Core transform design for high efficiency video coding

Jie Dong and Yan Ye

InterDigital Communications, LLC., 9710 Scranton Road, Suite 250, San Diego, CA 92121

ABSTRACT

High Efficiency Video Coding (HEVC) is the next generation video coding standard currently being developed by the Joint Collaborative Team on Video Coding (JCT-VC). It employs various coding unit sizes $2^K \times 2^K$, where K is a positive integer with the typical values from 3 to 6; it also uses larger transform sizes up to 32×32. This raises the interest in seeking high performance higher order integer transforms with low computation requirements. This paper presents approaches to designing order-N (N=4, 8, 16, 32) integer transforms, by which the derived integer transforms have special symmetry structures to ensure the matrix factorization. The proposed set of high order integer transforms with well selected elements demonstrates excellent coding performance, compared with the core transform design in HEVC.

Keywords: Integer transform, core transform, HEVC, video coding

1. INTRODUCTION

Transform coding is a fundamental part in video compression systems. It de-correlates the samples to be quantized so that the commonly used scalar quantization over individual values does not cause too much coding efficiency loss in comparison with vector quantization. Transform coding also compacts the energy of the samples into a few transform coefficients to facilitate entropy coding.

The performance of a transform is usually evaluated by transform coding gain (T_{CG}) [1], the maximum gain of transform coding over Pulse-code Modulation (PCM) coding. T_{CG} is determined by transform type, transform size, and the statistics of the input signal, e.g., correlation. Regardless of input signal's statistics, larger-size transforms provide higher T_{CG}, and Karhunen-Loeve Transform (KLT) is the optimal transform, of which the basis vectors are derived based on the time- and space-variant second order statistics of the input signal. Given the original video signal (or its residual by predictive coding) as the input, where the correlation is relatively high, discrete cosine transform (DCT) [2] has basis vectors very close to KLT and is considered as a fixed approximation of KLT; the ideal transform size should be the same as the frame size to fully exploit the spatial correlation among pixels.

Usually, the gain of a transform coder is smaller than the T_{CG} of the transform it uses. T_{CG} is achieved, only if the quantization and entropy coding are both optimal, which, however, is impossible for a real transform coder. In order to fully demonstrate the performance that the transform can provide, we should not only pursue high T_{CG} while determining the type and size of the transform, but also consider the practical limitations of the transform coder, including the typical statistics of the input signal, the design of quantization and entropy coding, the framework of the video coding system upon which the transform coder is built, and the complexity restriction due to the mainstream hardware capability. How these limitations influence the transform design is introduced as below.

First, the prevailing video compression systems use block-based hybrid framework, which means each frame is divided into non-overlapped blocks and each block is the unit where the hybrid predictive coding and transform coding apply. To enable coding unit (CU) level rate-distortion (R-D) optimization and avoid encoding/decoding delay, the transform size should not be larger than the CU size. Second, larger transforms generally suffer more from non-optimal entropy coding, as it is much more difficult to accurately estimate the non-zero transform coefficients' positions. However, larger transforms benefit the high and ultra-high definition videos, of which the spatial correlation is very high and homogeneous regions are dominant [3]. Larger size transforms compact the higher correlated signals into fewer transform coefficients distributed in the low frequency domain. The positions of these transform coefficients are easily modeled and efficiently entropy coded. Third, larger transform has higher computational complexity. 2-D order-$2N$ transform requires four times of the arithmetic operations of 2-D order-N transform, whereas the performance is not

improved that much. Last but not the least, integer transforms are preferred to real-valued transforms. Real-valued transform, such as DCT, is of higher computational complexity and causes the encoder and decoder mismatch, even if implemented using high-precision fixed-point approximation. Integer transform, such as Integer Cosine Transform (ICT) [4], provides bit-exact implementation and significant complexity reduction, and, if well designed, achieves almost the same T_{CG} as DCT. In recent years, integer transform superseded DCT, and has been adopted in modern video coding standards, such as H.264/AVC [5], VC-1 [6], and HEVC [7].

The core transform design for high efficiency video coding (HEVC) [7], the emerging video coding standard developed by the Joint Collaborative Team on Video Coding (JCT-VC) of ITU-T/SG16/Q.6/VCEG and ISO/IEC/MPEG, takes all the above factors into consideration. HEVC is focused on coding high definition (HD) and ultra HD video, in which the area representing certain content becomes larger than that in lower resolution video. Therefore, HEVC enlarges the CU size up to $2^K \times 2^K$ (K is a positive integer set in the sequence header) to save the overhead, such as CU type and motion vector (MV), and scales the prediction and transform sizes accordingly. The core transform design adopts 2-D order-16 and order-32 integer transforms mainly for smooth regions and smaller order-4 and order-8 transforms mainly for detail-rich regions.

The trend of using large size transforms raises the interest in seeking high performance high order integer transforms with low computation requirements with the following desirable features. First, to provide the performance as good as the sub-optimal transform DCT, the basis vectors should resemble DCT's basis vectors, including small DCT distortion [8] and orthogonality. Second, the L^2 norms of the basis vectors are (almost) equal, in order to save the additional scaling process for correcting the different norms of basis vectors. Third, the transform matrix has an intrinsic symmetry structure, such that the multiplication with the transform matrix can be fully factorized to enable fast algorithms.

Finding a high order integer transform satisfying all the conditions above is very challenging, and tremendous research effort has been dedicated to this topic [3][9]-[16]. Some work [3][9]-[11] is more focused on good energy compaction capability, i.e., less DCT distortion, and derives the transform matrices by scaling the DCT matrix elements by a factor and then truncating the real numbers to the nearby integers based on certain constraints, such as equal L^2 norms or orthogonality. This approach is applicable for designing an arbitrary order integer transform, but cannot maintain the intrinsic symmetry structure in the DCT matrix that guarantees the fast algorithm. Other work [3][12]-[16] starts from designing the intrinsic symmetry structure for the fast algorithm, which is actually a significant constraint for seeking the basis vectors with high energy compaction capability. For example, with the proposed symmetry structures in [3], [12], and [13], no integer transform solution with small DCT distortion can be found. The work in [16] is so far the closest to the ideal integer transform that satisfies all the desired merits; however the cost is that the magnitudes of the transform matrix elements are very large, i.e., represented by 14 bits, and the forward order-32 transform cannot be realized even by 32-bit integer arithmetic.

In this paper, we propose approaches to derive order-4, order-8, order-16, and order-32 integer transforms, which satisfy all the desired merits except the orthogonality. The symmetry structure of the transform matrix that guarantees the fast algorithm is first developed. With this symmetry structure, the elements in the transform matrices are well selected, such that the basis vectors have almost the same L^2 norms and very small DCT distortion and the transform error introduced by non-orthogonality is negligible. Four specific integer transforms with the sizes 4×4, 8×8, 16×16, and 32×32 are proposed as the core transform design for HEVC. Compared with the core transform adopted in HEVC [11] and other contributions [15][16], the proposed integer transforms have lower complexity and more flexibility for implementation, while providing comparable coding performance.

The remainder of the paper is organized as follows. Section 2 presents the general matrices of the proposed integer transforms. In Section 3, a specific set of integer transforms are designed based on these general matrices, and proposed as the core transform for HEVC. Section 4 reports the experimental results, followed by the conclusion in Section 5.

2. GENERAL MATRICES OF THE PROPOSED INTEGER TRANSFORM

The DCT transform matrix can be fully factorized, which ensures the development of fast algorithm. Many factorization algorithms have been proposed in literature, which exploit different symmetry structures of the DCT matrix. Some algorithms [19] are not very efficient in terms of arithmetic operations, but can be systematically generalized to any high order DCT. Some algorithms require fewer computations, but are only applicable for certain order DCTs. For example,

the fast algorithm proposed in [18] is only for order-8 and order-16 DCTs. In this paper, the proposed integer transforms retain the symmetric structures of DCT, such that they can be factorized in the same way. Like DCT, the proposed 2-D integer transform is separable, which can be accomplished by applying two 1-D integer transforms sequentially. Therefore, in the following, we only discuss the 1-D integer transform design.

