A Deterministic Low-Complexity Approximate (Multiplier-Less) Technique for DCT Computation

Junqi Huang, *Member, IEEE*, T. Nandha Kumar, *Senior Member, IEEE*, Haider A. F. Almurib, *Senior Member, IEEE*, and Fabrizio Lombardi, *Fellow, IEEE*

***Abstract*— The approximate (multiplier-less) two-dimensional discrete cosine transform (DCT) is a widely adopted technique for image/video compression. This paper proposes a deterministic low-complexity approximate DCT technique that accurately configures the size of the transform matrix (*T*) according to the number of retained coefficients in the zigzag scanning process. This is achieved by establishing the relationship between the number of retained coefficients and the number of rows of the *T* matrix. The proposed technique referred to as the zigzag low-complexity approximate DCT (ZLCADCT), when compared with approximate DCT (ADCT), decreases the number of addition operations and the energy consumption while retaining the PSNR of the compressed image. In addition, the ZLCADCT eliminates the zigzag scanning process used in the ADCT. Moreover, to characterize the deterministic operation of the ZLCADCT, a detailed mathematical model is provided. A hardware platform based on FPGAs is then utilized to experimentally assess and compare the proposed technique; as modular, deterministic, low latency, and scalable, the proposed techniques can be implemented upon any change in the number of retaining coefficients by realizing only a partial reconfiguration of the FPGA resources for the additional required hardware. The extensive simulation and experimental results show the superior performance compared to previous ADCT techniques under different metrics.**

***Index Terms*— Approximate DCT, addition, FPGA, low power DCT, zigzag scanning.**

I. Introduction

IMAGE and video coding techniques (such as high efficiency video coding and HEVC) play an important role in image processing, storage and transmission. Techniques such as the discrete cosine transform (DCT) are used to attain a high image compression rate usually at the expense of a large computational complexity and high energy consumption [1]. However, human senses such as eyes are tolerant of errors and a small degradation in the image quality is unlikely to be recognized. Approximate computing techniques such as an approximate/multiplier-less DCT (ADCT) thus have been proposed to decrease the amount of computation and improve circuit energy performance by sacrificing some accuracy in the outputs.

ADCT has been investigated by focusing on the design of a low-complexity transform matrix to reduce the computational complexity at algorithmic level and energy consumption at circuit-level. An approximate 4×4 DCT (with no multiplication required) is proposed in [2]. Also, several 8×8 approximate DCT matrix techniques are proposed in [3]–[9]; these techniques only require some shifting and additions, so again without multiplication in processing. Among these 8×8 low-complexity matrixes, [5], [7] introduce two approximate DCT matrixes which require only 14 additions. This is the lowest number of addition operations found in the technical literature. Moreover, multiplication-free 16×16 DCT architectures applicable to HEVC and matrix computation are proposed in [10]–[12]. Besides, approximate components, such as multipliers and adders are developed to decrease the number of operations for DCT at logic and transistor levels.

At logic level, an inexact systolic array is introduced in [13] to reduce the computational complexity of DCT matrix multiplication by allowing small errors. Two approximate 4-2 compressors (used in a multiplier) are proposed in [14] for image multiplication; this scheme decreases energy consumption and delay, so improving performance. At the transistor level, two approximate XOR-based adders are introduced in [15]; these adders can be employed to design approximate DCT hardware to improve performance with respect to power consumption and delay. Junqi *et al.* [16] propose a frequency up-scaling technique that allows adder cells of DCT to generate some error and operate under higher frequency for accelerating the processing. Three inexact adders that reduce energy consumption for DCT, are proposed in [17] by simplifying a traditional full ripple carry adder; their performances are evaluated in [1].

A different technique for approximate DCT targets a low-power design at logic level; [18] has developed a low-power approximate multiplier for DCT computation with a critical path shorter than a traditional multiplier. A low-power DCT architecture is also proposed in [19] to reduce the computational path of the most significant coefficients as well as errors caused by voltage scaling. The approach proposed in [20] achieves error resiliency under

Manuscript received December 1, 2018; revised February 8, 2019; accepted February 21, 2019. This paper was recommended by Associate Editor M. Mozaffari Kermani. *(Corresponding author: T. Nandha Kumar.)*

J. Huang, T. N. Kumar, and H. A. F. Almurib are with the Department of Electrical and Electronic Engineering, University of Nottingham, Nottingham, Malaysia (e-mail: kecx5hjn@nottingham.edu.my; nandhakumaar.t@nottingham.edu.my; haider.abbas@nottingham.edu.my).

F. Lombardi is with the Department of Electrical and Computer Engineering, Northeastern University, Boston, MA 02115 USA (e-mail: lombardi@ece.neu.edu).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TCSI.2019.2902415

voltage over-scaling by using unequal error protection according to the importance of the output coefficients for DCT. An Energy aware DCT is proposed in [21] by using a coefficient elimination technique to reduce energy consumption. An energy-efficient approximate DCT architecture is investigated in [22] by considering the sub-blocks of input images, while [23] has proposed generalized scalable and reconfigurable architectures for DCT in which when computing a large size matrix, resources are appropriately configured. An approximate IDCT (Inverse DCT) architecture for any IDCT size has been proposed in [24]; this architecture exploits an approximate matrix decomposition and simplifies the cosine and sine terms to reduce the required hardware during the process of recovering images in HEVC. [25] has proposed a low-complexity algorithm to avoid multiplications in the process of an Adaptive Multiple Transform (AMT) which involves DCT matrix of sizes from 8×8 to 128×128 for Post-HEVC technology. An energy-efficient ADCT is proposed in [26] by using three approximate methods (including optimizing calculation of float-point DCT coefficients, using threshold setting to determine required additions and partly adopting truncated approximate adders), while some floating point calculations and the threshold units may consume too much energy compared with integer ADCT.

All the above techniques accomplish a reduction in energy consumption for image compression at different levels (algorithmic, logic, transistor) and calculate the DCT with compressed sub-blocks; they are then scanned for retaining the number of coefficients of the compressed data to be transferred. The bit rate of transmission and compressed image quality can be changed by controlling the number of final retained coefficients and removing less significant coefficients during the process of scanning. However, ADCT matrixes proposed by [2]–[12] in the algorithm level all focus on simplifying the elements in the transform matrix to reduce the number of additions. The number of retained coefficients during the process of scanning is not considered in the design. This means that all coefficients should be calculated, even if some unused coefficients are finally removed by scanning. Therefore, the computation and related power consumption required for the ultimately not retained coefficients are redundant and they can be avoided. In previous literatures about ADCT techniques, the unused coefficients cannot be fully avoided in the calculation of ADCT, and these designs cannot dynamically be adjustable according to the different number of retained coefficients in scanning process.

The approximate nature of ADCT [7] originates from the feature that no multiplication is utilized, i.e., only additions are employed. ADCT [27] just prunes the transfer matrix (T) by adopting pruned DCT method in [28] to further enhance the approximation without considering the effect of the pruned matrix on the number of coefficients to be transmitted. Different from [27], the proposed ZLCADCT is a deterministic approach to configure the T matrix by establishing the relationship between the number of retained coefficients and the number of rows of the T matrix. Therefore, ZLCADCT eliminates the zigzag scanning process and significantly reduces the number of required hardware resources

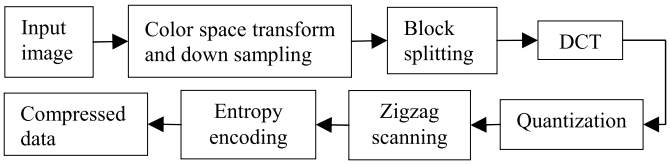


Fig. 1. The flow chart of JPEG encoder [29].

(adders, gates and LUTs). In addition, it reduces the figures of merit such as delay and energy consumption while retaining a nearly similar image quality compared with [7] and [27] (as measured by the PSNR). Moreover, the proposed technique is scalable and experimentally implemented using FPGAs; it is realized modularly, so that a change in the number of retaining coefficients does not require a complete reconfiguration of the FPGAs. Simulation and experimental results show that when compared with previous approximate techniques, the proposed technique offers superior performance.

In addition, the proposed method can be used in the image compression standard such as JPEG (Fig.1) to improve the resource utilization and energy consumption by retaining the output image quality. However, as stated in the title, this manuscript focuses on the optimization of DCT algorithm.

This paper is organized as follows. Section 2 presents the preliminaries such as the approximate DCT of [7], the butterfly algorithm and the zigzag scanning. Section 3 deals with the proposed zigzag low-complexity approximate DCT technique (ZLCADCT) as well as its model. Section 4 analyzes performance and complexity of the proposed techniques. Section 5 presents the simulation results, while Section 6 shows the hardware platform and the experimental results for the FPGA implementation of the proposed as well as existing techniques. Section 7 concludes the paper.

II. Preliminaries

Due to its energy properties, 8×8 Integer DCT has become one of the most commonly used transforms in a video compression system. Eq. (1) and (2) show the basic step for the DCT and inverse DCT (IDCT) respectively where X is the 8×8 matrix from the input image. Y is the 8×8 output matrix of the DCT and A denotes the conventional floating point 8×8 DCT matrix. For reducing the computational complexity, A is divided into D and T such that the DCT can be calculated in the integer domain (D is the 8×8 diagonal matrix and T is the 8×8 integer matrix). Then, the matrix D is extracted to generate $E_f = DE_1D^T \cdot E_f$ is computed as part of a quantization process ($\otimes$ denotes the dot product between two matrixes, while E_1 denotes the 8×8 matrix whose elements are all '1'). Therefore, $T\cdot X\cdot T^T$ is the main computation of the integer DCT, and by simplifying the matrix T, it is then possible to decrease the computational complexity of DCT.

$$Y = AXA^T = (DT)X\left(T^TD^T\right) = \left(TXT^T\right)\otimes\left(DE_1D^T\right) = \left(TXT^T\right)\otimes E_f \quad (1)$$

$$X = A^TYA = \left(T^TD^T\right)Y(DT) = T^T\left(Y\otimes(DE_1D^T)\right)T = T^T\left(Y\otimes E_f\right)T \quad (2)$$

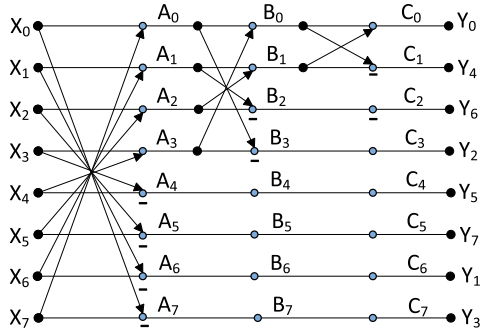


Fig. 2. Flow diagram of butterfly algorithm for matrix T of [7].

