VIDEO PROCESSING USING MULTIPLIERLESS 2D-DCT WITH ALGEBRAIC INTEGERS AND MR-DCT

A Thesis

Presented to

The Graduate Faculty of The University of Akron

In Partial Fulfillment

of the Requirements for the Degree

Master of Science

Sushmabhargavi Nimmalapalli

December 2018

VIDEO PROCESSING USING MULTIPLIERLESS 2D-DCT WITH ALGEBRAIC INTEGERS AND MR-DCT

Sushmabhargavi Nimmalapalli

Thesis

Approved:

Advisor
Dr. Arjuna Madanayake

Co-Advisor
Dr. Subramaniya Hariharan

Faculty Reader
Dr. Kye-Shin Lee

Department Chair
Dr. Robert Veillette

Accepted:

Dean of the College
Dr. Donald P. Visco Jr.

Dean of the Graduate School
Dr. Chand K Midha

Date

ABSTRACT

Discrete cosine transform (DCT) has various applications in the field of image processing related to image compression. Image compression aims at reducing redundancy in image data such that only a minimal number of samples are stored or transmitted. Here, the purpose of DCT and its role in image compression is studied and explained.

The ultimate purpose of this research is to develop an $8 \times 8 \times 8$ 3D-DCT architecture by applying the methods of two-dimensional (2D) 8×8 DCT and MR-DCT. The 2D 8×8 DCT algorithm is based on the Loeffler one-dimensional (1D) DCT; it operates with exact computation and is an error-free arithmetic system up to the final reconstruction step (FRS). The next stage is the modified round DCT, which is a low-complexity DCT approximation that requires only 14 additions. A digital architecture is proposed for the complete system and is implemented on a field-programmable gate array (FPGA) platform for on-chip verification.

ACKNOWLEDGMENTS

First and foremost, I would sincerely like to thank my advisor, Dr. Arjuna Madanayake, for always being there to guide, motivate, share ideas and encourage me throughout my Master's program. This journey would not have been successful without his support.

I would like to thank my committee members, Dr. Subramaniya Hariharan and Dr. Kye-Shin Lee, for their contribution to this project, their advice, and for taking time out of their busy schedules to attend my defense.

I would like to thank Dr. Renato Cintra for his contribution to the thesis in developing the algorithms for 2D-DCT, and MR-DCT. I would also like to thank Dr. Diego Coelho, Dr. Vassil Dimitrov, and Dr. Tisserand for their contribution to the thesis in developing the algorithms for 2D-DCT.

It will be incomplete if I do not mention my brothers and sisters at Akron and my friends who have helped me throughout my Master's program by providing motivation and advice.

Finally, I take this opportunity to dedicate this thesis to my father Venkata Sudhaker Nimmalapalli, my mother Sreematha Anapothula, and my brother Sunny Rahul Nimmalapalli. Without them, none of this would have been possible.

No.	Team members	Work Contributed
1	Dr.Diego Coelho	Major contribution in deriving the algorithms for Classes and FRS. Huge effort put in, to execute and publish the paper on 8×8 2D-DCT.
2	Dr.Renato Cintra	Great effort and support provided in developing the algorithms using the method of AI and Loeffler factorization. MR-DCT algorithm developer which helped as 1D-DCT.
3	Dr.Arjuna Madanayake	Advisor/Reviewer. Immense support provided during the entire research work.
4	Dr.Vassil Dimitrov	Advisor to Dr.Diego. Co-author of 8 x 8 2D-DCT.
5	Dr.Tisserand	Reviewer and advisor for the work. Provided with immense support for deriving FRS algorithms.
6	Sushmabhargavi Nimmalapalli	Implemented algorithms developed in Dr.Diego and Dr.Cintra on Simulink. Implemented hardware block for 2D-DCT using ML-605 kit and calculated the frequency and slice distributions. Generated reports for time, power and area on ASIC. Implemented 3D-DCT using 2D-DCT and MRDCT.

Special Acknowledgment

I would like to acknowledge the National Science Foundation (NSF) for their financial support of this research.

TABLE OF CONTENTS

Page

LIST OF TABLES viii

LIST OF FIGURES x

CHAPTER

I. INTRODUCTION TO DIGITAL IMAGE AND VIDEO PROCESSING 1

1.1 Digital Images 2

1.2 Image processing using DCT 4

1.3 Video processing using DCT 5

1.4 Contributions of the thesis 5

1.5 Thesis outline 6

II. REVIEW ON THE DISCRETE COSINE TRANSFORM 8

2.1 Grayscale image 8

2.2 Colored image 9

2.3 Representation of a Digital image 10

2.4 Introduction to DCT 11

III. 2D-DCT IMPLEMENTATION USING MULTIPLIERLESS ARCHITECTURE 19

3.1 Introduction to Algebraic Integers 20

3.2 The Algebraic Integer Representation 22

3.3 The 2D DCT and the AI basis representation 27

3.4 Final Reconstruction Step 38

IV. 3D-DCT AND MONOCHROME VIDEO 50

4.1 Introduction to MR-DCT 50

4.2 Implementation of 3D-DCT 52

V. DIGITAL FPGA IMPLEMENTATION OF THE ARCHITECTURE 56

5.1 Designing of algorithms for classes on simulink. 56

5.2 Design of algorithms on Simulink for the final reconstruction step 56

5.3 Design and Implementation of the 2-D architecture using XILINX ML605 . 60

5.4 Design and Implementation of the 3-D DCT architecture 68

VI. CONCLUSIONS AND FUTURE WORK 70

6.1 Conclusions 70

6.2 Future Work 70

BIBLIOGRAPHY 72

LIST OF TABLES

Table Page

2.1 Different Image Formats. 10

3.1 Quantities required by the Loeffler algorithm for an 8-point DCT and their respective products by an arbitrary algebraic integer 24

3.2 AI representation classification. 28

3.3 Comparison for fast algorithms for the computation of 2D 8-point DCT with algebraic integer theory 37

3.4 Fast algorithms for FRS for Dyadic Approximation for an 11-bit word length and its arithmetic cost for Class A, and Class B. 40

3.5 Fast algorithms for FRS for Dyadic Approximation for an 11-bit word length and its arithmetic cost for Class C, Class D, and Class E. 41

3.6 Scale factors and maximum/minimum relative errors 47

3.7 Fast algorithms for FRS for Expansion Factor Method for an 11-bit word length ($\alpha^* = 1849.39$) and its arithmetic cost for Class A, Class B, and Class C. 48

3.8 Fast algorithms for FRS for Expansion Factor Method for an 11-bit word length ($\alpha^* = 1849.39$) and its arithmetic cost for Class D, and Class E. 49

4.1 Comparison of MR-DCT with other proposed DCTs 54

5.1 Comparison of FPGA implementation metrics. 65

5.2 Comparison of ASIC implementation metrics. 67

5.3 Measurements recorded for 3D-DCT using ML605 kit 69

5.4 Measurements recorded for 3D-DCT using ASIC. 69

LIST OF FIGURES

Figure Page

2.1 Figures representing monochrome and color (RGB) images. 9

3.1 Loeffler algorithm for the 8-point DCT computation, where dashed lines represent product by -1. 25

3.2 The 8-point DCT algorithm for a real quantized input sequence. 26

3.3 2D representation of the input coefficient class: (a) before the transformation, (b) after the application of the 1D DCT over its columns, and (c) after the application of the 2D DCT. 29

3.4 The 8-point DCT algorithm for an input sequence in the AI representation belonging to Class B. 32

3.5 The 8-point DCT algorithm for an input sequence in the AI representation belonging to Class C. 35

3.6 The 8-point 2D DCT AI-based fast algorithm. 36

3.7 Graphical representation of the entire 2D DCT parallel architecture using AI representation including FRS. The blocks DCT_A, DCT_B, and DCT_C represent the 1D DCT in the AI-based representation for inputs in classes A, B, and C, respectively, and are implemented by the algorithms in the Figs. 3.2, 3.4, and 3.5. 37

3.8 FRS block using dyadic approximation for (a) Class B, (b) Class C, (c) Class D, and (d) Class E for an 11-bit wordlength. 42

3.9 Pictorial representation of 2D-DCT 46

4.1 Signal flow graph for MR-DCT. 51

4.2 Overview of 3-D DCT 53

4.3 3D-DCT output of 8×8 input of first frame representing each pixel and its coefficient value with its corresponding color 55

5.1 Figure representing a Class A algorithm on Simulink. 57

5.2 Figure representing a Class B algorithm on Simulink. 58

5.3 Figure representing a Class C algorithm on Simulink. 59

5.4 Figures representing algorithms on Simulink for the final reconstruction step (FRS) using a dyadic approximation method. 61

5.5 Simulink design of 2D-DCT with final reconstruction step (FRS) 62

5.6 2D-DCT hardware block generated on FPGA 62

5.7 DCT outputs of Matlab using matlab ”dct2” command 63

5.8 DCT outputs of hardware designed and generated on FPGA 63

5.9 Comparison of matlab and hardware generated outputs for image from real-time 64

5.10 Figure representing MR-DCT algorithm on Simulink. 68

CHAPTER I

INTRODUCTION TO DIGITAL IMAGE AND VIDEO PROCESSING

Due to the increasing requirements for transmission of images via computers and mobile environments, the research in the field of image compression has increased significantly. Image compression plays a crucial role in digital image processing, as it is very important for the efficient transmission and storage of images [1].

There is no general agreement among researchers regarding where image processing stops and areas such as image analysis and computer vision begin. Sometimes a distinction is made by defining image processing as a discipline in which both the input and the output of a process are images. This is a limiting and somewhat artificial boundary. The area of image analysis (image understanding) lies between the areas of image processing and computer vision.

There are no clear-cut boundaries in the continuum from image processing at one end to complete vision at the other. However, one useful paradigm is to consider three types of computerized processes in this continuum: low-, mid-, and high-level processes. Low-level processing involves primitive operations such as image processing to reduce noise, enhance contrast, and sharpen images. A low-level process is characterized by the fact that both its inputs and outputs are images. Mid-level processing of images involves tasks such as segmentation, description of an object to reduce it to a form suitable for computer

processing, and classification of individual objects. A mid-level process is characterized by the fact that its inputs generally are images, but its outputs are attributes extracted from those images. Finally, higher-level processing involves making sense of an ensemble of recognized objects, by performing image analysis and, at the far end of the continuum, by performing the cognitive functions normally associated with human vision [2].

1.1 Digital Images

A digital image is composed of a finite number of elements, each of which has a particular location and value. Thus a digital image is represented by a matrix of values, where each value is a function of the information surrounding the corresponding point in the image. A single element in an image is called a *picture element* or *pixel*. In a color system, a pixel includes information for all color components. However, unlike humans, who are limited to the visual band of the electromagnetic (EM) spectrum, imaging machines cover almost the entire EM spectrum, ranging from gamma waves to radio waves. They can operate also on images generated by sources that humans are not accustomed to associating with images.

The digital video is represented in the equivalent manner by a three-dimensional (3D) matrix set of values, where frames of video represent the third dimension in the set of values. Digital video and image processing are characterized by the need for extensive experimental work to establish the viability of proposed solutions to a given problem. An important characteristic underlying the design of image processing systems is the significant level of testing and experimentation that normally is required before arriving at an

acceptable solution. This characteristic implies that the ability to formulate approaches and quickly prototype candidate solutions generally plays a major role in reducing the cost and time required to arrive at a viable system implementation.

Signals captured from the physical world are translated into digital form by process called *digitization*. Digitization involves two processes: sampling and quantization, which can be conducted in any order. When an image is sampled, two-dimensional (2D) space is partitioned into small, discrete regions. Quantization assigns an integer to the amplitude of the signal in each interval or region [3].

Some of the first applications of digital video and image processing were to improve the quality of the captured images—but as the power of computers grew, so did the number of applications where video and image processing could make a difference. Today, video and image processing are used in many diverse applications, such as astronomy (to enhance the quality of astronomical images), medicine (to measure and understand some parameters of the human body, such as blood flow in broken veins), image compression (to reduce the memory requirement when storing an image), sports (to capture the motion of an athlete in order to understand and improve the performance), rehabilitation (to assess the locomotion abilities of a patient), motion pictures (to capture the motion of actors in order to produce special effects based on graphics), surveillance (detect and track individuals and vehicles), manufacturing (to assess the quality of products), control of robots (to detect an object so that a robot can grasp it and pick it up), television production (mixing graphics and live video, e.g., a weather forecast), biometrics (to measure some unique parameters of a person), and photo editing (improving the quality or adding effects to photographs).

The different flavors of video and image processing are often grouped into the general categories listed below. There is no unique definition for each of the different categories—and, to make matters worse, they also overlap significantly. Here is one set of definitions: *video and image compression* is probably the most well defined category and comprises various methods used for compressing video and image data, while *image manipulation* comprises methods (for example, rotating or scaling an image, or improving the quality by changing the contrast).

1.2 Image processing using DCT

Image processing originates from the more general field of signal processing and covers methods used to segment the object of interest [4]. Segmentation here refers to methods which in some way enhance the object while suppressing the rest of the image (for example the edges). A discrete cosine transform (DCT) expresses a finite sequence of data points in terms of a sum of cosine functions oscillating at different frequencies. DCTs are important to numerous applications in science and engineering, from audio to images (in JPEG or MPEG). The use of cosine functions is more critical in compression than the use of sine functions, since fewer cosine functions create the approximated signal as needed. The DCT, and in particular the DCT-II, which is often used in signal and image processing, in particular the DCT which is used in JPEG image compression. For analysis of two-dimensional (2D) signals such as images, we need a 2D version of the DCT. Since the 2D DCT can be computed by applying one dimensional (1D) transforms separately to the rows and columns, we say that the 2D DCT is separable in the two dimensions [5].

1.3 Video processing using DCT

Video processing covers most of the image processing methods. Here the goal is to analyze the image with the purpose of first finding objects of interest and then extracting some parameters of the position and size of these objects. Digital video processing is the best way to support video communications, and video processing using DCT has proven to be more efficient than spatial domain. One reason for considering DCT is that data is arranged in a block-by-block fashion in this system, where the inverse DCT can be obtained easily. Video compression in different formats is possible using DCT, as the data is divided into three segments. Once the 2D-DCT is applied separately for each image to create a video, the final 1D-DCT is applied to the images in a row- and column-wise fashion for the final transformation. Orthogonal transforms are tested for 1D-DCT to be applied for the frames.

1.4 Contributions of the thesis

The work done can be summarized as follows:

1. A massively parallel 2D-DCT architecture is proposed where the design process begins with algorithms that are designed without multipliers. The DCT's are subjected to different levels of pipelining to improve speed performance. Design simulations are carried out in Matlab and Simulink, and the results are verified with the 2D-DCT command outputs on Matlab.

2. The successful working of this 2D-DCT block is used to generate a digital three-dimensional (3D) DCT architecture with the addition of 1D-DCT block, for applying

to a video in real time. The architecture of the DCT has less complexity compared to other DCT algorithms proposed so far, and is more efficient in terms of area, time and power consumption.