2.1 Even-odd part decomposition

Denote an order-*2N* DCT matrix as $\mathbf{T}_{DCT}^{(2N)}$ ($N = 2^K, K$ is a positive integer). The basis vectors of $\mathbf{T}_{DCT}^{(2N)}$ have alternating even and odd symmetries and thus can be decomposed to even and odd parts, where the even part is identical to $\mathbf{T}_{DCT}^{(N)}$ scaled by a factor $\frac{1}{\sqrt{2}}$, as shown in (1).

$$\mathbf{T}_{DCT}^{(2N)} = \begin{bmatrix} \frac{1}{\sqrt{2}}\mathbf{T}_{DCT}^{(N)} & 0 \\ 0 & \mathbf{P}_{DCT}^{(N)} \end{bmatrix} \begin{bmatrix} \mathbf{I}_N & \mathbf{J}_N \\ \mathbf{I}_N & -\mathbf{J}_N \end{bmatrix} \tag{1}$$

In (1), $\mathbf{P}_{DCT}^{(N)}$, an *N*×*N* matrix, is the odd part of $\mathbf{T}_{DCT}^{(2N)}$, $\mathbf{I}_N$ is the order-*N* identity matrix and $\mathbf{J}_N$ is the opposite diagonal identity matrix, obtained by flipping the columns of $\mathbf{I}_N$ horizontally. This property is kept in the integer transform design, as shown in (2),

$$\mathbf{T}_{2N} = \begin{bmatrix} s_N\mathbf{T}_N & 0 \\ 0 & \mathbf{P}_N \end{bmatrix} \begin{bmatrix} \mathbf{I}_N & \mathbf{J}_N \\ \mathbf{I}_N & -\mathbf{J}_N \end{bmatrix} \tag{2}$$

where $\mathbf{T}_{2N}$ is the transform matrix of the order-*2N* integer transform and its even and odd parts are denoted as $s_N\mathbf{T}_N$ and $\mathbf{P}_N$, respectively. To construct a larger integer transform in such a recursive process, $\mathbf{T}_2$, as the initial matrix, is defined as in (3). Note that the scaling factor s_N is used to ensure the L^2 norms of the basis vectors in $\mathbf{T}_{2N}$ are (almost) equal.

$$\mathbf{T}_2 = \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \tag{3}$$

Denote $\mathbf{n} = [n_0, n_1, \dots, n_{2N-1}]$ as the input signal to the order-*2N* transform and $\mathbf{u} = [u_0, u_1, \dots, u_{2N-1}]$ as the output. The vector comprising the even points of $\mathbf{u}$, denoted as $\mathbf{u}_e = [u_0, u_2, \dots, u_{2N-2}]$, is determined only by the even part $s_N\mathbf{T}_N$; likewise, the vector comprising the odd points, $\mathbf{u}_o = [u_1, u_3, \dots, u_{2N-1}]$, is determined by the odd part.

As the even part of the transform matrix is constructed in such a recursive way, the problem of integer transform design narrows down to designing the odd part. In order-4, order-8, order-16, and order-32 DCT matrices, the odd parts are represented by (4) to (7), respectively, in which the values of the matrices' elements are shown in Table 1.

$$\mathbf{P}_{DCT}^{(2)} = \begin{bmatrix} A_2 & B_2 \\ B_2 & -A_2 \end{bmatrix} \tag{4}$$

$$\mathbf{P}_{DCT}^{(4)} = \begin{bmatrix} A_4 & B_4 & C_4 & D_4 \\ B_4 & -D_4 & -A_4 & -C_4 \\ C_4 & -A_4 & D_4 & B_4 \\ D_4 & -C_4 & B_4 & -A_4 \end{bmatrix} \tag{5}$$

$$\mathbf{P}_{DCT}^{(8)} = \begin{bmatrix} A_8 & B_8 & C_8 & D_8 & E_8 & F_8 & G_8 & H_8 \\ B_8 & E_8 & H_8 & -F_8 & -C_8 & -A_8 & -D_8 & -G_8 \\ C_8 & H_8 & -D_8 & -B_8 & -G_8 & E_8 & A_8 & F_8 \\ D_8 & -F_8 & -B_8 & H_8 & A_8 & G_8 & -C_8 & -E_8 \\ E_8 & -C_8 & -G_8 & A_8 & -H_8 & -B_8 & F_8 & D_8 \\ F_8 & -A_8 & E_8 & G_8 & -B_8 & D_8 & H_8 & -C_8 \\ G_8 & -D_8 & A_8 & -C_8 & F_8 & H_8 & -E_8 & B_8 \\ H_8 & -G_8 & F_8 & -E_8 & D_8 & -C_8 & B_8 & -A_8 \end{bmatrix} \tag{6}$$

$$\mathbf{P}_{DCT}^{(16)} = \begin{bmatrix} A_{16} & B_{16} & C_{16} & D_{16} & E_{16} & F_{16} & G_{16} & H_{16} & I_{16} & J_{16} & K_{16} & L_{16} & M_{16} & N_{16} & O_{16} & P_{16} \\ B_{16} & E_{16} & H_{16} & K_{16} & N_{16} & -P_{16} & -M_{16} & -J_{16} & -G_{16} & -D_{16} & -A_{16} & -C_{16} & -F_{16} & -I_{16} & -L_{16} & -O_{16} \\ C_{16} & H_{16} & M_{16} & -O_{16} & -J_{16} & -E_{16} & -A_{16} & -F_{16} & -K_{16} & -P_{16} & L_{16} & G_{16} & B_{16} & D_{16} & I_{16} & N_{16} \\ D_{16} & K_{16} & -O_{16} & -H_{16} & -A_{16} & -G_{16} & -N_{16} & L_{16} & E_{16} & C_{16} & J_{16} & -P_{16} & -I_{16} & -B_{16} & -F_{16} & -M_{16} \\ E_{16} & N_{16} & -J_{16} & -A_{16} & -I_{16} & O_{16} & F_{16} & D_{16} & M_{16} & -K_{16} & -B_{16} & -H_{16} & P_{16} & G_{16} & C_{16} & L_{16} \\ F_{16} & -P_{16} & -E_{16} & -G_{16} & O_{16} & D_{16} & H_{16} & -N_{16} & -C_{16} & -I_{16} & M_{16} & B_{16} & J_{16} & -L_{16} & -A_{16} & -K_{16} \\ G_{16} & -M_{16} & -A_{16} & -N_{16} & F_{16} & H_{16} & -L_{16} & -B_{16} & -O_{16} & E_{16} & I_{16} & -K_{16} & -C_{16} & -P_{16} & D_{16} & J_{16} \\ H_{16} & -J_{16} & -F_{16} & L_{16} & D_{16} & -N_{16} & -B_{16} & P_{16} & A_{16} & O_{16} & -C_{16} & -M_{16} & E_{16} & K_{16} & -G_{16} & -I_{16} \\ I_{16} & -G_{16} & -K_{16} & E_{16} & M_{16} & -C_{16} & -O_{16} & A_{16} & -P_{16} & -B_{16} & N_{16} & D_{16} & -L_{16} & -F_{16} & J_{16} & H_{16} \\ J_{16} & -D_{16} & -P_{16} & C_{16} & -K_{16} & -I_{16} & E_{16} & O_{16} & -B_{16} & L_{16} & H_{16} & -F_{16} & -N_{16} & A_{16} & -M_{16} & -G_{16} \\ K_{16} & -A_{16} & L_{16} & J_{16} & -B_{16} & M_{16} & I_{16} & -C_{16} & N_{16} & H_{16} & -D_{16} & O_{16} & G_{16} & -E_{16} & P_{16} & F_{16} \\ L_{16} & -C_{16} & G_{16} & -P_{16} & -H_{16} & B_{16} & -K_{16} & -M_{16} & D_{16} & -F_{16} & O_{16} & I_{16} & -A_{16} & J_{16} & N_{16} & -E_{16} \\ M_{16} & -F_{16} & B_{16} & -I_{16} & P_{16} & J_{16} & -C_{16} & E_{16} & -L_{16} & -N_{16} & G_{16} & -A_{16} & H_{16} & -O_{16} & -K_{16} & D_{16} \\ N_{16} & -I_{16} & D_{16} & -B_{16} & G_{16} & -L_{16} & -P_{16} & K_{16} & -F_{16} & A_{16} & -E_{16} & J_{16} & -O_{16} & -M_{16} & H_{16} & -C_{16} \\ O_{16} & -L_{16} & I_{16} & -F_{16} & C_{16} & -A_{16} & D_{16} & -G_{16} & J_{16} & -M_{16} & P_{16} & N_{16} & -K_{16} & H_{16} & -E_{16} & B_{16} \\ P_{16} & -O_{16} & N_{16} & -M_{16} & L_{16} & -K_{16} & J_{16} & -I_{16} & H_{16} & -G_{16} & F_{16} & -E_{16} & D_{16} & -C_{16} & B_{16} & -A_{16} \end{bmatrix} \quad (7)$$

