Chapter 8

# Fast Algorithm Designs of Multiple-Mode Discrete Integer Transforms with Cost-Effective and Hardware-Sharing Architectures for Multistandard Video Coding Applications

Chih-Peng Fan

Additional information is available at the end of the chapter

http://dx.doi.org/10.5772/64985

**Abstract**

In this chapter, first we give a brief view of transform-based video coding. Second, the basic matrix decomposition scheme for fast algorithm and hardware-sharing-based integer transform design are described. Finally, two case studies for fast algorithm and hardware-sharing-based architecture designs of discrete integer transforms are presented, where one is for the single-standard multiple-mode video transform-coding application, and the other is for the multiple-standard multiple-mode video transform-coding application.

**Keywords:** video coding, transform coding, fast algorithm, matrix factorization, hardware sharing, multiple modes, multiple standards

## 1. Introduction

Video-coding system has generally utilized block-based transform-coding skills to shrink the data rates by joining quantization and entropy coding. Among some block-based transforms, the discrete cosine transform (DCT) [1] and integer transforms have extensively been used to still image and video-coding specifications, such as JPEG [2], MPEG-1/2 [3, 4], MPEG-4 [5], H.264/AVC [6, 7], AVS [8, 9], VC-1 [10], VP8 [11], and HEVC [12]. Because integer transforms perform the low complexity and effective coding performance, the advanced video coding (AVC) in ITU-T H.264 [6, 7, 13, 14], which is also known as MPEG-4 part 10, applies integer transforms for transform process. The 4 × 4 and 8 × 8 transforms in [13, 14] were calculated exactly to prevent non-adaptation issues of inverse transforms for high-quality moving visual images. The VC-1 specification [10, 15, 16] employed 4 × 4 and 8 × 8 integer transforms, and it was developed by Microsoft Corporation and standardized by the Society of Motion Picture and Television Engineers (SMPTE). The 8 × 8 integer transform is utilized to obtain the high-coding performance in the Audio Video Coding Standard (AVS) for China [8, 9]. In [11], the VP8 video-coding standard was developed for Internet browser applications. The Joint Collaborative Team on Video Coding proposed the high-efficiency video coding (HEVC) specification [12]. By HEVC, the compression efficiency was greatly better than that achieved using the H.264/AVC high-profile-coding specification.

To support the single-standard H.264/AVC video coding, several transform architectures in [17–24] have been developed to approach the multiple transform modes in H.264. To support the single-standard H.265/HEVC video coding, several transform architectures in [25–32] have been developed to approach the multiple transform modes in HEVC. Besides, supporting multiple-standard functions in video coding has been an important issue in multimedia applications recently, such as H.264/AVC, MPEG-1/2/4, VC-1, AVS, and VP8 standards, and several transform architectures in [33–41] have also been developed to complete the multiple transform functions. Owing to the growth of multistandard video-coding applications, how to achieve low-computational complexities and implement by hardware-sharing-based cost-effective architectures simultaneously are interesting research topics for the VLSI design of video codecs.

## 2. Matrix decomposition preprocessing for fast algorithm and hardware-sharing-based designs

Based on the resemblance property, the 8 × 8 inverse integer transforms [41] in H.264/AVC, AVS, VC-1, VP8, MPEG-1/2/4, and HEVC specifications are revealed in Eq. (1), and **Table 1** depicts the coefficient values in the transforms.

$$C_{8\times 8} = \begin{bmatrix} a & b & f & c & a & d & g & e \\ a & c & g & -e & -a & -b & -f & -d \\ a & d & -g & -b & -a & e & f & c \\ a & e & -f & -d & a & c & -g & -b \\ a & -e & -f & d & a & -c & -g & b \\ a & -d & -g & b & -a & -e & f & -c \\ a & -c & g & e & -a & b & -f & d \\ a & -b & f & -c & a & -d & g & -e \end{bmatrix} \tag{1}$$

| Transform sizes | VC-1 | AVS | VP8 | MPEG-1/2/4 | H.264/AVC | HEVC |
|---|---|---|---|---|---|---|
| 4 × 4 | √ | √ | √ | N/A | √ | √ |
| 8 × 8 | √ | √ | N/A | √ | √ | √ |
| 16 × 16 | N/A | N/A | N/A | N/A | N/A | √ |
| 32 × 32 | N/A | N/A | N/A | N/A | N/A | √ |

**Table 1.** The transform modes in several video-coding standards [41].

In Eq. (1), it is decomposed by Eq. (2) as

$$C_{8\times8} = P_1 \cdot A_0 \cdot P_r. \tag{2}$$

In Eq. (2), $A_0$ is divided into two modules, $U_{4\times4}$ and $D_{4\times4}$, where $P_1 = \begin{bmatrix} 1&0&0&0&0&0&0&-1\\ 0&1&0&0&0&0&-1&0\\ 0&0&1&0&0&-1&0&0\\ 0&0&0&1&-1&0&0&0\\ 0&0&0&1&1&0&0&0\\ 0&0&1&0&0&1&0&0\\ 0&1&0&0&0&0&1&0\\ 1&0&0&0&0&0&0&1 \end{bmatrix}$,

$$P_r = \begin{bmatrix} 1&0&0&0&0&0&0&0\\ 0&0&1&0&0&0&0&0\\ 0&0&0&0&1&0&0&0\\ 0&0&0&0&0&0&1&0\\ 0&1&0&0&0&0&0&0\\ 0&0&0&1&0&0&0&0\\ 0&0&0&0&0&1&0&0\\ 0&0&0&0&0&0&0&1 \end{bmatrix}, A_0 = \begin{bmatrix} a&f&a&g&0&0&0&0\\ a&g&-a&-f&0&0&0&0\\ a&-g&-a&f&0&0&0&0\\ a&-f&a&-g&0&0&0&0\\ 0&0&0&0&-e&d&-c&b\\ 0&0&0&0&-d&b&-e&-c\\ 0&0&0&0&-c&e&b&d\\ 0&0&0&0&-b&-c&-d&-e \end{bmatrix}.$$

Thus

$$A_0 = U_{4\times4} \oplus D_{4\times4} \tag{3}$$

and $C_{8\times8}$ becomes

$$C_{8\times8} = P_1 \cdot (U_{4\times4} \oplus D_{4\times4}) \cdot P_r. \tag{4}$$

In (3), "⊕ " is the direct sum operator, and the two diagonal blocks $U_{4\times4}$ and $D_{4\times4}$ are processing in parallel. To cut down the computational operations and achieve effective hardware shares, the upper diagonal matrix $U_{4\times4}$ and the down diagonal matrix $D_{4\times4}$ are further decomposed into the cascaded multiplication form or the addition form of sparse matrices. After matrix factorizations, the chosen sparse matrices have the coefficients which are 1, −1, 0, or an integer, and an integer value can equal the combination of powers of two. Besides, zero factors in the chosen sparse matrices could be factorized as many as possible [42].

