Computation of 2D 8x8 DCT Based on the Loeffler Factorization Using Algebraic Integer Encoding

Diego F. G. Coelho, Sushmabhargavi Nimmalapalli, Vassil Dimitrov, Arjuna Madanayake, Renato J. Cintra, Arnaud Tisserand

Computation of 2D 8×8 DCT Based on the Loeffler Factorization Using Algebraic Integer Encoding

Diego F. G. Coelho, *Student Member, IEEE,* Sushmabhargavi Nimmalapalli, Vassil S. Dimitrov, Arjuna Madanayake, *Member, IEEE,* Renato J. Cintra, *Senior Member, IEEE,* and Arnaud Tisserand, *Senior Member, IEEE*

Abstract—This paper proposes a computational method for 2D 8×8 DCT based on algebraic integers. The proposed algorithm is based on the Loeffler 1D DCT algorithm, and is shown to operate with exact computation—i.e., error-free arithmetic—up to the final reconstruction step (FRS). The proposed algebraic integer architecture maintains error-free computations until an entire block of DCT coefficients having size 8×8 is computed, unlike algorithms in the literature which claim to be error-free but in fact introduce arithmetic errors between the column- and row-wise 1D DCT stages in a 2D DCT operation. Fast algorithms are proposed for the final reconstruction step employing two approaches, namely, the expansion factor and dyadic approximation. A digital architecture is also proposed for a particular FRS algorithm, and is implemented on an FPGA platform for on-chip verification. The FPGA implementation operates at 360 MHz, and is capable of a real-time throughput of $3.6 \cdot 10^8$ 2D DCTs of size 8×8 every second, with corresponding pixel rate of $2.3 \cdot 10^{10}$ pixels per second. The digital architecture is synthesized using 180 nm CMOS standard cells and shows a chip area of 7.41 mm^2. The CMOS design is predicted to operate at 893 MHz clock frequency, at a dynamic power consumption 13.22 mW/MHz $\cdot$ V_{sup}^2.

Index Terms—Algebraic integers, Error-free computation, DCT, Multidimension

1 Introduction

Proposed in 1974 by Ahmed *et al.* [1], the discrete cosine transform (DCT) is a pivotal tool in signal processing problems [2], [3], such as image compression [4], noise reduction [5], and watermarking methods [6], [7]. Among the several existing discrete transforms, the DCT has the distinctive characteristic of optimally approximating the Karhunen-Loève transform (KLT) for highly correlated stationary Markov signals of type I [3]. This is relevant because images often follow such model [3].

- *Authors thank CNPq and FACEPE, Brazil, and NSERC, Canada, for partial support. The authors would like to thank the referees of the paper for their numerous suggestions that greatly improved the quality of the paper.*

- *D. F. G. Coelho was with the Graduate Program in Electrical and Computer Engineering, University of Calgary. Now he is with Microsemi Corporation, Calgary, Canada. E-mail: diegofgcoelho@gmail.com*
- *Sushmabhargavi Nimmalapalli is with the Graduate Program in Electrical and Computer Engineering, University of Akron.*
- *V. S. Dimitrov is with the Department of Electrical and Computer Engineering, University of Calgary. E-mail: vdvsd103@gmail.com*
- *A. Madanayake is with the Department of Electrical and Computer Engineering, University of Akron. E-mail: arjuna@uakron.edu*
- *R. J. Cintra is with the Signal Processing Group, Departamento de Estatística, Universidade Federal de Pernambuco and the Department of Electrical and Computer Engineering, University of Calgary. E-mail: rjdsc@de.ufpe.br*
- *A. Tisserand is with CNRS (French National Center for Scientific Research) and Lab-STICC laboratory in Lorient, France.*

The 8-point DCT of type II, hereafter referred only as DCT [3], has been employed in different image and video compression standards [8], including JPEG [9], MPEG-1 [10], H.264 [11], and HEVC [12]. Due to such wide acceptance, several fast algorithms were proposed for the 8-point DCT [3]. A particularly relevant fast algorithm is the one proposed by Loeffler *et al.* described in [13], which is capable of computing the 8-point DCT with the minimum possible number of multiplications [13]–[15]. Because of this, the Loeffler factorization for the 8-point DCT is considered to be a reference method for comparing DCT algorithms.

The theory of algebraic integers (AI) was first introduced in the context of digital signal processing in 1985 by Cozzens and Finkelstein [16], [17] aiming at the computation of the discrete Fourier transform (DFT). The method included the use of residue number systems in order to reduce the dynamic range of the quantities involved in the DFT computation. In [17], it was shown that it is possible to numerically evaluate the DFT in exact format and without error propagation, achieving arbitrary precision according to a final reconstruction step (FRS). The FRS is responsible for mapping back the quantities from algebraic integer representation into usual fixed-point representation. The irrational quantities required in the FRS are approximated by rational quantities that can be efficiently implemented in hardware.

Several fast algorithms based on algebraic integer theory have been proposed for the computation of the 1D and 2D DCT [18]–[21]. These architectures are able to compute the 1D DCT without multipliers within an error-free structure.

Usual computation of the 2D DCT is accomplished by column- and row-wise calls of the 1D DCT. However simply computing the 1D DCT by means of AI-based algorithm does not result in a *bona fide* AI-based 2D DCT computation. Indeed, from the

2D DCT point-of-view, the FRS blocks from AI-based 1D DCT appear as an intermediate computation. Such intermediate reconstruction precludes error-free computation and undermines one of the purposes of employing algebraic integers. This drawback was identified in [22], [23] where an error-free computation of 2D DCT was proposed for the Arai algorithm [24]. Therefore, the output of the algorithm used both in [22] and [23] is a non-uniform scaled version of the 2D DCT spectrum.

The error-free characteristic of the methods proposed in [16]–[23] is a by-product of the algebraic integer encoding, possibly not the most important. Additional advantages of AI-based fast algorithms include (i) parallelization and (ii) low latency due to the accumulation of multiplicative complexity at the FRS.

This paper aims at introducing a 2D DCT algorithm based on the AI representation that combines (i) high throughput; (ii) low latency; (iii) parallelization; and (iv) error-free architecture. This is achieved by means of the Loeffler fast algorithm for the 1D 8-point DCT using the encoding proposed in [25]. The error-free architecture is possible only because we propose new fast algorithms for the 1D DCT tailored for the inputs required by the 2D architecture. The use of the proposed dedicated algorithms allows the removal of the FRS at the end of each 1D DCT when applied to the columns of the 8×8 blocks. A digital circuit capable of computing the 2D DCT with the above properties becomes an attractive tool according to several metrics [2].

This article unfolds as follows. Section 2 reviews the algebraic integer representation proposed in [25] and employed in the present work. Section 2 summarizes the 1D DCT for real quantized input sequences and the Loeffler DCT fast algorithm. Section 3 reviews the 2D DCT and presents the fast algorithms needed to compute the 2D DCT using algebraic integer theory in a error-free fashion. A comparison between the introduced scheme with previous works that propose AI-based error-free 2D DCT computation is supplied. Section 4 details the FRS for the 2D DCT computation and introduces two different methods for the efficient decoding of algebraic integer quantities. In Section 5, an FPGA implementation for the proposed 2D DCT architecture is proposed. Section 6 concludes the paper.

2 The Algebraic Integer Representation

2.1 Review of 8-point DCT AI Basis

2.1.1 The AI Basis

The 8-point 1D DCT is a linear orthogonal transformation given by [3], [4]:

$$X_k = \frac{1}{2}\sum_{n=0}^{7}\beta_k x_n \cos\left[\frac{\pi(2n+1)k}{16}\right], \quad k=0,1,\ldots,7, \qquad (1)$$

where $\beta_0 = 1/\sqrt{2}$ and $\beta_k = 1$, for $k = 1,2,\ldots,7$.

In [25], the authors characterized the ring spanned by the set $\mathbf{Z}$ whose elements are 1 and c_k, where $c_k = 2\cos(k\pi/16)$, for $k = 1,2,\ldots,7$. The vector space span$(\mathbf{Z})$ generated by linear combination of the elements of $\mathbf{Z}$ is suitable for the computation of the 8-point DCT. This is due to the fact that the 8-point DCT requires the quantities $\cos[\pi k(2n+1)/16]$, $n,k = 0,1,\ldots,7$ [3]. Hereafter, we denote $\boldsymbol{\zeta} = \begin{bmatrix}1 & c_1 & c_2 & c_3 & c_4 & c_5 & c_6 & c_7\end{bmatrix}^\top$ as the basis element vector.

2.1.2 Encoding and Decoding

The encoding of a given real number x over the considered AI basis is denoted by $f_{\text{enc}}(x;\boldsymbol{\zeta}) = \mathbf{x}$, where $\mathbf{x} = \begin{bmatrix}a_0 & a_1 & a_2 & a_3 & a_4 & a_5 & a_6 & a_7\end{bmatrix}^\top$ is the encoded integer vector, $a_k \in \mathbb{Z}$, $k = 0,1,\ldots,7$, and $^\top$ denotes transposition.

The decoding operation is given directly by the dot product operation [22]:

$$f_{\text{dec}}(\mathbf{x};\boldsymbol{\zeta}) = \mathbf{x}^\top \cdot \boldsymbol{\zeta} = a_0 + \sum_{k=1}^{7} a_k \cdot c_k = \hat{x}. \qquad (2)$$

In [25], it was shown that the above representation is dense and can provide arbitrary precision, i.e., it is always possible to determine a vector $\mathbf{x}$ such that $|x-\hat{x}| < \varepsilon$, for any $\varepsilon > 0$. Authors have also pointed out that in usual applications, such as in the context of image compression, the input data are real, discrete, and quantized [26] in the form of an integer [3]. In such conditions, a real quantized input m, the AI-encoded data can be trivially obtained according to $f_{\text{enc}}(m;\boldsymbol{\zeta}) = \begin{bmatrix}m & 0 & 0 & 0 & 0 & 0 & 0\end{bmatrix}^\top$.

2.1.3 Arithmetic Operations

AI-based addition and multiplication operations over the considered basis were defined in [25], being the only elementary operations required by the Loeffler DCT algorithm. Since AI quantities are represented by arrays of integers, the addition and subtraction operations obey the usual vector addition and subtraction rule.

For the multiplication operation, the product of an arbitrary algebraic integer in the proposed representation by one of the basis elements obey the relations described in Table 1. Such multiplications are trivial in the sense that only additions, subtractions, and permutations of the input coefficients are needed. In hardware implementation, these operations are performed by simple wiring and adders/subtractors.

