_{(4,1)} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & -1 \\ 0 & 0 & 1 & 0 \end{pmatrix}.$$

$P_{(4,1)}$ is defined so that its action on the column vector $(\gamma(1)\gamma(3)\gamma(5)\gamma(7))^t$ is precisely $(\gamma(3^0)\gamma(3^1)\gamma(3^2)\gamma(3^3))^t$, thereby reflecting the ring structure in $\mathbb{Z}_{32}$. Then,

$$P_{(4,1)}\tilde{G}_4P_{(4,1)}^{-1} = \begin{pmatrix} \gamma(3^0) & \gamma(3^1) & \gamma(3^2) & \gamma(3^3) \\ \gamma(3^1) & \gamma(3^2) & \gamma(3^3) & \gamma(3^4) \\ \gamma(3^2) & \gamma(3^3) & \gamma(3^4) & \gamma(3^5) \\ \gamma(3^3) & \gamma(3^4) & \gamma(3^5) & \gamma(3^6) \end{pmatrix}$$

$$= \begin{pmatrix} \gamma(3^0) & \gamma(3^1) & \gamma(3^2) & \gamma(3^3) \\ \gamma(3^1) & \gamma(3^2) & \gamma(3^3) & -\gamma(3^0) \\ \gamma(3^2) & \gamma(3^3) & -\gamma(3^0) & -\gamma(3^1) \\ \gamma(3^3) & -\gamma(3^0) & -\gamma(3^1) & -\gamma(3^2) \end{pmatrix}. \tag{6}$$

The last matrix above is a signed-circulant matrix. Finally, reversing the order of the columns in $P_{(4,1)}\tilde{G}_4P_{(4,1)}^{-1}$ yields

$$G_4 = \begin{pmatrix} \gamma(3^3) & \gamma(3^2) & \gamma(3^1) & \gamma(3^0) \\ -\gamma(3^0) & \gamma(3^3) & \gamma(3^2) & \gamma(3^1) \\ -\gamma(3^1) & -\gamma(3^0) & \gamma(3^3) & \gamma(3^2) \\ -\gamma(3^2) & -\gamma(3^1) & -\gamma(3^0) & \gamma(3^3) \end{pmatrix}. \tag{7}$$

The matrix G_4 can be viewed as an element in the regular representation of the polynomial ring in the variable u moduluo $u^4 + 1$. To be precise, if

$$\begin{pmatrix} w_0 \\ w_1 \\ w_2 \\ w_3 \end{pmatrix} = \begin{pmatrix} \gamma(3^3) & \gamma(3^2) & \gamma(3^1) & \gamma(3^0) \\ -\gamma(3^0) & \gamma(3^3) & \gamma(3^2) & \gamma(3^1) \\ -\gamma(3^1) & -\gamma(3^0) & \gamma(3^3) & \gamma(3^2) \\ -\gamma(3^2) & -\gamma(3^1) & -\gamma(3^0) & \gamma(3^3) \end{pmatrix} \cdot \begin{pmatrix} v_0 \\ v_1 \\ v_2 \\ v_3 \end{pmatrix} \tag{8}$$

then also

$$(\gamma(3^3) - \gamma(3^0)u - \gamma(3^1)u^2 - \gamma(3^2)u^3) \cdot (v_0 + v_1u + v_2u^2 + v_3u^3) = (w_0 + w_1u + w_2u^2 + w_3u^3) \bmod u^4 + 1. \tag{9}$$

Observe that the first factor in the above equation can be written as

$$\sum_{j=0}^{3} \gamma(3^{3+j})u^j. \tag{10}$$

The top left 4×4 block of (4) equals $\sqrt{2}$ times the 4-point DCT matrix C_4. It itself yields a similar block diagonalization. Namely, let $P_{(4,2)}$ be the permutation matrix acting on the rows of C_4 reordering them as follows: 0, 2, 1, 3. Let $P_{(4,3)}$ be the permutation matrix acting on the columns of C_4 by keeping the first two columns fixed and reversing the order of the last two columns. Let $R_{(4,1)} = F \otimes I_2$. Then

$$P_{(4,2)}(\sqrt{2}\, C_4)P_{(4,3)}R_{(4,1)} = 2\begin{pmatrix} \gamma(4) & \gamma(4) & & \\ \gamma(4) & -\gamma(4) & & \\ & & \gamma(2) & \gamma(6) \\ & & \gamma(6) & -\gamma(2) \end{pmatrix}. \tag{11}$$

Notice that the 2×2 block on the bottom right is again a signed-circuit matrix. We will define G_2 to be this block, after the order of its columns has been reversed:

$$G_2 = \begin{pmatrix} \gamma(6) & \gamma(2) \\ -\gamma(2) & \gamma(6) \end{pmatrix}. \tag{12}$$

The matrix G_2 can also be viewed as an element in the regular representation of a polynomial quotient ring. Namely, if

$$\begin{pmatrix} w_0 \\ w_1 \end{pmatrix} = \begin{pmatrix} \gamma(6) & \gamma(2) \\ -\gamma(2) & \gamma(6) \end{pmatrix} \begin{pmatrix} v_0 \\ v_1 \end{pmatrix} \tag{13}$$

then also

$$(\gamma(6) - \gamma(2)u)(v_0 + v_1u) = (w_0 + w_1u) \bmod u^2 + 1. \tag{14}$$

Also, the 2×2 subblock on the top left of (11) is the 2-point DCT matrix C_2, and it yields the diagonalization

$$C_2F = 2\begin{pmatrix} \gamma(4) & \\ & \gamma(4) \end{pmatrix}. \tag{15}$$

The 1×1 subblock $(\gamma(4))$ will be denoted by G_1. Putting all these factorizations together yields, after slight arithmetic manipulation, the following factorization for the 8-point DCT matrix:

$$C_8 = P_8K_8B \tag{16}$$

where P_8 is the signed-permutation matrix

$$P_8 = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \end{bmatrix} \tag{17}$$

$$K_8 = \tfrac{1}{2}\begin{pmatrix} G_1 & & & \\ & G_1 & & \\ & & G_2 & \\ & & & G_4 \end{pmatrix} \tag{18}$$

and B is the rational matrix

$$B = B_1 B_2 B_3 \tag{19}$$

where

$$B_1 = \begin{bmatrix} 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 \end{bmatrix} \tag{20}$$

$$B_2 = \begin{bmatrix} 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & -1 & 0 & 0 & 0 & 0 \\ 0 & 1 & -1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \tag{21}$$

$$B_3 = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & -1 \\ 0 & 1 & 0 & 0 & 0 & 0 & -1 & 0 \\ 0 & 0 & 1 & 0 & 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 1 & -1 & 0 & 0 & 0 \end{bmatrix} \tag{22}$$

This factorization leads to an algorithm for computing the product of an arbitrary vector by C_8. First we compute the product by the matrix B, which can be done (via its factorization) with $2 + 4 + 8 = 14$ additions. We then compute the product of the result by K_8; the details will be discussed shortly. And we finish with a signed permutation.

Multiplication by K_8 will be done by computing independently the various products by $(1/2)G_j$. Each of these is equivalent to multiplication of polynomials modulo an irreducible polynomial, for which fast algorithms are known [27]. One way of doing these is via the Toom–Cooke algorithms, which is a version of Lagrange interpolation. Computing the product by $(1/2)G_1$ requires one "essential" (in the language of [27]) multiplication. A product by $(1/2)G_2$) can be done with three essential multiplications. A product by $(1/2)G_4$ can be done with seven essential multiplications, but quite a few additions.

The product by $(1/2)G_2$ can be done using a standard "rotator" product. We use the identity

$$\begin{pmatrix} x_0 & -x_1 \\ x_1 & x_0 \end{pmatrix}\begin{pmatrix} y_0 \\ y_1 \end{pmatrix} = \begin{pmatrix} 1 & -1 & 0 \\ 0 & 1 & 1 \end{pmatrix}\begin{pmatrix} (x_0 + x_1)y_0 \\ x_1(y_0 + y_1) \\ (x_0 - x_1)y_1 \end{pmatrix}. \tag{23}$$

The sums involving the x_j are precomputed; the above suggests an algorithm with 3 additions and 3 multiplications.

As for the product by $(1/2)G_4$, the minimal algorithm using 7 multiplications is quite impractical. An algorithm using 9 multiplications can be obtained using the nesting procedure. For the general problem of multiplying polynomials modulo $u^4 + 1$, partition the representation matrix–vector product

$$\begin{pmatrix} x_0 & -x_3 & -x_2 & -x_1 \\ x_1 & x_0 & -x_3 & -x_2 \\ x_2 & x_1 & x_0 & -x_3 \\ x_3 & x_2 & x_1 & x_0 \end{pmatrix}\begin{pmatrix} y_0 \\ y_1 \\ y_2 \\ y_3 \end{pmatrix} \tag{24}$$

into 2×2 blocks of size 2×2 each, say

$$\begin{pmatrix} X_0 & -X_1 \\ X_1 & X_0 \end{pmatrix}\begin{pmatrix} Y_0 \\ Y_1 \end{pmatrix} \tag{25}$$

with

$$X_0 = \begin{pmatrix} x_0 & -x_3 \\ x_1 & x_0 \end{pmatrix} \quad X_1 = \begin{pmatrix} x_2 & x_1 \\ x_3 & x_2 \end{pmatrix} \tag{26}$$

and

$$Y_0 = \begin{pmatrix} y_0 \\ y_1 \end{pmatrix} \quad Y_1 = \begin{pmatrix} y_2 \\ y_3 \end{pmatrix}. \tag{27}$$

We can use the recipe of (23) but replace each product with a matrix–vector product and each sum with a vector sum. The matrix–vector products are all of the form

$$\begin{pmatrix} x_0 & x_1 \\ x_2 & x_0 \end{pmatrix}\begin{pmatrix} y_0 \\ y_1 \end{pmatrix} = \begin{pmatrix} 1 & -1 & 0 \\ 1 & 0 & 1 \end{pmatrix}\begin{pmatrix} x_0(y_0 + y_1) \\ (x_0 - x_1)y_1 \\ (x_2 - x_0)y_0 \end{pmatrix} \tag{28}$$

which can be done with 3 multiplies and 3 additions (the sums involving the x_j are precomputed). Since a rotator

can be computed with 3 additions and 3 multiplies, the matrix-vector product of (24) can be done with $3 \times 3 = 9$ multiplications and $(3 \times 3) + (3 \times 3) = 18$ additions. Thus, with this implementation, the product by C_8 can be done with 14 multiplications and 35 additions. An attractive feature of this algorithm is that each computation path contains only one multiplication. That is, the computation never involves products of factors which are themselves sums of products. This is significant when one is concerned about bit requirements for accuracy of computation.

Alternately, the product of $(1/2)G_4$ with a vector can be done using the following identity:

$$G_4 = \tfrac{1}{2} D_4^{-1} H_{4,1} \begin{pmatrix} 1 & & \\ & G_1 & \\ & & G_2 \end{pmatrix} H_{4,2} \qquad (29)$$

where

$$D_4 = \begin{pmatrix} \gamma(5) & & & \\ & \gamma(1) & & \\ & & \gamma(3) & \\ & & & \gamma(7) \end{pmatrix}$$

$$H_{4,1} = \begin{pmatrix} 1 & 1 & -1 & 0 \\ -1 & 1 & 0 & 1 \\ -1 & -1 & -1 & 0 \\ 1 & -1 & 0 & 1 \end{pmatrix}$$

and

$$H_{4,2} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 \\ 1 & 0 & 0 & -1 \\ 0 & 1 & -1 & 0 \end{pmatrix}.$$

Thus, the product by $(1/2)G_4$ can be done using 3 additions (multiplication by $H_{4,2}$), followed by one multiplication by $(1/4)$, one by $(1/4)\gamma(4)$, and one by a rotator (multiplication by $[1/4](1 \oplus G_1 \oplus G_2)$), followed by 6 more additions (multiplication by $H_{4,1}$), followed by 4 multiplications (multiplication by D_4^{-1}). Altogether, we can compute the product by $(1/2)G_4$ with 8 multiplications and 12 additions, and hence the product by C_8 with 13 multiplications and 29 additions. This algorithm uses considerably fewer additions than the previous one, and one less multiplication, but some of the multiplications are nested.

