A new approach to compatible adaptive block-size transforms

Jie Dong, Jian Lou, Ci-Xun Zhang and Lu Yu

Institute of Information and Communication Engineering
Zhejiang University, Hangzhou, P. R. China, 310027

ABSTRACT

Adaptive Block-size Transforms (ABT) has been widely used in image/video coding, since it exploits the maximum feasible signal length for transform coding. However, if the transforms in an ABT coding system are Integer Cosine Transforms (ICT), not only separate transform units but also different scaling matrices are required, which consume a vast amount of resources in practical implementations. In this paper, a new approach to compatible ABT is presented, by which 8×8, 8×4, 4×8 and 4×4 transforms can be processed in one transform unit. Furthermore, with Pre-scaled Integer Transform (PIT), the compatibility of scaling matrices especially for 8×4 and 4×8 ICT can be achieved and a single scaling matrix is required. Simulation results and analysis reveal that this approach greatly saves hardware resources and makes the implementation of ABT much easier without loss of performance.

Keywords: ABT, ICT, PIT, Compatible ABT

1. INTRODUCTION

Integer Cosine Transform (ICT) was first introduced by W.K. Cham in 1989 1 and it has been further developed in recent years. ICT has almost the same compression efficiency as Discrete Cosine Transform (DCT) and it can be implemented using simple integer arithmetic without mismatch among decoders. Due to these advantages, ICT has become the hotspot for transform coding in currently developed video coding standards, including the international standard H.264 and the emerging national Audio Video coding Standard (AVS) of China.

On the other hand, Adaptive Block-size Transform (ABT) has been widely used in image/video coding for many years. Large transforms provide a better energy compaction and a better preservation of detail in a quantized signal than a small transform does. However, larger transforms introduce more ringing artifacts caused by quantization. By choosing the transform size according to the signal properties, the tradeoff between energy compaction and preserved detail on one hand and ringing artifacts on the other can be optimized.

In many existing ABT systems, ICT are used. These ICT-based ABT are adopted by recent international and industrial video coding standards to provide better performance than previous standards. In the Fidelity Range Extensions (FRExt) of H.264, both 8×8 and 4×4 ICT are employed 2. In Microsoft's Windows Media Video 9 (WMV-9), 4 block-size ICT are employed, including 8×8, 8×4, 4×8 and 4×4 3. Also the ABT with block-size 8×8, 8×4, 4×8 and 4×4 was once adopted by H.264, but finally it was taken out due to the high complexity it caused 4. The basis vectors of ICT kernel contain only integers and are not normalized, so besides transform, a scaling process with a scaling matrix is necessary. So, as for ICT-based ABT, not only separate transform units but also different scaling matrices are required, which consume a vast amount of resources in practical implementations.

In this paper, we propose a new approach to compatible ABT, including choosing compatible ICT kernels so that different ICT with block-size 8×8, 8×4, 4×8 and 4×4 can be performed using one transform unit. This approach also introduces Pre-scaled Integer Transform (PIT) 5, which enables the compatibility of scaling matrices especially for 8×4 and 4×8 ICT and reduces the complexity on the decoder side.

The remainder of the paper is organized as follows. Section 2 presents the basic theory of ICT briefly. Section 3 describes the rule of choosing a set of various block-size ICT kernels so that compatibility of transforms can be

This research is supported by NSFC under contract No. 60333020.

Visual Communications and Image Processing 2005, edited by Shipeng Li, Fernando Pereira, Heung-Yeung Shum, Andrew G. Tescher, Proc. of SPIE Vol. 5960 (SPIE, Bellingham, WA, 2005) · 0277-786X/05/$15 · doi: 10.1117/12.631408

achieved. Section 4 describes how the compatible scaling matrix is generated and how it is used. The novel scheme of PIT is also introduced in this section. An example of compatible ABT with block-size from 8×8 down to 4×4 is given in Section 5. Section 6 reports the experimental results and analysis, followed by conclusions in Section 7.

2. THEORY OF INTEGER COSINE TRANSFORM

Suppose the inputs of the forward DCT are n×m video signals or predicted residuals, the forward DCT for n×m block signals can be defined as

$$F_{n\times m} = T_n \times f_{n\times m} \times T_m^T, \tag{1}$$

where $f_{n\times m}$ and $F_{n\times m}$ stand for the signals in the spatial and transform domain respectively. T_n and T_m are the normalized orthogonal transform matrices which contain float coefficients. ICT originates from DCT in order to simplify DCT and enable bit-exact implementations. Its transform matrix, known as ICT kernel, is generated from DCT matrix with the principle of dyadic symmetry and contains only integers. Since an ICT kernel is not normalized, a scaling process with a scaling matrix is required so as to ensure the conservation of energy and the reversibility.

