A Universal Approach to Developing Fast Algorithm for Simplified Order-16 ICT

Jie Dong, King Ngi Ngan, Chi Keung Fong and Wai Kuen Cham
Department of Electronic Engineering
The Chinese University of Hong Kong, Shatin, Hong Kong SAR
{jdong, knngan, ckfong, wkcham}@ee.cuhk.edu.hk

***Abstract*—Simplified order-16 Integer Cosine Transform (ICT) has been proved to be an efficient coding tool especially for High-Definition (HD) video coding and is much simpler than ICT and Discrete Cosine Transform (DCT). To further reduce the computational complexity, a universal approach to developing fast algorithm mainly for, but not restricted to, simplified order-16 ICT is proposed in this paper. The fast algorithm developed by the proposed approach involves additions and shiftings only and can save about 90% of the computational time compared with matrix multiplication.**

I. INTRODUCTION

Integer Cosine Transform (ICT) is generated from Discrete Cosine Transform (DCT) by replacing the real-numbered elements of the DCT matrix with integers and still maintaining the structure, i.e., relative magnitudes and orthogonality, among the matrix elements [1]. It can be implemented using integer arithmetic without mismatch between encoder and decoder and if well-designed, can provide almost the same compression efficiency as DCT. Furthermore, ICT has fast algorithms, which can be developed in the similar way for DCT. However, the orthogonality of certain ICT depends on the elements of the transform matrix, if the ICT is larger than order-4. Due to the constraint of orthogonality, the magnitudes of elements tend to be quite large for large ICTs, e.g. order-16 ICT [2].

Simplified ICT was proposed by using the principle of dyadic symmetry and is naturally orthogonal no matter what the elements are [3]. So the elements in transform matrices can be selected without orthogonality constraint. Usually, the magnitudes of elements are designed to be very small, e.g., represented by 4~5 bits, for simple implementation. If well-designed, it has performance comparable to ICT. However, fast algorithms for simplified ICTs are not guaranteed, and little research effort has been devoted to it.

In recent years, ICT has been widely used in video coding, such as the order-4 and order-8 ICTs in H.264 [4] and the order-8 ICT in the Audio Video Coding Standard of China (AVS) [5]. According to our study, order-16 ICT can be a very efficient coding tool especially for High-Definition (HD) video coding [6]. However, in order to make a tradeoff between the performance and the computational complexity, we proposed a simplified order-16 ICT instead of ICT [6].

To further reduce computational complexity, in this paper, we extend the transform matrix of the simplified order-16 ICT in [6] to a general one and propose a universal approach to developing fast algorithms mainly for, but not restricted to, the general transform matrix. With the proposed approach, matrix multiplication that is often used for transformation is decomposed to butterfly operations by a universal method. The approaches to developing fast algorithms for forward and inverse transforms are very similar and in this paper, we describe the one for forward transforms in detail.

The remainder of the paper is organized as follows. Section II proposes the approach to developing fast algorithms for simplified order-16 ICT. Section III develops the fast algorithm for the simplified order-16 ICT in [6] by the approach introduced in Section II. Section IV analyzes the complexity and reports the experimental results, followed by the conclusion in Section V.

II. THE APROACH TO DEVELOPING FAST ALGORITHMS

A. General transform matrix for simplified order-16 ICT

We modify the structure of the order-16 DCT matrix by using the principle of dyadic symmetry and propose a general transform matrix as (1). $T_{16\times16}$ may contain at most 15 different integers, denoted by *a, b, c, ... o*, and the basis vectors of $T_{16\times16}$ are naturally orthogonal to each other. The structure of $T_{16\times16}$ is suitable for developing fast algorithms as illustrated later and also provides engineers the freedom to make a good tradeoff between the performance and the magnitudes of the elements, i.e., the computational complexity, while designing a simplified order-16 ICT. $T_{16\times16}$ has symmetry with respect to the dashed line, with even and odd symmetry alternating.