By Eq. (1), for VC-1 the values of the coefficient set {*a, b, c, d, e, f, g*} are {12, 16, 15, 9, 4, 16, 6}, and those for AVS are {8, 10, 9, 6, 2, 10, 4}. Next, those for MPEG-1/2/4 are {362, 502, 426, 284, 100, 473, 196}, and those for H.264/AVC are {8, 12, 10, 6, 3, 8, 4}. Finally, those for HEVC are {64, 89, 75, 50, 18, 83, 36}.

The general 4 × 4 inverse integer transform matrices [41] can be presented in Eq. (5) as

$$M_{4\times 4} = \begin{bmatrix} h & i & h & j \\ h & j & -h & -i \\ h & -j & -h & i \\ h & -i & h & -j \end{bmatrix}. \tag{5}$$

By Eq. (5), for VC-1 the values of the coefficient set {h, i, j} are {17, 22, 10}, and those for VP8 are {128, 167, 70}. Next, those for AVS-M are {2, 3, 1}, and those for H.264/AVC are {1, 1, 0.5}. Finally, those for HEVC are {64, 83, 36}.

## 3. Case study [32]: single-standard multiple-mode transform design

### 3.1. Hardware-sharing based 32 × 32 integer core transform for HEVC

The one-dimensional (1D) 32 × 32 inverse core transform for HEVC is described in [30]. By the symmetrical property, the 32 × 32 inverse core transform is presented as

$$H_{i32} = P_A \cdot C_{A1}, \tag{6}$$

where $C_{A1} = \begin{bmatrix} C_{11} & C_{12} \\ C_{21} & C_{22} \end{bmatrix}$, $P_A = \begin{bmatrix} I_{16x16} & -\tilde{I}_{16x16} \\ \tilde{I}_{16x16} & I_{16x16} \end{bmatrix}$, $\tilde{I}_{16x16} = \begin{bmatrix} 0 & 0 & \cdots & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 \\ \vdots & \vdots & \iddots & 0 & \vdots \\ 0 & 1 & 0 & \vdots & 0 \\ 1 & 0 & \cdots & 0 & 0 \end{bmatrix}$, and $I_{16\times 16}$ is a 16 × 16 identity matrix. In Eq. (6), $P_A$ is the butterfly-like postprocessing, and $C_{A1}$ is the sparse matrix. By swapping each column of $C_{A1}$, it becomes

$$C_{A1} = C_{A2} \cdot P_{Ar}. \tag{7}$$

By Eqs. (6) and (7), $H_{i32}$ becomes

$$H_{i32} = P_A \cdot C_{A2} \cdot P_{Ar}, \tag{8}$$

where $P_{\mathrm{Ar}}$ is the permutation matrix. In Eq. (7), $C_{A2}$ is expressed by

$$C_{A2} = \begin{bmatrix} T_{A11} & 0_{16x16} \\ 0_{16x16} & T_{A22} \end{bmatrix} = T_{A11} \oplus T_{A22}, \tag{9}$$

where "⊕" means the direct sum operation, and then $T_{A11}$ and $T_{A22}$ are 16 × 16 matrices, which are revealed in [32]. The matrix $P_{\mathrm{Ar}}$ in Eq. (8) is expressed as

$$P_{Ar} = P(2,16), \tag{10}$$

where the permutation matrix $P(m, n)$ is defined in [43], and the notation "⊗" means the Kronecker product. In Eq. (9), $A_{A22}$ is presented as

$$T_{A22} = T_{M1} + T_{N1}, \tag{11}$$

First, the lower half of $C_{N1}$ is divided into sixteen 8 × 1 column vectors $X_i$, where $i = 0, 1, 2, \ldots,$ 15, and then $T_{N1}$ becomes

$$T_{N1} = \begin{bmatrix} & 0_{8x16} & \\ - - - & - - - & - - - \\ X_0 \quad X_1 & \ldots & X_{15} \end{bmatrix}. \tag{12}$$

Second, the coefficients in a single column vector can be shared. The vector coefficient computations are achieved by integrating several base coefficients [32]. After realizing the column vectors of $T_{N1}$, the lower half of $T_{N1}$ is factorized as an integration of eight 1 × 16 row vectors depicted as $Y_i$, where $i = 8, 9, \ldots,$ and 15, and $T_{N1}$ becomes

$$T_{N1} = \begin{bmatrix} 0_{8x16} \\ - - - \\ Y_8 \\ Y_9 \\ \vdots \\ Y_{15} \end{bmatrix}. \tag{13}$$

Adder tree structures are utilized to calculate the aggregate results for the row vectors $Y_8$–$Y_{15}$ [32]. By the duplicate operations for $T_{N1}$, $T_{M1}$ is presented as

$$T_{M1} = \begin{bmatrix} \hat{X}_0 \cdots \hat{X}_{15} \\ ----- \\ 0_{8x16} \end{bmatrix}, \tag{14}$$

where $\hat{X}_i$ is an 8 × 1 column vector, where $i = 0, 1, 2, \ldots,$ and 15. Then, $T_{M1}$ becomes

$$T_{M1} = \begin{bmatrix} Y_0 \\ \vdots \\ Y_7 \\ --- \\ 0_{8x16} \end{bmatrix}, \tag{15}$$

where $Y_i$ is a 16 × 1 row vector, where $i = 0, 1, \ldots,$ and 7. The realization of $T_{M1}$ equals that of $T_{N1}$. Finally, the operations of $T_{M1}$ and $T_{N1}$ are merged to $T_{A22}$. The computational operations $T_{A22}$ require 630 additions and 326 shift operations [32]. The matrix $T_{A11}$ in Eq. (9), which is also denoted as $H_{i16}$, is the 1D 16 × 16 inverse core transform in HEVC [30].

### 3.2. Hardware-sharing-based 16 × 16 integer core transform for HEVC

The 16 × 16 integer core transform in [30] changes into

$$H_{i16} = P_B \cdot C_{B1}, \tag{16}$$

where $P_B = \begin{bmatrix} I_{8x8} & -\tilde{I}_{8x8} \\ \tilde{I}_{8x8} & I_{8x8} \end{bmatrix}$, and $C_{B1}$ is revealed in [32]. By swapping each column of $C_{B1}$, it will be

$$C_{B1} = C_{B2} \cdot P_{Br}, \tag{17}$$

where $P_{\mathrm{Br}} = P(8,2)$. By Eqs. (16) and (17), $H_{i16}$ is expressed by

$$H_{i16} = T_{A11} = P_B \cdot C_{B2} \cdot P_{Br}. \tag{18}$$

In Eq. (18), $C_{B2}$ is presented as

$$C_{B2} = \begin{bmatrix} T_{B11} & 0_{8x8} \\ 0_{8x8} & T_{B22} \end{bmatrix} = T_{B11} \oplus T_{B22}, \tag{19}$$

and $T_{B22}$ becomes

$$T_{B22} = T_{M2} + T_{N2}, \tag{20}$$

where $T_{M2} = \begin{bmatrix} -9 & 25 & -43 & 57 & -70 & 80 & -87 & 90 \\ -25 & 70 & -90 & 80 & -43 & -9 & 57 & -87 \\ -43 & 90 & -57 & -25 & 87 & -70 & -9 & 80 \\ -57 & 80 & 25 & -90 & 9 & 87 & -43 & -70 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix},$

$$T_{N2} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ -70 & 43 & 87 & -9 & -90 & -25 & 80 & 57 \\ -80 & -9 & 70 & 87 & 25 & -57 & -90 & -43 \\ -87 & -57 & -9 & 43 & 80 & 90 & 70 & 25 \\ -90 & -87 & -80 & -70 & -57 & -43 & -25 & -9 \end{bmatrix}.$$

