IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received December 17, 2021, accepted January 21, 2022, date of publication January 25, 2022, date of current version January 31, 2022.

Digital Object Identifier 10.1109/ACCESS.2022.3146162

Effective Hardware Accelerator for 2D DCT/IDCT Using Improved Loeffler Architecture

ZHIWEI ZHOU AND ZHONGLIANG PAN
School of Physics and Telecommunication Engineering, South China Normal University, Guangzhou 510006, China

Corresponding author: Zhongliang Pan (panzhongliang@m.scnu.edu.cn)

This work was supported in part by the Guangzhou Science and Technology Project 201904010107, in part by the Guangdong Provincial Natural Science Foundation of China under Grant 2019A1515010793, and in part by the Guangdong Province Science and Technology Project 2016B090918071.

ABSTRACT This paper proposes an effective hardware accelerator for 2D 8 × 8 discrete cosine transform (DCT) and inverse discrete cosine transform (IDCT) using an improved Loeffler architecture. The accelerator optimizes the data stream of the Loeffler 8-point 1D DCT/IDCT according to the characteristics of image and video processing. An 8-stage pipeline structure greatly improves the processing speed by reasonably dividing the number of clock cycles and simplifying the arithmetic operations in each cycle. The multiplication-free approximation of the DCT coefficients is implemented through adders and shifters, combined with both fixed-point and canonic signed digit (CSD) coding. In particular, the proposed fast parallel transposed matrix architecture achieves the function of row-column coefficient conversion with lower circuit complexity. The FPGA implementation of the proposed architecture uses a Virtex-7 XC7VX330T device, running at 288 MHz with a throughput of 558 M Pixel/sec, and a Full HD real-time frame rate of up to 269 fps. Only 33 cycles are required to complete the 8 × 8 blocks of 2D DCT/IDCT, which can be used as a high-performance hardware accelerator for image and video compression encoding.

INDEX TERMS Loeffler algorithm, DCT, IDCT, parallel transpose, hardware accelerator.

I. INTRODUCTION

In recent years, portable multimedia devices have experienced a huge demand under the rapid development of computer information technology [1]. These devices require real-time processing of high-resolution, high-quality digital images and video data with stringent requirements on available resources, memory and power consumption [2]. Discrete cosine transform (DCT) and inverse discrete cosine transform (IDCT) have a wide range of applications in image and video coding due to their good energy compression performance, such as JPEG [3], MPEG [4]–[6], H.26x [7], [8]. In addition, the new compression scheme high efficiency video coding (HEVC) [9] uses DCT/IDCT integer conversion to achieve efficient compression performance at about half the bit rate needed to maintain the same video quality as H.264.

The computation of 8-point 1D DCT/IDCT appeared more fast algorithms in the early stage, most of which require 12-13 multiplications and 29 additions to implement [10]–[15]. The fast algorithm proposed by Loeffler *et al.* [16] reduces the number of multiplications required for 8-point DCT/IDCT to the theoretical limit, which is a fast and effective solution among many methods. Aakif *et al.* [17] proposed 1D DCT architecture using a multiplication-free approach by replacing the multiplication in Loeffler algorithm with adders and shifters. Shen *et al.* [18] implemented the Loeffler 8-point DCT by algebraic integer coding and using adder and shifter resources. The calculation of 2D DCT/IDCT can usually be decomposed into a transformation of two one-dimensional sequences corresponding to rows and columns, requiring a transpose matrix to cache the transformation coefficients of 1D DCT/IDCT. In [19], the redundant row-column exchange of the transposed buffer causes the conversion time to require 65 clock cycles, and the data read and write are inefficient.

Approximate architectures for hardware implementation of 2D DCT/IDCT are broadly classified into two types: algebraic integer coding [20]–[23] and fixed-point operations [24]–[28]. In the algebraic integer encoding implementation, the real cosine values are mapped to a group of integers

The associate editor coordinating the review of this manuscript and approving it for publication was Wei Wang.

and the final reconstruction step (FRS) is responsible for encoding these integers into an immobile point approximation representation. Madanayake *et al.* [21] proposed a row parallel 2D 8 × 8 DCT architecture based on algebraic integer coding, which can eliminate the quantization errors between intermediate reconstruction steps. Edirisuriya *et al.* [22] further improved the performance by reducing the number of four channels of 1D DCT decomposition based on this architecture. Coelho *et al.* [23] proposed 2D DCT algebraic integer method based on Loeffler 1D DCT algorithm with optimization of the FRS stage. Another widely used architecture focuses on approximate reconstruction of DCT coefficients through fixed-point representation. Subramanian *et al.* [26] proposed a 2D 8 × 8 DCT/IDCT VLSI full-pipeline less multiplication architecture, running at a frequency of 166 MHz with a latency of 65 clock cycles. Xing *et al.* [27] combined three approximation methods into a multiplier-less DCT architecture for wireless capsule endoscopy applications. Mert *et al.* [28] used a DSP block inside the FPGA to implement 2D DCT for future video coding (FVC) with transformable cell sizes of 4 × 4 and 8 × 8.

To achieve a low complexity, high real-time architecture suitable for computing 2D DCT/IDCT in image and video processing applications. In this paper, the data stream of Loeffler 8-point 1D DCT/IDCT is optimized and an efficient hardware architecture for 2D DCT/IDCT is proposed. The main recommendations and contributions are as follows:

1) An 8-stage pipeline structure is used to optimize the data stream of Loeffler 8-point DCT/IDCT and to improve the processing performance of the pipeline by reasonably extending the cycle and simplifying the computation operation of each cycle.
2) Addition and shift operations are combined with CSD coding to achieve multiplication-free operations, and the fast parallel transpose matrix greatly reduce the time for row-column coefficient conversion.
3) Low-cost and high-speed architecture that requires only 33 cycles to complete 2D 8 × 8 DCT/IDCT, which can be used as a hardware accelerator for image and video compression encoding.

The rest of the paper is organized as follows: Section II reviews the data stream structure of the Loeffler algorithm. Section III explains the optimization of the proposed method. Section IV shows the implementation of the hardware architecture in detail. Section V presents the comparison results of hardware resource consumption and processing performance. Finally, a conclusion is given in Section VI.

