Research Journal of Information Technology

ISSN 1815-7432

Academic Journals Inc.

www.academicjournals.com

Research Journal of Information Technology 6 (1): 27-45, 2014
ISSN 1815-7432 / DOI: 10.3923/rjit.2014.27.45

Low Complexity and Low-Cost Hardware Sharing Design of Fast Multistandard 2-D DCT/IDCT for Image/Video Coding

[1]Mohamed El Aakif, [2]Said Belkouch and [1]Moha M'rabet Hassani
[1]Laboratory of Electronic and Instrumentation, Faculty of Science Semlalia, Cadi Ayyad University, Marrakech, Morocco
[2]Department of Electrical Engineering, ENSAM, Cadi Ayyad University, Marrakech, Morocco

Corresponidng Author: Mohamed El Aakif, Laboratory of Electronic and Instrumentation, Faculty of Science Semlalia, Cadi Ayyad University, Marrakech, Morocco

ABSTRACT

The multistandard video and image codec using a single platform is the recent trend in multimedia technologies, in which the Two-Dimensional forward and inverse Discrete Cosine Transform (2-D DCT/IDCT) is used as a transform core. Therefore, the 2-D DCT/IDCT core should support multiple transforms and that to be implemented with low cost hardware, while keeping the required performances for real time applications. In this study, a fast multistandard DCT/IDCT generalized algorithms are developed with low complexity for 2×2, 4×4 and 8×8 forward/inverse transforms in H.264/Advanced Video Coding (AVC), 8×8 forward/inverse transform in Audio Video Coding Standard (AVS), 4×4 and 8×8 forward/inverse transforms in VC-1 and 8×8 forward/inverse discrete cosine transform in JPEG and MPEG-1/2/4. The hardware implementation of these generalized algorithms is performed with low cost hardware, in which the hardware sharing design and shared factorization of coefficient multiplications are used. Furthermore the multiplications are avoided by using additions and shifts. The multistandard 2-D DCT/IDCT is achieved with the proposed 1-D DCT/IDCT sharing architecture and a robust transpose buffer. These pipelined hardware architectures are described in VHDL hardware language, synthesized, verified and implemented on low cost FPGA. The implementation results of the proposed multistandard 2-D DCT/IDCT show a decrease of computational complexity by 29.7% of additions while the number of shifts remains almost the same and an increase of maximum frequency and throughput by 18.2% in comparison with other previous design.

Key words: Multistandard DCT/IDCT, fast algorithm, multiplierless, hardware sharing design, shared factorization

INTRODUCTION

Actually, several standards of image and video coding use the forward and inverse discrete cosine transform (DCT/IDCT) as the key component in image and video compression. The transform coding is an efficient technique for energy compaction in image and video coding. Indeed, the DCT is used to transform image from the spatial domain to the frequency domain which leads to remove the spatial correlation. The IDCT is used to perform the inverse operation of DCT by decoding the data in frequency domain. The 8×8 2-D DCT/IDCT is employed in JPEG (ISO and IEC, 2009), MPEG-1/2 (ISO and IEC, 1993; Video Coding Standard, 1995) and MPEG-4 standard (ISO/IEC 14496-2, 2004). The H.264/AVC standard (Wiegand *et al*., 2003) uses 4×4 and 8×8 integer transforms, the 4×4 integer transform is used to perform fast data compression, while the 8×8 integer transform is employed to achieve better energy compaction in high quality video

(Malvar *et al*., 2003; Wien, 2003). For these reasons, the high profile of H.264/AVC Fidelity Range Extension (FREXt) leads to select an adaptive mode amongst 4×4 and 8×8 transforms (Marpe *et al*., 2005). The VC-1 standard (SMPTE Standard, 2006) uses 4×4 and 8×8 integer transforms for the video coding, this video standard is developed by Microsoft Corporation and standardized by the Society of Motion Picture and Television Engineers (SMPTE). The AVS video standard is developed by China Audio Video Coding Standard Working Group (Gao *et al*., 2004; Yu *et al*., 2009), in which an 8×8 integer transform is used to achieve high efficiency and to be applied to code HDTV data. Nowadays, the trend of the mobile devices is to have different functions such as Video on Demand (VOD), Digital Multimedia Broadcasting (DMB), Portable Multimedia Player (PMP), cell phone, camera and so on. Due to this reason, it is necessary to support the extensively used video compression standards in a single system-on-chip (SoC) platform.

In recent years, there is a growing interest to develop multistandard DCT/IDCT architectures for advanced multimedia applications. Therefore, the hardware implementation of multiple transforms into a single chip increases the area and power consumption which has a negative impact on the overall system. Hence, the objective is to find a suitable implementation method of multiple transforms that achieves high performances with low cost hardware. The circuit sharing is an efficient method for the hardware cost reduction, so that the area of the integrated multitransform to be smaller than the total areas of different single transforms.

The multiple transforms that support the JPEG, MPEG, VC-1 and H.264 video decoder are proposed by Lee and Cho (2008). Fan and Su (2008) and Su and Fan (2008) presented respectively hardware sharing architectures between H.264/AVC and VC-1 and between H.264/AVC and AVS, in which the inverse integer transform matrices are decomposed by using the sparse matrix factorizations. A high parallel architecture for all transforms of H.264/AVC is proposed by Li *et al*. (2008), where the matrix decomposition is used in inverse transform architecture for circuit saving. Huang *et al*. (2008) proposed the 2-D transform architecture with a unique kernel to support H.264/AVC, JPEG and MPEG-1/2/4. A flexible transform processor design is proposed by Park *et al*. (2006) to achieve IDCT in MPEG and integer transform in H.264/AVC. Qi *et al*. (2010) proposed the multistandard inverse transforms for MPEG-2, MPEG-4 ASP, H.264/AVC and VC-1 video decoder. Unlike previous designs, a multiple inverse transforms design that support JPEG, MPEG-1/2/4, H.264/AVC, AVS and VC-1 standard video coding is presented by Fan *et al*. (2011).

In this study, the proposed hardware architecture of fast DCT/IDCT supports multistandard video coding as Fan *et al*. (2011). This architecture is proposed with low computational complexity; it also supports three types of variable-sized transform such as 2×2, 4×4 and 8×8 transform. Indeed, we have used matrix decomposition method to develop the generalized algorithms for multistandard DCT/IDCT. Moreover, we have reduced as maximum possible the number of multiplications. The shared factorization method is used to implement these multiplications with a minimum number of shifts and additions. Our architecture of multistandard DCT/IDCT is based on hardware sharing design, while providing the flexibility for adding other standard video. The Altera Cyclone II FPGA has been used to perform our pipelined hardware implementations.

REVIEWS OF DCT/IDCT ALGORITHMS

4×4 transform algorithms: The 4×4 forward and inverse transforms are defined respectively in Eq. 1 and 2 (Hwangbo and Kyung, 2010) as:

$$W = C_{f4} \cdot X \cdot C_{f4}^{T} \tag{1}$$

$$X' = C_{i4}^{T} \cdot W' \cdot C_{i4} \tag{2}$$

where, X is a 4×4 residual block input to the forward transform and W' is an inversely quantized 4×4 block input to the inverse transform.