By the duplicate processed of $T_{N1}$ in Section 3.1, $T_{N2}$ turns into

$$T_{N2} = \begin{bmatrix} 0_{4x8} \\ ------ \\ U_0 \ \dots \ U_7 \end{bmatrix}, \tag{21}$$

where $U_i$ is an $8 \times 1$ column vector, where $i = 0, 1, 2, \dots,$ and 7. Next, $T_{N2}$ also is

$$T_{N2} = \begin{bmatrix} 0_{4x8} \\ --- \\ V_4 \\ \vdots \\ V_7 \end{bmatrix}, \tag{22}$$

where $V_i$ is a 1 × 8 row vector, where $i = 4, 5, 6$, and 7. Adder tree schemes are applied to compute the summed outcomes of $V_4$–$V_7$ [32]. By the same processes of $T_{M1}$ in Section 3.1, $T_{M2}$ becomes

$$T_{M2} = \begin{bmatrix} \hat{U}_0 & \cdots & \hat{U}_7 \\ -- & -- & -- \\ & 0_{4x8} & \end{bmatrix}, \tag{23}$$

where $\hat{U}_i$ is a 4 × 1 column vector, where $i = 0, 1, 2, \ldots$, and 7. Next, $T_{M2}$ also is

$$T_{M2} = \begin{bmatrix} V_0 \\ \vdots \\ V_3 \\ --- \\ 0_{4x8} \end{bmatrix}, \tag{24}$$

where $V_i$ is a 1 × 8 row vector, where $i = 0, 1, 2$, and 3. Then, adder trees are used to treat the row vectors $V_0$–$V_3$ [32]. Finally, the calculations of $T_{M2}$ and $T_{N2}$ are merged to $T_{B22}$. The computational operations of $T_{B22}$ are 164 additions and 106 shift operations [32]. Meantime, the $T_{B11}$ in Eq. (19), which is also denoted as $H_{i8}$, is the 1D 8 × 8 inverse core transform in HEVC [30].

### 3.3. Hardware-sharing-based 8 × 8 integer core transform for HEVC

The 8 × 8 integer transform in [30] is described as

$$H_{i8} = P_C \cdot C_{C1}, \tag{25}$$

where $P_C = \begin{bmatrix} I_{4\text{x}4} & -\tilde{I}_{4\text{x}4} \\ \tilde{I}_{4\text{x}4} & I_{4\text{x}4} \end{bmatrix}$, and $C_{C1} = \begin{bmatrix} 64 & 0 & 83 & 0 & 64 & 0 & 36 & 0 \\ 64 & 0 & 36 & 0 & -64 & 0 & -83 & 0 \\ 64 & 0 & -36 & 0 & -64 & 0 & 83 & 0 \\ 64 & 0 & -83 & 0 & 64 & 0 & -36 & 0 \\ 0 & -18 & 0 & 50 & 0 & -75 & 0 & 89 \\ 0 & -50 & 0 & 89 & 0 & -18 & 0 & -75 \\ 0 & -75 & 0 & 18 & 0 & 89 & 0 & 50 \\ 0 & -89 & 0 & -75 & 0 & -50 & 0 & -18 \end{bmatrix}$. After swapping each column in $C_{C1}$, it changes into

$$C_{C8} = C_{C2} \cdot P_{Cr}, \tag{26}$$

where $P_{Cr} = \begin{bmatrix} 1&0&0&0&0&0&0&0\\ 0&0&1&0&0&0&0&0\\ 0&0&0&0&1&0&0&0\\ 0&0&0&0&0&0&1&0\\ 0&1&0&0&0&0&0&0\\ 0&0&0&1&0&0&0&0\\ 0&0&0&0&0&1&0&0\\ 0&0&0&0&0&0&0&1 \end{bmatrix}$. Based on Eqs. (25) and (26), $H_{i8}$ is presented by

$$H_{i8} = T_{B11} = P_C \cdot C_{C2} \cdot P_{Cr}, \tag{27}$$

In Eq. (27), $C_{C2}$ becomes

$$C_{C2} = \begin{bmatrix} T_{C11} & 0_{4x4} \\ 0_{4x4} & T_{C22} \end{bmatrix} = T_{C11} \oplus T_{C22}, \tag{28}$$

where $T_{C11} = \begin{bmatrix} 64 & 83 & 64 & 36 \\ 64 & 36 & -64 & -83 \\ 64 & -36 & -64 & 83 \\ 64 & -83 & 64 & -36 \end{bmatrix}$ and $T_{C22} = \begin{bmatrix} -18 & 50 & -75 & 89 \\ -50 & 89 & -18 & -75 \\ -75 & 18 & 89 & 50 \\ -89 & -75 & -50 & -18 \end{bmatrix}$.

In Eq. (28), $T_{C22}$ is factorized as

$$T_{C22} = S_1 + S_2, \tag{29}$$

where $S_1 = \begin{bmatrix} -18 & 0 & 0 & 89 \\ 0 & 89 & -18 & 0 \\ 0 & 18 & 89 & 0 \\ -89 & 0 & 0 & -18 \end{bmatrix}$. Moreover, $S_1$ is expressed by

$$S_1 = Z_1 + (18 \cdot Z_2), \tag{30}$$

where $Z_1 = \begin{bmatrix} 0 & 0 & 0 & -1 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix}$ and $Z_2 = \begin{bmatrix} -1 & 0 & 0 & 5 \\ 0 & 5 & -1 & 0 \\ 0 & 1 & 5 & 0 \\ -5 & 0 & 0 & -1 \end{bmatrix}$. In Eq. (29), $S_2$ is presented as

$$S_2 = 25 \cdot Z_3, \tag{31}$$

where $Z_3 = \begin{bmatrix} 0 & 2 & -3 & 0 \\ -2 & 0 & 0 & -3 \\ -3 & 0 & 0 & 2 \\ 0 & -3 & -2 & 0 \end{bmatrix}$. By Eqs. (29)– (31), $T_{C22}$ becomes

$$T_{C22} = Z_1 + (18 \cdot Z_2) + (25 \cdot Z_3). \tag{32}$$

In Eq. (32), the computations of $T_{C22}$ require 36 additions and 28 shift operations [32]. The matrix $T_{C11}$ in Eq. (28) is also the 1D 4 × 4 inverse core transform matrix in HEVC.