2.1 The ICT kernel

An integer matrix, which can be referred here as an ICT kernel, must retain the same dyadic symmetry in the DCT matrix. To satisfy this condition, an n×n ICT kernel, ICT_n, should contain at most n different integers and certain coefficient t_{ij} can be calculated by (2), where t_{ij} represents the coefficient in the i^{th} row and the j^{th} column and n_k is t_{k0}, $0 \le k \le n\text{-}1$.

$$\begin{cases} t_{ij} = n_k & \text{if } \cos(\frac{2j+1}{2n} i\pi) = \cos(\frac{1}{2n} k\pi) \\ & 0 \le i \le n-1, 1 \le j \le n-1 \\ t_{ij} = -n_k & \text{if } \cos(\frac{2j+1}{2n} i\pi) = -\cos(\frac{1}{2n} k\pi) \\ & 0 \le i \le n-1, 1 \le j \le n-1 \end{cases} \tag{2}$$

Accordingly, the 8×8 ICT kernel can be denoted as

$$ICT_8 = \begin{bmatrix} n_0 & n_0 & n_0 & n_0 & n_0 & n_0 & n_0 & n_0 \\ n_1 & n_3 & n_5 & n_7 & -n_7 & -n_5 & -n_3 & -n_1 \\ n_2 & n_6 & -n_6 & -n_2 & -n_2 & -n_6 & n_6 & n_2 \\ n_3 & -n_7 & -n_1 & -n_5 & n_5 & n_1 & n_7 & -n_3 \\ n_4 & -n_4 & -n_4 & n_4 & n_4 & -n_4 & -n_4 & n_4 \\ n_5 & -n_1 & n_7 & n_3 & -n_3 & -n_7 & n_1 & -n_5 \\ n_6 & -n_2 & n_2 & -n_6 & -n_6 & n_2 & -n_2 & n_6 \\ n_7 & -n_5 & n_3 & -n_1 & n_1 & -n_3 & n_5 & -n_7 \end{bmatrix} \tag{3}$$

To guarantee the orthogonality, (4) is also an indispensable rule for ICT_8.

$$n_1 \times n_3 = n_1 \times n_5 + n_3 \times n_7 + n_5 \times n_7 \tag{4}$$

Besides, the coefficients in an ICT kernel should be selected carefully. Firstly, to obtain a close decorrelating ability to DCT, the row vectors of ICT_n should resemble those of DCT. Taking 8×8 ICT kernel as an example, if n_7 and n_6 are unities in the 2nd and 3rd row vector respectively, the values of n_1, n_3, n_5 and n_2 should approximate to 5.072, 4.2620, 2.8478 and 2.4142 respectively. On the other hand, considering the computational complexity, the magnitude of these integers should be very small, which restricts the first aspect. So the tradeoff between complexity and performance should be carefully made according to different applications.

The 4×4 ICT can be easily deduced from (2) with n = 4.

$$\mathrm{ICT}_4 = \begin{bmatrix} m_0 & m_0 & m_0 & m_0 \\ m_1 & m_3 & -m_3 & -m_1 \\ m_2 & -m_2 & -m_2 & m_2 \\ m_3 & -m_1 & m_1 & -m_3 \end{bmatrix} \tag{5}$$

The orthogonality is automatically satisfied in (5), since the 4×4 DCT is a dyadic matrix in nature.

2.2 The scaling process

The ICT kernel should be used with the cooperation of scaling process, since it is not normalized with those integer coefficients. For forward ICT, (1) should be changed to (6),

$$\mathrm{F}_{n\times m} = (\mathrm{ICT}_n \times \mathrm{f}_{n\times m} \times \mathrm{ICT}_m^{\mathrm{T}}) // \mathrm{R}_{n\times m} \tag{6}$$

where

$$\mathrm{R}_{n\times m} = \mathrm{R}_{n\times 1} \times \mathrm{R}_{m\times 1}^{\mathrm{T}} \tag{7}$$

Symbol // indicates that each element of the left matrix is divided by the element at the same position of the right one. $\mathrm{R}_{n\times 1}$ is an n×1 matrix, whose i^{th} component, $\mathrm{R}_{n\times 1}(i)$, is the norm of the i^{th} row vector in ICT_n and can be calculated by (8).

$$\mathrm{R}_{n\times 1}(i) = \sqrt{\sum_{j=0}^{n} \mathrm{ICT}_n(i,j)^2} \quad 0 \le i \le n \tag{8}$$

The divisions in (6) is usually approximated by integer multiplication and shifting 6 as shown in (9), and $\mathrm{R}_{n\times m}$ is replaced by scaling matrix $\mathrm{S}_{n\times m}$,

$$\mathrm{F}_{n\times m} = (\mathrm{ICT}_n \times \mathrm{f}_{n\times m} \times \mathrm{ICT}_m^{\mathrm{T}}) \otimes \mathrm{S}_{n\times m} >> N \tag{9}$$

where $\mathrm{S}_{n\times m}$ contains only integers and

$$\mathrm{S}_{n\times m}(i,j) \times \mathrm{R}_{n\times m}(i,j) \approx 2^N . \tag{10}$$

The symbol $\otimes$ means that each element of the left matrix is multiplied by the element at the same position of the right one.