The work is sponsored in part by the Chinese University of Hong Kong Focused Investment Scheme C under Project 1903003

$$T_{16\times16}=\left[\begin{array}{cccccccc|cc}
o & o & o & o & o & o & o & o & o & o & \cdots \\
a & b & c & d & e & f & g & h & -h & -g & \cdots \\
i & j & k & l & -l & -k & -j & -i & -i & -j & \cdots \\
e & f & g & h & -a & -b & -c & -d & d & c & \cdots \\
m & n & -n & -m & -m & -n & n & m & m & n & \cdots \\
c & d & -a & -b & -g & -h & e & f & -f & -e & \cdots \\
j & -l & -i & -k & k & i & l & -j & -j & l & \cdots \\
h & g & -f & -e & d & c & -b & -a & a & b & \cdots \\
o & -o & -o & o & o & -o & -o & o & o & -o & \cdots \\
g & -h & -e & f & c & -d & -a & b & -b & a & \cdots \\
k & -i & l & j & -j & -l & i & -k & -k & i & \cdots \\
b & -a & -d & c & -f & e & h & -g & g & -h & \cdots \\
n & -m & m & -n & -n & m & -m & n & n & -m & \cdots \\
d & -c & b & -a & -h & g & -f & e & -e & f & \cdots \\
l & -k & j & -i & i & -j & k & -l & -l & k & \cdots \\
f & -e & h & -g & b & -a & d & -c & c & -d & \cdots
\end{array}\right] \quad (1)$$

B. *Decompose matrix multiplication to butterfly operations*

In image/video coding, 2-D transform can be separated into two 1-D transforms, so we focus on the fast algorithm for 1-D simplified order-16 ICT, which means matrix multiplication with a 16-point vector is decomposed to butterfly operations. The butterfly structure should have 4 stages, because 16 equals 2^4.

The butterfly structure of stage 1 can be easily developed as shown in the left block in Fig. 1. Obviously, stage 1 exploits the symmetries with respect to the dashed line in (1), so the rest 3 stages only deal with the left half of $T_{16\times16}$, denoted as $T_{16\times8}$.

The original input vector, $y=[y_0, y_1, \ldots y_{15}]$, is projected to the even symmetry basis vectors in $T_{16\times16}$ and is equivalent to the top 8 outputs of stage 1, $x=[x_0, x_1, \ldots x_7]$, are projected to the $2i^{th}$, (i=0, 1, …7), rows of $T_{16\times8}$. So the upper-right module in Fig. 1 completes the matrix multiplication, $T_{8u}\times x$, where T_{8u} is shown in (2). Similarly, y projects to the odd symmetry basis vectors in $T_{16\times16}$ is equivalent to the bottom 8 outputs of stage 1 project to the $(2i+1)^{th}$, (i=0, 1, …7), rows of $T_{16\times8}$, and the bottom-right module in Fig. 1 completes the matrix multiplication with T_{8d} as shown in (3).

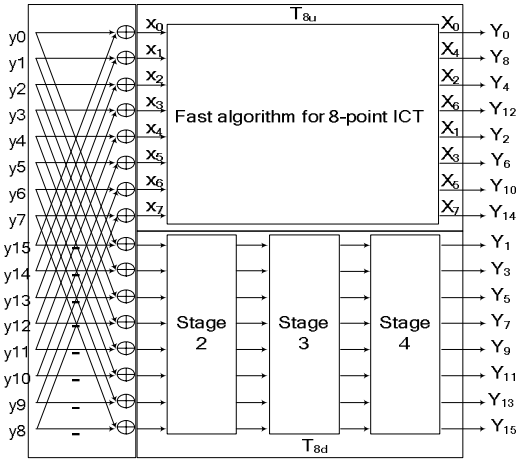

Figure 1. Flow diagram of fast simplified ICT

T_{8u} is exactly the general transform matrix of an order-8 ICT. Actually, there are some order-8 ICTs with performance comparable to DCT and also with efficient fast algorithms. For example, the set {*i, j, k, l, m, n, o*} is {12, 10, 6, 3, 8, 4, 8} [4][7] or {10, 9, 6, 2, 10, 4, 8} [5][6]. So if we set the elements {*i, j, k, l, m, n, o*} to be those in well-designed order-8 ICTs when designing a simplified order-16 ICT, it will be convenient to develop the fast algorithm from the upper-right module in Fig. 1 by borrowing the fast algorithm for the corresponding order-8 ICT. Furthermore, using an efficient order-8 ICT as a module in a simplified order-16 ICT improves the overall performance.