Remark 2.1: Various similar algorithms in the literature for the DCT on 8 points [5], [24], [21] use 13 multiplications (some of which are nested) and 29 additions. Note when reading these references that they claim 12 multiplications because they renormalize the top output; in their definition for the DCT matrix, the first row of the matrix in (3) is rational. Their DCT matrix is no longer orthogonal. The constructions given here, when applied to this renormalized DCT matrix, will yield an analogous algorithm with one multiplication reduced to a multiplication by 1.

Remark 2.2: Our factorization into a direct sum of polynomial products could have been obtained by noticing that the N-point DCT matrix is essentially embedded in a $4N$-point DFT. This was first pointed out by [11]. However, we have decided here to obtain the factorization directly from the definition of the DCT.

Remark 2.3: The direct sum in equation 18 could be viewed as a system of 4 polynomial products modulo $u - 1$, $u + 1$, $u^2 + 1$, and $u^4 + 1$. It is well known [27] that these can be brought to a cyclic convolution (i.e., a single polynomial product modulo $u^8 - 1$) using only rational operations. The relationship between the DCT and cyclic convolution was already pointed out in [7].

Observe that (29) yields a procedure for transforming the product by G_4, which can be thought of as the "core" half of the 8-point DCT computation, into essentially a 4-point DCT followed by multiplication by a diagonal matrix. Similarly, G_2 can be factored to a diagonal matrix times something which is essentially a 2-point DCT matrix. Namely,

$$G_2 = \tfrac{1}{2} \begin{pmatrix} \gamma(6) & \\ & \gamma(2) \end{pmatrix}^{-1} \begin{pmatrix} 1 & 1 \\ -1 & 1 \end{pmatrix} \cdot \begin{pmatrix} 1 & \\ & G_1 \end{pmatrix} \begin{pmatrix} 1 & 0 \\ -1 & 1 \end{pmatrix}. \qquad (30)$$

This factorization gives an alternate method for computing the product by G_2 with 3 additions and 3 multiplications, but these are nested. In general, G_{2^k} can be factored to a diagonal matrix times a matrix which is essentially the core of $C_{2^{k-1}}$, thereby yielding a recursive algorithm for the DCT on 2^m points. More on this in Section VII.

III. The 2-D DCT on 8×8 Points

Computation of the 8×8 DCT involves the product of the matrix $C_8 \otimes C_8$ with a 64-point vector. A standard theorem on tensor products [27] applied to (16) yields the identity

$$C_8 \otimes C_8 = (P_8 \otimes P_8)(K_8 \otimes K_8)(B \otimes B). \qquad (31)$$

Also,

$$K_8 \otimes K_8 = \tfrac{1}{4} \oplus G_j \otimes G_k \qquad (32)$$

where $\oplus$ denotes the matrix direct sum, and the indices j, k run through the values 1, 1, 2, 4 in lexicographic order. Equation (31) suggests the following algorithm for computing the product by $C_8 \otimes C_8$: compute the product

by $B \otimes B$ using, say, the row-column method, with $2 \times 8 \times 14 = 224$ additions. Then compute separately the products by $G_j \otimes G_k$, and finish with a signed-permutation defined by $P_8 \otimes P_8$. (The factor $1/4$ can either be computed at the end with shifts or, preferably, incorporated into the products by the $G_i \otimes G_j$. We will therefore ignore this factor in the ensuing discussion). The key to our new algorithms is that the products by $G_j \otimes G_k$ can be done much more efficiently than by the row-column method when both j and k are greater than or equal to 2. The remainder of this section carefully develops algorithms for computing these products. As is typical with fast algorithms for signal processing, these algorithms are based upon matrix factorizations. These factorizations are based on some deep algebraic structures, which we discuss here in great detail. However, we feel that after plodding through the motivating example, the general results which appear in Section VII will be straighforward.

Let us consider first the simplest case, $G_2 \otimes G_2$, in considerable detail. Since the product of a 2-vector by G_2 can be done with 3 multiplications and 3 additions, the product of a 4-vector by $G_2 \otimes G_2$ can be done in "row-column" fashion using 4 products of 2-vectors by G_2, hence with 12 multiplications and 12 additions.

We can improve upon this. By direct computation, using the trigonometric identity

$$2\gamma(a)\gamma(b) = \gamma(a + b) + \gamma(a - b) \tag{33}$$

we obtain

$$G_2 \otimes G_2 = \begin{pmatrix} \gamma(6) & \gamma(2) \\ -\gamma(2) & \gamma(6) \end{pmatrix} \otimes \begin{pmatrix} \gamma(6) & \gamma(2) \\ -\gamma(2) & \gamma(6) \end{pmatrix}$$

$$= \begin{pmatrix} \gamma(6)\gamma(6) & \gamma(6)\gamma(2) & \gamma(2)\gamma(6) & \gamma(2)\gamma(2) \\ -\gamma(6)\gamma(2) & \gamma(6)\gamma(6) & -\gamma(2)\gamma(2) & \gamma(2)\gamma(6) \\ -\gamma(2)\gamma(6) & -\gamma(2)\gamma(2) & \gamma(6)\gamma(6) & \gamma(6)\gamma(2) \\ \gamma(2)\gamma(2) & -\gamma(2)\gamma(6) & -\gamma(6)\gamma(2) & \gamma(6)\gamma(6) \end{pmatrix}$$

$$= \tfrac{1}{2}\begin{pmatrix} \gamma(0) - \gamma(4) & \gamma(4) & \gamma(4) & \gamma(0) + \gamma(4) \\ -\gamma(4) & \gamma(0) - \gamma(4) & -\gamma(0) - \gamma(4) & \gamma(4) \\ -\gamma(4) & -\gamma(0) - \gamma(4) & \gamma(0) - \gamma(4) & \gamma(4) \\ \gamma(0) + \gamma(4) & -\gamma(4) & -\gamma(4) & \gamma(0) - \gamma(6) \end{pmatrix}$$

$$= \tfrac{1}{2}\begin{pmatrix} 1 & 0 & 0 & 1 \\ 0 & 1 & -1 & 0 \\ 0 & -1 & 1 & 0 \\ 1 & 0 & 0 & 1 \end{pmatrix} + \tfrac{1}{2}\begin{pmatrix} -\gamma(4) & \gamma(4) & \gamma(4) & \gamma(4) \\ -\gamma(4) & -\gamma(4) & -\gamma(4) & \gamma(4) \\ -\gamma(4) & -\gamma(4) & -\gamma(4) & \gamma(4) \\ \gamma(4) & -\gamma(4) & -\gamma(4) & -\gamma(4) \end{pmatrix}. \tag{34}$$

Multiplying a vector by the first summand of the last expression above can be done with 2 additions (and/or subtractions) and 2 multiplications by $1/2$. Multiplying a vector by the second summand can be done with 4 additions and 2 multiplications by $\gamma(4)/2$. These could then be combined with 4 additions. Hence, multiplying a vector by $G_2 \otimes G_2$ can be done with 10 additions, 2 multiplications, and 2 multiplications by $1/2$.

We see two things from the construction above. First, the two summands were matrices of rank 2 and there is a change of basis which makes them sparse. Second, the particular number theoretic properties of the cosine function yielded enormous simplifications. From a dense matrix of irrational numbers, we obtained a sparse system with very few nonrational entries. We will next obtain an explicit recipe for achieving this sparseness, which will extend to the general case. The number theoretic simplifications will fall out of the recipe. A general theory accounting for these is beyond the scope of this paper; the interested reader is referred to [10] where we describe the simplifications using a totally different approach.

We should observe two things about the matrix G_2. First, it is of a very special form: the diagonal entries are identical, and the antidiagonal terms are negatives of each other. Second, the entries in the matrix satisfy an algebraic relation, namely, $\gamma(2)^2 + \gamma(6)^2 = 1$.

As for the first observation regarding the special form of the matrix, the set of all such matrices with entries in some field Φ form an algebra over Φ. That is, they are closed under multiplication and Φ-linear combinations. The first assertion is easily verified:

$$\begin{pmatrix} f_0 & -f_1 \\ f_1 & f_0 \end{pmatrix}\begin{pmatrix} g_0 & -g_1 \\ g_1 & g_0 \end{pmatrix} = \begin{pmatrix} f_0g_0 - f_1g_1 & -(f_1g_0 + f_0g_1) \\ f_1g_0 + f_0g_1 & f_0g_0 - f_1g_1 \end{pmatrix} \tag{35}$$

and the second assertion is obvious. Let us call this algebra $\beta_2(\Phi)$. If Φ is an extension of $\mathbb{Q}$, then the set $\beta_2(\mathbb{Q})$ of matrices of the form in (35) whose entries are all in $\mathbb{Q}$ is a subalgebra (in fact, a field) of $\beta_2(\Phi)$.

Consider next the algebra $\mathbb{Q}[u]/\langle u^2 + 1\rangle$, whose elements are all linear polynomials in u with coefficients from $\mathbb{Q}$, with operations addition and multiplication modulo $u^2 + 1$. Observe that

$$(f_0 + f_1 u)(g_0 + g_1 u) = (f_0 g_0 - f_1 g_1) + (f_1 g_0 + f_0 g_1)u \bmod (u^2 + 1). \tag{36}$$

Equations (35) and (36) highlight the isomorphism

$$\rho_2 \colon \mathbb{Q}[u]/\langle u^2 + 1\rangle \to \beta_2(\mathbb{Q}) \tag{37}$$

given by

$$\rho_2(f_0 + f_1 u) = \begin{pmatrix} f_0 & -f_1 \\ f_1 & f_0 \end{pmatrix}. \tag{38}$$

We use the subscript 2 to highlight the fact that the range of ρ_2 is in $M_2(\mathbb{Q})$, the ring of 2×2 matrices with entries from $\mathbb{Q}$. That ρ_2 is a homomorphism is verified by noting that for every $p_1, p_2 \in \mathbb{Q}[u]/\langle u^2 + 1\rangle$,

$$\rho_2(p_1 + p_2) = \rho_2(p_1) + \rho_2(p_2) \tag{39}$$

$$\rho_2(p_1 p_2) = \rho_2(p_1)\rho_2(p_2). \tag{40}$$

And it is easy to verify that ρ_2 is one to one and onto.

We can write

$$\Phi[u]/\langle u^2 + 1\rangle \cong \mathbb{Q}[u]/\langle u^2 + 1\rangle \otimes \Phi. \tag{41}$$

This notation highlights the fact that, whereas the coefficients of the polynomials in $\Phi[u]/\langle u^2 + 1\rangle$ are from Φ, the modulo $u^2 + 1$ structure is defined over $\mathbb{Q}$ (that is, the polynomial $u^2 + 1$ has rational coefficients). In fact, $\mathbb{Q}[u]/\langle u^2 + 1\rangle$ is naturally embedded in $\Phi[u]/\langle u^2 + 1\rangle$.

Denote by Ψ the field extension of $\mathbb{Q}$ obtained by adjoining i to it. Then

$$\Psi \cong \mathbb{Q}[u]/\langle u^2 + 1\rangle \tag{42}$$

ρ_2 can be viewed as a homomorphism of Ψ into $\beta_2(\mathbb{Q})$ with

$$\rho_2(i) = \begin{pmatrix} 0 & -1 \\ 1 & 0 \end{pmatrix}. \tag{43}$$

We can extend ρ_2 to a homomorphism

$$\tilde{\rho}_2 \colon M_n(\Psi) \to M_{2n}(\mathbb{Q}) \tag{44}$$

by having ρ_2 act independently on the entries of matrices. Thus, for $M = (m_{i,j}) \in M_n(\Psi)$,

$$\tilde{\rho}_2(M) = (\rho_2(m_{i,j})). \tag{45}$$

It is not difficult to verify that $\tilde{\rho}_2$ is a ring homomorphism.