After the scaling process, quantization is applied. That is

$$\mathrm{QF}_{n\times m}(QP) = \mathrm{F}_{n\times m} // \mathrm{Q}_{n\times m}(QP) \tag{11}$$

where the matrix $\mathrm{Q}_{n\times m}(QP)$ determines the quantization degree with a Quantization Parameter (QP).

The scaling process is usually combined with the quantization process to reduce computational complexity, so (6) and (11) can be combined to (12),

$$\mathrm{QF}_{n\times m}(QP) = (\mathrm{ICT}_n \times \mathrm{f}_{n\times m} \times \mathrm{ICT}_{\mathrm{m}}^{\mathrm{T}}) // \mathrm{QR}_{n\times m}(QP) \tag{12}$$

where

$$\mathrm{QR}_{n\times m}(QP) = \mathrm{R}_{n\times m} \otimes \mathrm{Q}_{n\times m}(QP) \tag{13}$$

Similarly, the divisions in (12) can also be implemented by integer multiplication and shifting as (14), just as (9) dose. Here $\mathrm{QR}_{n\times m}(QP)$ is replace by quantization matrix $\mathrm{A}_{n\times m}(QP)$, which contains only integers.

$$\mathrm{QF}_{n\times m}(QP) = (\mathrm{ICT}_n \times \mathrm{f}_{n\times m} \times \mathrm{ICT}_{\mathrm{m}}^{\mathrm{T}}) \otimes \mathrm{A}_{n\times m}(QP) >> M \tag{14}$$

The relationship of $\mathrm{A}_{n\times m}(QP)$, $\mathrm{R}_{n\times m}$ and $\mathrm{Q}_{n\times m}(QP)$ should satisfy

$$\mathrm{A}_{n\times m}(QP,i,j) \times \mathrm{R}_{n\times m} \times \mathrm{Q}_{n\times m}(QP,i,j) \approx 2^M . \tag{15}$$

The inverse ICT is calculated by

$$\mathrm{f}'_{n\times m} = \mathrm{ICT}_n^{\mathrm{T}} \times (\mathrm{F}_{n\times m} // \mathrm{R}_{n\times m}) \times \mathrm{ICT}_m , \tag{16}$$

where $\mathrm{f}'_{n\times m}$ is the reconstructed signal in the spatial domain. Like forward ICT, the scaling process is usually implemented by (17).

$$\mathrm{f}'_{n\times m} = \left(\mathrm{ICT}_n^{\mathrm{T}} \times (\mathrm{F}_{n\times m} \otimes \mathrm{S}_{n\times m}) \times \mathrm{ICT}_m\right) >> N \tag{17}$$

When the scaling process is combined with the dequantization process, (18) is implemented,

$$\mathrm{f}'_{n\times m} = (\mathrm{ICT}_n^{\mathrm{T}} \times (\mathrm{QF}_{n\times m}(QP) \otimes \mathrm{B}_{n\times m}(QP)) \times \mathrm{ICT}_m) >> L \tag{18}$$

where $\mathrm{B}_{n\times m}(QP)$ is the dequantization matrix, satisfying

$$\mathrm{B}_{n\times m}(QP, i, j) \times \mathrm{R}_{n\times m} / \mathrm{Q}_{n\times m}(QP, i, j) \approx 2^L . \tag{19}$$

3. COMPATIBLE TRANSFORM

Like DCT, ICT can be separated into two 1-D transforms and can further be decomposed into butterfly operations due to the dyadic symmetry. As to the ICT with block-size 8×8, 8×4, 4×8 and 4×4, two butterfly structures are necessary, which are used to process 8-point and 4-point 1-D transform respectively. Suppose the 8×8 ICT kernel, ICT_8, is given by (3). To help design butterfly structures and show the dyadic symmetry, two modules, defined as T_{4u} and T_{4d}, can always be extracted from ICT_8 7, and how these two modules work in a butterfly structure is shown in Figure 1.