### 3.4. Hardware-sharing-based 4 × 4 integer core transform for HEVC

The 4 × 4 integer core transform matrix is indicated as

$$H_{i4} = P_D \cdot C_{D1}, \tag{33}$$

where $P_D = \begin{bmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & -1 \\ 1 & 0 & -1 & 0 \end{bmatrix}$ and $C_{D1} = \begin{bmatrix} 64 & 0 & 64 & 0 \\ 64 & 0 & -64 & 0 \\ 0 & -36 & 0 & 83 \\ 0 & -83 & 0 & -36 \end{bmatrix}$. By swapping each column of $C_{D1}$, it changes into

$$C_{D1} = C_{D2} \cdot P_{D2}. \tag{34}$$

where $P_{\mathrm{Dr}} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$. From Eqs. (33) and (34), $H_{i4}$ is described by

$$H_{i4} = T_{C11} = P_D \cdot C_{D2}.P_{Dr}. \tag{35}$$

In Eq. (34), $C_{D2}$ is rewritten as

$$C_{D2} = T_{D11} \oplus T_{D22}. \tag{36}$$

In Eq. (36), $T_{D11}$ becomes

$$T_{D11} = 64 \cdot Z_4, \tag{37}$$

where $Z_4 = \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}$. In Eq. (36), $T_{D22}$ is indicated by $Z_5$ and $Z_6$ as

$$T_{D22} = 36 \cdot Z_5 + 11 \cdot Z_6, \tag{38}$$

where $Z_5 = \begin{bmatrix} 2 & 1 \\ 1 & -2 \end{bmatrix}$ and $Z_6 = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}$. Thus, the computations of $T_{D22}$ are 10 additions and 10 shift operations [32]. Based on Eqs. (35)– (38), $H_{i4}$ is changed into

$$H_{i4} = P_D \cdot [(64 \cdot Z_4) \oplus (36 \cdot Z_5 + 11 \cdot Z_6)] \cdot P_{Dr}. \tag{39}$$

By the abovementioned discussions, the hardware modules of 4 × 4, 8 × 8, and 16 × 16 inverse core transforms are shared to implement $H_{i8}$, $H_{i16}$, and $H_{i32}$, respectively [32]. By sharing the hardware of $H_{i4}$ in Eq. (39), the cost-effective design of the 8 × 8, 16 × 16, and 32 × 32 inverse core transforms is obtained progressively. First, the hardware-sharing-based eight-point inverse transform is presented as

$$H_{i8} = P_C \cdot \{H_{i4} \oplus [Z_1 + (18 \cdot Z_2) + (25 \cdot Z_3)]\} \cdot P_{Cr}. \tag{40}$$

Next, the hardware-sharing-based 16-point inverse transform is described as

$$H_{i16} = P_B \cdot \{H_{i8} \oplus [T_{M2} + T_{N2}]\} \cdot P_{Br}. \tag{41}$$

Finally, the hardware-sharing-based 32-point inverse transform is depicted as

$$H_{i32} = P_A \cdot \{H_{i16} \oplus [T_{M1} + T_{N1}]\} \cdot P_{Ar}. \tag{42}$$

In this section, the hardware-sharing transform architecture cuts down the hardware cost because the same submodules and coefficients of the transforms are extracted to be shared. **Figure 1** illustrates the architecture of the hardware-sharing-based inverse core transform design for 4 × 4/8 × 8/16 × 16/32 × 32 transforms [32].

## 3.5. Architecture comparison

The proposed 1D inverse core transform in [32] involves four inputs to sustain 4 × 4, 8 × 8, 16 × 16, and 32 × 32 transform modes. Several multiplexers are utilized to acquire the transform outputs of the 32 × 32 inverse core transform by the shared design of 4 × 4, 8 × 8, and 16 × 16 inverse core transforms [32]. **Table 2** lists the number of adders and shifters needed to calculate four modes of the 1D inverse core transform for HEVC. The developed architecture in [32] does not require any multiplier, and the fixed-coefficient multiplications are replaced with simple additions and shift operations. **Table 3** shows the comparison of three 16-point inverse transform designs. Compared with the previous works in [29] and [31], the applied architecture contains fewer adders. However, several more shifters are required. Compared with the cost of adders, the shifters need lower hardware expense. Thus, the used architecture decreases the hardware cost more efficiently than previous transform schemes do.

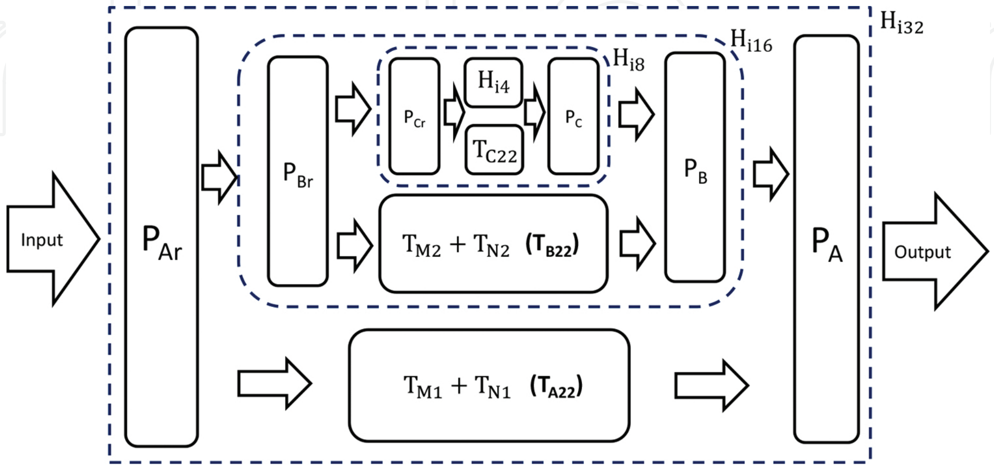

**Figure 1.** The hardware-sharing-based inverse core transform structure for HEVC.

| Transform sizes | 32 × 32 | 16 × 16 | 8 × 8 | 4 × 4 |
|---|---|---|---|---|
| No. of shifters | 256 | 93 | 40 | 11 |
| No. of adders | 461 | 146 | 64 | 10 |

**Table 2.** The 1D inverse transform architecture at different transform modes [32].

| Designs | No. of shifters | No. of adders |
|---|---|---|
| Ahmed [29] | 132 | 232 |
| Haggag [31] | 58 | 242 |
| Design in Section 3.2 | 93 | 146 |

**Table 3.** Hardware comparison of three 1D 16-point transform designs [32].

## 4. Case study [41]: multiple-standard multiple-mode transform design

### 4.1. Hardware-sharing design for 8 × 8 transforms mode

For H.264/AVC, the transform matrix is employed as a foundation matrix for the multistandard hardware-sharing scheme. Based on Eq. (3), the cost of the upper diagonal matrix in Eq. (43) is eight adders and two shifters.