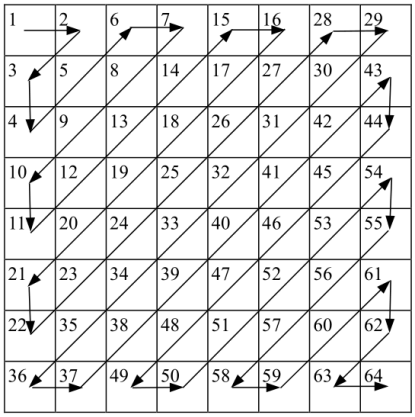


Fig. 3. Zigzag scanning [30].

Different types of 8×8 low-complexity matrix T have been proposed in [3]–[9]. Among them, the matrix T requiring the least number of computational resources is presented in [7]. It only requires 14 additions when using the butterfly algorithm. (3) shows the matrix T proposed in [7], while (3) is its D matrix. Fig. 2 illustrates the flow diagram of the butterfly algorithm for $T \cdot X$ using the matrix T of [7].

$$T = \begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 0 & 1 & 0 & 0 & 0 & 0 & -1 & 0 \\ 1 & 0 & 0 & -1 & -1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & -1 \\ 1 & -1 & -1 & 1 & 1 & -1 & -1 & 1 \\ 0 & 0 & 0 & 1 & -1 & 0 & 0 & 0 \\ 0 & -1 & 1 & 0 & 0 & 1 & -1 & 0 \\ 0 & 0 & 1 & 0 & 0 & -1 & 0 & 0 \end{pmatrix} \tag{3}$$

$$D = \text{diag}\left(\frac{1}{\sqrt{8}}, \frac{1}{\sqrt{2}}, \frac{1}{2}, \frac{1}{\sqrt{2}}, \frac{1}{\sqrt{8}}, \frac{1}{\sqrt{2}}, \frac{1}{2}, \frac{1}{\sqrt{2}}\right) \tag{4}$$

For image coding after completing DCT, every compressed sub-block must undergo the so-called zigzag scanning (shown in Fig. 3). The zigzag scanning finally determines the number of coefficients that must be retained as compressed data; for example, if 10 coefficients are retained, all coefficient data numbered from 11 to 64 are removed and set to zero. Fig. 4 shows the flow diagram in terms of computational steps for the entire DCT.

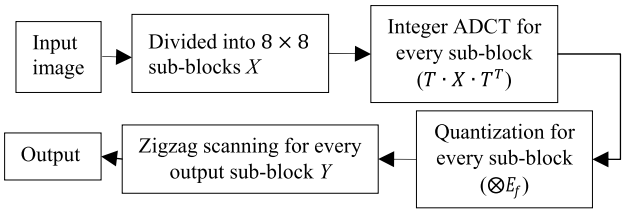


Fig. 4. Computational flow diagram for a 8×8 integer DCT.

III. Proposed Approximate DCT Technique

In this section, the proposed technique (ZLCADCT) is initially presented; then its mathematical modelling is pursued in detail. ZLCADCT configures the T matrix by establishing a relationship between the number of retained coefficients and the number of rows of the T matrix, thereby reducing power/energy consumption and circuit delay. Zigzag scanning is finally removed at the end of the DCT transform; moreover, the output image quality can be changed when the number of coefficients to be retained from the input is changed.

A. Proposed ZLCADCT

The proposed ZLCADCT utilizes a deterministic method to achieve a low complexity approximate DCT by preprocessing the transform matrix T such that a new T_p matrix is found as a function only of the number of retained coefficients (achieved by zigzag scanning) and reducing the number of additions to be performed. In zigzag scanning (with an ordering from the upper-left to the bottom-right), only some coefficients are retained; therefore, processing those coefficients that are not retained is redundant. In ZLCADCT, the implementation of the required zigzag scanning is performed in the earlier stage of DCT, i.e., in the T matrix. So, it is possible to reduce the number of calculations by adjusting the matrix T, while computing the integer DCT ($T \cdot X \cdot T^T$) for only those coefficients that are retained. The resulting matrix is now denoted as T_p. The following definitions are initially introduced in this manuscript.

1. The *Actual Coefficient Retained* (*ACR*) is defined as the number of coefficients generated when calculating the new output matrix $Y = T_p \cdot X \cdot {T_p}^T$.
2. The *Targeted Coefficient Retained* (*TCR*) is defined as the number of coefficients of the new output matrix Y that are planned to be retained, i.e., *TCR* is contained in *ACR* and obviously, $TCR < ACR$.

T_p is generated based on the targeted coefficients to be retained. Table 1 shows a few *TCR* values and their corresponding number of rows ('m') to be retained in the T_p matrix, as well as the output vectors created by T_p in $X1 = T_p \cdot X$. During the operation $X1 = T_p \cdot X$, for example if the number of targeted coefficients to be retained (*TCR*) in the process of zigzag scanning is only 6, then only the first three rows ($m = 3$) of the T matrix are retained (Table 1) by using the truncation measure from [27] while the coefficients for the other rows are set to zero, thus systematically generating the new T_p matrix (as in (5)). Its butterfly flow diagram (Fig. 5)

TABLE I
TCR RANGE AND OUTPUT VECTOR FOR DIFFERENT 'm' VALUES

m	*TCR*	Output vector $Z_{0\ldots m-1,j}(1 \ldots m)$
1	$TCR = 1$	$Z_{0,j}(1)$
2	$TCR \in [2,3]$	$Z_{0,j}(1)$ *to* $Z_{1,j}(2)$
3	$TCR \in [4,6]$	$Z_{0,j}(1)$ *to* $Z_{2,j}(3)$
4	$TCR \in [7,10]$	$Z_{0,j}(1)$ *to* $Z_{3,j}(4)$
5	$TCR \in [11,15]$	$Z_{0,j}(1)$ *to* $Z_{4,j}(5)$
6	$TCR \in [16,21]$	$Z_{0,j}(1)$ *to* $Z_{5,j}(6)$
7	$TCR \in [22,28]$	$Z_{0,j}(1)$ *to* $Z_{6,j}(7)$
8	$TCR \in [29,36]$	$Z_{0,j}(1)$ *to* $Z_{7,j}(8)$

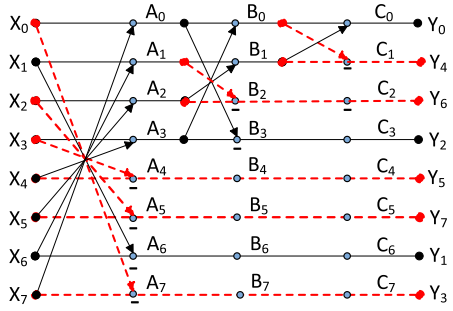


Fig. 5. Butterfly flow diagram of T_p using the truncation measure of [27] for the proposed ZLCADCT when 6 coefficients are retained.

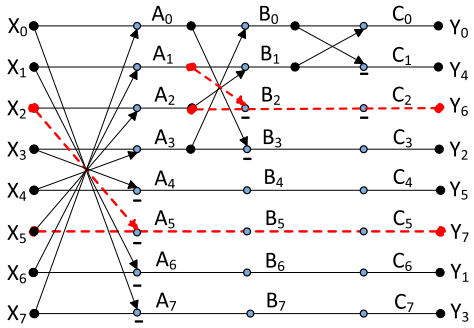


Fig. 6. Butterfly flow diagram of T_p using the truncation measure from [27] for proposed ZLCADCT when 21 coefficients are retained.

shows that only 9 additions (*X1* requires 72 additions) are required (the red dashed lines identify the redundant operations not performed by ZLCADCT). Upon applying the new T_p matrix on the input image *X*, the resulting matrix *X1* has coefficients only for three rows (same as the size of the T_p matrix). Therefore, the coefficients in the other rows are not computed; in the first step, T_p substantially reduces 40 addition operations compared with ADCT [7]. As a further example, consider when 21 coefficients are retained in the zigzag scanning; then, only the first six rows are calculated. The new T_p matrix is given in (6), and its butterfly flow diagram is shown in Fig. 6.

$$T_p = \begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 0 & 1 & 0 & 0 & 0 & 0 & -1 & 0 \\ 1 & 0 & 0 & -1 & -1 & 0 & 0 & 1 \end{pmatrix} \quad (5)$$

TABLE II
LUT FOR THE *TCR* RANGE

TCR ∈		Number of retained coefficients for each row ('k')							
		1	2	3	4	5	6	7	8
Number of rows ('m')	1	[1,2)	[2,6)	[6,7)	[7,15)	[15,16)	[16,28)	[28,29)	[29,64]
	2	[3,5)	[5,8)	[8,14)	[14,17)	[17,27)	[27,30)	[30,43)	[43,64]
	3	[4,9)	[9,13)	[13,18)	[18,26)	[26,31)	[31,42)	[42,44)	[44,64]
	4	[10,12)	[12,19)	[19,25)	[25,32)	[32,41)	[41,45)	[45,54)	[54,64]
	5	[11,20)	[20,24)	[24,33)	[33,40)	[40,46)	[46,53)	[53,55)	[55,64]
	6	[21,23)	[23,34)	[34,39)	[39,47)	[47,52)	[52,56)	[56,61)	[61,64]
	7	[22,35)	[35,38)	[38,48)	[48,51)	[51,57)	[57,60)	[60,62)	[62,64]
	8	[36,37)	[37,49)	[49,50)	[50,58)	[58,59)	[59,63)	[63,64)	64

$$T_p = \begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 0 & 1 & 0 & 0 & 0 & 0 & -1 & 0 \\ 1 & 0 & 0 & -1 & -1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & -1 \\ 1 & -1 & -1 & 1 & 1 & -1 & -1 & 1 \\ 0 & 0 & 0 & 1 & -1 & 0 & 0 & 0 \end{pmatrix} \quad (6)$$