$$\mathrm{T}_{4u} = \begin{bmatrix} \mathrm{n}_0 & \mathrm{n}_0 & \mathrm{n}_0 & \mathrm{n}_0 \\ \mathrm{n}_2 & \mathrm{n}_6 & -\mathrm{n}_6 & -\mathrm{n}_2 \\ \mathrm{n}_4 & -\mathrm{n}_4 & -\mathrm{n}_4 & \mathrm{n}_4 \\ \mathrm{n}_6 & -\mathrm{n}_2 & \mathrm{n}_2 & -\mathrm{n}_6 \end{bmatrix} \tag{20}$$

$$\mathrm{T}_{4d} = \begin{bmatrix} \mathrm{n}_1 & \mathrm{n}_3 & \mathrm{n}_5 & \mathrm{n}_7 \\ \mathrm{n}_3 & \mathrm{n}_7 & -\mathrm{n}_1 & -\mathrm{n}_5 \\ \mathrm{n}_5 & -\mathrm{n}_1 & -\mathrm{n}_7 & \mathrm{n}_3 \\ \mathrm{n}_7 & -\mathrm{n}_5 & \mathrm{n}_3 & -\mathrm{n}_1 \end{bmatrix} \tag{21}$$

It is not difficult to find that the dyadic symmetry in T_{4u} is the same as that in a 4×4 ICT kernel represented by (5). So T_{4u} belonging to an 8×8 ICT kernel can be used as a 4×4 ICT kernel. As long as such T_{4u} is chosen as the 4×4 ICT kernel, ICT_4, for the ABT scheme, the compatibility of butterfly structure can be achieved.

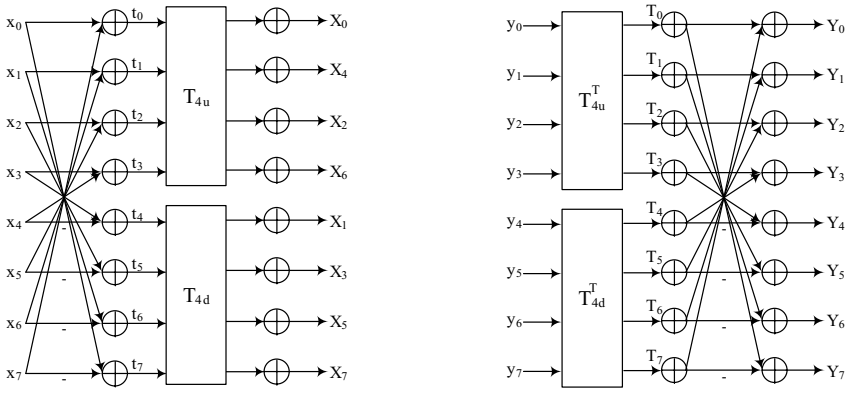

Figure 1. The 8×8 forward and inverse transform butterfly structures

Figure 1 illustrates the 8x8 forward and inverse transform butterfly structures respectively. The 4×4 ICT module T_{4u} and T_{4u}^T can be used for 8×8 ICT.

4. COMPATIBLE SCALING

4.1 The relationship of various block-size scaling matrices

If the T_{4u} mentioned above is chosen as the ICT_4 in the compatible ABT scheme, it is obvious that the relationship between $R_{4\times1}$ and $R_{8\times1}$ can be expressed by

$$R_{4\times1}(i) = R_{8\times1}(2i)/\sqrt{2}\,. \tag{22}$$

According to (7), the relationship of $R_{4\times4}$, $R_{4\times8}$, $R_{8\times4}$ and $R_{8\times8}$ can be deduced as below.

$$R_{4\times4}(i,j) = R_{8\times8}(2i,2j)/2 \tag{23}$$

$$R_{8\times4}(i,j) = R_{8\times8}(i,2j)/\sqrt{2} \tag{24}$$

$$R_{4\times8}(i,j) = R_{8\times8}(2i,j)/\sqrt{2} \tag{25}$$

As it can be seen, the magnitude of the elements from the $2i^{th}$ row and the $2j^{th}$ column of $R_{8\times8}$, $R_{8\times8}(2i, 2j)$, are nicely twice as large as that from the i^{th} row and the j^{th} column of $R_{4\times4}$, $R_{4\times4}(i, j)$. So when $R_{n\times n}$ is replaced by scaling matrix $S_{n\times n}$ in practical implementation, just as (9) dose, $S_{8\times8}(2i, 2j)$ will be half of $S_{4\times4}(i, j)$. This difference can be easily ironed out by shifting, which means that $S_{8\times8}(2i, 2j)$ can be used as $S_{4\times4}(i, j)$, if the number of right shifting bits N in (9) for the 4×4 scaling is 1 bit less that needed in 8×8 scaling.

However, as for $R_{4\times8}$ and $R_{8\times4}$, because of the divisor $\sqrt{2}$ in (24) and (25), the scaling matrix $S_{8\times8}$ cannot be compatibly used for $S_{4\times8}$ and $S_{8\times4}$ as mentioned above. To solve this problem, a new technology, named PIT, is introduced to this compatible scheme.