The 1-D 4×4 transform matrices C_{f4_VC-1} and C_{i4_VC-1} of VC-1 standard are given as:

$$C_{f4\_VC-1} = C_{i4\_VC-1} = \begin{pmatrix} 17 & 17 & 17 & 17 \\ 22 & 10 & -10 & -22 \\ 17 & -17 & -17 & 17 \\ 10 & -22 & 22 & -10 \end{pmatrix} \tag{3}$$

In the H.264/AVC standard, the 1-D 4×4 transform matrices C_{f4_AVC} and C_{i4_AVC} are given as:

$$C_{f4\_AVC} = \begin{pmatrix} 1 & 1 & 1 & 1 \\ 2 & 1 & -1 & -2 \\ 1 & -1 & -1 & 1 \\ 1 & -2 & 2 & -1 \end{pmatrix} \tag{4}$$

$$C_{i4\_AVC} = \begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & \frac{1}{2} & -\frac{1}{2} & -1 \\ 1 & -1 & -1 & 1 \\ \frac{1}{2} & -1 & 1 & -\frac{1}{2} \end{pmatrix} \tag{5}$$

The 4×4 forward and inverse Hadamard transforms are defined in Eq. 6 and 7 as:

$$Y_D = H_4 \cdot W_D \cdot H_4^{T}/2 \tag{6}$$

$$W_{QD} = H_4^{T} \cdot Z_D \cdot H_4 \tag{7}$$

where, W_D is the 4×4 block containing DC components for each of the 16 4×4 submacroblocks and Z_D is an inversely quantized 4×4 DC block. The transform matrix H_4 is given as:

$$H_4 = \begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 \\ 1 & -1 & 1 & -1 \end{pmatrix} \tag{8}$$

The 2×2 forward and inverse Hadamard transforms are defined in Eq. 9 and 10 as:

$$Y_D = H_2 \cdot W_D \cdot H_2^T \tag{9}$$

$$W_{QD} = H_2^T \cdot Z_D \cdot H_2 \tag{10}$$

where, W_D is the block of 2×2 DC chroma coefficients and Y_D is the block after transformation, Z_D is an inversely quantized 2×2 DC chroma block. The transform matrix H_2 is given as:

$$H_2 = \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix} \tag{11}$$

8×8 transform algorithms: The 8×8 forward and inverse transforms are defined respectively in Eq. 12 and 13 (Hwangbo and Kyung, 2010) as:

$$W = C_8 \cdot X \cdot C_8^T \tag{12}$$

$$X' = C_8^T \cdot W' \cdot C_8 \tag{13}$$

where, X is a 8×8 residual block input to the forward transform and W' is an inversely quantized 8×8 block input to the inverse transform.

In the H.264/AVC standard, the 1-D 8×8 integer transform matrix C_{8_AVC} is given as:

$$C_{8\_AVC} = \begin{pmatrix} 8 & 8 & 8 & 8 & 8 & 8 & 8 & 8 \\ 12 & 10 & 6 & 3 & -3 & -6 & -10 & -12 \\ 8 & 4 & -4 & -8 & -8 & -4 & 4 & 8 \\ 10 & -3 & -12 & -6 & 6 & 12 & 3 & -10 \\ 8 & -8 & -8 & 8 & 8 & -8 & -8 & 8 \\ 6 & -12 & 3 & 10 & -10 & -3 & 12 & -6 \\ 4 & -8 & 8 & -4 & -4 & 8 & -8 & 4 \\ 3 & -6 & 10 & -12 & 12 & -10 & 6 & -3 \end{pmatrix} \tag{14}$$

The 1-D 8×8 integer transform matrix C_{8_AVS} of AVS standard is given as:

$$C_{8\_AVS} = \begin{pmatrix} 8 & 8 & 8 & 8 & 8 & 8 & 8 & 8 \\ 10 & 9 & 6 & 2 & -2 & -6 & -9 & -10 \\ 10 & 4 & -4 & -10 & -10 & -4 & 4 & 10 \\ 9 & -2 & -10 & -6 & 6 & 10 & 2 & -9 \\ 8 & -8 & -8 & 8 & 8 & -8 & -8 & 8 \\ 6 & -10 & 2 & 9 & -9 & -2 & 10 & -6 \\ 4 & -10 & 10 & -4 & -4 & 10 & -10 & 4 \\ 2 & -6 & 9 & -10 & 10 & -9 & 6 & -2 \end{pmatrix} \tag{15}$$

In AVS standard, the 1-D 8×8 integer transform matrix $C_{8\_VC\text{-}1}$ is given as:

$$C_{8\_AVS} = \begin{pmatrix} 12 & 12 & 12 & 12 & 12 & 12 & 12 & 12 \\ 16 & 15 & 9 & 4 & -4 & -9 & -15 & -16 \\ 16 & 6 & -6 & -16 & -16 & -6 & 6 & 16 \\ 15 & -4 & -16 & -9 & 9 & 16 & 4 & -15 \\ 12 & -12 & -12 & 12 & 12 & -12 & -12 & 12 \\ 9 & -16 & 4 & 15 & -15 & -4 & 16 & -9 \\ 6 & -16 & 16 & -6 & -6 & 16 & -16 & 6 \\ 4 & -9 & 15 & -16 & 16 & -15 & 9 & -4 \end{pmatrix} \tag{16}$$

According to Lee and Cho (2008), the 1-D 8×8 integer transform matrix $C_{8\_MPEG1/2/4}$ of JPEG and MPEG1/2/4 standards is given as:

$$C_{8\_MPEG1/2/4} = \begin{pmatrix} 362 & 362 & 362 & 362 & 362 & 362 & 362 & 362 \\ 502 & 426 & 284 & 100 & -100 & -284 & -426 & -502 \\ 473 & 196 & -196 & -473 & -473 & -196 & 196 & 473 \\ 426 & -100 & -502 & -284 & 284 & 502 & 100 & -426 \\ 362 & -362 & -362 & 362 & 362 & -362 & -362 & 362 \\ 284 & -502 & 100 & 426 & -426 & -100 & 502 & -284 \\ 196 & -473 & 473 & -196 & -196 & 473 & -473 & 196 \\ 100 & -284 & 426 & -502 & 502 & -426 & 284 & -100 \end{pmatrix} \tag{17}$$

PROPOSED GENERALIZED ALGORITHMS FOR MULTISTANDARD DCT/IDCT

Proposed 1-D 4×4 DCT/IDCT generalized algorithm: In Eq. 1 and 2, the 1-D 4×4 DCT matrices C_{f4}^T and H_4^T and IDCT matrices C_{i4} and H_4 can be respectively expressed as follows:

$$C_{f4}^T = T_4,\ H_4^T = T_4 \tag{18}$$

$$C_{i4} = T_4^T,\ H_4 = T_4^T \tag{19}$$

Where:

$$T_4 = \begin{pmatrix} a & f & a & g \\ a & g & -a & -f \\ a & -g & -a & f \\ a & -f & a & -g \end{pmatrix} \tag{20}$$

where a, f and g are the coefficients in DCT/IDCT matrix for different transform standards. Table 1 presents these coefficients of T_4 matrix.