Next, for the operation $Y = X1 \cdot T_p^T$, the resulting size of the Y matrix is $m \times m$. For example, $TCR = 6$ results in a Y matrix of size 3×3 with the number of actual coefficients retained (*ACR*) of 9. Therefore, the resulting three coefficients are not necessary. To address this concern, a look-up-table (LUT) is needed (Table 2) for the number of coefficients retained in each row ('k') as a function of *TCR* and 'm'. This LUT is used to check the number of coefficients to be retained in each row of Y, so that T_p can be adjusted to be of a $k \times 8$ size and so avoiding the generation of additional and unnecessary coefficients in each row. Thus, for $TCR = 6$, as gray highlighted but with right slash too in Table 2, the numbers of retained coefficients 'k' are 3 in the first row (because *TCR*∈ [6,7])), 2 in the second row (*TCR*∈ [5,8])) and 1 for the third row (*TCR*∈ [4,9])). Furthermore, (7), (8) and (9) illustrate the difference for the operation of Y in the presence or absence of a LUT. With a LUT, the last coefficient ($\sum_{m=0}^{7} T_{2m} \sum_{n=0}^{7} T_{1n} X_{nm}$) in the second row and the last two coefficients ($\sum_{m=0}^{7} T_{1m} \sum_{n=0}^{7} T_{2n} X_{nm}$ and $\sum_{m=0}^{7} T_{2m} \sum_{n=0}^{7} T_{2n} X_{nm}$) in the third row of Y are zero. The final 8×8 output matrix Y is given in (10); the 6 retained coefficients are the same for both ZLCADCT and when employing ADCT [7] by applying the zigzag scanning to retain these coefficients. Hence, ZLCADCT retains the targeted coefficients with no zigzag scanning while substantially reducing the number of addition operations (as shown in detail in later sessions). The algorithm of ZLCADCT and the process step of ZLCADCT are shown below. Fig. 7 shows the flow diagram in terms of the entire ZLCADCT.

The process of processing an image by ZLCADCT is given by the following steps:

1. Read the image and divide it into 8×8 sub-blocks (input matrix X).
2. For a given *TCR* value provided by the user, calculate the value of 'm' from Table 1 to determine T_p ($m \times 8$ size).
3. For every sub-block matrix X.

$$X1 \cdot T_p^T = \begin{pmatrix} \sum_{m=0}^{7} T_{0m} \sum_{n=0}^{7} T_{0n} X_{nm} & \sum_{m=0}^{7} T_{1m} \sum_{n=0}^{7} T_{0n} X_{nm} & \sum_{m=0}^{7} T_{2m} \sum_{n=0}^{7} T_{0n} X_{nm} \\ \sum_{m=0}^{7} T_{0m} \sum_{n=0}^{7} T_{1n} X_{nm} & \sum_{m=0}^{7} T_{1m} \sum_{n=0}^{7} T_{1n} X_{nm} & \mathbf{\sum_{m=0}^{7} T_{2m} \sum_{n=0}^{7} T_{1n} X_{nm}} \\ \sum_{m=0}^{7} T_{0m} \sum_{n=0}^{7} T_{2n} X_{nm} & \mathbf{\sum_{m=0}^{7} T_{1m} \sum_{n=0}^{7} T_{2n} X_{nm}} & \mathbf{\sum_{m=0}^{7} T_{2m} \sum_{n=0}^{7} T_{2n} X_{nm}} \end{pmatrix} \tag{8}$$

$$X1 \cdot T_p^T = \begin{pmatrix} \sum_{m=0}^{7} T_{0m} \sum_{n=0}^{7} T_{0n} X_{nm} & \sum_{m=0}^{7} T_{1m} \sum_{n=0}^{7} T_{0n} X_{nm} & \sum_{m=0}^{7} T_{2m} \sum_{n=0}^{7} T_{0n} X_{nm} \\ \sum_{m=0}^{7} T_{0m} \sum_{n=0}^{7} T_{1n} X_{nm} & \sum_{m=0}^{7} T_{1m} \sum_{n=0}^{7} T_{1n} X_{nm} & 0 \\ \sum_{m=0}^{7} T_{0m} \sum_{n=0}^{7} T_{2n} X_{nm} & 0 & 0 \end{pmatrix} \tag{9}$$

$$Y = T_p \cdot X1 \cdot T_p^T = \begin{pmatrix} \sum_{m=0}^{7} T_{0m} \sum_{n=0}^{7} T_{0n} X_{nm} & \sum_{m=0}^{7} T_{1m} \sum_{n=0}^{7} T_{0n} X_{nm} & \sum_{m=0}^{7} T_{2m} \sum_{n=0}^{7} T_{0n} X_{nm} & 0 & \cdots & 0 \\ \sum_{m=0}^{7} T_{0m} \sum_{n=0}^{7} T_{1n} X_{nm} & \sum_{m=0}^{7} T_{1m} \sum_{n=0}^{7} T_{1n} X_{nm} & 0 & \cdots & \cdots & 0 \\ \sum_{m=0}^{7} T_{0m} \sum_{n=0}^{7} T_{2n} X_{nm} & 0 & 0 & \cdots & \cdots & 0 \\ 0 & \vdots & \vdots & \ddots & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix} \tag{10}$$

Algorithm 1 ZLCADCT Algorithm

```
procedure ZLCADCT (X, TCR)
    'm' ← TCR range
    T_p ← T_p m×8
for j ←1, 8 do
    X1(:, j) ← T_p · X(:, j)
end for
for i ←1, m do
    'k' ← TCR ∨ 'i' ∈ range in LUT
    T_p ← T_p k×8
    Y(i, :) ← X1(i, :) · T_p^T
end for
    return Y
end procedure
```

3.1. Compute every column of X by T_p (using the fast butterfly algorithm) in turn and assign a 0 to the not required coefficients; then save the results to matrix $X1$.

3.2. Go back to step 3.1 until the calculation of each sub-block X is finished.

4. For the matrix $X1$

4.1. For every 'm' rows

4.1.1. Determine the range that covers the value of *TCR* by using the LUT in Table 2.

4.1.2. According to the range, find the corresponding 'k' value as the number of retained coefficients.

4.1.3. Adjust the T_p to $k \times 8$ size and set 0 to the coefficients that are not required to be calculated.

4.1.4. Go back to step 4.1.1 until $Y = X1 \cdot T_p^T$ is completed

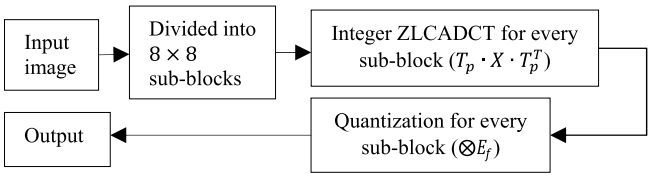


Fig. 7. Computational flow diagram for 8×8 ZLCADCT.

5. Quantize every sub-block matrix Y to find the final results.

6. End of processing for the current image.

$$T_p \cdot X = \begin{pmatrix} \sum_{n=0}^{7} T_{0n} X_{n0} & \sum_{n=0}^{7} T_{0n} X_{n1} & \cdots & \sum_{n=0}^{7} T_{0n} X_{n7} \\ \sum_{n=0}^{7} T_{1n} X_{n0} & \sum_{n=0}^{7} T_{1n} X_{n1} & \cdots & \sum_{n=0}^{7} T_{1n} X_{n7} \\ \sum_{n=0}^{7} T_{2n} X_{n0} & \sum_{n=0}^{7} T_{2n} X_{n1} & \cdots & \sum_{n=0}^{7} T_{2n} X_{n7} \end{pmatrix} \tag{7}$$

Without using LUT: (8), as shown at the top of this page.

Using LUT: (9) and (10), as shown at the top of this page.

B. Mathematical Modeling

Consider the traditional DCT when computed on 8×8 image blocks. The DCT matrix $T(T_{0,0}, \cdots T_{i,j})$ and the input block matrix X $(X_{0,0}, \cdots X_{i,j})$ are given by (11); then the DCT results in a matrix $Y(Y_{0,0}, \cdots Y_{i,j}) = T \cdot X \cdot T^T$ with coefficients given in (12), as shown at the top of the next page. The zigzag scanning (Fig. 3) is then performed to select the coefficients from (12) for

$$T = \begin{pmatrix} T_{00} & T_{01} & \cdots & T_{07} \\ T_{10} & T_{11} & \cdots & T_{17} \\ \vdots & \vdots & \ddots & \vdots \\ T_{70} & T_{71} & \cdots & T_{77} \end{pmatrix} \quad X = \begin{pmatrix} X_{00} & X_{01} & \cdots & X_{07} \\ X_{10} & X_{11} & \cdots & X_{17} \\ \vdots & \vdots & \ddots & \vdots \\ X_{70} & X_{71} & \cdots & X_{77} \end{pmatrix} \tag{11}$$

$$Y = T \cdot X \cdot T^T = \begin{pmatrix} \sum_{m=0}^{7} T_{0m} \sum_{n=0}^{7} T_{0n} X_{nm} & \sum_{m=0}^{7} T_{1m} \sum_{n=0}^{7} T_{0n} X_{nm} & \cdots & \sum_{m=0}^{7} T_{7m} \sum_{n=0}^{7} T_{0n} X_{nm} \\ \sum_{m=0}^{7} T_{0m} \sum_{n=0}^{7} T_{1n} X_{nm} & \sum_{m=0}^{7} T_{1m} \sum_{n=0}^{7} T_{1n} X_{nm} & \cdots & \sum_{m=0}^{7} T_{7m} \sum_{n=0}^{7} T_{1n} X_{nm} \\ \vdots & \vdots & \ddots & \vdots \\ \sum_{m=0}^{7} T_{0m} \sum_{n=0}^{7} T_{7n} X_{nm} & \sum_{m=0}^{7} T_{1m} \sum_{n=0}^{7} T_{7n} X_{nm} & \cdots & \sum_{m=0}^{7} T_{7m} \sum_{n=0}^{7} T_{7n} X_{nm} \end{pmatrix} \tag{12}$$

where

$$Y_{ij} = \sum_{m=0}^{7} T_{jm} \sum_{n=0}^{7} T_{in} X_{nm} \quad \text{for } i, j = 0 \ldots 7 \tag{13}$$

data storage or transmission (see (11)-(13) at the top of this page).

In ZLCADCT, instead of T, a new matrix T_p is first generated based on the number of retained coefficients. This new matrix is designed to eliminate those calculations performed on the unused (not retained) coefficients. The *Non-Zero Partition* (*NZP*) is defined here as the partition of the new output matrix Y in which a non-zero multiplication of $Y = T_p \cdot X \cdot {T_p}^T$ takes place, i.e., *NZP* does not contain those zero multiplications that always results in zero vectors, rows or columns.