II. LOEFFLER TRANSFORM ALGORITHM

The 2D DCT/IDCT [29] computation for an N×N pixel matrix can be defined as

$$y(u,v) = \frac{2c(u)c(v)}{N}\sum_{i=0}^{N-1}\sum_{j=0}^{N-1} x(i,j)\cos\frac{(2i+1)u\pi}{2N}\cos\frac{(2j+1)v\pi}{2N}, \quad (1)$$

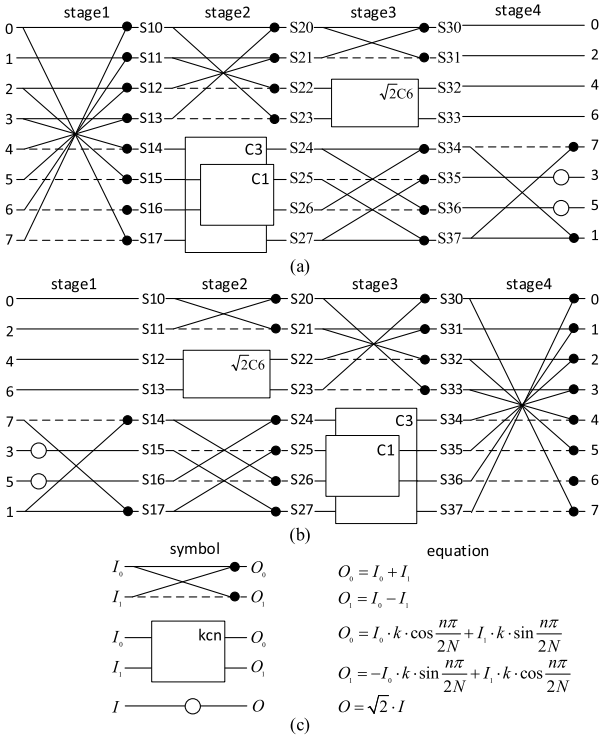


FIGURE 1. Loeffler algorithm structure. (a) 8-point DCT. (b) 8-point IDCT. (c) Symbolic operators of the algorithm.

$$x(i,j) = \frac{2}{N}\sum_{u=0}^{N-1}\sum_{v=0}^{N-1} c(u)c(v)y(u,v)\cos \times\frac{(2i+1)u\pi}{2N}\cos\frac{(2j+1)v\pi}{2N}, \quad (2)$$

$$c(u), c(v) = \begin{cases} \sqrt{\frac{1}{2}}\,u, & v = 0 \\ 1\,u, & v = 1, 2, \ldots, N-1, \end{cases} \quad (3)$$

where $x(i,j)$ is the original image, and $y(u,v)$ is the size of each frequency component after transformation. The 2D DCT/IDCT is decomposed into two sequences corresponding to rows and columns, which are transformed twice along the row and column directions, respectively. The calculation on 1D DCT/IDCT is given as

$$y(u) = \sqrt{\frac{2}{N}}c(u)\sum_{i=0}^{N-1} x(i)\cos\frac{(2i+1)u\pi}{2N}, \quad (4)$$

$$x(i) = \sqrt{\frac{2}{N}}\sum_{u=0}^{N-1} c(u)y(u)\cos\frac{(2i+1)u\pi}{2N}. \quad (5)$$

If $N = 8$, it requires 64 multiplications and 56 additions to calculate 8-point DCT/IDCT. Although the calculation of 2D DCT/IDCT using the row and column separation method still requires 1024 multiplications and 896 additions, which is a huge amount of computation. Loeffler *et al.* [16] proposed a fast algorithm for 8-point 1D DCT/IDCT, which reduced the computation to 11 multiplications and 29 additions, reaching the lowest value of theoretical computation. Fig. 1 shows

TABLE 1. Loeffler algorithm 8-point DCT data stream.

I	Stage				O
	1	2	3	4	
0	0+7	0+3	0+1		0
1	1+6	1+2	0-1		4
2	2+5	1-2	(2+3)·f+(c-f)·3		2
3	3+4	0-3	(2+3)·f-(c+f)·2		6
4	3-4	(4+7)·d+(e-d)·7	4+6	7-4	7
5	2-5	(5+6)·b+(g-b)·6	7-5	5·h	3
6	1-6	(5+6)·b-(g+b)·5	4-6	6·h	5
7	0-7	(4+7)·d-(e+d)·4	7+5	7+4	1

TABLE 2. Loeffler algorithm 8-point IDCT data stream.

I	Stage				O
	1	2	3	4	
0->0		0+1	0+3	0+7	0
4->1		0-1	1+2	1+6	1
2->2		(2+3)·f-(c+f)·3	1-2	2+5	2
6->3		(2+3)·f+(c-f)·2	0-3	3+4	3
7->4	7-4	4+6	(4+7)·d-(e+d)·7	3-4	4
3->5	5·h	7-5	(5+6)·b-(g+b)·6	2-5	5
5->6	6·h	4-6	(5+6)·b+(g-b)·5	1-6	6
1->7	7+4	7+5	(4+7)·d+(e-d)·4	0-7	7

the data stream structure of the Loeffler algorithm and the explanation of its operation symbols.

The butterfly operation (the second symbol in Fig. 1(c)) originally required 4 multiplications and 2 additions, and a total of 14 multiplications and 26 additions are required to compute an 8-point 1D DCT. The equivalence transformation of the butterfly operation is expressed as

$$y_0 = ax_0 + bx_1 = (b-a)x_1 + a(x_0 + x_1), \quad (6)$$

$$y_1 = -bx_0 + ax_1 = -(a+b)x_0 + a(x_0 + x_1). \quad (7)$$

Only 3 multiplications and 3 additions are required for a single butterfly operation by deformation, resulting in only 11 multiplications and 29 additions required to compute an 8-point 1D DCT. According to the trigonometric properties, the sine and cosine functions can be converted to each other, such as $\sin\frac{n\pi}{2N} = \cos\left(\frac{\pi}{2} - \frac{n\pi}{2N}\right)$. Combining the above equations, the 4-stage conversion process of the 8-point DCT data stream of the Loeffler algorithm is shown in Table 1. Table 2 presents the IDCT data stream, which can be regarded as the reverse execution of the 4-stage of DCT, and the difference is that the sign of the coefficient changes during the butterfly operation.

In Table 1, the input registers from 0 to 7 represent 8-point pixels, respectively. In each stage, the numbers represent the register position numbers and the letters from a to h are constant factors as shown in Table 3. For example, the formula 1+6 (in the first column of the second row) indicates that the values of register 1 and register 6 are added and stored into register 1. The formula $(5+6)\cdot b + (g-b)\cdot 6$ (in the second column of the sixth row) indicates that the result of adding registers 5 and registers 6 is multiplied by $\cos(\frac{\pi}{16})$, while register 6 is multiplied by $\cos\frac{7\pi}{16} - \cos\frac{\pi}{16}$, and the two results are added and stored in register 5. In addition, the input registers in Table 2 need to exchange the values of some registers when IDCT is performed, so as to match the transformed results of the corresponding data points.