Table 1: The 4×4 DCT and IDCT coefficients for different coding standards

	DCT			IDCT		
Variables	H.264/AVC	Hadamard	VC-1	H.264/AVC	Hadamard	VC-1
a	1	1	17	1	1	17
f	2	1	22	1	1	22
g	1	1	10	1/2	1	10

In Eq. 20, the T_4 matrix can be decomposed as the product of 4 matrices expressed as follows:

$$T_4 = A_4 \cdot B_4 \cdot C_4 \cdot D_4 \tag{21}$$

$$T_4^T = D_4^T \cdot C_4^T \cdot B_4^T \cdot A_4^T \tag{22}$$

Where:

$$A_4 = \begin{pmatrix} 1 & 0 & 0 & 1 \\ 0 & 1 & 1 & 0 \\ 0 & 1 & -1 & 0 \\ 1 & 0 & 0 & -1 \end{pmatrix}, B_4 = \begin{pmatrix} 1 & 1 & 0 & 0 \\ 1 & -1 & 0 & 0 \\ 0 & 0 & g & -f \\ 0 & 0 & f & g \end{pmatrix}, C_4 = \begin{pmatrix} a & 0 & 0 & 0 \\ 0 & a & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}, D_4 = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{23}$$

Let B_{42} the 2×2 submatrix of B_4 given in Eq. 24:

$$B_{42} = \begin{pmatrix} g & -f \\ f & g \end{pmatrix} \tag{24}$$

The hardware implementation of B_{42} requires 4 multiplications and 2 additions as shown below:

$$\begin{cases} O_0 = g \cdot I_0 + f \cdot I_1 \\ O_1 = -f \cdot I_0 + g \cdot I_1 \end{cases} \tag{25}$$

where, I_0 and I_1 are the inputs, O_0 and O_1 are the outputs.

We can rewrite Eq. 25 as follows:

$$\begin{cases} O_0 = f \cdot (I_0 + I_1) - (f - g) \cdot I_0 \\ O_1 = (f + g) \cdot I_1 - f \cdot (I_0 + I_1) \end{cases} \tag{26}$$

Unlike Eq. 25, the hardware implementation of B_{42} based on Eq. 26 requires only 3 multiplications and 3 additions. According to Eq. 24, B_{42}^T is expressed as follows:

$$B_{42}^T = \begin{pmatrix} g & f \\ -f & g \end{pmatrix} \tag{27}$$

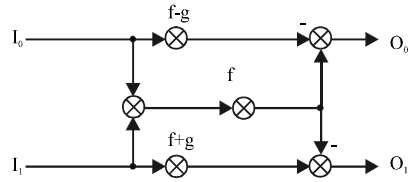

Fig. 1: The FGA of B_{42}

In the same way of B_{42}, the hardware implementation of B_{42}^T requires just 3 multiplications and 3 additions as shown below:

$$\begin{cases} O_0 = (f+g)\cdot I_0 - f\cdot(I_0+I_1) \\ O_1 = f\cdot(I_0+I_1) - (f-g)\cdot I_1 \end{cases} \tag{28}$$

In this study, we use just the unsigned coefficient multiplication. Therefore, we have used in Eq. 26 and 28 (f-g) instead of (g-f), because (f-g≥0) in different video standards. Figure 1 and 2 show respectively the hardware implementation of B_{42} and B_{42}^T as the Flow-Graph Algorithm (FGA).

We need 16 multiplications and 12 additions for the hardware implementation of T_4, when there is no matrix decomposition. Whereas, we need just 5 multiplications and 9 additions if we use the matrix decomposition indicated in Eq. 21. Indeed, according to Eq. 23, A_4 requires 4 additions, C_4 requires 2 multiplications, B_4 needs just 5 additions and 3 multiplications thanks to Eq. 26, while D_4 do not require any operation which only permutes the outputs. In the same way, the hardware implementation of T_4^T requires just 5 multiplications and 9 additions.

Proposed 1-D 8×8 DCT/IDCT generalized algorithm: In Eq. 12 and 13, the 1-D 8×8 DCT matrix C_8^T and IDCT matrix C_8 can be respectively expressed as follows:

$$C_8^T = T_8,\ C_8 = T_8^T \tag{29}$$

Where:

$$T_8 = \begin{pmatrix} a & b & f & c & a & d & g & e \\ a & c & g & -e & -a & -b & -f & -d \\ a & d & -g & -b & -a & e & f & c \\ a & e & -f & -d & a & c & -g & -b \\ a & -e & -f & d & a & -c & -g & b \\ a & -d & -g & b & -a & -e & f & -c \\ a & -c & g & e & -a & b & -f & d \\ a & -b & f & -c & a & -d & g & -e \end{pmatrix} \tag{30}$$

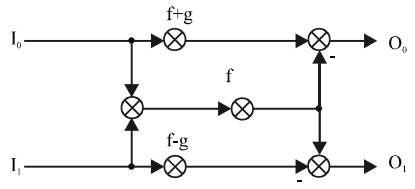

Fig. 2: The FGA of B_{42}^{T}

where a~g are the coefficients in DCT/IDCT matrix for different transform standards. Table 2 presents these coefficients of T_8 matrix.

In Eq. 30, the T_8 matrix can be decomposed as the product of 5 matrices expressed as follows:

$$T_8 = A_8 \cdot B_8 \cdot C_8 \cdot D_8 \cdot E_8 \tag{31}$$

$$T_8^T = E_8^T \cdot D_8^T \cdot C_8^T \cdot B_8^T \cdot A_8^T \tag{32}$$

Where:

$$A_8 = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & -1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & -1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & -1 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & -1 \end{pmatrix}, B_8 = \begin{pmatrix} 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & -1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & -1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & -b & -d & c & e \\ 0 & 0 & 0 & 0 & c & -b & e & d \\ 0 & 0 & 0 & 0 & -d & -e & -b & c \\ 0 & 0 & 0 & 0 & e & c & d & b \end{pmatrix}$$

$$C_8 = \begin{pmatrix} 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & g & -f & 0 & 0 & 0 & 0 \\ 0 & 0 & f & g & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix}, D_8 = \begin{pmatrix} a & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & a & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix}$$

$$E_8 = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix} \tag{33}$$

Table 2: The 8×8 DCT/IDCT coefficients for different coding standards

Variables	JPEG, MPEG 1/2/4	H.264/AVC	AVS	VC-1
a	362	8	8	12
b	502	12	10	16
c	426	10	9	15
d	284	6	6	9
e	100	3	2	4
f	473	8	10	16
g	196	4	4	6

In Eq. 33 and 24, B_{42} is a submatrix of C_8. According to Eq. 26, the hardware implementation of B_{42} requires 3 multiplications and 3 additions. Thus, the hardware implementation of C_8 requires 3 multiplications and 5 additions.

Let B_{84} the 4×4 submatrix of B_8 given in Eq. 34:

$$B_{84} = \begin{pmatrix} -b & -d & c & e \\ c & -b & e & d \\ -d & -e & -b & c \\ e & c & d & b \end{pmatrix} \tag{34}$$

B_{84} can be decomposed as the sum of 4 matrices expressed as follows:

$$B_{84} = S_1 + S_2 + S_3 + S_4 \tag{35}$$

Where:

$$S_1 = \begin{pmatrix} -b & 0 & 0 & e \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ e & 0 & 0 & b \end{pmatrix}, S_2 = \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & -b & e & 0 \\ 0 & -e & -b & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}, S_3 = \begin{pmatrix} 0 & 0 & 0 & 0 \\ c & 0 & 0 & d \\ -d & 0 & 0 & c \\ 0 & 0 & 0 & 0 \end{pmatrix}, S_4 = \begin{pmatrix} 0 & -d & c & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & c & d & 0 \end{pmatrix} \tag{36}$$

The hardware implementations of S1, S2, S3 and S4 can be expressed, respectively as Eq. 37, 38, 39 and 40; each one needs just 3 multiplications and 3 additions:

$$\begin{cases} O_{10} = e \cdot (I_0 + I_3) - (b + e) \cdot I_0 \\ O_{13} = (b - e) \cdot I_3 + e \cdot (I_0 + I_3) \end{cases} \tag{37}$$

$$\begin{cases} O_{21} = (b - e) \cdot I_2 - b \cdot (I_1 + I_2) \\ O_{22} = (b + e) \cdot I_1 - b \cdot (I_1 + I_2) \end{cases} \tag{38}$$

$$\begin{cases} O_{30} = c \cdot (I_1 + I_2) - (c + d) \cdot I_2 \\ O_{33} = c \cdot (I_1 + I_2) - (c - d) \cdot I_1 \end{cases} \tag{39}$$

$$\begin{cases} O_{41} = c \cdot (I_0 + I_3) - (c + d) \cdot I_0 \\ O_{42} = c \cdot (I_0 + I_3) - (c - d) \cdot I_3 \end{cases} \tag{40}$$

$$\begin{cases} O_0 = O_{10} + O_{30} \\ O_1 = O_{21} + O_{41} \\ O_2 = O_{22} + O_{42} \\ O_3 = O_{13} + O_{33} \end{cases} \tag{41}$$

where, I_0, I_1, I_2 and I_3 are the inputs, O_0, O_1, O_2 and O_3 are the outputs.