Remark 1: The partition matrix *NZP* is a square matrix, i.e., of $m \times m$ size, where $1 \leq m \leq 8$. Hence, $ACR = m^2$. For example, for a *TCR* of value 10, *NZP* is of size 4×4, or $m = 4$, and *ACR* will be of value 16.

Next, consider the first multiplication operation to calculate Y, i.e., $X1 = T \cdot X$. The matrix T is replaced by the matrix T_p of $m \times 8$ size. Therefore, the new $X1 = T_p \cdot X$ has 8 columns (denoted by $X1_{i,0}, X1_{i,1}, \cdots, X1_{i,7}$) that contains the necessary first m elements, i.e. $i = 0 \ldots m-1$. The remaining elements $i = m \ldots 7$ will all be zero by definition. Similarly, the number of final elements in each row $(Y_{0,j}, Y_{1,j}, \cdots, Y_{7,j})$ of Y is determined by the processing of $Y = X1 \cdot {T_p}^T$. The use of T_p instead of T allows each row of Y to retain only the first 'm' elements; as only 'm' rows are considered in $X1$, then only these 'm' rows should be used when calculating $Y = X1 \cdot T_p^T$.

Remark 2: Since 'm' is limited in the range $1 \leq m \leq 8$, then *TCR* can have eight ranges. Mathematically, $TCR \in \left[1 + \sum_{t=0}^{m-1} t, \sum_{t=0}^{m} t\right]$.

Table 1 shows the ranges that *TCR* takes depending on the value of 'm'. For example, if $m = 4$, *TCR* can take one of the values in the range from 7 to 10. The reciprocal relationship is also true, i.e., if $7 \leq TCR \leq 10$, then 'm' should be 4 and the size of *NZP* is 4×4. So, $Y = T_p \cdot X \cdot T_p^T$ contains at least $8 \times 8 - m \times m = 48$ coefficients of zero value. These 48 coefficients are not stored/transmitted and therefore they are not calculated in ZLCADCT. Table 1 is then used to obtain the value of 'm' for determining the size of T_p, i.e., $m \times 8$ once the *TCR* is chosen.

Considering Table 1, for example, when $m = 1$, the resulting T_p has one row, i.e. $T_p = T_{pi,0\ldots7} = [T_{p0,0}, T_{p0,1}, \cdots, T_{p0,7}] = [1, 1, 1, 1, 1, 1, 1, 1]$ where $i = 0$. Then for a given input matrix X, the element of the output vector $Z_{i,j}$ for each column (j-th) of the output matrix $X1$ is given by $Z_{m-1,j}(m) = Z_{0,j}(1) = T_{p0,0} \cdot X_{0,j} + T_{p0,1} \cdot X_{1,j} + \cdots + T_{p0,7} \cdot X_{7,j} = X_{0,j} + X_{1,j} + \cdots + X_{7,j}$ where $j = 0 \ldots 7$. Similarly, when $m = 5$, T_p has five rows, and for each column (j-th) of $X1$, five output vector elements $Z_{0\ldots m-1,j}(1 \ldots m)$ ($Z_{0,j}(1)$ to $Z_{4,j}(5)$) are created. For example considering the fifth row of T_p, ($T_{p\ 4,0\ldots7} = \left[T_{p\ 4,0}, T_{p\ 4,1}, \cdots, T_{p\ 4,7}\right] = [1, -1, -1, 1, 1, -1, -1, 1]$). The fifth output element is given by $Z_{m-1,j}(m) = Z_{4,j}(5) = T_{p4,0} \cdot X_{0,j} + T_{p4,1} \cdot X_{1,j} + \cdots + T_{p4,7} \cdot X_{7,j} = X_{0,j} - X_{1,j} - X_{2,j} + X_{3,j} + X_{4,j} - X_{5,j} - X_{6,j} + X_{7,j}$. By generalizing the above two examples, the first term ($\sum_{t=0}^{3} \left[\left(X_{t,j} + X_{7-t,j}\right) \cdot (-1)^{\left\lfloor log_2^{t+1} \right\rfloor \cdot \left(\frac{m-1}{4}\right)} \right]$) of (15) is obtained. $\sum_{t=0}^{3} \left[\left(X_{t,j} + X_{7-t,j}\right)\right]$ is used to determine the input elements, while $(-1)^{\left\lfloor log_2^{t+1} \right\rfloor \cdot \left(\frac{m-1}{4}\right)}$ is used to determine the sign of the input elements according to $T_{p\ m-1,0\ldots7}$ (the m-th row of T_p). In general, the calculation of the output vector $Z_{i,j}$ ($i = 0 \ldots m-1$ and $j = 0 \ldots 7$) for each column (j-th) of $X1$ (given by (14)) requires the general expression given by (15). In (15), $\left\lfloor log_2^{t+1} \right\rfloor$ denotes the floor function for the maximum integer that is not larger than log_2^{t+1}.

$$X1_{m\times8} = T_{p\ m\times8} \cdot X_{8\times8} = \begin{pmatrix} Z_{0,0}(1) & Z_{0,1}(1) & \cdots & Z_{0,7}(1) \\ Z_{1,0}(2) & Z_{1,1}(2) & \cdots & Z_{1,7}(2) \\ \vdots & \vdots & \ddots & \vdots \\ Z_{m-1,0}(m) & Z_{m-1,1}(m) & \cdots & Z_{m-1,7}(m) \end{pmatrix} \tag{14}$$

Eq. 15 calculates the j-th column vector of $X1 = T_p \cdot X = \left[X1_{0,j}, X1_{1,j}, \cdots, X1_{m-1,j}\right]$ by using the $m \times 8$ matrix T_p to process j-th column vector of input X, i.e. $[X_{0,j}, X_{1,j}, \cdots, X_{7,j}]$, and $X1 = \left[Z_{0,j}(1), Z_{1,j}(2), \ldots, Z_{m-1,j}(m)\right]$ is finally produced. All eight column vectors of X are processed using (15), as shown at the top of the next page, and the final $m \times 8$ matrix $X1$ is shown in (14).

$$Z_{m-1,j}(m)=\begin{cases}\sum_{t=0}^{3}\left[\left(X_{t,j}+X_{7-t,j}\right)\cdot(-1)^{\left\lfloor log_2^{t+1}\right\rfloor\cdot\left(\frac{m-1}{4}\right)}\right], & m\in\{1,5\}\\ \begin{pmatrix}X_{\left\lfloor log_6^{m-1}\right\rfloor,j}-X_{3-\left\lfloor log_6^{m-1}\right\rfloor,j}\\ -X_{4+\left\lfloor log_6^{m-1}\right\rfloor,j}+X_{7-\left\lfloor log_6^{m-1}\right\rfloor,j}\end{pmatrix}\cdot(-1)^{\left\lfloor log_6^{m-1}\right\rfloor}, & m\in\{3,7\}\\ X_{log_2^{6-m}-1,j}-X_{8-log_2^{6-m},j}, & m\in\{2,4\}\\ X_{log_2^{10-m}+1,j}-X_{6-log_2^{10-m},j}, & m\in\{6,8\}\end{cases}\tag{15}$$

$$Y_{i,k_m-1}(k_m)=\begin{cases}\sum_{t=0}^{3}\left[\left(Z_{i,t}+Z_{i,7-t}\right)\cdot(-1)^{\left\lfloor log_2^{t+1}\right\rfloor\cdot\left(\frac{k_m-1}{4}\right)}\right], & k_m\in\{1,5\}\\ \begin{pmatrix}Z_{i,\left\lfloor log_6^{k_m-1}\right\rfloor}-Z_{i,3-\left\lfloor log_6^{k_m-1}\right\rfloor}\\ -Z_{i,4+\left\lfloor log_6^{k_m-1}\right\rfloor}+Z_{i,7-\left\lfloor log_6^{k_m-1}\right\rfloor}\end{pmatrix}\cdot(-1)^{\left\lfloor log_6^{k_m-1}\right\rfloor}, & k_m\in\{3,7\}\\ Z_{i,log_2^{6-k_m}-1}-Z_{i,8-log_2^{6-k_m}}, & k_m\in\{2,4\}\\ Z_{i,log_2^{10-k_m}+1}-Z_{i,6-log_2^{10-k_m}}, & k_m\in\{6,8\}\end{cases}\tag{17}$$

The resulting $m\times m$ *NZP* matrix of $Y=T_p\cdot X\cdot {T_p}^T$ is given in Table 2 (gray highlighted) for $TCR=6$, then 'm' is 3 and *ACR* is 9. The remaining 55 coefficients are not calculated. However, a conventional DCT calculation (using the matrix T in (11) and (12)) requires calculating all 64 coefficients for the entire 8×8 matrix and then retaining the 6 coefficients (after performing zigzag scanning). Therefore, a considerable improvement is accomplished by generating only the *NZP* matrix; so, the scanning process for transmitting/storing the selected coefficients can be removed.

By definition, the number of coefficients (9 coefficients) gray highlighted in Table 2 is not always the same as those generated using ADCT [7] with a zigzag scanning (gray highlighted with right slashes). The *NZP* matrix generates additional terms, except for the case when the number of retained coefficients is one ($m=1$). Therefore, the use of the *NZP* matrix may incur in additional overhead.

Remark 3: For a size 'm' block, the *TCR* range is constant and therefore Table 2 can be used as a LUT, so the unnecessary coefficients found for $Y=X1\cdot {T_p}^T$ can be entirely avoided.