Table 1. Values of the elements in order-4, order-8, order-16, and order-32 DCT matrices

$[A_2, B_2]$	$\left[\frac{1}{\sqrt{2}}\cos(\frac{\pi}{8}) \quad \frac{1}{\sqrt{2}}\cos(\frac{3\pi}{8})\right]$
$[A_4, B_4, C_4, D_4]$	$\left[\frac{1}{2}\cos(\frac{\pi}{16}) \quad \frac{1}{2}\cos(\frac{3\pi}{16}) \quad \frac{1}{2}\cos(\frac{5\pi}{16}) \quad \frac{1}{2}\cos(\frac{7\pi}{16})\right]$
$[A_8, B_8, C_8, \dots, G_8, H_8]$	$\left[\frac{1}{\sqrt{8}}\cos(\frac{\pi}{32}) \quad \frac{1}{\sqrt{8}}\cos(\frac{3\pi}{32}) \quad \frac{1}{\sqrt{8}}\cos(\frac{5\pi}{32}) \quad \frac{1}{\sqrt{8}}\cos(\frac{7\pi}{32}) \quad \dots\dots \quad \frac{1}{\sqrt{8}}\cos(\frac{13\pi}{32}) \quad \frac{1}{\sqrt{8}}\cos(\frac{15\pi}{32})\right]$
$[A_{16}, B_{16}, C_{16}, \dots, O_{16}, P_{16}]$	$\left[\frac{1}{4}\cos(\frac{\pi}{64}) \quad \frac{1}{4}\cos(\frac{3\pi}{64}) \quad \frac{1}{4}\cos(\frac{5\pi}{64}) \quad \frac{1}{4}\cos(\frac{7\pi}{64}) \quad \dots\dots \quad \frac{1}{4}\cos(\frac{29\pi}{64}) \quad \frac{1}{4}\cos(\frac{31\pi}{64})\right]$

In integer transform design, the general matrix of the odd part is borrowed from DCT with the matrix elements' relative magnitudes unchanged. A simple way to obtain the odd part of an integer transform is to scale the elements in the same order DCT matrix by a factor, and truncate the scaled real numbers to the nearby integers based on certain constraints. However, this approach cannot maintain the intrinsic symmetry structure in the DCT matrix that guarantees the fast algorithm. In Sections 2.2, 2.3, and 2.4, we propose approaches to deriving the odd parts of integer transforms from order-4 up to order-32, where DCT matrix's intrinsic symmetry structure is incorporated. Note that, the odd part of order-4 integer transform does not need further factorization, since its size is 2×2, as show in (4).

2.2 Odd part of order-8 integer transform

The odd part of order-8 DCT, as shown in (5), can be re-written as in (8), only if the elements $[A_4, B_4, C_4, D_4]$ are defined as in the second row of Table 1 [18]. Otherwise, e.g., $[A_4, B_4, C_4, D_4]$ are replaced by integers with the relative magnitudes unchanged, the simplest way to derive the matrix for integer transforms, (5) will not be equal to (8). The reason is obvious because (5) becomes an integer matrix, whereas the first and fourth rows of (8) still have real-valued elements.

$$\mathbf{P}_{DCT}^{(4)} = \begin{bmatrix} \frac{1}{\sqrt{2}}(B_4 + C_4) & \frac{1}{\sqrt{2}}(A_4 + D_4) & \frac{1}{\sqrt{2}}(A_4 - D_4) & \frac{1}{\sqrt{2}}(B_4 - C_4) \\ B_4 & -D_4 & -A_4 & -C_4 \\ C_4 & -A_4 & D_4 & B_4 \\ \frac{1}{\sqrt{2}}(B_4 - C_4) & -\frac{1}{\sqrt{2}}(A_4 - D_4) & \frac{1}{\sqrt{2}}(A_4 + D_4) & -\frac{1}{\sqrt{2}}(B_4 + C_4) \end{bmatrix} \quad (8)$$

In order to keep the factorizable structure in (8) and provide more freedom in searching good matrix elements that compose of DCT-like basis vectors, we generalize (8), and propose the odd part of order-8 integer transform and its factorized form in (9) and (10), respectively,

$$\mathbf{P}_4 = \begin{bmatrix} h_4 e_4 (b_4 + c_4) & h_4 f_4 (a_4 + d_4) & h_4 f_4 (a_4 - d_4) & h_4 e_4 (b_4 - c_4) \\ k_4 j_4 b_4 & -k_4 i_4 d_4 & -k_4 i_4 a_4 & -k_4 j_4 c_4 \\ k_4 j_4 c_4 & -k_4 i_4 a_4 & k_4 i_4 d_4 & k_4 j_4 b_4 \\ h_4 e_4 (b_4 - c_4) & -h_4 f_4 (a_4 - d_4) & h_4 f_4 (a_4 + d_4) & -h_4 e_4 (b_4 + c_4) \end{bmatrix} \quad (9)$$

$$\mathbf{P}_4 = \begin{bmatrix} h_4 & 0 & 0 & h_4 \\ 0 & 0 & k_4 & 0 \\ 0 & k_4 & 0 & 0 \\ h_4 & 0 & 0 & -h_4 \end{bmatrix} \times \begin{bmatrix} e_4 & 0 & f_4 & 0 \\ 0 & -i_4 & 0 & j_4 \\ j_4 & 0 & -i_4 & 0 \\ 0 & f_4 & 0 & e_4 \end{bmatrix} \times \begin{bmatrix} a_4 & 0 & 0 & -b_4 \\ 0 & c_4 & -d_4 & 0 \\ 0 & d_4 & c_4 & 0 \\ b_4 & 0 & 0 & a_4 \end{bmatrix} \tag{10}$$

where the elements $[a_4, b_4, c_4, \ldots, h_4, k_4]$ are all integers. To make $\mathbf{P}_4$ resemble $\mathbf{P}_{DCT}^{(4)}$ in terms of the relative magnitudes among matrix elements, the following three constraints should be imposed.

Relation 1

1. $(b_4 + c_4)/(b_4 - c_4) \approx a_4/d_4 \approx A_4/D_4$ and $(a_4 + d_4)/(a_4 - d_4) \approx b_4/c_4 \approx B_4/C_4$
2. $(i_4 a_4)/(j_4 b_4) \approx (e_4(b_4 + c_4))/(f_4(a_4 + d_4)) \approx A_4/B_4$
3. $h_4 f_4(a_4 + d_4) \approx k_4 j_4 b_4$ and $h_4 e_4(b_4 + c_4) \approx k_4 i_4 a_4$

2.3 Odd part of order-16 integer transform

The derivation of the odd part of order-16 integer transform is similar to that for the order-8 one. Based on [18], the odd part of order-16 DCT shown in (6) is re-written as in (11).

$$\mathbf{P}_{DCT}^{(8)} = \begin{bmatrix} \frac{1}{\sqrt{2}}(D_8 + E_8) & \frac{1}{\sqrt{2}}(C_8 + F_8) & \frac{1}{\sqrt{2}}(B_8 + G_8) & \frac{1}{\sqrt{2}}(A_8 + H_8) & \frac{1}{\sqrt{2}}(A_8 - H_8) & \frac{1}{\sqrt{2}}(B_8 - G_8) & \frac{1}{\sqrt{2}}(C_8 - F_8) & \frac{1}{\sqrt{2}}(D_8 - E_8) \\ E_8 I_8 + D_8 J_8 & -F_8 I_8 + C_8 J_8 & B_8 I_8 - G_8 J_8 & -A_8 I_8 - H_8 J_8 & H_8 I_8 - A_8 J_8 & -G_8 I_8 - B_8 J_8 & -C_8 I_8 - F_8 J_8 & D_8 I_8 - E_8 J_8 \\ D_8 I_8 + E_8 J_8 & -C_8 I_8 + F_8 J_8 & G_8 I_8 - B_8 J_8 & -H_8 I_8 - A_8 J_8 & -A_8 I_8 + H_8 J_8 & B_8 I_8 + G_8 J_8 & F_8 I_8 + C_8 J_8 & -E_8 I_8 + D_8 J_8 \\ D_8 & -F_8 & -B_8 & H_8 & A_8 & G_8 & -C_8 & -E_8 \\ E_8 & -C_8 & -G_8 & A_8 & -H_8 & -B_8 & F_8 & D_8 \\ -E_8 I_8 + D_8 J_8 & -F_8 I_8 - C_8 J_8 & B_8 I_8 + G_8 J_8 & A_8 I_8 - H_8 J_8 & -H_8 I_8 - A_8 J_8 & -G_8 I_8 + B_8 J_8 & -C_8 I_8 + F_8 J_8 & -D_8 I_8 - E_8 J_8 \\ -D_8 I_8 + E_8 J_8 & -C_8 I_8 - F_8 J_8 & G_8 I_8 + B_8 J_8 & H_8 I_8 - A_8 J_8 & A_8 I_8 + H_8 J_8 & B_8 I_8 - G_8 J_8 & F_8 I_8 - C_8 J_8 & E_8 I_8 + D_8 J_8 \\ \frac{1}{\sqrt{2}}(D_8 - E_8) & -\frac{1}{\sqrt{2}}(C_8 - F_8) & \frac{1}{\sqrt{2}}(B_8 - G_8) & -\frac{1}{\sqrt{2}}(A_8 - H_8) & \frac{1}{\sqrt{2}}(A_8 + H_8) & -\frac{1}{\sqrt{2}}(B_8 + G_8) & \frac{1}{\sqrt{2}}(C_8 + F_8) & -\frac{1}{\sqrt{2}}(D_8 + E_8) \end{bmatrix} \tag{11}$$