$$U_{4\times 4\_AVC} = \begin{bmatrix} 8 & 8 & 8 & 4 \\ 8 & 4 & -8 & -8 \\ 8 & -4 & -8 & 8 \\ 8 & -8 & 8 & -4 \end{bmatrix} = 8 \cdot C_1 \cdot C_2, \tag{43}$$

where $C_1 = \begin{bmatrix} 1 & 0 & 0 & 1 \\ 0 & -1 & 1 & 0 \\ 0 & 1 & 1 & 0 \\ 1 & 0 & 0 & -1 \end{bmatrix}$, and $C_2 = \begin{bmatrix} 1 & 0 & 1 & 0 \\ 0 & -0.5 & 0 & 1 \\ 1 & 0 & -1 & 0 \\ 0 & 1 & 0 & 0.5 \end{bmatrix}$. For AVS, the upper diagonal matrix $U_{4\times 4\_\mathrm{AVS}}$ in Eq. (44) costs 10 adders and four shifters.

$$U_{4\times 4\_AVS} = \begin{bmatrix} 8 & 10 & 8 & 4 \\ 8 & 4 & -8 & -10 \\ 8 & -4 & -8 & 10 \\ 8 & -10 & 8 & -4 \end{bmatrix} = 8 \cdot C_1 \cdot (C_2 + C_3), \tag{44}$$

where $C_3 = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0.25 \\ 0 & 0 & 0 & 0 \\ 0 & 0.25 & 0 & 0 \end{bmatrix}$. In Eq. (45), the upper diagonal matrix $U_{4\times 4\_\mathrm{VC1}}$ for VC1 needs 14 adders and eight shifters.

$$U_{4\times 4\_VC1} = \begin{vmatrix} 12 & 16 & 12 & 6 \\ 12 & 6 & -12 & -16 \\ 12 & -6 & -12 & 16 \\ 12 & -16 & 12 & -6 \end{vmatrix} = 8 \cdot C_1 \cdot (C_4 + C_5 \cdot C_2), \tag{45}$$

where and $C_4 = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0.5 \\ 0 & 0 & 0 & 0 \\ 0 & 0.5 & 0 & 0 \end{bmatrix}$, and $C_5 = \begin{bmatrix} 1.5 & 0 & 0 & 0 \\ 0 & 1.5 & 0 & 0 \\ 0 & 0 & 1.5 & 0 \\ 0 & 0 & 0 & 1.5 \end{bmatrix}$. For HEVC, the $8 \times 8$ transform matrix is acquired by the AVS design in Eq. (44), and the design in Eq. (46) costs 16 adders and 12 shifters.

$$U_{4\times 4_{HEVC}} = \begin{bmatrix} 64 & 83 & 64 & 36 \\ 64 & 36 & -64 & -83 \\ 64 & -36 & -64 & 83 \\ 64 & -83 & 64 & -36 \end{bmatrix} = 2 \cdot C_1 \cdot [32 \cdot (C_2 + C_3) - U_1], \tag{46}$$

where $U_1 = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 2 & 0 & -1.5 \\ 0 & 0 & 0 & 0 \\ 0 & -1.5 & 0 & -2 \end{bmatrix}$. For MPEG-1/2/4, the upper diagonal matrix is factorized by

$$U_{4\times4\_MPEG} = \begin{bmatrix} 362 & 473 & 362 & 196 \\ 362 & 196 & -362 & -473 \\ 362 & -196 & -362 & 473 \\ 362 & -473 & 362 & -196 \end{bmatrix} = C_1 \cdot [256 \cdot (C_4 + C_5 \cdot C_2) - (U_2 + U_3)], \tag{47}$$

where $U_2 = \begin{bmatrix} 22 & 0 & 22 & 0 \\ 0 & 0 & 0 & 0 \\ 22 & 0 & -22 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$,and $U_3 = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 4 & 0 & 39 \\ 0 & 0 & 0 & 0 \\ 0 & 39 & 0 & -4 \end{bmatrix}$. In Eq. (47), the parameter "22" of $U_2$ is implemented by $(C_5 \cdot C_5 \ll 4) - (C_1 \ll 1)$, where "$\ll 1$" is left shifting one bit, and the cost in Eq. (47) requires 28 adders and 26 shifters.

By Eq. (3), on the other side, the down diagonal matrix $D_{4\times4\_AVC}$ for H.264/AVC becomes Eq. (48), and it needs 17 adders and eight shifters.

$$D_{4\times4\_AVC} = \begin{bmatrix} -3 & 6 & -10 & 12 \\ -6 & 12 & -3 & -10 \\ -10 & 3 & 12 & 6 \\ -12 & -10 & -6 & -3 \end{bmatrix} = 8 \cdot U_4 \cdot (D_4 + D_5) \cdot (D_2 + U_3), \tag{48}$$

where $U_4 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & -1 \end{bmatrix}$, $D_4 = \begin{bmatrix} -1 & -1 & 1 & 0 \\ 1 & 0 & 1 & -1 \\ -1 & 1 & 0 & -1 \\ 0 & 1 & 1 & 1 \end{bmatrix}$, $D_5 = \begin{bmatrix} -0.5 & 0 & 0 & 0 \\ 0 & 0 & 0.5 & 0 \\ 0 & 0.5 & 0 & 0 \\ 0 & 0 & 0 & 0.5 \end{bmatrix}$,

$D_2 = \begin{bmatrix} 0.25 & 0 & 0 & 0 \\ 0 & 0.25 & 0 & 0 \\ 0 & 0 & -0.25 & 0 \\ 0 & 0 & 0 & 0.25 \end{bmatrix}$, $U_3 = \begin{bmatrix} 0 & 0 & 0 & -1 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix}$.

For AVS, the $D_{4\times4\_AVS}$ matrix becomes (49), and $D_4$ and $D_5$ are shared with the design in Eq. (48), and then $U_3$ and $U_4$ are also partially shared with the scheme in Eq. (48). In Eq. (49), it costs 24 adders and 12 shifters

$$D_{4\times4\_AVS}=\begin{bmatrix}-2 & 6 & -9 & 10\\ -6 & 10 & -2 & -9\\ -9 & 2 & 10 & 6\\ -10 & -9 & -6 & -2\end{bmatrix}=4\cdot U_4\cdot(D_4+D_5)\cdot D_3\cdot(D_1+U_3), \tag{49}$$

where $U_3=\begin{bmatrix}0 & -1 & 0 & 0\\ 0 & 0 & 0 & -1\\ 1 & 0 & 0 & 0\\ 0 & 0 & 1 & 0\end{bmatrix}$, $D_3=\begin{bmatrix}1 & 0 & 0 & 0\\ 0 & -1 & 0 & 0\\ 0 & 0 & -1 & 0\\ 0 & 0 & 0 & 1\end{bmatrix}$, and $D_1=\begin{bmatrix}1.5 & 0 & 0 & 0\\ 0 & 1.5 & 0 & 0\\ 0 & 0 & -1.5 & 0\\ 0 & 0 & 0 & 1.5\end{bmatrix}$.

