2018-04-12

Advanced Methods for Efficient Digital Signal Processing and Matrix-Based Computations

Gomes Coelho, Diego Felipe

UNIVERSITY OF CALGARY

Advanced Methods for Efficient Digital Signal Processing and Matrix-Based Computations

by

Diego Felipe Gomes Coelho

A THESIS

SUBMITTED TO THE FACULTY OF GRADUATE STUDIES

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE

DEGREE OF DOCTOR OF PHILOSOPHY

GRADUATE PROGRAM IN ELECTRICAL AND COMPUTER ENGINEERING

CALGARY, ALBERTA

APRIL, 2018

Abstract

Modern engineering and scientific problems demand a great amount of data processing power. The type of data that needs to be processed varies from application to application. Image processing, genome matching, physics phenomena simulation, and cryptography are a few examples of processing-power demanding applications. In a wide range of those computationally intensive applications, the arithmetic complexity plays an important role, having direct impact on the implementation performance. In this thesis, we present several methods that are novel contributions of the author to some computationally intensive problems. The introduced methods reduce the overall computing time or other relevant hardware' and software implementation metrics by decreasing the arithmetic complexity associated with each task. Verified results are shown with peer-reviewed journal papers in reputable journals. In particular, problems on signal processing, eigenvalue computation, and matrix inversion for radar image classification are considered.

Acknowledgements

The author thanks Dr. Vassil Dimitrov, his supervisor, for the guidance, thought provoking discussions and freedom for researching topics that led to the completion of the PhD. The author also thanks Dr. Vassil Dimitrov for his generous financial support in providing all the necessary resources for research, maintenance, and for facilitating the expansion of the candidate scientific network and connections for future developments.

The author is extremely thankful for the mentoring, suggestions, scientific inspiring discussions, and collaboration with Dr. Renato Cintra, his former supervisor and current collaborator. The author is also thankful for the scientific and technical training in his early career as researcher and engineer.

The author would like to thank his former professors during his early education as an undergrad student, in particular, Dr. Ricardo Campello and Dr. Hélio Magalhães. The author also thanks his collaborators, Dr. Fábio Bayer, Dr. Arjuna Madanayake, Mr. Viduneth Ariyarathna, Dr. Nilanka Rajapaksha, Mr. Sunera Kulasekera, Mr. Paulo Oliveira, Ms. Raíza Oliveira, Mr. Thiago Trugillo, and Dr. Alejandro Frery.

The author would like to say thank you to his parents, Mrs. Geny Gomes da Silva and Mr. Décio Coelho, and his siblings Eng. Gláuber Coelho and Ms. Alana Coelho.

The author also thanks the Natural Sciences and Engineering Research Council of Canada (NSERC) and the University of Calgary for their partial support.

Table of Contents

List of Tables

List of Figures and Illustrations

List of Symbols and Abbreviations

Acronyms and Symbols	Definition
DFT	Discrete Fourier Transform
DCT	Discrete Cosine Transform
DST	Discrete Sine Transform
FFT	Fast Fourier Transform
AI	Algebraic Integer
GE	Gauss-Eisenstein Integer
FRS	Final Reconstruction Step
MCM	Multiple Constant Multiplication
CRS	Compressed Row Storage
CCS	Compressed Column Storage
CSD	Canonical Signed Digit
VLSI	Very Large Scale of Integration
JPEG	Joint Picture Experts Group
MPEG	Motion Picture Experts Group
HEVC	High Efficiency Video Coding
SAR	Synthetic Aperture Array
PolSAR	Polarimetric SAR
GPU	Graphical Processing Unit
GPGPU	General Purpose GPU
$a, b, \ldots$	Scalar
$\mathbf{u}, \mathbf{v}, \ldots$	Vector
$\mathbf{A}, \mathbf{M}, \ldots$	Matrix