2.2 Loeffler 1D DCT Multiplicands

The Loeffler DCT algorithm has a signal flow graph (SFG) with four stages (cf. Figure 1 in [25]) and it requires the following multiplicands: $\{c_1, \sqrt{2}c_2, c_3, c_4, c_5, \sqrt{2}c_6, c_7\}$. If the multiplicands in Stage 2 to Stage 4 are combined, then only six resulting multiplicands are required: $c_4\cdot c_2$, $c_4\cdot c_6$, $c_4\cdot c_3$, $c_4\cdot c_5$, $c_4\cdot c_1$, and $c_4\cdot c_7$. Employing trigonometric rules, we obtain that $c_i\cdot c_k = c_{i+k} + c_{i-k}$, for any $i,k \in \mathbb{Z}$ and $c_i = -c_{16-i}$, for $i = 8,9,\ldots,16$. Therefore, the quantities required by Loeffler DCT computation possess simple and multiplierless representations over the representation introduced in [25].

In [25], it was shown that the ring implied by the AI formalism for the 1D DCT case is over-complete; and thus the coefficients linked to the basis element c_4 are not required. On the other hand, the 2D DCT demands the coefficients associated to c_4.

2.3 1D AI-based Fast Algorithm

The representation proposed in [25] when applied to the 1D 8-point DCT furnishes a multiplierless algorithm that requires a total of 20 additions (cf. Figure 3 in [25]). Indeed, due to Loeffler DCT definition, the algorithm output is a scaled DCT with scaling factor of 2. If required, the output can be re-scaled by a simple bit-shifting or it can be inserted into the decoding stage. Therefore, the scaling by 2 does not add to the increase of arithmetic complexity when performing the processing.

TABLE 1
Quantities required by Loeffler algorithm for 8-point DCT and their respective products by an arbitrary algebraic integer

x	$f_{\text{enc}}(x;\boldsymbol{\zeta})\cdot\mathbf{u}$
1	$[u_0 \quad u_1 \quad u_2 \quad u_3 \quad u_4 \quad u_5 \quad u_6 \quad u_7]^\top$
c_1	$[2u_1 \quad u_0+u_2 \quad u_1+u_3 \quad u_2+u_4 \quad u_3+u_5 \quad u_4+u_6 \quad u_5+u_7 \quad u_6]^\top$
c_2	$[2u_2 \quad u_1+u_3 \quad u_0+u_4 \quad u_1+u_5 \quad u_2+u_6 \quad u_3+u_7 \quad u_4 \quad u_5-u_7]^\top$
c_3	$[2u_3 \quad u_2+u_4 \quad u_1+u_5 \quad u_0+u_6 \quad u_1+u_7 \quad u_2 \quad u_3-u_7 \quad u_4-u_6]^\top$
c_4	$[2u_4 \quad u_3+u_5 \quad u_2+u_6 \quad u_1+u_7 \quad u_0 \quad u_1-u_7 \quad u_2-u_6 \quad u_3-u_5]^\top$
c_5	$[2u_5 \quad u_4+u_6 \quad u_3+u_7 \quad u_2 \quad u_1-u_7 \quad u_0-u_6 \quad u_1-u_5 \quad u_2-u_4]^\top$
c_6	$[2u_6 \quad u_5+u_7 \quad u_4 \quad u_3-u_7 \quad u_2-u_6 \quad u_1-u_5 \quad u_0-u_4 \quad u_1-u_3]^\top$
c_7	$[2u_7 \quad u_6 \quad u_5-u_7 \quad u_4-u_6 \quad u_3-u_5 \quad u_2-u_4 \quad u_1-u_3 \quad u_0-u_2]^\top$

3 The 2D DCT and the AI basis representation

3.1 The 2D DCT

Let $x_{m,n}$ be a 2D array for $m,n = 0,1,\ldots,7$. The 2D 8-point DCT is a linear transformation defined as [3], [4]:

$$X_{l,k} = \frac{1}{4}\sum_{m=0}^{7}\sum_{n=0}^{7}\alpha_k\beta_l x_{m,n}\cos\left[\frac{\pi(2n+1)l}{16}\right]\cos\left[\frac{\pi(2m+1)k}{16}\right], \tag{3}$$

where $l,k = 0,1,\ldots,7$, $\alpha_0 = \beta_0 = 1/\sqrt{2}$ and $\alpha_k = \beta_l = 1$, for $l,k = 1,2,\ldots,7$.

As adopted by several image encoding schemes [3], [27]–[29], the 2D DCT computation is performed by successive calls of the 1D DCT applied to the columns of the input 2D data, then to the rows of the resulting matrix. For blocks of size 8×8, sixteen calls of the 1D DCTs are required to furnish the 2D DCT.

3.2 2D AI-based Fast Algorithm

When the 2D input array is real and quantized, several simplifications arises. These simplifications can be exploited to provide efficient fast algorithms for the 2D DCT over the AI basis representation proposed in [25] without the need of FRS for each 1D DCT between the computation over the columns and rows.

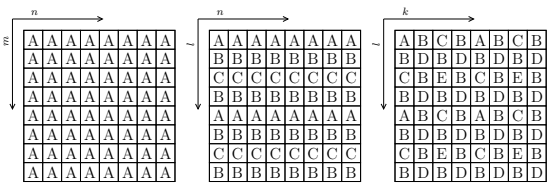
Considering the 2D DCT computation by means of column- and row-wise calls of the 1D DCT, we notice the following structure. If the 2D input data consists of integer elements, then the AI-encoded quantities resulting from the column-wise calls of the 1D DCT have the following configuration: (i) the elements in the 0th and 4th rows have always non-null first coefficient in its AI-based representation; (ii) the elements in 1st, 3rd, 5th, and 7th rows exhibit non-null odd-index coefficients; and (iii) the 2nd and 6th rows have non-null coefficients only in the 3rd and 7th coefficients on its respective AI-based representation. Such fixed patterns are due to the algorithm for class A input (cf. Figure 3 in [25]).

In view of their patterns, we categorize the AI quantities into the five classes as shown in Table 2, where non-null coefficients locations are represented by the cross symbol. If we represent the two-dimensional input sequence in graphical format as in Figure 1(a), we have the configuration in Figure 1(b) after the application of 1D DCT over the columns. Letters A, B, C, D, and E represent the class to which the quantity belongs according to Table 2.

For an error-free realization of the 2D DCT without FRS blocks between the column- and row-wise 1D DCT calls, we need to derive tailored AI-based DCT algorithms considering input data in Class B and C. For input in Class A, the DCT algorithm collapses to the method detailed in [25]. For such, we consider the multiplication rules in Table 1. The obtained procedures are detailed in the algorithms in Figure 2, for Class B data; and in Figure 3, for Class C data. The algorithm in Figure 2 requires 136 additions and 14 bit-shifting operations; whereas the procedure in Figure 3 demands 74 additions and 8 bit-shifting operations.

TABLE 2
AI representation classification. Cross symbols correspond to non-null coefficients

Class	AI representation
A	$\mathbf{u} = [\times \;\; 0 \;\; 0 \;\; 0 \;\; 0 \;\; 0 \;\; 0 \;\; 0]^\top$
B	$\mathbf{u} = [0 \;\; \times \;\; 0 \;\; \times \;\; 0 \;\; \times \;\; 0 \;\; \times]^\top$
C	$\mathbf{u} = [0 \;\; 0 \;\; \times \;\; 0 \;\; 0 \;\; 0 \;\; \times \;\; 0]^\top$
D	$\mathbf{u} = [\times \;\; 0 \;\; \times \;\; 0 \;\; \times \;\; 0 \;\; \times \;\; 0]^\top$
E	$\mathbf{u} = [\times \;\; 0 \;\; 0 \;\; 0 \;\; \times \;\; 0 \;\; 0 \;\; 0]^\top$



(a) Input data (b) Column-wise 1D DCT output (c) 2D DCT output

Fig. 1. 2D representation of the input coefficient class before the transformation (a), after the application of the 1D DCT over its columns (b) and after the application of the 2D DCT (c).

The outputs of the algorithms in Figures 2 and 3 also follow a fixed pattern that determines the class of each output element in the AI representation. The class of each element of the output sequence is shown in Figure 1(c).

For a given 8×8 block $\mathbf{x}$, let $x_{m,\cdot}$ and let $x_{\cdot,n}$ denote the mth row and the nth column of $\mathbf{x}$, respectively. Let also the operators $\text{DCT}_\text{A}(\cdot)$, $\text{DCT}_\text{B}(\cdot)$, and $\text{DCT}_\text{C}(\cdot)$ be instantiations of the algorithms for Class A input (cf. Figure 3 in [25]), Class B as in Figure 2, and Class C as in Figure 3, respectively. For example, $\text{DCT}_\text{A}(x_{\cdot,n})$ denotes the computation of the 1D DCT over the nth column of the block $\mathbf{x}$ whose coefficients belong to Class A in the AI-based representation. Let also $\mathbb{Z}_\text{X}$ be the set of 8-point

Input: $\mathbf{x}_n \in \mathbb{Z}_{\mathrm{B}}$ for $n = 0, 1, \ldots, 7$
Output: $\mathbf{X}_k \in \mathrm{span}(\mathbf{Z})$, for $k = 0, 1, \ldots, 7$