Now consider the matrix

$$V_2 = \begin{pmatrix} 1 & i \\ 1 & -i \end{pmatrix} \tag{46}$$

and let

$$X = \begin{pmatrix} x_0 & -x_1 \\ x_1 & x_0 \end{pmatrix} \tag{47}$$

where x_0, x_1 are elements in any extension of $\mathbb{Q}$. Direct computation yields the well-known identity

$$V_2 X V_2^{-1} = \begin{pmatrix} x_0 + ix_1 & 0 \\ 0 & x_0 - ix_1 \end{pmatrix}. \tag{48}$$

This identity is valid, in particular, for all $x_0, x_1 \in \Phi$. We say that V_2 diagonalizes the algebra $\beta_2(\Phi)$.

One can verify the following identity by direct computation:

$$D_{2,2} \stackrel{\text{def}}{=} \tilde{\rho}_2(V_2)(G_2 \otimes G_2)\tilde{\rho}_2(V_2)^{-1} = \begin{pmatrix} \gamma(6)\rho_2(1) - \gamma(2)\rho_2(i) & \rho_2(0) \\ \rho_2(0) & \gamma(6)\rho_2(1) + \gamma(2)\rho_2(i) \end{pmatrix} \cdot (I_2 \otimes G_2). \tag{49}$$

$D_{2,2}$ is block diagonal. But, moreover, notice the similarity between the right-hand side of (48) and the first factor of the right-hand side of (49). It is as if we had substituted $x_0 = \gamma(6)$ and $x_1 = -\gamma(2)$ and then applied $\tilde{\rho}_2$ to the matrix in (48) to obtain the matrix in (49). The reason we write "as if" is subtle; see the remark after equation (61).

Explicit computation of the last expression yields further simplifications:

$$D_{2,2} = \begin{pmatrix} -\gamma(4) & \gamma(4) & 0 & 0 \\ -\gamma(4) & -\gamma(4) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}. \tag{50}$$

The block diagonalization was predicted above, but the special nature of the entries in the diagonal blocks is due to the fact that the entries in G_2 satisfy an algebraic relation.

$\tilde{\rho}_2(V_2)$ is the change of basis which transforms $G_2 \otimes G_2$ to a sparse matrix, and in the process, the number theoretic properties of the cosine function yields substantial

simplification. From (49) and (50) we obtain directly

$$(G_2 \otimes G_2) = \begin{pmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & -1 \\ -1 & 0 & 1 & 0 \end{pmatrix} \cdot \begin{pmatrix} -\gamma(4)/2 & \gamma(4)/2 & 0 & 0 \\ -\gamma(4)/2 & -\gamma(4)/2 & 0 & 0 \\ 0 & 0 & 1/2 & 0 \\ 0 & 0 & 0 & 1/2 \end{pmatrix} \cdot \begin{pmatrix} 1 & 0 & 0 & -1 \\ 0 & 1 & 1 & 0 \\ 1 & 0 & 0 & 1 \\ 0 & 1 & -1 & 0 \end{pmatrix}. \tag{51}$$

From this factorization, one can easily construct an algorithm for computing the product of a 4-vector by $G_2 \otimes G_2$ with 2 multiplications, 10 additions, and 2 shifts.

We will now generalize the above construction. Let p, $q \in \mathbb{Q}[u]$ be of degree m and n, respectively, with q irreducible, let Ψ be the field extension of $\mathbb{Q}$ obtained by adjoining to it the roots of q, and suppose that q splits p (that is, p factors into linear factors over Ψ). Let Φ be an extension of $\mathbb{Q}$. Consider the algebras $\Phi[u]/\langle p(u)\rangle$ and $\Phi[u]/\langle q(u)\rangle$. These are isomorphic to the matrix algebras α and β, generated by the companion matrices C_p and C_q of p and q, respectively, with coefficients from Φ. The companion matrix of a polynomial $p(u) = a_0 + a_1 u + \cdots + a_{m-1}u^{m-1} + u^m$ is the $m \times m$ matrix

$$C_p = \begin{pmatrix} 0 & 0 & \cdots & 0 & -a_0 \\ 1 & 0 & \cdots & 0 & -a_1 \\ 0 & 1 & \cdots & 0 & -a_2 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & \cdots & 1 & -a_{m-1} \end{pmatrix}. \tag{52}$$

Also define the matrix algebra $\beta(\mathbb{Q})$ to be the one generated by C_q with rational coefficients. $\beta(\mathbb{Q})$ is a field isomorphic to Ψ and is a subalgebra of β.

Denote the roots of $p(u)$ by $\{u_1, u_2, \cdots, u_m\}$, and let

$$V_p = \begin{pmatrix} 1 & u_1 & u_1^2 & \cdots & u_1^{m-1} \\ 1 & u_2 & u_2^2 & \cdots & u_2^{m-1} \\ \vdots & \vdots & \vdots & & \vdots \\ 1 & u_m & u_m^2 & \cdots & u_m^{m-1} \end{pmatrix}. \tag{53}$$

V_p is called the Vandermonde matrix generated by the roots of p (relative to the particularly chosen order). Because of our splitting assumption, the entries in V and V^{-1} lie in Ψ. The following identity is well known:

$$V\left(\sum_{j=0}^{m-1} x_j C_p^j\right) V^{-1} = \begin{pmatrix} d_1 & & & \\ & d_2 & & \\ & & \ddots & \\ & & & d_m \end{pmatrix} \tag{54}$$

where $d_k = \Sigma_j^{n-1} \phi_j u_k^j$.

Let ρ: $\Psi \to \beta(\mathbb{Q})$ be an isomorphism. ρ extends to a ring homomorphism

$$\tilde{\rho}\colon M_m(\Psi) \to M_{mn}(\mathbb{Q}) \tag{55}$$

by having ρ act on the entries of matrices in M_m independently. We will call application of $\tilde{\rho}$ the "blow-up" construction; in the mathematics literature it is called an induced representation. The following two identities will play central roles in the sequel. First,

$$\tilde{\rho}(V)\left\{\left(\sum_{j=0}^{m-1} x_j C_p^j\right) \otimes I_n\right\} \tilde{\rho}(V)^{-1} = \begin{pmatrix} \tilde{d}_1 & & & \\ & \tilde{d}_2 & & \\ & & \ddots & \\ & & & \tilde{d}_m \end{pmatrix} \tag{56}$$

where $\tilde{d}_k = \Sigma_j^{n-1} x_j \rho(u_k)^j$. This follows directly from (54) with the observation that the blocks in the middle factor of the left-hand side of the equation are all scalar multiples of the identity. Second, for every $A \in M_m(\Psi)$ and $B \in \beta$,

$$(I_m \otimes B)\tilde{\rho}(A) = \tilde{\rho}(A)(I_m \otimes B). \tag{57}$$

This identity follows from the fact that the image of $\tilde{\rho}$ is a block matrix with each block in β, so that each block commutes with B, and the fact that $I_m \otimes B$ is block diagonal.

Combining (56) and (57) we obtain our main block-diagonalization result. For $A = \Sigma_j \phi_j C_p^j \in \alpha$ and $B \in \beta$,

$$\begin{aligned} \tilde{\rho}(V_p)(A \otimes B)\tilde{\rho}(V_p)^{-1} &= \tilde{\rho}(V_p)(A \otimes I_n)(I_m \otimes B)\tilde{\rho}(V_p)^{-1} \\ &= \tilde{\rho}(V_p)(A \otimes I_n)\tilde{\rho}(V_p)^{-1}(I_m \otimes B) \\ &= \begin{pmatrix} \tilde{a}_1 & & & \\ & \tilde{a}_2 & & \\ & & \ddots & \\ & & & \tilde{a}_m \end{pmatrix} (I_m \otimes B) \end{aligned} \tag{58}$$

where $\tilde{a}_k = \Sigma_j \phi_j \rho(u_k)^j$.

In our block diagonalization of $G_2 \otimes G_2$ we took $p(u) = q(u) = u^2 + 1$, $\Psi = \mathbb{Q}(i)$, $\Phi = \mathbb{Q}(\gamma(2))$, $\beta \simeq \Phi[u]/\langle u^2 + 1\rangle$, and $A = B = G_2$. The particular ρ here was denoted previously by ρ_2.

Let us next consider $G_2 \otimes G_4$. This time we take $p(u) = u^2 + 1$ and $q(u) = u^4 + 1$. Let $w = e^{2\pi i/8}$. We take

$$\Psi = \mathbb{Q}(w) = \simeq \mathbb{Q}[u]/\langle u^4 + 1 \rangle \tag{59}$$

$\Phi = \mathbb{Q}(\gamma(1))$, $A = G_2$ and $B = G_4$, and this time ρ is a representation of Ψ into the ring of 4×4 matrices with entries in $\mathbb{Q}$ generated by the companion matrix

$$C_{u^4+1} = \begin{pmatrix} 0 & 0 & 0 & -1 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix} \tag{60}$$

defined by $\rho(w) = C_{u^4+1}$. We will denote this representation by ρ_4.

Again, starting with (54) and using the blow-up construction, we obtain

$$\begin{aligned} D_{2,4} &\stackrel{\text{def}}{=} \tilde{\rho}_4(V_2)(G_2 \otimes G_4)\tilde{\rho}_4(V_2)^{-1} \\ &= \begin{pmatrix} \gamma(6)\rho_4(1) - \gamma(2)\rho_4(i) & \rho_2(0) \\ \rho_2(0) & \gamma(6)\rho_4(1) + \gamma(2)\rho_4(i) \end{pmatrix} (I_2 \otimes G_4) \\ &= \begin{bmatrix} -\gamma(5) & -\gamma(1) & \gamma(3) & \gamma(7) & & & & \\ -\gamma(7) & -\gamma(5) & -\gamma(1) & \gamma(3) & & & & \\ -\gamma(3) & -\gamma(7) & -\gamma(5) & -\gamma(1) & & & & \\ \gamma(1) & -\gamma(3) & -\gamma(7) & -\gamma(5) & & & & \\ & & & & \gamma(1) & \gamma(3) & \gamma(7) & \gamma(5) \\ & & & & -\gamma(5) & \gamma(1) & \gamma(3) & \gamma(7) \\ & & & & \gamma(7) & -\gamma(5) & \gamma(1) & \gamma(3) \\ & & & & -\gamma(3) & -\gamma(7) & -\gamma(5) & -\gamma(1) \end{bmatrix}. \end{aligned} \tag{61}$$

Remark 3.1: The reader may try to extend ρ_4 to a homomorphism of $\Phi \otimes \Psi$ into Ψ, and then extend it to a homomorphism $\tilde{\rho}_4$ acting on $M_m(\Phi \otimes \Psi)$. This would be attractive, for then we would apply $\tilde{\rho}_4$ to both sides of (48). But then, what would $\rho_4(\gamma(2))$ be? We would like it to be $\gamma(2)I_4$, but it may also be $(1/2)\rho_4(w + w^{-1})$ which is a rational matrix. In other words, such an extension would not be well defined. This is why we used the expression "as if" in the paragraph after (49).