Chapter 1

Introduction

1.1 Early Signal Processing and Numerical Problems

The theory and practice of signal processing is the central element of engineering solutions for scientific and technological problems [1]. Signal processing techniques have been applied in many fields such as medicine [2, 3], chemistry [4, 5], physics [6, 7], telecommunication [8, 9, 10], and geosciences [11, 12, 13], to name a few. The early signal processing methods were purely based on continuous time techniques. The physical realization of continuous time signal processing methods was achieved by means of analog circuits built with basic elements, such as resistors, capacitors, inductors, and amplifiers. Often and again, the physical implementation of such analog circuits required a large number of components, making it difficult to ensure the correct operation and reproducibility. The physical characteristics of those basic elements, such as frequency response, were crucial for a successful implementation. Small deviations in the intrinsic characteristics of those basic elements could result in total failure in the operation of the final circuit. Furthermore, other non-controllable variables due to the environment such as noise and temperature could deeply affect the circuit output.

The discovery of the valve and the transistor was a milestone in physics and electrical engineering related disciplines, paving the way for the development of digital circuits, and later on, digital computers. The success that came of digital computers had an extensive impact on many scientific and technological fields, opening the way for the birth of new areas of knowledge such as Electronics Engineering, Computer Engineering, and Computer Science. It laid the groundwork for the proposition and development of different methods and devices for the solution of engineering and scientific problems that would make use of digital circuits and computer architectures [1, 14].

The success of digital circuits and computers in signal processing applications owes to the

advantages of digital circuits in opposition to analog circuits. Namely, we can mention: (i) reproducibility; (ii) modularity; (iii) flexibility; and (iv) simplicity.

The reproducibility comes from the fact that digital circuits and computers represent quantities with values that are determined by a physical state that is interpreted either as 'on' or 'off'. Those quantities are not allowed to have intermediate states and its value is therefore known and robust to fluctuations due to noise and temperature variations [1, 15]. Even in cases where the represented quantities show an error, it is possible to recover its true value through the use of error correcting codes [16].

The modularity and flexibility is the result of the easy assembly of digital circuits by employing several instantiations of components from a small set of different types of elements, such as flip-flops and logical gates.

The simplicity is possibly the most alluring characteristic of digital circuits in opposition to analog circuits. Because digital circuits are reproducible, modular, and flexible, they are easy to build. This is of great importance to industry because it allows companies to have their final product ready for the market in reduced time as compared to the amount of time that it would take to design, validate, and verify the equivalent product using analog circuits. This has several implications, such as competitive prices, scalability, easy maintenance, and reduced cost of operation.

However, there are limitations associated with digital circuits and computers. Because digital circuits and computers can represent only finite quantities, the signals of interest have to be quantized [1]. Not only that, the number of samples has to be finite. This implies that the signals of interest have to be represented both as finite and quantized sequences. This type of signals are labelled digital signals and the collection of all techniques devoted to analyze, understand, and process digital signals in all its different applications is called digital signal processing [1].

Digital signal processing techniques are currently applied in many areas such as biomedical image processing [17, 18], control systems [19], hyperspectral imaging [20], reservoir simulation [21, 22], and geophysics [12, 11]. Early hardware implementations of digital signal proces-

sing methods for engineering problems shared common problems. The main drawback faced by engineers were performance problems due to low processing speed of such devices [23, 14]. The low processing speed had direct impact on the performance of such systems, which often were not able to meet the computational complexity requirements for various applications.

Different factors such as memory access and latency can contribute to the computational complexity of a particular procedure [24]. Among all the factors, the arithmetic complexity is often of great importance. The arithmetic complexity of an algorithm can be defined as the amount of arithmetic operations (typically addition and multiplication) required to perform a particular computational task [14, 25].