Stage 1 Outputs:

$$\mathbf{A}_0 = [0\ \mathbf{x}_0(2)+\mathbf{x}_7(2)\ 0\ \mathbf{x}_0(4)+\mathbf{x}_7(4)\ 0\ \mathbf{x}_0(6)+\mathbf{x}_7(6)\ 0\ \mathbf{x}_0(8)+\mathbf{x}_7(8)]^\top$$
$$\mathbf{A}_1 = [0\ \mathbf{x}_1(2)+\mathbf{x}_6(2)\ 0\ \mathbf{x}_1(4)+\mathbf{x}_6(4)\ 0\ \mathbf{x}_1(6)+\mathbf{x}_6(6)\ 0\ \mathbf{x}_1(8)+\mathbf{x}_6(8)]^\top$$
$$\mathbf{A}_2 = [0\ \mathbf{x}_2(2)+\mathbf{x}_5(2)\ 0\ \mathbf{x}_2(4)+\mathbf{x}_5(4)\ 0\ \mathbf{x}_2(6)+\mathbf{x}_5(6)\ 0\ \mathbf{x}_2(8)+\mathbf{x}_5(8)]^\top$$
$$\mathbf{A}_3 = [0\ \mathbf{x}_3(2)+\mathbf{x}_4(2)\ 0\ \mathbf{x}_3(4)+\mathbf{x}_4(4)\ 0\ \mathbf{x}_3(6)+\mathbf{x}_4(6)\ 0\ \mathbf{x}_3(8)+\mathbf{x}_4(8)]^\top$$
$$\mathbf{A}_4 = [0\ \mathbf{x}_3(2)-\mathbf{x}_4(2)\ 0\ \mathbf{x}_3(4)-\mathbf{x}_4(4)\ 0\ \mathbf{x}_3(6)-\mathbf{x}_4(6)\ 0\ \mathbf{x}_3(8)-\mathbf{x}_4(8)]^\top$$
$$\mathbf{A}_5 = [0\ \mathbf{x}_2(2)-\mathbf{x}_5(2)\ 0\ \mathbf{x}_2(4)-\mathbf{x}_5(4)\ 0\ \mathbf{x}_2(6)-\mathbf{x}_5(6)\ 0\ \mathbf{x}_2(8)-\mathbf{x}_5(8)]^\top$$
$$\mathbf{A}_6 = [0\ \mathbf{x}_1(2)-\mathbf{x}_6(2)\ 0\ \mathbf{x}_1(4)-\mathbf{x}_6(4)\ 0\ \mathbf{x}_1(6)-\mathbf{x}_6(6)\ 0\ \mathbf{x}_1(8)-\mathbf{x}_6(8)]^\top$$
$$\mathbf{A}_7 = [0\ \mathbf{x}_0(2)-\mathbf{x}_7(2)\ 0\ \mathbf{x}_0(4)-\mathbf{x}_7(4)\ 0\ \mathbf{x}_0(6)-\mathbf{x}_7(6)\ 0\ \mathbf{x}_0(8)-\mathbf{x}_7(8)]^\top$$

Auxiliary additions in Stage 2:

$$a_0 = \mathbf{A}_4(2)+\mathbf{A}_7(8) \quad a_1 = \mathbf{A}_4(2)-\mathbf{A}_7(8) \quad a_2 = \mathbf{A}_4(8)+\mathbf{A}_7(2) \quad a_3 = \mathbf{A}_4(8)-\mathbf{A}_7(2)$$
$$a_4 = \mathbf{A}_4(4)+\mathbf{A}_7(6) \quad a_5 = \mathbf{A}_4(4)-\mathbf{A}_7(6) \quad a_6 = \mathbf{A}_4(6)+\mathbf{A}_7(4) \quad a_7 = \mathbf{A}_4(6)-\mathbf{A}_7(4)$$
$$a_8 = \mathbf{A}_5(2)+\mathbf{A}_6(8) \quad a_9 = \mathbf{A}_5(2)-\mathbf{A}_6(8) \quad a_{10} = \mathbf{A}_5(8)+\mathbf{A}_6(2) \quad a_{11} = \mathbf{A}_5(8)-\mathbf{A}_6(2)$$
$$a_{12} = \mathbf{A}_5(4)+\mathbf{A}_6(6) \quad a_{13} = \mathbf{A}_5(4)-\mathbf{A}_6(6) \quad a_{14} = \mathbf{A}_5(6)+\mathbf{A}_6(4) \quad a_{15} = \mathbf{A}_5(6)-\mathbf{A}_6(4)$$

Stage 2 Outputs:

$$\mathbf{B}_0 = [\ 0\ \ \mathbf{A}_0(2)+\mathbf{A}_3(2)\ \ 0\ \ \mathbf{A}_0(4)+\mathbf{A}_3(4)\ \ 0\ \ \mathbf{A}_0(6)+\mathbf{A}_3(6)\ \ 0\ \ \mathbf{A}_0(8)+\mathbf{A}_3(8)]^\top$$
$$\mathbf{B}_1 = [\ 0\ \ \mathbf{A}_1(2)+\mathbf{A}_2(2)\ \ 0\ \ \mathbf{A}_1(4)+\mathbf{A}_2(4)\ \ 0\ \ \mathbf{A}_1(6)+\mathbf{A}_2(6)\ \ 0\ \ \mathbf{A}_1(8)+\mathbf{A}_2(8)]^\top$$
$$\mathbf{B}_2 = [\ 0\ \ \mathbf{A}_1(2)-\mathbf{A}_2(2)\ \ 0\ \ \mathbf{A}_1(4)-\mathbf{A}_2(4)\ \ 0\ \ \mathbf{A}_1(6)-\mathbf{A}_2(6)\ \ 0\ \ \mathbf{A}_1(8)-\mathbf{A}_2(8)]^\top$$
$$\mathbf{B}_3 = [\ 0\ \ \mathbf{A}_0(2)-\mathbf{A}_3(2)\ \ 0\ \ \mathbf{A}_0(4)-\mathbf{A}_3(4)\ \ 0\ \ \mathbf{A}_0(6)-\mathbf{A}_3(6)\ \ 0\ \ \mathbf{A}_0(8)-\mathbf{A}_3(8)]^\top$$
$$\mathbf{B}_4 = [\ 2a_4\ \ 0\ \ a_0+a_6\ \ 0\ \ a_1+a_2\ \ 0\ \ a_5-a_3\ \ 0]^\top$$
$$\mathbf{B}_5 = [\ 2a_8\ \ 0\ \ a_9+a_{12}\ \ 0\ \ a_{13}+a_{14}\ \ 0\ \ a_{15}+a_{10}\ \ 0]^\top$$
$$\mathbf{B}_6 = [-2a_{11}\ \ 0\ \ a_{10}-a_{15}\ \ 0\ \ -a_{13}+a_{14}\ \ 0\ \ a_{12}-a_9\ \ 0]^\top$$
$$\mathbf{B}_7 = [\ -2a_7\ \ 0\ \ -a_3-a_5\ \ 0\ \ -a_1+a_2\ \ 0\ \ a_6-a_0\ \ 0]^\top$$

Auxiliary additions in Stage 3:

$$b_0 = \mathbf{B}_2(2)+\mathbf{B}_3(2) \quad b_1 = \mathbf{B}_2(2)-\mathbf{B}_3(2) \quad b_2 = \mathbf{B}_2(4)+\mathbf{B}_3(4) \quad b_3 = \mathbf{B}_2(4)-\mathbf{B}_3(4)$$
$$b_4 = \mathbf{B}_2(6)+\mathbf{B}_3(6) \quad b_5 = \mathbf{B}_2(6)-\mathbf{B}_3(6) \quad b_6 = \mathbf{B}_2(8)+\mathbf{B}_3(8) \quad b_7 = \mathbf{B}_2(8)-\mathbf{B}_3(8)$$
$$b_8 = b_0+b_7 \quad b_9 = b_0-b_7 \quad b_{10} = b_1+b_6 \quad b_{11} = b_1-b_6$$
$$b_{12} = b_2+b_5 \quad b_{13} = b_2-b_5 \quad b_{14} = b_3+b_4 \quad b_{15} = b_3-b_4$$

Stage 3 Outputs:

$$\mathbf{C}_0 = [\ 0\ \ \mathbf{B}_0(2)+\mathbf{B}_1(2)\ \ 0\ \ \mathbf{B}_0(4)+\mathbf{B}_1(4)\ \ 0\ \ \mathbf{B}_0(6)+\mathbf{B}_1(6)\ \ 0\ \ \mathbf{B}_0(8)+\mathbf{B}_1(8)]^\top$$
$$\mathbf{C}_1 = [\ 0\ \ \mathbf{B}_0(2)-\mathbf{B}_1(2)\ \ 0\ \ \mathbf{B}_0(4)-\mathbf{B}_1(4)\ \ 0\ \ \mathbf{B}_0(6)-\mathbf{B}_1(6)\ \ 0\ \ \mathbf{B}_0(8)-B1(8)]^\top$$
$$\mathbf{C}_2 = [\ 0\ \ b_9+b_{13}\ \ 0\ \ b_8-b_{15}\ \ 0\ \ -b_{11}+b_{12}\ \ 0\ \ -b_{10}+b_{14}]^\top$$
$$\mathbf{C}_3 = [\ 0\ \ -b_{10}-b_{14}\ \ 0\ \ -b_{11}-b_{12}\ \ 0\ \ -b_{15}-b_8\ \ 0\ \ b_{13}-b_9]^\top$$
$$\mathbf{C}_4 = [\ \mathbf{B}_4(1)+\mathbf{B}_6(1)\ \ 0\ \ \mathbf{B}_4(3)+\mathbf{B}_6(3)\ \ 0\ \ \mathbf{B}_4(5)+\mathbf{B}_6(5)\ \ 0\ \ \mathbf{B}_4(7)+\mathbf{B}_6(7)\ \ 0]^\top$$
$$\mathbf{C}_5 = [-\mathbf{B}_5(1)+\mathbf{B}_7(1)\ \ 0\ \ -\mathbf{B}_5(3)+\mathbf{B}_7(3)\ \ 0\ \ -\mathbf{B}_5(5)+\mathbf{B}_7(5)\ \ 0\ \ -\mathbf{B}_5(7)+\mathbf{B}_7(7)\ \ 0]^\top$$
$$\mathbf{C}_6 = [\ \mathbf{B}_4(1)-\mathbf{B}_6(1)\ \ 0\ \ \mathbf{B}_4(3)-\mathbf{B}_6(3)\ \ 0\ \ \mathbf{B}_4(5)-\mathbf{B}_6(5)\ \ 0\ \ \mathbf{B}_4(7)-\mathbf{B}_6(7)\ \ 0]^\top$$
$$\mathbf{C}_7 = [\ \mathbf{B}_5(1)+\mathbf{B}_7(1)\ \ 0\ \ \mathbf{B}_5(3)+\mathbf{B}_7(3)\ \ 0\ \ \mathbf{B}_5(5)+\mathbf{B}_7(5)\ \ 0\ \ \mathbf{B}_5(7)+\mathbf{B}_7(7)\ \ 0]^\top$$