TABLE 3. Corresponding value of constant factor.

a	b	c	d
$\cos\frac{\pi}{4}$	$\cos\frac{\pi}{16}$	$\sqrt{2}\cos\frac{\pi}{8}$	$\cos\frac{3\pi}{16}$
e	f	g	h
$\cos\frac{5\pi}{16}$	$\sqrt{2}\cos\frac{3\pi}{8}$	$\cos\frac{7\pi}{16}$	$\sqrt{2}$

III. OPTIMIZATION OF THE PROPOSED METHOD

A. REDISTRIBUTION OF LOEFFLER DATA STREAMS

Although the Loeffler algorithm has reached the theoretical minimum for the number of multiplications and additions, the calculation amount of each clock cycle is unevenly distributed. The formula $(5+6)\cdot b + (g-b)\cdot 6$ includes 2 multiplications and 2 additions on the critical path, which in turn affects the operating speed of the circuit design system. Based on the correlation of the data stream, the addition and multiplication in the algorithm must be executed sequentially. Consider the characteristic that the 2D DCT/IDCT conversion circuit uses serial clock input pixels in actual image and video compression processing applications. In an 8×8 size image block, for a 1D DCT/IDCT module, it takes 8 clock cycles to acquire a row of pixels. The original Loeffler algorithm completes the transformation of a row of pixels every 4 clock cycles, then within 8 cycles, it still needs to wait 4 cycles before starting the transformation of the next row of pixels.

The proposed method redistributes the data stream of Loeffler algorithm and applies pipelining techniques to distribute the operations evenly over the steps. By adding registers 8 and 9, the 1D DCT/IDCT is extended to 8 clock cycles, making each clock cycle operation contain only one addition or one multiplication. By splitting the critical long path, a single logic operation is independent of each clock cycle, thereby increasing the operating frequency of the system. The improved Loeffler 8-point 1D DCT/IDCT data streams are shown in Table 4 and Table 5, respectively.

B. MULTIPLICATION-FREE OPERATION

Since the multipliers in the algorithm are fixed constants, the floating-point operations are converted to integer multiplication by fixed-point approximation estimation. The canonic signed digit (CSD) [30] encoding is introduced to reduce the number of operations of the shift multiplication. By concatenating more than three "1" binary values, they are transformed into the form of subtracting 1 from the power of 2. For

TABLE 4. Improved Loeffler algorithm 8-point DCT data stream.

		Clock cycle								Output
		1	2	3	4	5	6	7	8	reg
	0	0+7	0+3	0+1						0
	1	1+6	1+2	0-1						4
	2	2+5	1-2					(c-f)·3	8+2	2
Input	3	3+4	0-3					(c+f)·2	8-3	6
reg	4	3-4	(e-d)·7			8+4	4+6	7-4		7
	5	2-5			(g-b)·6	9+5	7-5		5·h	3
	6	1-6			(g+b)·5	9-6	4-6		6·h	5
	7	0-7	(e+d)·7			8-7	7+5	7+4		1
Added	8		4+7	8·d		2+3	8·f			
reg	9		5+6	9·b						

TABLE 5. Improved Loeffler algorithm 8-point IDCT data stream.

		Clock cycle								Output
		1	2	3	4	5	6	7	8	reg
	0->0						0+1	0+3	0+7	0
	4->1						0-1	1+2	1+6	1
	2->2		(c+f)·3		8-2			1-2	2+5	2
Input	6->3		(c-f)·2		8+3			0-3	3+4	3
reg	7->4		7-4	4+6	(e+d)·7			8-4	3-4	4
	3->5	h·5		7-5		(g+b)·6		9-5	2-5	5
	5->6	h·6		4-6		(g-b)·5		9+6	1-6	6
	1->7		7+4	7+5	(e-d)·4			8+7	0-7	7
Added	8		2+3	8·f	4+7		8·d			
reg	9					5+6	9·b			

TABLE 6. CSD coding of multiplicative constant coefficients.

coefficient	coefficient×256	CSD
e-d	-71	111_1100_$\bar{1}$001
e+d	355	001_0110_0011
d	213	000_1101_0101
b	251	001_0000_$\bar{1}$011
g-b	-201	111_0011_100$\bar{1}$
g+b	301	001_0010_1101
f	139	000_1000_1011
c-f	196	000_1100_0100
c+f	473	010_0$\bar{1}$01_1001
h	362	001_0110_1010

1	2	3	4	5	6	7	8
9	10	11	12	13	14	15	16
17	18	19	20	21	22	23	24
25	26	27	28	29	30	31	32
33	34	35	36	37	38	39	40
41	42	43	44	45	46	47	48
49	50	51	52	53	54	55	56
57	58	59	60	61	62	63	64

(a)

1	9	17	25	33	41	49	57
2	10	18	26	34	42	50	58
3	11	19	27	35	43	51	59
4	12	20	28	36	44	52	60
5	13	21	29	37	45	53	61
6	14	22	30	38	46	54	62
7	15	23	31	39	47	55	63
8	16	24	32	40	48	56	64

(b)

FIGURE 2. Original method 8 × 8 matrix row-column transpose. (a) Pixel-by-pixel sequential address write storage. (b) Pixel-by-pixel discontinuous address readout.

example, the formula $1111 = 10000-1$, expressed by $1000\bar{1}$. By expanding the constant factor b in Table 3 by 256 times and then taking the integer as 251, the multiplication using CSD code can be expressed as

$$\begin{aligned} x \cdot 251 &= x \cdot 11111011_2 = x \cdot 1000\bar{1}011_{CSD} \\ &= (x \ll 8) - (x \ll 3) + (x \ll 1) + x. \end{aligned} \quad (8)$$

After CSD encoding, the fixed number multiplication operation is reduced from 7 shifts and 6 additions to 3 shifts and 3 additions. The use of adders and shifters to implement multiplication-free operations greatly reduces the consumption of internal multiplier resources. Table 6 shows all the multiplication coefficients in the algorithm, where the valid data width of the CSD encoding is 11bit and the first bit is the sign bit.

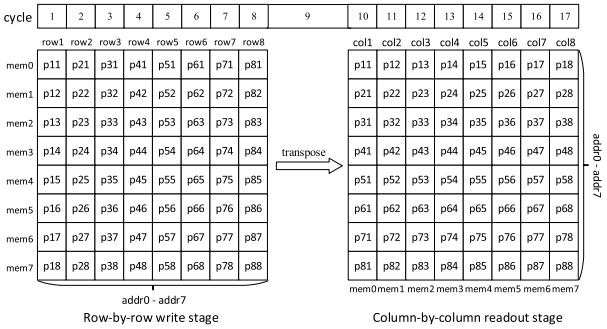


FIGURE 3. Row-column parallel transpose of 8 × 8 matrix.