Early applications of digital signal processing were highly influenced by the arithmetic complexity of the employed methods. In particular, multiplication and division operations were the most time and energy consuming among the four basic arithmetic operations [26, 14]. This is due to the fact that a multiplication and division operations require more hardware resources in order to be performed when compared to additions and subtractions. For example, consider the multiplication operation of an unknown x and the constant integer 45 in a usual hardware implementation. One can easily perform the multiplication $x \cdot 45$ as $x \cdot (32+8+4+1)$, which requires a total of three additions and three bit-shifting operations. A more efficient way to compute $x \cdot 45$ is $x \cdot (8+1) \cdot (4+1)$, which reduces both the number of additions and bit-shifting operations to two. In both cases, note that a multiplication requires a number of addition operations. As a consequence, the hardware implementation of a multiplication operation requires more resources, such as registers, adders, and energy in comparison to the implementation of a simple addition or subtraction. A similar phenomenon occurs with divisions operation, where several multiplications and additions are necessary to compute the ratio of two quantities [26]. Although the example given above is simple, the theory for finding optimal solutions for multiplication and division and many other arithmetic operations is still not completely understood and a very active topic [25, 26, 27, 28].

1.2 Modern Digital Signal Processing and Numerical Methods

In this scenario, the signal processing community, concerned with the performance of the digital signal processing applications, developed algorithms for particular computationally intensive tasks. These algorithms were designed aiming to reduce the total number of multiplications and divisions for particular tasks. The outcome of such well crafted algorithms were: (i) energy efficient hardware implementations; (ii) reduced computing time; and in some cases (iii) simpler hardware.

The advancements on designing and the implementation of digital circuits and devices made a broad impact on digital signal processing applications. Devices responsible for performing a large amount of arithmetic operations were highly improved by the use of fast algorithms. For example, modern central processing units (CPU) on personal computers can performs multiplications *nearly* as fast as addition operations [29, 30, 31, 32, 33], [34, p. 14-8]. This is due to the development of particular floating-point units (FPU), which are responsible for performing operations in floating-point quantities with highly optimized hardware and algorithms such as carry-look ahead and fused multiply-add [35, 36, 37]. A similar situation also occurs with general purpose graphical processing units (GPGPU), used for graphics rendering and computationally intensive applications [38].

Although the advancements on modern digital computers and algorithms have improved the performance of modules in charge of performing arithmetic operations, the arithmetic complexity continues to be a challenge for engineers in a wide range of applications nowadays. This is due to the advent of consumer electronics devices, such as mobile devices, wearable devices, home automation devices, and communication systems. These devices are capable of generating data or are required to process it in a gigantic amount, which demands heavy signal processing tasks with high dependence on its arithmetic complexity.

Consider one example of a computationally intensive method in radar image processing. One of the most prominent methods for image classification of polarimetric synthetic aperture radar

(PolSAR) data is based on the modelling of noise using the complex Wishart distribution [39]. The image classification requires the computation of statistics defined through a hypothesis test [40]. The statistics are calculated using the PolSAR images, where each pixel is a square matrix, whose size is associated with the imaging device and method [41, 42]. Those statistics require the computation of several inversions and determinants of those matrices representing the pixels of the PolSAR image. A typical *small* PolSAR image is roughly 10^3 wide by 10^3 high [43], where each pixel is a 3×3 matrix [44]. For a single image, one needs over a million matrix inversions and determinant computations, in the simplest possible case. Considering that thousands of PolSAR images are taken in just one flight over an area, billions of matrix inversions and determinant operations are performed for just a single area that one wants to classify.

Reducing the arithmetic complexity has several advantages. It does not allow one only to provide a particular procedure output in a reduced time, but it also reduces the associated computational error [14]. In some cases, the resulting hardware implementation is simpler than the current methods. This is of practical importance, because reducing the number of components reduces the chances of malfunctioning and failure. Furthermore, a reduction in the total number of operations reduces the amount of energy required for the execution of a procedure. This is of practical importance for *eco-friendly* or *green* devices for the solution of computationally intensive problems.