4.2 The PIT scheme

The PIT scheme is first proposed for the purpose of reducing memory requirement and computational burden for the inverse ICT without increasing additional complexity for the forward ICT 5. Figure 2 shows the block diagram of a conventional ICT scheme, where forward scaling and quantization are combined as one step on the encoder side, and inverse scaling and dequantization are combined as another step on the decoder side.

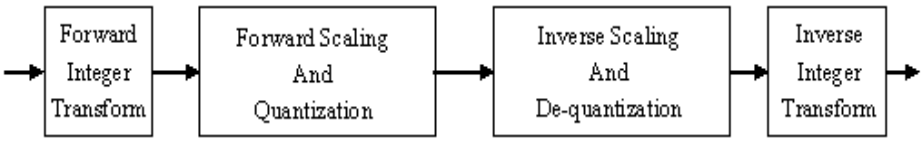

Figure 2. Block diagram of a conventional ICT scheme

With PIT scheme, the scaling process of the inverse ICT is moved to the encoder side and combined with that of the forward ICT as one single process. So (6) should be changed to (26) as below.

$$F_{n\times m} = (ICT_n \times f_{n\times m} \times ICT_m^T)//(R_{n\times m} \otimes R_{n\times m}) \tag{26}$$

So the scaling matrix $S_{n\times m}$ should be changed to a new one, defined as $P_{n\times m}$, and the forward ICT process can be represented by (27).

$$F_{n\times m} = (ICT_n \times f_{n\times m} \times ICT_m^T) \otimes P_{n\times m} >> N' \tag{27}$$

Combined with the quantization process, the whole process, including the combined scaling and quantization can be implemented by (28),

$$QF_{n\times m}(QP) = (ICT_n \times f_{n\times m} \times ICT_m^T) \otimes C_{n\times m}(QP) >> M' \tag{28}$$

where $C_{n\times m}(QP)$ is the new quantization matrix instead of $A_{n\times m}(QP)$ and M' is determined by the quantization degree.

Since the scaling process of the inverse ICT is moved to the encoder side, there is no scaling process on the decoder side and thus no need to store the scaling matrix. Figure 3 shows the block diagram of a PIT scheme.

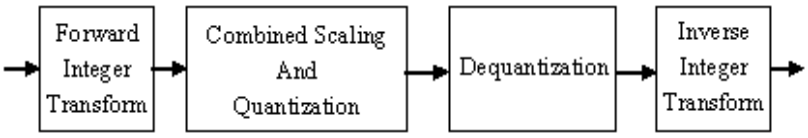

Figure 3. Block diagram of a PIT scheme

It should be noticed that when the PIT scheme is used, some requirements should be imposed on the ICT kernel itself. This is because when the inverse scaling is moved to the encoder side, the additional $R_{n\times m}$ in (26) can be viewed as an implicit frequency weighting matrix applied to the transformed signals. In order not to change too much of the energy distribution of the signals in the transform domain, the coefficients of the frequency weighting matrix, $R_{n\times m}$, should be closed to each other. From (7) and (8), it can be deduced that the norm of the row vectors of ICT_n and ICT_m should be close to each other. The similarity factor η is defined to represent the similarity of the coefficients of the ICT kernel 5. Plenty of experimental results show that the small the η is, the better performance the PIT scheme can achieve 5.

$$\eta = \frac{\max\limits_{0\le i<n,0\le j<m}(R_{n\times m}(i,j)) - \min\limits_{0\le i<n,0\le j<m}(R_{n\times m}(i,j))}{\max\limits_{0\le i<n,0\le j<m}(R_{n\times m}(i,j))} \tag{29}$$

4.3 The compatible scaling matrix

The PIT scheme introduced above enable the compatibility of scaling matrix especially for 8×4 and 4×8 ICT. If we denote the $(R_{n\times m}\otimes R_{n\times m})$ in (26) as $RR_{n\times m}$, according to (23), (24) and (25), it is not difficult to find that the relationship of $RR_{4\times4}$, $RR_{4\times8}$, $RR_{8\times4}$ and $RR_{8\times8}$ can be deduced as below.