Finally, consider $G_4 \otimes G_4$. This time take $p(u) = q(u) = u^4 + 1$, $A = B = G_2$ and again $\Psi = \mathbb{Q}(w)$ and $\Phi = \mathbb{Q}(\gamma(1))$. ρ is again ρ_4, and the blow-up construction now yields

$$\begin{aligned} D_{4,4} &\stackrel{\text{def}}{=} \tilde{\rho}_4(V_4)(G_4 \otimes G_4)\tilde{\rho}_4(V_4)^{-1} \\ &= 2 \begin{pmatrix} H_1 & & & \\ & H_2 & & \\ & & H_3 & \\ & & & H_4 \end{pmatrix} \end{aligned} \tag{62}$$

with

$$H_1 = \begin{pmatrix} 0 & -\gamma(2) & 0 & \gamma(6) \\ -\gamma(6) & 0 & -\gamma(2) & 0 \\ 0 & -\gamma(6) & 0 & -\gamma(2) \\ \gamma(2) & 0 & -\gamma(6) & 0 \end{pmatrix} \tag{63}$$

$$H_2 = \begin{pmatrix} 0 & \gamma(4) & 0 & \gamma(4) \\ -\gamma(4) & 0 & \gamma(4) & 0 \\ 0 & -\gamma(4) & 0 & \gamma(4) \\ -\gamma(4) & 0 & -\gamma(4) & 0 \end{pmatrix} \tag{64}$$

$$H_3 = \begin{pmatrix} -\gamma(6) & 0 & \gamma(2) & 0 \\ 0 & -\gamma(6) & 0 & \gamma(2) \\ -\gamma(2) & 0 & -\gamma(6) & 0 \\ 0 & -\gamma(2) & 0 & -\gamma(6) \end{pmatrix} \tag{65}$$

and

$$H_4 = I_4. \tag{66}$$

Remark 3.2: We have used the expression $\tilde{\rho}_4$ in two different contexts. In (61) it was a homomorphism of $M_2(\Psi)$ into $M_8(\mathbb{Q})$, and in (62) it was a homomorphism of $M_4(\Psi)$ into $M_{16}(\mathbb{Q})$, We do not want to introduce more notation, and we anticipate no confusion. The notation highlights the fact that each $\tilde{\rho}_4$ expands each dimension by a factor of 4.

Algorithmically, the above identities imply that computing the product by $G_2 \otimes G_2$ can be done with 2 multiplications, the product by $G_2 \otimes G_4$ is rationally equivalent to 2 products by G_4, and the product by $G_4 \otimes G_4$ is

rationally equivalent to a direct sum of 4 products by G_2 and 4 products by $\gamma(4)$. In principle, therefore, the product by $G_2 \otimes G_4$ can be done with 14 multiplications, and the product by $G_4 \otimes G_4$ can be done with $4 \times 3 + 4 = 16$ multiplications. Similarly, because

$$G_4 \otimes G_2 = P_S(G_2 \otimes G_4)P_S^{-1} \tag{67}$$

where P_S is the perfect-shuffle permutation, the product by $G_4 \otimes G_2$ is algorithmically equivalent to that by $G_2 \otimes G_4$. Hence, the 2-D DCT on 8×8 points can be done with 86 multiplications. In [10] it is shown that this is the minimal number of multiplications necessary to compute the 8×8-point DCT. However, in practice, products by G_4 will be computed with 8 multiplications, following the factorization of (29), hence the total number of multiplications in a very practical implementation of the 8×8-point DCT will be 94.

From (50), (61), and (62) we obtain

$$(G_2 \otimes G_2) = (2\tilde{\rho}_2(V_2)^{-1})(\tfrac{1}{2}D_{2,2})\tilde{\rho}_2(V_2) \tag{68}$$

$$(G_2 \otimes G_4) = (2\tilde{\rho}_4(V_2)^{-1})(\tfrac{1}{2}D_{2,4})\tilde{\rho}_4(V_2) \tag{69}$$

$$(G_4 \otimes G_4) = (4\tilde{\rho}_4(V_4)^{-1})(\tfrac{1}{4}D_{4,4})\tilde{\rho}_4(V_4). \tag{70}$$

The fractional factors in front of the matrices $D_{j,k}$ do not affect the computational complexity of the algorithm, as they can be incorporated into the constants. The factors in front of the blown-up Vandermonde-inverse matrices are there so that their computations will involve only additions.

Algorithms for computing the products by the blown-up Vandermonde matrices and their inverses are based on their well-known factorizations; these are the same factorizations that lead to FFT algorithms. Namely, for example,

$$V_2 = \begin{pmatrix} 1 & i \\ 1 & -i \end{pmatrix} = \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}\begin{pmatrix} 1 & 0 \\ 0 & i \end{pmatrix} \tag{71}$$

so that

$$\tilde{\rho}_2(V_2) = \begin{pmatrix} \rho_2(1) & \rho_2(i) \\ \rho_2(1) & -\rho_2(i) \end{pmatrix} = \begin{pmatrix} \rho_2(1) & \rho_2(1) \\ \rho_2(1) & -\rho_2(1) \end{pmatrix}\begin{pmatrix} \rho_2(1) & \rho_2(0) \\ \rho_2(0) & -\rho_2(i) \end{pmatrix}. \tag{72}$$

Observe that again we used the fact that the "blow up" operation is a ring homomorphism. Thus, one can multiply a 4-point vector by $\tilde{\rho}_2(V_2)$ by first multiplying two 2-point vectors by $\rho_2(1)$ and $\rho_2(i)$, respectively, using only permutations and sign changes, and then multiplying by the matrix

$$\begin{pmatrix} \rho_2(1) & \rho_2(1) \\ \rho_2(1) & -\rho_2(1) \end{pmatrix} \tag{73}$$

using 4 additions. Similarly,

$$2\tilde{\rho}_2(V_2)^{-1} = 2\begin{pmatrix} \rho_2(1) & \rho_2(i) \\ \rho_2(1) & -\rho_2(i) \end{pmatrix}^{-1} = \begin{pmatrix} \rho_2(1) & \rho_2(0) \\ \rho_2(0) & -\rho_2(i) \end{pmatrix}\begin{pmatrix} \rho_2(1) & \rho_2(1) \\ \rho_2(1) & -\rho_2(1) \end{pmatrix}. \tag{74}$$

Thus, multiplying a 4-point vector by $2\tilde{\rho}_2(V_2)^{-1}$ can be done by first multiplying by the matrix

$$\begin{pmatrix} \rho_2(1) & \rho_2(1) \\ \rho_2(1) & -\rho_2(1) \end{pmatrix} \tag{75}$$

using 4 additions, and then multiplying two 2-point vectors by $\rho_2(1)$ and $-\rho_2(i)$, respectively, using only permutations and sign changes. Similarly, multiplication of an 8-vector by either $\tilde{\rho}_4(V_2)$ or $4\tilde{\rho}_4(V_2)^{-1}$ can be done with 8 additions. And finally, the factorization

$$V_4 = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}\begin{pmatrix} 1 & 1 & 0 & 0 \\ 1 & -1 & 0 & 0 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & -1 \end{pmatrix} \cdot \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & i \end{pmatrix}\begin{pmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 1 & 0 & -1 & 0 \\ 0 & 1 & 0 & -1 \end{pmatrix}\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & w & 0 & 0 \\ 0 & 0 & w^2 & 0 \\ 0 & 0 & 0 & w^3 \end{pmatrix} \tag{76}$$

yields an analogous blow-up identity from which we can construct an algorithm for multiplying a 16-point vector by $\tilde{\rho}_4(V_4)$ or $4\tilde{\rho}_4(V_4)^{-1}$ each with 32 additions and some permutations and sign-changes. Altogether, the algorithm just described for the 8×8-point DCT will require 94 multiplications and 454 additions. In Table I we tabulate the number of arithmetic operations our algorithm uses in each stage. Column 1 gives the number of occurrences of each "subproblem" indicated in column 2. Columns 3, 4, and 5 give, respectively, the number of multiplications, additions, and bit shifts required to perform each subproblem. Columns 5, 6, and 7 give these numbers times the entries in column 1. The next to the last row in the table gives the number of preadditions. The last row gives the total arithmetic count.

Remark 3.3: In [7] another algorithm is given for the 8×8-point DCT which uses 96 multiplications and 484

TABLE I
COUNT OF ARITHMETIC OPERATIONS FOR DCT ON 8 × 8 POINTS

No. Times	Operator	Mults	Adds	Shifts	Total Mults	Total Adds	Total Shifts
4	$G_1 \otimes G_1$	1	0	1	0	0	4
4	$\sqrt{2}\, G_2$	3	3	0	12	12	0
4	$\sqrt{2}\, G_4$	8	12	0	32	48	0
1	$G_2 \otimes G_2$	2	10	2	2	10	2
2	$G_2 \otimes G_4$	16	40	0	32	80	0
1	$G_4 \otimes G_4$	16	80	0	16	80	0
	Preadditions					224	
Totals					94	454	6

additions. The definition of the DCT there does not have the DC component normalized so that the operator there is not orthogonal; with that modification, the algorithm there is easily modified to one with 94 multiplications, and is similar in spirit to our algorithm. We do require a bit fewer additions.

IV. The Scaled DCT

In most applications, the DCT is followed by scaling and quantization. The most popular application is image data compression, where image blocks (matrices of pixel values) $(p_{i,j})$ are subjected to a DCT yielding $(\hat{p}_{i,j})$, then scaled by predetermined constants $s_{i,j}$ yielding $(\hat{p}_{i,j}/s_{i,j})$ which are then either rounded or truncated to the nearest integer and then entropy coded, usually with Huffman or some arithmetic coding scheme. Because of the scaling, we can instead of computing the DCT itself, compute rather a scaled DCT. Consider then the following factorization, which follows from (16), (29), and (30):

$$C_8 = P_8 D_8 R_{8,1} M_8 R_{8,2} \tag{77}$$

where P_8 is as given in (17), D_8 is the 8×8 diagonal matrix whose diagonal elements are, in sequence from top left to bottom right, $1/2$ times $2\gamma(0)$, $\gamma(4)$, $\gamma(6)$, $\gamma(2)$, $\gamma(5)$, $\gamma(1)$, $\gamma(3)$, $\gamma(7)$,

$$R_{8,1} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & -1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & -1 & 0 \\ 0 & 0 & 0 & 0 & -1 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & -1 & -1 & -1 & 0 \\ 0 & 0 & 0 & 0 & 1 & -1 & 0 & 1 \end{bmatrix} \tag{78}$$

$$M_8 = \begin{bmatrix} 1 & & & & & & & \\ & 1 & & & & & & \\ & & 1 & & & & & \\ & & & \gamma(4) & & & & \\ & & & & 1 & & & \\ & & & & & \gamma(4) & & \\ & & & & & & \gamma(6) & \gamma(2) \\ & & & & & & -\gamma(2) & \gamma(6) \end{bmatrix} \tag{79}$$

and

$$R_{8,2} = \tilde{B}_1 B_2 B_3 \tag{80}$$

with B_2 and B_3 as defined in (19) and

$$\tilde{B}_1 = \begin{bmatrix} 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ -1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & -1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 0 & -1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 0 & -1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 \end{bmatrix}. \tag{81}$$

Equation (77) suggests that we can compute the scaled DCT on 8 points by first computing the product by $R_{8,1} M_8 R_{8,2}$ and then incorporating the factors $P_8 D_8$ into the scaling. This is because the P_8 is just a permutation, and D_8 is diagonal, so that its action is simply pointwise multiplication. Thus we can compute the scaled-DCT on 8 points with 5 multiplications and 28 additions [2].