To remove the additional terms generated by *NZP* matrix during the process of calculating $Y=X1\cdot {T_p}^T$, a LUT (Table 2) is introduced by ZLCADCT. The size of T_p is determined by the number of retained coefficients $k\in\{k_1,k_2,\cdots k_m\}$ for the different 'm' row vectors. For example, when $k=1$, for the i-th row vector of input $X1$, T_p has only one row, i.e. $T_p=T_{p\ 0,0\ldots 7}=\left[T_{p\ 0,0},T_{p\ 0,1},\ldots,T_{p\ 0,7}\right]=[1,1,1,1,1,1,1,1]$. Therefore, the output vector $Y_{i,j}$ for the i-th row is given by only one vector element $Y_{i,k-1}(k)=Y_{i,0}(1)=T_{p0,0}\cdot Z_{i,0}+T_{p0,1}\cdot Z_{i,1}+\cdots+T_{p0,7}\cdot Z_{i,7}=Z_{i,0}+Z_{i,1}+\cdots+Z_{i,7}$, where $0\le i\le m-1$ *and* $j=0$. Similarly, when $k=5$, T_p has five rows, and five output vector elements $Y_{i,k-1}(k)$ ($Y_{i,0}(1)$ *to* $Y_{i,4}(5)$) are generated for the i-th row. Therefore, for the fifth row of T_p, $T_{p\ 4,0\ldots 7}=\left[T_{p\ 4,0},T_{p\ 4,1},\cdots,T_{p\ 4,7}\right]=[1,-1,-1,1,1,-1,-1,1]$, and the fifth output element $Y_{i,k-1}(k)=Y_{i,4}(5)=T_{p4,0}\cdot Z_{i,0}+T_{p4,1}\cdot Z_{i,1}+\cdots+T_{p4,7}\cdot Z_{i,7}=Z_{i,0}-Z_{i,1}-Z_{i,2}+Z_{i,3}+Z_{i,4}-Z_{i,5}-Z_{i,6}+Z_{i,7}$ can be found. By generalizing the above expressions leads to the first term ($\sum_{t=0}^{3}\left[\left(Z_{i,t}+Z_{i,7-t}\right)\cdot(-1)^{\left\lfloor log_2^{t+1}\right\rfloor\cdot\left(\frac{k_m-1}{4}\right)}\right]$) of (17), as shown at the top of this page. $\sum_{t=0}^{3}\left[\left(Z_{i,t}+Z_{i,7-t}\right)\right]$ is used to determine the input elements while $(-1)^{\left\lfloor log_2^{t+1}\right\rfloor\cdot\left(\frac{k_m-1}{4}\right)}$ is used to determine the sign of the input elements according to $T_{p\ k-1,0\ldots 7}$ (the k-th row of T_p). In general, the calculation of the output vector $Y_{i,j}$ ($i=0\ldots m-1$ and $j=0\ldots k-1$) for each (i-th) row of Y (given by (16)) requires the general expression given by (17).

$$\begin{aligned}Y&=X1_{m\times 8}\cdot {T_p}^T_{\ k_m\times 8}\\&=\begin{pmatrix}Y_{0,0}(1) & Y_{0,1}(2) & \cdots & Y_{0,k_1-1}(k_1)\\ Y_{1,0}(1) & Y_{1,1}(2) & \cdots & Y_{1,k_2-1}(k_2)\\ \vdots & \vdots & \ddots & \vdots\\ Y_{m-1,0}(1) & Y_{m-1,1}(2) & \cdots & Y_{m-1,k_m-1}(k_m)\end{pmatrix}\end{aligned}\tag{16}$$

where,

$$k\in\{k_1,k_2,\cdots k_m\}\quad\text{and } k\le m\tag{18}$$

Eq. (17) calculates the i-th row vector of $Y=X1\cdot {T_p}^T=\left[Y_{0,0},Y_{1,1},\cdots,Y_{m-1,k-1}\right]$ by using the $k\times 8$ matrix T_p to process the i-th vector of the input $X1$, i.e. $\left[Z_{i,0},Z_{i,1},\cdots,Z_{i,7}\right]$, and then output matrix Y is finally produced by $Y=[Y_{0,0\ldots k_1-1}(1\ldots k_1),Y_{1,0\ldots k_2-1}(1\ldots k_2),\cdots,Y_{m-1,0\ldots k_m-1}(1\ldots k_m)]$. All '$m$' row vectors of $X1$ are processed using (17) and the final matrix Y obtained is shown in (16).

IV. Performance and Complexity

In this section, the analysis of the proposed new schemes for DCT is presented with respect to performance and computational complexity; comparison with previous DCT schemes is also pursued.

A. Performance

The performance of ZLCADCT is evaluated by employing the PSNR (Peak Signal to Noise Ratio) as metric [17]; the PSNR is used to assess the quality of processed images using (19) and (20).The *Mean Square Error* (*MSE*) (defined in (20)) is first calculated and evaluates the variation of pixel value between two images; $p_{i,j}$ refers to the pixel value of the original image of size $a \times b$, while $\hat{p}_{i,j}$ is the pixel value for the final processed image.

$$PSNR = 10log\frac{(2^n-1)^2}{MSE} \tag{19}$$

where:

$$MSE = \frac{1}{a \times b}\sum_{i=1}^{a}\sum_{j=1}^{b}(p_{i,j} - \hat{p}_{i,j})^2 \tag{20}$$

Based on (19) and (20), both ADCT [7] and ZLCADCT generates the same output matrix (i.e. (10)). So, the values of $\hat{p}_{i,j}$ for both these two methods are identical and therefore, both the *MSE* and the PSNR have the same values.

However, for ADCT [27], the additional non-zero coefficients $\hat{p}_{i,j}$(eg. those coefficients highlighted in bold font in (8)) will cause the difference $p_{i,j} - \hat{p}_{i,j}$ for these pixels to be smaller than $p_{i,j} - 0$, because the values of these coefficients for both ADCT [7] and ZLCADCT are all zero. Therefore, the *MSE* of ADCT [27] will be smaller than ADCT [7] and ZLCADCT, and the PSNR of [27] will in turn be higher than for these two methods.

B. Complexity

Regarding computational complexity, ZLCADCT removes the unnecessary additions from the process of calculating both $X1 = T_p \cdot X$ and $Y = X1 \cdot T_p^T$ so making their executions faster than ADCT [7] and [27]. In the ZLCADCT, for processing of $X1 = T_p \cdot X$, each column of $X1$ is calculated using $14 - (8 - m) = m + 6$ additions ($1 \leq m \leq 8$); so, the total number of additions required by $X1$ is $8\,(m + 6) = 8m + 48$. Next, for processing $Y = X1 \cdot {T_p}^T$, each non-zero row of $X1$ must be checked from LUT. If one row is required to retain 'k' ($1 \leq k < m$) rather than 'm' coefficients, then the number of additions for processing this row can be reduced from $m+6$ to $k+6$ by removing the unnecessary (redundant) additions. For example, if *TCR* is 6 (as explained previously in Section 3A), the number of additions required by using LUT for calculating the second row of $X1$ decreases from 9 to 8 ($k = 2$) and from 9 to 7 ($k = 1$) for the third row. Thus, the total number of additions required for computing $T_p \cdot X \cdot T_p^T$ using ZLCADCT is 96, a further saving of 3 additions when compared with [27]. The generic equation to determine the number of additions required by ZLCADCT is given in (21), as shown at the bottom of this page; note that NA is dependent on both 'm' and *TCR*.

Where the value of 'm' can be obtained according to the *TCR* value. The range covering a *TCR* can be found in Table 1. $\left\lceil 1 - TCR/\sum_{t=0}^{m} t \right\rceil$ denotes the ceiling function for the least integer that is not smaller than $1 - TCR/\sum_{t=0}^{m} t$.

The total number of additions required by [27] is given by (22). As per *Remark 2* in Section 3B, the largest *TCR* value for a specific 'm' is $\sum_{t=0}^{m} t$. Thus, as $-2\sum_{t=0}^{m-1} t + TCR = -2\sum_{t=0}^{m-1} t + \sum_{t=0}^{m} t = m - \sum_{t=0}^{m-1} t = m - m\,(m-1)/2 = (m - m^2)/2 \leq 0$ in (21), then $NA_{ZLCADCT}$ will be smaller than $NA_{ADCT[27]}$.

$$NA_{ADCT[27]}(m) = m^2 + 14m + 48 \tag{22}$$

For ADCT [7], $T \cdot X \cdot T^T$ for an 8×8 input block requires executing 16 times the full fast butterfly algorithm that in turns requires 14 additions each time, so resulting in a total of 224 additions for any value of *TCR*. As explained above, compare with [7], if $TCR = 6$, ADCT [27] requires 99 additions and ZLCADCT requires the least (96 additions).

V. Evaluation by Simulation

In this section, an evaluation of the proposed techniques is pursued with respect to image compression. Initially, an image (Lena) is processed by ADCT [7], ADCT [27] and ZLCADCT for different numbers of retained coefficients. For evaluation purposes, the selection of the rows to be pruned for the random technique of [27] is executed using the proposed technique.

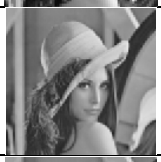
Table 3 compares ADCT [7], ADCT [27] and ZLCADCT in terms of output images for Lena when the numbers of retained coefficients are 1, 3, 6, 10, 15, 21 and 28. Metrics such as the number of additions, energy consumption and PSNR of the output image for the entire computation of $T \cdot X \cdot T^T$ are determined; the results are presented in Table 4. The number of coefficients for transmission/storage increases by using [27], so ZLCADCT is also evaluated to keep the number of transmitted coefficients the same as in [7]. In Table 4, Column 1 shows the number of *TCRs* (i.e., *TCR*); Column 2 reports the number of *ACRs* for [27]. Column 3 shows the number of retained rows for DCT matrix. Columns 4 to 6 show the number of additions required by ADCT [7], ADCT [27], and ZLCADCT. The number of additions required by ZLCADCT is significantly smaller (28% for a single retained coefficient) at a low number of retained coefficients compared with ADCT [7]; and it can be even lower than ADCT [27]. By increasing the number of retained coefficients, the difference between the numbers of additions by ZLCADCT and ADCT [7] decreases.

As hardware, a 32-bit Ripple Carry Adder (*RCA*) (made of mirror full adder cells) is used to perform the addition. The eight input cases (i.e., the inputs A, B and Cin: from 000 to 111) of an exhaustive simulation are applied to a 32-nm single full adder cell. LTSPICE is used to establish the average energy consumption; the average energy consumption of a single full adder is found to be 3.58492E-16 J. The energy consumption of DCT using the matrix T_p is given by using the

$$NA_{ZLCADCT}(m, TCR) = \begin{cases} m^2 + 13m - 2\sum_{t=0}^{m-1} t + TCR + 48, & m \in \{1,3,5,7\} \\ m^2 + 13m - 2\sum_{t=0}^{m-1} t + TCR + 48 - 6\left\lceil 1 - \frac{TCR}{\sum_{t=0}^{m} t} \right\rceil, & m \in \{2,4,6,8\} \end{cases} \tag{21}$$

TABLE III

OUTPUT IMAGES OF ADCT [7], ADCT [27] AND ZLCADCT FOR DIFFERENT NUMBER OF RETAINED COEFFICIENTS

TCR	ADCT [27]	ADCT [7]or ZLCADCT
1		
3		
6		
15		
21		
28		

total number of additions in the butterfly algorithm; Columns 7 to 9 of Table 4 show the energy consumption of ADCT [7], ADCT [27] and ZLCADCT. Significant savings in energy consumption is achieved by ZLCADCT at a lower number of retained coefficients; as previously when the number of coefficients increases, also the energy consumption gradually increases.