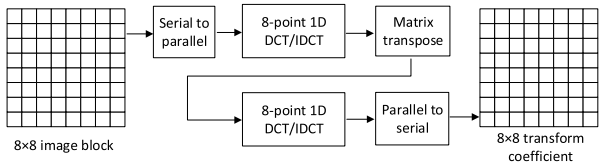


FIGURE 4. 2D DCT/IDCT structure diagram.

C. PARALLEL TRANSPOSE MATRIX

Fig. 2 shows that the original method uses a serial clock to synchronize a single pixel on an 8 × 8 image block to access a 64-depth RAM. The addresses from 1 to 8 are the first row of the matrix, and from 9 to 16 correspond to the second row of the matrix in turn. The pixels read according to the address order of [1, 9, 17, 25, 3, 41, 49, 57] are the first column of the matrix, realizing the function of row-column transposition. In the first 64 cycles, it is stored according to the pattern of Fig. 2(a), and the second 64 cycles is read according to the pattern of Fig. 2(b). A total of 128 clock cycles are required to complete the transposition function, with high data latency and low efficiency.

The proposed method is based on the parallel input and output of data streams, where eight 8-depth, 12-width memories are used and only 17 clock cycles are required to implement the transpose function. Fig. 3 depicts the memory transposition process, where p_{xy} represents the xth row and yth column of the matrix. In the write stage (from 1 to 8 cycles), the first cycle writes the first row of 8-point pixels in parallel to the 0-depth location of the registers in $mem0 - mem7$. Then, the pixels of each row are written to the corresponding location in memory in parallel cycle-by-cycle, and the storage of row pixels is completed after 8 cycles. Then in the 9th cycle, the values of the $addr0 - addr7$ registers in $mem0 - mem7$ are taken out at the same time, and the column pixels of the original matrix are output after pixel-by-pixel splicing, so as to realize the transposition of row-column coefficients. In the readout stage (from 10 to 17 cycles), 1 to 8 columns of pixels of the original matrix are output in parallel.

D. HARDWARE IMPLEMENTATION OF 2D DCT/IDCT

Fig. 4 shows the structure of the 2D DCT/IDCT, containing 8-point 1D DCT/IDCT, matrix transpose (MT), serial to parallel (SP) and parallel to serial (PS). First, the 8 × 8 image block performs data splicing on a row of 8-point pixels through the SP unit, and outputs the row pixels cycle-by-cycle to the first-level 8-point 1D DCT/IDCT unit for row transform. Then, the MT unit completes the row-column transposition and outputs the column pixels to the second-level 8-point 1D DCT/IDCT unit for column transform. Finally, the PS unit outputs the parallel data cycle-by-cycle into the positions corresponding to the transform coefficient matrix. The SP and PS units ensure that the serial and parallel conversion between 8 × 8 image block data streams can proceed smoothly.

IV. HARDWARE ARCHITECTURE

Fig. 5 shows the hardware architecture of the 8-point DCT with improved Loeffler algorithm. The entire architecture is divided into an 8-stage pipeline structure, where each register in the same stage performs only a single addition, subtraction or multiplication operation. The input data $x0 - x7$ during stage 1 are added and subtracted respectively and then stored in the register of the next stage. The addition of registers 8 and 9 is an important process for implementing critical path computation splitting. The corresponding constant factors for the Shift-Mul unit in each stage are used in the CSD coding results in Table 6. Since the constant factor is expanded in the Shift-Mul unit, the output result needs to be shifted right to reduce the corresponding multiple during stage 8.

The hardware architecture of the 8-point IDCT is shown in Fig. 6. The input data $y0 - y7$ are the coefficients after DCT, and the coefficient registers need to be swapped positions before the operation is performed. The data stream of the hardware architecture can be simplified as the reverse operation of DCT (see Fig. 5), transitioning from stage 8 to stage 1. The shift operations in stages 2, 6, 8 are used to match the correct result of the calculation. The output data $x0 - x7$ are the original input data obtained after IDCT. The transformation result obtained by Loeffler algorithm is $\sqrt{8}$ times of the original 1D DCT/IDCT, and the result will be enlarged by 8 times after 2D DCT/IDCT. The hardware implementation shifts the result by 3bit to the right to reduce the size by 8 times.

Table 7 describes the different valid widths of the input and output data of the 8-point 1D DCT/IDCT in the four operating modes, where the row/column 1D DCT and row/column 1D IDCT correspond to the 8-stage pipeline architecture of Fig. 5 and Fig. 6, respectively. When the row transform module is in 1D DCT mode, the input data is the image or the difference of the image, the data range is -255 to 255, and the valid data width is 9bit. While the row transform is in 1D IDCT mode, the input data is the coefficient after 2D DCT, and the data range is -721 to 721, and the valid data width is 11bit.

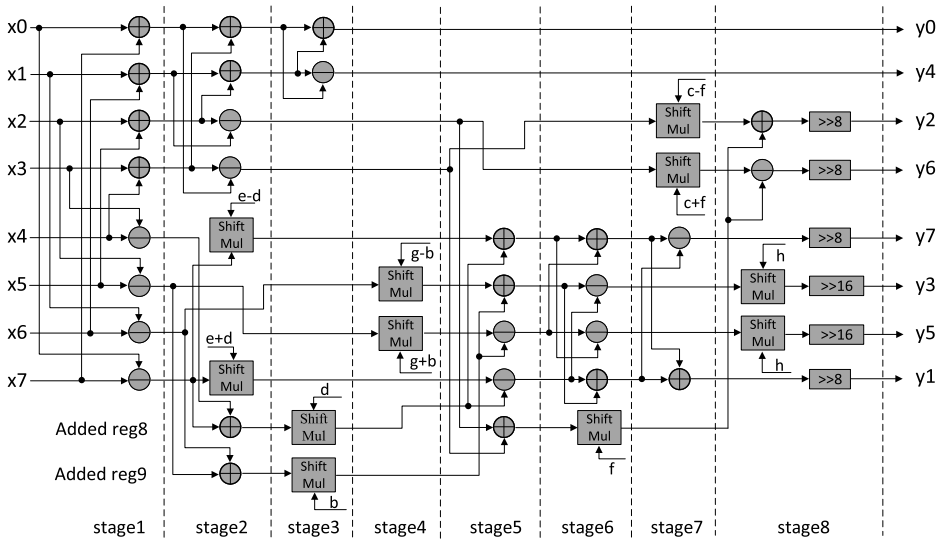


FIGURE 5. 8-point DCT 8-stage pipeline hardware structure diagram.