Output:

$$\mathbf{X}_0 = [\ 0\ \ 2\mathbf{C}_0(2)\ \ 0\ \ 2\mathbf{C}_0(4)\ \ 0\ \ 2\mathbf{C}_0(6)\ \ 0\ \ 2\mathbf{C}_0(8)]^\top$$
$$\mathbf{X}_1 = [\ \mathbf{C}_4(1)+\mathbf{C}_7(1)\ \ 0\ \ \mathbf{C}_4(3)+\mathbf{C}_7(3)\ \ 0\ \ \mathbf{C}_4(5)+\mathbf{C}_7(5)\ \ 0\ \ \mathbf{C}_4(7)+\mathbf{C}_7(7)\ \ 0]^\top$$
$$\mathbf{X}_2 = [\ 0\ \ \mathbf{C}_2(2)\ \ 0\ \ \mathbf{C}_2(4)\ \ 0\ \ \mathbf{C}_2(6)\ \ 0\ \ \mathbf{C}_2(8)]^\top$$
$$\mathbf{X}_3 = [\ 2\mathbf{C}_5(5)\ \ 0\ \ \mathbf{C}_5(3)+\mathbf{C}_5(7)\ \ 0\ \ \mathbf{C}_5(1)\ \ 0\ \ \mathbf{C}_5(3)-\mathbf{C}_5(7)\ \ 0]^\top$$
$$\mathbf{X}_4 = [\ 0\ \ 2\mathbf{C}_1(2)\ \ 0\ \ 2\mathbf{C}_1(4)\ \ 0\ \ 2\mathbf{C}_1(6)\ \ 0\ \ 2\mathbf{C}_1(8)]^\top$$
$$\mathbf{X}_5 = [\ 2\mathbf{C}_6(5)\ \ 0\ \ \mathbf{C}_6(3)+\mathbf{C}_6(7)\ \ 0\ \ \mathbf{C}_6(1)\ \ 0\ \ \mathbf{C}_6(3)-\mathbf{C}_6(7)\ \ 0]^\top$$
$$\mathbf{X}_6 = [\ 0\ \ \mathbf{C}_3(2)\ \ 0\ \ \mathbf{C}_3(4)\ \ 0\ \ \mathbf{C}_3(6)\ \ 0\ \ \mathbf{C}_3(8)]^\top$$
$$\mathbf{X}_7 = [\ -\mathbf{C}_4(1)+\mathbf{C}_7(1)\ \ 0\ \ -\mathbf{C}_4(3)+\mathbf{C}_7(3)\ \ 0\ \ -\mathbf{C}_4(5)+\mathbf{C}_7(5)\ \ 0\ \ -\mathbf{C}_4(7)+\mathbf{C}_7(7)\ \ 0]^\top$$

Fig. 2. The 8-point DCT algorithm for input sequence in AI representation belonging to Class B.

integer vectors belonging to the Class X, where $\mathrm{X} \in \{\mathrm{A}, \mathrm{B}, \mathrm{C}\}$. Thus, a complete fast algorithm for the computation of the 2D DCT using AI representation can be expressed in terms of $\mathrm{DCT}_{\mathrm{A}}(\cdot)$, $\mathrm{DCT}_{\mathrm{B}}(\cdot)$, and $\mathrm{DCT}_{\mathrm{C}}(\cdot)$, as shown in Figure 4. Notice that the output are in AI-based representation. The algorithm in Figure 4 along with the FRS is graphically depicted in Figure 5.

The computational cost of the algorithm shown in Figure 4 can be derived by noticing that it demands 10, 4, and 2 instantiations of $\mathrm{DCT}_{\mathrm{A}}$, $\mathrm{DCT}_{\mathrm{B}}$, and $\mathrm{DCT}_{\mathrm{C}}$, respectively. Considering the number of additions and bit-shifting operations required by $\mathrm{DCT}_{\mathrm{A}}$, $\mathrm{DCT}_{\mathrm{B}}$, and $\mathrm{DCT}_{\mathrm{C}}$, the computation of the 2D DCT using AI-based representation requires a total of 892 additions and 92 bit-shifting operations. Arithmetic operation count and comparison are show in Table 3.

3.3 Comparison with Previous Works

Different works [18], [30]–[32] proposed fast algorithms for the AI-based 2D DCT computation claiming to provide error-free computation. However, in previous works, the computation of 2D DCT was based in several instantiations of the 1D DCT with its FRS blocks placed in-between the column- and row-wise DCT computations. Such *intermediate* reconstruction step introduces numerical inaccuracies, which makes the implementation incapable of error-free computation [18], [30]–[32]. This also reflects on deterioration of performance metrics such as throughput, maximum operating frequency, area, and latency as the FRS blocks are computationally intensive and hardware demanding.

Nonetheless, there is one algorithm for full error-free computation of the 2D DCT using the Arai factorization [24]. The Arai algorithm over algebraic integers was employed in two

Input: $\mathbf{x}_n \in \mathbb{Z}_{\mathrm{C}}$ for $n = 0,1,\ldots,7$
Output: $\mathbf{X}_k \in \mathrm{span}(\mathbf{Z})$, for $k = 0,1,\ldots,7$

Stage 1 Outputs:

$$\mathbf{A}_0 = [0\ 0\ \mathbf{x}_0(3)+\mathbf{x}_7(3)\ 0\ 0\ 0\ \mathbf{x}_0(7)+\mathbf{x}_7(7)\ 0]^\top$$
$$\mathbf{A}_1 = [0\ 0\ \mathbf{x}_1(3)+\mathbf{x}_6(3)\ 0\ 0\ 0\ \mathbf{x}_1(7)+\mathbf{x}_6(7)\ 0]^\top$$
$$\mathbf{A}_2 = [0\ 0\ \mathbf{x}_2(3)+\mathbf{x}_5(3)\ 0\ 0\ 0\ \mathbf{x}_2(7)+\mathbf{x}_5(7)\ 0]^\top$$
$$\mathbf{A}_3 = [0\ 0\ \mathbf{x}_3(3)+\mathbf{x}_4(3)\ 0\ 0\ 0\ \mathbf{x}_3(7)+\mathbf{x}_4(7)\ 0]^\top$$
$$\mathbf{A}_4 = [0\ 0\ \mathbf{x}_3(3)-\mathbf{x}_4(3)\ 0\ 0\ 0\ \mathbf{x}_3(7)-\mathbf{x}_4(7)\ 0]^\top$$
$$\mathbf{A}_5 = [0\ 0\ \mathbf{x}_2(3)-\mathbf{x}_5(3)\ 0\ 0\ 0\ \mathbf{x}_2(7)-\mathbf{x}_5(7)\ 0]^\top$$
$$\mathbf{A}_6 = [0\ 0\ \mathbf{x}_1(3)-\mathbf{x}_6(3)\ 0\ 0\ 0\ \mathbf{x}_1(7)-\mathbf{x}_6(7)\ 0]^\top$$
$$\mathbf{A}_7 = [0\ 0\ \mathbf{x}_0(3)-\mathbf{x}_7(3)\ 0\ 0\ 0\ \mathbf{x}_0(7)-\mathbf{x}_7(7)\ 0]^\top$$

Auxiliary additions in Stage 2:

$$a_0 = \mathbf{A}_4(3)+\mathbf{A}_7(7) \quad a_1 = \mathbf{A}_4(7)+\mathbf{A}_7(3) \quad a_2 = \mathbf{A}_4(3)-\mathbf{A}_7(7) \quad a_3 = -\mathbf{A}_4(7)+\mathbf{A}_7(3)$$
$$a_4 = \mathbf{A}_5(3)+\mathbf{A}_6(7) \quad a_5 = \mathbf{A}_5(3)-\mathbf{A}_6(7) \quad a_6 = \mathbf{A}_5(7)+\mathbf{A}_6(3) \quad a_7 = \mathbf{A}_5(7)-\mathbf{A}_6(3)$$

Stage 2 Outputs:

$$\mathbf{B}_0 = [0\ \ 0\ \ \mathbf{A}_0(3)+\mathbf{A}_3(3)\ \ 0\ \ 0\ \ 0\ \ \mathbf{A}_0(7)+\mathbf{A}_3(7)\ \ 0]^\top$$
$$\mathbf{B}_1 = [0\ \ 0\ \ \mathbf{A}_1(3)+\mathbf{A}_2(3)\ \ 0\ \ 0\ \ 0\ \ \mathbf{A}_1(7)+\mathbf{A}_2(7)\ \ 0]^\top$$
$$\mathbf{B}_2 = [0\ \ 0\ \ \mathbf{A}_1(3)-\mathbf{A}_2(3)\ \ 0\ \ 0\ \ 0\ \ \mathbf{A}_1(7)-\mathbf{A}_2(7)\ \ 0]^\top$$
$$\mathbf{B}_3 = [0\ \ 0\ \ \mathbf{A}_0(3)-\mathbf{A}_3(3)\ \ 0\ \ 0\ \ 0\ \ \mathbf{A}_0(7)-\mathbf{A}_3(7)\ \ 0]^\top$$
$$\mathbf{B}_4 = [0\ \ a_0\ \ 0\ \ a_1\ \ 0\ \ a_2\ \ 0\ \ a_3]^\top$$
$$\mathbf{B}_5 = [0\ \ a_4\ \ 0\ \ a_5\ \ 0\ \ a_6\ \ 0\ \ a_7]^\top$$
$$\mathbf{B}_6 = [0\ -a_7\ \ 0\ \ a_6\ \ 0\ -a_5\ \ 0\ \ a_4]^\top$$
$$\mathbf{B}_7 = [0\ \ a_3\ \ 0\ \ -a_2\ \ 0\ \ a_1\ \ 0\ \ -a_0]^\top$$

Auxiliary additions in Stage 3:

$$b_0 = \mathbf{B}_2(3)+\mathbf{B}_3(7) \quad b_1 = \mathbf{B}_2(3)-\mathbf{B}_3(7) \quad b_2 = \mathbf{B}_2(7)+\mathbf{B}_3(3) \quad b_3 = \mathbf{B}_2(7)-\mathbf{B}_3(3)$$

Stage 3 Outputs:

$$\mathbf{C}_0 = [\ 0\ \ 0\ \ \mathbf{B}_0(3)+\mathbf{B}_1(3)\ \ 0\ \ 0\ \ 0\ \ \mathbf{B}_0(7)+\mathbf{B}_1(7)\ \ 0]^\top$$
$$\mathbf{C}_1 = [\ 0\ \ 0\ \ \mathbf{B}_0(3)-\mathbf{B}_1(3)\ \ 0\ \ 0\ \ 0\ \ \mathbf{B}_0(7)-\mathbf{B}_1(7)\ \ 0]^\top$$
$$\mathbf{C}_2 = [\ 2(b_0-b_3)\ \ 0\ \ 0\ \ 0\ \ 2b_2\ \ 0\ \ 0\ \ 0]^\top$$
$$\mathbf{C}_3 = [-2(b_0+b_3)\ \ 0\ \ 0\ \ 0\ \ -2b_1\ \ 0\ \ 0\ \ 0]^\top$$
$$\mathbf{C}_4 = [\ 0\ \ \mathbf{B}_4(2)+\mathbf{B}_6(2)\ \ 0\ \ \mathbf{B}_4(4)+\mathbf{B}_6(4)\ \ 0\ \ \mathbf{B}_4(6)+\mathbf{B}_6(6)\ \ 0\ \ \mathbf{B}_4(8)+\mathbf{B}_6(8)]^\top$$
$$\mathbf{C}_5 = [\ 0\ \ -\mathbf{B}_5(2)+\mathbf{B}_7(2)\ \ 0\ \ -\mathbf{B}_5(4)+\mathbf{B}_7(4)\ \ 0\ \ -\mathbf{B}_5(6)+\mathbf{B}_7(6)\ \ 0\ \ -\mathbf{B}_5(8)+\mathbf{B}_7(8)]^\top$$
$$\mathbf{C}_6 = [\ 0\ \ \mathbf{B}_4(2)-\mathbf{B}_6(2)\ \ 0\ \ \mathbf{B}_4(4)-\mathbf{B}_6(4)\ \ 0\ \ \mathbf{B}_4(6)-\mathbf{B}_6(6)\ \ 0\ \ \mathbf{B}_4(8)-\mathbf{B}_6(8)]^\top$$
$$\mathbf{C}_7 = [\ 0\ \ \mathbf{B}_5(2)+\mathbf{B}_7(2)\ \ 0\ \ \mathbf{B}_5(4)+\mathbf{B}_7(4)\ \ 0\ \ \mathbf{B}_5(6)+\mathbf{B}_7(6)\ \ 0\ \ \mathbf{B}_5(8)+\mathbf{B}_7(8)]^\top$$