$$RR_{4\times4}(i,j) = RR_{8\times8}(2i,2j)/4 \tag{30}$$

$$RR_{8\times4}(i,j) = RR_{8\times8}(i,2j)/2 \tag{31}$$

$$RR_{4\times8}(i,j) = RR_{8\times8}(2i,j)/2 \tag{32}$$

As it can be seen, the magnitude of the elements from the $2i^{th}$ row and the $2j^{th}$ column of $RR_{8\times8}$, $RR_{8\times8}(2i, 2j)$, is four times as large as that from the i^{th} row and the j^{th} column of $RR_{4\times4}$, $RR_{4\times4}(i, j)$. Similarly, the magnitude of elements $RR_{8\times8}(i, 2j)$ and $RR_{8\times8}(2i, j)$ is twice as large as that of $RR_{8\times4}(i, j)$ and $RR_{4\times8}(i, j)$ respectively. So when $RR_{n\times n}$ is replaced by the new PIT scaling matrix $P_{n\times n}$ in practical implementation, just as (27) dose, $P_{8\times8}(2i, 2j)$ will be quarter of $P_{4\times4}(i, j)$, and $P_{8\times8}(i, 2j)$ and $P_{8\times8}(2i, j)$ will be half of $P_{8\times4}(i, j)$ and $P_{4\times8}(i, j)$ respectively. These difference can also be easily ironed out by shifting, which means that $P_{8\times8}(2i, 2j)$ can be used as $P_{4\times4}(i, j)$, if the number of right shifting bits N' in (27) for the 4×4 scaling is 2 bits less that needed in 8×8 scaling. As for 4×8 and 8×4 scaling, the element of position (2i, j) and (i, 2j) in $P_{8\times8}$ can be used as $P_{4\times8}(i, j)$ and $P_{8\times4}(i, j)$, if the number of right shifting bits N' in (27) for both 4×8 and 8×4 scaling is 1 bit less that needed in 8×8 scaling.

In short, the scaling matrix for 8×4, 4×8 and 4×4 ICT can be involved in that for 8×8 scaling, if right shifting is done properly. Thus the compatibility of scaling matrix is achieved. As mentioned above, scaling process is usually combined with quantization process. Under this circumstance, the 8×8 quantization matrix $C_{8\times8}(QP)$ can also be used for 8×4, 4×8 and 4×4 scaling and quantization, if the quantization degree for every transform coefficient in one block is equal.

For the inverse ICT, there is no scaling process and no need to store the scaling matrix also if the PIT scheme is applied.

5. AN EXAMPLE OF ABT COMPATIBLE SCHEME

In this section, an example of compatible ICT-based ABT scheme with block-size 8×8, 4×8, 8×4 and 4×4 is given out. This example is implemented using simple 16-bit integer arithmetic and has been integrated into one of the H.264's software platform JM50c, which supports the ICT-based ABT with block-size 8×8, 4×8, 8×4 and 4×4.

Firstly, an 8×8 ICT kernel and a 4×4 ICT kernel have to be found, which are compatible as described in section 3 and the norm of their row vectors should be very close to each other as mentioned in section 4.2. On the other hand, the ICT kernel should also satisfied the basic requirements introduced in section 2.1 to bring a good tradeoff between computational complexity and coding efficiency. After taking all these requirements into consideration, we choose the ICT kernels shown below as the ICT_4 and ICT_8 respectively. It is not difficult to find that the ICT_4 is exactly the same as the T_{4u} module of ICT_8, so the transform unit used to process the T_{4u} of the ICT_8 can be reused to process the ICT_4.

$$ICT_8 = \begin{bmatrix} 8 & 8 & 8 & 8 & 8 & 8 & 8 & 8 \\ 10 & 9 & 6 & 2 & -2 & -6 & -9 & -10 \\ 10 & 4 & -4 & -10 & -10 & -4 & 4 & 10 \\ 9 & -2 & -10 & -6 & 6 & 10 & 2 & -9 \\ 8 & -8 & -8 & 8 & 8 & -8 & -8 & 8 \\ 6 & -10 & 2 & 9 & -9 & -2 & 10 & -6 \\ 4 & -10 & 10 & -4 & -4 & 10 & -10 & 4 \\ 2 & -6 & 9 & -10 & 10 & -9 & 6 & -2 \end{bmatrix} / 2 \tag{33}$$

$$ICT_4 = \begin{bmatrix} 4 & 4 & 4 & 4 \\ 5 & 2 & -2 & -5 \\ 4 & -4 & -4 & 4 \\ 2 & -5 & 5 & -2 \end{bmatrix} \tag{34}$$

Figure 4 illustrates the 8-point forward and inverse transform butterfly structures respectively. It is obvious that the butterfly structures inside the frames are the 4-point transform structures, which can be used to process the transform of ICT_4 in this example.