$$T_{8u}=\left[\begin{array}{cccccccc}
o & o & o & o & o & o & o & o \\
i & j & k & l & -l & -k & -j & -i \\
m & n & -n & -m & -m & -n & n & m \\
j & -l & -i & -k & k & i & l & -j \\
o & -o & -o & o & o & -o & -o & o \\
k & -i & l & j & -j & -l & i & -k \\
n & -m & m & -n & -n & m & -m & n \\
l & -k & j & -i & i & -j & k & -l
\end{array}\right] \quad (2)$$

$$T_{8d}=\left[\begin{array}{cccccccc}
a & b & c & d & e & f & g & h \\
e & f & g & h & -a & -b & -c & -d \\
c & d & -a & -b & -g & -h & e & f \\
h & g & -f & -e & d & c & -b & -a \\
g & -h & -e & f & c & -d & -a & b \\
b & -a & -d & c & -f & e & h & -g \\
d & -c & b & -a & -h & g & -f & e \\
f & -e & h & -g & b & -a & d & -c
\end{array}\right] \quad (3)$$

T_{8d} is a type of dyadic transform and there is no symmetry in it. So we cannot decompose it to butterfly operations just as we do for T_{8u}. Instead, we decompose T_{8d} to the multiplication of three 8×8 sparse matrices, denoted as M_2, M_3 and M_4, which represent the three stages in the bottom-right module in Fig. 1. There are some considerations for the three matrices. First, they should contain integers only. Secondly, the integers should be very small such that only additions and shiftings are involved and multiplications can be avoided in fast algorithms. Last but not least, the matrices should be sparse, which means that many zeros appear in the matrices.

First, we check whether M_2, M_3 and M_4 probably exist, which means that (4) must be satisfied.

$$T_{8d}=M_2\times M_3\times M_4 \quad (4)$$

Obviously, row vectors in T_{8d} are orthogonal and have the same length, where the length of a row vector a is defined as $a \times a^T$. We denote the length of every row vector of T_{8d} as n_{8d}, and obviously $|\det(T_{8d})|$ is equal to n_{8d}^4, where $\det(\cdot)$ and $|\cdot|$ mean determinant and absolute value, respectively. To simplify the problem, we assume M_2, M_3 and M_4 are also row-orthogonal and in each matrix, the row vectors have the same length. The length of the row vectors from the three matrices are denoted as n_2, n_3 and n_4, respectively. Similarly, $|\det(M_2)|$, $|\det(M_3)|$ and $|\det(M_4)|$ are equal to n_2^4, n_3^4 and n_4^4, respectively. If (4) is satisfied, (5) is necessarily true.

$$|\det(T_{8d})|=n_{8d}^4=|\det(M_2)|\times|\det(M_3)|\times|\det(M_4)|=n_2^4\times n_3^4\times n_4^4 \quad (5)$$

Therefore,

$$n_{8d}= n_2\times n_3\times n_4. \quad (6)$$

So when designing a simplified order-16 ICT with a fast algorithm, we should make sure n_{8d} of T_{8d} is the product of at least 3 prime numbers. Otherwise, (4) has no solution under this assumption.

With the constraint of (6), the lengths of row vectors in M_2, M_3 and M_4 are much smaller than that of row vectors in T_{8d}, which means that the elements in the three matrices have very small magnitudes and may also contain many zeros.