Output:

$$\mathbf{X}_0 = [\ 0\ \ 0\ \ 2\mathbf{C}_0(3)\ \ 0\ \ 0\ \ 0\ \ 2\mathbf{C}_0(7)\ \ 0]^\top$$
$$\mathbf{X}_4 = [\ 0\ \ 0\ \ 2\mathbf{C}_1(3)\ \ 0\ \ 0\ \ 0\ \ 2\mathbf{C}_1(7)\ \ 0]^\top$$
$$\mathbf{X}_2 = [\ \mathbf{C}_2(1)\ \ 0\ \ 0\ \ 0\ \ \mathbf{C}_2(5)\ \ 0\ \ 0\ \ 0]^\top$$
$$\mathbf{X}_6 = [\ \mathbf{C}_3(1)\ \ 0\ \ 0\ \ 0\ \ \mathbf{C}_3(5)\ \ 0\ \ 0\ \ 0]^\top$$
$$\mathbf{X}_7 = [\ 0\ \ -\mathbf{C}_4(2)+\mathbf{C}_7(2)\ \ 0\ \ -\mathbf{C}_4(4)+\mathbf{C}_7(4)\ \ 0\ \ -\mathbf{C}_4(6)+\mathbf{C}_7(6)\ \ 0\ \ -\mathbf{C}_4(8)+\mathbf{C}_7(8)]^\top$$
$$\mathbf{X}_3 = [\ 0\ \ \mathbf{C}_5(4)+\mathbf{C}_5(6)\ \ 0\ \ \mathbf{C}_5(2)+\mathbf{C}_5(8)\ \ 0\ \ \mathbf{C}_5(2)-\mathbf{C}_5(8)\ \ 0\ \ \mathbf{C}_5(4)-\mathbf{C}_5(6)]^\top$$
$$\mathbf{X}_5 = [\ 0\ \ \mathbf{C}_6(4)+\mathbf{C}_6(6)\ \ 0\ \ \mathbf{C}_6(2)+\mathbf{C}_6(8)\ \ 0\ \ \mathbf{C}_6(2)-\mathbf{C}_6(8)\ \ 0\ \ \mathbf{C}_6(4)-\mathbf{C}_6(6)]^\top$$
$$\mathbf{X}_1 = [\ 0\ \ \mathbf{C}_4(2)+\mathbf{C}_7(2)\ \ 0\ \ \mathbf{C}_4(4)+\mathbf{C}_7(4)\ \ 0\ \ \mathbf{C}_4(6)+\mathbf{C}_7(6)\ \ 0\ \ \mathbf{C}_4(8)+\mathbf{C}_7(8)]^\top$$

Fig. 3. The 8-point DCT algorithm for input sequence in AI representation belonging to Class C.

Input: $\mathbf{x}_{m,n} \in \mathbb{Z}_{\mathrm{A}}$ for $m,n = 0,1,\ldots,7$
Output: $\mathbf{X}_{k,l} \in \mathrm{span}(\mathbf{Z})$, for $k,l = 0,1,\ldots,7$

Compute the 1D DCT over the columns of $\mathbf{x}_{m,n}$:

$$\mathbf{X}_{\cdot,l} = \mathrm{DCT}_{\mathrm{A}}(\mathbf{x}_{\cdot,l}), \qquad l = 0,1,\ldots,7$$

Compute the 1D DCT over the rows of $\mathbf{x}_{m,n}$:

$$\mathbf{X}_{k,\cdot} = \mathrm{DCT}_{\mathrm{A}}(\mathbf{X}_{k,\cdot}), \qquad k = 0,4$$
$$\mathbf{X}_{k,\cdot} = \mathrm{DCT}_{\mathrm{B}}(\mathbf{X}_{k,\cdot}), \qquad k = 1,3,5,7$$
$$\mathbf{X}_{k,\cdot} = \mathrm{DCT}_{\mathrm{C}}(\mathbf{X}_{k,\cdot}), \qquad k = 2,6$$

Return $\mathbf{X}_{k,l}$

Fig. 4. The 8-point 2D DCT AI-based fast algorithm.

different architectures: a row-parallel 8×8 2D DCT architecture using algebraic integer-based exact computation [22] and a single-channel architecture for algebraic integer-based 8×8 2D DCT computation [23]. The work in [22] is a extension of the work in [33], where the first error-free 2D DCT was proposed. Since these architectures are based on the Arai algorithm [24], they are capable of providing the non-uniformly scaled 2D DCT spectrum. This is due to the fact that Arai 1D DCT is capable of only provide a scaled spectrum, thus eliminating some multiplicands required for the non-scaled 1D DCT. The implementations in [22] and [23] can be modified in order to provide exact 2D DCT spectrum at the cost of changes on the FRS.

However, our proposed architecture is error-free up to the FRS after the application of the 1D DCT in both dimensions and does not share the intricate details of non-uniform scale as in [22], [23]. Table 3 summarizes the comparison between the proposed method and previous works.

4 Final Reconstruction Step

The final reconstruction step (FRS) block performs the AI decoding described in (2). It maps AI quantities back to fixed-

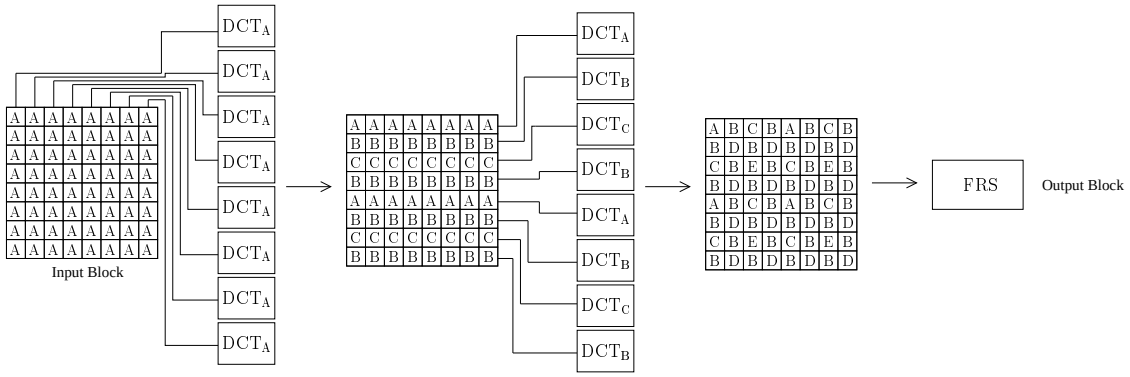


Fig. 5. Graphical representation of the whole 2D DCT parallel architecture using AI representation including FRS. The blocks $\mathrm{DCT_A}$, $\mathrm{DCT_B}$, and $\mathrm{DCT_C}$ represent the 1D DCT in AI-based representation for inputs in the classes A, B, and C, respectively, and are implemented by the algorithms in Figure 3 in [25], Figure 2, and Figure 3.

TABLE 3
Comparison for fast algorithms for the computation of 2D 8-point DCT with algebraic integer theory

Algorithm	Error-free?	Complexity		Uniform Scale?
		Add	Shift	
Dimitrov *et al.* [18]	No	384	0	No
Madanayake *et al.* [22]	Yes	1064	32	No
Edirisuriya *et al.* [23]	Yes	1064	32	No
Pradini *et al.* [30]	No	304	0	Yes
Wahid *et al.* [31]	No	384	64	Yes
Wahid *et al.* [32]	No	384	64	No
Rajapaksha *et al.* [34]	Yes	1064	32	No
Fu *et al.* [35]	No	352	360	Yes
Proposed	Yes	892	92	Yes

point representation. In the proposed design implementation of the 2D DCT, the FRS is performed only at the very final stage after all the computations required by the 2D DCT are completed over the AI representation. No intermediate reconstructions are required.

In this section, we consider two methods for AI decoding: (i) dyadic approximation [3] and (ii) the expansion factor method [3], [36] with a modified cost function for optimized results. The dyadic approximation method is suitable for scenarios where the exact spectrum is required, whereas the expansion factor is applicable when a scaled version is acceptable.

In both cases, the FRS is reduced to the evaluation of the product of a few integers by known integer constants. This operation can be understood as an instance of the multiple constant multiplication (MCM) problem with a small number of constants—no more than four, as will be clear in the next sections. Several methods for MCM evaluation with different constants have been developed by means of optimization and graph theory [37]–[40]. For solving the present MCM problems, we employ the method described in [41], which is based on number recoding and subexpression factorization and has not been applied to the design of FRS block in previous works [18], [22], [23], [30]–[32], [34], [35]. MCM evaluation can save up to 40% of area in FPGA implementation and provide faster computation when compared to routine methods [41].

4.1 Dyadic Approximation Method

The irrational quantities required by the FRS can be approximated with arbitrary precision by dyadic integers [3]. Dyadic integers are of the form $p/2^k$, where p is an odd integer and $k \in \mathbb{N}$. Thus, they can be efficiently implemented in hardware [2], [3], [42].

In order to implement the FRS with a minimum arithmetic cost, we first approximate each of the involved irrational constants by a dyadic integer. The accuracy of its approximation is determined by the wordlength employed, which can vary according to specific applications. For the sake of clarity, adopting the 11-bit wordlength, we have:

$$\boldsymbol{\zeta} \approx \begin{bmatrix} 1 & \frac{4017}{2^{11}} & \frac{3784}{2^{11}} & \frac{3406}{2^{11}} & \frac{2896}{2^{11}} & \frac{2276}{2^{11}} & \frac{1567}{2^{11}} & \frac{799}{2^{11}} \end{bmatrix}^\top. \quad (4)$$

For the decoding of 2D DCT output coefficients into fixed-point representation, we need to consider the quantities in each AI number class and design specific algorithms for each class. Considering classes shown in Table 2, we have the algorithms shown in Table 4 for the approximate $\boldsymbol{\zeta}$ with 11-bit wordlength. The operation $x \ll k$ denotes the left shift of k bits over the integer quantity x (i.e., $x \cdot 2^k$); whereas $x \gg k$ denotes the right shift of k bits.