3. An $8 \times 8 \times 8$ 3D-DCT architecture is constructed that provides a scope for future work in video compression. The performance characteristics are calculated using using field-programmable gate array (FPGA) and application-specific integrated circuit (ASIC) analysis.

1.5 Thesis outline

The remainder of the thesis is organized as follows:

Chapter 2 provides a comprehensive review of the digital image and video processing concepts including the analysis of 2D-DCT and 3D-DCT, and their architecture and methods of implementation. Later in this chapter, we discuss the different algorithms implemented for DCT and learn about the best methods implemented to date.

Chapter 3 begins by discussing the methods used in this study for the implementation of the 2D-DCT and 3D-DCT architectures. Next, a new approach to DCT using algebraic integer (AI) representation is discussed, including its application in the construction of the 2D-DCT architecture. The advantages of this approach and methodology are revealed through its use in a DCT representation. The components of the DCT representation (referred to here as classes) and their algorithms are studied, using only adders/subtractors and shifters. The proposed method is thus considered to be a multiplier-less architecture,

which later helps us to study the benefits of the architecture without considering multipliers. Next, the algorithms associated with the architecture are explained. The results are compared with previous studies and other proposed research that best describes the work on digital image processing with a 2D-DCT architecture. Finally, different 1D-DCT methods are studied in order to determine which works better with the proposed 2D-DCT for the construction of 3D-DCT architecture.

Chapter 4 explains the concept of a modified round discrete cosine transform (MR-DCT) identifying its benefits and the idea of developing the 3D-DCT architecture from a 2D-DCT and subjecting it to video processing. This chapter also presents the modeling of 3D-DCT block for video processing using 2D-DCT and MR-DCT.

Chapter 5 begins with the design and implementation of the 2D-DCT architecture on FPGAs. Initially, the design with the mentioned algorithms is created and is hardware co-simulated in the ML605 FPGA platform. This chapter provides a brief introduction to the FPGA, explains the design and implementation of the 2D-DCT architecture, and explains how the 3D-DCT architecture is implemented.

Chapter 6 is the final chapter. It describes possible future investigations based on the work completed in this study.

CHAPTER II

REVIEW ON THE DISCRETE COSINE TRANSFORM

The discrete cosine transform (DCT) is an essential tool in digital signal processing. In recent years, the signal processing literature has been populated with low-complexity methods for the efficient computation of an eight-point DCT [6]. One of the many techniques in the category of image processing is image compression. Image compression aims at reducing redundancy in image data such that only a minimal number of samples is stored or transmitted. First, let us consider the purpose of DCT and its role in image compression. For processing 1-D or 2-D signals, a common method is to divide the signal into "frames" and then apply an invertible transform to each frame that compresses the information into a few coefficients [1].

An image is represented as a 2D function $f(x,y)$ where x and y are spatial coordinates and the amplitude of 'f' at any pair of coordinates (x,y) is referred to as the *intensity* of the image at that point.

2.1 Grayscale image

The term *monochrome image* refers to two dimensional light intensity function $f(x,y)$, where x and y denote spatial coordinates, and the value of f at any point (x,y) is proportional to the brightness (or gray level) of the image at that point. A grayscale image, also

(a) Grayscale Image (b) RGB Image

Figure 2.1: Figures representing monochrome and color (RGB) images.

referred to as an intensity image, is represented by a 2D matrix, where the elements of the matrix may take integer values ranging from 0 to 255 (for uint8 type).

2.2 Colored image

An image with red, green, and blue (RGB) values, sometimes referred to as a *true-color image*, is represented as an $m \times n \times 3$ data array that defines red, green, and blue color components for each individual pixel. The color of each pixel is determined by the combination of the red, green, and blue intensities stored in each color plane at the pixel's location. An RGB image may be viewed as "stack" of three grayscale images that used as the red, green and blue inputs of a color monitor. The number of bits used to represent the pixel values of the component images determines the bit depth of the RGB image. For example, if each component image is an 8-bit image, the corresponding RGB image is said to be 24 bits deep.

Table 2.1: Different Image Formats.

Format Name	Description	Recognized Extension
TIFF	Tagged Image File Format	.tif, .ti
JPEG	Joint Photograph Experts Group	.jpg, .jpeg
GIF	Graphics Interchange Format	.gif
MPEG	Moving Picture Experts Group	.mp, .mp3
BMP	Windows Bitmap	.bmp
PNG	Portable Network Graphics	.png
XWD	X Window Dump	.xwd

2.3 Representation of a Digital image

An image may be continuous with respect to the x and y coordinates and also continuous in amplitude. Converting such an image to digital form requires the coordinates as well as the amplitude to be digitized. Digitizing the coordinate's values is called *sampling* and digitizing the amplitude values is called *quantization*. A Digital image is an image $f(x,y)$ that has been discretized both in spatial coordinates and brightness. A digital image can be considered as a matrix whose row and column indices identify a point in the image, and the corresponding matrix element value identifies the gray level at that point. When a digital image is represented in the form of matrix, f is considered to be an image with m rows and

n columns, and the matrix is given by

$$\begin{bmatrix} f(1,1) & f(1,2) & f(1,3) & ..... & f(1,N) \\ f(2,1) & f(2,2) & f(2,3) & ..... & f(2,N) \\ f(3,1) & f(3,2) & f(2,3) & ..... & f(3,N) \\ . & . & . & ..... & . \\ . & . & . & ..... & . \\ f(M,1) & f(M,2) & . & ..... & f(M,N) \end{bmatrix}$$

2.4 Introduction to DCT

The standard, classic, and well known 2D DCT is largely used in the MPEG or JPEG world for very efficient image compression. Formally, the discrete cosine transform is a linear, invertible function f: $\mathbf{R}^N \rightarrow \mathbf{R}^N$ (where $\mathbf{R}^N$ denotes the set of real numbers), or equivalently an invertible $N \times N$ square matrix. There are several variants of the DCT with slightly modified definitions. The N real numbers x_0, ..., x_{N-1} are transformed into the N real numbers X_0,..., X_{N-1} according to one of the formulas presented in the following subsections.

2.4.1 Overview of DCT

Here is a brief overview of how DCT is performed:

1. How to perform a 2D DCT ? It is a transform that is both forward and inverse. We can perform manual calculations for the small size matrices using inner product notation. In matlab the commands $dct2$ and $idct2$ are used.

2. How to quantize DCT coefficients? DCT coefficients are quantized using various step sizes for different DCT coefficients based on visual sensitivity to different frequencies.

3. What is a quantization matrix ? A quantization matrix specifies the default quantization step size for each coefficient. The matrix can be scaled using a user chosen parameter (QP) to obtain different trade-offs between quality and size.

2.4.2 DCT-I

The basic equation for a DCT is expressed as follows:

$$X_k = \frac{1}{2}(x_0 + (-1)^k x_{N-1}) + \sum_{n=1}^{N-2} x_n \cos\left[\frac{\pi}{N-1} nk\right] \qquad k = 0, \ldots, N-1 \qquad (2.1)$$

Sometimes this equation is further modified by multiplying the x_0 and x_{N-1} terms by $\sqrt{2}$, and correspondingly by multiplying the X_0 and X_{N-1} terms by $\frac{1}{\sqrt{2}}$. This makes the DCT-I matrix orthogonal, if one further multiplies by an overall scale factor of $\sqrt{\frac{2}{N-1}}$, but it breaks the direct correspondence with a real-even DFT.

The DCT-I is exactly equivalent (up to an overall scale factor of 2), to a DFT of $2N-2$ real numbers with even symmetry. For example, a DCT-I of N=5 real numbers

(*a,b,c,d,* and *e*) is exactly equivalent to a DFT of eight real numbers (*a,b,c,d,e,d,c* and *b*; even symmetry), divided by two. (In contrast, DCT types II-IV involve a half-sample shift in the equivalent DFT.)

Note, however, that the DCT-I is not defined for N less than 2. (All other DCT types are defined for any positive N.)

Thus, the DCT-I corresponds to the boundary conditions: x_n is even around $n = 0$ and even around $n = N - 1$; similarly for X_k.

2.4.3 DCT-II

The DCT-II is probably the most commonly used form, and it is often simply referred to as "the DCT" [7, 8].

$$X_k = \sum_{n=0}^{N-1} x_n \cos\left[\frac{\pi}{N}\left(n+\frac{1}{2}\right)k\right] \qquad k = 0,\ldots,N-1. \tag{2.2}$$

This transform is exactly equivalent (up to an overall scale factor of 2) to a DFT of $4N$ real inputs of even symmetry where the even-indexed elements are zero. That is, it is half of the DFT of the $4N$ inputs y_n, where $y_{2n} = 0$, $y_{2n+1} = x_n$ for $0 \le n < N$, $y_{2N} = 0$, and $y_{4N-n} = y_n$ for $0 < n < 2N$. DCT II transformation is also possible using a 2N signal followed by a multiplication by a half shift.

Sometimes the X_0 term is further multiplied by $1/\sqrt{2}$, and the resulting matrix is multiplied by an overall scale factor of $\sqrt{\frac{2}{N}}$ (see below for the corresponding change in DCT-III). This makes the DCT-II matrix orthogonal, but it breaks the direct correspondence with a real-even DFT of half-shifted input. This is the normalization used by Matlab, for

example. In many applications, such as JPEG, the scaling is arbitrary because the scale factors can be combined with a subsequent computational step (e.g., the quantization step in JPEG [9]), and a scaling can be chosen that allows the DCT to be computed with fewer multiplications [10] [11].

The DCT-II implies the boundary conditions: x_n is even around $n = -1/2$ and even around $n = \frac{N-1}{2}$; X_k is even around $k = 0$ and odd around $k = N$.

The 2D DCT-II of $N \times N$ blocks are computed, and the results are quantized and entropy coded. In this case, N is typically 8 and the DCT-II formula is applied to each row and column of the block. The result is an 8×8 transform coefficient array in which the $(0,0)$ element (the one at top left) is the DC (zero-frequency) component, and entries with increasing vertical and horizontal index values represent higher vertical and horizontal spatial frequencies.

2.4.4 Multi-Dimensional DCT

Multidimensional DCTs (MD DCTs) have several applications, and 3-D DCT-II has several new applications like hyperspectral imaging coding systems, variable temporal length 3-D DCT coding, video coding algorithms, adaptive video coding and 3-D compression. Due to advances in hardware and software and the introduction of several fast algorithms, the necessity of using M-D DCTs is rapidly increasing. DCT-IV has gained popularity for its applications in fast implementation of real-valued polyphase filtering banks, lapped orthogonal transforms, and cosine-modulated wavelet bases.

2.4.5 M-D DCT-II

In MD DCT-II, for example, a two-dimensional DCT-II of an image or a matrix is simply the one-dimensional DCT-II, from above, performed along the rows and then along the columns (or vice versa). That is, the 2D DCT-II is given by the formula (omitting normalization and other scale factors, as above):

$$X_{k_1,k_2} = \sum_{n_1=0}^{N_1-1} \left(\sum_{n_2=0}^{N_2-1} x_{n_1,n_2} \cos\left[\frac{\pi}{N_2} \left(n_2 + \frac{1}{2} \right) k_2 \right] \right) \cos\left[\frac{\pi}{N_1} \left(n_1 + \frac{1}{2} \right) k_1 \right]$$

$$= \sum_{n_1=0}^{N_1-1} \sum_{n_2=0}^{N_2-1} x_{n_1,n_2} \cos\left[\frac{\pi}{N_1} \left(n_1 + \frac{1}{2} \right) k_1 \right] \cos\left[\frac{\pi}{N_2} \left(n_2 + \frac{1}{2} \right) k_2 \right]. \tag{2.3}$$

The inverse of a multi-dimensional DCT is just a separable product of the inverse(s) of the corresponding one-dimensional DCT(s), e.g. the one-dimensional inverses applied along one dimension at a time in a row-column algorithm. The 3-D DCT-II is only the extension of the 2-D DCT-II in three-dimensional space and can be calculated by the formula

$$X_{k_1,k_2,k_3} = \sum_{n_1=0}^{N_1-1} \sum_{n_2=0}^{N_2-1} \sum_{n_3=0}^{N_3-1} x_{n_1,n_2,n_3} \cos\left[\frac{\pi}{N_1} \left(n_1 + \frac{1}{2} \right) k_1 \right] \cos\left[\frac{\pi}{N_2} \left(n_2 + \frac{1}{2} \right) k_2 \right]$$

$$\cos\left[\frac{\pi}{N_3} \left(n_3 + \frac{1}{2} \right) k_3 \right], \quad \forall k_i = 0, 1, 2, \ldots, N_i - 1. \tag{2.4}$$

The inverse of 3-D DCT-II is 3-D DCT-III and can be computed from the formula given by

$$x_{n_1,n_2,n_3} = \sum_{k_1=0}^{N_1-1} \sum_{k_2=0}^{N_2-1} \sum_{k_3=0}^{N_3-1} X_{k_1,k_2,k_3} \cos\left[\frac{\pi}{N_1} \left(n_1 + \frac{1}{2} \right) k_1 \right] \cos\left[\frac{\pi}{N_2} \left(n_2 + \frac{1}{2} \right) k_2 \right]$$

$$\cos\left[\frac{\pi}{N_3}\left(n_3+\frac{1}{2}\right)k_3\right], \quad \forall n_i = 0,1,2,\ldots,N_i-1. \tag{2.5}$$

Technically, computing a two-, three- (or multi-) dimensional DCT by sequences of one-dimensional DCTs along each dimension is a row-column algorithm. With multi-dimensional fast Fourier transform (FFT) algorithms, however, there exist other methods to compute the same DCT while performing the computations in a different order (i.e., interleaving/combining the algorithms for the different dimensions). Owing to the rapid growth in applications based on the 3D DCT, several fast algorithms are developed for the computation of 3D DCT-II. Vector-radix algorithms are applied for computing the MD-DCT to reduce the computational complexity and to increase the computational speed. To compute 3D DCT-II efficiently, a fast algorithm, vector-radix decimation in frequency (VR DIF) algorithm was developed.

2.4.6 Why use 8×8 blocks?

The DCT treats the block as if it were periodic and thus must reconstruct the resulting jump at the boundaries. When using 64×64 blocks, there will most likely be a huge jump at the boundaries, and a large number of high-frequency components will be needed to reconstruct the block to a satisfactory precision. The use of ”8” rather than ”64” results from a trade-off that can not be theoretically optimized; however, it seems to work well for typical images, where the variation across 8×8 blocks is typically small enough that blocking artifacts can be avoided without having to encode too much high-frequency information.

In addition, since 8 is a power of 2, even if the optimal trade-off from an information content standpoint had been, for example, 10, one might still have chosen 8 because the transform is much simpler and faster to perform.