As a measure of performance, the delay must also be assessed. The delay of the butterfly algorithm is determined by its critical path full adder at each level (so not depending on the number of adders in each level of addition); for the proposed matrix T_p, the critical path of the butterfly algorithm is always along the 3 additions. Therefore, when the number of retained coefficients increases, the delay of DCT using the proposed matrix T_p depends only on the 3 level additions. Columns 10, 11 and 12 of Table 4 show the software-based delay evaluation by Matlab. For any *TCR* value, ZLCADCT has the lowest delay because its required calculation has the least complexity compared with ADCT [7] and [27]. For both ZLCADCT and ADCT [27], the delay is increased as *TCR* increases, while the delay of ADCT [7] is unchanged.

Next, PSNR values of output images for ADCT [7], ADCT [27] and ZLCADCT are considered. The Table 5 shows the values of the PSNR obtained for different *TCR*s when applied on five input images (lena, cameraman, baboon, eagle and moon) and processed using ADCT [7], ADCT [27] and ZLCADCT. As it can be observed from the results, when only one coefficient is retained, the outputs of ZLCADCT and ADCT [27] are the same; therefore, their PSNR values are equal only in this case. However, when the number of retained coefficients increases, the PSNR of ADCT [27] can be higher than ADCT [7] and ZLCADCT because they retain more coefficients than ADCT [7] using zigzag scanning and ZLCADCT. For example, when 6 coefficients are required for compressing images, only 6 coefficients are retained using ADCT [7] (after zigzag scanning) and ZLCADCT, but ADCT [27] retains three more coefficients (highlighted by a bold font in (8)). ADCT [27] can have a marginally higher PSNR due to the additional elements in the resulting matrix. However, by using ZLCADCT, the additional elements are avoided and ZLCADCT has a PSNR whose value is the same as ADCT [7]. Moreover, when the number of retained coefficients is larger than 1, this scheme reduces the number of additions and energy consumption. Next, the average PSNR value (rows 6 and 13) and its percentage (rows 7 and 14) for different *TCR* shown in Table 5 can be used to determine the desired value of the *TCR*depending on the percentage of the PSNR required. The percentage values are calculated by using average PSNR values divided by the PSNR value of $TCR = 63$. For example, using ZLCADCT approach, if the output image quality is required to be at 60% of the PSNR value then *TCR* value of 15 has to be selected.

VI. EXPERIMENTAL EVALUATION

ZLCADCT is based on preprocessing T and it is also modularly configurable; for example, one row of the T_p matrix and the resulting computation in the butterfly algorithm are required when retaining a coefficient, this matrix is referred hereafter as the base matrix; subsequently for every increase in the number of retained coefficients, the matrix T_p has additional rows compared to the previous T_p matrix and the corresponding calculations must be performed. For example, when 2-3 coefficients are retained, an additional row is added to the base T_p matrix and a few more additions are needed in the base butterfly algorithm. So, when a new row is included in the matrix T_p, the number of additions in the butterfly algorithm is increased by 1. This feature is exploited when an implementation is considered because ZLCADCT requires to selectively switch the hardware as a function of the number of retained coefficients, thus resulting in a reduced number of additions. In this paper, ZLCADCT is implemented in hardware by using an FPGA-based platform. Fig. 8 shows the block diagram of implementing $Y = T_p \cdot X \cdot T_p^T$ in such FPGA platform. The FPGA platforms used in the experiments are the Spartan 3E and Spartan 6 XC6SLX45 at a clock speed of 50MHz and 100MHz respectively. The Xilinx ISE 14.7 is used for implementing the design to the FPGAs; the RS232 serial data transmission (*DT*) baud rate is set to 9600.

TABLE IV

COMPARISON AMONG ADCT [7], ADCT [27] AND ZLCADCT WHEN COMPUTING $T \cdot X \cdot T^T$ FOR DIFFERENT *TCR* FOR 512 × 512 LENA AS AN INPUT IMAGE

TCR	*ACR*	*m*	Number of additions			Estimated Energy consumption(J)			Matlab implementation Delay(s)		
			ADCT [7]	ADCT [27]	ZLCADCT	ADCT [7]	ADCT [27]	ZLCADCT	ADCT [7]	ADCT [27]	ZLCADCT
1	1	1	224	63	63	2.56967E-12	7.22719E-13	7.22719E-13	0.89174	0.66657	0.64775
3	4	2	224	80	79	2.56967E-12	9.17739E-13	9.06267E-13	0.89174	0.70945	0.69138
6	9	3	224	99	96	2.56967E-12	1.1357E-12	1.10129E-12	0.89174	0.75237	0.71599
10	16	4	224	120	114	2.56967E-12	1.37661E-12	1.30778E-12	0.89174	0.76163	0.74088
15	25	5	224	143	133	2.56967E-12	1.64046E-12	1.52574E-12	0.89174	0.79125	0.75129
21	36	6	224	168	153	2.56967E-12	1.92725E-12	1.75517E-12	0.89174	0.82011	0.79121
28	49	7	224	195	174	2.56967E-12	2.23699E-12	1.99608E-12	0.89174	0.87752	0.84789

TABLE V

PSNR OF FOUR DIFFERENT INPUT IMAGES COMPUTED BY ADCT [7], ADCT [27] AND ZLCADCT FOR DIFFERENT *TCR*

Applied measure	PSNR / Input images	***TCR***						
		1	**3**	**6**	**10**	**15**	**21**	**28**
ADCT [27]	Lena	23.665	24.6831	25.5064	29.0783	29.5925	30.2545	31.1538
	Cameraman	19.4785	20.319	21.1865	23.5678	24.4284	25.284	26.6829
	baboon	19.7051	20.1917	20.9795	22.4631	23.5461	24.6208	26.2839
	eagle	19.6944	20.6155	21.4448	24.3211	24.9546	25.7347	26.5863
	moon	25.4545	26.5577	27.3203	31.3366	31.8525	32.6273	33.3232
	Average value	21.5995	22.4734	23.2875	26.15338	26.87482	27.70426	28.80602
	Average (%)	50.30%	52.34%	54.23%	60.91%	62.59%	64.52%	67.09%
ADCT [7] or ZLCADCT	Lena	23.665	24.6146	25.2641	27.9908	28.7141	29.4039	30.2238
	Cameraman	19.4785	20.2603	20.9831	22.792	23.5013	24.2461	25.2993
	baboon	19.7051	20.1185	20.6814	21.708	22.4431	23.2003	24.1371
	eagle	19.6944	20.5464	21.1677	23.3922	24.1524	24.8455	25.5907
	moon	25.4545	26.5051	27.1862	30.4515	31.1244	31.9182	32.657
	Average value	21.5995	22.40898	23.0565	25.2669	25.98706	26.7228	27.58158
	Average (%)	50.30%	52.19%	53.70%	58.84%	60.52%	62.24%	64.24%

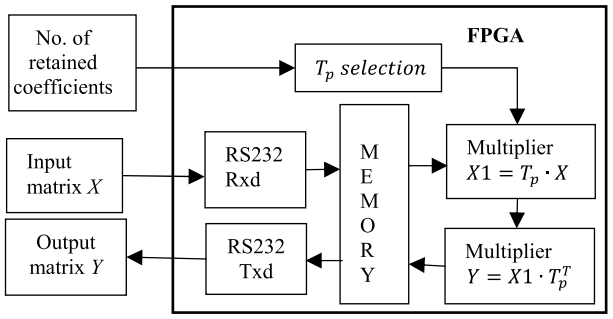


Fig. 8. Block diagram of the proposed FPGA platform for implementing $Y = T_p \cdot X \cdot T_p^T$.

The hardware platform requires two inputs (input pixel image and the number of the coefficients to be retained) and generates one output (the compressed image). First, the input pixel image (in this case the image of Lena) is segmented as several 8 × 8 sub-blocks in Matlab. Then, every sub-block is transmitted serially to the hardware using RS232; they are stored in the memory block. Meanwhile, the 'T_p *selection*' block determines the 'm' value and the 'k' value for T_p depending on proposed LUT and the number of retained coefficients (*TCR*) received by the hardware. Two modular multipliers ($X1 = T_p \cdot X$ *and* $Y = X1 \cdot T_p^T$) which are based on 32-bits RCAs are utilized for calculating $T_p \cdot X \cdot T_p^T$ according to the 'm' and 'k' values respectively . Modularity here refers to the design capability with respect to *TCR*, i.e., based on *TCR*, the size of the multipliers is adjusted without recurring in the full reconfiguration of the FPGAs. This is achieved by using 'm' and 'k' values on the fast butterfly algorithm; furthermore, in the hardware architecture of fast butterfly algorithm for two proposed modular multipliers, adders at the same stage for processing the input signals normally are designed to work in parallel. For example, in the hardware design of $X1 = T_p \cdot X$ (Fig.2), there are three stages in total to create $A_0, A_1, \ldots, A_7$, $B_0, B_1, \ldots, B_7$ and $C_0, C_1, \ldots, C_7$ successively. The required adders at the same stage are designed to work in parallel (such as 8 adders for creating $A_0, A_1, \ldots, A_7$ work in parallel instead of sequential operations) for improving the performance in processing speed. Finally, the output of $Y = T_p \cdot X \cdot T_p^T$ is stored in memory and then serially transmitted back to the host through RS232 and processed by Matlab for quantization; decompression and the measurement of the PSNR are finally performed by Matlab.