4.2 Expansion Factor

The expansion factor method returns a scaled version of DCT spectrum and is based on finding an appropriate real constant $\alpha^* > 1$ such that $\alpha^* \cdot \boldsymbol{\zeta}$ is as close as possible to a vector of integers. This provides means of performing the decoding operation with multiplications by known integer constants that can be efficiently performed with:

$$\alpha^* \cdot f_{\mathrm{dec}}(\mathbf{x}; \boldsymbol{\zeta}) \approx \mathbf{x}^\top \cdot \mathrm{round}(\alpha^* \cdot \boldsymbol{\zeta}), \quad (5)$$

where $\mathrm{round}(\cdot)$ operates over each component of its vector argument. Previous works have considered an expansion factor α^* satisfying the following optimization problem:

$$\alpha^* = \arg\min_{\alpha > 1} \| \alpha \cdot \boldsymbol{\zeta} - \mathrm{round}(\alpha \cdot \boldsymbol{\zeta}) \|. \quad (6)$$

In this context, all the components of the basis vector $\boldsymbol{\zeta}$ are taken into account with the same weight. However, this is not suitable for this problem. In fact, the required number of multiplications by each of the components of $\boldsymbol{\zeta}$ is not uniform. This can be seen by the output pattern of the 2D DCT coefficients in Figure 1(c). Therefore, in order to obtain a more precise estimation for α^*,

TABLE 4
Fast algorithms for FRS for dyadic approximation for 11-bit wordlength and its arithmetic cost

Class	Algorithm	Output $f_{\text{dec}}(x;\boldsymbol{\zeta})$	Arithmetic Cost	
			Additions	Shifts
A	$f_{\text{enc}}(x;\boldsymbol{\zeta}) = u_0$	u_0	0	0
B	$t_1 = u_1 - u_3 \ll 2$ $t_2 = u_3 - u_5 \ll 1$ $t_3 = t_2 - u_7 \ll 4$ $t_4 = -u_7 + u_1$ $t_5 = u_5 + u_7 \ll 2$ $t_6 = u_3 + u_1$ $t_7 = t_4 - t_1 \ll 4$ $t_8 = -t_1 + t_5 \ll 2$ $t_9 = -t_2 + t_6 \ll 2$ $t_{10} = -t_3 \ll 1 + t_7$ $t_{11} = t_3 + t_8 \ll 2$ $t_{12} = t_{11} \ll 4 + t_{10}$ $f_{\text{dec}}(x;\boldsymbol{\zeta}) = (t_9 \ll 10 + t_{12}) \gg 11$	$\frac{4017\cdot u_1 + 3406\cdot u_3 + 2276\cdot u_5 + 799\cdot u_7}{2^{11}}$	13	12
C	$t_1 = u_2 \ll 3 + u_6$ $t_2 = u_6 \ll 5 - u_2$ $t_3 = t_1 - t_1 \ll 5$ $t_4 = t_2 + t_1 \ll 3$ $f_{\text{dec}}(x;\boldsymbol{\zeta}) = (t_3 + t_4 \ll 6) \gg 11$	$\frac{3784\cdot u_2 + 1567\cdot u_6}{2^{11}}$	5	6
D	$t_1 = u_2 \ll 3 - u_6$ $t_2 = u_4 \ll 2 - u_4$ $t_3 = t_1 + t_2 \ll 1$ $t_4 = u_4 - u_2 \ll 2$ $t_5 = t_4 \ll 4 + t_1$ $t_6 = t_5 - t_3 \ll 5$ $t_7 = t_3 + u_6 \ll 2$ $f_{\text{dec}}(x;\boldsymbol{\zeta}) = (t_6 + t_7 \ll 9) \gg 11 + u_0$	$\frac{2048\cdot u_0 + 3784\cdot u_2 + 2896\cdot u_4 + 1567\cdot u_6}{2^{11}}$	9	9
E	$t_1 = -u_4 + u_4 \ll 4$ $t_2 = t_1 \ll 2 + t_1$ $f_{\text{dec}}(x;\boldsymbol{\zeta}) = (u_4 \ll 12 - t_2 \ll 4) \gg 11 + u_0$	$\frac{2048\cdot u_0 + 2896\cdot u_4}{2^{11}}$	4	5

we must take into account the relative frequency of occurrence of multiplications of the coefficients of $\boldsymbol{\zeta}$. This results in the following optimization problem:

$$\alpha^* = \arg\min_{\alpha>1} \| \boldsymbol{f}^\top \cdot (\alpha\cdot\boldsymbol{\zeta} - \text{round}(\alpha\cdot\boldsymbol{\zeta})) \|, \tag{7}$$

where $\boldsymbol{f}$ represents the vector with the relative frequency of the occurrence of multiplications by each coefficient of $\boldsymbol{\zeta}$. Clearly, $\boldsymbol{f}$ has the same dimension as $\boldsymbol{\zeta}$, and for this particular case of 2D DCT with the representation proposed in [25], we have that

$$\boldsymbol{f} = \left[\tfrac{24}{220} \quad \tfrac{32}{220} \quad \tfrac{24}{220} \quad \tfrac{32}{220} \quad \tfrac{20}{220} \quad \tfrac{32}{220} \quad \tfrac{24}{220} \quad \tfrac{32}{220}\right]^\top. \tag{8}$$

The problem in (7) is non-linear and has no closed solution in terms of simple algebraic functions. In order to solve (7) we employ exhaustive search methods. Although exhaustive search methods are not considered to be efficient for solving optimization problems in general, the search space for finding suitable expansion factors can be made small enough without imposing prohibitive limitations.

For instance, considering the search space $[0, 2048]$ (11-bit wordlength) and a step size of 10^{-2}, we obtain the optimal value of $\alpha^* = 1844.95$, leading to

$$1844.95\cdot\boldsymbol{\zeta} = \begin{bmatrix} 1844.95 \\ 3618.97\ldots \\ 3409.00\ldots \\ 3068.02\ldots \\ 2609.13\ldots \\ 2049.98\ldots \\ 1412.05\ldots \\ 719.85\ldots \end{bmatrix} \approx \begin{bmatrix} 1845 \\ 3619 \\ 3409 \\ 3068 \\ 2609 \\ 2050 \\ 1412 \\ 720 \end{bmatrix}. \tag{9}$$

Table 5 shows the optimal expansion factors for some wordlengths N for searches with steps of 10^{-2}. Minimum and maximum relative errors are also shown.

For the decoding of 2D DCT output coefficient into fixed-point representation, we need to consider the different number classes shown in Table 2 and design specific algorithms for them. We adopted the MCM method described in [41]. We derived the algorithms shown in Table 6 for the optimal constant $\alpha^* = 1844.95$ with 11-bit wordlength.

TABLE 5
Scale factors and maximum/minimum relative errors

N	α^*	$\zeta - \frac{\text{round}(\alpha^* \cdot \zeta)}{\alpha^*}$	
		$\min(\cdot)$	$\max(\cdot)$
5	25.99	$5.22 \cdot 10^{-4}$	$9.41 \cdot 10^{-3}$
6	43.28	$5.18 \cdot 10^{-3}$	$6.47 \cdot 10^{-3}$
7	69.26	$1.81 \cdot 10^{-4}$	$3.75 \cdot 10^{-3}$
8	253.83	$1.96 \cdot 10^{-4}$	$1.08 \cdot 10^{-3}$
9	341.01	$1.8 \cdot 10^{-11}$	$7.65 \cdot 10^{-4}$
10	341.01	$1.8 \cdot 10^{-11}$	$7.65 \cdot 10^{-4}$
11	1844.95	$5.03 \cdot 10^{-4}$	$8.31 \cdot 10^{-5}$
12	1844.95	$5.03 \cdot 10^{-4}$	$8.31 \cdot 10^{-5}$

5 Digital Implementation

5.1 FPGA Implementation

A fully-parallel architecture for the real-time implementation of the proposed 2D DCT using AI encoding has been designed, simulated and implemented using field programmable gate array (FPGA) technology. The architecture assumes 64 parallel input channels pertaining to the 64 locations of an 8×8 matrix of pixel values, which are assumed to be of 8-bit signed values using twos complement format. The inputs are assumed to be normalized in the range -1 to $127/128$. The AI encoded architectures corresponding to DCT_A, DCT_B, and DCT_C are realized in parallelized digital hardware.

The 64 output coefficients are maintained in the AI-encoded infinite precision format up to the FRS block for conversion to fixed-point representation for subsequent processing. The algorithms for multiple constant multiplication described in Table 4 and 6 were used in the FRS design. The FRS was also made fully-parallel. Both the AI-encoded DCT architecture as well as the FRS are fine-grain pipelined for low critical path delay.

The resulting digital design was simulated using bit-true and cycle accurate models using $1.5 \cdot 10^4$ number of randomly generated 8×8 input vectors to verify correct operation. The verified design was thereafter targeted to a Xilinx Virtex-6 XC6VLX240T-1FFG1156 FPGA device installed on a Xilinx ML605 evaluation platform. The design was subjected to physical implementation and test using $1.5 \cdot 10^4$ test matrices provided to the implementation using stepped hardware co-simulation on the JTAG port. The FPGA resources consumption and metrics are shown in Table 7 along with competitors designs available metrics in Table 3[1]. The throughput is calculated as the number of output coefficients in each cycle and the designs in the works in Table 7 are classified into fully parallel 64 coefficients per clock cycle (FPar64); row parallel 8 coefficients per clock cycle (RPar8); fully serial 1 coefficient per clock cycle (FSer1).

The block and pixel rate represent the number of 8×8 blocks and pixels processed per second. The maximum clock frequency for potential real-time operation is 360 MHz. This implies a throughput of 360 million 2D DCT computations of size 8×8 every second, if this core is used as part of a larger image/video processing system designed on the same FPGA technology. This throughput is equivalent to a pixel rate of 23,040 billion pixels/second, and a sustained data processing rate of 184.32 Gbps (internal to the core). Getting such a high data rate into the processing core is a challenging problem itself. The intention here is to give the reader a sense of the capabilities of the FPGA realization, if a suitable data source and algorithm is in fact available to feed it. Given that the proposed core is expected to be part of a larger ultra-high definition video processing system, the obtained throughput from the FPGA implementation is quite sufficient for today's most challenging UHD video applications.