Compression is always a trade-off. While sharper images can always be obtained by keeping more of the information, sharp images can also be obtained with 4×4, 8×8 or 64×64 blocks simply by keeping the entire high-frequency information. Experience shows that in 8×8 blocks, much of the information can be dropped without creating unacceptable blocking artifacts. Certainly for 4×4 blocks the boundary jump would be even less, but there would also be less opportunity for compression. Consider an 8×8 block made up of four 4×4 blocks: If you transform each of the 4×4 blocks separately, their averages (zero-frequency components) must be stored with the same (high) precision for all four of them. If they are transformed together as an 8×8 block, instead of four averages, one average and three oscillating components will be obtained that can be stored with lower precision. This cannot be expressed in a single precise formula; one has to look at how data in real-world images are actually distributed and then make the required trade-offs.

2.4.7 Summary of Advantages of DCT

In summary, the use of DCT for image processing has the following advantages:

1. DCT is advantageous for image coding because its feature is based on a real transform and is easier to perform than DFT.

2. DCT can ignore most high-frequency coefficients, since they are nearly zero.

3. Different coefficients can be quantized with different levels of accuracy based on human sensitivity.

CHAPTER III

2D-DCT IMPLEMENTATION USING MULTIPLIERLESS ARCHITECTURE

In this chapter, algebraic integers(AIs) and their role in the construction of algorithms for 2D-DCT is discussed. The initial focus is on the application of 1D-DCT in signal processing.

The discrete cosine transform (DCT) is a pivotal tool for solving signal processing problems [12, 13] such as image compression [14], noise reduction [15], and watermarking methods [16, 17]. 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 [13]. This is relevant because images often follow such a model [13].The 8-point DCT of type II, hereafter referred only as DCT, has been employed in different image and video compression standards [18], including JPEG [19], MPEG-1 [20], H.264 [21], and HEVC [22].

Due to such wide acceptance, several fast algorithms were proposed for the 8-point DCT [13]. A particularly relevant fast algorithm is the one proposed by Loeffler *et al.* described in [23], which is capable of computing the 8-point DCT with the minimum possible number of multiplications [23–25].Because of this, the Loeffler factorization for the 8-point DCT is considered to be a reference method for comparing DCT algorithms.

3.1 Introduction to Algebraic Integers

The theory of algebraic integers was first introduced in the context of digital signal processing in 1985 by Cozzens and Finkelstein [26, 27], who aimed to compute 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 [27], 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 the algebraic integer representation into a typical 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 [28–31]. These architectures are able to compute the 1D DCT without multipliers within an error-free structure.

The typical computation of a 2D DCT is accomplished by column- and row-wise calls of the 1D DCT. However, simply computing the 1D DCT by means of an AI-based algorithm does not result in a *bona fide* AI-based 2D DCT computation. Indeed, from the standpoint of a 2D DCT point-of-view, the FRS blocks from an 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, as it uses 1D DCT its FRS. An error-free computation of 2D DCT was proposed for the Arai

algorithm [32]. Therefore, the output of the algorithm used both in [33] and [34] is a non-uniform scaled version of the 2D DCT spectrum.

The error-free characteristic of the methods proposed in [26–31, 33, 34] is a by-product of the algebraic integer encoding, possibly not the most important. The additional advantages of AI-based fast algorithms are the (i) parallelization and (ii) low latency due to the accumulation of multiplicative complexity at the FRS.

This chapter introduces a 2D DCT (type II) 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 [4]. The fully 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 that is capable of computing the 2D DCT with these particular characteristics makes it attractive to a designer who needs to consider several metrics when deciding which particular method he/she is using for an application [12]. The adoption of this building block will be determined by the specific application requirements and constraints.

3.2 The Algebraic Integer Representation

3.2.1 Review of 8-point DCT AI Basis

The *AI Basis*

The 8-point 1D DCT is a linear orthogonal transformation given by [13, 14]:

$$X_k = \frac{1}{2}\sum_{n=0}^{7} \beta_k x_n \cos\left[\frac{\pi(2n+1)k}{16}\right], \quad k = 0,1,\ldots,7, \tag{3.1}$$

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

In [4], 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 a linear combination of the elements of $\mathbf{Z}$ is suitable for the computation of the 8-point DCT, due to the fact that the 8-point DCT requires the quantities $\cos[\pi k(2n+1)/16]$, $n,k = 0,1,\ldots,7$ [13]. 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.

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 [33]:

$$f_{\text{dec}}(\mathbf{x};\boldsymbol{\zeta}) = \mathbf{x}^{\top} \cdot \boldsymbol{\zeta} = a_0 + \sum_{k=1}^{7} a_k \cdot c_k = \hat{x}. \tag{3.2}$$

In [4], 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 $\varepsilon > 0$. The authors have also pointed out that in typical applications, such as in the context of image compression, the input data are real, discrete, and quantized [35] in the form of an integer [13].

In such conditions, a real quantized input m, the AI-encoded data can be trivially obtained according to $f_{\text{enc}}(m;\boldsymbol{\zeta}) = \begin{bmatrix} m & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}^{\top}$.

Arithmetic Operations

AI-based addition and multiplication operations over the considered basis were defined in [4] as 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 obeys 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 3.1. 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.

Table 3.1: Quantities required by the Loeffler algorithm for an 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	$[2u_7 \quad u_6 \quad u_5-u_7 \quad u_4-u_6 \quad u_3-u_5 \quad u_2-u_4 \quad u_1-u_3 \quad u_0-u_2]^\top$

3.2.2 Loeffler 1D DCT Multiplicands

The Loeffler DCT algorithm has a four-stage signal flow graph (SFG), shown in Figure 3.1 [23], 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 Stages 2 to 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 $c_i \cdot c_k = c_{i+k} + c_{i-k}$ for any $i,k \in \mathbb{Z}$, and $c_i = -c_{16-i}$ for $i = 8,9,\ldots,16$. Therefore, the quantities required by the Loeffler DCT computation possess simple and multiplierless representations over the representation introduced in [4].

In [4], it was shown that the ring implied by the AI formalism for the 1D DCT case is over-complete; 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.

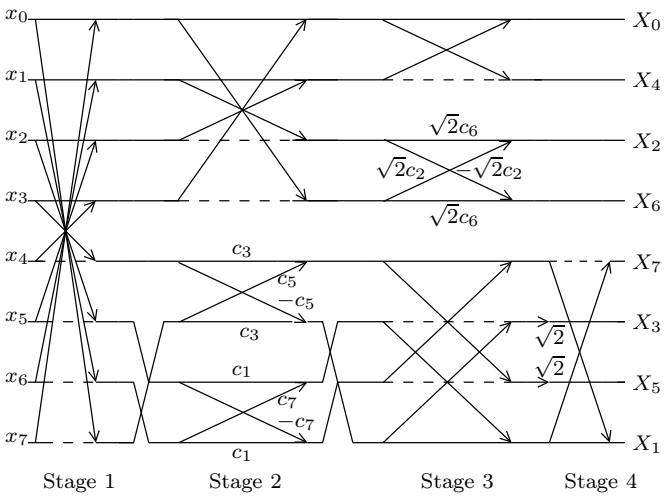


Figure 3.1: Loeffler algorithm for the 8-point DCT computation, where dashed lines represent product by -1.

3.2.3 1D AI-based Fast Algorithm

The representation proposed in [4], when applied to the 1D 8-point DCT, furnishes the Algorithm 3.2, which is multiplierless. It requires a total of 20 additions. Indeed, due to the definition in the Loeffler DCT, the algorithm output is a scaled DCT with a scaling factor of 2. If required, the output can be re-scaled by simple bit-shifting, or it can be inserted into the decoding stage. Therefore, scaling by a factor of 2 does not contribute to an increase in arithmetic complexity when performing the processing.

Input: $x_n \in \mathbb{Z}$ for $n = 0, 1, \ldots, 7$
Output: $\mathbf{X}_k \in \text{span}(\mathbf{Z})$, for $k = 0, 1, \ldots, 7$

Additions in Stage 1:

$A_0 = x_0 + x_7 \; A_4 = x_3 - x_4$

$A_1 = x_1 + x_6 \; A_5 = x_2 - x_5$

$A_2 = x_2 + x_5 \; A_6 = x_1 - x_6$

$A_3 = x_3 + x_4 \; A_7 = x_0 - x_7$

Additions in Stage 2:

$B_0 = A_0 + A_3 \; B_2 = A_1 - A_2$

$B_1 = A_1 + A_2 \; B_3 = A_0 - A_3$

Additions in Stage 3:

$C_0 = B_0 + B_1 \; C_2 = B_2 + B_3$

$C_1 = B_0 - B_1 \; C_3 = B_2 - B_3$

Additions in Stage 4:

$D_0 = -A_5 + A_6 \; D_2 = A_4 + A_7$

$D_1 = A_4 - A_7 \quad D_3 = -A_5 - A_6$

Output:

$$\mathbf{X}_0 = [2C_0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0]^\top$$

$$\mathbf{X}_1 = [0 \quad -D_3 \quad 0 \quad D_2 \quad 0 \quad -D_1 \quad 0 \quad D_0]^\top$$

$$\mathbf{X}_2 = [0 \quad 0 \quad C_2 \quad 0 \quad 0 \quad 0 \quad -C_3 \quad 0]^\top$$

$$\mathbf{X}_3 = [0 \quad -D_1 \quad 0 \quad D_3 \quad 0 \quad D_0 \quad 0 \quad D_2]^\top$$

$$\mathbf{X}_4 = [2C_1 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0]^\top$$

$$\mathbf{X}_5 = [0 \quad D_2 \quad 0 \quad -D_0 \quad 0 \quad D_3 \quad 0 \quad D_1]^\top$$

$$\mathbf{X}_6 = [0 \quad 0 \quad -C_3 \quad 0 \quad 0 \quad 0 \quad -C_2 \quad 0]^\top$$

$$\mathbf{X}_7 = [0 \quad -D_0 \quad 0 \quad -D_1 \quad 0 \quad -D_2 \quad 0 \quad -D_3]^\top$$

Figure 3.2: The 8-point DCT algorithm for a real quantized input sequence.

3.3 The 2D DCT and the AI basis representation

3.3.1 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 [13, 14]:

$$X_{l,k} = \frac{1}{4}\sum_{m=0}^{7}\sum_{n=0}^{7}\alpha_k\beta_l x_{m,n}\cos\left[\frac{\pi(2n+1)l}{16}\right]\cos\left[\frac{\pi(2m+1)k}{16}\right], \tag{3.3}$$

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 [13, 36–38], 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.

3.3.2 2D AI-based Fast Algorithm

When the 2D input array is real and quantized, several simplifications arise. These simplifications can be exploited to provide efficient (fast) algorithms for the 2D DCT over the AI basis representation proposed in [4] without a 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 Row 0 and Row 4 will always have

Table 3.2: AI representation classification.

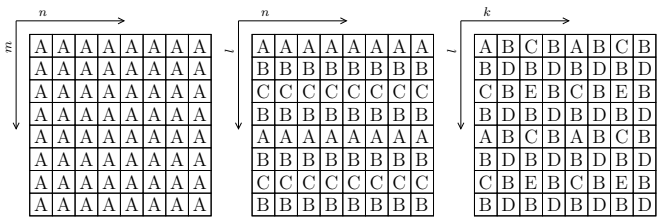
Class	AI representation
A	$\mathbf{u} = [\times \ 0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0]^\top$
B	$\mathbf{u} = [0 \ \times \ 0 \ \times \ 0 \ \times \ 0 \ \times]^\top$
C	$\mathbf{u} = [0 \ 0 \ \times \ 0 \ 0 \ 0 \ \times \ 0]^\top$
D	$\mathbf{u} = [\times \ 0 \ \times \ 0 \ \times \ 0 \ \times \ 0]^\top$
E	$\mathbf{u}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

Note: Multiplication symbols correspond to non-null coefficients.

a non-null first coefficient in its AI-based representation; (ii) the elements in Row 1, Row 3, Row 5, and Row 7 will exhibit exhibit non-null odd-index coefficients; and (iii) Row 2 and Row 6 will 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 3.2 as proposed in [4].

In view of their patterns, we categorize the AI quantities into the five classes shown in Table 3.2, where non-null coefficient locations are represented by a multiplication symbol. If we represent the two-dimensional input sequence in graphical format as in Figure 3.3(a), we obtain the configuration shown in Figure 3.3(b) after the application of 1D DCT over the columns. The letters A, B, C, D, and E represent the class to which the quantity belongs, according to the definitions of the AI representations in Table 3.2.

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



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

Figure 3.3: 2D representation of the input coefficient class: (a) before the transformation, (b) after the application of the 1D DCT over its columns, and (c) after the application of the 2D DCT.

that consider input data in Class B and Class C. For input in Class A, the DCT algorithm collapses to the method detailed in [4]. For such, we consider the multiplication rules outlined in Table 3.1. The obtained procedures are detailed in the algorithm in Fig. 3.4 (for Class B data), and in the algorithm in Fig. 3.5 (for Class C data). The algorithm in Fig. 3.4 requires 136 additions and 14 bit-shifting operations; whereas the algorithm in Fig. 3.5 demands 74 additions and 8 bit-shifting operations. The outputs of the algorithms in Figs. 3.4 and 3.5 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 3.3(c).

For a given 8×8 block $\mathbf{x}$, let $x_{m,\cdot}$ and let $x_{\cdot,n}$ denote the m th row and the n th column of $\mathbf{x}$, respectively.

Let also the operators $\mathrm{DCT_A}(\cdot)$, $\mathrm{DCT_B}(\cdot)$, and $\mathrm{DCT_C}(\cdot)$ be instantiations of algorithms in the Fig. 3.2, Fig. 3.4, and Fig. 3.5, respectively. For example, $\mathrm{DCT_A}(x_{\cdot,n})$ denotes the computation of the 1D DCT over the n th column of the block $\mathbf{x}$ whose coefficients belong to Class A in the AI-based representation.