The direct hardware implementation of B_{84} needs 16 multiplications and 12 additions. Whereas, thanks to Eq. 35 B_{84} requires just 12 multiplications and 16 additions. So we have replaced 4 multiplications by 4 additions, this is a gain in term of additions, because the hardware implementation of each multiplication needs more of one addition. Thus, the hardware implementation of B_8 needs 12 multiplications and 20 additions. In the same way as B_4, C_8 requires 3 multiplications and 5 additions. D_8 needs 2 multiplications and A_8 needs 8 additions. E_8 do not require any operation, because it only permutes the outputs. Therefore, the hardware implementation of T_8 requires 17 multiplications and 33 additions, instead of 64 multiplications and 56 additions if there is no decomposition matrix. In the same way, according to Eq. 32, the hardware implementation of T_8^T requires just 17 multiplications and 33 additions. Indeed, B_{42}^T is a submatrix of C_8^T and the hardware implementation of B_{42}^T is expressed in Eq. 28. On the other hand, B_{84}^T is a submatrix of B_8^T, so according to Eq. 35 B_{84}^T can be expressed as follows:

$$B_{84}^T = S_1^T + S_2^T + S_3^T + S_4^T \tag{42}$$

Thus, the hardware implementations of B_{84}^T can be expressed as follows:

$$\begin{cases} O_{10} = e \cdot (I_0 + I_3) - (b + e) \cdot I_0 \\ O_{13} = (b - e) \cdot I_3 + e \cdot (I_0 + I_3) \end{cases} \tag{43}$$

$$\begin{cases} O_{21} = (b + e) \cdot I_2 - b \cdot (I_1 + I_2) \\ O_{22} = (b - e) \cdot I_1 - b \cdot (I_1 + I_2) \end{cases} \tag{44}$$

$$\begin{cases} O_{30} = c \cdot (I_1 + I_2) - (c + d) \cdot I_1 \\ O_{33} = c \cdot (I_1 + I_2) - (c - d) \cdot I_2 \end{cases} \tag{45}$$

$$\begin{cases} O_{41} = c \cdot (I_0 + I_3) - (c - d) \cdot I_3 \\ O_{42} = c \cdot (I_0 + I_3) - (c + d) \cdot I_0 \end{cases} \tag{46}$$

$$\begin{cases} O_0 = O_{10} + O_{30} \\ O_1 = O_{21} + O_{41} \\ O_2 = O_{22} + O_{42} \\ O_3 = O_{13} + O_{33} \end{cases} \tag{47}$$

where, I_0, I_1, I_2 and I_3 are the inputs, O_0, O_1, O_2 and O_3 are the outputs.

PROPOSED HARDWARE SHARING DESIGN OF FAST MULTISTANDARD DCT/IDCT

Proposed sharing hardware design of multistandard 1-D DCT/IDCT: According to Eq. 1 and 12, the 1-D of 4×4 and 8×8 DCT are computed respectively as follows:

$$W_{4\_1-D} = X \cdot C_{f4}^T,\ W_{8\_1-D} = X \cdot C_8^T \tag{48}$$

In the same way, in Eq. 2 and 13, the 1-D of 4×4 and 8×8 IDCT are computed respectively as follows:

$$X'_{4\_1-D} = W' \cdot C_{i4},\ X'_{8\_1-D} = W' \cdot C_8 \tag{49}$$

According to Eq. 48, 18 and 29, the 1-D of 4×4 and 8×8 DCT implementation can be achieved respectively as the hardware implementation of T_4 and T_8.

In the same way, according to Eq. 49, 19 and 29, the 1-D of 4×4 and 8×8 IDCT implementation can be achieved respectively as the hardware implementation of T_4^T and T_8^T.

In Eq. 21, the product of A_4, B_4 and C_4 is expressed as follows:

$$\begin{aligned} T_{4\_abc} &= A_4 \cdot B_4 \cdot C_4 \\ &= \begin{pmatrix} a & a & f & g \\ a & -a & g & -f \\ a & -a & -g & f \\ a & a & -f & -g \end{pmatrix} \end{aligned} \tag{50}$$

According to Eq. 31, the product of B_8, C_8 and D_8 is expressed as follows:

$$\begin{aligned} T_{8\_bcd} &= B_8 \cdot C_8 \cdot D_8 \\ &= \begin{pmatrix} a & a & f & g & 0 & 0 & 0 & 0 \\ a & -a & g & -f & 0 & 0 & 0 & 0 \\ a & -a & -g & f & 0 & 0 & 0 & 0 \\ a & a & -f & -g & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & -b & -d & c & e \\ 0 & 0 & 0 & 0 & c & -b & e & d \\ 0 & 0 & 0 & 0 & -d & -e & -b & c \\ 0 & 0 & 0 & 0 & e & c & d & b \end{pmatrix} \end{aligned} \tag{51}$$

In Eq. 50 and 51, we note that T_{4_abc} is a submatrix of T_{8_bcd}. According to Eq. 21, D_4 serves just to permute the outputs. Therefore, in hardware implementation of T_4 and T_8, the FGA of T_4 is a sub block of the one of T_8. In Eq. 9, 21 and 23, we note that H_2^T is a submatrix of B_4, thus the FGA of H_2^T is a sub block of the one of T_4. From the decomposition of T_4 and T_8 matrices; we have developed the sharing FGA of 1-D 2×2, 4×4 and 8×8 DCT which is presented in Fig. 3. In the same way, from the decomposition of T_4^T and T_8^T matrices; we have developed the sharing FGA of 1-D 2×2, 4×4 and 8×8 IDCT which is presented in Fig. 4.