Extending (11) to a more general matrix that is also applicable for integer elements, we propose the odd part of order-16 integer transform and its factorized form in (12) and (13), respectively, where $[a_4, b_4, c_4, \ldots, k_4, l_4]$ are all integers.

$$\mathbf{P}_8 = \begin{bmatrix} l_8(d_8 + e_8) & l_8(c_8 + f_8) & l_8(b_8 + g_8) & l_8(a_8 + h_8) & l_8(a_8 - h_8) & l_8(b_8 - g_8) & l_8(c_8 - f_8) & l_8(d_8 - e_8) \\ e_8 i_8 + d_8 j_8 & -f_8 i_8 + c_8 j_8 & b_8 i_8 - g_8 j_8 & -a_8 i_8 - h_8 j_8 & h_8 i_8 - a_8 j_8 & -g_8 i_8 - b_8 j_8 & -c_8 i_8 - f_8 j_8 & d_8 i_8 - e_8 j_8 \\ d_8 i_8 + e_8 j_8 & -c_8 i_8 + f_8 j_8 & g_8 i_8 - b_8 j_8 & -h_8 i_8 - a_8 j_8 & -a_8 i_8 + h_8 j_8 & b_8 i_8 + g_8 j_8 & f_8 i_8 + c_8 j_8 & -e_8 i_8 + d_8 j_8 \\ k_8 d_8 & -k_8 f_8 & -k_8 b_8 & k_8 h_8 & k_8 a_8 & k_8 g_8 & -k_8 c_8 & -k_8 e_8 \\ k_8 e_8 & -k_8 c_8 & -k_8 g_8 & k_8 a_8 & -k_8 h_8 & -k_8 b_8 & k_8 f_8 & k_8 d_8 \\ -e_8 i_8 + d_8 j_8 & -f_8 i_8 - c_8 j_8 & b_8 i_8 + g_8 j_8 & a_8 i_8 - h_8 j_8 & -h_8 i_8 - a_8 j_8 & -g_8 i_8 + b_8 j_8 & -c_8 i_8 + f_8 j_8 & -d_8 i_8 - e_8 j_8 \\ -d_8 i_8 + e_8 j_8 & -c_8 i_8 - f_8 j_8 & g_8 i_8 + b_8 j_8 & h_8 i_8 - a_8 j_8 & a_8 i_8 + h_8 j_8 & b_8 i_8 - g_8 j_8 & f_8 i_8 - c_8 j_8 & e_8 i_8 + d_8 j_8 \\ l_8(d_8 - e_8) & -l_8(c_8 - f_8) & l_8(b_8 - g_8) & -l_8(a_8 - h_8) & l_8(a_8 + h_8) & -l_8(b_8 + g_8) & l_8(c_8 + f_8) & -l_8(d_8 + e_8) \end{bmatrix} \tag{12}$$

$\mathbf{P}_8 =$

$$\begin{bmatrix} 0 & 0 & 0 & l_8 & -l_8 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & -j_8 & i_8 \\ i_8 & j_8 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & k_8 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & k_8 & 0 & 0 \\ j_8 & -i_8 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & i_8 & j_8 \\ 0 & 0 & 0 & l_8 & l_8 & 0 & 0 & 0 \end{bmatrix} \times \begin{bmatrix} -1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & -1 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & -1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & -1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \times \begin{bmatrix} -1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & -1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & -1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & -1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 \end{bmatrix} \times \begin{bmatrix} d_8 & 0 & 0 & 0 & 0 & 0 & 0 & -e_8 \\ 0 & f_8 & 0 & 0 & 0 & 0 & c_8 & 0 \\ 0 & 0 & b_8 & 0 & 0 & -g_8 & 0 & 0 \\ 0 & 0 & 0 & h_8 & a_8 & 0 & 0 & 0 \\ 0 & 0 & 0 & -a_8 & h_8 & 0 & 0 & 0 \\ 0 & 0 & g_8 & 0 & 0 & b_8 & 0 & 0 \\ 0 & -c_8 & 0 & 0 & 0 & 0 & f_8 & 0 \\ e_8 & 0 & 0 & 0 & 0 & 0 & 0 & d_8 \end{bmatrix} \tag{13}$$

The following two conditions need to be satisfied to ensure the relative magnitudes among the matrix elements remain similar to those in a DCT matrix.

Relation 2

1. $a_8 : b_8 : c_8 : d_8 : e_8 : f_8 : g_8 : h_8 \approx A_8 : B_8 : C_8 : D_8 : E_8 : F_8 : G_8 : H_8$
2. $i_8 : j_8 : k_8 : l_8 \approx \cos\left(\frac{3\pi}{8}\right) : \sin\left(-\frac{3\pi}{8}\right) : 1 : \frac{1}{\sqrt{2}}$

2.4 Odd part of order-32 integer transform

The derivation of the odd part of order-32 integer transform and its factorization is inspired by Chen's DCT fast algorithm [19], as the LLM fast algorithm [18] used in Sections 2.2 and 2.3 are only for order-8 and order-16 DCT. Chen's algorithm is not the most efficient in terms of arithmetic operations, but provides a systematic approach to developing the fast algorithm for order-2^K DCT, where K can be arbitrarily large. In [19], the odd part of an order-N DCT is factorized into $(2\log_2 N - 3)$ matrices with the size 16×16 and each matrix can be calculated by a close-form expression. Hence, the odd part of order-32 DCT is factorized to seven matrices, as expressed by (14).

$$\mathbf{P}_{DCT}^{(16)} = \mathbf{R}_7 \times ... \times \mathbf{R}_3 \times \mathbf{R}_2 \times \mathbf{R}_1 \tag{14}$$

In practice, transforming a 1-D 16-point signal by (14) is realized by seven serial stages with each stage accomplishing the function of one matrix multiplication, and therefore, the latency is very high.

To reduce the latency, we propose to factorize the odd part $\mathbf{P}_{16}$ of an order-32 integer transform into five integer matrices, as shown in (15),

$$\mathbf{P}_{16} = \mathbf{W} \times \mathbf{M}_4 \times \mathbf{M}_3 \times \mathbf{M}_2 \times \mathbf{M}_1 \tag{15}$$

where $\mathbf{M}_4 \times \mathbf{M}_3 \times \mathbf{M}_2 \times \mathbf{M}_1$ is shown in (16).

$$\mathbf{M}_4 \times \mathbf{M}_3 \times \mathbf{M}_2 \times \mathbf{M}_1 = \begin{bmatrix} 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & -1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & -1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & -1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & -1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 \end{bmatrix} \times \begin{bmatrix} y_0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & y_0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & -y_2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & y_1 & 0 & 0 \\ 0 & 0 & 0 & -y_2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & y_1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & -y_1 & 0 & 0 & 0 & 0 & 0 & 0 & -y_2 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & -y_1 & 0 & 0 & 0 & 0 & -y_2 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & y_0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & y_0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & y_0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & y_0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & y_0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -y_2 & 0 & y_1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -y_2 & 0 & 0 & 0 & y_1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & y_1 & 0 & 0 & 0 & 0 & 0 & y_2 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & y_1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & y_2 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & y_0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & y_0 \end{bmatrix} \times \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & -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 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \times \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & x_0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & x_0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & x_0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & x_0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & x_1 & 0 & 0 & 0 & 0 & 0 & 0 & -x_1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & x_1 & 0 & 0 & 0 & 0 & -x_1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & x_1 & 0 & 0 & -x_1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & x_1 & -x_1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & x_1 & x_1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & x_1 & 0 & 0 & x_1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & x_1 & 0 & 0 & 0 & 0 & x_1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & x_1 & 0 & 0 & 0 & 0 & 0 & 0 & x_1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & x_0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & x_0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & x_0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ x_0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \tag{16}$$

In (16), $\mathbf{M}_2$ and $\mathbf{M}_4$ are exactly the same as $\mathbf{R}_2$ and $\mathbf{R}_4$ defined in [19], since they are binary matrices; $\mathbf{M}_1$ and $\mathbf{M}_3$ keep the same sign distribution and relative magnitudes among elements as in $\mathbf{R}_1$ and $\mathbf{R}_3$, respectively, as long as the following two conditions are satisfied.