Among the three matrices, M_4 will be found first. Both sides of (4) is post-multiplied by the inverse of M_4, which is equal to (M_4^T/n_4), so (4) changes to (7) as below.

$$T_{8d}\times M_4^T/n_4=M_2\times M_3 \quad (7)$$

It is noticed that M_4^T can be regarded as a set of eight column vectors, and $T_{8d}\times M_4^T/n_4$ contains only integers. So we first establish a set of column vectors by exhaustive search, in which each column vector, b_i, satisfies two conditions. One is that the length of b_i is n_4, and the other is the elements in $(T_{8d}\times b_i/n_4)$ are all integers. If (4) has solutions, we can pick out eight orthogonal column vectors from the set to form M_4^T and thus get M_4.

To find M_3, both sides of (7) is post-multiplied by the inverse of M_3, which is equal to (M_3^T/n_3), so (7) changes to (8) as below.

$$(T_{8d}\times M_4^T/n_4)\times M_3^T/n_3=M_2 \quad (8)$$

M_3 can be obtained by the similar method of searching M_4. Finally, with M_3 and M_4 known, we can easily obtain M_2 by (8).

C. *Decomposition of other matrices*

If T_{8u} and T_{8d} have different structures with (2) and (3), the method of how to decompose T_{8d} to three matrix multiplications, introduced in *B*, is also applicable as long as tow conditions are satisfied. One is that T_{8u} and T_{8d} are row-orthogonal and the other is that the lengths of the row vectors in each matrix are the same.

III. The Fast Algorithm for Simplified Order-16 ICT

In this section, we develop a fast algorithm for the simplified order-16 ICT in [6] by the approach introduced in Section II. For this example, the element set of the transform matrix, {*a*, *b*, *c*, *d*, …, *n*, *o* }, is designed to be {11, 11, 11, 9, 8, 6, 4, 1, 10, 9, 6, 2, 10, 4, 8}, which shows good performance for HD video coding [6]. The corresponding T_{8u} and T_{8d} are shown as (9) and (10).

$$T_{8u}=\begin{bmatrix} 8 & 8 & 8 & 8 & 8 & 8 & 8 & 8 \\ 10 & 9 & 6 & 2 & -2 & -6 & -9 & -10 \\ 10 & 4 & -4 & -10 & -10 & -4 & 4 & 10 \\ 9 & -2 & -10 & -6 & 6 & 10 & 2 & -9 \\ 8 & -8 & -8 & 8 & 8 & -8 & -8 & 8 \\ 6 & -10 & 2 & 9 & -9 & -2 & 10 & -6 \\ 4 & -10 & 10 & -4 & -4 & 10 & -10 & 4 \\ 2 & -6 & 9 & -10 & 10 & -9 & 6 & -2 \end{bmatrix} \quad (9)$$

$$T_{8d}=\begin{bmatrix} 11 & 11 & 11 & 9 & 8 & 6 & 4 & 1 \\ 8 & 6 & 4 & 1 & -11 & -11 & -11 & -9 \\ 11 & 9 & -11 & -11 & -4 & -1 & 8 & 6 \\ 1 & 4 & -6 & -8 & 9 & 11 & -11 & -11 \\ 4 & -1 & -8 & 6 & 11 & -9 & -11 & 11 \\ 11 & -11 & -9 & 11 & -6 & 8 & 1 & -4 \\ 9 & -11 & 11 & -11 & -1 & 4 & -6 & 8 \\ 6 & -8 & 1 & -4 & 11 & -11 & 9 & -11 \end{bmatrix} \quad (10)$$

T_{8u} is exactly the same as the order-8 ICT in AVS [5], so the fast algorithm for that order-8 ICT can be used as the butterfly structure for the upper-right module in Fig. 1. Interested readers may refer to [6] for details.

For T_{8d} shown in (10), the corresponding n_{8d} is 561 and the determinant of T_{8d} is 9.9049×10^{10} equal to 561^4. To show that n_{8d} is a product of three integers as (5), we let n_2, n_3 and n_4 be 3, 11 and 17 with the order changeable. Then the method introduced in Section II.*B* is used to search M_2, M_3 and M_4, and (11) shows one of the solutions of (4). The magnitudes of elements in (11) are so small that the fast algorithm can be implemented by using only additions and shiftings.