For VC-1, the $D_{4\times4\_\text{VC1}}$ matrix is factorized by Eq. (50), and the design requires 21 adders and 12 shifters

$$D_{4\times4\_VC1}=\begin{bmatrix}-4 & 9 & -15 & 16\\ -9 & 16 & -4 & -15\\ -15 & 4 & 16 & 9\\ -16 & -15 & -9 & -4\end{bmatrix}=8\cdot U_4\cdot(D_4\cdot D_6+D_5)\cdot(D_2+U_3), \tag{50}$$

where $D_6=\begin{bmatrix}1.5 & 0 & 0 & 0\\ 0 & 1.5 & 0 & 0\\ 0 & 0 & 1.5 & 0\\ 0 & 0 & 0 & 1.5\end{bmatrix}$. For HEVC, the $D_{4\times4\_\text{HEVC}}$ matrix is expressed by Eq. (51), and it expends 44 adders and 20 shifters

$$D_{4\times4\_HEVC}=\begin{bmatrix}-18 & 50 & -75 & 89\\ -50 & 89 & -18 & -75\\ -75 & 18 & 89 & 50\\ -89 & -75 & -50 & -18\end{bmatrix}=D_{4\times4\_AVS}\cdot9+[4\cdot(U_5\cdot D_1+U_6)-U_7], \tag{51}$$

where $U_5=\begin{bmatrix}0 & 0 & -1 & 0\\ 0 & 0 & 0 & 1\\ 1 & 0 & 0 & 0\\ 0 & 1 & 0 & 0\end{bmatrix}$, $U_6=\begin{bmatrix}0 & -1 & 0 & 0\\ 1 & 0 & 0 & 0\\ 0 & 0 & 0 & -1\\ 0 & 0 & 1 & 0\end{bmatrix}$, $U_7=\begin{bmatrix}0 & 0 & 0 & 1\\ 0 & 1 & 0 & 0\\ 0 & 0 & 1 & 0\\ -1 & 0 & 0 & 0\end{bmatrix}$. For MPEG-1/2/4, based on $D_{4\times4\_\text{AVS}}$, the $D_{4\times4\_\text{MPEG}}$ matrix is presented by Eq. (52), and the design costs 48 adders and 32 shifters

$$D_{4\times4\_MPEG} = \begin{bmatrix} -100 & 284 & -426 & 502 \\ -284 & 502 & -100 & -426 \\ -426 & 100 & 502 & 284 \\ -502 & -426 & -284 & -100 \end{bmatrix} = D_{4\times4\_AVS} \cdot 50 + [16 \cdot (U_5 \cdot D_1 + U_6) + 2 \cdot U_7]. \tag{52}$$

### 4.2. Hardware-sharing design for 4 × 4 transforms mode

For AVS-M, the matrix $M_{4\times4\_\mathrm{AVS}}$ is presented by (53), and it spends 10 adders and six shifters

$$M_{4\times4\_AVS} = \begin{bmatrix} 2 & 3 & 2 & 1 \\ 2 & 1 & -2 & -3 \\ 2 & -1 & -2 & 3 \\ 2 & -3 & 2 & -1 \end{bmatrix} = C_1 \cdot (2 \cdot C_2 + U_8), \tag{53}$$

where $U_8 = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix}$. For VC-1, $M_{4\times4\_\mathrm{VC1}}$ is expressed by Eq. (54), and the design requires 14 adders and 12 shifters

$$M_{4\times4\_VC1} = \begin{bmatrix} 17 & 22 & 17 & 10 \\ 17 & 10 & -17 & -22 \\ 17 & -10 & -17 & 22 \\ 17 & -22 & 17 & -10 \end{bmatrix} = C_1 \cdot (16 \cdot C_2 + U_9), \tag{54}$$

where $U_9 = \begin{bmatrix} 1 & 0 & 1 & 0 \\ 0 & -2 & 0 & 6 \\ 1 & 0 & -1 & 0 \\ 0 & 6 & 0 & 2 \end{bmatrix}$. For VP8, all coefficients in 4 × 4 transform matrix are multiplied by 128 to get integer values, and it costs 18 adders and 14 shifters

$$M_{4\times4\_VP8} = \begin{bmatrix} 128 & 167 & 128 & 70 \\ 128 & 70 & -128 & -167 \\ 128 & -70 & -128 & 167 \\ 128 & -167 & 128 & -70 \end{bmatrix} = C_1 \cdot (128 \cdot C_2 + U_{10}), \tag{55}$$

http://dx.doi.org/10.5772/64985

where $U_{10} = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & -6 & 0 & 39 \\ 0 & 0 & 0 & 0 \\ 0 & 39 & 0 & 6 \end{bmatrix}$. The matrix $U_{4\times4\_AVC}/8$ equals the 4 × 4 inverse transform matrix in H. 264/AVC. In addition, the matrix $U_{4\times4\_HEVC}$ equals the 4 × 4 inverse transform matrix in HEVC. Thus, several multiplexers are used to share the hardware between the submatrices to decrease hardware cost.

### 4.3. Architecture comparison

The applied hardware-sharing-based 1D multistandard inverse integer transform scheme has two inputs, which sustain 4 × 4 and 8 × 8 transform modes. The hardware blocks of processing the 4 × 4 inverse transforms are shared with that of the upper diagonal matrix $U_{8\times8}$. Thus, several multiplexers are utilized for $U_{8\times8}$ to compute the 4 × 4 inverse transforms without additional operations. For the multistandard applications, the hardware-sharing architecture of the fast 1D 4 × 4 and 8 × 8 inverse integer transforms is illustrated in [41]. The shifters are also realized by wiring. Compared with the individual designs without hardware shares, **Table 4** depicts that the used scheme in [41] decreases the number of shifters and adders by 50 and 75%, respectively.

| Different 1D inverse integer transform modes | No. of adders | No. of shifters |
|---|---|---|
| Individual designs without hardware shares | 336 | 180 |
| Hardware-sharing-based design in Section 4 | 82 | 90 |
| **Reduction of cost** | **75%** | **50%** |

**Table 4.** Hardware comparison between two architectures [41].

To implement the discussed architecture, a cell-based VLSI design flow is utilized to design, simulate, and verify the cost-effective hardware-sharing architecture. For fair comparisons among different transform structures, the normalized mode gain, which is required to normalize the gate counts, is described as follows: By matrix dimensions and without missing generality [40], the normalized mode gains defined for the 32 × 32, 16 × 16, 8 × 8, and 4 × 4 inverse integer transform matrices are 16, 4, 1, and 1/4, respectively.