Similarly, because of the identity

$$\begin{aligned}(P_8 D_8 R_{8,1} M_8 R_{8,2}) \otimes (P_8 D_8 R_{8,1} M_8 R_{8,2}) \\ = ((P_8 D_8) \otimes (P_8 D_8))((R_{8,1} M_8 R_{8,2}) \otimes (R_{8,1} M_8 R_{8,2}))\end{aligned} \tag{82}$$

we see that we can compute the 2-dimensional scaled DCT on 8×8 points by first computing a product by

$$(R_{8,1} M_8 R_{8,2}) \otimes (R_{8,1} M_8 R_{8,2}) \tag{83}$$

and then incorporating the product by

$$(P_8 D_8) \otimes (P_8 D_8) \tag{84}$$

into the scaling. This again is so because

$$(P_8 D_8) \otimes (P_8 D_8) = (P_8 \otimes P_8)(D_8 \otimes D_8) \tag{85}$$

is a product of a diagonal matrix followed by a signed-permutation matrix. The actual scaled-DCT computation can be done following the formula obtained by rewriting expression (83):

$$(R_{8,1} \otimes R_{8,1})(M_8 \otimes M_8)(R_{8,2} \otimes R_{8,2}). \tag{86}$$

The preadditions and postadditions (products by $(R_{8,1} \otimes R_{8,1})$ and $(R_{8,2} \otimes R_{8,2})$) are done in row-column fashion

TABLE II
COUNT OF ARITHMETIC OPERATIONS FOR SCALED-DCT ON 8 × 8 POINTS

No. Times	Operator	Mults	Adds	Shifts	Total Mults	Total Adds	Total Shifts
16	1	0	0	0	0	0	0
16	G_1	1	0	0	16	0	0
8	G_2	3	3	0	24	24	0
4	$\sqrt{2}\, G_2$	3	3	0	12	12	0
4	$G_1 \otimes G_1$	0	0	1	0	0	4
1	$G_2 \otimes G_2$	2	10	2	2	10	2
	Preadditions					288	
	Postadditions					128	
Totals					54	462	6

with 128 and 288 additions, respectively. The core of the 8 × 8 scaled DCT is the computation of the product by $M_8 \otimes M_8$, which will not be done in row-column fashion. Rather, the first, second, third, and fifth columns of the 8 × 8 data matrix will each be multiplied by M. Each of these will involve 2 multiplications by $\gamma(4)$ plus the product by the G_2, which can be done with 3 multiplications and 3 additions. The fourth and sixth columns will be multiplied by $\gamma(4)M$. Each of these can be done with 4 multiplications by $\gamma(4)$, 2 multiplications by 2, plus the product the $\gamma(4)G_2$, which can be done with 3 multiplications and 3 additions. The seventh and eighth columns will be handled simultaneously to account for the product by $G_2 \otimes M$. A 16-dimensional column vector $\bar{v}$ is formed by interleaving the entries of these two columns. The first, second, third, and fifth pairs of entries are each multiplied by G_2, while the fourth and sixth pairs are multiplied by $\gamma(4)G_2$. Each of these takes 3 multiplications and three additions. Finally, the seventh and eighth pairs of entries are multiplied simultaneously by $G_2 \otimes G_2$, as discussed in the last paragraph of Section II, with 2 multiplications by $\gamma(4)$ and 10 additions. Altogether, the entire algorithm calls for 54 multiplications and 462 additions, plus 6 multiplications by $1/2$. Table II breaks down the arithmetic count for this computation. In the Appendix we give flowgraphs describing the algorithm.

V. INVERSE DCT'S

Because C_8 is orthogonal, $C_8^{-1} = C_8^t$, and so from (16) we have

$$C_8^{-1} = B^t K_8^t P_8^t \tag{87}$$

and

$$K_8^t = G_1 \oplus G_1 \oplus G_2^t \oplus G_4^t. \tag{88}$$

Therefore, the product of a vector by C_8^{-1} can be computed by first permuting the data (this can be incorporated into the next stage), then computing the product by K_8^t, and ending with the product by B^t (which involves additions only). Since G_2^t and G_4^t are again elements in the regular representation of the polynomial quotient rings modulo $u^2 + 1$ and $u^4 + 1$, respectively, computing their products can be done using the methods discussed in Section II.

Alternately, we can use the identities

$$G_4^t = \tfrac{1}{2} H_{4,2}^t \begin{pmatrix} 1 & & \\ & G_1 & \\ & & G_2^t \end{pmatrix} H_{4,1}^T D_4^{-1} \tag{89}$$

and

$$G_2^t = \tfrac{1}{2} \begin{pmatrix} 1 & -1 \\ 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & \\ & G_1 \end{pmatrix} \begin{pmatrix} 1 & -1 \\ 1 & 1 \end{pmatrix} \begin{pmatrix} \gamma(6) & \\ & \gamma(2) \end{pmatrix}^{-1} \tag{90}$$

to obtain algorithms for the products by G_4^t and G_2^t.

A second approach to computing the inverse DCT is via the more direct sequence of (16),

$$C_8^{-1} = B^{-1} K_8^{-1} P_8^{-1} \tag{91}$$

and

$$K_8^{-1} = 1 \oplus G_1^{-1} \oplus G_2^{-1} \oplus G_4^{-1}. \tag{92}$$

Here too G_2^{-1} and G_4^{-1} are elements in the regular representation of polynomial quotient rings modulo $u^2 + 1$ and $u^4 + 1$, respectively, so computing their products efficiently is understood. And here too one can alternately invoke the identities

$$G_4^{-1} = 2 H_{4,2}^{-1} \begin{pmatrix} 1 & & \\ & G_1^{-1} & \\ & & G_2^{-1} \end{pmatrix} H_{4,1}^{-1} D_4 \tag{93}$$

and

$$G_2^{-1} = \begin{pmatrix} 1 & 0 \\ 1 & 1 \end{pmatrix} \begin{pmatrix} 1 & \\ & G_1^{-1} \end{pmatrix} \begin{pmatrix} 1 & -1 \\ 1 & 1 \end{pmatrix} \begin{pmatrix} \gamma(6) & \\ & \gamma(2) \end{pmatrix}. \tag{94}$$

For the 2-dimensional case,

$$(C_8 \otimes C_8)^{-1} = C_8^{-1} \otimes C_8^{-1} = (B^t \otimes B^t)(K_8^t \otimes K_8^t)(P_8^t \otimes P_8^t). \tag{95}$$

The algorithmic implications of the above formula are straightforward; we will only comment on the core of the computation, the product by $(K_8^t \otimes K_8^t)$. This involves

computing products by of vectors by matrices of the form

$$(G_i^t \otimes G_j^t) = (G_i \otimes G_j)^t \tag{96}$$

so we can use the identities obtained from (68)–(70)

$$(G_2^t \otimes G_2^t) = \tilde{\rho}_2(V_2)^t(\tfrac{1}{2}D_{2,2})(2\tilde{\rho}_2(V_2)^*) \tag{97}$$

$$(G_2 \otimes G_4) = \tilde{\rho}_4(V_2)^t(\tfrac{1}{2}D_{2,4})(2\tilde{\rho}_4(V_2)^*) \tag{98}$$

$$(G_4 \otimes G_4) = \tilde{\rho}(V_4)^t(\tfrac{1}{4}D_{4,4})(4\tilde{\rho}_4(V_4)^*) \tag{99}$$

where * denotes the inverse transpose. Alternately, we can use the identity

$$C_8^{-1} \otimes C_8^{-1} = (B^{-1} \otimes B^{-1})(K_8^{-1} \otimes K_8^{-1})(P_8^{-1} \otimes P_8^{-1}) \tag{100}$$

and again use (68)–(70) to obtain simplifications for the expressions for $(G_i^{-1} \otimes G_j^{-1})$.

Similar algorithmic designs can be made for the scaled DCT. For example, from (77) we obtain

$$C_8^{-1} = R_{8,2}^t M_8^t R_{8,1}^t D_8 P_8^t \tag{101}$$

from which we get the factorization

$$\begin{aligned}(C_8 \otimes C_8)^{-1} &= ((R_{8,2}^t M_8^t R_{8,1}^t) \otimes (R_{8,2}^t M_8^t R_{8,1}^t)) \\ &\quad \cdot ((D_8 P_8^t) \otimes (D_8 P_8^t)).\end{aligned} \tag{102}$$

The factor $(D_8P_8^t) \otimes (D_8P_8^t)$ can be incorporated into the scaling. The first factor on the right-hand side of (102) is equal to

$$(R_{8,2}^t \otimes R_{8,2}^t)(M_8^t \otimes M_8^t)(R_{8,1}^t \otimes R_{8,1}^t) \tag{103}$$

and an algorithm based on this factorization can be designed as was done in Section II. An explicit algorithm using this idea is presented in detail in [9].

Alternately, we can use the factorization

$$C_8^{-1} = R_{8,2}^{-1} M_8^{-1} R_{8,1}^{-1} D_8^{-1} P_8^{-1} \tag{104}$$

and obtain other algorithms. The details are omitted. We just mention that some massaging may be useful to move some scalar factors from the inverse $R_{8,1}^{-1}$ into the core of the algorithm.

The two constructions will induce different quantization matrices, and dynamic range considerations may make one more attractive than the other. The former has a slightly faster parallel implementation (its tree has depth one less than that of the latter). These two issues are discussed in greater detail in [9].

VI. Variations on a Theme

The factorization of G_2 given in (30) is only one of four similar factorizations. The others are

$$G_2 = \tfrac{1}{2}\begin{pmatrix}\gamma(2) & \\ & \gamma(6)\end{pmatrix}^{-1}\begin{pmatrix}1 & 1\\ 1 & -1\end{pmatrix} \cdot \begin{pmatrix}1 & \\ & \gamma(4)\end{pmatrix}\begin{pmatrix}0 & 1\\ 1 & 1\end{pmatrix} \tag{105}$$

$$G_2 = \tfrac{1}{2}\begin{pmatrix}\gamma(2)+\gamma(6) & \\ & \gamma(2)-\gamma(6)\end{pmatrix}^{-1} \cdot \begin{pmatrix}1 & 1\\ -1 & 1\end{pmatrix}\begin{pmatrix}1 & \\ & 2\gamma(4)\end{pmatrix}\begin{pmatrix}1 & 1\\ 0 & 1\end{pmatrix} \tag{106}$$

and

$$G_2 = \tfrac{1}{2}\begin{pmatrix}\gamma(2)-\gamma(6) & \\ & \gamma(2)+\gamma(6)\end{pmatrix}^{-1} \cdot \begin{pmatrix}1 & 1\\ 1 & -1\end{pmatrix}\begin{pmatrix}1 & \\ & 2\gamma(4)\end{pmatrix}\begin{pmatrix}-1 & 1\\ 1 & 0\end{pmatrix}. \tag{107}$$