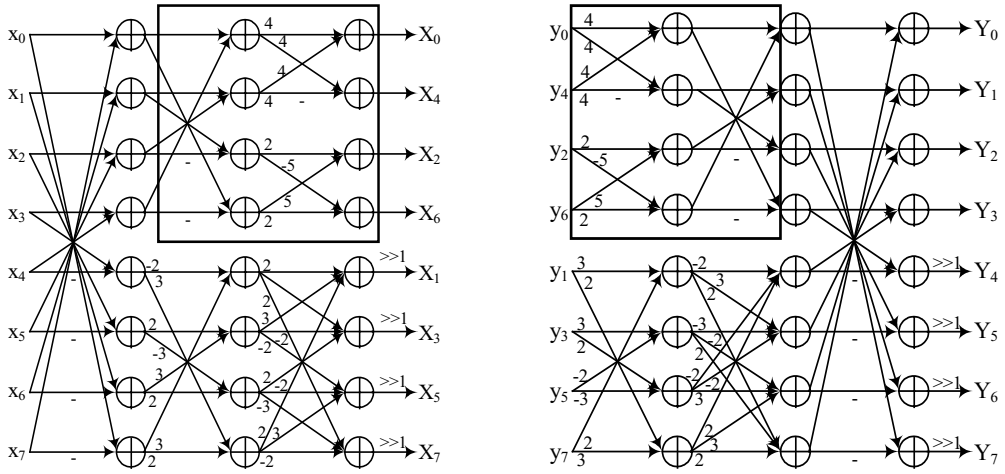

Figure 4. The forward and inverse transform butterfly structures for ICT_8

The scaling matrix $P_{8\times8}$ given by (35) is applied to the implementation of (27) for the 8×8 ICT and the N' is equal to 20 in this case. As introduced in section 4.3, the $P_{8\times8}$ can be compatibly used for $P_{8\times4}$, $P_{4\times8}$ and $P_{4\times4}$.

The elements from the positions of (2i, 2j) in $P_{8\times8}$ form $P_{4\times4}$, and similarly, the elements from the positions of (i, 2j) and (2i, j) in $P_{8\times8}$ form $P_{8\times4}$ and $P_{4\times8}$ respectively, but different N' should be applied for these different block-size ICT. In this example, N' is 18 for the 4×4 ICT and 19 for both 4×8 and 8×4 ICT.

$$P_{8\times 8} = \begin{bmatrix} 8192 & 9489 & 9039 & 9489 & 8192 & 9489 & 9039 & 9489 \\ 9489 & 10992 & 10471 & 10992 & 9489 & 10992 & 10471 & 10992 \\ 9039 & 10471 & 9975 & 10471 & 9039 & 10471 & 9975 & 10471 \\ 9489 & 10992 & 10471 & 10992 & 9489 & 10992 & 10471 & 10992 \\ 8192 & 9489 & 9039 & 9489 & 8192 & 9489 & 9039 & 9489 \\ 9489 & 10992 & 10471 & 10992 & 9489 & 10992 & 10471 & 10992 \\ 9039 & 10471 & 9975 & 10471 & 9039 & 10471 & 9975 & 10471 \\ 9489 & 10992 & 10471 & 10992 & 9489 & 10992 & 10471 & 10992 \end{bmatrix} \quad (35)$$

Since the scaling process is usually combined with quantization, (28) is more likely to be applied in practical implementations. When $C_{8\times 8}$ is used instead of $P_{8\times 8}$, the compatibility can be achieved the same way if the quantization degree is equal to each transform coefficients in one transform block.

For a given QP, the quantization matrix $C_{8\times 8}(QP)$ is unique, so 8×8-sized matrix is needed for every QP. To reduce the required memory, a novel quantization method called logarithmic quantization scheme is introduced in H.264 8. For every increase of 6 in QP, the quantization degree doubles and can be controlled by right shifting one more bit with no change in the multiplicators in $C_{n\times m}$, and therefore the numbers of QP $C_{n\times m}$ should cover reduce to 6 due to the periodicity. Since this example is integrated into JM50C, the combination of scaling and quantization is implemented as (36),

$$\mathrm{QF}_{n\times m}(QP) = (\mathrm{ICT}_n \times \mathrm{f}_{n\times m} \times \mathrm{ICT}_{\mathrm{m}}^{\mathrm{T}}) \otimes \mathrm{C}_{n\times m}(QP\%6) >> M' \quad (36)$$

where

$$M' = N' + (QP/6). \quad (37)$$

As for the inverse ICT, the butterfly structure for the inverse 8×8 transform can be reused by the inverse 4×4 transform likewise, and no scaling process is needed.

6. EXPERIMENTAL RESULTS AND ANALYSIS

In our experiments, three different ICT-based ABT schemes all with block-size 8×8, 4×8, 8×4 and 4×4 have been tested using the software platform JM50c. The first is what was once adopted by H.264 and already existed in JM50c. The second scheme uses the proposed ICT kernel (33) and (34), but it is not a compatible scheme and PIT is not employed. The third is the compatible scheme proposed in section 5. All of them are implemented with simple 16-bit integer arithmetic. The sequences used for the simulations are high-definition (HD) videos with the resolution 720×576. This is because ABT scheme is more likely to be applied to HD sequence and bring higher performance. The test conditions described in Table 1 are applied in the simulations. Compared with the first scheme, the performance of the other two, named Non-compatible ABT and Compatible ABT respectively, are given in Table 2 in the form of average Peak Signal-to-Noise Ratio (PSNR) gains with equal bit rates using the method proposed in 9.