Relation 3

1. $x_0 : x_1 \approx \sqrt{2} : 1$
2. $y_0 : y_1 : y_2 \approx 1 : \sin\left(\frac{\pi}{8}\right) : \cos\left(\frac{\pi}{8}\right)$

W can be expressed as the product of three integer matrices, **X**, **Y**, and **Z**, as shown in (17), and (18)-(24) shows the definitions of the three matrices.

$$\mathbf{W} = \mathbf{Z} \times \mathbf{X} \times \mathbf{Y} \tag{17}$$

$$\mathbf{X} = \begin{bmatrix} \mathbf{X}_1 & 0 & 0 & 0 \\ 0 & \mathbf{X}_2 & 0 & 0 \\ 0 & 0 & \mathbf{X}_3 & 0 \\ 0 & 0 & 0 & \mathbf{X}_4 \end{bmatrix} \tag{18}$$

$$\mathbf{X}_1 = \begin{bmatrix} o_{16} & a_{16}z_0 - o_{16}z_1 & o_{16}z_0 + a_{16}z_1 & a_{16} \\ h_{16} & -i_{16}z_0 + h_{16}z_1 & -h_{16}z_0 - i_{16}z_1 & i_{16} \\ -i_{16} & -h_{16}z_0 - i_{16}z_1 & i_{16}z_0 - h_{16}z_1 & h_{16} \\ -a_{16} & o_{16}z_0 + a_{16}z_1 & -a_{16}z_0 + o_{16}z_1 & o_{16} \end{bmatrix} = \begin{bmatrix} p_0 & p_1 & p_2 & p_3 \\ p_4 & p_5 & -p_6 & p_7 \\ -p_7 & -p_6 & -p_5 & p_4 \\ -p_3 & p_2 & -p_1 & p_0 \end{bmatrix} \tag{19}$$

$$\mathbf{X}_2 = \begin{bmatrix} l_{16}z_0 + e_{16}z_1 & l_{16} & e_{16} & -e_{16}z_0 + l_{16}z_1 \\ -d_{16}z_0 - m_{16}z_1 & d_{16} & m_{16} & m_{16}z_0 - d_{16}z_1 \\ m_{16}z_0 - d_{16}z_1 & -m_{16} & d_{16} & d_{16}z_0 + m_{16}z_1 \\ -e_{16}z_0 + l_{16}z_1 & -e_{16} & l_{16} & -l_{16}z_0 - e_{16}z_1 \end{bmatrix} = \begin{bmatrix} q_0 & q_1 & q_2 & q_3 \\ -q_4 & q_5 & q_6 & -q_7 \\ -q_7 & -q_6 & q_5 & q_4 \\ q_3 & -q_2 & q_1 & -q_0 \end{bmatrix} \tag{20}$$

$$\mathbf{X}_3 = \begin{bmatrix} n_{16} & -n_{16}z_2 + c_{16}z_3 & c_{16}z_2 + n_{16}z_3 & c_{16} \\ f_{16} & f_{16}z_2 - k_{16}z_3 & -k_{16}z_2 - f_{16}z_3 & k_{16} \\ -k_{16} & -k_{16}z_2 - f_{16}z_3 & -f_{16}z_2 + k_{16}z_3 & f_{16} \\ -c_{16} & c_{16}z_2 + n_{16}z_3 & n_{16}z_2 - c_{16}z_3 & n_{16} \end{bmatrix} = \begin{bmatrix} r_0 & r_1 & r_2 & r_3 \\ r_4 & r_5 & -r_6 & r_7 \\ -r_7 & -r_6 & -r_5 & r_4 \\ -r_3 & r_2 & -r_1 & r_0 \end{bmatrix} \tag{21}$$

$$\mathbf{X}_4 = \begin{bmatrix} g_{16}z_2 + j_{16}z_3 & j_{16} & g_{16} & j_{16}z_2 - g_{16}z_3 \\ -p_{16}z_2 - b_{16}z_3 & b_{16} & p_{16} & -b_{16}z_2 + p_{16}z_3 \\ -b_{16}z_2 + p_{16}z_3 & -p_{16} & b_{16} & p_{16}z_2 + b_{16}z_3 \\ j_{16}z_2 - g_{16}z_3 & -g_{16} & j_{16} & -g_{16}z_2 - j_{16}z_3 \end{bmatrix} = \begin{bmatrix} v_0 & v_1 & v_2 & -v_3 \\ -v_4 & v_5 & v_6 & -v_7 \\ -v_7 & -v_6 & v_5 & v_4 \\ -v_3 & -v_2 & v_1 & -v_0 \end{bmatrix} \tag{22}$$

$$\mathbf{Y} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \tag{23}$$

$$\mathbf{Z} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \tag{24}$$

Y and **Z** are both binary matrices with only one non-zero element in each row/column, meaning they are used for re-ordering the 1-D 16-point data being processed in the serial stages (15). **X** is the only effective matrix in (17), and multiplying with **X** can be realized by multiplying with four 4×4 integer matrices, which in total reduces the arithmetic operations to one fourth. Actually, **X** can be further factorized into three sparser matrices, such that $\mathbf{P}_{16}$ is fully factorized to seven matrices as for $\mathbf{P}_{DCT}^{(16)}$ in (14). By doing this, the computation requirement is further reduced, but the latency becomes unacceptable for practical implementation. Therefore, factorizing $\mathbf{P}_{16}$ by (15) is proposed to balance these two aspects. The intermediate numbers $[a_{16}, b_{16}, c_{16}, \dots, o_{16}, p_{16}, z_0, z_1, z_2, z_3]$, as in (19) to (22), should satisfy the following two relations, in order to keep the relative magnitudes of DCT unchanged.

1. $a_{16} : b_{16} : c_{16} : \dots : o_{16} : p_{16} \approx A_{16} : B_{16} : C_{16} : \dots : O_{16} : P_{16}$
2. $z_0 : z_1 : z_2 : z_3 \approx \cos\left(\frac{7\pi}{16}\right) : \cos\left(\frac{\pi}{16}\right) : \cos\left(\frac{5\pi}{16}\right) : \cos\left(\frac{3\pi}{16}\right)$

Consequently, the actual integer elements $[p_0, p_1, p_2, p_3, p_4, p_5, p_6, p_7]$, $[q_0, q_1, q_2, q_3, q_4, q_5, q_6, q_7]$, $[r_0, r_1, r_2, r_3, r_4, r_5, r_6, r_7]$, and $[v_0, v_1, v_2, v_3, v_4, v_5, v_6, v_7]$ in $\mathbf{X}_1$, $\mathbf{X}_2$, $\mathbf{X}_3$, and $\mathbf{X}_4$, respectively, have the following four relations.