The hardware-sharing-based design in Section 3 supports 4 × 4, 8 × 8, 16 × 16, and 32 × 32 inverse transform modes for HEVC. Thus, the normalized mode gain of the design is 21.25 (i.e., 16 + 4 + 1 + 0.25). Similarly, five 8 × 8 and five 4 × 4 inverse transform functions are provided by the hardware-shared design in Section 4. Therefore, the normalized mode gain is assigned by 6.25 (i.e., 5 + 1.25) [41]. Afterwards, the normalized gate counts are defined by [40, 41]

$$Normalized\ gate\ counts = {}^{Gate\ counts}\!\Big/_{Normalized\ mode\ gain}. \tag{56}$$

**Table 5** shows the hardware cost comparisons among different 1D multiple transform architectures, which includes single-standard multiple-mode [32] and multiple-standard multiple-mode [41] transform designs.

| Architecture | Ahmed et al. [29] | Hardware-sharing based-design in Section 3 | Shen et. al. [26] | Martuza et. al. [28] | Qi et al. [36] | Wang et al. [38] | Hardware-sharing-based design in Section 4 |
|---|---|---|---|---|---|---|---|
| Gate counts | 144.8K | 115.7 K | 134.8 K | 39.4 K | 18 K | 23.06 K | 27.4 K |
| Normalized mode gain | 21.25 | 21.25 | 25.75 | 5 | 3.5 | 4.5 | 6.25 |
| Normalized gate counts | 6.81 K | 5.44 K | 5.23 K | 7.88 K | 5.14 K | 5.12 K | 4.38 K |
| Supporting modes | Single-standard Multiple-mode | Single-standard Multiple-mode | Multiple-standard Multiple-mode | Multiple-standard Multiple-mode | Multiple-standard Multiple-mode | Multiple-standard Multiple-mode | Multiple-standard Multiple-mode |
| Supporting standards/ Transforms | **HEVC**: 4 × 4, 8 × 8, 16 × 16, 32 × 32 modes | **HEVC**: 4 × 4, 8 × 8, 16 × 16, 32 × 32 modes | **H.264/AVC, VC-1**: 4 × 4,8 × 8 modes **MPEG-1/2/4, AVS**: 8 × 8 mode; **HEVC**: 4 × 4, 8 × 8, 16 × 16, 32 × 32 modes | **H.264/ AVC, VC-1, AVS, HEVC**: 4 × 4, 8 × 8 modes | **H.264/AVC, VC-1**: 4 × 4, 8 × 8 modes; **MPEG-1/2/4**: 8 × 8 mode | **H.264/ AVC;, VC-1**: 4 × 4, 8 × 8 modes; **MPEG-1/2/4, AVS**: 8 × 8 mode | **H.264/AVC, VC-1, HEVC**: 4 × 4, 8 × 8 modes; **MPEG-1/2/4, AVS**: 8 × 8 mode; **VP8, AVS-M**: 4 × 4 mode |

**Table 5.** Hardware cost comparisons among different 1D multiple transform architectures [32, 41].

## 5. Conclusion

For the single-standard multiple-mode transform design, this chapter discussed the 4 × 4, 8 × 8, 16 × 16, and 32 × 32 inverse core transforms in HEVC with a cost-effective and hardware-efficient design. By the symmetrical characteristics of the elements, the core transform matrices were factorized into several submatrices. Thus, the hardware of the $(N/2) \times (N/2)$ inverse core transform was shared with that of the $N \times N$ inverse core transform for $N = 32$, 16, and 8. Compared with the direct design without hardware shares, the applied transform scheme in Section 3 decreased the hardware cost of adders and shifters by 32 and 36%, respectively. Besides, for VLSI implementation, the design in Section 3 requires less normalized gate counts than the design does in [29].

For the multiple-standard multiple-mode transform design, this chapter also discussed the fast algorithm and hardware-sharing-based design of 4 × 4 and/or 8 × 8 inverse transforms among H.264/AVC, VC-1, HEVC, MPEG-1/2/4, AVS, and VP8 for multistandard video decoders. By only shifters and adders, the decomposition scheme of matrices was used to develop the hardware-shared scheme. The used structure in Section 4 decreased the number of shifters and adders by 50 and 75% more than the individual fast algorithm-based implementation did. Besides, for VLSI implementation, the design in Section 4 requires less normalized gate counts than the designs do in [26, 28, 36, 38].

## Acknowledgements

This work was supported by Ministry of Science and Technology, Taiwan, R.O.C. under Grant MOST 105-2221-E-005-078.

## Author details

Chih-Peng Fan

Address all correspondence to: cpfan@dragon.nchu.edu.tw

Department of Electrical Engineering, National Chung Hsing University, Taichung, Taiwan, ROC

## References

[1] J. R. Rao and P. Yip, Discrete Cosine Transform: Algorithms, Advantage, Applications, New York, NY: Academic, 1990.

[2] ISO/IEC JTC 1/SC 29/WG 1—Coding of Still Pictures, 2009.

[3] ISO/IEC 11172-2 MPEG-1 Video Coding Standard, Information Technology—Coding of Moving Pictures and Associated Audio for Digital Storage Media at up to about 1,5 Mbit/s – Part 2: Video, 1993.

[4] ISO/IEC 13818-2 MPEG-2 Video Coding Standard, Information Technology—Generic Coding of Moving Pictures and Associated Audio Information: Video, 1995.

[5] ISO/IEC 14496-2 MPEG-4 Video Coding Standard, Information Technology—Coding of Audio-Visual Objects – Part 2: Visual, 2004.

[6] T. Wiegand and G. Sullivan, Draft ITU-T Recommendation and Final Draft International Standard of Joint Video Specification, (ITU-T rec. H.264/ISO/IEC 14496-10 AVC, presented at Joint Video Team (JVC) of ISO/IEC MPEG and ITU-T VCEG), 2003.

[7] Iain E. G. Richardson, H.264 and MPEG-4 Video Compression—Video Coding for Next-generation Multimedia, John Wiley & Sons, 111 River Street, Hoboken NJ07030-5774, New Jersey, United States, 2003.

[8] W. Gao, C. Reader, F. Wu, Y. He, L. Yu, H. Lu, S. Yang, T. Huang, and X. Pan, AVS—The Chinese Next-Generation Video Coding Standard, National Association of Broadcasters (NAB) Conference, 2004.

[9] L. Yu, S. Chen, and J. Wang, Overview of AVS video coding standards, Signal Processing: Image Communication, vol. 24, issue 4, pp. 247–262, April 2009.

[10] SMPTE, Standard for Television: VC-1 Compressed Video Bitstream Format and Decoding Process, SMPTE 421M-2006.

[11] J. Bankoski, P. Wilkins, and Y. Xu, Technical overview of VP8, an open source video codec for the web, IEEE International Conference on Multimedia and Expo (ICME), pp. 1–6, July 11–15, 2011.

[12] M. T. Pourazad, C. Doutre, M. Azimi, and P. Nasiopoulos, HEVC: the new gold standard for video compression: How does HEVC compare with H.264/AVC ?, IEEE Consumer Electronics Magazine, vol. 1, pp. 36–46, July 2012.

[13] H. S. Malvar, A. Hallapuro, M. Karczewicz, and L. Kerofsky, Low-complexity transform and quantization in H.264/AVC, IEEE Transactions on Circuits and Systems for Video Technology, vol. 13, no. 7, pp. 598–603, July 2003.

[14] S. Gordon, D. Marple, and T. Wiegand, Simplified use of 8x8 transforms—updated proposal and results, JVT-K028, 11th Meeting, Munich, Germany, March 2004.