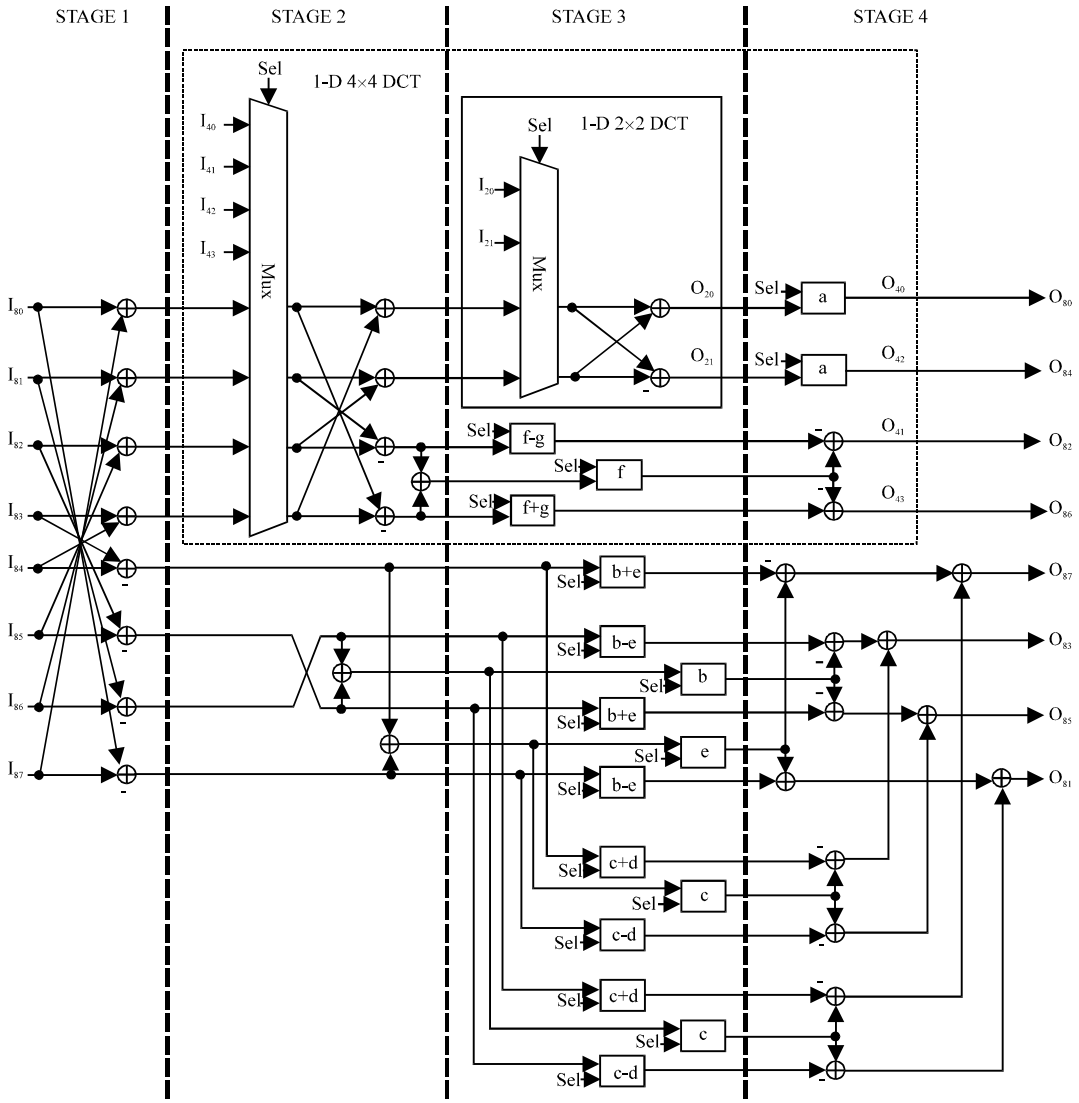

Fig. 3: The proposed sharing FGA of pipelined multistandard 1-D DCT

As indicated by Fig. 3, the proposed sharing FGA of multistandard 1-D DCT contains 1-D 2×2, 4×4 and 8×8 DCT. This sharing FGA includes 8 operation modes, each mode can be selected by the signal Sel. We have developed the proposed sharing FGA of pipelined multistandard 1-D DCT with 4 stages only. Furthermore, we have balanced between the speed and the number of stages by optimizing the critical paths in pipelined operation. Indeed, we have avoided series of a block multiplication with other operators in stage 3 and 4. In Fig. 3, we have saved 6 paths between stage 2 and stage 3 in the lower half of FGA, by using the common paths; this is in order to reduce the required hardware of registers in pipelined operation. In the same way, we have developed the proposed sharing FGA of multistandard 1-D IDCT presented in Fig. 4. Table 3 lists the selection signals Sel of multiplexers and multiplication blocks for multiple operational modes.

In Fig. 3 and 4, we have performed the hardware implementation of multiplication blocks by using multiplierless method, in which adders and shifters are used instead of

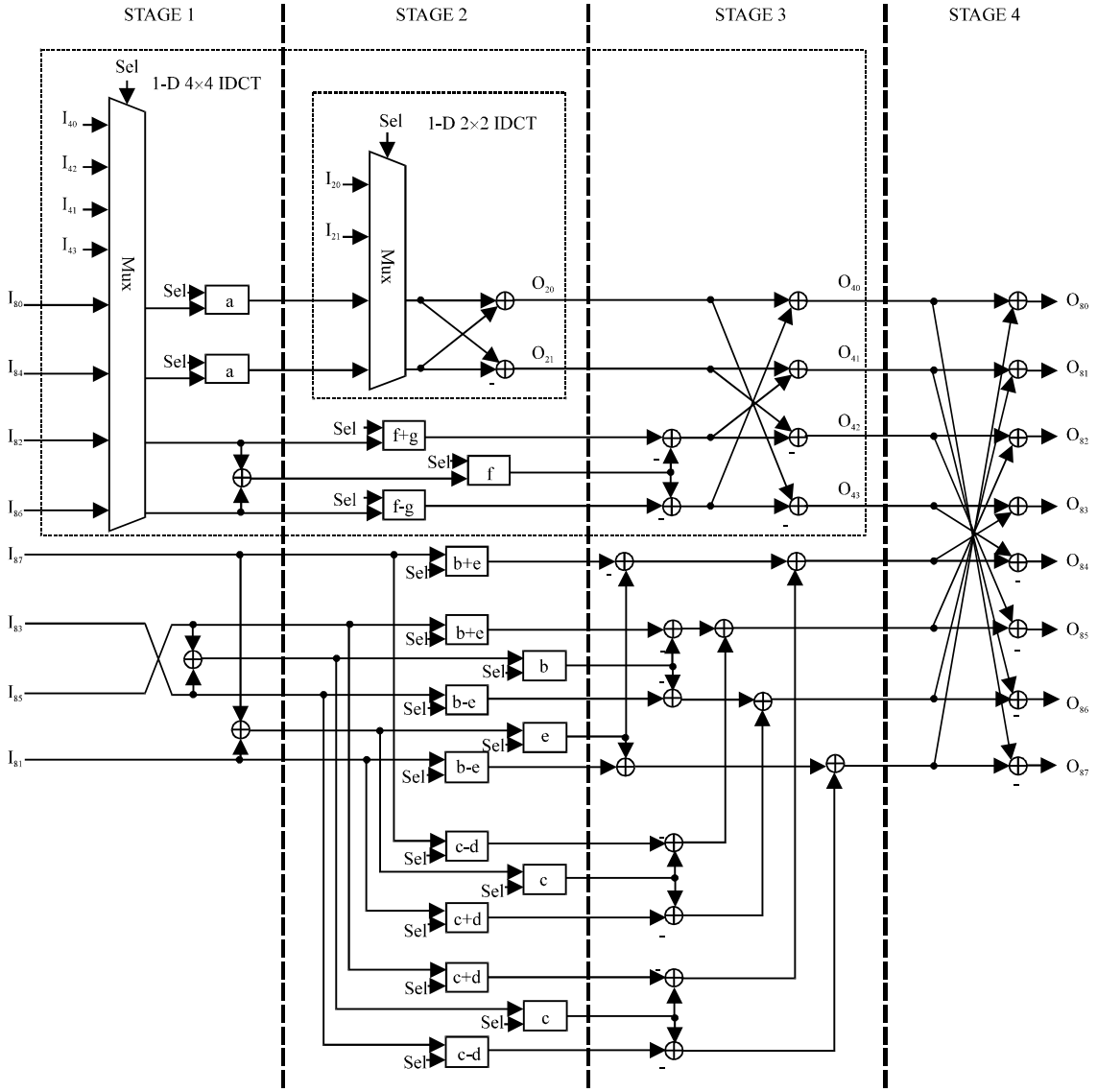

Fig. 4: The proposed sharing FGA of pipelined multistandard 1-D IDCT

Table 3: Selection signals of multiplexers for multiple operation modes of the proposed multistandard 1-D DCT/IDCT