Each of the four factorizations will lead to a different factorization of the form in (77), which in turn, would lead to a different (but very similar) algorithm using different scalings. Likewise, we can factor G_4 in ways similar but different from (29):

$$G_4 = \tfrac{1}{2}D_{4,2}^{-1}H_{4,3}\begin{pmatrix}1 & & & \\ & \gamma(4) & & \\ & & & \\ & & & G_2\end{pmatrix}H_{4,4} \tag{108}$$

where

$$D_{4,2} = \begin{pmatrix}\gamma(3) & & & \\ & \gamma(7) & & \\ & & \gamma(5) & \\ & & & \gamma(1)\end{pmatrix}$$

$$H_{4,3} = \begin{pmatrix}1 & 1 & 0 & -1\\ -1 & 1 & -1 & 0\\ 1 & 1 & 0 & 1\\ -1 & 1 & 1 & 0\end{pmatrix}$$

$$H_{4,4} = \begin{pmatrix}0 & 0 & 1 & 0\\ 0 & -1 & 0 & 1\\ 1 & 0 & 0 & 1\\ 0 & -1 & -1 & 0\end{pmatrix}$$

$$G_4 = \tfrac{1}{2}D_{4,3}^{-1}H_{4,5}\begin{pmatrix}1 & & & \\ & \gamma(4) & & \\ & & \gamma(2)-\gamma(6) & \gamma(2)+\gamma(6)\\ & & -(\gamma(2)+\gamma(6)) & \gamma(2)-\gamma(6)\end{pmatrix} \cdot H_{4,6} \tag{109}$$

where

$$D_{4,3} = \begin{pmatrix} \gamma(3) + \gamma(5) & & & \\ & \gamma(7) + \gamma(1) & & \\ & & \gamma(5) - \gamma(3) & \\ & & & \gamma(1) - \gamma(7) \end{pmatrix}$$

$$H_{4,5} = \begin{pmatrix} 1 & 1 & 1 & 0 \\ -1 & 1 & 0 & 1 \\ 1 & 1 & -1 & 0 \\ -1 & 1 & 0 & -1 \end{pmatrix}$$

$$H_{4,6} = \begin{pmatrix} 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 1 & -1 & 0 & 0 \\ 0 & 0 & 1 & 1 \end{pmatrix}$$

and

$$G_4 = \tfrac{1}{2} D_{4,4}^{-1} H_{4,7} \begin{pmatrix} 1 & & & \\ & \gamma(4) & & \\ & & \gamma(2) - \gamma(6) & \gamma(2) + \gamma(6) \\ & & -\gamma(2) - \gamma(6)) & \gamma(2) - \gamma(6) \end{pmatrix} H_{4,8} \tag{110}$$

where

$$D_{4,4} = \begin{pmatrix} \gamma(3) - \gamma(5) & & & \\ & \gamma(7) - \gamma(1) & & \\ & & \gamma(5) + \gamma(3) & \\ & & & \gamma(1) + \gamma(7) \end{pmatrix}$$

$$H_{4,7} = \begin{pmatrix} -1 & 1 & 0 & -1 \\ 1 & 1 & 1 & 0 \\ -1 & 1 & 0 & 1 \\ 1 & 1 & -1 & 0 \end{pmatrix}$$

$$H_{4,8} = \begin{pmatrix} 1 & 0 & -1 & 0 \\ 0 & -1 & 0 & 0 \\ 1 & -1 & 0 & 0 \\ 0 & 0 & 1 & -1 \end{pmatrix}.$$

These can all be verified by direct computation. Each of these will lead to a new, yet similar, algorithm. The following identities will enable one to reduce the number of arithmetic operations in constructing these algorithms. Let

$$\hat{G}_2 = \begin{pmatrix} \gamma(2) - \gamma(6) & \gamma(2) + \gamma(6) \\ -\gamma(2) - \gamma(6)) & \gamma(2) - \gamma(6) \end{pmatrix}. \tag{111}$$

Then

$$\tilde{\rho}_2(V_2)(G_2 \otimes \tilde{G}_2)\tilde{\rho}_2(V_2)^{-1} = \begin{pmatrix} -1 & 1 & 0 & 0 \\ -1 & -1 & 0 & 0 \\ 0 & 0 & \sqrt{2} & 0 \\ 0 & 0 & 0 & \sqrt{2} \end{pmatrix} \tag{112}$$

and

$$\tilde{\rho}_2(V_2)(\hat{G}_2 \otimes \hat{G}_2)\tilde{\rho}_2(V_2)^{-1} = \begin{pmatrix} -\sqrt{2} & \sqrt{2} & 0 & 0 \\ -\sqrt{2} & -\sqrt{2} & 0 & 0 \\ 0 & 0 & 2 & 0 \\ 0 & 0 & 0 & 2 \end{pmatrix}. \tag{113}$$

As an application of these different factorizations, one may design a two-dimensional scaled DCT algorithm on 8×8 points using the factorization of (77) as one factor in the tensor product, and a second factorization in which the "core" is of the form

$$\tilde{M}_8 = \begin{bmatrix} 1 & & & & & & & \\ & 1 & & & & & & \\ & & 1 & & & & & \\ & & & 2\gamma(4) & & & & \\ & & & & 1 & & & \\ & & & & & 2\gamma(4) & & \\ & & & & & & \hat{G}_2 & \end{bmatrix}. \tag{114}$$

The multiplications in the algorithm obtained via this factorization arise from the tensor product $M_8 \otimes \tilde{M}_8$. When the details are worked out, the reader can verify that the algorithm calls for 54 multiplications and 462 additions plus 2 multiplications by 2 (the algorithm of section 4 used the same number of multiplications and additions plus 6 multiplications by $1/2$).

The various forms of factorizations yield different scaling factors which, in practice, can be incorporated into the scaling and quantization steps in such applications as image data compression. Depending on the quantization matrix for the particular application, some factorization might be more appealing than another as far as the dynamic range of the induced quantization matrix is concerned. See, for example, [9].

VII. The General Theory

Much of the material here is taken from [10], where the multiplicative complexity of the DCT on arbitrary inputs of sizes which are powers of 2 has been determined. Computing the one-dimensional DCT of size N involves multiplying an arbitrary N-dimensional real vector by the $N \times N$ matrix C_N whose (j, k)th entry is

$$\Gamma_N(j(2k+1)) \stackrel{\text{def}}{=} c(j)\cos\left(\frac{2\pi j(2k+1)}{4N}\right)$$

where $c(0) = 1/\sqrt{N}$ and $c(j) = \sqrt{2/N}$ for $1 \le j \le N-1$. Computing the multidimensional DCT on an input array of size $N_1 \times N_2 \times \cdots \times N_L$ involves multiplying an arbitrary real vector of dimension $N_1 N_2 \cdots N_L$ by the matrix $C_{N_1} \otimes C_{N_2} \otimes \cdots \otimes C_{N_L}$, where the symbol $\otimes$ denotes the tensor (Kroenecker) product. The unnormalized DCT is a scaled version of the DCT, with a defining matrix $\tilde{\kappa}_N$, is similar to the one above except that the scaling factors $c(j)$ are omitted. For notational convenience, we will use the unnormalized DCT in this section. Translating the results given here to the normalized (orthogonal) DCT is straightforward.

We have the well-known trigonometric identity

$$2\Gamma_N(a)\Gamma_N(b) = \Gamma_N(a+b) + \Gamma_N(a-b). \tag{115}$$

Taking $a = jk$ and $b = j$ in the above equation and rearranging terms, we obtain

$$\Gamma_N(j(k+1)) = 2\Gamma_N(jk)\Gamma_N(j) - \Gamma_N(j(k-1)). \tag{116}$$

This equation shows that all $\Gamma_N(jk)$ can be defined recursively as polynomials in $\Gamma_N(j)$. The following result, due to Lehmer [15], is proved in [10]. We will use the notation $(j, m) = 1$ to denote that j and m are relatively prime.

Theorem 7.1: For $(j, 4N) = 1$, the degree of $\mathbb{Q}(\Gamma_N(j))$, as an extension of the rationals $\mathbb{Q}$, is $\phi(4N)/2$, where $\phi(K)$ is the Euler function which counts the number of integers between 1 and $K-1$ which are relatively prime to K.

Corollary 7.1: For $(j, 4N) = 1$, the elements $\{\Gamma_N(0), \Gamma_N(j), \cdots, \Gamma_N([\phi(4N)/2] - 1)\}$ form a basis for $\mathbb{Q}(\Gamma_N(1))$ as a vector space over $\mathbb{Q}$.

For the special case when $N = 2^n$, an integer power of 2, $\phi(4N) = 2N$, and so the degree of the extension of $\mathbb{Q}(\Gamma_N(1))$ over $\mathbb{Q}$ is N, and the following holds:

Corollary 7.2: For $N = 2^n$, the N elements

$$1 \cup \{\Gamma_N(2^{n-k}(2j+1)): 1 \le k \le n, 0 \le j < 2^{k-1}\} \tag{117}$$

form a basis for $\mathbb{Q}(\Gamma_N(1))$ as a vector space over $\mathbb{Q}$.

Consider the ring $\mathbb{Z}_{4N}$ of integers between 0 and $4N-1$ with operations addition and multiplication modulo $4N$. The odd integers in $\mathbb{Z}_{4N}$ form a multiplicative group $\mathfrak{U}_{4N}$ called the group of units of $\mathbb{Z}_{4N}$. It is well known that $\mathfrak{U}_{4N} \simeq \mathbb{Z}_2 \oplus \mathbb{Z}_N$. Let $g \in \mathfrak{U}_{4N}$ be a generator for the summand which is isomorphic to $\mathbb{Z}_N$. Then $(g^{N/2})^2 = 1$, so that $g^{N/2} = 2N \pm 1 \in \mathfrak{U}_N$. Hence

$$\Gamma_N(g^{(N/2)+a}) = \Gamma_N(g^{N/2}g^a) = \Gamma_N(2Ng^a \pm g^a) = -\Gamma_N(g^a). \tag{118}$$

The last equality follows because g^a is odd. Since, up to sign, there are only $N/2$ distinct values $\Gamma_N(j)$ with j odd, the above equation states that the elements $\Gamma_N(g^j)$, $j = 0, 1, \cdots, (N/2)-1$, are all distinct in absolute value. This argument can be extended to show the following:

Theorem 7.2: For $N = 2^n$, the N elements

$$1 \cup \{\Gamma_N(2^{n-k}(g^j)): 1 \le k \le n, 0 \le j < 2^{k-1}\} \tag{119}$$

form a basis for $\mathbb{Q}(\Gamma_N(1))$ as a vector space over $\mathbb{Q}$.

The basis just given is, except for signs, identical to the basis given in Corollary 7.2. Hence, if we form the

$N/2$-dimensional column vectors

$$V_{N,1} = (\Gamma_N(1) \quad \Gamma_N(3) \quad \cdots \quad \Gamma_N(N-1))^t \tag{120}$$

and

$$V_{N,2} = (\Gamma_N(g^0) \quad \Gamma_N(g^1) \quad \cdots \quad \Gamma_N(g^{(N/2)-1}))^t \tag{121}$$

then there exits a signed permutation matrix P_N (each of whose rows and columns have entries which are all 0 except in one position, where it is either 1 or -1) such that

$$V_{N,2} = P_N V_{N,1}. \tag{122}$$

$$P_{(N/2),3} L_{N/2} P^{-1}_{(N/2),3} = \begin{pmatrix} \Gamma_N(g^0) & \Gamma_N(g^1) & \cdots & \Gamma_N(g^{(N/2)-1}) \\ \Gamma_N(g^1) & \Gamma_N(g^2) & \cdots & -\Gamma_N(g^0) \\ \Gamma_N(g^2) & \Gamma_N(g^3) & \cdots & -\Gamma_N(g^1) \\ \vdots & & & \vdots \\ \Gamma_N(g^{(N/2)-1}) & -\Gamma_N(g^0) & \cdots & -\Gamma_N(g^{(N/2)-2}) \end{pmatrix}.$$

Recall the trigonometric identity

$$\Gamma_N(j(2k+1)) = \begin{cases} \Gamma_N(j(2N-2k-1)) & \text{if } j \text{ is even} \\ -\Gamma_N(j(2N-2k-1)) & \text{if } j \text{ is odd.} \end{cases} \tag{123}$$

Let $P_{N,1}$ be the permutation matrix acting on N-dimensional vectors so that their odd indexed entries appear first, in order, followed by their even indexed entries, in order. Let $P_{N,2}$ be the permutation matrix acting on N-dimensional vectors by keeping the first half of their entries fixed and reversing the order of the second half. Let I_n denote the $N \times N$ identity matrix, define

$$F = \tfrac{1}{2}\begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}$$

and let $F_N = F \otimes I_{N/2}$. The following follows directly from (123).

Lemma 7.1:

$$P_{N,1} \tilde{\kappa}_N P_{N,2} F_N = \begin{pmatrix} \tilde{\kappa}_{N/2} & \\ & L_{N/2} \end{pmatrix} \tag{124}$$

where $L_{N/2}$ is the $(N/2) \times (N/2)$ matrix whose (j, k)th energy is $\Gamma_N((2j+1)(2k+1))$.