[15] S. Srinivasan, P. Hsu, T. Holcomb, K. Mukerjee, S. L. Regunathan, B. Lin, J. Liang, M. C. Lee, and J. Ribas-Corbera, Windows media video 9: overview and applications, Signal Processing: Image Communication, vol. 19, issue 9, pp. 851–875, October 2004.

[16] S. Srinivasan and S. L. Regunathan, An overview of VC-1, Proceedings of the SPIE, Visual Communications and Image Processing (VCIP), Beijing, China, vol. 5960, pp. 720–728, July 2005.

[17] T. C. Wang, Y. W. Huang, H. C. Fang, and L. G. Chen, Parallel 4x4 2D transform and inverse transform architecture for MPEG-4 AVC/H.264, IEEE International Symposium on Circuits and Systems, vol. 2, pp. 800–803, 2003.

[18] Z. Y. Cheng, C. H. Chen, B. D. Liu, and J. F. Yang, High throughput 2-D transform architectures for H.264 advanced video coders, IEEE Asia-Pacific Conference on Circuits and Systems, pp. 1141–1144, December 2004.

[19] K. H. Chen, J. I. Guo, and J. S. Wang, A high-performance direct 2-D transform coding IP design for MPEG-4 AVC/H.264, IEEE Transactions on Circuits and Systems for Video Technology, vol. 16, no. 4, pp. 472–483, April 2006.

[20] G. A. Su and C. P. Fan, Cost effective hardware sharing architecture for fast 1-D 8x8 forward and inverse integer transforms of H.264/AVC high profile, IEEE Asia Pacific Conference on Circuits and Systems, pp. 1332–1335, November 2008.

[21] T. T. T. Do and T. M. Le, High throughput area-efficient SoC-based forward/inverse integer transform for H.264/AVC, IEEE International Symposium on Circuits and Systems, pp. 4113–4116, May 2010.

[22] W. Hwangbo and C. M. Kyung, A multi-transform architecture for H.264/AVC high-profile coders, IEEE Transactions on Multimedia, vol. 12, no. 3, pp. 157–167, April 2010.

[23] M. L. Hsia and Oscal T. C. Chen, Low-complexity inverse integer transform in H.264/AVC, IEEE International Conference on Multimedia & Expo, pp. 826–830, July 2010.

[24] M. Nadeem, S. Wong, and G. Kuzmanov, Inverse integer transform in H.264/AVC intra-frame encoder, Sixth IEEE International Symposium on Electronic Design, Test and Application, pp. 228–233, 2011.

[25] R. Jeske, J. C. de Souza, G. Wrege, R. Conceicao, M. Grellert, J. Mattos, and L. Agostini, Low cost and high throughput multiplierless design of a 16 point 1-D DCT of the new HEVC video coding standard, Conference on Programmable Logic (SPL), pp. 1–6, March 2012

[26] S. Shen, W. Shen, Y. Fan, and Xiaoyang Zeng, A unified 4/8/16/32-point integer IDCT architecture for multiple video coding standards, IEEE International Conference on Multimedia and Expo (ICME), pp. 788–793, July 2012.

[27] W. Zhao, T. Onoye, and T. Song, High-performance multiplierless transform architecture for HEVC, IEEE International Symposium on Circuits and Systems (ISCAS), pp. 1668–1671, 2013.

[28] M. Martuza, K. A. Wahid, Implementation of a cost shared transform architecture for multiple video codecs, Journal of Real-Time Image Processing, vol. 10, no. 1, pp. 151–162, March 2015.

[29] A. Ahmed, M. U. Shahid, and A. Rehman, N point DCT VLSI architecture for emerging HEVC standard, VLSI Design, volume 2012, Article ID 752024, pp. 1–13, 2012.

[30] Joint Collaborative Team—Video Coding, CE10: Core transform design for HEVC, JCTVC-G495, Geneva, Switzerland, 21–30, November 2011.

[31] M. N. Haggag, M. El-Sharkawy, and G. Fahmy, Efficient fast multiplication-free integer transformation for the 2-D DCT H.265 standard, IEEE International Conference on Image Processing, pp. 3769–3772, September 2010.

[32] C. W. Chang, H. F. Hsu, C. P. Fan, C. B. Wu, and Robert C. H. Chang, A fast algorithm-based cost-effective and hardware-efficient unified architecture design of 4×4, 8×8, 16×16, and 32×32 inverse core transforms for HEVC, Journal of Signal Processing Systems, vol. 82, no. 1, pp. 69–89, 2016.

[33] S. Lee and K. Cho, Architecture of transform circuit for video decoder supporting multiple standards, Electronics Letters, vol. 44, no. 4, pp. 274–275, February 2008.

[34] C. P. Fan and G. A. Su, Efficient low cost sharing design of fast 1-D inverse integer transform algorithms for H.264/AVC and VC-1, IEEE Signal Processing Letters, vol. 15, pp. 926–929, December 2008.

[35] G. A. Su and C. P. Fan, Low-cost hardware sharing architecture of fast 1-D inverse transforms for H.264/AVC and AVS applications, IEEE Transactions on Circuits and Systems, Part II, vol. 55, no. 12, pp. 1249–1253, December 2008.

[36] H. Qi, Q. Huang, and W. Gao, A low-cost very large scale integration architecture for multistandard inverse transform, IEEE Transactions on Circuits and Systems, Part II, vol. 57, no. 7, pp. 551–555, July 2010.

[37] Y. K. Lai and Y. F. Lai, A Reconfigurable IDCT architecture for universal video decoders, IEEE Transactions on Consumer Electronics, vol. 56, no. 3, pp. 1872–1879, August 2010.

[38] K. Wang, J. Chen, W. Cao, Y. Wang, L. Wang, and J. Tong, A reconfigurable multi-transform VLSI architecture supporting video codec design, IEEE Transactions on Circuits and Systems II: Express Briefs, vol. 58, no. 7, pp. 432–436, July 2011.

[39] K. Wahid, M. Martuza, M. Das, and C. McCrosky, Resource shared architecture of multiple transforms for multiple video codecs, 24th Canadian Conference on Electrical and Computer Engineering (CCECE), pp. 000947–000950, 2011.

[40] C. P. Fan, C. W. Chang, and S. J. Hsu, Cost effective hardware sharing design of fast algorithm based multiple forward and inverse transforms for H.264/AVC, MPEG-1/2/4, AVS, and VC-1 video encoding and decoding applications, IEEE Transactions on Circuits and Systems for Video Technology, vol. 24, no. 4, pp. 714–720, April 2014.

[41] C. W. Chang, H. F. Hsu, and C. P. Fan, High-efficiency multiple 4x4 and 8x8 inverse transform design with a cost-effective unified architecture for multistandard video decoders, 2014 IEEE Asia Pacific Conference on Circuits & Systems, Okinawa, Japan, pp. 507–510, November 2014.

[42] C. W. Chang, Fast algorithm based cost-effective and hardware-sharing architecture designs of multiple-mode discrete integer transforms for multi-standard video Codecs, Ph.D. dissertation, National Chung Hsing University, Taiwan, 2015.

[43] http://en.wikipedia.org/wiki/Kronecker_product