Operation modes	Sel = ($Sel_2Sel_1Sel_0$)	Sel_2	Sel_1	Sel_0
1D 2×2 forwad/inverse hadamard transform	0	0	0	0
1D 4×4 forwad/inverse hadamard transform	1	0	0	1
1D 4×4 DCT/IDCT for H.264 / AVC	2	0	1	0
1D 4×4 DCT/IDCT for VC-1	3	0	1	1
1D 8×8 DCT/IDCT for H.264 / AVC	4	1	0	0
1D 8×8 DCT/IDCT for AVS	5	1	0	1
1D 8×8 DCT/IDCT for VC-1	6	1	1	0
1D 8×8 DCT/IDCT for JPEG, MPEG 1/2/4	7	1	1	1

multipliers. Each multiplication block supports all modes of multistandard video coding, for this purpose we have used the shared factorizations as shown in Table 4 and 5 to reduce the amounts

Table 4: Shared factorizations for multistandard 1-D DCT coefficients and complexity

	1-D 8×8 DCT				1-D 4×4 DCT			Complexity		
Variables	H.264/ AVC	AVS	VC-1	JPEG, MPEG 1/2/4	H.264/ AVC	Hadamard	VC-1	Adds	Shifts	Latency (add)
a	$8 = 2^3$	8	$12 = (8-2^1)\cdot 2^1$	$362 = (12\cdot 2^5-(8-2^1))-2^4$	1	1	$17 = (2^4+1)$	4	5	3
f+g	$12 = 3\cdot 2^2$	$14 = (12+2)$	$22 = 12\cdot 2^1-2$	$669 = 22\cdot 2^5-(32+3)$	$3 = 2+1$	2^1	$32 = 2^5$	5	5	3
f	$8 = 2^3$	$10 = 8+2$	$16 = 2^4$	$473 = (2^9-10\cdot 2^2)+1$	2^1	1	$22 = 10\cdot 2^1+2$	4	6	3
f-g	$4 = 2^2$	$6 = 4+2^1$	$10 = 6+4$	$277 = (2^8+1)+10\cdot 2^1$	1	0	$12 = 6\cdot 2^1$	4	5	3
b+e	$15 = 2^4-1$	$12 = 2^4-2^2$	$20 = 2^4+2^2$	$602 = (20\cdot 2^5-15\cdot 2^1)-2^3$				5	5	3
e	$3 = 2+1$	2^1	$4 = 2^2$	$100 = 3\cdot 2^5+4$				2	3	2
b-e	$9 = 8+1$	$8 = 2^3$	$12 = 8+2^2$	$402 = 12\cdot 2^5+9\cdot 2^1$				3	4	2
c+d	$16 = 2^4$	$15 = 16-1$	$24 = 16+2^3$	$710 = (24\cdot 2^5-15\cdot 2^2)+2^1$				4	5	3
c	$10 = 9+1$	$9 = 2^3+1$	$15 = 2^4-1$	$426 = (2^7+9\cdot 2^5)+10$				5	4	3
c-d	$4 = 2^2$	$3 = 2^1+1$	$6 = 3\cdot 2^1$	$142 = (2^7+2^4)-2^1$				3	5	2
b	$12 = 10+2^1$	$10 = 2^3+2^1$	$16 = 2^4$	$502 = 2^9-10$				3	4	2

Table 5: Shared factorizations for multistandard 1-D IDCT coefficients and complexity

	1-D 8×8 IDCT				1-D 4×4 IDCT			Complexity		
Variables	H.264/ AVC	AVS	VC-1	JPEG, MPEG 1/2/4	H.264/ AVC	Hadamard	VC-1	Adds	Shifts	Latency (add)
a	$8 = 2^3$	8	$12 = (8-2^1)\cdot 2^1$	$362 = (12\cdot 2^5-(8-2^1))-2^4$	1	1	$17 = (2^4+1)$	4	5	3
f+g	$12 = 3\cdot 2^2$	$14 = (12+2)$	$22 = 12\cdot 2^1-2$	$669 = 22\cdot 2^5-(32+3)$	$3/2 = (2+1)\cdot 2^{-1}$	2^1	$32 = 2^5$	5	6	3
f	$8 = 2^3$	$10 = 8+2^1$	$16 = 2^4$	$473 = (2^9-10\cdot 2^2)+1$	1	1	$22 = 10\cdot 2^1+2$	4	6	3
f-g	$4 = 2^2$	$6 = 4+2^1$	$10 = 6+4$	$277 = (2^8+1)+10\cdot 2^1$	$1/2 = 2^{-1}$	0	$12 = 6\cdot 2^1$	4	6	3
b+e	$15 = 2^4-1$	$12 = 2^4-2^2$	$20 = 2^4+2^2$	$602 = (20\cdot 2^5-15\cdot 2^1)-2^3$				5	5	3
e	$3 = 2+1$	2^1	$4 = 2^2$	$100 = 3\cdot 2^5+4$				2	3	2
b-e	$9 = 8+1$	$8 = 2^3$	$12 = 8+2^2$	$402 = 12\cdot 2^5+9\cdot 2^1$				3	4	2
c+d	$16 = 2^4$	$15 = 16-1$	$24 = 16+2^3$	$710 = (24\cdot 2^5-15\cdot 2^2)+2^1$				4	5	3
c	$10 = 9+1$	$9 = 2^3+1$	$15 = 2^4-1$	$426 = (2^7+9\cdot 2^5)+10$				5	4	3
c-d	$4 = 2^2$	$3 = 2^1+1$	$6 = 3\cdot 2^1$	$142 = (2^7+2^4)-2^1$				3	5	2
b	$12 = 10+2^1$	$10 = 2^3+2^1$	$16 = 2^4$	$502 = 2^9-10$				3	4	2

of additions and shifts. Furthermore, we have reduced the latency of the multiplierless computation to a maximum of 3 additions delay for each block multiplication. This is in order to increase the maximal frequency in pipelined operation.

As indicated by Table 4 and 5, we have minimized the required hardware of coefficient multiplications in term of additions and shifts, by looking for an efficient shared factorization between different values of multistandard transform, while keeping a maximum of 3 additions delay for each coefficient multiplication. Taking 'a' as an example of coefficient multiplication, Fig. 5 presents the flow graph of this coefficient, in which 4 additions and 5 shifts are only required for its hardware implementation. The signal 'Sel' is used to select the output mode of multistandard video.