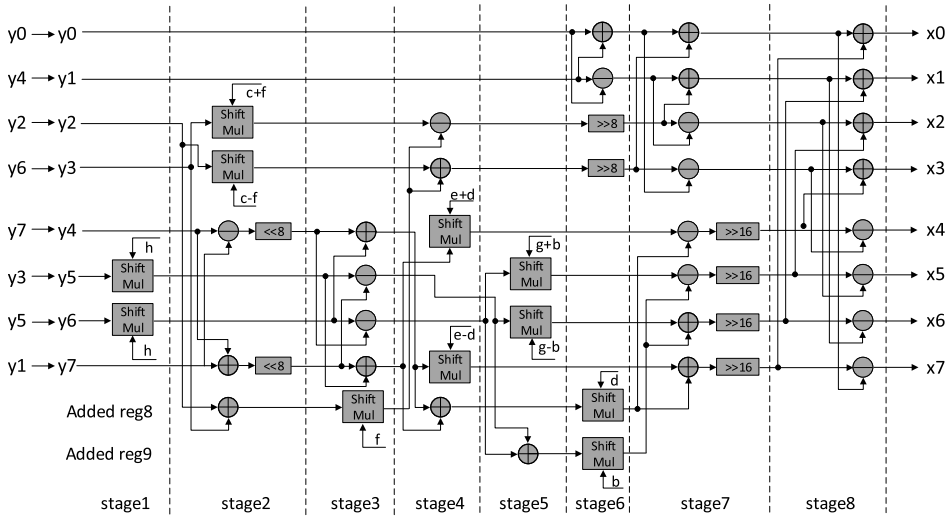


FIGURE 6. 8-point IDCT 8-stage pipeline hardware structure diagram.

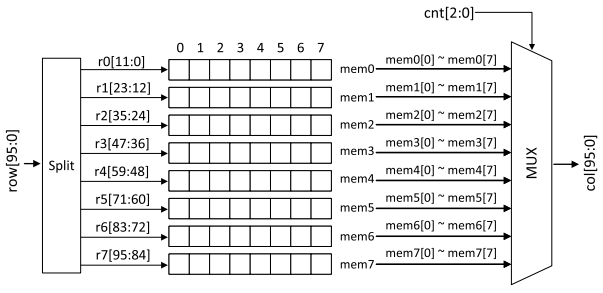


FIGURE 7. Matrix transpose hardware architecture diagram.

TABLE 7. 8-point 1D DCT/IDCT operating mode.

Mode	Input $x0-x7$	Data width	Output $y0-y7$	Data width
Row 1D DCT	-255∼255	9bit	-2040∼2040	12bit
Column 1D DCT	-2040∼2040	12bit	-721∼721	11bit
Row 1D IDCT	-721∼721	11bit	-2040∼2040	12bit
Column 1D IDCT	-2040∼2040	12bit	-255∼255	9bit

Fig. 7 depicts the hardware architecture of row-column matrix transposition, where *mem*0−*mem*7 are memories generated by LUT and register resources, and each memory has 8-depth and 12-width. The width of the memory depends on the valid width 12bit of the output data of row 1D DCT/IDCT in Table 7. First, the input data *row*[95:0] is split into 8 independent transformation coefficients by split unit respectively.

TABLE 8. Computational complexity of the approximation algorithm and reference algorithm of 8-point 1D DCT/IDCT.

	Reference algorithm	FGA Based [17]	AI-Based Loeffler [18]	Unified Based [31]	RAG-n Based [32]	Real-valued Based [33]	Proposed	
Transform	DCT/IDCT	DCT	DCT	DCT/IDCT	DCT	DCT	DCT	IDCT
Multiplier	11	0	0	0	0	0	0	0
Adder	29	67	25	100	69	54	51	51
Shifter	0	45	N/A	60	52	38	34	36

Then the data is written to $mem0-mem7$ in parallel and stored in full after 8 cycles. Finally, all the values in the memory are spliced from low to high to form the column data $col[95:0]$, through the 8-to-1 data selector, the transposition result is output in parallel cycle-by-cycle.

Fig. 8 presents the 2D 8×8 DCT/IDCT pipeline architecture. The row/column 1D DCT module uses the same hardware architecture, the difference is in the valid width of the input and output data, as well as the IDCT module. The 8×8 block is the pixel matrix to be transformed, and each row of 8-point is respectively spliced into parallel row data through the SP unit (see Fig. 4). The 64 pixels consist of 8 rows, each row contains 8 independent pixels. Thus, all the pixels of the 8×8 block are pre-stored in the generated memory with 8-depth and 88-width. Among them, 8-depth means 8 rows of pixels, and 88-width is composed of 8 independent 11bit pixels, which can simultaneously ensure the valid data width of the row 1D DCT/IDCT module input. First, each row of pixels is parallel data spliced by 8-point, and eight rows of pixels are output to the row 1D DCT/IDCT module cycle-by-cycle. After 8 cycles, the eighth row of data output is finished, while the row 1D DCT/IDCT module outputs the transformation result of the first row. While the row conversion coefficients are being output cycle-by-cycle, the transpose module is caching the results. Then after 8 cycles, the caching of the transform coefficients of the eight rows is completed, and the parallel row-column transposition is completed in the next cycle. Based on the same principle, when the transpose module outputs column data cycle-by-cycle, the column 1D DCT/IDCT module is running at the same time. After 8 cycles, the eighth column data output is completed, while column 1D DCT/IDCT starts to output the transformation result of the first row to the output 2D DCT/IDCT module. Finally, 8 cycles are needed to obtain the computed 8×8 block transform coefficient matrix. Efficient processing of the data stream is achieved by pipelining, which requires 33 cycles to complete the 8×8 blocks of 2D DCT/IDCT.

V. EXPERIMENTAL RESULTS

We compare the computational complexity of the proposed 8-point 1D DCT/IDCT 8-stage pipeline architecture with others using approximate methods, as shown in Table 8. The reference algorithm is based on the Loeffler [16] structure in Fig. 1 and uses multipliers, while the rest of the methods use adders and shifters to implement multiplication-free operations. The hardware implementations of [17] and [18] are both based on Loeffler architecture, [17] proposes flow-graph algorithm (FGA) method instead of multipliers, while [18] uses a algebraic integer (AI) coding approach to implement multiplication-free. Although the Unified algorithm proposed in [31] can efficiently compute the 8-point 1D DCT/IDCT, it consumes more adders and shifters. In [32], the RAG-n Based approach is used to approximate the DCT coefficients, which reduces the resource consumption. The adder and shifter used in the proposed architecture are comparable to [33], but our approximation using CSD coding is simple, fast and has lower latency.