Note that the proposed design achieves the highest maximum frequency among all the competitors designs. The work in [31] proposes a 2D DCT algorithms, but only the FPGA implementations results for 1D DCT are presented and therefore used in Table 7. The only works containing complete error-free implementation of 2D DCT are [22], [23] and [34]. The proposed design demands a total of 26,000 registers and 30,200 look-up tables (LUTs) used for logic against 10,282 registers and 12,007 LUTs required by the design in [22]. Although the number of registers and LUTs used are around three times larger, the proposed design offers 100 times higher block processing rate and 1000 times higher pixel rate compared to [22].

1. Note, however, that we do not include the work [18] in Table 7. This is because the work in [18] does not present any FPGA implementation metric.

5.2 ASIC Synthesis

Apart from the FPGA implementation, the design is subjected to the application specific integrated circuit (ASIC) synthesis. The employed ASIC technology is the AMS 180 nm with the software Genus version 15.23. The supply voltage (V_{sup}) for the ASIC synthesis is 1.8 V. Table 8 results shows the ASIC metrics.

Note that because the proposed architecture is capable of computing the DCT coefficients in a parallel fashion reducing the critical path, the maximum operating frequency achieved is very high compared to its competitors. The proposed scheme requires around 3–4 times the area in the design in [35]. However, this area requirement is translated into a thousandfold pixel rate improvement.

6 Conclusions

In this paper we proposed a new error-free fast algorithm for the computation of the 2D DCT. The algorithm is based on algebraic integer representation proposed in [25]. The proposed fast algorithm does not require any intermediate FRS between the the column- and row-wise 1D DCT calls.

This work provided FPGA and ASIC implementation results. The FPGA results shows that the proposed fast algorithm provided higher maximum operating frequency when compared to competitors. This is important in applications requiring high data throughput and real time processing of images or videos. Future works may include the design of 3D DCT algorithms using the algebraic integer based numerical representation employed in this work.

References

[1] N. Ahmed, T. Natarajan, and K. R. Rao, "Discrete cosine transform," *IEEE Transactions on Computers*, vol. C-23, no. 1, pp. 90–93, Jan. 1974.

[2] A. V. Oppenheim, R. W. Schafer, M. T. Yoder, and W. T. Padgett, *Discrete Time Signal Processing*, 3rd ed., A. V. Oppenheim, Ed. Upper Saddle River, NJ: Prentice-Hall, Inc., Aug. 2009, vol. 1.

[3] V. Britanak, P. Yip, and K. R. Rao, *Discrete Cosine and Sine Transforms*. Academic Press, 2007.

[4] R. C. Gonzalez and R. E. Woods, *Digital Image Processing*, 2nd ed. Prentice-Hall, Inc., 2001.

TABLE 6
Fast algorithms for FRS for expansion factor method for 11-bit wordlength ($\alpha^* = 1844.95$) and its arithmetic cost

Class	Algorithm	Output $f_{\text{dec}}(x;\boldsymbol{\zeta})$	Arithmetic Cost	
			Additions	Shifts
A	$t_1 = -u_0 + u_0 \ll 2$ $t_2 = u_0 \ll 11 + u_0$ $t_3 = t_1 \ll 4 + t_1$ $f_{\text{enc}}(x;\boldsymbol{\zeta}) = t_2 - t_3 \ll 2$	$1845 \cdot u_0$	4	4
B	$t_1 = t_2 + t_3 \ll 8$ $t_2 = -u_7 \ll 4 + u_1$ $t_3 = u_3 \ll 2 + u_7$ $t_4 = u_5 - u_3 \ll 1$ $t_5 = -u_1 \ll 4 + u_1$ $t_6 = u_1 \ll 1 + u_5$ $t_7 = t_4 + t_5 \ll 4$ $t_8 = t_7 \ll 1 - t_1$ $t_9 = t_6 \ll 9 + t_1$ $f_{\text{dec}}(x;\boldsymbol{\zeta}) = t_8 + t_9 \ll 2$	$3619 \cdot u_1 + 3068 \cdot u_3 + 2050 \cdot u_5 + 720 \cdot u_7$	10	10
C	$t_1 = u_2 - u_6 \ll 1$ $t_2 = u_6 \ll 2 + u_2$ $t_3 = u_2 + u_2 \ll 8$ $t_4 = t_2 + t_1 \ll 6$ $t_5 = t_1 - t_1 \ll 2$ $t_6 = t_5 \ll 8 + t_4$ $f_{\text{dec}}(x;\boldsymbol{\zeta}) = t_3 \ll 4 + t_6$	$3409 \cdot u_2 + 1412 \cdot u_6$	7	7
D	$t_1 = -t_5 \ll 7 + t_6$ $t_2 = u_0 + u_4$ $t_3 = u_2 \ll 1 + u_0$ $t_4 = u_0 \ll 2 - u_4$ $t_5 = u_6 + u_2 \ll 1$ $t_6 = t_3 + t_4 \ll 2$ $t_7 = u_6 \ll 2 + u_2$ $t_8 = u_4 \ll 2 - u_6$ $t_9 = t_7 + t_2$ $t_{10} = t_2 \ll 4 + t_8$ $t_{11} = t_10 \ll 7 + t_9$ $t_{12} = -t_1 \ll 2 + t_1$ $f_{\text{dec}}(x;\boldsymbol{\zeta}) = t_{11} + t_{12} \ll 2$	$1845 \cdot u_0 + 3049 \cdot u_2 + 2609 \cdot u_4 + 1412 \cdot u_6$	13	12
E	$t_1 = u_0 \ll 4 + t_3$ $t_2 = u_0 + u_4$ $t_3 = -u_4 \ll 2 + u_0$ $t_4 = u_4 \ll 9 + t_2$ $t_5 = t_1 - t_1 \ll 2$ $t_6 = t_4 + t_2 \ll 11$ $f_{\text{dec}}(x;\boldsymbol{\zeta}) = t_5 \ll 2 + t_6$	$1845 \cdot u_0 + 2609 \cdot u_4$	7	6

TABLE 7
Comparison of FPGA implementation metrics. The FPar64 means fully parallel 64 coefficients per clock cycle, RPar8 means row parallel 8 coefficients per clock cycle, and FSer1 means fully serial 1 coefficient per clock cycle

Method	Max. Freq.	Board	Throughput	Block Rate	Pixel Rate	No. of Slice LUT'S	No. of Slice registers
Madanayake *et al.* [22]	307 MHz	Xilinx Virtex-6 (XC6VLX240T)	RPar8	$4.78 \cdot 10^6$	$3.83 \cdot 10^7$	12,007	10,282
Edirisuriya *et al.* [23]	316 MHz	Xilinx Virtex-6 (XC6VLX240T)	FSer1	$4.93 \cdot 10^6$	$4.93 \cdot 10^6$	–	–
Wahid *et al.* [31] (1D DCT)	36.7 MHz	Actel A500K (A500K050)	–	–	–	–	–
Wahid *et al.* [32]	101 MHz	Xilinx Virtex-E (XCV200E-8)	–	–	–	–	–
Rajapaksha *et al.* [34]	302 MHz	Achronix SPD60	RPar8	$4.71 \cdot 10^6$	$3.77 \cdot 10^7$	–	–
Proposed	360 MHz	Xilinx Virtex-6 (XC6VLX240T)	FPar64	$3.6 \cdot 10^8$	$2.30 \cdot 10^{10}$	30,200	26,000

TABLE 8
Comparison of ASIC implementation metrics. The FPar64 means fully parallel 64 coefficients per clock cycle, RPar8 means row parallel 8 coefficients per clock cycle, and FSer1 means fully serial 1 coefficient per clock cycle

Method	Max. Freq.	Technology	Throughput	Block Rate	Pixel Rate	Area (mm^2)	Dynamic Power (W)	Norm. Dyn. Power ($mW/MHz \cdot V^2_{sup}$)
Pradini *et al.* [30]	210 MHz	0.18 μm CMOS	FPar64	$4.20 \cdot 10^7$	$2.68 \cdot 10^9$	–	–	–
Fu *et al.* [35]	75 MHz	0.18 μm CMOS	FSer1	$1.71 \cdot 10^6$	$7.50 \cdot 10^7$	2.16	–	–
Proposed	893 MHz	0.18 μm CMOS	FPar64	$8.93 \cdot 10^8$	$5.71 \cdot 10^{10}$	7.22	11.85	13.269

[5] S. K. Gupta, J. Jain, and R. Pachauri, "Improved noise cancellation in discrete cosine transform domain using adaptive block LMS filter," *International Journal of Engineering Science and Advanced Technology*, vol. 2, no. 3, pp. 498–502, Jun. 2012.

[6] Q. C. S. An and C. Wang, "A computation structure for 2-D DCT watermarking," in *52nd IEEE International Midwest Symposium on Circuits and Systems*, 2009, pp. 577–580.

[7] J. Xiao and Y. Wang, "Toward a better understanding of DCT coefficients in watermarking," in *Pacific-Asia Workshop on Computational Intelligence and Industrial Application*, vol. 2, 2008, pp. 206–209.

[8] V. Bhaskaran and K. Konstantinides, *Image and Video Compression Standards*. Kluwer Academic Publishers, Jun. 1997.

[9] G. K. Wallace, "The JPEG still picture compression standard," *IEEE Transactions on Consumer Electronics*, vol. 38, no. 1, pp. xviii–xxxiv, Feb. 1992.

[10] N. Roma and L. Sousa, "Efficient hybrid DCT-domain algorithm for video spatial downscaling," *EURASIP Journal on Advances in Signal Processing*, vol. 2007, no. 57291, Jun. 2007.

[11] T. Wiegand, G. J. Sullivan, G. Bjontegaard, and A. Luthra, "Overview of the H.264/AVC video coding standard," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 13, no. 7, pp. 560–576, Jul. 2003.

[12] M. T. Pourazad, C. Doutre, M. Azimi, and P. Nasiopoulos, "HEVC: The new gold standard for video compression: How does HEVC compare with H.264/AVC?" *IEEE Consumer Electronics Magazine*, vol. 1, no. 3, pp. 36–46, Jul. 2012.

[13] C. Loeffler, A. Ligtenberg, and G. S. Moschytz, "A practical fast 1-D DCT algorithms with 11 multiplications," in *International Conference on Acoustics, Speech, and Signal Processing*, vol. 2, May 1989, pp. 988–991.