Proposed multistandard 2-D DCT/IDCT sharing hardware design: We have performed the 2-D DCT/IDCT hardware implementation by using the row-column approach (Chang *et al*., 2000;

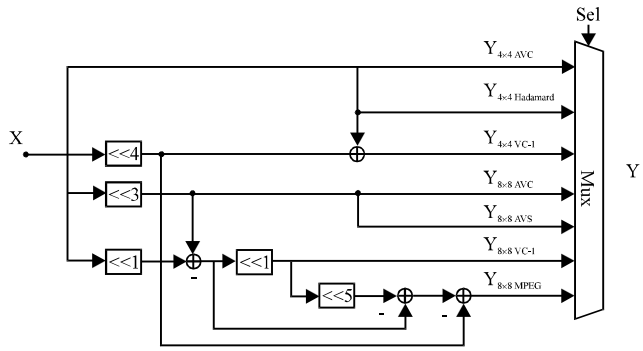

Fig. 5: Flow graph of ('a' block) coefficient multiplication

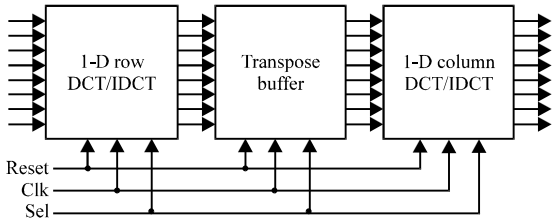

Fig. 6: The 2-D DCT/IDCT architecture

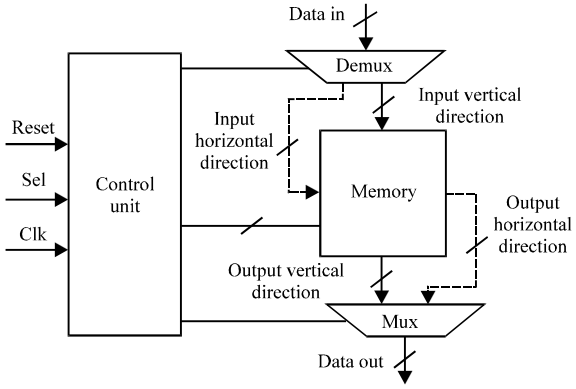

Fig. 7: Low area architecture of transpose buffer supporting multistandard transform

Hyesook *et al.*, 2000). As indicated by Fig. 6, the hardware architecture of 2-D DCT/IDCT requires two blocks of 1-D DCT/IDCT and one block for transpose buffer which is used to connect the two 1-D DCT/IDCT architectures. In this work, we have used in these architectures 16-bit arithmetic accuracy as (Fan *et al.*, 2011).

In the 2-D DCT/IDCT architecture, we have used efficient low area architecture for the transpose buffer supporting 2×2, 4×4 and 8×8 modes, while keeping the high-throughput in the full pipeline. Indeed, for 2-D 2×2, 4×4 and 8×8 DCT/IDCT, we obtain respectively 2, 4 and 8 output pixels of our 2-D DCT/IDCT at every clock cycle in the full pipeline without interruption. Figure 7 shows the low area transpose buffer architecture which contains one memory of 64-word 16-bit wide.

For 8×8 operation mode, the 8×8 array of 16-bit wide words is achieved to receive the inputs from the horizontal as well as from the vertical direction and to shift data in the same direction during every 8 clock cycles. At the beginning, the control unit delays the block memory operation by 4 clock cycles in order to achieve the full pipeline in 1-D row DCT/IDCT architecture. Then, the first 1-D DCT/IDCT architecture writes the output values line by line in vertical direction until the full pipeline in transpose buffer. Thereafter, when the second 1-D DCT/IDCT architecture reads the input values column by column from one direction of memory (Horizontal or Vertical), the first 1-D DCT/IDCT architecture writes simultaneously the output values line by line in the same direction (Horizontal or Vertical) during 8 clock cycles. The control unit block generates the control signals to manage the operation of this architecture. Indeed, these control signals allow to shift data and to switch the select lines of the multiplexer and demultiplexer, this is in order to switch the direction of data flow after every 8 clock cycles.

We have also performed the control unit to support 2-D 2×2 and 4×4 DCT/IDCT. Indeed, it delays the block memory operation for 2×2 and 4×4 modes of DCT/IDCT respectively by 1 and 3 clock cycles in order to achieve the full pipeline in 1-D row DCT/IDCT architecture. The control unit switches also the direction of data flow respectively after every 2 and 4 clock cycles for 2×2 and 4×4 modes.

IMPLEMENTATION RESULTS AND PERFORMANCE ANALYSIS

We have used Altera Cyclone II (EP2C35F672C6) FPGA, to perform the hardware implementation of the proposed multistandard 1-D and 2-D DCT/IDCT; we have also implemented the shifters by wiring. Table 6 shows the architecture comparison of different 1-D forward and inverse transforms for H.264/AVC, AVS, VC-1, JPEG and MPEG-1/2/4. In term of computational complexity, the amount of additions of our proposed 1-D DCT/IDCT design is reduced by 29.7% compared to 1-D IDCT design presented by Fan *et al.* (2011), this reduction is obtained thanks to the efficiency of our proposed generalized algorithms supporting multistandard DCT/IDCT which can decrease the amount of coefficient multiplications and that the shared factorization is achieved efficiently between different values of each coefficient multiplication. On the other hand, the required shifters of our DCT/IDCT design remain almost the same in comparison with Fan *et al.* (2011). In term of speed, the maximum frequency and throughput of the proposed 1-D DCT/IDCT design are increased by 24% in comparison with Fan *et al.* (2011), such as the architecture presented by Fan *et al.* (2011) used TSMC 0.18-µm CMOS standard cell technology for syntheses and chip layout. This increase of speed is explained by our optimization of critical paths and that the latency of the multiplierless computation is reduced to a maximum of 3 additions delay in each block multiplication. Furthermore, our proposed multistandard 1-D DCT/IDCT design requires 19 multiplexers while the architecture presented by Fan *et al.* (2011) needs 8, but the advantage of the proposed architecture is that there is the possibility to add other standards without adding any multiplexer. Whereas, there is no possibility to add any other standard in the architecture of Fan *et al.* (2011).

The proposed multistandard 2-D DCT/IDCT supports JPEG/MPEG-1/2/4, H.264/AVC 2×2/4×4/8×8, VC-1 4×4/8×8 and AVS 8×8. Fan *et al.* (2011) also used the row-column approach (Chang *et al.*, 2000; Hyesook *et al.*, 2000) in 2-D DCT/IDCT architecture. As shown in Table 7, the computational complexity of the proposed multistandard 2-D DCT/IDCT is decreased by 29.7% of additions while the number of shifts remains almost the same compared to Fan *et al.* (2011). At every clock cycle, we obtain as Fan *et al.* (2011) 2, 4 and 8 output pixels respectively at 2×2, 4×4

Table 6: Comparison for various multistandard 1-D DCT/IDCT architectures

Features	Multistandard 1-D IDCT (Fan *et al*., 2011)	Proposed multistandard 1-D DCT	Proposed multistandard 1-D IDCT
Adds	138	97	97
Shifts	82	79	81
Total logic elements	-	2158	2177
Max. Freq. (MHz)	125	156.91	155.69
Max. Throughput (Mpixels sec^{-1})	1000	1255	1245
Supporting standards	JPEG/MPEG-1/2/4	JPEG/MPEG-1/2/4	JPEG/MPEG-1/2/4
	H.264 2×2/4×4/8×8	H.264 2×2/4×4/8×8	H.264 2×2/4×4/8×8
	VC-1 4×4/8×8	VC-1 4×4/8×8	VC-1 4×4/8×8
	AVS 8×8	AVS 8×8	AVS 8×8