IV. Complexity Analysis and Experimental Results

$$T_{8d}=M_2\times M_3\times M_4=\begin{bmatrix} -2 & 0 & 1 & -1 & -1 & 3 & -1 & 0 \\ 3 & -1 & 1 & 1 & 0 & 2 & 0 & 1 \\ -1 & -3 & 1 & 0 & 1 & 0 & 2 & -1 \\ 0 & 1 & 0 & 1 & 3 & 1 & -1 & -2 \\ 1 & -1 & -3 & -2 & 0 & 1 & 0 & -1 \\ 1 & 1 & 1 & 0 & -2 & 0 & 1 & -3 \\ 0 & -2 & 0 & 1 & -1 & -1 & -3 & -1 \\ -1 & 0 & -2 & 3 & -1 & 1 & 1 & 0 \end{bmatrix}\begin{bmatrix} 0 & 0 & 0 & 0 & 1 & -1 & 0 & 1 \\ 0 & 0 & 1 & 1 & 0 & 0 & 1 & 0 \\ 0 & -1 & 0 & -1 & 0 & 0 & 1 & 0 \\ 0 & -1 & 1 & 0 & 0 & 0 & -1 & 0 \\ 0 & -1 & -1 & 1 & 0 & 0 & 0 & 0 \\ -1 & 0 & 0 & 0 & 0 & -1 & 0 & -1 \\ -1 & 0 & 0 & 0 & -1 & 0 & 0 & 1 \\ -1 & 0 & 0 & 0 & 1 & 1 & 0 & 0 \end{bmatrix}\begin{bmatrix} 0 & -2 & 0 & -1 & 0 & 2 & -1 & 1 \\ 0 & -1 & 0 & 2 & 1 & -1 & 0 & 2 \\ 0 & -2 & 1 & 1 & 0 & 0 & 1 & -2 \\ -2 & 0 & -1 & 0 & 2 & 0 & -1 & -1 \\ -1 & 0 & 2 & 0 & -1 & -1 & -2 & 0 \\ -2 & 0 & 1 & -1 & 0 & 0 & 2 & 1 \\ -1 & 1 & 0 & 2 & -1 & 2 & 0 & 0 \\ -1 & -1 & -2 & 0 & -2 & -1 & 0 & 0 \end{bmatrix} \quad (11)$$

If the transform is implemented by matrix multiplication, the computational load will be very heavy, due to many additions and multiplications introduced by dot products. The fast algorithm developed in Section III can implement the transform by only shiftings and additions. Table I compares the numbers of additions, shiftings and multiplications required by transforming a 16-point vector using the fast algorithm and matrix multiplication respectively. Without multiplication, the computational complexity is reduced significantly. However, the number of additions required for the fast algorithm is considerable, because multiplications with small magnitude integers are replaced by combinations of additions and shiftings.

The 2-D simplified order-16 ICT is implemented using both fast algorithm and matrix multiplication by the C language. First, we generate 10 thousand 16×16 random blocks, in which the numbers are uniformly distributed from -256 to 255. Then, the 2-D simplified order-16 ICT with the transform matrix in Section III are applied to these blocks using the fast algorithm and matrix multiplication respectively and the total execution time of the two methods is compared. The results are shown in the third row of Table II, which are recorded by clockticks event, the smallest unit of time recognized by the CPU. So using the fast algorithm can save about 91.4% of the computational time compared with using matrix multiplication.

The efficiency of the fast algorithm for simplified order-16 ICT is compared with that of the fast DCT (FDCT) [8], because there is no published research work related to fast algorithms for order-16 ICT or simplified ICT. We do the experiment as described above to the case of DCT, which means that the time for computing order-16 DCT using FDCT and matrix multiplication is recorded respectively. The second row of Table II shows FDCT can save 93.6% of the computational time compared with using matrix multiplication. The proposed fast algorithm for simplified order-16 ICT is a little less efficient than the FDCT for DCT because of the different dyadic symmetries in T_{8d}.