1.3 Research Motivation and Challenges

The research community in signal processing and numerical methods have contributed to the state-of-the-art algorithms and methods for reducing processing time of common computationally intensive tasks, including the computation of discrete Fourier transform (DFT) [14], discrete cosine transform (DCT) [45], eigenvalue and eigenvector estimation, and matrix inversion [46].

The demand for intensive computations of the above-mentioned common tasks makes the efficient implementation of the computation of DFT, DCT, eigenvalue and eigenvector estimation, and

matrix inversion of utmost importance. Aligned with the aforementioned problems, three research challenges are addressed in this thesis.

1.3.1 Challenge I: High Accuracy Implementation of Numerical Methods

Due to the internal structure, digital computers can represent only finite quantities. Generally, numerical methods require quantities that are not exactly represented in usual fixed or floating point format. This is the case with irrational quantities required by discrete-time transforms, such as DFT and DCT, e.g., $\cos(\pi/4) = \sqrt{2}/2$ is present in both the DCT and DFT for sequences of size eight [14]. In fact, this is true for any non dyadic integer quantity, such as a simple $1/3$ [14].

Besides the inherent error in representing quantities that are not dyadic integers in digital computers, the successive computations in a given task can generate or amplify numerical errors [1]. This often occurs in matrix computations such as functions of matrices, norm calculations, matrix multiplication and eigenvalue decomposition [47]. This is particularly true for ill-conditioned problems that are common in areas such as geophysics [48, 13] and hyperspectral imaging [49, 50]. The first challenge of this thesis is to propose algorithms with high accuracy for the computationally intensive procedures.

1.3.2 Challenge II: Fast Execution of Computationally Intensive Methods

Because of the large amount of data and information that is generated nowadays, it is of utmost importance to be able to process it efficiently. This is particularly important for radar image processing [39] and matrix computation problems [51].

The proposition of efficient fast algorithms is not a trivial task, as it requires knowledge about complexity theory, algorithm analysis, and implementation techniques. The second challenge of this thesis is the proposition of algorithms for reducing the computing time of numerically intensive tasks.

1.3.3 Challenge III: Energy Efficient Implementation of Numerical Methods

The increased demand for computationally intensive methods requires a considerable amount of energy for its execution. This is because on average, the proposition of algorithms that reduce the execution time, requires some additional operations or the computation of temporary variables or quantities that demand more memory and energy. The merging of algorithms and methods for fast execution of numerically intensive procedures that save energy is not trivial. One can find some remarkable examples, for instance, the radix-2 Cooley-Tukey fast algorithm for the DFT computation of sequence with sizes power-of-two is a relevant example showcasing that it is possible to improve computing speed and reduce energy consumption. The third challenge of this thesis is the proposition of algorithms that execute not only a particular task in a fast paced manner, but also saves energy along the way.

1.4 Objectives and Contributions

In order to address the above-mentioned challenges, this thesis aims to investigate and propose efficient algorithms and methods for numerically intensive problems in digital signal processing and matrix-based computations, including the (i) DFT computation; (ii) DCT computation; (iii) eigenvalue computation; (iv) matrix inversion and determinant computation that meets the aforementioned challenges. The main contributions of this thesis are:

Contribution I: A non-conventional numerical representation for the computation of the DFT based on algebraic integer theory. In this contribution it is shown that the proposed representation can exactly represent the unity-of-roots required by the DFT of sequences of particular sizes. The fast algorithms are proposed and shown to address Challenge I and II. This contribution was published as:

- DFT Computation using Gauss-Eisenstein Basis: FFT Algorithms and VLSI Architectures, **D. F. G. Coelho**, R. J. Cintra, N. Raja-

paksha, G. J. Mendis, A. Madanayake, V. S. Dimitrov - Transactions on Computers, IEEE, Vol 66, No. 8, pp. 1442–1448, Aug 2017.