[14] M. T. Heideman and C. S. Burrus, *Multiplicative complexity, convolution, and the DFT*. New York: Springer-Verlag, 1988, originally presented as the author's thesis (Ph. D.–Rice University) under title: Applications of multiplicative complexity theory to convolution and the discrete Fourier transform.

[15] P. Duhamel and H. H'Mida, "New 2^n DCT algorithms suitable for VLSI implementation," in *IEEE International Conference on Acoustics, Speech, and Signal Processing*, vol. 12, 1987, pp. 1805–1808.

[16] J. H. Cozzens and L. A. Finkelstein, "Computing the discrete Fourier transform using residue number systems in a ring of algebraic integers," *IEEE Transactions on Information Theory*, vol. 31, no. 5, pp. 580–588, Sep. 1985.

[17] ——, "Range and error analysis for a fast Fourier transform computed over $Z[\omega]$," *IEEE Transactions on Information Theory*, vol. 33, no. 4, p. 9, Jul. 1987.

[18] V. Dimitrov, K. A. Wahid, and G. Jullien, "Multiplication-free 8×8 2D DCT architecture using algebraic integer encoding," *Electronics Letters*, vol. 40, no. 20, pp. 1310–1311, Sep. 2004.

[19] V. Dimitrov and K. A. Wahid, "On the error-free computation of fast cosine transform," *Information Theories and Applications*, vol. 12, no. 4, pp. 321–327, 2005.

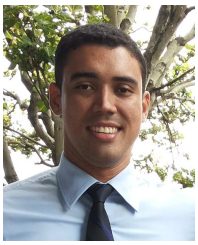
[20] K. A. Wahid, V. S. Dimitrov, and G. A. Jullien, "On the error-free realization of a scaled DCT algorithm and its VLSI implementation," *IEEE Transactions on Circuits and Systems II: Express Briefs*, vol. 54, no. 8, pp. 700–704, Jul. 2007.

[21] K. Wahid, *Error-free Implementation of the Discrete Cosine Transform: Algorithms and Architectures using Multidimensional Algebraic Integer Quantization*. University of Saskatchewan: Lambert Academic Publishing, Nov. 2010.

[22] A. Madanayake, R. J. Cintra, D. Onen, V. S. Dimitrov, N. T. Rajapaksha, L. T. Bruton, and A. Edirisuriya, "A row parallel 8×8 2D DCT architecture using algebraic integer based exact arithmetic," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 22, no. 6, pp. 915–929, Jun. 2012.

[23] 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 Transactions on Circuits and Systems for*

Video Technology, vol. 23, no. 12, pp. 2083–2089, Jun. 2013.
[24] Y. Arai, T. Agui, and M. Nakajima, "A fast DCT-SQ scheme for images," *IEICE Transactions*, vol. E71, no. 11, pp. 1095–1097, Nov. 1988.
[25] D. F. G. Coelho, R. J. Cintra, S. Kulasekera, A. Madanayake, and V. S. Dimitrov, "Error-free computation of 8-point discrete cosine transform based on the Loeffler factorisation and algebraic integers," *IET Signal Processing*, vol. 10, no. 2, Mar. 2016.
[26] A. V. Oppenheim, R. W. Schafer, and J. R. Buck, *Discrete-time signal processing*, 2nd ed., A. V. Oppenheim, Ed. Upper Saddle River, NJ: Prentice-Hall, Inc., 1999, vol. 1.
[27] S. An and C. Wang, "A computation structure for 2-D DCT watermarking," in *52nd IEEE International Midwest Symposium on Circuits and Systems*, Ag 2009, pp. 577–580.
[28] F. M. Bayer and R. J. Cintra, "Image compression via a fast DCT approximation," *IEEE Latin America Transactions*, vol. 8, no. 6, pp. 708–713, Jan. 2011.
[29] J. Goebel, G. Paim, L. Agostini, B. Zatt, and M. Porto, "An HEVC multi-size DCT hardware with constant throughput and supporting heterogeneous CUs," in *IEEE International Symposium on Circuits and Systems (ISCAS)*, May 2016, pp. 2202–2205.
[30] A. Pradini, T. M. Roffi, R. Dirza, and T. Adiono, "VLSI design of a high-throughput discrete cosine transform for image compression systems," in *International Conference on Electrical Engineering and Informatics (ICEEI)*, Jul. 2011, pp. 1–6.
[31] K. Wahid, V. Dimitrov, and G. Jullien, "Error-free computation of 8×8 2D DCT and IDCT using two-dimensional algebraic integer quantization," in *17th IEEE Symposium on Computer Arithmetic (ARITH'05)*, Jun. 2005, pp. 214–221.
[32] ——, "New encoding of 8×8 DCT to make H.264 lossless," in *IEEE Asia Pacific Conference on Circuits and Systems*, 2006, pp. 780–783.
[33] H. L. P. A. Madanayake, R. J. Cintra, D. Onen, V. S. Dimitrov, and L. T. Bruton, "Algebraic integer based 8×8 2-D DCT architecture for digital video processing," in *IEEE International Symposium of Circuits and Systems (ISCAS)*, May 2011, pp. 1247–1250.
[34] N. Rajapaksha, A. Edirisuriya, A. Madanayake, R. J. Cintra, D. Onen, I. Amer, and V. S. Dimitrov, "Asynchronous realization of algebraic integer-based 2D DCT using achronix speedster SPD60 FPGA," *Journal of Electrical and Computer Engineering*, Feb. 2013, article ID 834793.
[35] M. Fu, G. A. Jullien, V. S. Dimitrov, and M. Ahmadi, "A low-power DCT IP core based on 2D algebraic integer encoding," in *IEEE International Symposium on Circuits and Systems (ISCAS)*, vol. 2, May 2004, pp. II–765–8 Vol.2.
[36] G. Plonka, "A global method for invertible integer DCT and integer wavelet algorithms," *Applied and Computational Harmonic Analysis*, vol. 16, no. 2, pp. 90–110, Mar. 2003.
[37] S. Athar and O. Gustafsson, "Optimization of AIQ representations for low complexity wavelet transforms," in *20th European Conference on Circuit Theory and Design (ECCTD)*, Aug. 2011, pp. 314–317.
[38] O. Gustafsson and L. Wanhammar, "A novel approach to multiple constant multiplication using minimum spanning trees," in *The 45th Midwest Symposium on Circuits and Systems (MWSCAS)*, vol. 3, Aug. 2002, pp. 652–655.
[39] O. Gustafsson, H. Ohlsson, and L. Wanhammar, "Improved multiple constant multiplication using a minimum spanning tree," in *Thirty-Eighth Asilomar Conference on Signals, Systems and Computers*, vol. 1, no. 1, Nov. 2004, pp. 63–66.
[40] O. Gustafsson, "Towards optimal multiple constant multiplication: A hypergraph approach," in *The 42nd Asilomar Conference on Signals, Systems and Computers*, 2008, pp. 1805–1809.
[41] N. Boullis and A. Tisserand, "Some optimizations of hardware multiplication by constant matrices," *IEEE Transactions on Computers*, vol. 54, no. 10, pp. 1271–1282, Oct. 2005.
[42] R. E. Blahut, *Fast Algorithms for Digital Signal Processing*. Cambridge University Press, Jun. 2010.



Diego F. G. Coelho received the B.Sc. degree in Electronics Engineering and M.Sc degree in Statistics from the Universidade Federal de Pernambuco (UFPE), Recife, Brazil, in 2012 and 2015, respectively, and the Ph.D. degree from University of Calgary, Calgary, Canada, in 2018. His interests include digital signal and image processing, information theory, optimization, matrix computation, numerical analysis, high performance computing, and 3D image reconstruction. Dr. Coelho is currently with Microsemi Corporation, Calgary, Canada.

Sushmabhargavi Nimmalapalli was born in Hyderabad, India, in 1995. She received the B.Tech.Eng. degree in Electronics and Communication Engineering (First Division with Distinction) from Kakatiya Institute of Technology and Science, Warangal, India, in 2016. She is currently working towards the M.S degree in Electrical and Computer Engineering at the University of Akron, in the area of signal processing. From 2015 to 2016, she worked as a project team leader at the Department of Electronics and Communication Engineering, Kakatiya Institute of Technology and Science, for a project based on image processing. During her tenure as an undergraduate student, she was awarded best academic student award in the Department of Electronics and Communication Engineering. Her current research interests include signal & image processing and VLSI.

Vassil S. Dimitrov is a professor at the Department of Electrical and Computer Engineering, University of Calgary, Canada, and member of the Management Board of the Centre for Information Security and Cryptography. He has vast experience in the domain of cryptography and efficient implementation of cryptographic protocols. Dr. Dimitrov has published three books, more than 100 papers in peer-reviewed journals and holds three patents. He has extensively taught courses on digital signal processing, cryptography, information theory and computational complexity. Since 1997 he is a member of the New York Academy of Sciences. Prior to his professorship at the University of Calgary, he held academic position at the University of Windsor, Canada and Helsinki University of Technology, Finland. He is doing extensive consulting work in the domains of cryptography, big data analysis, and high performance computing.

Arjuna Madanayake (M'03) is an Associate Professor at the Department of Electrical and Computer Engineering at the University of Akron. He completed both M.Sc. (2004) and PhD (2008) Degrees, in Electrical Engineering, from the University of Calgary, Canada. Dr. Madanayake obtained a BSc in Electronic and Telecommunication Engineering (with First Class Honors) from the University of Moratuwa in Sri Lanka, in 2002. His research interests include multidimensional signal processing, analog/digital and mixed-signal electronics, FPGA systems, and VLSI for fast algorithms.

Renato J. Cintra (SM'2010) received the B.Sc., M.Sc., and D.Sc. degrees in electrical engineering from the Universidade Federal de Pernambuco (UFPE), Recife, Brazil, in 1999, 2001, and 2005, respectively. He is an associate professor at the Department of Statistics, UFPE. He was a visiting researcher at the INSA, Lyon, France, during 2015 and currently is a visiting professor at the University of Calgary, Canada. He serves as an Associate Editor for the IEEE Geoscience and Remote Sensing Letters; the Circuits, Systems, and Signal Processing journal; the IET Circuits, Devices & Systems; and the Journal of Communication and Information Systems. His long-term topics of research include theory and methods for digital signal and image processing, numerical analysis, and applied mathematics.

Arnaud Tisserand , PhD 1997, is senior researcher at CNRS (French National Center for Scientific Research) in computer science in Lab-STICC laboratory. His research interests include computer arithmetic, computer architecture, digital security, VLSI and FPGA design, design automation, low-power design and applications in applied cryptography, scientific computing, digital signal processing. He is Associate Editor of the IEEE Transactions on Computers and senior member of the IEEE (SSCS, CAS).