TABLE 9. Comparison of PSNR of different methods.

	PSNR (dB) of 512×512 size images				
	Image 1	Image 2	Image 3	Image 4	Avg.
[19]	32.73	25.44	31.09	33.62	30.72
[34]	39.02	35.15	38.60	39.36	38.03
[35]	37.18	33.59	33.89	36.94	35.40
[36]	33.66	27.46	31.74	34.55	31.85
Proposed	39.68	39.22	36.56	40.08	38.89

To verify the computational accuracy of the proposed approximate architecture, we perform simulation tests by inputting classical images and compare the quality of the reconstructed images with other hardware architectures [19], [34], [35] and [36]. The peak signal-to-noise ratio (PSNR) is the ratio of maximum signal power to noise power, which can measure the image quality after processing, and the larger the unit dB value means less distortion. The PSNR comparison results of different methods are shown in Table 9. Fig. 9 shows the results of the reconstructed images of the original image 1 and other DCT approximation architectures. In Table 9, the average PSNR of the proposed architecture is higher than [19], [35] and [36], and our image restoration quality is better in Fig. 9. Our result is comparable to [34], but the butterfly operation of [34] uses a large number of multipliers and increases the resource consumption.

We use Verilog to describe the circuit model of the optimized 2D DCT/IDCT hardware architecture, and synthesize and implement it with Xilinx Vivado tools. The proposed architecture is implemented on a Virtex-7 XC7VX330T FPGA device and compared with other existing methods on equivalent hardware platforms, as shown in Table 10. Kitsos *et al.* [19] used the 2D-DCT-4ROM architecture to process 8×8 blocks and required 65 clock cycles to

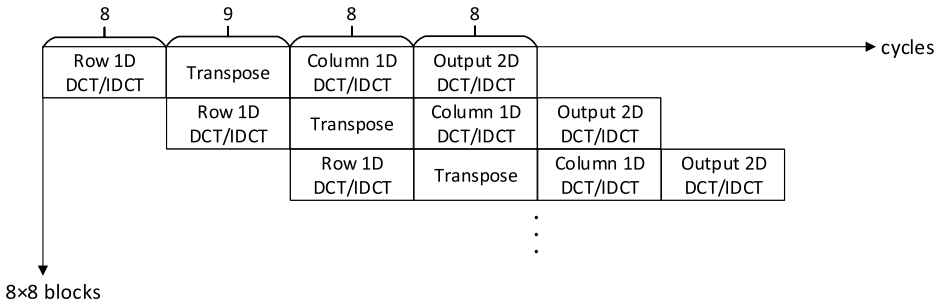


FIGURE 8. 2D 8 × 8 DCT/IDCT pipeline architecture.

TABLE 10. FPGA hardware implementation results comparison.

	[19]	[34]	[35]	[36]	[37]	Proposed	
Device	Virtex-7 XC7VX330T	Virtex-7 XC7VX330T	Virtex-6 XC6VLX240T	Arria-10 AX115N3F4512	Virtex-6 XC6VLX240T	Virtex-7 XC7VX330T	
LUT	2021	1700	17173	2140	30701	1904	2092
Register	1110	N/A	4571	3287	10965	2595	2780
DSP	0	64	0	N/A	0	0	0
Block RAM	N/A	N/A	8	0	16	0	0
Clock cycle	65	7	N/A	N/A	N/A	33	33
Frequency (MHz)	256	239	143	164	104	288	283
Throughput (Mega Pixels/sec)	252	488	N/A	N/A	N/A	558	549
Block Size	8×8	8×8	4×4, 8×8	8×8	4-/8-/16-/32-point	8×8	8×8
Transform	2D DCT	2D DCT	2D DCT	2D DCT	2D DCT/IDCT	2D DCT	2D IDCT

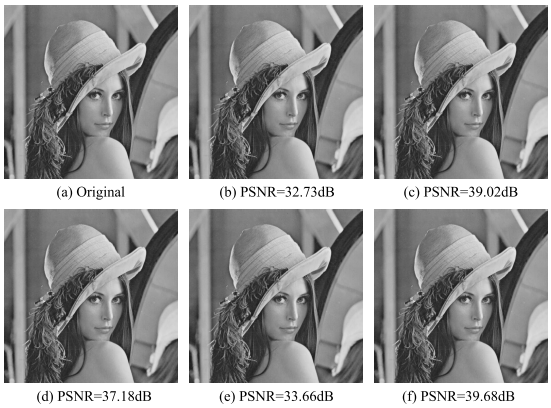


FIGURE 9. Comparison of Lena images using different DCT approximation architectures. (a) Original image. (b) Results of [19]. (c) Results of [34]. (d) Results of [35]. (e) Results of [36]. (f) Our results.

achieve a throughput of 252 M Pixels/sec at a clock rate of 256 MHz. Although only 2021 LUTs and 1110 registers are used, the ROM-based storage computing architecture consumes more block RAM resources and data reading and writing is not efficient. The four-stage architecture proposed by Chen *et al.* [34] supports implementation in a variety of FPGA platforms. Although the processing time of a single 8 × 8 block is 7 cycles, it consumes a large amount of on-chip DSP resources. Both the architectures proposed by Mert *et al.* [35] and Singhadia *et al.* [37] consume a lot of LUT and register resources. The architecture of [35] supports 2D DCT transformation units as 4 × 4 and 8 × 8, while [37] supports 4-/8-/16-/32-point length 2D DCT/IDCT. In [36], a new method called multiple transform selection (MTS) is proposed and selects the appropriate transform type for 2D DCT, running at 164 MHz on Arria 10 FPGA. The proposed architecture supports 2D DCT/IDCT of 8 × 8 image blocks. Compared with [19] and [34], our architecture achieves a better balance between resource consumption and processing speed. The application of 8-stage high-performance pipeline architecture and fast row-column transposition friendly hardware technology greatly improves computational efficiency. The accelerator requires only 33 cycles to complete the 2D 8 × 8 DCT at a low circuit complexity, with a throughput of 558 M Pixels/sec at a clock rate of 288 MHz.

TABLE 11. Comparison of the proposed 2D IDCT architecture with other hardware methods.