Relation 4

1. $p_0 : p_1 : p_2 : p_3 : p_4 : p_5 : p_6 : p_7 \approx o_{16} : (a_{16} z_0 - o_{16} z_1) : (o_{16} z_0 + a_{16} z_1) : a_{16} : h_{16} : (-i_{16} z_0 + h_{16} z_1) : (h_{16} z_0 - i_{16} z_1) : i_{16}$
2. $q_0 : q_1 : q_2 : q_3 : q_4 : q_5 : q_6 : q_7 \approx (l_{16} z_0 + e_{16} z_1) : l_{16} : e_{16} : (-e_{16} z_0 + l_{16} z_1) : (d_{16} z_0 + m_{16} z_1) : d_{16} : m_{16} : (-m_{16} z_0 + d_{16} z_1)$
3. $r_0 : r_1 : r_2 : r_3 : r_4 : r_5 : r_6 : r_7 \approx n_{16} : (-n_{16} z_2 + c_{16} z_3) : (c_{16} z_2 + n_{16} z_3) : c_{16} : f_{16} : (f_{16} z_2 - k_{16} z_3) : (k_{16} z_2 + f_{16} z_3) : k_{16}$
4. $v_0 : v_1 : v_2 : v_3 : v_4 : v_5 : v_6 : v_7 \approx (g_{16} z_2 + j_{16} z_3) : j_{16} : g_{16} : (-j_{16} z_2 + g_{16} z_3) : (p_{16} z_2 + b_{16} z_3) : b_{16} : p_{16} : (b_{16} z_2 - p_{16} z_3)$

3. PROPOSED CORE TRANSFORM DESIGN FOR HEVC

The proposed core transform design for HEVC includes a set of order-4, order-8, order-16, and order-32 integer transforms. They use the factorizable general transform matrices as presented in Section 2, and thus the fast algorithms can be easily developed. This section introduces how to determine the values of the variables used to represent the general matrices.

To ensure that the energy compaction capability of the integer transform is comparable to DCT, the relative magnitudes among the transform matrices' elements should approximate those of DCTs, as specified by Relations 1 to 4 in Sections 2.2 to 2.4. We first scale the real-valued numbers on the right side of the Relations by certain factors, and then use the scaled values as the starting points to search the integers in a certain neighborhood. The best set of the integers for the left side of the Relations achieves the balance of three important aspects: the variance of the L^2 norms of the basis vectors, the deviation from DCT, and the transform error introduced by non-orthogonality.

Given the transform matrix of an order-N integer transform, denoted as $\mathbf{T}_N$, the transformation process is shown in (25), where $\mathbf{X}_N$ and $\mathbf{Y}_N$ represent the input and output $N \times N$ signals, respectively.

$$\mathbf{Y}_N = \mathbf{T}_N \mathbf{X}_N \mathbf{T}_N^T \tag{25}$$

The L^2 norm of the ith basis vector of $\mathbf{T}_N$ is calculated by (26), and the L^2 norms of all the basis vectors are denoted as a vector $\boldsymbol{l} = [l_0, l_1, \ldots, l_{N-1}]^T$.

$$l_i = \sqrt{\sum_{j=0}^{N-1} (\mathbf{T}_N(i,j))^2} \tag{26}$$

For energy conservation, $\mathbf{Y}_N$ should be normalized by element-wise dividing by a scaling matrix $\mathbf{L}_N$ ($\mathbf{L}_N = \boldsymbol{l}\boldsymbol{l}^T$), which is an additional process in comparison of DCT. However, when the L^2 norms are very close or even equal to each other, $\mathbf{L}_N$ becomes a constant matrix, and the element-wise matrix division can be approximated by a scalar matrix division, which can be combined with the following scalar quantization. Hence, making the L^2 norms almost equal can save the normalization step and is a highly desired feature in integer transform design.

Deviation from DCT is measured by DCT distortion [8], calculated by (27),

$$d = 1 - \frac{1}{N} \left\| \mathrm{diag}(\mathbf{T}_{DCT}^{(N)} \mathbf{t}_N^T) \right\|_2^2 \tag{27}$$

where diag(**X**) denotes the main diagonal of matrix **X,** and $\mathrm{t_N}$ is the normalized transform matrix, obtained by (28).

$$\mathbf{t}_N(i,j) = \frac{\mathbf{T}_N(i,j)}{l_i} \tag{28}$$

This criterion reflects the signal energy that leaks out of DCT subbands. A smaller value of DCT distortion means less deviation from DCT and thus implies less performance degradation compared with DCT.

Orthogonality is a strong condition in designing integer transforms. The magnitudes of the elements in the transform matrix have to be very large especially for high order transforms to fulfill orthogonality, and become even larger if other constrains are imposed at the same time. If orthogonality is not fulfilled, the average energy of the reconstruction error in the spatial domain is larger than that of the quantization error in the transform domain, and the difference is known as transform error, denoted as σ_e^2. In this paper, orthogonality is not one of the design constraints, but the elements of the transform matrices are well selected, such that the transform error σ_e^2 is negligible. According to [3], the transform error of an order-*N* integer transform can be calculated by (29),

$$\sigma_e^2 = \frac{1}{N} \sigma_x^2 \sum_{i=0}^{N-1} \sum_{j=0}^{N-1} |\mathbf{M}(i,j)| \tag{29}$$

where σ_x^2 is the variance of the input source, and **M** is the error matrix, defined by (30).

$$\mathbf{M} = (\mathbf{t}_N^T \mathbf{t}_N - \mathbf{I}_N)^{\mathbf{T}} (\mathbf{t}_N^T \mathbf{t}_N - \mathbf{I}_N) \tag{30}$$

We conducted brute-force search with all the aforementioned aspects taken into account, and the best solution, as shown in Table 2, was proposed as the core transform design for HEVC [17]. The specific odd parts, obtained by substituting the parameter values in Table 2 into the general matrices introduced in Section 2, are shown in (31) to (34). Fig. 1 shows the data flow of the fast algorithm for the forward order-32 integer transform. Since the order-*N* integer transform is used as the even part of the order-*2N* one, as introduced in Section 2.1, its fast algorithm is also shown as part of the fast algorithm of order-*2N* transform in the same recursive manner.

Table 2. Parameter values for the general transform matrices proposed as the core transform design for HEVC

Order-4	a_2, b_2	167	70								
Order-8	$a_4, b_4, c_4, d_4, e_4, f_4, i_4, j_4, h_4, k_4$	3	2	5	1	144	99	72	99	1	2
Order-16	$a_8, b_8, c_8, d_8, e_8, f_8, g_8, h_8$	60	57	53	46	39	28	18	5		
	i_8, j_8, k_8, l_8	10	-22	24	17						
Order-32	x_0, x_1	7	5								
	y_0, y_1, y_2	13	5	12							
	$p_0, p_1, p_2, p_3, p_4, p_5, p_6, p_7$	1	5	31	32	23	19	26	21		
	$q_0, q_1, q_2, q_3, q_4, q_5, q_6, q_7$	31	13	28	8	16	30	11	27		
	$r_0, r_1, r_2, r_3, r_4, r_5, r_6, r_7$	8	21	23	31	27	2	32	16		

	$v_0, v_1, v_2, v_3, v_4, v_5, v_6, v_7$	30	19	25	10	29	31	5	13
s_N	s_2, s_4, s_8, s_{16}	128	2	4	2				

$$\mathbf{P}_2 = \begin{bmatrix} 167 & 70 \\ 70 & -167 \end{bmatrix} \tag{31}$$

$$\mathbf{P}_4 = \begin{bmatrix} 360 & 297 & 198 & 72 \\ 297 & -72 & -360 & -198 \\ 198 & -360 & 72 & 297 \\ 72 & -198 & 297 & -360 \end{bmatrix} \tag{32}$$

$$\mathbf{P}_8 = \begin{bmatrix} 1445 & 1377 & 1275 & 1105 & 935 & 663 & 425 & 119 \\ 1402 & 886 & 174 & -710 & -1270 & -1434 & -1146 & -398 \\ 1318 & 86 & -1074 & -1370 & -490 & 966 & 1446 & 622 \\ 1104 & -672 & -1368 & 120 & 1440 & 432 & -1272 & -936 \\ 936 & -1272 & -432 & 1440 & -120 & -1368 & 672 & 1104 \\ 622 & -1446 & 966 & 490 & -1370 & 1074 & 86 & -1318 \\ 398 & -1146 & 1434 & -1270 & 710 & 174 & -886 & 1402 \\ 119 & -425 & 663 & -935 & 1105 & -1275 & 1377 & -1445 \end{bmatrix} \tag{33}$$