Table 7: Comparison for various multistandard 2-D DCT/IDCT architectures

Features	Multistandard 2-D DCT (Huang *et al*., 2008)	Multistandard 2-D DCT (Park *et al*., 2006)	Multistandard 2-D IDCT (Lee and Cho, 2008)	Multistandard 2-D IDCT (Fan *et al*., 2011)	Proposed multistandard 2-D DCT	Proposed multistandard 2-D IDCT
Adds	-	-	-	276	194	194
Shifts	-	-	-	164	158	162
Total Logic Elements	-	-	-	-	5586	5564
Max. Freq. (MHz)	50	80	136	125	149.68	147.78
Processing rate (Pixels $cycle^{-1}$)	8	8	> 2	2 at 2×2 mode	2 at 2×2 mode	2 at 2×2 mode
				4 at 4×4 mode	4 at 4×4 mode	4 at 4×4 mode
				8 at 8×8 mode	8 at 8×8 mode	8 at 8×8 mode
Max. Throughput (Mpixels sec^{-1})	400	-	> 45	1000	1197	1182
Supporting standards	JPEG/MPEG-1/2/4 H.264 2×2/4×4/8×8	JPEG/ MPEG-1/2/4 H.264 4×4	JPEG/MPEG-1/2/4 H.264 2×2/4×4/8×8 VC-1 4×4/8×8	JPEG/ MPEG-1/2/4 H.264 2×2/4×4/8×8 VC-1 4×4/8×8 AVS 8×8	JPEG/ MPEG-1/2/4 H.264 2×2/4×4/8×8 VC-1 4×4/8×8 AVS 8×8	JPEG/ MPEG-1/2/4 H.264 2×2/4×4/8×8 VC-1 4×4/8×8 AVS 8×8

and 8×8 mode of our 2-D DCT/IDCT in the full pipeline without interruption. The maximum frequency and throughput of the proposed multistandard 2-D DCT/IDCT is increased by 18.2 % in comparison with other architecture presented by Fan *et al*. (2011), this is obtained thanks to the efficiency of the proposed multistandard 1-D DCT/IDCT architecture.

CONCLUSION

In this study, the new multistandard 1-D DCT/IDCT generalized algorithms and their hardware sharing architectures have been proposed for H.264/AVC, AVS, VC-1, JPEG and MPEG 1/2/4 by using matrix decompositions, optimization of critical paths and shared factorization of coefficient multiplications using only shifts and additions. The proposed multistandard 1-D DCT/IDCT is achieved with low complexity and high speed compared to previous published design. The proposed multistandard 2-D DCT/IDCT is based on 1-D hardware sharing architecture and operates up to 147 MHz clock rate. Furthermore the proposed multistandard 2-D DCT/IDCT sharing design provides the flexibility to support other standard video, by upgrading only the shared factorization of each coefficient multiplication block.

REFERENCES

Chang, T.S., C.S. Kung and C.W. Jen, 2000. A simple processor core design for DCT/IDCT. Trans. Circ. Syst. Video Technol., 10: 439-447.

Fan, C.P. and G.A. Su, 2008. Efficient low-cost sharing design of fast 1-D inverse integer transform algorithms for H.264/AVC and VC-1. Signal Process. Lett., 15: 926-929.

Fan, C.P., C.H. Fang, C.W. Chang and S.J. Hsu, 2011. Fast multiple inverse transforms with low-cost hardware sharing design for multistandard video decoding. Trans. Circ. Syst. II, Exp. Briefs, 58: 517-521.

Gao, W., C. Reader, F. Wu, Y. He and L. Yu *et al.*, 2004. AVS-The Chinese next-generation video coding standard. Las Vegas, pp: 1-10. http://www.avs.org.cn/reference/AVS%20NAB%20Paper%20Final03.pdf

Huang, C.Y., L.F. Chen and Y.K. Lai, 2008. A high-speed 2-D transform architecture with unique kernel for multi-standard video applications. Proceedings of the International Symposium on Circuits and Systems, May 18-21, 2008, Seattle, WA., pp: 21-24.

Hwangbo, W. and C.M. Kyung, 2010. A multitransform architecture for H.264/AVC high-profile coders. Trans. Multimed., 12: 157-167.

Hyesook, L., V. Piuri and E.E. Swartzlander, 2000. A serial-parallel architecture for two-dimensional discrete cosine and inverse discrete cosine transforms. Trans. Comput., 49: 1297-1309.

ISO and IEC, 1993. Information Technology-coding of moving pictures and associated audio for digital storage media at up to about 1,5 Mbit/s-Part 2: Video. ISO/IEC 11172-2 MPEG-1. http://www.iso.org/iso/home/store/catalogue_tc/catalogue_detail.htm?csnumber=22411.

ISO/IEC 14496-2, 2004. Information technology-coding of audio-visual objects-Part 2: Visual. ISO/IEC 14496-2 MPEG-4. http://www.iso.org/iso/home/store/catalogue_tc/catalogue_detail.htm?csnumber=39259.

ISO and IEC, 2009. Coding of still pictures. ISO/IEC JTC 1/SC 29/WG 1. http://www.itscj.ipsj.or.jp/pr/sc29/29w02901_2009-10.pdf

Lee, S. and K. Cho, 2008. Architecture of transform circuit for video decoder supporting multiple standards. Elect. Lett., 44: 274-275.

Li, Y., Y. He and S.L. Mei, 2008. A highly parallel joint VLSI architecture for transforms in H.264/AVC. J. Signal Process. Syst., 50: 19-32.

Malvar, H.S., A. Hallapuro, M. Karczewicz and L. Kerofsky, 2003. Low-complexity transform and quantization in H.264/AVC. Trans. Circ. Syst. Video Technol., 13: 598-603.

Marpe, D., T. Wiegand and S. Gordon, 2005. H.264/MPEG4-AVC fidelity range extensions: Tools, profiles, performance and application areas. Proceedings of the International Conference on Image Processing, September 11-14, 2005, Genova, Italy, pp: 593-596.

Park, J.H., S.H. Lee, K.S. Lim, J.H. Kim and S. Kim, 2006. A flexible transform processor architecture for multi-CODECs (JPEG, MPEG-2, 4 and H.264). Proceedings of the International Symposium on Circuits Systems, May 21-24, 2006, Island of Kos, pp: 5347-5350.

Qi, H., Q. Huang and W. Gao, 2010. A low-cost very large scale integration architecture for multistandard inverse transform. IEEE Trans. Circ. Syst. II, Exp. Briefs, 57: 551-555.

SMPTE Standard, 2006. VC-1 compressed video bitstream format and decoding process: Amendment 2. SMPTE 421M, The Society of Motion Picture and Television Engineers, Barker Avenue, White Plains, NY., USA.

Su, G.A. and C.P. Fan, 2008. Low-cost hardware-sharing architecture of fast 1-D inverse transforms for H.264/AVC and AVS applications. IEEE Trans. Circ. Syst. II, Exp. Briefs, 55: 1249-1253.

Video Coding Standard, 1995. MPEG2 video IS. ISO/IEC 13818-2: 1995 (E). http://www.ece.cmu.edu/~ece796/documents/MPEG-2_Video_IS.doc.

Wiegand, T., G. Sullivan and A. Luthra, 2003. Draft ITU-T recommendation and final draft international standard of joint video specification (ITU-T rec. H.264/ISO/IEC 14496-10 AVC. Proceedings of the 8th International Meeting on Joint Video Team (JVC) of ISO/IEC MPEG and ITU-T VCEG, May 23-27, 2003, Geneva, Switzerland pp: 1-156.

Wien, M., 2003. Variable block-size transform for H.264/AVC. IEEE Trans Circ. Syst. Video Technol., 13: 604-613.

Yu, L., S. Chen and J. Wang, 2009. Overview of AVS-video coding standards. Signal Process. Image Commun., 24: 247-262.