Define the $2^n \times 2^n$ matrix

$$\mathfrak{L}_n = \begin{pmatrix} 1 & & & & & \\ & L_1 & & & & \\ & & L_2 & & & \\ & & & L_4 & & \\ & & & & \ddots & \\ & & & & & L_{2^{n-1}} \end{pmatrix}.$$

We will use the direct-sum notation and write

$$\mathfrak{L}_n = 1 \oplus L_1 \oplus L_2 \oplus L_4 \oplus \cdots \oplus L_{2^{n-1}}. \tag{125}$$

Arguing inductively as above, we obtain [10],

Lemma 7.2: For $N = 2^n$, there exists a signed permutation matrix $\mathcal{P}_N$ and a rational matrix $\mathcal{R}_N$ such that

$$\mathcal{P}_N \tilde{\kappa}_N = \mathfrak{L}_N. \tag{126}$$

For $N = 2^n$, using (118) and denoting by $P_{(N/2),3}$ the permutation matrix given in (122) acting on the vector

$$(\Gamma_N(g^0) \quad \Gamma_N(g^1) \quad \cdots \quad \Gamma_N(g^{(N/2)-1}))^t \tag{127}$$

we have

Let $P_{N,4}$ be the permutation matrix acting on N-dimensional vectors by reversing the order of their entries. Let C_N be the companion matrix to the polynomial $u^n + 1$:

$$C_N = \begin{pmatrix} 0 & 0 & \cdots & 0 & -1 \\ 1 & 0 & \cdots & 0 & 0 \\ 0 & 1 & \cdots & 0 & 0 \\ \vdots & & & & \vdots \\ 0 & 0 & \cdots & 1 & 0 \end{pmatrix}.$$

Then

$$P_{(N/2),3} L_{(N/2)} P^{-1}_{(N/2),3} P_{(N/2),4} = \sum_{j=0}^{(N/2)-1} \Gamma_N(g^j) C^j_{N/2}. \tag{128}$$

Define

$$G_{2^k} = \sum_{j=0}^{2^{k-1}} \Gamma_N(2^{n-k}(2j+1)) C^j_{2^k} \tag{129}$$

and

$$\mathcal{G}_n = 1 \oplus G_1 \oplus G_2 \oplus G_4 \oplus \cdots \oplus G_{2^{n-1}}. \tag{130}$$

By the above discussion and Lemma 7.2, we have the following result [10] which generalizes the factorization of (16).

Theorem 7.3: For $N = 2^n$, there exists a signed permutation matrix $\tilde{\mathcal{P}}_N$ and a rational matrix $\tilde{\mathcal{R}}_N$ such that

$$\tilde{\kappa}_N = \tilde{\mathcal{P}}_N \mathcal{G}_n \tilde{\mathcal{R}}_N. \tag{131}$$

From (115) we obtain

$$\begin{aligned} &2\Gamma_{N/2}(2m+1)\Gamma_{N/2}((2m+1)(2n+1)) \\ &\quad = \Gamma_{N/2}((2m+1)(2n)) + \Gamma_{N/2}((2m+1)(2n+1)) \\ &\quad = \Gamma_N(n(2m+1)) + \Gamma_{N/2}((n+1)(2m+1)). \end{aligned} \tag{132}$$

Define the $N \times N$ diagonal matrix

$$D_N = 2 \begin{pmatrix} \Gamma_{2N}(1) & & & & \\ & \Gamma_{2N}(3) & & & \\ & & \Gamma_{2N}(5) & & \\ & & & \ddots & \\ & & & & \Gamma_{2N}(2N-1) \end{pmatrix}$$

and the $N \times N$ matrix

$$Y_N = \begin{pmatrix} 1 & 0 & \cdots & 0 & 0 \\ 1 & 1 & \cdots & 0 & 0 \\ 0 & 1 & \cdots & 0 & 0 \\ \vdots & & & & \vdots \\ 0 & 0 & \cdots & 1 & 0 \\ 0 & 0 & \cdots & 1 & 1 \end{pmatrix}.$$

Recalling the definition of L_N from Lemma 7.1, we have by (132)

$$D_N L_N = \tilde{\kappa}_N^t Y_N. \tag{133}$$

Define

$$\mathcal{K}_n = 1 \oplus \tilde{\kappa}_1 \oplus \tilde{\kappa}_2 \oplus \tilde{\kappa}_4 \oplus \cdots \oplus \tilde{\kappa}_{2^{n-1}} \tag{134}$$

$$\mathcal{Y}_n = 1 \oplus 1 \oplus Y_2 \oplus Y_4 \oplus \cdots \oplus Y_{2^{n-1}} \tag{135}$$

and

$$\mathcal{D}_n = 1 \oplus D_1 \oplus D_2 \oplus D_4 \oplus \cdots \oplus D_{2^{n-1}}. \tag{136}$$

Then from (133) we have

$$\mathcal{D}_n \mathcal{L}_n = \mathcal{K}_n^t \mathcal{Y}_n \tag{137}$$

which together with Lemma 7.2 yields [10] the generalization of (29), (30), and (77).

Theorem 7.4: For $N = 2^n$,

$$\tilde{\kappa}_N = \mathcal{P}_n^{-1} \mathcal{D}_n^{-1} \mathcal{K}_n^t \mathcal{Y}_n \mathcal{R}_n^{-1} \tag{138}$$

where $\mathcal{P}_n$ and $\mathcal{R}_n$ are given as in Lemma 7.2.

The multidimensional case involves the tensor product of the various contributing one dimensional cases. We will restrict our discussion here to the two-dimensional case; the general case follows readily. Computation for the $M \times N$ unnormalized DCT involves the product of the matrix $\tilde{\kappa}_M \otimes \tilde{\kappa}_N$ with an MN-point vector. Analogous to (31) and (32), we have

$$\tilde{\kappa}_M \otimes \tilde{\kappa}_N = (P_m \otimes P_n)(\mathcal{G}_m \otimes \mathcal{G}_n)(\mathcal{R}_m \otimes \mathcal{R}_n) \tag{139}$$

and

$$\mathcal{G}_m \otimes \mathcal{G}_n = \oplus\, G_j \otimes G_k. \tag{140}$$

The product by $(\mathcal{R}_m \otimes \mathcal{R}_n)$ can be done in row-column fashion, and $(P_m \otimes P_n)$ is a signed-permutation matrix. Efficient algorithms can be constructed by exploiting the algebraic structure of the tensor products $(\mathcal{G}_m \otimes \mathcal{G}_n)$.

For every integer M define $w_M = e^{2\pi i/2M}$, and denote by V_M the Vandermonde matrices

$$V_M = \begin{pmatrix} 1 & w_M & w_M^2 & \cdots & w_M^{M-1} \\ 1 & w_M^3 & w_M^6 & \cdots & w_M^{3(M-1)} \\ \vdots & & & & \vdots \\ 1 & w_M^{2M-1} & w_M^{2(2M-1)} & \cdots & w_M^{(2M-1)(M-1)} \end{pmatrix}. \tag{141}$$

Recalling C_M as the companion matrix to $u^M + 1$, we have that $V_M C_M V_M^{-1}$ *is diagonal, with the* (j, j)th entry being $\Sigma_{k=0}^{M-1} w_M^{(2j+1)k}$. For $M = 2^m$, we have from (129) that $V_M G_M V_M^{-1}$ is also diagonal.

Without loss of generality, we assume $M \le N$. Using the blow-up construction [3] we define

$$\overline{V}_{M,N} = \begin{pmatrix} I_N & C_N & C_N^2 & \cdots & C_N^{M-1} \\ I_N & C_N^3 & C_N^6 & \cdots & C_N^{3(M-1)} \\ \vdots & & & & \vdots \\ I_N & C_N^{2M-1} & C_N^{2(2M-1)} & \cdots & C_N^{(2M-1)(M-1)} \end{pmatrix} \tag{142}$$

where I_N is the $N \times N$ identity matrix. Then

$$D_{M,N} \overset{\text{def}}{=} \overline{V}_{M,N}(G_M \otimes G_N)\overline{V}_{M,N}^{-1} = \begin{pmatrix} G_{1,N} & 0 & \cdots & 0 \\ 0 & G_{2,N} & \cdots & 0 \\ \vdots & & \ddots & \vdots \\ 0 & 0 & 0 & G_{M,N} \end{pmatrix} \tag{143}$$

where the $G_{j,N}$ are all polynomials in C_N with coefficients in the field $\mathcal{Q}(\Gamma_N(1))$. Furthermore, many of the $G_{j,N}$ are sparse and some have entries which are rationally related. We have observed this for special cases in (62) through (66). We have not yet worked out the general theory that predicts these algebraic simplifications, but we can predict them using the results of [4]. In all the examples we have worked out, the final direct sum systems we obtain are identical to those is [4], even though the constructions are totally different. Tying the two together is still an open issue.

For the special cases $M = 2^m$ and $N = 2^n$ with $M \le N$ we expect to essentially reduce the computation to M distinct one-dimensional DCT's on N points. By essentially we mean that there is an overhead involving additions only. For the general multidimensional case on $M_j = 2^{m_j}$ points along each dimension, with $M_1 \le M_2 \le \cdots \le M_L$, we reduce the task to essentially a direct sum system of $M_1 M_2 \cdots M_{L-1}$ one-dimensional DCT's on M_L points.

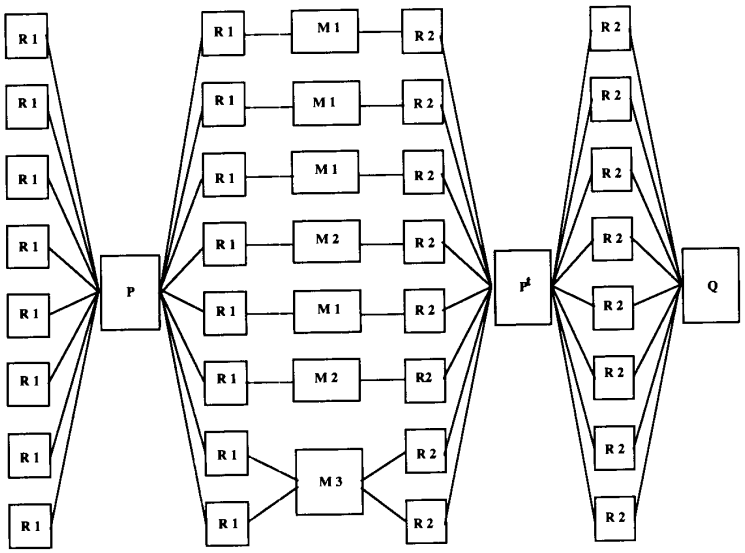

Fig. 1. Overall block diagram for scaled-DCT on 8 × 8 points.

VIII. Conclusion

We have presented efficient practical algorithms for various two-dimensional DCT based computations on 8 × 8 points. These include forward and inverse DCT's and scaled-DCT's. The algorithms are highly pipelined and parallelizable. In our own compression/decompression routines at IBM, we have chosen the scaled DCT versions on 8 × 8 points. These use 54 multiplications and 462 additions each, and also have the added property that no computation path uses more than one multiplication. Thus we are able to do all our computations with 16-b fixed-point arithmetic and still obtain sufficient accuracy (our input is 8-b maximum).

Our algorithms were based on various algebraic properties of the DCT matrix and theorems regarding the structure of tensor products of matrices which can be viewed as elements in the regular representation of certain fields defined by polynomial multiplication modulo irreducible polynomials. Finally, we have shown that our constructions may be extended to yield algorithms for DCT's and scaled-DCT's of arbitrary dimensions on input sizes which are powers of 2.

Appendix

We give here flowcharts for implementing the scaled-DCT discussed in Section IV. Fig. 1 gives the overall picture. The data are entries from an 8 × 8 matrix. Each row is passed through an addition stage $\mathfrak{R}1$ which is described explicitly in Fig. 2. Arrows indicate subtractions. The outputs are then permuted (transposed) in $\mathcal{P}$, and next the columns are passed through $\mathfrak{R}1$ first three stages of the circuit perform the "preadditions" and as can be seen, these are done in "row-column" fashion.