$$\mathbf{P}_{16} = \begin{bmatrix} 2912 & 2821 & 2779 & 2723 & 2685 & 2460 & 2340 & 2145 & 2015 & 1690 & 1510 & 1205 & 1036 & 665 & 455 & 91 \\ 2821 & 2639 & 2107 & 1505 & 660 & -90 & -1040 & -1690 & -2340 & -2730 & -2920 & -2810 & -2429 & -1981 & -1183 & -455 \\ 2821 & 2093 & 959 & -413 & -1765 & -2590 & -2860 & -2535 & -1495 & -130 & 1220 & 2355 & 2884 & 2667 & 1911 & 728 \\ 2730 & 1456 & -399 & -2135 & -2935 & -2305 & -715 & 1235 & 2665 & 2795 & 1735 & -115 & -1974 & -2828 & -2457 & -1001 \\ 2548 & 728 & -1757 & -2807 & -1925 & 405 & 2535 & 2665 & 975 & -1495 & -2915 & -2085 & 112 & 2324 & 2821 & 1183 \\ 2457 & -182 & -2618 & -2289 & 415 & 2790 & 2210 & -715 & -2795 & -1950 & 950 & 2855 & 1708 & -1288 & -2912 & -1456 \\ 2275 & -910 & -2870 & -721 & 2490 & 2200 & -1300 & -2860 & -390 & 2600 & 1900 & -1460 & -2765 & -210 & 2730 & 1729 \\ 2093 & -1729 & -2506 & 1197 & 2690 & -705 & -2925 & 130 & 2860 & 455 & -2875 & -980 & 2569 & 1519 & -2366 & -1911 \\ 1911 & -2366 & -1519 & 2569 & 980 & -2875 & -455 & 2860 & -130 & -2925 & 705 & 2690 & -1197 & -2506 & 1729 & 2093 \\ 1729 & -2730 & -210 & 2765 & -1460 & -1900 & 2600 & 390 & -2860 & 1300 & 2200 & -2490 & -721 & 2870 & -910 & -2275 \\ 1456 & -2912 & 1288 & 1708 & -2855 & 950 & 1950 & -2795 & 715 & 2210 & -2790 & 415 & 2289 & -2618 & 182 & 2457 \\ 1183 & -2821 & 2324 & -112 & -2085 & 2915 & -1495 & -975 & 2665 & -2535 & 405 & 1925 & -2807 & 1757 & 728 & -2548 \\ 1001 & -2457 & 2828 & -1974 & 115 & 1735 & -2795 & 2665 & -1235 & -715 & 2305 & -2935 & 2135 & -399 & -1456 & 2730 \\ 728 & -1911 & 2667 & -2884 & 2355 & -1220 & -130 & 1495 & -2535 & 2860 & -2590 & 1765 & -413 & -959 & 2093 & -2821 \\ 455 & -1183 & 1981 & -2429 & 2810 & -2920 & 2730 & -2340 & 1690 & -1040 & 90 & 660 & -1505 & 2107 & -2639 & 2821 \\ 91 & -455 & 665 & -1036 & 1205 & -1510 & 1690 & -2015 & 2145 & -2340 & 2460 & -2685 & 2723 & -2779 & 2821 & -2912 \end{bmatrix} \tag{34}$$

4. EXPERIMENTAL RESULTS

The proposed core transform design, as presented in Section 3, was integrated into HEVC's test model HM4.0 [20], using 16-bit integer arithmetic, and tested under the common test conditions [21] and the special test conditions defined by the core experiment group 10 (CE10) [22]. The common test conditions are mainly focused on HD and ultra HD video sequences; the mandatory part includes five classes of video sequences in terms of resolutions, i.e., Class A 2560×1600, Class B 1080*p*, Class C 832×480, Class D 416×240, and Class E 720*p*. Each sequence is coded by three types of coding structures: all intra (AI), random access (RA), and low delay (LD), and two configurations of coding tools: high efficiency (HE) and low complexity (LOCO). With certain coding structure and coding tool configuration, a sequence is coded at four *QP*s (22, 27, 32, and 37) to generate an operational R-D curve over a wide range of bit-rate, and the average bit-rate reduction compared with HEVC, known as BD-rate [23], is calculated and used to measure the R-D performance of a proposed coding tool. The special test conditions defined by CE10 require testing the R-D performance at low *QP*s (1, 5, 9, and 13) and high *QP*s (36, 42, 47, and 51) to see if there is unusual behavior at these uncommon test points.

With the test condition LD and HE, the BD-rate of the proposed core transform design together with the BD-rates of other contributions that were seriously considered by JCT-VC is shown in Table 3. For the performance under other test conditions, interested readers are referred to [11], [15], [16], and [17] for detailed data. As can be seen, the performance among all the core transform designs for HEVC is less that 1% BD bit-rate difference, no matter what the *QP* is. Similar results can be observed under other test conditions. Therefore, the features that the existing core transform designs have and the number of arithmetic operations they require, which are compared in Table 4 and Table 5, respectively, become more important in evaluation.

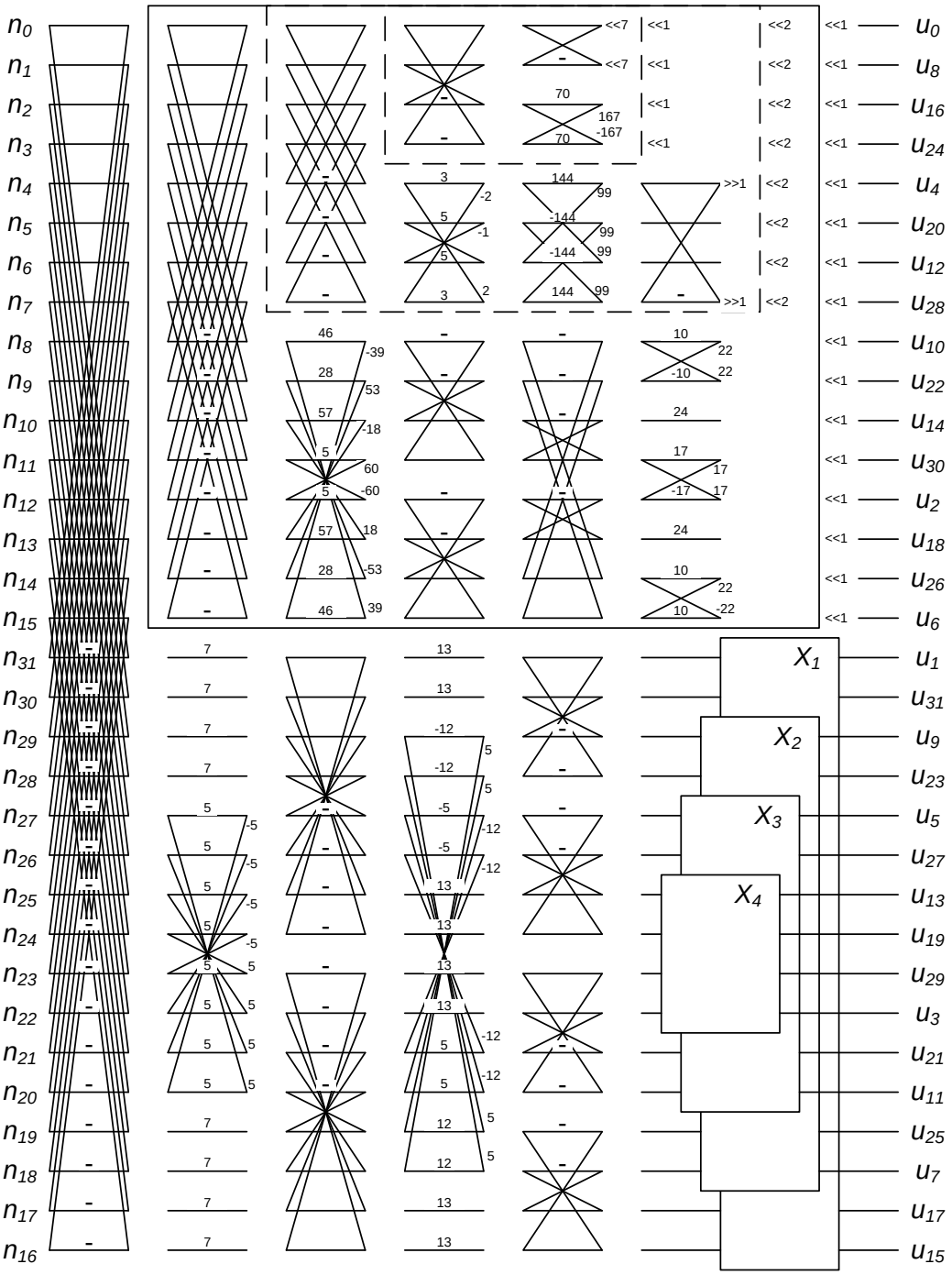

Fig. 1 Fast algorithm for the forward order-32 integer transform with the fast algorithms for the forward order-4, order-8, and order-16 integer transforms embedded

As shown in Table 4, all the core transform designs have small DCT distortion, and therefore their performances all approach the DCT's, as evidenced in Table 3. The core transform design adopted in HEVC cannot be fully factorized, which means it does not have a fast algorithm, and thus the required arithmetic operations are much higher than other designs especially for high order transforms (see Table 5). In the other designs, Joshi's [15] still has higher complexity, because its basis vectors have quite different L^2 norms, which need to be corrected by an additional scaling process. Furthermore, the core transform design in [15] is presented as a fast algorithm, which does not have an equivalent

realization of matrix multiplication. This limits the flexibility of implementation. Alshina's design [16] has all the desired features. However, the symmetry structure proposed in [16] is not as efficient as the structure proposed in this paper. As a result, it needs many more multiplications for high order transforms and the magnitudes of the elements in the transform matrices are larger than those in the proposed design. No matter what the transform size is, the elements are represented by 14 bits, and the forward order-32 transform cannot even be implemented using 32-bit integer arithmetic. The core transform design proposed in this paper achieves a good balance of R-D performance and computation requirement, and allows more flexibility for implementation.