Using one sub-block of 512 × 512 Lena image as input, Table 6 compares ADCT [7], ADCT [27] and ZLCADCT in terms of resource utilization (adders, XOR gates and LUTs), power consumption (including quiescent power and dynamic power consumptions) and delay when Spartan 3E FPGA is utilized in the proposed platform. However, for determining the PSNR, 4096 sub-blocks of Lena image are used. The

TABLE VI

COMPARISON OF ADCT [7], ADCT [27] AND ZLCADCT USING SPARTAN 3E FPGA AND ADCT [7] MATRIX FOR LENA INPUT IMAGE

SPARTAN 3E 1600E												
Resources/ Metrics	**ADCT [7]**				**ADCT [27]**				**ZLCADCT**			
	TCR=1	*TCR*=3	*TCR*=10	*TCR*=21	*TCR*=1	*TCR*=3	*TCR*=10	*TCR*=21	*TCR*=1	*TCR*=3	*TCR*=10	*TCR*=21
Adders	129	129	129	129	10	21	49	85	10	20	43	70
XOR gates	14336	14336	14336	14336	4032	5120	7680	10752	4032	5056	7296	9792
LUTs	12519	12519	12519	12519	1280	2146	4643	8371	1280	2046	4035	6666
Power consumption (mw)	260	260	260	260	229	232	238	248	229	231	236	244
Implementation time (in sec)	276	276	276	276	83	84	104	170	84	90	112	152
Total processing time (in sec)	1.0528	1.0528	1.0528	1.0528	0.73949	0.75348	0.81013	0.91873	0.72013	0.74947	0.77127	0.83272
PSNR for 512 × 512 Lena	23.665	24.614	27.990	29.403	23.665	24.683	29.078	30.254	23.665	24.614	27.990	29.403

TABLE VII

COMPARISON OF ADCT [7], ADCT [27] AND ZLCADCT USING SPARTAN 6 FPGA AND ADCT [7] MATRIX FOR LENA INPUT IMAGE

SPARTAN 6 XC6SLX45												
Resources/ Metrics	**ADCT[7]**				**ADCT [27]**				**ZLCADCT**			
	TCR=1	*TCR*=3	*TCR*=10	*TCR*=21	*TCR*=1	*TCR*=3	*TCR*=10	*TCR*=21	*TCR*=1	*TCR*=3	*TCR*=10	*TCR*=21
Adders	130	130	130	130	2	12	38	72	2	11	32	57
XOR gates	14336	14336	14336	14336	4032	5120	7680	10752	4032	5056	7296	9792
LUTs	11128	11128	11128	11128	1971	2888	4394	7894	1971	2636	4279	6546
Power consumption (mw)	106	106	106	106	64	67	73	91	64	65	72	84
Implementation time (in sec)	210	210	210	210	60	81	112	144	71	80	111	135
Total processing time (in sec)	1.0491	1.0491	1.0491	1.0491	0.72896	0.73139	0.79558	0.89449	0.72932	0.73087	0.76122	0.82051
PSNR for 512 × 512 Lena	23.665	24.6146	27.9908	29.4039	23.665	24.6831	29.0783	30.2545	23.665	24.6146	27.9908	29.4039

TABLE VIII

COMPARISON OF ADCT [4], ADCT [27] AND ZLCADCT USING SPARTAN 3E FPGA AND ADCT [4] MATRIX FOR LENA INPUT IMAGE

SPARTAN 3E 1600E												
Resources/ Metrics	**ADCT [4]**				**ADCT [27]**				**ZLCADCT**			
	TCR=1	*TCR*=3	*TCR*=10	*TCR*=21	*TCR*=1	*TCR*=3	*TCR*=10	*TCR*=21	*TCR*=1	*TCR*=3	*TCR*=10	*TCR*=21
Adders	145	145	145	145	10	31	61	99	10	29	54	83
XOR gates	16384	16384	16384	16384	4032	6400	9216	12544	4032	6208	8704	11456
LUTs	16003	16003	16003	16003	1280	3115	5914	9961	1280	2861	5146	8095
Power consumption (mw)	267	267	267	267	229	234	241	251	229	233	239	247
Implementation time (in sec)	770	770	770	770	78	92	133	174	77	81	118	168
Total processing time (in sec)	1.0611	1.0611	1.0611	1.0611	0.73751	0.74459	0.80116	0.92309	0.73645	0.73955	0.7603	0.82145
PSNR for 512 × 512 Lena	23.665	26.4785	29.9551	32.594	23.665	26.8968	31.6145	34.2788	23.665	26.4785	29.9551	32.594

other performance metrics reported in this table are defined as follows:

1. *Implementation time* refers to the total time taken to implement the design onto the FPGAs.
2. *Total processing time* is the sum of the multiplication and the serial data transmission times.

Compared to ADCT [7], hardware requirements (i.e., the numbers of adders, XOR gates and LUTs) show a significant decrease for any *TCR* using ADCT [27]; moreover, ZLCADCT

TABLE IX
COMPARISON OF ADCT [4], ADCT [27] AND ZLCADCT USING SPARTAN 6 FPGA AND ADCT [4] MATRIX FOR LENA INPUT IMAGE

SPARTAN 6 XC6SLX45												
Resources/ Metrics	**ADCT [4]**				**ADCT [27]**				**ZLCADCT**			
	TCR=1	*TCR*=3	*TCR*=10	*TCR*=21	*TCR*=1	*TCR*=3	*TCR*=10	*TCR*=21	*TCR*=1	*TCR*=3	*TCR*=10	*TCR*=21
Adders	146	146	146	146	2	22	50	86	2	20	43	70
XOR gates	16384	16384	16384	16384	4032	6400	9216	12544	4032	6208	8704	11456
LUTs	13138	13138	13138	13138	1971	3674	5420	8909	1971	3288	5187	7697
Power consumption (mw)	114	114	114	114	64	70	77	94	64	68	75	88
Implementation time (in sec)	660	660	660	660	61	89	111	174	62	80	103	136
Total processing time (in sec)	1.0433	1.0433	1.0433	1.0433	0.72732	0.73457	0.79664	0.90155	0.72645	0.73126	0.75933	0.81608
PSNR for 512 × 512 Lena	23.665	26.4785	29.9551	32.594	23.665	26.8968	31.6145	34.2788	23.665	26.4785	29.9551	32.594

TABLE X
AVERAGE REDUCTION (FOUR *TCR* CASES) OF HARDWARE RESOURCES/METRICS FOR ADCT [27] AND ZLCADCT COMPARED WITH ADCT [7] AND [4]

Resources/ Metrics	**Applied DCT**	**Average hardware resource reduction/performance metrics improvement**			
		Using ADCT matrix [7]		**Using ADCT matrix [4]**	
		SPARTAN 3E 1600E	**SPARTAN 6 XC6SLX45**	**SPARTAN 3E 1600E**	**SPARTAN 6 XC6SLX45**
Adder	ADCT [27]	68.02%	76.15%	65.34%	72.60%
	ZLCADCT	72.29%	80.38%	69.66%	76.88%
XOR gate	ADCT [27]	51.90%	51.90%	50.88%	50.88%
	ZLCADCT	54.35%	54.35%	53.61%	53.61%
LUT	ADCT [27]	67.17%	61.48%	68.33%	61.99%
	ZLCADCT	71.99%	65.33%	72.85%	65.48%
Power	ADCT [27]	8.94%	30.42%	10.58%	33.11%
	ZLCADCT	9.62%	32.78%	11.24%	35.31%
Implementation time	ADCT [27]	60.05%	52.74%	84.51%	83.52%
	ZLCADCT	60.33%	52.74%	85.58%	85.57%
Total processing time	ADCT [27]	23.49%	24.93%	24.46%	24.28%
	ZLCADCT	27.01%	27.51%	27.96%	27.32%

further reduces the hardware resources. Then, ZLCADCT requires the least power consumption compared to ADCT [7] and ADCT [27]. By increasing *TCR*, more coefficients must be generated and transmitted. Thus, the implementation time of ZLCADCT generally increases, also resulting in a modest increase of the total processing time (note that for ADCT [7] these times are constant as independent of *TCR*). As expected, the PSNR of ZLCADCT is the same as for ADCT [7], while ADCT [27] shows a marginally higher value of PSNR by increasing *TCR*.

Table 7 shows the results when ADCT [7], ADCT [27] and ZLCADCT are implemented and synthesized by using Spartan 6 FPGAs and Lena as an input image; also, in this case, the same trends as for the Spartan 3E FPGAs are reported.

The hardware performances of ZLCADCT and ADCT [27] are validated by using the approximate DCT matrix of [4]. Tables 8 and 9 show the result of different metrics when ADCT [4], ADCT [27] and ZLCADCT approaches are implemented in Spartan 3E and the Spartan 6 FPGAs (by setting the parameter 'a' to 0). The results show that the proposed method has better performance with respect to metrics such as number of gates, number of LUTs, power consumption, total processing time and PSNR.

Next, consider the average value of each of the hardware resource utilization and performance metrics across the four *TCRs* of Tables 6, 7, 8 and 9; the average percentage reductions and improvement for ADCT [27] and ZLCADCT (compared with ADCT [4], [7]) are presented in Table 10. Two implementations using different FPGAs (Spartan 3E and Spartan 6) are presented. From the results, it can be found that ZLCADCT outperforms ADCT [27].

VII. CONCLUSION

This paper has initially presented a zigzag low-complexity approximate DCT (ZLCADCT) approach that configures the T matrix by establishing the relationship between the number of retained coefficients and the number of rows of the T matrix. It significantly reduces the computational complexity of an approximate DCT scheme (such as ADCT [7]) and avoids the additional unnecessary coefficients generated by [27]. A detailed mathematical analysis of ZLCADCT has been presented to show the properties of T_p and a novel method for removing the additions required to process

the unused coefficients of a compressed image. A LUT is employed in ZLCADCT to enhance performance; ZLCADCT does not require coefficient scanning and its subsequent processing, while keeping the number of coefficients the same as ADCT [7].

The performance of ZLCADCT has been extensively evaluated by both simulation using Matlab/Spice and experimentally by employing a hardware platform based on FPGAs. The simulation results have shown that in the best case for the number of targeted coefficients to be retained (i.e., $TCR = 1$), ZLCADCT requires only 28.12% of the adders compared with ADCT [7] and the energy utilized by ADCT [7] is 3.5 times larger than that for ZLCADCT. Larger values of *TCR* modestly increase both the percentages of the required number of adders and energy consumption by ZLCADCT; moreover, for the entire range of *TCR* values, ZLCADCT retains the same image quality of the compressed image as [7]. The experimental evaluation based on FPGAs of the proposed techniques has shown significant reductions with respect to the number of adders, XOR gates, LUTs, power consumption, implementation time, total processing time when compared with ADCT; the PSNR of ZLCADCT has also been retained as for ADCT over the entire range of *TCR* values.

REFERENCES

[1] H. A. F. Almurib, T. N. Kumar, and F. Lombardi, "Approximate DCT image compression using inexact computing," *IEEE Trans. Comput.*, vol. 67, no. 2, pp. 149–159, Feb. 2018.