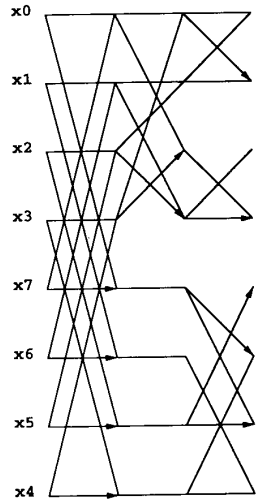

Fig. 2. Circuit element $\mathfrak{R}1$ for preadditions stage.

Next, the various resulting vectors are passed through the multipliers $\mathfrak{M}1$, $\mathfrak{M}2$, and $\mathfrak{M}3$. These circuits are detailed in Figs. 3, 4, and 5, respectively. The bottoms of Figs. 3 and 4 are circuits for complex multiplication (in this case, in fact, rotators for computing the products by G_2 and $(\sqrt{2}/2)G_2$, respectively. The constants in the multipliers are $a_0 = \sqrt{2}/2$, $a_1 = \gamma(6) - \gamma(2)$, $a_2 = -\gamma(2)$, $a_3 = \gamma(6) + \gamma(2)$, $b_1 = \sqrt{2}\, a_1/2$, $b_2 = \sqrt{2}\, a_2/2$, $b_3 = \sqrt{2}\, a_3/2$. Fig. 5 for multiplier $\mathfrak{M}3$ highlights the simultaneous processing of two columns by first interleaving them, passing four pairs of values through the rotator $\mathfrak{Q}1$ which multiplies by G_2, two pairs through the rotator $\mathfrak{Q}2$ which multiplies by $(\sqrt{2}/2)G_2$, and the remaining four

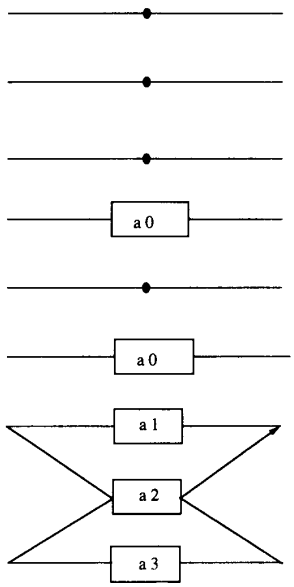

Fig. 3. Circuit for $\mathfrak{M}1$ of multiplication stage.

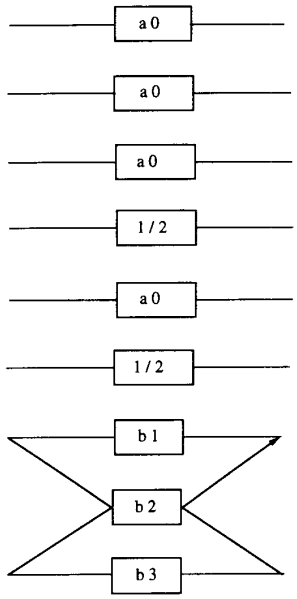

Fig. 4. Circuit for $\mathfrak{M}2$ of multiplication stage.

values are passed through a circuit $\mathfrak{Q}3$ which is given in Fig. 6. The constant in Fig. 6 is $c_0 = \sqrt{2}/4$.

The output of the multiplier stage are then processed with "postadditions" which are also done in row-column fashion. This is represented in Fig. 1 by the two columns of circuits $\mathfrak{R}2$ with the permutation $\mathcal{P}^t$ separating them. The circuit for $\mathfrak{R}2$ is given in Fig. 7. The final stage in Fig. 1 is the quantization step $\mathfrak{Q}$.

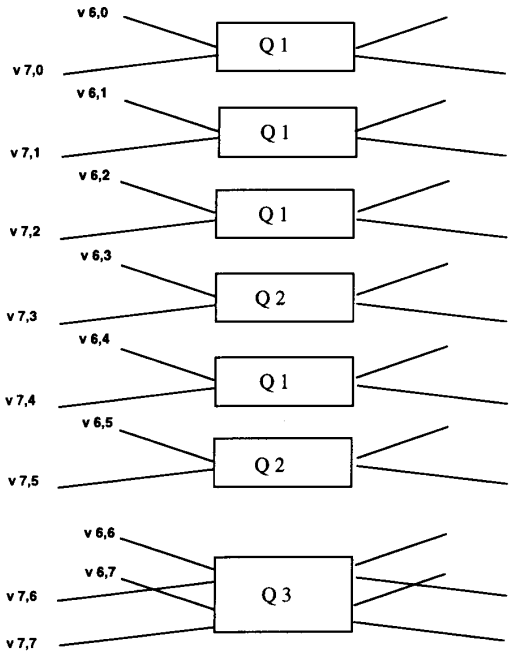

Fig. 5. Circuit for $\mathfrak{M}3$ of multiplication stage.

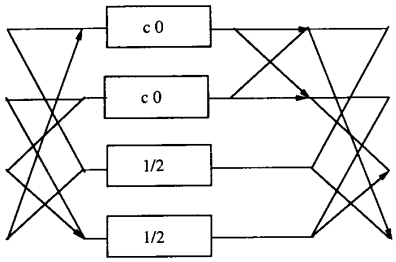

Fig. 6. Circuit for $\mathfrak{Q}3$ of multiplication stage.

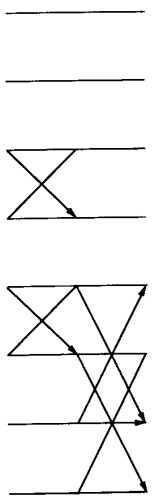

Fig. 7. Circuit element $\mathfrak{R}2$ for postadditions stage.

REFERENCES

[1] N. Ahmed, T. Natarajan, and K. R. Rao, "Discrete cosine transform," *IEEE Trans. Comput.*, vol. C-23, pp. 90-93, Jan. 1974.

[2] Y. Arai, T. Agui, and M. Nakajima, "A fast DCT-SQ scheme for images," *Trans. IEICE*, vol. E-71, no. 11, Nov. 1988.

[3] L. Auslander, E. Feig, and S. Winograd, "Abelian semi-simple algebras and algorithms for the discrete Fourier transform," *Adv. Appl. Math.*, vol. 5, pp. 31-55, 1984.

[4] L. Auslander and S. Winograd, "The multiplicative complexity of certain semilinear systems defined by polynomials," *Adv. Appl. Math.*, vol. 1, pp. 257-299, 1980.

[5] W. H. Chen, C. H. Smith, and S. C. Fralick, "A fast computational algorithm for the discrete cosine transform," *IEEE Trans. Commun.*, vol. COM-25, pp. 1004-1009, Sept. 1977.

[6] P. Duhamel, "New 2^n DCT algorithms suitable for VLSI implementation," in *Proc. ICASSP-87* (Dallas, TX), Apr. 1987, pp. 1805-1808.

[7] P. Duhamel and C. Guillemot, "Polynomial transform computation of the 2-D DCT," in *Proc. ICASSP-90* (Albuquerque. NM), 1990, pp. 1515-1518.

[8] Y. Eutani and K. Ohzeki, "A new fast algorithm for DCT," presented at the Fourth DSP Workshop, Mohonk Mountain House, New Palz, NY, Sept. 16-19, 1990.

[9] E. Feig, "A fast scaled DCT algorithm," *Proc. SPIE Int. Soc. Opt. Eng.*, vol. 1244, pp. 2-13, 1990.

[10] E. Feig and S. Winograd, "On the multiplicative complexity of discrete cosine transforms," *IEEE Trans. Inform. Theory*, vol. 38, no. 4, pp. 1387-1391, July 1992.

[11] M. T. Heiderman, *Multiplicative Complexity, Convolution, and DFT*. Springer-Verlag, 1988, pp. 116-117.

[12] H. S. Hou, "A fast recursive algorithm for computing the discrete cosine transform," *IEEE Trans. Acoust., Speech, Signal Processing*, vol. ASSP-35, no. 10, pp. 1455-1461, 1987.

[13] S. Lang, *Algebra*. Reading, MA: Addison-Wesley, 1965.

[14] B. G. Lee, "A new algorithm to compute the discrete cosine transform," *IEEE Trans. Acoust., Speech, Signal Processing*, vol. ASSP-32, no. 6, pp. 1243-1245, Dec. 1984.

[15] D. H. Lehmer, *Amer. Math. Monthly*, vol. 40, pp. 165-166, 1933; also in I. Niven, *Irrational Numbers*. New York: Wiley, 1967, pp. 37-38.

[16] C. Loeffler, A. Ligtenberg, and G. S. Moschytz, "Algorithm-architecture mapping for custom DCT chips," in *Proc. Int. Symp. Circuits Syst.* (Helsinki, Finland), June 1988, pp. 1953-1956.

[17] J. Makhoul, "A fast cosine transform in one and two dimensions," *IEEE Trans. Acoust., Speech, Signal Processing*, vol. ASSP-28, no. 1, pp. 27-34, Feb. 1980.

[18] M. J. Narasimha and A. M. Peterson, "On the computation of the discrete cosine transform," *IEEE Trans. Commun.*, vol. COM-26, no. 6, pp. 934-936, June 1978

[19] K. R. Rao and P. Yip, *Discrete Cosine Transform-Algorithms, Advantages, Applications*. New York: Academic, 1990.

[20] K. Shanmugam, "Comments on discrete cosine transforms," *IEEE Trans. Comput.*, vol. C-24, p. 759, July 1975.

[21] N. Suehiro and M. Hatori, "Fast algorithms for the DFT and other sinusoidal transforms," *IEEE Trans. Acoust., Speech, Signal Processing*, vol. ASSP-34, pp. 642-644, 1986.

[22] B. D. Tseng and W. C. Miller, "On computing the discrete cosine transform," *IEEE Trans. Comput.*, vol. C-27, pp. 966-968, Oct. 1978.

[23] M. Vetterli and H. J. Nussbaumer, "Simple FFT and DCT algorithms with reduced number of operations," *Signal Processing*, Aug. 1984.

[24] M. Vetterli, "Fast 2-D discrete cosine transform," in *Proc. ICASSP-85* (Tampa, FL), Mar. 1985.

[25] S. Winograd, "On computing the discrete Fourier transform," *Math. Comput.*, vol. 32, pp. 175-199, 1978.

[26] S. Winograd, "On the multiplicative complexity of the discrete Fourier transform," *Adv. Math.*, vol. 32, no. 2, pp. 83-117, 1979.

[27] S. Winograd, *Arithmetic Complexity of Computations*, CBMS-NSF Regional Conference Series in Applied Math., 1980.

Ephraim Feig (A'84-SM'89-F'92) received the B.S. degree in mathematics from the City College of New York in 1971 and the Ph.D. degree in mathematics from the Graduate Center of the City University of New York in 1981.

He is currently Manager of Signal Processing and Coding in the Mathematical Sciences Department at IBM Research, Yorktown Heights, NY. His interests are in image data compression and processing, coding and modulation, radar, magnetic resonance imaging, coding for error control, fast algorithms, and complexity theory. He has published extensively in all these fields and has won eight IBM Invention Achievement Awards for related patents activity. He has taught at Columbia University, CCNY, Richmond College, the College of Staten Island, New York Polytechnic, Bronx Community College, and IBM.

Shmuel Winograd (S'59-M'62-SM'73-F'74) received the Ph.D. degree in mathematics from New York University in 1968.

He is currently an IBM Fellow and Director of the Mathematical Sciences Department at IBM, T. J. Watson Research Center, Yorktown Heights, NY. His research interests are in complexity of computation with particular emphasis on signal processing and linear algebra.

Dr. Winograd received, in 1974, the IEEE CS McDowell Award; in 1978 he was elected to the National Academy of Sciences; and in 1983 he was elected to the American Academy of Arts and Sciences. In 1987 he was Docteur Honoris Causa, Institut National Polytechnic de Grenoble, and in 1989 he was elected to the American Philosophical Society.