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:

$$a_0 = \mathbf{A}_4(2) + \mathbf{A}_7(8)\ \ a_1 = \mathbf{A}_4(2) - \mathbf{A}_7(8)\ \ a_2 = \mathbf{A}_4(8) + \mathbf{A}_7(2)\ \ a_3 = \mathbf{A}_4(8) - \mathbf{A}_7(2)$$

$$a_4 = \mathbf{A}_4(4) + \mathbf{A}_7(6)\ \ a_5 = \mathbf{A}_4(4) - \mathbf{A}_7(6)\ \ a_6 = \mathbf{A}_4(6) + \mathbf{A}_7(4)\ \ a_7 = \mathbf{A}_4(6) - \mathbf{A}_7(4)$$

$$a_8 = \mathbf{A}_5(2) + \mathbf{A}_6(8)\ \ a_9 = \mathbf{A}_5(2) - \mathbf{A}_6(8)\ \ a_{10} = \mathbf{A}_5(8) + \mathbf{A}_6(2)\ a_{11} = \mathbf{A}_5(8) - \mathbf{A}_6(2)$$

$$a_{12} = \mathbf{A}_5(4) + \mathbf{A}_6(6)\ a_{13} = \mathbf{A}_5(4) - \mathbf{A}_6(6)\ a_{14} = \mathbf{A}_5(6) + \mathbf{A}_6(4)\ a_{15} = \mathbf{A}_5(6) - \mathbf{A}_6(4)$$

Stage 2 Outputs:

$$\mathbf{B}_0 = [\ \ 0\ \ \mathbf{A}_0(2) + \mathbf{A}_3(2)\ \ 0\ \ \mathbf{A}_0(4) + \mathbf{A}_3(4)\ \ 0\ \ \mathbf{A}_0(6) + \mathbf{A}_3(6)\ \ 0\ \ \mathbf{A}_0(8) + \mathbf{A}_3(8)]^\top$$

$$\mathbf{B}_1 = [\ \ 0\ \ \mathbf{A}_1(2) + \mathbf{A}_2(2)\ \ 0\ \ \mathbf{A}_1(4) + \mathbf{A}_2(4)\ \ 0\ \ \mathbf{A}_1(6) + \mathbf{A}_2(6)\ \ 0\ \ \mathbf{A}_1(8) + \mathbf{A}_2(8)]^\top$$

$$\mathbf{B}_2 = [\ \ 0\ \ \mathbf{A}_1(2) - \mathbf{A}_2(2)\ \ 0\ \ \mathbf{A}_1(4) - \mathbf{A}_2(4)\ \ 0\ \ \mathbf{A}_1(6) - \mathbf{A}_2(6)\ \ 0\ \ \mathbf{A}_1(8) - \mathbf{A}_2(8)]^\top$$

$$\mathbf{B}_3 = [\ \ 0\ \ \mathbf{A}_0(2) - \mathbf{A}_3(2)\ \ 0\ \ \mathbf{A}_0(4) - \mathbf{A}_3(4)\ \ 0\ \ \mathbf{A}_0(6) - \mathbf{A}_3(6)\ \ 0\ \ \mathbf{A}_0(8) - \mathbf{A}_3(8)]^\top$$

$$\mathbf{B}_4 = [\ \ 2a_4\ \ 0\ \ a_0 + a_6\ \ 0\ \ a_1 + a_2\ \ 0\ \ a_5 - a_3\ \ 0]^\top$$

$$\mathbf{B}_5 = [\ \ 2a_8\ \ 0\ \ a_9 + a_{12}\ \ 0\ \ a_{13} + a_{14}\ \ 0\ \ a_{15} + a_{10}\ \ 0]^\top$$

$$\mathbf{B}_6 = [\ -2a_{11}\ \ 0\ \ a_{10} - a_{15}\ \ 0\ \ -a_{13} + a_{14}\ \ 0\ \ a_{12} - a_9\ \ 0]^\top$$

$$\mathbf{B}_7 = [\ -2a_7\ \ 0\ \ -a_3 - a_5\ \ 0\ \ -a_1 + a_2\ \ 0\ \ a_6 - a_0\ \ 0]^\top$$

Class B algorithm continued...

Auxiliary additions in Stage 3:

$b_0 = \mathbf{B}_2(2) + \mathbf{B}_3(2)\; b_1 = \mathbf{B}_2(2) - \mathbf{B}_3(2)\; b_2 = \mathbf{B}_2(4) + \mathbf{B}_3(4)\; b_3 = \mathbf{B}_2(4) - \mathbf{B}_3(4)$

$b_4 = \mathbf{B}_2(6) + \mathbf{B}_3(6)\; b_5 = \mathbf{B}_2(6) - \mathbf{B}_3(6)\; b_6 = \mathbf{B}_2(8) + \mathbf{B}_3(8)\; b_7 = \mathbf{B}_2(8) - \mathbf{B}_3(8)$

$b_8 = b_0 + b_7 \qquad b_9 = b_0 - b_7 \qquad b_{10} = b_1 + b_6 \qquad b_{11} = b_1 - b_6$

$b_{12} = b_2 + b_5 \qquad b_{13} = b_2 - b_5 \qquad b_{14} = b_3 + b_4 \qquad b_{15} = b_3 - b_4$

Stage 3 Outputs:

$$\mathbf{C}_0 = [\; 0 \quad \mathbf{B}_0(2) + \mathbf{B}_1(2) \quad 0 \quad \mathbf{B}_0(4) + \mathbf{B}_1(4) \quad 0 \quad \mathbf{B}_0(6) + \mathbf{B}_1(6) \quad 0 \quad \mathbf{B}_0(8) + \mathbf{B}_1(8)]^\top$$

$$\mathbf{C}_1 = [\; 0 \quad \mathbf{B}_0(2) - \mathbf{B}_1(2) \quad 0 \quad \mathbf{B}_0(4) - \mathbf{B}_1(4) \quad 0 \quad \mathbf{B}_0(6) - \mathbf{B}_1(6) \quad 0 \quad \mathbf{B}_0(8) - B1(8)]^\top$$

$$\mathbf{C}_2 = [\; 0 \quad b_9 + b_{13} \quad 0 \quad b_8 - b_{15} \quad 0 \quad -b_{11} + b_{12} \quad 0 \quad -b_{10} + b_{14}]^\top$$

$$\mathbf{C}_3 = [\; 0 \quad -b_{10} - b_{14} \quad 0 \quad -b_{11} - b_{12} \quad 0 \quad -b_{15} - b_8 \quad 0 \quad b_{13} - b_9]^\top$$

$$\mathbf{C}_4 = [\; \mathbf{B}_4(1) + \mathbf{B}_6(1) \quad 0 \quad \mathbf{B}_4(3) + \mathbf{B}_6(3) \quad 0 \quad \mathbf{B}_4(5) + \mathbf{B}_6(5) \quad 0 \quad \mathbf{B}_4(7) + \mathbf{B}_6(7) \quad 0]^\top$$

$$\mathbf{C}_5 = [-\mathbf{B}_5(1) + \mathbf{B}_7(1) \quad 0 \quad -\mathbf{B}_5(3) + \mathbf{B}_7(3) \quad 0 \quad -\mathbf{B}_5(5) + \mathbf{B}_7(5) \quad 0 \quad -\mathbf{B}_5(7) + \mathbf{B}_7(7) \quad 0]^\top$$

$$\mathbf{C}_6 = [\; \mathbf{B}_4(1) - \mathbf{B}_6(1) \quad 0 \quad \mathbf{B}_4(3) - \mathbf{B}_6(3) \quad 0 \quad \mathbf{B}_4(5) - \mathbf{B}_6(5) \quad 0 \quad \mathbf{B}_4(7) - \mathbf{B}_6(7) \quad 0]^\top$$

$$\mathbf{C}_7 = [\; \mathbf{B}_5(1) + \mathbf{B}_7(1) \quad 0 \quad \mathbf{B}_5(3) + \mathbf{B}_7(3) \quad 0 \quad \mathbf{B}_5(5) + \mathbf{B}_7(5) \quad 0 \quad \mathbf{B}_5(7) + \mathbf{B}_7(7) \quad 0]^\top$$

Class B algorithm continued...

Output:

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

Figure 3.4: The 8-point DCT algorithm for an input sequence in the AI representation belonging to Class B.

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

$$a_0 = \mathbf{A}_4(3) + \mathbf{A}_7(7)\ a_1 = \mathbf{A}_4(7) + \mathbf{A}_7(3)\ a_2 = \mathbf{A}_4(3) - \mathbf{A}_7(7)\ a_3 = -\mathbf{A}_4(7) + \mathbf{A}_7(3)$$

$$a_4 = \mathbf{A}_5(3) + \mathbf{A}_6(7)\ a_5 = \mathbf{A}_5(3) - \mathbf{A}_6(7)\ a_6 = \mathbf{A}_5(7) + \mathbf{A}_6(3)\ a_7 = \mathbf{A}_5(7) - \mathbf{A}_6(3)$$

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:

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

Class C algorithm continued...

Stage 3 Outputs:

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

Class C algorithm continued...

Output:

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

Figure 3.5: The 8-point DCT algorithm for an input sequence in the 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 $\mathbf{x}_{m,n}$ using the algorithm in Fig. 3.2 :

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

Compute the 1D DCT over the rows of $\mathbf{x}_{m,n}$ using algorithm in the Fig. 3.2; :

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

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

Figure 3.6: The 8-point 2D DCT AI-based fast algorithm.

Let $\mathbb{Z}_{\mathrm{X}}$ be the set of 8-point integer vectors belonging to Class X, where $\mathrm{X} \in \{\mathrm{A},\mathrm{B},\mathrm{C}\}$. Thus, the complete fast algorithm for the computation of the 2D DCT using AI representation can be expressed in terms of Algorithms 3.2, 3.4, and 3.5, as shown in Algorithm 3.6.

Notice that the output are in AI-based representation.

Algorithm 3.6 along with the FRS is graphically depicted on Figure 3.7.

The computational cost of algorithm in the Fig. 3.6 can be derived by noticing that it demands 10, 4, and 2 instantiations of algorithms in the Figs. 3.2, 3.4, and 3.5, respectively. Considering the number of additions and bit-shiftings required by these three algorithms, the computation of the 2D DCT using AI-based representation requires a total of 892 additions and 92 bit-shifting operations. Counts for arithmetic operations for various algorithms are shown in Table 3.3.

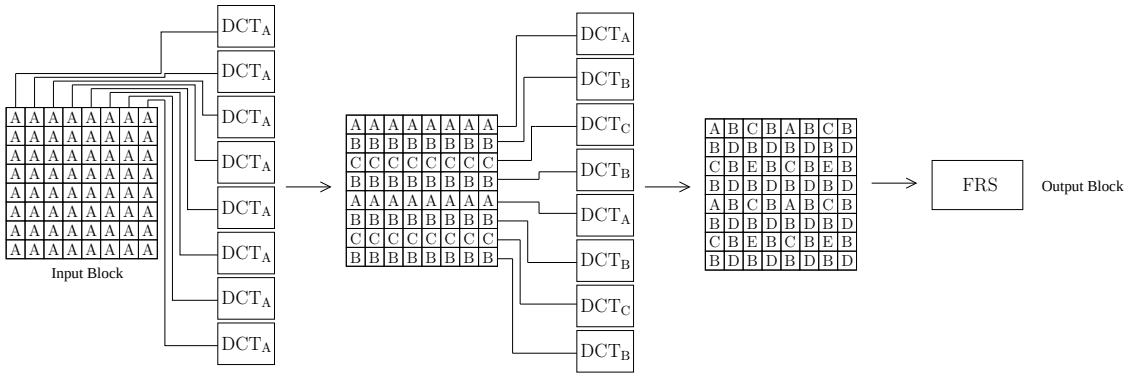


Figure 3.7: Graphical representation of the entire 2D DCT parallel architecture using AI representation including FRS. The blocks $\mathrm{DCT_A}$, $\mathrm{DCT_B}$, and $\mathrm{DCT_C}$ represent the 1D DCT in the AI-based representation for inputs in classes A, B, and C, respectively, and are implemented by the algorithms in the Figs. 3.2, 3.4, and 3.5.

Table 3.3: 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.* [28]	No	384	0	No
Madanayake *et al.* [33]	Yes	1064	32	No
Edirisuriya *et al.* [34]	Yes	1064	32	No
Pradini *et al.* [39]	No	304	0	Yes
Wahid *et al.* [40]	No	384	64	Yes
Wahid *et al.* [41]	No	384	64	No
Rajapaksha *et al.* [42]	Yes	1064	32	No
Fu *et al.* [43]	No	352	360	Yes
Present work	Yes	892	92	Yes

3.4 Final Reconstruction Step

The final reconstruction step (FRS) block performs the AI decoding described in Eq. (3.2). It maps AI quantities back to a fixed-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 [13] and (ii) the expansion factor method [13, 44] 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 become clear in the following sections. Several methods for MCM evaluation with different constants have been developed by means of optimization and graph theory [45–48].

For solving the present MCM problems, we employ the method described in [49], which is based on number recoding and subexpression factorization and which has not been applied to the design of FRS block in previous studies [28, 33, 34, 39–43]. MCM evaluation

can save up to 40% of area in FPGA implementation and provide faster computation when compared to routine methods [49].

3.4.1 Dyadic Approximation Method

The irrational quantities required by the FRS can be approximated with arbitrary precision by dyadic integers [13]. Dyadic integers are of the form $p/2^k$, where p is an odd integer and $k \in \mathbb{N}$ N; this form enables them to be efficiently implemented in hardware [12, 13, 50]. 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 obtain:

$$\zeta \approx \left[1 \quad \frac{4017}{2^{11}} \quad \frac{3784}{2^{11}} \quad \frac{3406}{2^{11}} \quad \frac{2896}{2^{11}} \quad \frac{2276}{2^{11}} \quad \frac{1567}{2^{11}} \quad \frac{799}{2^{11}} \right]^{\top}. \tag{3.4}$$

For the decoding of 2D DCT output coefficients into a fixed-point representation, we need to consider the quantities in each AI number class and design specific algorithms for each class. Considering the classes shown in Table 3.2, we can use the algorithms shown in Tables 3.4 and 3.5 for the approximate ζ with 11-bit wordlength. The operation $x \ll k$ denotes the left shift of k bits over the integer quantity x (i.e., $x \cdot 2^k$), while $x \gg k$ denotes the right shift of k bits.

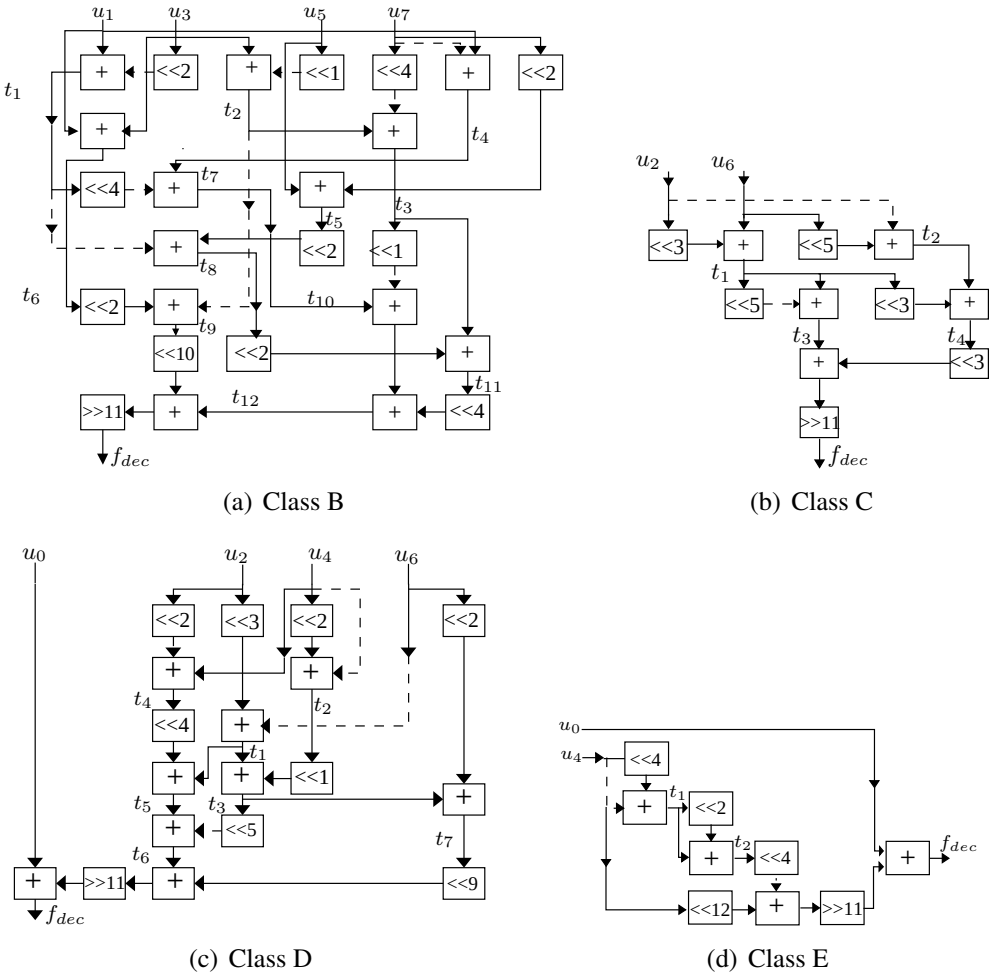
For the purpose of clarity, we also show the graphical representation of the FRS block for the dyadic approximation method for an 11-bit word length in the Figure 3.8.