	[38]	[39]	[40]	Proposed
Transform unit	8×8	8×8	8×8	8×8
Max frequency (MHz)	200	222	246	283
Full HD frame rate (fps)	31	72	117	265

Table 11 shows the comparison between the proposed 2D IDCT architecture and other hardware methods. Cichoń *et al.* [38] and Kalali *et al.* [39] implemented 2D IDCT architectures by using high level synthesis (HLS) tools. The architecture in [38] uses only the Impulse C HLS tool, while [39] uses three HLS tools: Xilinx Vivado HLS, LegUp and Matlab Simulink HDL Coder. The description of parallel algorithms can be implemented quickly with the HLS tool,

FIGURE 10. Results of image noise removal using 2D 8 × 8 DCT+IDCT. (a) Original image. (b) Image with multiplicative noise added. (c) Light filtering mask. (d) Moderate filtering mask. (e) Heavy filtering mask.

TABLE 12. The hardware execution time of noise removal for different resolution images.

256×256	512×512	640×480
0.24 ms	0.95 ms	1.12 ms
1024×768	1280×720	1920×1080
2.86 ms	3.36 ms	7.52 ms

but it consumes too many resources and has high latency, resulting in generally low frame rates in Full HD. In addition, due to the stable operation of the pipeline architecture, our running maximum frequency and real-time frame rate have been increased compared to [40].

Fig. 10 shows the application of 2D 8×8 DCT+IDCT technique in the field of image noise removal. Different levels of multiplicative noise are added to the original image by different sigma values, as shown in Fig. 10(b). The filter mask size is 8 × 8, and the heavier the mask filtering, the more high frequency information is removed from the image. Fig. 10(c)-(e) are the images after 2D DCT, which are filtered by light mask, moderate mask and heavy mask, and finally restored by 2D IDCT. It can be seen that the image noise reduction effect is good, as the filter mask gradually increases, the noise removal is obvious, but some detailed information will be lost. Table 12 gives the hardware execution time required to remove noise from images of different resolutions. It only takes 7.52 ms to process a 1920 × 1080 image, which includes the processing time of 2D DCT, noise filtering and 2D IDCT, and the real-time frame rate can reach 133 fps. Experimental results show that the proposed architecture is more comparable than existing hardware methods, and can be applied to low-cost and high-speed hardware image and video compression processing.

VI. CONCLUSION

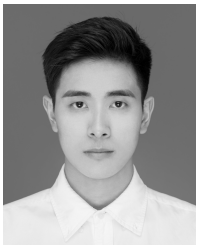
This paper proposes a fast and efficient hardware architecture for computing 2D 8 × 8 DCT/IDCT. Optimization of the data stream of the Loeffler 8-point 1D DCT/IDCT greatly improves the processing performance of the 8-stage pipeline structure. In addition, the processing method of approximate DCT coefficients without multiplication and the row-column fast parallel transposition provide superior compression performance under very low circuit complexity. Experimental results show that the accelerator has low resource consumption and high-speed transform performance, and is suitable for applications where high real-time performance and high bandwidth are required in image and video hardware compression coding.

REFERENCES

[1] M. Siekkinen, E. Masala, and J. K. Nurminen, "Optimized upload strategies for live scalable video transmission from mobile devices," *IEEE Trans. Mobile Comput.*, vol. 16, no. 4, pp. 1059–1072, Apr. 2017.

[2] A. Shabani, M. Sabri, B. Khabbazan, and S. Timarchi, "Area and power-efficient variable-sized DCT architecture for HEVC using muxed-MCM problem," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 68, no. 3, pp. 1259–1268, Mar. 2021.