[2] F. M. Bayer, R. J. Cintra, A. Madanayake, and U. S. Potluri, "Multiplierless approximate 4-point DCT VLSI architectures for transform block coding," *Electron. Lett.*, vol. 49, no. 24, pp. 1532–1534, Nov. 2013.

[3] S. Bouguezel, M. O. Ahmad, and M. N. S. Swamy, "Low-complexity 8× 8 transform for image compression," *Electron. Lett.*, vol. 44, no. 21, pp. 1249–1250, 2008.

[4] S. Bouguezel, M. O. Ahmad, and M. N. S. Swamy, "A low-complexity parametric transform for image compression," in *Proc. IEEE Int. Symp. Circuits Syst. (ISCAS)*, Rio de Janeiro, Brazil, May 2011, pp. 2145–2148.

[5] F. M. Bayer and R. J. Cintra, "DCT-like transform for image compression requires 14 additions only," *Electron. Lett.*, vol. 48, no. 15, pp. 919–921, Jul. 2012.

[6] U. S. Potluri, A. Madanayake, R. J. Cintra, F. M. Bayer, and N. Rajapaksha, "Multiplier-free DCT approximations for RF multi-beam digital aperture-array space imaging and directional sensing," *Meas. Sci. Technol.*, vol. 23, no. 11, 2012, Art. no. 114003.

[7] U. S. Potluri, A. Madanayake, R. J. Cintra, F. M. Bayer, S. Kulasekera, and A. Edirisuriya, "Improved 8-point approximate DCT for image and video compression requiring only 14 additions," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 61, no. 6, pp. 1727–1740, Jun. 2014.

[8] K. Saraswathy, D. Vaithiyanathan, and R. Seshasayanan, "Notice of Violation of IEEE Publication Principles A DCT approximation with low complexity for image compression," in *Proc. Int. Conf. Commun. Signal Process.*, Melmaruvathur, India, Apr. 2013, pp. 465–468.

[9] R. J. Cintra and F. M. Bayer, "A DCT approximation for image compression," *IEEE Signal Process. Lett.*, vol. 18, no. 10, pp. 579–582, Oct. 2011.

[10] A. Edirisuriya, A. Madanayake, R. J. Cintra, and F. M. Bayer, "A multiplication-free digital architecture for 16× 16 2-D DCT/DST transform for HEVC," in *Proc. IEEE 27th Conv. Elect. Electron. Eng. Isr.*, Eilat, Israel, Nov. 2012, pp. 1–5.

[11] F. M. Bayer, R. J. Cintra, A. Edirisuriya, and A. Madanayake, "A digital hardware fast algorithm and FPGA-based prototype for a novel 16-point approximate DCT for image compression applications," *Meas. Sci. Technol.*, vol. 23, no. 11, 2012, Art. no. 114010.

[12] T. L. T. da Silveira, F. M. Bayer, R. J. Cintra, S. Kulasekera, A. Madanayake, and A. J. Kozakevicius, "An orthogonal 16-point approximate DCT for image and video compression," *Multidimens. Syst. Signal Process.*, vol. 27, no. 1, pp. 87–104, Jan. 2016.

[13] K. Chen, F. Lombardi, and J. Han, "Matrix multiplication by an inexact systolic array," in *Proc. 2015 IEEE/ACM Int. Symp. Nanosc. Architectures (NANOARCH)*, Boston, MA, USA, Jul. 2015, pp. 151–156.

[14] A. Momeni, J. Han, P. Montuschi, and F. Lombardi, "Design and analysis of approximate compressors for multiplication," *IEEE Trans. Comput.*, vol. 64, no. 4, pp. 984–994, Apr. 2015.

[15] Z. Yang, A. Jain, J. Liang, J. Han, and F. Lombardi, "Approximate XOR/XNOR-based adders for inexact computing," in *Proc. 13th IEEE Int. Conf. Nanotechnol.*, Beijing, China, Aug. 2013, pp. 690–693.

[16] H. Junqi, T. N. Kumar, H. Abbas, and F. Lombardi, "Simulation-based evaluation of frequency upscaled operation of exact/approximate ripple carry adders," in *Proc. IEEE Int. Symp. Defect Fault Tolerance VLSI Nanotechnol. Syst. (DFT)*, Cambridge, U.K., Oct. 2017, pp. 1–6.

[17] H. A. F. Almurib, T. N. Kumar, and F. Lombardi, "Inexact designs for approximate low power addition by cell replacement," in *Proc. Design, Automat. Test Eur. Conf. Exhib. (DATE)*, Dresden, Germany, Mar. 2016, pp. 660–665.

[18] C. Liu, J. Han, and F. Lombardi, "A low-power, high-performance approximate multiplier with configurable partial error recovery," in *Proc. Design, Automat. Test Eur. Conf. Exhib. (DATE)*, Dresden, Germany, Mar. 2014, pp. 1–4.

[19] N. Banerjee, G. Karakonstantis, and K. Roy, "Process variation tolerant low power DCT architecture," in *Proc. Design, Automat. Test Eur. Conf. Exhib. (DATE)*, Nice, France, 2007, pp. 630–635.

[20] G. Karakonstantis, D. Mohapatra, and K. Roy, "System level DSP synthesis using voltage overscaling, unequal error protection & adaptive quality tuning," in *Proc. IEEE Workshop Signal Process. Syst.*, Tampere, Finland, Oct. 2009, pp. 133–138.

[21] T. Darwish and M. Bayoumi, "Energy aware distributed arithmetic DCT architectures," in *Proc. IEEE Workshop Signal Process. Syst.*, Seoul, South Korea, Aug. 2003, pp. 351–356.

[22] B. Garg, N. K. Bharadwaj, and G. K. Sharma, "Energy scalable approximate DCT architecture trading quality via boundary error-resiliency," in *Proc. 27th IEEE Int. Syst.-on-Chip Conf. (SOCC)*, Las Vegas, NV, USA, Sep. 2014, pp. 306–311.

[23] M. Jridi and P. K. Meher, "Scalable approximate dct architectures for efficient hevc-compliant video coding," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 27, no. 8, pp. 1815–1825, Aug. 2017.

[24] S. Chatterjee and K. Sarawadekar, "WHT and matrix decomposition based approximated IDCT architecture for HEVC," *IEEE Trans. Circuits Syst., II, Exp. Briefs*, to be published. doi: 10.1109/TCSII.2018.2874071.

[25] S. B. Jdidia, M. Jridi, F. Belghith, and N. Masmoudi, "Low-complexity algorithm using DCT approximation for POST-HEVC standard," in *Proc. SPIE*, vol. 10649, Apr. 2018, Art. no. 106490Y.

[26] 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)*, Florence, Italy, May 2018, pp. 1–4.

[27] R. J. Cintra, F. M. Cintra, V. A. Coutinho, S. Kulasekera, A. Madanayake, and A. Leite, "Energy-efficient 8-point DCT approximations: Theory and hardware architectures," *Circuits, Syst., Signal Process.*, vol. 35, no. 11, pp. 4009–4029, Nov. 2016.

[28] V. de A. Coutinho, R. J. Cintra, F. M. Bayer, S. Kulasekera, and A. Madanayake, "Low-complexity pruned 8-point DCT approximations for image encoding," in *Proc. Int. Conf. Electron., Commun. Comput. (CONIELECOMP)*, Cholula, Mexico, Feb. 2015, pp. 1–7.

[29] M. K. Islam, M. Moznuzzaman, M. F. Khatun, and R. Yesmin, "A proposed modification of baseline JPEG standard image compression technique," *Int. J. Sci. Eng. Res.*, vol. 6, no. 8, pp. 180–186, 2015.

[30] L. Dubois, W. Puech, and J. Blanc-Talon, "Fast protection of H.264/AVC by reduced selective encryption of CAVLC," in *Proc. 19th Eur. Signal Process. Conf.*, Barcelona, Spain, Aug./Sep. 2011, pp. 2185–2189.

Junqi Huang (M'18) received the B.S. degree in electronic engineering from Guangzhou University, Guangzhou, China, in 2011, and the M.S. degree in information technology management from Hong Kong Baptist University, China, in 2013. He is currently pursuing the Ph.D. degree in electronic engineering with the University of Nottingham at Malaysia Campus, Malaysia. His research interests are in the fields of approximate computing, image processing, and VLSI design.

T. Nandha Kumar (M'06–SM'11) received the bachelor's degree from the University of Madras, the master's degree from the National Institute of Technology, India, and the Ph.D. degree . in electrical and electronic engineering from The University of Nottingham. He is currently an Associate Professor with the University of Nottingham, Malaysia. Prior to this, he was a Senior Pre-Silicon Validation Engineer with Intel Corporation. He has published over 70 papers in the reputed International journals and conferences. His research interests are in the fields of emerging non-volatile memory, approximate computing, VLSI design, and test and fault/defect tolerance of digital systems. He is an Associate Editor of the *IET Circuits, Devices and Systems*.

Haider A. F. Almurib (S'00–M'05–SM'11) received the Ph.D. degree in electrical engineering from the University of Malaya, Kuala Lumpur, Malaysia, in 2006. He is currently a Professor with the Department of Electrical and Electronic Engineering, The University of Nottingham, Kuala Lumpur. Before this, he was a Senior Lecturer with the University of Malaya. He was with industries for four years as a Research and Development Engineer in the field of computer automation. His research interests include emerging non-volatile memory devices, embedded systems, fault tolerance of digital systems, nonlinear and intelligent control, electric machines and drives. He is currently an Associate Editor of *IET Power Electronics*.

Fabrizio Lombardi (M'81–SM'02–F'09) received the B.Sc. degree (Hons.) in electronic engineering from the University of Essex, U.K., in 1977, the master's degree in microwaves and modern optics, and the Diploma degree in microwave engineering from the Microwave Research Unit, University College London, in 1978, and the Ph.D. degree from the University of London in 1982.

He is currently the International Test Conference Endowed Chair Professor with Northeastern University, Boston, USA. His research interests are bio-inspired and nano manufacturing/computing, VLSI design, testing, and fault/defect tolerance of digital systems. He has extensively published in his research areas. He has co-authored/edited seven books. He was the Editor-in-Chief of the IEEE TRANSACTIONS ON COMPUTERS from 2007 to 2010 and the inaugural Editor-in-Chief of the IEEE TRANSACTIONS ON EMERGING TOPICS IN COMPUTING from 2013 to 2017. He is the Editor-in-Chief of the IEEE TRANSACTIONS ON NANOTECHNOLOGY.