At the same time, we also pay attention to the efficiency of the fast algorithm used in the bottom-right module in Fig. 1, which is special for the multiplication with matrix T_{8d}. Similarly, 10 thousand 8×8 random blocks are generated, in which the numbers are uniformly distributed from -256 to 255. The forth row of Table II shows the comparison of execution time using different methods and the fast algorithm can save almost 88.9% computational time compared with matrix multiplication. This proves that the fast algorithm for T_{8d} has the comparable efficiency of the whole one for the simplified order-16 ICT.

Therefore, if the structure of T_{8u} is different from (2), which means that some existing fast algorithms of order-8 ICTs cannot be borrowed and the method for decomposing T_{8d} has to be applied to T_{8u}, the fast algorithm will be a little less efficient, but overall, the computational complexity will be reduced significantly. To ensure only additions and shiftings are sufficient for the fast algorithm, magnitudes of the elements in $T_{16\times16}$ should be very small, e.g., represented by 4~5 bits. Otherwise, using only additions and shiftings to implement fast algorithms will not be so efficient as the example in this paper.

V. CONCLUSION

In this paper, a universal approach to developing fast algorithms for simplified order-16 ICT is proposed. The main contribution lies in two parts. First, a general transform matrix for simplified order-16 ICT is proposed, which is suitable for the development of fast algorithms. Secondly, multiplication with the proposed transform matrix is decomposed to butterfly operations by a universal method. Experimental results show that the fast algorithm developed by the proposed approach can save 90% of the computational time compared with matrix multiplication.

TABLE I. COMPLEXITY COMPARISON OF FAST ALGORITHM AND MATRIX MULTIPLICATION

Operation	Fast Simplified ICT	Matrix Multiplication
Addition	150	240
Multiplication	0	256
Shifting	32	0

TABLE II. EXECUTION TIME OF FAST ALGORITHMS AND MATRIX MULTIPLICATION (10^6 CLOCKTICKS EVENT)

	Fast Algorithm	Matrix Multiplication
DCT	112.0	1756.8
Simplified ICT	80.0	934.4
T_{8d}	12.8	115.2

REFERENCES

[1] Cham, W.K., "Development of integer cosine transforms by the principle of dyadic symmetry", Proc. IEE, I, 136,(4), pp. 276-282, 1989.

[2] Cham, W.K., Chan, Y.T., "An order-16 Integer Cosine Transform", IEEE Transactions on Signal Processing, vol. 39, No. 5, pp. 1205-1208, May 1991.

[3] K.T. Lo, W.K. Cham, "Development of simple orthogonal transforms for image compression", IEE Proc. Vis. Image Signal Process., Vol.142, No. 1, Feb. 1995.

[4] Gary J. Sullivan, Pankaj Topiwala, Ajay Luthra, "The H.264/AVC advanced video coding standard: overview and introduction to the fidelity range extensions", SPIE Conference on Applications of Digital Image Processing XXVII, Aug., 2004.

[5] AVS Video Group, "Information technology – Advanced coding of audio and video – Part 2: Video", AVS Doc. AVS-N1317, Sept. 2006.

[6] Jie Dong, King Ngi Ngan, Wai-kuen Cham,"Adaptive Block-size Transforms for AVS X-profile", Audio Video Coding Standard Workgroup of China, AVS M1771: March 2006, Video Proposal.

[7] Steve Gordon, Detlev Marpe, Thomas Wiegand, "Simplified Use of 8x8 Transforms", JVT Doc. JVT-J029, [Online] Available: http://ftp3.itu.ch/jvt-experts/, Hawaii, USA, Dec. 2003.

[8] Loeffler C., Ligtenberg A., Moschytz C.S., "Practical Fast 1D DCT Algorithm with Eleven Multiplications", Proc. ICASSP pp. 988-991, 1989.