Table 3.4: Fast algorithms for FRS for Dyadic Approximation for an 11-bit word length and its arithmetic cost for Class A, and Class B.

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

Table 3.5: Fast algorithms for FRS for Dyadic Approximation for an 11-bit word length and its arithmetic cost for Class C, Class D, and Class E.

Class	Algorithm	Output $f_{\text{dec}}(x;\boldsymbol{\zeta})$	Arithmetic Cost	
			Additions	Shifts
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



(a) Class B

(b) Class C

(c) Class D

(d) Class E

Figure 3.8: FRS block using dyadic approximation for (a) Class B, (b) Class C, (c) Class D, and (d) Class E for an 11-bit wordlength.

3.4.2 Expansion Factor

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

$$\alpha^* \cdot f_{\text{dec}}(\mathbf{x};\zeta) \approx \mathbf{x}^\top \cdot \text{round}(\alpha^* \cdot \zeta), \tag{3.5}$$

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

$$\alpha^* = \arg\min_{\alpha>1} \parallel \alpha \cdot \zeta - \text{round}(\text{ff} \cdot \text{1}) \parallel . \tag{3.6}$$

In this context, all components of the basis vector ζ are taken into account and have the same weight. However, such an outcome is not suitable for this problem.In fact, the required number of multiplications by each of the components of ζ is not uniform. This can be seen by the output pattern of the 2D DCT coefficients shown in Figure 3.3(c). Therefore, in order to obtain a more precise estimation for α^*, we must take into account the relative frequency of occurrence of the multiplications of the coefficients of ζ. This results in the following optimization problem:

$$\alpha^* = \arg\min_{\alpha>1} \| \boldsymbol{f}^\top \cdot (\alpha \cdot \zeta - \text{round}(\text{ff} \cdot \text{1})) \|, \tag{3.7}$$

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 [4], we obtain

$$\boldsymbol{f} = \left[\frac{24}{220} \quad \frac{32}{220} \quad \frac{24}{220} \quad \frac{32}{220} \quad \frac{20}{220} \quad \frac{32}{220} \quad \frac{24}{220} \quad \frac{32}{220} \right]^{\top}. \tag{3.8}$$

The problem in Eq. (3.7) is non-linear and has no closed solution in terms of simple algebraic functions. In order to solve Eq. (3.7) 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 that it will not impose 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 $\alpha^* = 1849.39$, leading to

$$1849.39 \cdot \zeta = \begin{bmatrix} 18449.39 \\ 3618.97\ldots \\ 3409.00\ldots \\ 3068.02\ldots \\ 2609.13\ldots \\ 2049.98\ldots \\ 1412.05\ldots \\ 719.85\ldots \end{bmatrix} \approx \begin{bmatrix} 1845 \\ 3619 \\ 3409 \\ 3068 \\ 2609 \\ 2050 \\ 1412 \\ 720 \end{bmatrix}. \tag{3.9}$$

Table 3.6 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 the fixed-point representation, we need to consider the different number classes shown in Table 3.2 and design specific algorithms for each; for this purpose, we adopted the MCM method described in [49]. We derived the algorithms shown in Table 3.7 for the optimal constant $\alpha^* = 1849.39$ with an 11-bit word length.

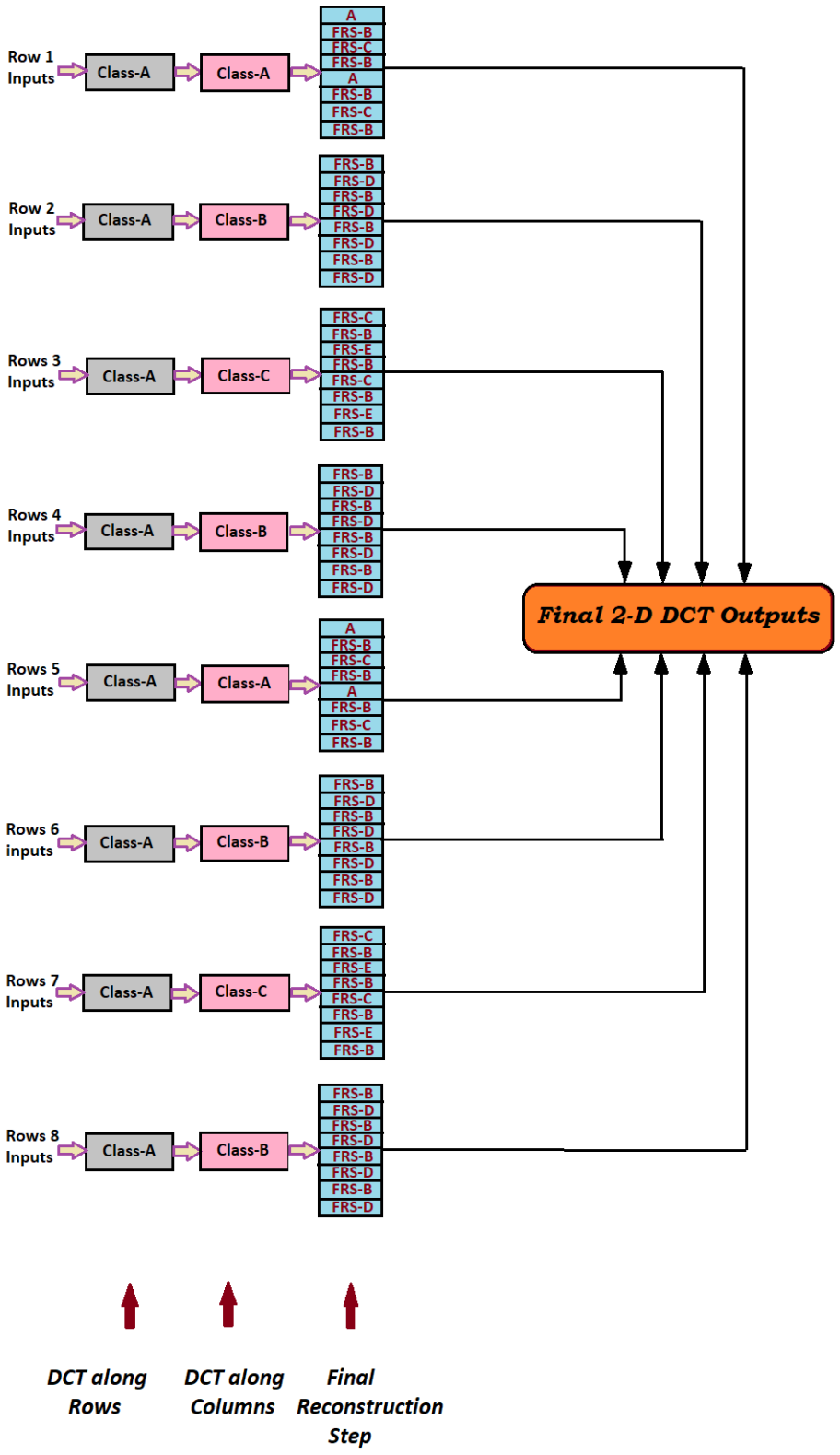


Figure 3.9: Pictorial representation of 2D-DCT

Table 3.6: 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	$5.03 \cdot 10^{-4}$	$8.31 \cdot 10^{-5}$

Thus, concluding this chapter with all the discussions of the theoretical and mathematical explanations of the algebraic integers, classes (their algorithms), and final reconstruction step (FRS) etc., [51]. All the reasons for the adoption of the present thesis are discussed in detail in this chapter. In the successive chapter 4, the final step of the thesis is discussed.

Table 3.7: Fast algorithms for FRS for Expansion Factor Method for an 11-bit word length ($\alpha^* = 1849.39$) and its arithmetic cost for Class A, Class B, and Class C.

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

Table 3.8: Fast algorithms for FRS for Expansion Factor Method for an 11-bit word length ($\alpha^* = 1849.39$) and its arithmetic cost for Class D, and Class E.

Class	Algorithm	Output $f_{\text{dec}}(x;\boldsymbol{\zeta})$	Arithmetic Cost	
			Additions	Shifts
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

CHAPTER IV

3D-DCT AND MONOCHROME VIDEO

After image processing, the next topic for discussion is the application of DCT for video compression. Once we have achieved the 2D-DCT for a particular image, we create a video, calculate its number of frames, and apply the 3D-DCT to the video by constructing an $8 \times 8 \times 8$ 3D-DCT architecture. The 3D-DCT is obtained from the combination of 2D-DCT and modified round DCT (MR-DCT). Let us begin with a discussion about the MR-DCT and its benefits over other 1D transforms.

4.1 Introduction to MR-DCT

MRDCT is a low-complexity 8-point orthogonal approximate DCT. The transform requires no multiplications or bit-shift operations. The derived fast algorithm requires only 14 additions, which is less than any existing DCT approximation. Moreover, in several image compression scenarios, this MR-DCT transform could outperform the well-known signed DCT (SCDT) as well as other state-of-the-art algorithms. This transform requires only 14 additions and has comparable or better image compression performance than the classic SDCT and the state-of-the-art BAS-2011 transform. Prominent approximation-based techniques include the SDCT [52], the level 1 approximation by Lengwehasatit and Ortega [53], the Bouguezel–Ahmad–Swamy (BAS) series of algorithms [54–56], and the

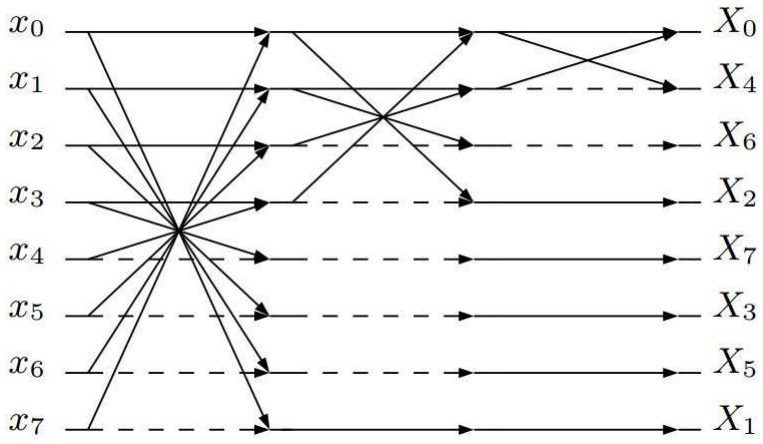


Figure 4.1: Signal flow graph for MR-DCT.

DCT round-off approximation [57]. In general, the transformation matrix entries required by approximate DCT methods are only $0, \pm 1/2, \pm 1, \pm 2$. This implies null multiplicative complexity, because the involved operations can be implemented exclusively by means additions and bit-shift operations. This algorithm attains the lowest computational complexity among available methods found in literature.At the same time, it could outperform state-of-the-art approximations.

Figure 4.1 represents the algorithm for the MR-DCT using a signal flow graph. Input data x_n, $n = 0, 1, ..., 7$, relates to output data X_k, $k = 0, 1, ..., 7$, according to $X = T * x$. Dashed arrows represent multiplication by -1.

where the matrix for ”T” is given as below:

$$\begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & -1 \\ 1 & 0 & 0 & -1 & -1 & 0 & 0 & 1 \\ 0 & 0 & -1 & 0 & 0 & 1 & 0 & 0 \\ 1 & -1 & -1 & 1 & 1 & -1 & -1 & 1 \\ 0 & -1 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & -1 & 1 & 0 & 0 & 1 & -1 & 0 \\ 0 & 0 & 0 & -1 & 1 & 0 & 0 & 0 \end{bmatrix}$$

4.2 Implementation of 3D-DCT

As discussed earlier, 3D-DCT is most popular transform for video compression. Here we are implementing 3D-DCT using a multplierless 2D and 1D transform of DCT so as to reduce the area and the complexity of the system. The design of the combined 2D-DCT and MR-DCT is presented in Chapter 5 as an implementation on Simulink.