Table 3 Comparisons of the existing core transform designs in BD-rate reduction (%)

Class	Sequence	Normal *QP*			Low *QP*			High *QP*		
		Joshi's	Alshina's	Proposed	Joshi's	Alshina's	Proposed	Joshi's	Alshina's	Proposed
B	*Kimono*	0	0	0.1	0	0.2	0.2	0.3	0.3	0.1
	ParkScene	0	0.1	0.1	0.1	0.1	0.1	0	0.4	-0.1
	Cactus	0	0	0	0.1	0.1	0.1	-0.1	-0.1	0
	BasketballDrive	0	0	0.1	0.3	0	0.1	0	-0.1	0
	BQTerrace	0	0	0.1	0.2	0	0.1	-0.2	-0.3	-0.5
C	*BasketballDrill*	-0.1	0.3	0.2	0.2	0.1	0	0.3	0.1	0.2
	BQMall	0	-0.1	0	0.2	0	0.1	-0.1	-0.4	0.1
	PartyScene	0	0	0.1	0.2	0	0.1	-0.2	0	-0.2
	RaceHorses	0	0	0	0.1	0.1	0.1	-0.1	-0.1	-0.3
D	*BasketballPass*	0	0	0.1	0.3	0	0	1.3	1.4	-0.2
	BQSquare	-0.1	0	0	0.2	0	0	-0.2	-0.3	0.3
	BlowingBubbles	-0.1	0	0	0.2	0	0.1	0.1	0.4	-0.6
	RaceHorses	0	0	0.1	0.2	0.1	0	0.1	0.1	0.2
E	*Vidyo1*	0.2	-0.1	0	0.4	0.1	0	1.9	0.6	-0.1
	Vidyo3	0.1	0.1	0.1	0.4	0	0.1	0.8	-0.1	-0.8
	Vidyo4	0	0.2	0.3	0.4	0.1	0	-0.4	0	-0.6
Average		0	0.03	0.08	0.22	0.06	0.07	0.22	0.19	-0.16

Table 4 Comparison of the features offered by each existing core transform design

	Small DCT distortion	Full factorization	No scaling process	Negligible transform error	Flexible implementation
HEVC core transform [7]	x		x	x	x
Joshi's [15]	x	x		x	
Alshina's [16]	x	x	x	x	x
Proposed	x	x	x	x	x

Table 5 Numbers of additions and multiplications for *N*-point 1-D transform ($N = 4$, 8, 16, and 32)

		HEVC core transform	Joshi's	Alshina's	Proposed
4-point	Addition	8	9	9	9
	Multiplication	6	3	3	3
8-point	Addition	28	26	29	31
	Multiplication	22	12	11	11
16-point	Addition	100	72	81	93
	Multiplication	86	36	31	21
32-point	Addition	372	186	229	279
	Multiplication	342	92	87	56

5. CONCLUSION

This paper presents a set of integer transforms with the sizes 4×4, 8×8, 16×16, and 32×32. They have high energy

compaction capability and almost equal L^2 norms of the basis vectors, and are factorizable, which guarantees the development of fast algorithm. Compared with the prior work, the proposed integer transforms have lower complexity and more flexibility for implementation while providing the comparable coding efficiency.

REFERENCES

[1] N. S. Jayant and P. Noll, *Digital Coding of Waveforms: Principles and Applications to Speech and Video*. Englewood Cliffs, NJ: Prentice-Hall, 1984.

[2] N. Ahmed, T. Natarajan, K.R. Rao, "Discrete cosine transfom," *IEEE Transactions on Computers*, vol. 23, no. 1, pp. 90-93, Jan. 1974.

[3] J. Dong, K. N. Ngan, C. K. Fong, and W. K. Cham, "2D order-16 integer transforms for HD video coding," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 19, no. 10, pp. 1463–1474, Oct. 2009.

[4] W. K. Cham, "Development of integer cosine transforms by the principle of dyadic symmetry," *IEE Proc., Part I*, vol. 136, no. 4, pp. 276–282, Aug. 1989.

[5] *Advanced Video Coding for Generic Audiovisual Services*, ITU-T Rec. H.264 and ISO/IEC 14496-10 AVC, Mar. 2005.

[6] S. Srinivasan *et al*., "Windows Media Video 9: overview and applications," *Signal Process.: Image Commun.*, pp. 851-875, 2004.

[7] B. Bross, W.-J. Han, G. J. Sullivan, J.-R. Ohm, and T. Wiegand, "High efficiency video coding (HEVC) text specification draft 7", document JCTVC-I1003, Joint Collaborative Team on Video Coding, May 2012.

[8] M. Wien and S. Sun, "ICT comparison for adaptive block transforms," document VCEG-L12, ITU-T SG16/Q6, Jan. 2001.

[9] W. K. Cham and Y. T. Chan, "An order-16 integer cosine transform," *IEEE Trans. Signal Process.*, vol. 39, no. 5, pp. 1205-1208, May 1991.

[10] P. Chen, Y. Ye, and M. Karczewicz, "Video coding using extended block sizes," document T09-SG16-C-0123, ITU-T Q.6/SG16, Geneva, Switzerland, Jan. 2009.

[11] A. Fuldseth, G. Bjøntegaard, and M. Budagavi, "CE10: core transform design for HEVC," document JCTVC-G495, Joint Collaborative Team on Video Coding, Nov. 2011.

[12] S. Ma and C.-C. Kuo, "High-definition video coding with supermacroblocks," in *Proc. SPIE Vis. Commun. Image Process.*, vol. 6508. Jan. 2007, pp. 650816-1–650816-12.

[13] C. K. Fong and W. K. Cham, "Simple order-16 integer transform for video coding," in *Proc. IEEE Int. Conf. Image Process.*, pp. 161-164, Sept. 2010.

[14] C. K. Fong and W. K. Cham, "LLM integer transform and its fast algorithm," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 22, no. 6, pp. 844-854, Jun. 2012.

[15] R. Joshi, J. Sole, and M. Karczewicz, "CE10: Scaled integer transforms supporting recursive factorization structure," document JCTVC-G579, Joint Collaborative Team on Video Coding, Nov. 2011.

[16] E. Alshina *et al*., "CE10: full factorization core transforms for HEVC," document JCTVC-G737, Joint Collaborative Team on Video Coding, Nov. 2011.

[17] J. Dong and Y. Ye, "Non-CE10: core transform design for HEVC", document JCTVC-G272, Joint Collaborative Team on Video Coding, Nov. 2011.

[18] C. Loeffler, A. Ligtenberg, and G. S. Moschytz, "Practical fast 1-D DCT algorithms with 11 multiplications," in *Proc. IEEE Int. Conf. Acoust. Speech Signal Process.*, vol. 2. May 1989, pp. 988–991.

[19] W. H. Chen, C. H. Smith, and S. C. Fralick, "A fast computational algorithm for the discrete cosine transform", *IEEE Trans. Commun.*, vol. 25, no. 9, pp.1004-1009, Sept. 1977.

[20] https://hevc.hhi.fraunhofer.de/svn/svn_HEVCSoftware/tags/HM-4.0/.

[21] F. Bossen, "Common test conditions and software reference configurations", document JCTVC-F900, Joint Collaborative Team on Video Coding, Jul. 2011.

[22] P. Topiwala, M. Budagavi, R. Joshi, A. Fuldseth, I. Kim, "CE10: Core Transform Design," document JCTVC-F910, Joint Collaborative Team on Video Coding, Jul. 2011.

[23] G. Bjontegaard, "Calculation of average PSNR differences between RD-curves," document VCEG-M33, ITU-T SG16/Q6, Apr. 2001.