Table 1. Test Conditions and Encoding Parameters

Stream Structure	IPPP… and IBBPBBP…
Frame Rate	60 fps
Intra Period	0.5 s
Search Range	± 32
Inter-frame Search	All modes Enabled
Reference Frame	2

Entropy Coding Method	CABAC
QP	Fixed (20, 24, 28, 32)
RDO	On

Table 2. The Performance of Different ABT Schemes

Sequence	Gain (dB) with stream structure IPPPPPP		Gain (dB) with stream structure IBBPBBP	
	Non-compatible ABT	Compatible ABT	Non-compatible ABT	Compatible ABT
City	0.0000353	0.00138	-0.00263	0.002862
Harbour	0.200495	0.049835	0.004184	0.005504
Night	-0.000341619	0.005748	0.000397	0.000638
Raven	0.019980144	0.004821	-0.00051	0.001095

From the experimental results, we can conclude that the performance of these three ABT scheme is almost the same. However, the proposed approach is superior to conventional non-compatible ABT schemes for its significantly complexity reduction that is analyzed below.

In the proposed approach, the 4-point 1-D transform butterfly structure that will be used in 8×4, 4×8 and 4×4 ICT can be involved in the 8-point one if the ICT_4 is chosen appropriately. So if the transform is implemented by butterfly operations, the area for an independent 4-point 1-D transform butterfly structure can be saved.

Using the compatible scaling matrix and introducing the PIT, the proposed approach makes the memory requirement greatly reduced. First, there is no scaling process and thus no scaling matrix stored on the decoder side. In conventional ABT systems, if scaling is combined with dequantization process, 2-Dmemory of 4×4+4×8+8×4+8×8=144 bytes are needed to store the dequantization matrix for each QP. If the dequantization matrix is fully expanded, a 3-D memory of 144×K bytes is required for totally K different QPs. In H.264, K is 6 because of the QP period. With this approach, such a large memory requirement can be avoided and a 1-D memory of only 6 bytes is needed instead. Otherwise, if the dequantization matrix is not expanded, the computational complexity will be much lower because 3-D lookup operation is replaced by 1-D lookup operation.

As for the encoder side, the scaling matrix for 8×8 ICT involves those for other block-size ICT. The memory is also greatly saved. Additional 4×4+8×4+4×8 = 80 16-bit integers for quantization matrix will be saved for each QP and K×80 16-bit integers will be saved for totally K different QPs. In H.264, 6×80=480 16-bit integers will be saved.

7. CONCLUSION

In this paper, the approach to compatible ICT-based ABT is presented. The rule of choosing compatible ICT kernels is given so that only one transform unit is required. With PIT, the compatibility of scaling matrices especially for 8×4 and 4×8 ICT can be achieved, and the computational burden for the decoder side is greatly reduced also. The method of generating the compatible scaling matrix and how to use it is also introduced. The analysis and simulations show that this approach greatly saves hardware resources and makes the implementation of ABT much easier without loss of performance.

REFERENCE

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. Draft Text of H.264/AVC Fidelity Range Extensions Amendment, Document JVT-L047, Jul. 2004.

3. Srinivasan. Sridhar, Hsu. Pohsiang, Holcomb. Tom, Mukerjee. Kunal, Regunathan. Shankar L, et. al. "Windows Media Video 9: overview and applications", Signal Processing: Image Communication Volume: 19, Issue: 9, pp. 851-875 October, 2004.
4. Gary Sullivan, Ajay Luthra, Thomas Wiegand, "(Draft) Report of 5th JVT Meeting, Geneva, CH, 9-17 October 2002", JVT Doc. JVT-E001, Oct. 2002
5. Ci-Xun Zhang, Jian Lou, Lu Yu, Jie Dong, Cham W.K "The Technique of Pre-Scaled Integer Transform in Hybrid Video Coding", accepted by ISCAS 2005.
6. G. Bjontegaard, "Response to call for proposals for H.26L", ITU-T SG16 Doc. Q15-F-11, Oct. 1998.
7. G.A. Ruiz, J.A. Michell, A.M. Burón, J.M. Solana, M.A. Manzano, J. Díaz, "An integer cosine transform chip design for image compression", Proc. SPIE, vol. 5117, pp.33-41, 2003.
8. Louis Kerofsky, Shawmin Lei, "Reduced bit-depth quantization", ITU-T SG16 Doc. VCEG-N20, Aug, 2001
9. G. Bjontegaard, "Calculation of average PSNR differences between RD-curves," ITU-T SG16 Doc. VCEG-M33, 2001.