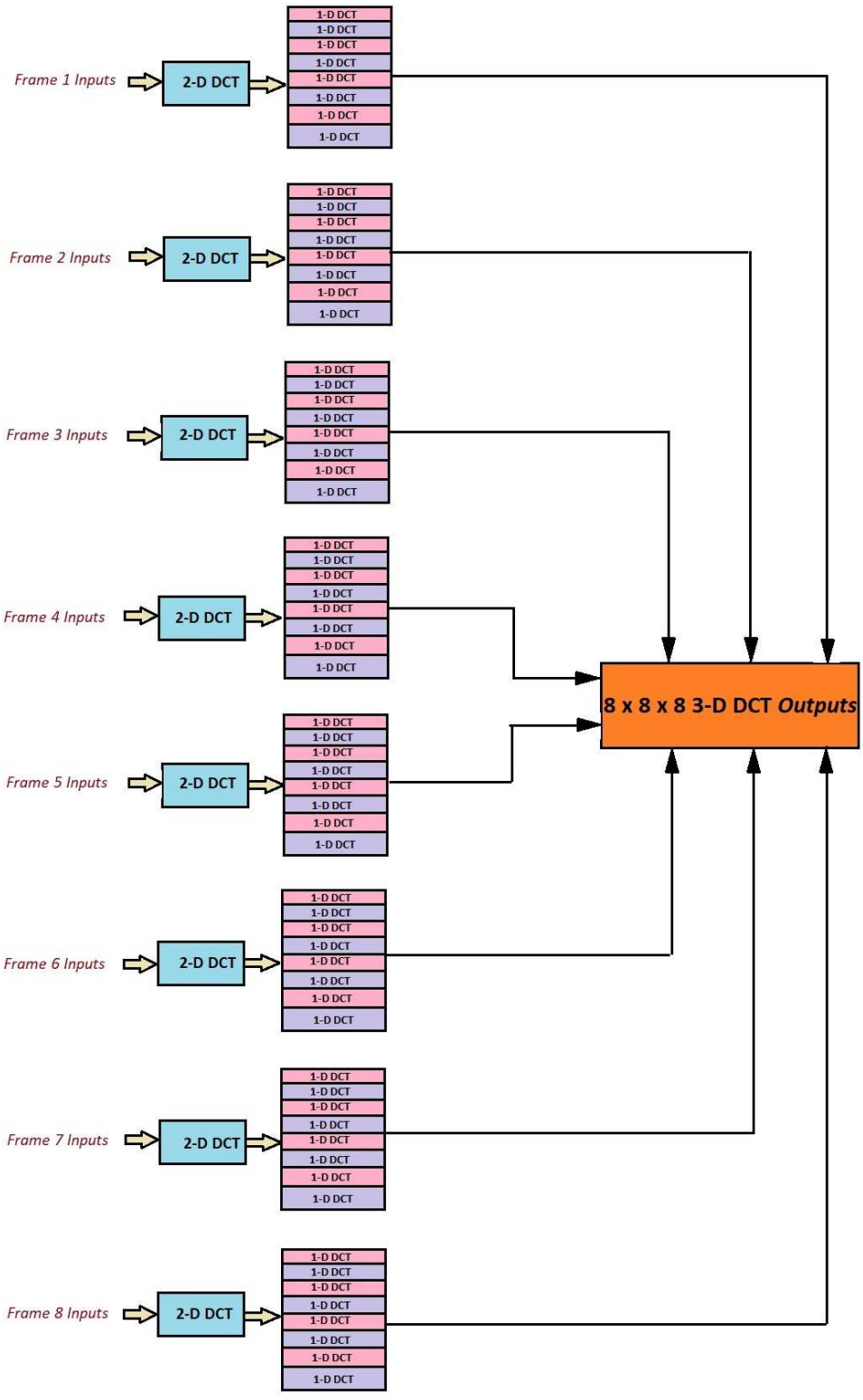


Figure 4.2: Overview of 3-D DCT

Table 4.1: Comparison of MR-DCT with other proposed DCTs

Method	Additions	Multiplications	Shifts	Total
MR-DCT	14	0	0	14
SDCT [52]	24	0	0	24
Level 1 approximation [53]	24	0	2	26
BAS-2008 transform [54]	18	0	2	20
BAS-2009 transform [55]	18	0	0	18
BAS-2011 transform [56]	18	0	2	20
CB-2011 transform [57]	22	0	0	22

A video from the real time is recorded with a frame rate of 5 frames/sec. Duration of video is considered to be 8 sec with total number of frames to be 40. Each frame is 640 x 480. Therefore the first eight frames are considered for the first 2D-DCT step and then 1D-DCT is applied to the result using the MR-DCT algorithm. All together 64 1D-DCT's are required to operate on 8×8 2D-DCT outputs.

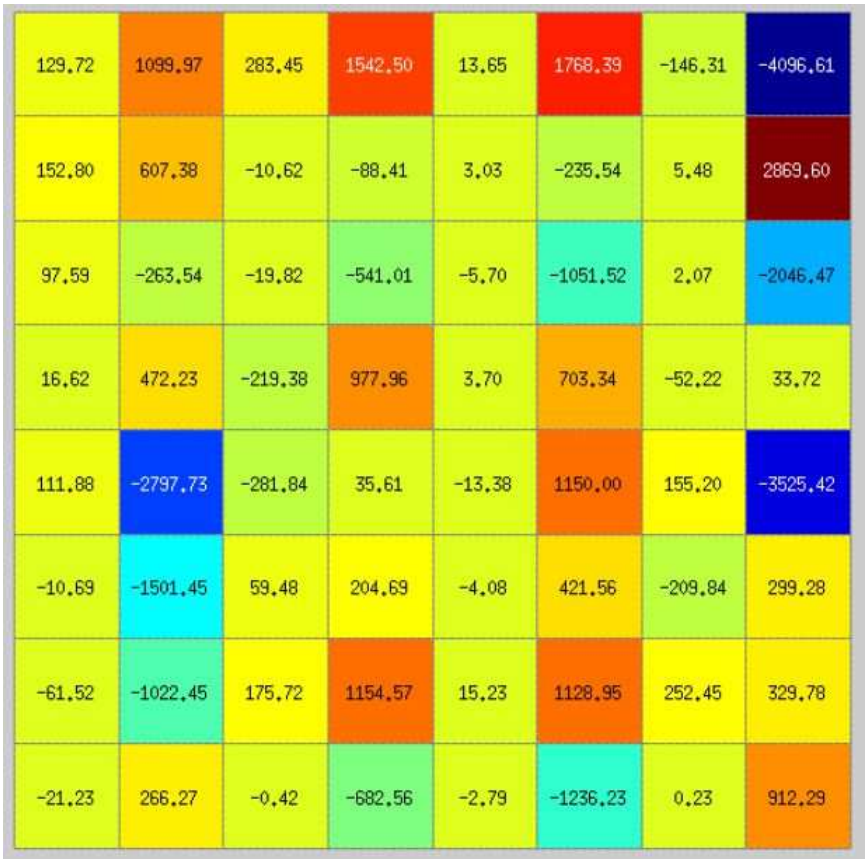


Figure 4.3: 3D-DCT output of 8×8 input of first frame representing each pixel and its coefficient value with its corresponding color

CHAPTER V

DIGITAL FPGA IMPLEMENTATION OF THE ARCHITECTURE

5.1 Designing of algorithms for classes on simulink.

In the work presented in this thesis, the major components of the 2D DCT and the 3D DCT architecture come from classes that are derived from the mathematical transform of the DCT. In Chapter 3, we discussed the equations and the number of operations required for the construction of the algorithms. In simulink, the precision and binary point are set to a particular number (for example, beginning with 8), and the algorithms are built. This chapter discusses the design of the algorithms on Simulink and the implementation of the algorithms in a field-programmable gate array (FGPA) and an application-specific integrated circuit (ASIC).

5.2 Design of algorithms on Simulink for the final reconstruction step

It has been established that the final reconstruction step (FRS) block performs the AI decoding described earlier and maps the AI quantities back to fixed 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

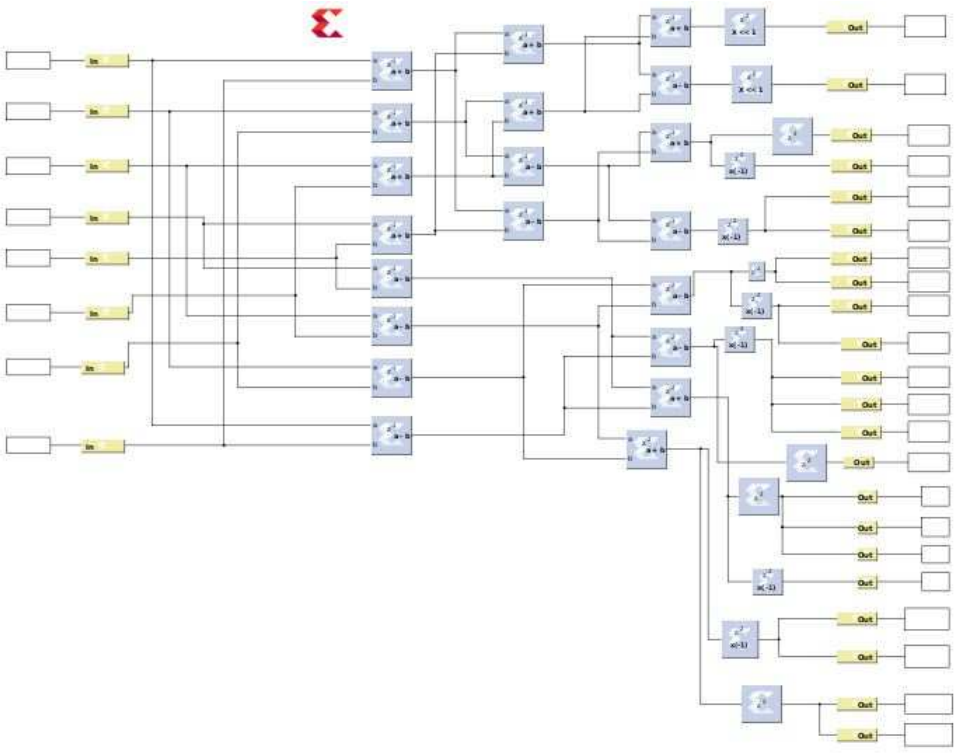


Figure 5.1: Figure representing a Class A algorithm on Simulink.

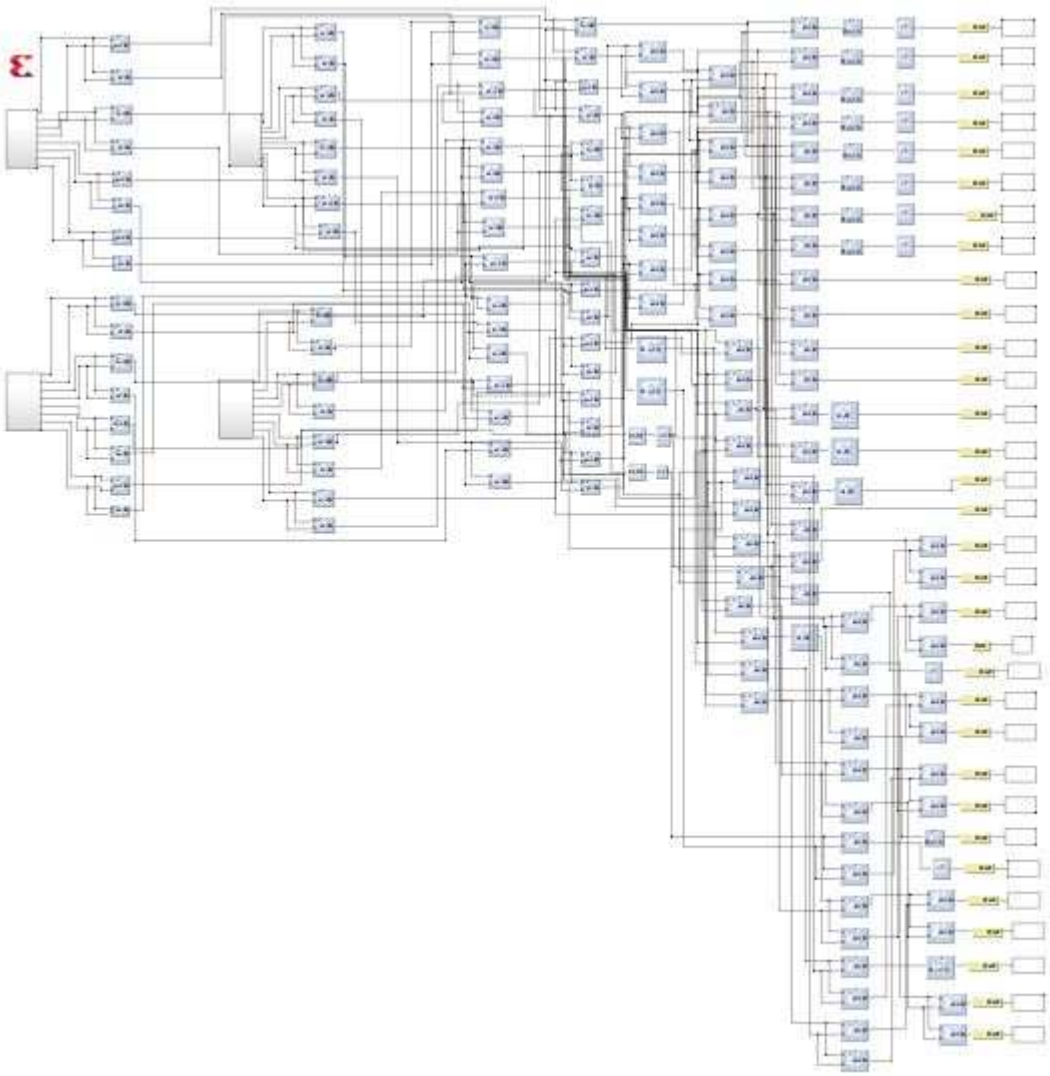


Figure 5.2: Figure representing a Class B algorithm on Simulink.

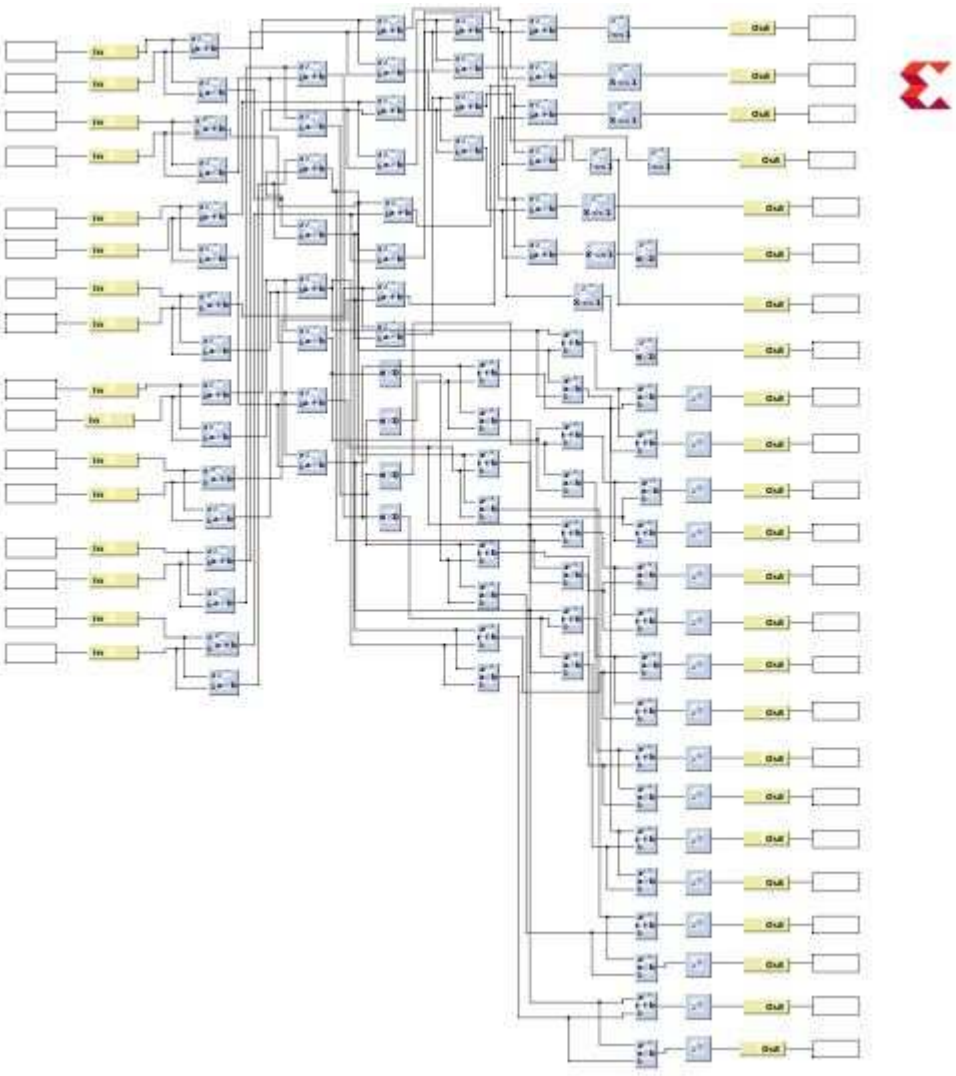


Figure 5.3: Figure representing a Class C algorithm on Simulink.