[3] G. K. Wallace, "The JPEG still picture compression standard," *IEEE Trans. Consum. Electron.*, vol. 38, no. 1, pp. 18–34, Feb. 1992.
[4] K. Brandenburg and G. Stoll, "ISO/MPEG-1 audio: A generic standard for coding of high-quality digital audio," *J. Audio Eng. Soc.*, vol. 42, no. 10, pp. 780–792, 1994.
[5] P. A. A. Assuncao and M. Ghanbari, "A frequency-domain video transcoder for dynamic bit-rate reduction of MPEG-2 bit streams," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 8, no. 8, pp. 953–967, Dec. 1998.
[6] H.-J. Lee, T. Chiang, and Y.-Q. Zhang, "Scalable rate control for MPEG-4 video," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 10, no. 6, pp. 878–894, Sep. 2000.
[7] T. Wiegand, M. Lightstone, D. Mukherjee, T. G. Campbell, and S. K. Mitra, "Rate-distortion optimized mode selection for very low bit rate video coding and the emerging H.263 standard," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 6, no. 2, pp. 182–190, Apr. 1996.
[8] H. Schwarz, D. Marpe, and T. Wiegand, "Overview of the scalable video coding extension of the H.264/AVC standard," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 17, no. 9, pp. 1103–1120, Sep. 2007.
[9] G. J. Sullivan, J.-R. Ohm, W.-J. Han, and T. Wiegand, "Overview of the high efficiency video coding (HEVC) standard," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 22, no. 12, pp. 1649–1668, Dec. 2012.
[10] W.-H. Chen, C. Smith, and S. Fralick, "A fast computational algorithm for the discrete cosine transform," *IEEE Trans. Commun.*, vol. 25, no. 9, pp. 1004–1009, Sep. 1977.
[11] Z. Wang, "Fast algorithms for the discrete W transform and for the discrete Fourier transform," *IEEE Trans. Acoust., Speech, Signal Process.*, vol. 32, no. 4, pp. 803–816, Aug. 1984.
[12] B. G. Lee, "A new algorithm to compute the discrete cosine transform," *IEEE Trans. Acoust., Speech, Signal Process.*, vol. 32, no. 6, pp. 1243–1245, Dec. 1984.
[13] M. Vetterli and H. J. Nussbaumer, "Simple FFT and DCT algorithms with reduced number of operations," *Signal Process.*, vol. 6, no. 4, pp. 267–278, Aug. 1984.
[14] N. Suehiro and M. Hatori, "Fast algorithms for the DFT and other sinusoidal transforms," *IEEE Trans. Acoust., Speech, Signal Process.*, vol. 34, no. 3, pp. 642–644, Jun. 1986.
[15] H. S. Hou, "A fast recursive algorithm for computing the discrete cosine transform," *IEEE Trans. Acoust., Speech, Signal Process.*, vol. 35, no. 6, pp. 1455–1461, Oct. 1987.
[16] C. Loeffler, A. Ligtenberg, and G. S. Moschytz, "Practical fast 1-D DCT algorithms with 11 multiplications," in *Proc. Int. Conf. Acoust., Speech, Signal Process.*, vol. 2, May 1989, pp. 988–991.
[17] M. El Aakif, S. Belkouch, N. Chabini, and M. M. Hassani, "Low power and fast DCT architecture using multiplier-less method," in *Proc. Faible Tension Faible Consommation (FTFC)*, May 2011, pp. 63–66.
[18] Y. Shen and H. Oh, "Pipelined implementation of AI-based Loeffler DCT," *IEICE Electron. Exp.*, vol. 10, Oct. 2013, Art. no. 20130319.
[19] P. Kitsos, N. S. Voros, T. Dagiuklas, and A. N. Skodras, "A high speed FPGA implementation of the 2D DCT for ultra high definition video coding," in *Proc. 18th Int. Conf. Digit. Signal Process. (DSP)*, Jul. 2013, pp. 1–5.
[20] V. Dimitrov, K. Wahid, and G. Jullien, "Multiplication-free 8×8 2D DCT architecture using algebraic integer encoding," *Electron. Lett.*, vol. 40, no. 20, pp. 1310–1311, Jan. 2004.
[21] A. Madanayake, "A row-parallel 8×8 2-D DCT architecture using algebraic integer-based exact computation," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 22, no. 6, pp. 915–929, Jun. 2012.
[22] A. Edirisuriya, A. Madanayake, R. J. Cintra, V. S. Dimitrov, and N. Rajapaksha, "A single-channel architecture for algebraic integer-based 8×8 2-D DCT computation," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 23, no. 12, pp. 2083–2089, Dec. 2013.
[23] D. F. G. Coelho, S. Nimmalapalli, V. S. Dimitrov, A. Madanayake, R. J. Cintra, and A. Tisserand, "Computation of 2D 8×8 DCT based on the Loeffler factorization using algebraic integer encoding," *IEEE Trans. Comput.*, vol. 67, no. 12, pp. 1692–1702, Dec. 2018.
[24] Y. L. Shen, S. Jung, and H. C. Oh, "Loeffler DCT accelerator for small portable devices," *IEICE Electron. Exp.*, vol. 12, no. 13, 2015, Art. no. 20150411.
[25] Z. Wu, J. Sha, Z. Wang, L. Li, and M. Gao, "An improved scaled DCT architecture," *IEEE Trans. Consum. Electron.*, vol. 55, no. 2, pp. 685–689, May 2009.
[26] P. Subramanian and A. S. C. Reddy, "VLSI implementation of fully pipelined multiplierless 2D DCT/IDCT architecture for JPEG," in *Proc. IEEE 10th Int. Conf. Signal Process.*, Oct. 2010, pp. 401–404.
[27] Y. Xing, Z. Zhang, Y. Qian, Q. Li, and Y. He, "An energy-efficient approximate DCT for wireless capsule endoscopy application," in *Proc. IEEE Int. Symp. Circuits Syst. (ISCAS)*, May 2018, pp. 1–4.
[28] A. C. Mert, E. Kalali, and I. Hamzaoglu, "An FPGA implementation of future video coding 2D transform," in *Proc. IEEE 7th Int. Conf. Consum. Electron.-Berlin (ICCE-Berlin)*, Sep. 2017, pp. 31–36.
[29] N. Ahmed, T. Natarajan, and K. R. Rao, "Discrete cosine transform," *IEEE Trans. Comput.*, vol. C-23, no. 1, pp. 90–93, Jan. 1974.
[30] Z. Tang, J. Zhang, and H. Min, "A high-speed, programmable, CSD coefficient FIR filter," *IEEE Trans. Consum. Electron.*, vol. 48, no. 4, pp. 834–837, Nov. 2002.
[31] S. Liu, R. Yu, H. Yang, and H. Huang, "Unified algorithms for computation of different points integer 1-D DCT/IDCT for the HEVC standard," in *Proc. Int. Conf. Softw. Intell. Technol. Appl. Int. Conf. Frontiers Internet Things*, 2014, pp. 207–211.
[32] M. Masera, M. Martina, and G. Masera, "Adaptive approximated DCT architectures for HEVC," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 27, no. 12, pp. 2714–2725, Dec. 2017.
[33] S. Chatterjee and K. Sarawadekar, "An optimized architecture of HEVC core transform using real-valued DCT coefficients," *IEEE Trans. Circuits Syst. II, Exp. Briefs*, vol. 65, no. 12, pp. 2052–2056, Dec. 2018.
[34] M. Chen, Y. Zhang, and C. Lu, "Efficient architecture of variable size HEVC 2D-DCT for FPGA platforms," *AEU-Int. J. Electron. Commun.*, vol. 73, pp. 1–8, Mar. 2017.
[35] A. C. Mert, E. Kalali, and I. Hamzaoglu, "High performance 2D transform hardware for future video coding," *IEEE Trans. Consum. Electron.*, vol. 63, no. 2, pp. 117–125, May 2017.
[36] W. Imen, B. Fatma, M. Amna, and N. Masmoudi, "DCT -II transform hardware-based acceleration for VVC standard," in *Proc. IEEE Int. Conf. Design Test Integr. Micro Nano-Syst. (DTS)*, Jun. 2021, pp. 1–5.
[37] A. Singhadia, M. Mamillapalli, and I. Chakrabarti, "Hardware-efficient 2D-DCT/IDCT architecture for portable HEVC-compliant devices," *IEEE Trans. Consum. Electron.*, vol. 66, no. 3, pp. 203–212, Aug. 2020.
[38] S. Cichoń and M. Gorgoń, "H.265 inverse transform FPGA implementation in impulse C," in *Proc. Federated Conf. Comput. Sci. Inf. Syst.*, Sep. 2017, pp. 607–611.
[39] E. Kalali and I. Hamzaoglu, "FPGA implementations of HEVC inverse DCT using high-level synthesis," in *Proc. Conf. Design Archit. Signal Image Process. (DASIP)*, Sep. 2015, pp. 1–6.
[40] D. Mukherjee and S. Mukhopadhyay, "Hardware efficient architecture for 2D DCT and IDCT using Taylor-series expansion of trigonometric functions," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 30, no. 8, pp. 2723–2735, Aug. 2020.



ZHIWEI ZHOU received the B.S. degree from the Dongguan City College, Dongguan, China, in 2019. He is currently pursuing the M.S. degree in circuits and systems with the School of Physics and Telecommunication Engineering, South China Normal University, Guangzhou, China. His research interests include digital circuit logic design and hardware image processing.

ZHONGLIANG PAN received the M.S. degree from Tsinghua University, Beijing, China, in 1991, and the Ph.D. degree in circuits and systems from the University of Electronic Science and Technology of China, Chengdu, China, in 1997. He was a Postdoctoral Research Associate with Sun Yat-sen University, Guangzhou, China, from 1998 to 1999. Since 2004, he has been a Professor with the School of Physics and Telecommunication Engineering, South China Normal University, Guangzhou. His current research interests include image processing and electronic design automation and testing.