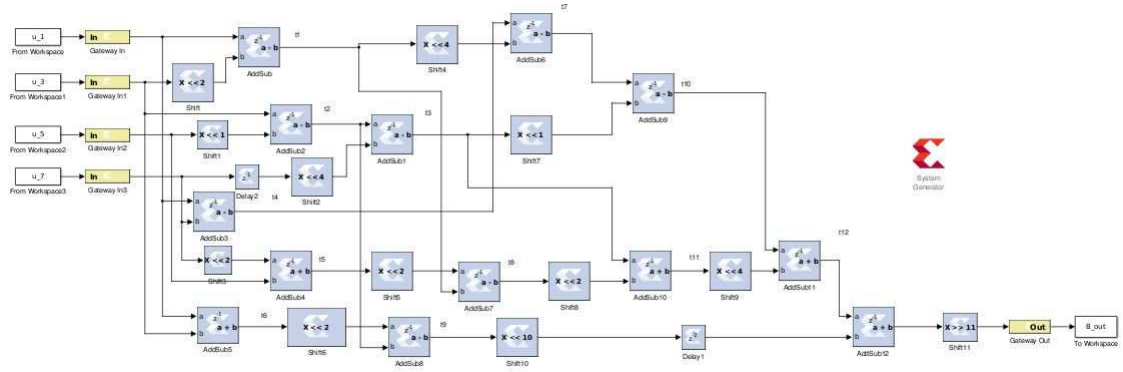
representation. One of the advantages of this method is that it does not require intermediate reconstructions.

5.3 Design and Implementation of the 2-D architecture using XILINX ML605

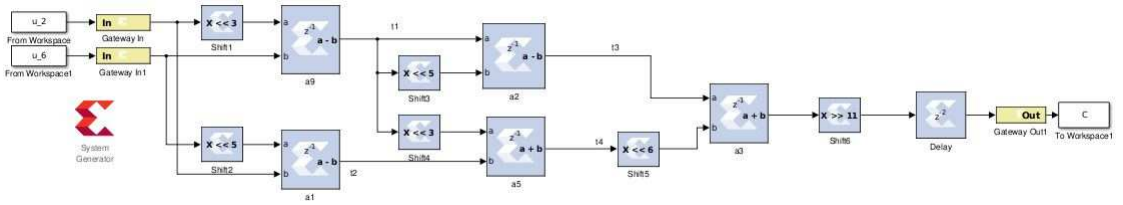
The 2D DCT is created using XSG Matlab/Simulink, realized, and tested within Virtex-6 LX240T FPGA on ML605. By using ML605. The Xilinx ISE, XSG and Matlab/Simulink components allow for rapid prototyping and on-chip verification of designs. The ASIC synthesis and place-and-route results give precise information about the hardware consumption and speed increment, as compared to **.xflow** results. For the ASIC synthesis, the hardware description language code was ported to 180-nm complementary metal-oxide-semiconductor (CMOS) technology using analog/mixed signal (AMS) standard cells and subject to synthesis and place-and-route using the Encounter tool in Cadence. The adopted figures of merit were: Maximum frequency (in MHz),throughput,block rate, pixel rate , area (in mm^2), power (in watts) and final Normalized Dynamic power derived from the supply voltage (in $mW/MHz.V_{sup}^2$).

5.3.1 FPGA Implementation

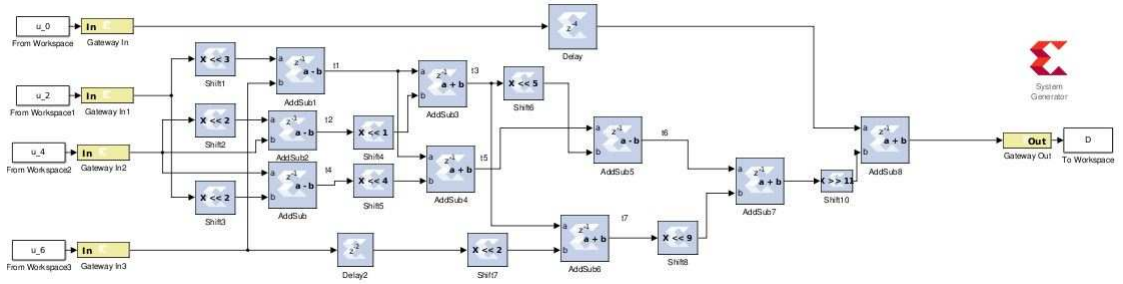
A fully parallel architecture for the real-time implementation of the proposed 2D DCT using AI encoding has been designed, simulated and implemented using FPGA technology. The architecture assumes 64 parallel input channels pertaining to the 64 locations of an 8×8 matrix of pixel values, which are assumed to be of 8-bit signed values using the two's complement format. The inputs are assumed to be normalized in the range of -1



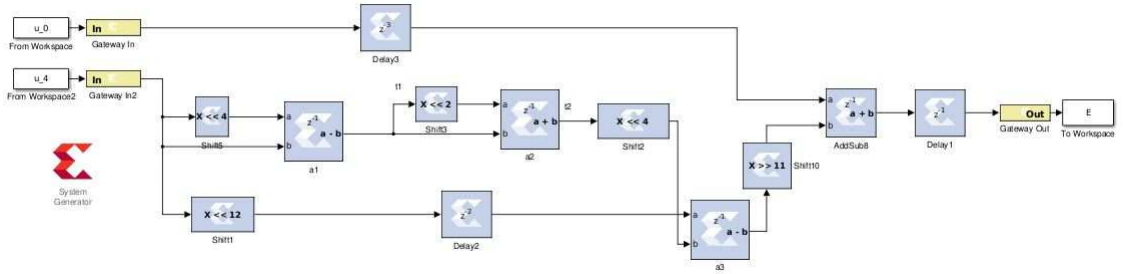
(a) FRS for Class B



(b) FRS for Class C



(c) FRS for Class D



(d) FRS for Class E

Figure 5.4: Figures representing algorithms on Simulink for the final reconstruction step (FRS) using a dyadic approximation method.

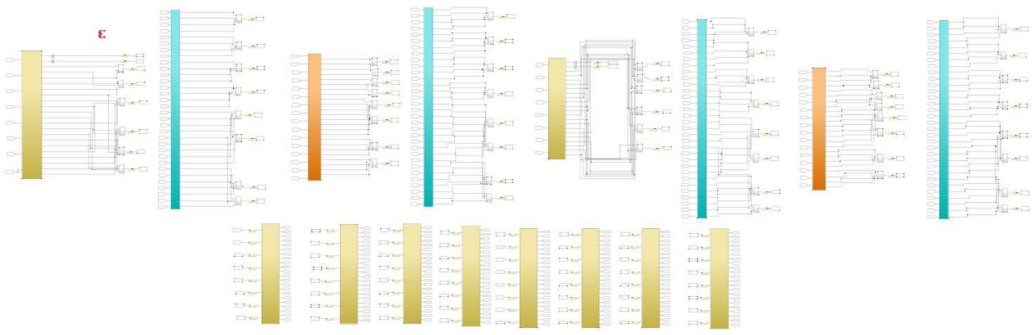


Figure 5.5: Simulink design of 2D-DCT with final reconstruction step (FRS)

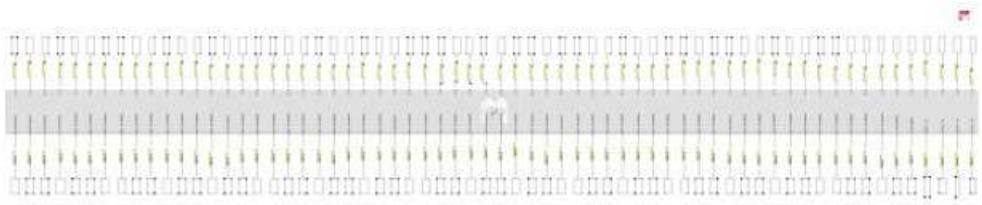


Figure 5.6: 2D-DCT hardware block generated on FPGA

to $127/128$. The AI encoded architectures corresponding to Algorithm 3.2, 3.4, and 3.5 are realized in parallelized digital hardware.

The 64 output coefficients are maintained in the AI-encoded infinite precision format until the FRS block for conversion to fixed-point representation for subsequent processing. The algorithms for multiple constant multiplication described in Table 3.4 and 3.7 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 $1.5 \cdot 10^4$ randomly generated 8×8 input vectors to verify correct operation. The verified design was thereafter targeted to a Xilinx Virtex-6 XC6VLX240T-1FFG1156 FPGA

3.3643	-0.3098	0.4548	0.3551	0.5791	0.2188	0.3708	0.0297
-0.6100	0.2699	0.1102	-0.2354	-0.5140	-0.2869	-0.1500	-0.3812
-0.2793	-0.0741	0.1890	0.0257	-0.3148	0.2910	0.0723	0.2295
-0.2911	0.1985	-0.3273	-0.4392	0.2968	0.0106	-0.0255	0.1280
0.4561	0.0028	0.0712	0.1715	-0.4814	0.0823	0.1132	-0.3704
0.3271	-0.1632	0.1900	-0.0458	-0.3309	0.1442	-0.5149	0.1270
-0.1919	0.1140	-0.1289	0.0325	0.4063	-0.0010	0.4086	-0.3796
-0.1354	-0.0900	-0.4708	0.0497	0.8741	-0.1767	-0.0494	0.5758

Figure 5.7: DCT outputs of Matlab using matlab ”dct2” command

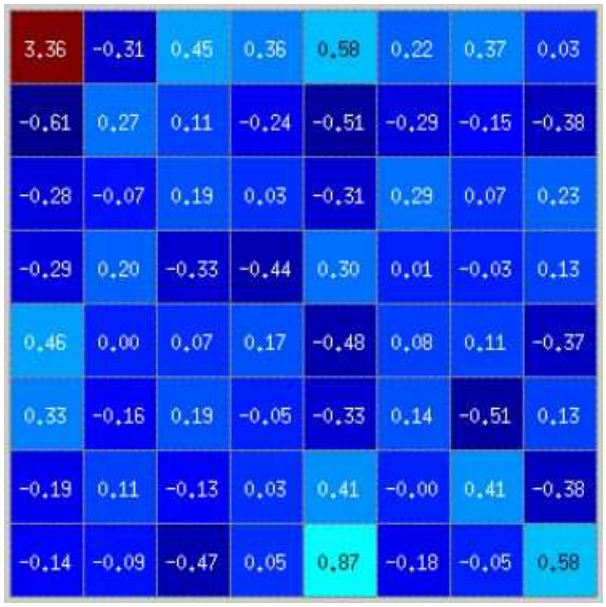
3.3643	-0.3098	0.4546	0.3550	0.5791	0.2188	0.3706	0.0295
-0.6101	0.2698	0.1101	-0.2356	-0.5142	-0.2871	-0.1501	-0.3813
-0.2793	-0.0742	0.1890	0.0256	-0.3149	0.2910	0.0723	0.2292
-0.2913	0.1982	-0.3274	-0.4392	0.2966	0.0105	-0.0256	0.1279
0.4561	0.0027	0.0710	0.1714	-0.4814	0.0823	0.1130	-0.3706
0.3271	-0.1633	0.1899	-0.0459	-0.3311	0.1440	-0.5149	0.1270
-0.1921	0.1138	-0.1289	0.0325	0.4060	-0.0012	0.4087	-0.3796
-0.1355	-0.0901	-0.4709	0.0496	0.8740	-0.1768	-0.0496	0.5757

Figure 5.8: DCT outputs of hardware designed and generated on FPGA

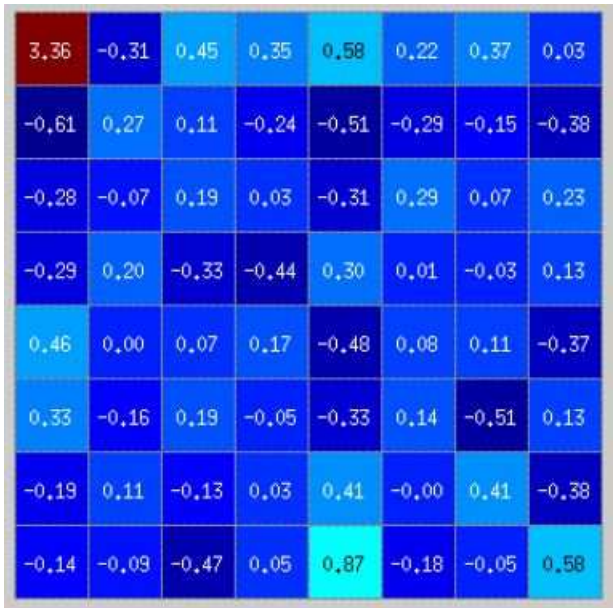
device installed on a Xilinx ML605 evaluation platform. The design was subjected to physical implementation and was tested using $1.5 \cdot 10^4$ test matrices provided to the implementation using stepped hardware co-simulation on the JTAG port. The FPGA resource consumption and metrics are shown in Table 5.1, and the available metrics of other designs are shown in Table 3.3.

The throughput is calculated as the number of output coefficients in each cycle; the designs from the works listed in Table 5.1 are classified into fully parallel 64 coefficients per clock cycle (FPar64), row parallel 8 coefficients per clock cycle (RPar8), and fully serial 1 coefficient per clock cycle (FSer1).

The block and pixel rates represent the number of 8×8 blocks and pixels processed per second. The maximum clock frequency for potential real-time operation is 360



(a) Matlab



(b) Hardware

Figure 5.9: Comparison of matlab and hardware generated outputs for image from real-time

Table 5.1: Comparison of FPGA implementation metrics.

Method	Max. Freq.	Board	Throughput	Block Rate	Pixel Rate
Madanayake *et al.* [33]	307 MHz	Xilinx Virtex-6 (XC6VLX240T)	RPar8*	$4.78 \cdot 10^6$	$3.83 \cdot 10^7$
Edirisuriya *et al.* [34]	316 MHz	Xilinx Virtex-6 (XC6VLX240T)	FSer1**	$4.93 \cdot 10^6$	$4.93 \cdot 10^6$
Wahid *et al.* [40] (1D DCT)	36.7 MHz	Actel A500K (A500K050)	–	–	–
Wahid *et al.* [41]	101 MHz	Xilinx Virtex-E (XCV200E-8)	–	–	–
Rajapaksha *et al.* [42]	302 MHz	Achronix SPD60	RPar8*	$4.71 \cdot 10^6$	$3.77 \cdot 10^7$
Proposed	360 MHz	Xilinx Virtex-6 (XC6VLX240T)	FPar64***	$3.6 \cdot 10^8$	$2.30 \cdot 10^{10}$

*RPar8 means fully parallel 64 coefficients per clock cycle,
**FSer1 means row parallel 8 coefficients per clock cycle,
***FPar64 means fully serial 1 coefficient per clock cycle.

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 with a sustained data processing rate of 184.32 Gbps (internal to the core). Obtaining such a high data rate for 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 [40] proposes a 2D DCT algorithm, but only the FPGA implementations results for 1D DCT are presented and therefore used in Table 5.1. The only works containing complete error-free implementation of 2D DCT are [33, 34, 42]. The proposed design demands a total of 26,000 registers and 30,200 look-up tables (LUTs) used for logic versus the 10,282 registers and 12,007 LUTs required by the design in [33]. Although the number of registers and LUTs used are around three times larger, the proposed design offers a block processing rate that is 100 times higher and a pixel rate that is 1000 times higher than the design in [33].

Table 5.2: Comparison of ASIC implementation metrics.

Method	Pradini *et al.* [39]	Fu *et al.* [43]	**Present Work**
Maximum Frequency	210 MHz	75 MHz	893 MHz
Technology	0.18 μm CMOS	0.18 μm CMOS	0.18 μm CMOS
Throughput	FPar64*	FSer1**	FPar64***
Block rate	4.20×10^7	1.71×10^6	8.93×10^8
Pixel rate	2.68×10^9	7.50×10^7	5.71×10^{10}
Area	–	2.16 mm^2	7.22 mm^2
Dynamic Power	–	–	11.85 W
Normal Dynamic Power	–	–	13.269 mW/MHz·V^2_{sup}

* FPar64 = fully parallel 64 coefficients per clock cycle.
**RPar8 = row parallel 8 coefficients per clock cycle.
***FSer1 = fully serial 1 coefficient per clock cycle.

5.3.2 ASIC Synthesis

Aside 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 for the ASIC synthesis is 1.8 V [51]. The results for the ASIC metrics are shown in Table 5.2.

5.3.3 Implementation of MR-DCT

As referred earlier Modified Round DCT was developed by Dr.Cintra which is constructed using only 14 additions.The modified round DCT (MR-DCT) [3], is initially designed on Simulink, and its subsystem is then connected to the outputs from the 2D DCT blocks.

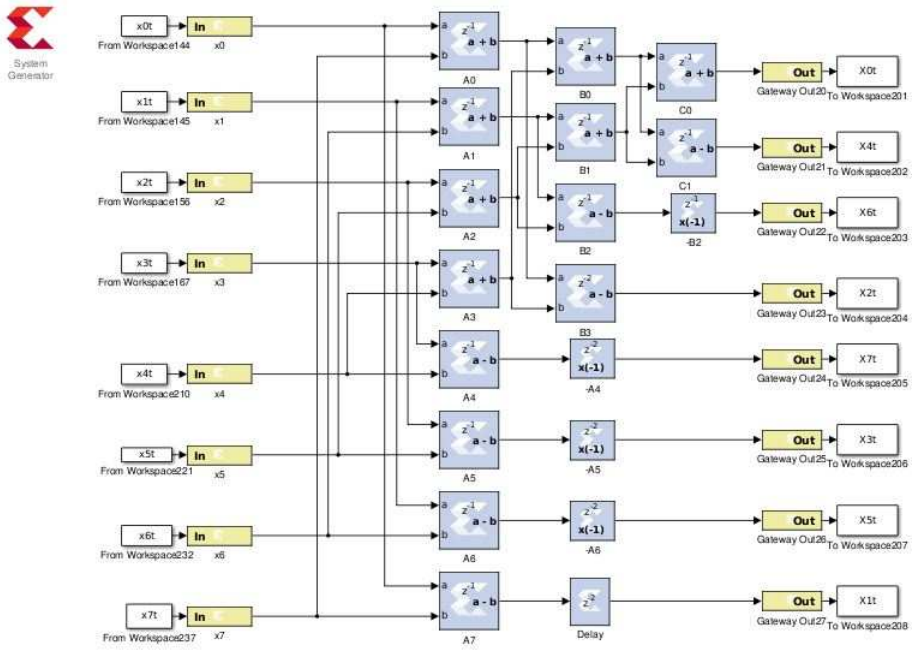


Figure 5.10: Figure representing MR-DCT algorithm on Simulink.

5.4 Design and Implementation of the 3-D DCT architecture

After the implementation of 3D-DCT architecture on Simulink, it is fully pipe lined to perform the operations at a time. Then FPGA and ASIC analysis is performed same as for 2D-DCT. This provides the metrics like best time achievable, maximum frequency and the area occupied by the architecture.

5.4.1 FPGA Implementation

The maximum clock frequency of the 3D-DCT architecture for potential real time operation is 357.2 MHz. The best time achievable for the architecture is 2.8 ns.It uses a total of 34,302 number of flip flops, which is 11% of the total number of flip flops available for building an architecture. It uses 34,130 number of look up tables (LUT's), which is 22%

Table 5.3: Measurements recorded for 3D-DCT using ML605 kit

Metrics	Results
Maximum Frequency	357.2 MHz
Time	2.8 ns
Flip Flops	34,302
LUT's	34,130
CLB's	9,589

Table 5.4: Measurements recorded for 3D-DCT using ASIC.

Metrics	Results
Technology used here	0.18 μm CMOS
Throughput	FPar64*
Maximum frequency	663.57MHz
Dynamic power	106.06 $Watts$
Area	70.49 mm^2

* FPar64 = fully parallel 64 coefficients per clock cycle.

of the total number of LUT's available for building an architecture, and 9,589 number of combinational logic blocks (CLB's), which is 25% of the available CLB's.

5.4.2 ASIC Synthesis

ASIC synthesis for 3D-DCT architecture generated the reports for area,time,power and gate distribution. The time generated is the reciprocal of the maximum frequency which is 663.57MHz. The dynamic power is given as 106.06 Watts and the area required for the architecture is given as 70.49 mm^2.

CHAPTER VI

CONCLUSIONS AND FUTURE WORK

6.1 Conclusions

In this study, a digital 8 $\times$ 8 $\times$ 8 3D-DCT architecture is developed from 8 $\times$ 8 2D DCT and MR-DCT (with only 14 additions). A massively parallel and pipe lined architecture is proposed for real-time implementations that is capable of compressing the videos for a provided number of pixels. The proposed method can be considered to be less complex since it uses only adders/subtractors and shifters. This implementation is very advantageous in terms of time, power and complexity.

6.2 Future Work

1. The performance of the architecture can be verified in real-time by implementing it on the chip.

2. The constructed 8 $\times$ 8 $\times$ 8 3D DCT digital hardware can be used to explore many applications that involve digital image and video processing. There is scope for wide extension of the 3D DCT from monochrome video implementation to a full-color video.

3. Efforts can be made to reduce the size of the architecture with a few modifications in the design so as to reduce the size of the chip.

BIBLIOGRAPHY

[1] A.M. Raid, W.M. Khedr, M.A. El-Dosuky, and W. Ahmed. Jpeg image compression using discrete cosine transform-a survey. *arXiv preprint arXiv:1405.6147*, 2014.

[2] Robert J Schalkoff. *Digital image processing and computer vision*, volume 286. Wiley New York, 1989.

[3] F. M. Bayer and R. J. Cintra. Dct-like transform for image compression requires 14 additions only. *Electronics Letters*, 48(15):919–921, July 2012.

[4] 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*, 10(6):633–640, 2016.

[5] A.B. Watson. Image compression using the discrete cosine transform. *Mathematica*, 4(1):81, 1994.

[6] V. Lecuire, L. Makkaoui, and J.-M. Moureaux. Fast zonal dct for energy conservation in wireless image sensor networks. *Electronics Letters*, 48(2):125–127, 2012.

[7] N. Ahmed, T. Natarajan, and K. R. Rao. Discrete cosine transform. *IEEE Transactions on Computers*, C-23(1):90–93, Jan 1974.

[8] K.R. Rao. Discrete cosine transform-algorithms, advantage and applications. *Proceedings of the Korean Institute of Communication Sciences Conference*, pages 3–6, 1989.

[9] William. B. Pennebaker and Joan L. Mitchell. *JPEG: Still image data compression standard*. Springer Science & Business Media, 1992.

[10] Y. Arai, T. Agui, and M. Nakajima. A fast dct-sq scheme for images. *IEICE transactions (1976-1990)*, 71:1095–1097, 1988.

[11] X. Shao and S.G. Johnson. Type-ii/iii dct/dst algorithms with reduced number of arithmetic operations. *Signal Processing*, 88(6):1553–1564, 2008.

[12] Alan V. Oppenheim, Ronald W. Schafer, Mark T. Yoder, and Wayne T. Padgett. *Discrete Time Signal Processing*, volume 1. Prentice-Hall, Inc., Upper Saddle River, NJ, 3rd edition, August 2009.

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

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

[15] 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*, 2(3):498–502, June 2012.

[16] 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*, pages 577–580, 2009.

[17] J. Xiao and Y. Wang. Toward a better understanding of DCT coefficients in watermarking. In *Pacific-Asia Workshop on Computational Intelligence and Industrial Application*, volume 2, page 2:206–209, 2008.

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

[19] G. K. Wallace. The JPEG still picture compression standard. *IEEE Transactions on Consumer Electronics*, 38(1):xviii–xxxiv, February 1992.

[20] N. Roma and L. Sousa. Efficient hybrid DCT-domain algorithm for video spatial downscaling. *EURASIP Journal on Advances in Signal Processing*, 2007(57291), June 2007.

[21] 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*, 13(7):560–576, July 2003.

[22] 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*, 1(3):36–46, July 2012.

[23] 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*, volume 2, pages 988–991, May 1989.

[24] M. T. Heideman and C. S. Burrus. *Multiplicative complexity, convolution, and the DFT*. Springer-Verlag, New York, 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.

[25] P. Duhamel and Hedi H'Mida. New 2^n DCT algorithms suitable for VLSI implementation. In *IEEE International Conference on Acoustics, Speech, and Signal Processing*, pages 1805–1808, 1987.

[26] 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*, 31(5):580–588, September 1985.

[27] J. H. Cozzens and L. A. Finkelstein. Range and error analysis for a fast Fourier transform computed over $Z[\omega]$. *IEEE Transactions on Information Theory*, 33(4):9, July 1987.

[28] V. Dimitrov, K. A. Wahid, and G. Jullien. Multiplication-free 8×8 2D DCT architecture using algebraic integer encoding. *Electronics Letters*, 40(20):1310–1311, September 2004.

[29] V. Dimitrov and K. A. Wahid. On the error-free computation of fast cosine transform. *Information Theories and Applications*, 12(4):321–327, 2005.

[30] 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*, 54(8):700–704, July 2007.

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

[32] Y. Arai, T. Agui, and M. Nakajima. A fast DCT-SQ scheme for images. *IEICE Transactions*, E71(11):1095–1097, November 1988.

[33] 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*, 22(6):915–929, June 2012.

[34] 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*, 23(12):2083–2089, June 2013.

[35] Alan V. Oppenheim, Ronald W. Schafer, and John R. Buck. *Discrete-time signal processing*, volume 1. Prentice-Hall, Inc., Upper Saddle River, NJ, 2nd edition, 1999.

[36] S. An and C. Wang. A computation structure for 2-D DCT watermarking. In *52nd IEEE International Midwest Symposium on Circuits and Systems*, pages 577–580, Ag 2009.

[37] F. M. Bayer and R. J. Cintra. Image compression via a fast DCT approximation. *IEEE Latin America Transactions*, 8(6):708–713, January 2011.

[38] 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)*, pages 2202–2205, May 2016.

[39] 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)*, pages 1–6, July 2011.

[40] 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)*, pages 214–221, June 2005.

[41] K. Wahid, V. Dimitrov, and G. Jullien. New encoding of 8×8 DCT to make H.264 lossless. In *IEEE Asia Pacific Conference on Circuits and Systems*, pages 780–783, 2006.

[42] 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*, February 2013. Article ID 834793.

[43] 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)*, volume 2, pages II–765–8 Vol.2, May 2004.

[44] Gerlind Plonka. A global method for invertible integer DCT and integer wavelet algorithms. *Applied and Computational Harmonic Analysis*, 16(2):90–110, March 2003.

[45] S. Athar and O. Gustafsson. Optimization of AIQ representations for low complexity wavelet transforms. In *20th European Conference on Circuit Theory and Design (ECCTD)*, pages 314–317, August 2011.

[46] 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)*, volume 3, pages 652–655, August 2002.

[47] H. Ohlsson O. Gustafsson and L. Wanhammar. Improved multiple constant multiplication using a minimum spanning tree. In *Thirty-Eighth Asilomar Conference on Signals, Systems and Computers*, volume 1, pages 63–66, November 2004.

[48] O. Gustafsson. Towards optimal multiple constant multiplication: A hypergraph approach. In *The 42nd Asilomar Conference on Signals, Systems and Computers*, pages 1805–1809, 2008.

[49] N. Boullis and A. Tisserand. Some optimizations of hardware multiplication by constant matrices. *IEEE Transactions on Computers*, 54(10):1271–1282, October 2005.

[50] R. E. Blahut. *Fast Algorithms for Digital Signal Processing*. Cambridge University Press, June 2010.

[51] D. F. Coelho, S. Nimmalapalli, V. Dimitrov, Arjuna Madanayake, R. J. Cintra, and A. Tisserand. Computation of 2d 8×8 dct based on the loeffler factorization using algebraic integer encoding. *IEEE Transactions on Computers*, 2018.

[52] T. I. Haweel. A new square wave transform based on the dct. *Signal processing*, 81(11):2309–2319, 2001.

[53] K. Lengwehasatit and A. Ortega. Scalable variable complexity approximate forward dct. *IEEE Transactions on Circuits and Systems for Video Technology*, 14:1236–1248, 2004.

[54] S. Bouguezel, M. O. Ahmad, and MNS Swamy. Low-complexity $8\times$ 8 transform for image compression. *Electronics Letters*, 44(21):1249–1250, 2008.

[55] S. Bouguezel, M. O. Ahmad, and MNS Swamy. A fast $8\times$ 8 transform for image compression. In *Microelectronics (ICM), 2009 International Conference on*, pages 74–77. IEEE, 2009.

[56] S. Bouguezel, M. O. Ahmad, and MNS Swamy. A low-complexity parametric transform for image compression. In *2011 IEEE International Symposium on Circuits and Systems (ISCAS), 2011 IEEE International Symposium on*, pages 2145–2148. IEEE, 2011.

[57] R. J. Cintra and F. M. Bayer. A dct approximation for image compression. *IEEE Signal Processing Letters*, 18(10):579–582, 2011.