Dr. V. S. Dimitrov and Dr. R. J. Cintra contributed with invaluable comments and suggestions. Dr. R. J. Cintra also gave directions on the algorithm development, error analysis, and manuscript writing. Dr. A. Madanayake, Mr. N. Rajapaksha, and Mr. G. J. Mendis implemented the hardware design. I proposed the introduced representation, designed the fast algorithms, investigated the error due to the proposed encoding, implemented matlab simulations and verification regarding all the steps for evaluating the algorithm performance and effectiveness. I also wrote the manuscript.

Contribution II: A fast algorithm for the computation of the DCT based on algebraic integers that address Challenge I and II. It is shown that the proposed algorithm for the DCT is able to compute the components of the DCT without error up to the final stage. This drastically reduces the numerical error incurred in the subsequent computations. This contribution was published as:

- Error-Free Computation of 8-point DCT Based on the Loeffler Factorization and Algebraic Integers, **D. F. G. Coelho**, R. J. Cintra, S. Kulasekera, A. Madanayake, V. S. Dimitrov - Signal Processing, IET, Vol 10, No. 6, pp. 633–640, July 2016.

Dr. V. S. Dimitrov and Dr. R. J. Cintra contributed with invaluable comments and suggestions. Dr. R. J. Cintra also gave directions on the algorithm design, encoding, decoding, and manuscript writing. Dr. A. Madanayake and Mr. S. Kulasekera implemented the hardware design. I proposed the new numerical representation, showing its suitability to compute the DCT. I designed the fast algorithms and implemented matlab simulations and verifications regarding encoding, decoding, and

algorithm performance. I also wrote the manuscript.

Contribution III: A fast algorithm for the DCT computation based on summation-by-parts formula, the discrete counter part of integration-by-parts formula. The algorithm is shown to reduce the overall arithmetic complexity of the DCT computation, reaching the theoretical lower bound for the number of multiplications. The algorithm also outperforms state-of-the-art algorithms when the input signal has some specific characteristics. This contribution addresses Challenge I and III and it was published as:

- Efficient Computation of the 8-point DCT via Summation by Parts, **D. F. G. Coelho**, R. J. Cintra, V. S. Dimitrov - Journal of Signal Processing Systems, Springer, pp. 1–10, Aug 2017.

Dr. V. S. Dimitrov made important comments suggesting potential applications of the proposed algorithm. Dr. R. J. Cintra gave directions on the algorithm design, complexity reduction, application contexts, and manuscript writing. I proposed the new fast algorithm based on the summation-by-parts formula, previously formalized by Dr. R. J. Cintra in a past work of his authorship, which is cited in the paper and chapter. I analyzed the proposed algorithm complexity and compared to state-of-the-art methods. I also wrote the manuscript and made matlab implementation and verification.

Contribution IV: A method for the fast estimation of the largest eigenvalue and its corresponding eigenvector of tridiagonal matrices. The method is based on the computation of the square of the input matrix. A fast algorithm for the matrix squaring for tridiagonal matrices is proposed. It is shown that not only the computation is faster than the usual methods, but it also improves accuracy by reducing the number of arithmetic operations that can strongly contribute to numerical errors. This contribution

addresses the Challenge I, II, and III. It was published as the following papers:

- Efficient Computation of Tridiagonal Matrices Largest Eigenvalue, **D. F. G. Coelho**, V. Dimitrov, L. Rakai - Journal of Computational and Applied Mathematics, Elsevier, Vol 330, pp. 268–275, Mar 2018;
- Fast estimation of tridiagonal matrices eigenvalue, **D. F. G. Coelho**, V. Dimitrov - IEEE 30th Canadian Conference of Electrical and Computer Engineering (CCECE), Windsor, ON, Canada, pp. 1–4, Apr 2017.

Dr. V. S. Dimitrov contributed with comments and suggestions. Dr. L. Rakai gave suggestions on the writing and comments about the implementation. I proposed the new fast algorithm for eigenvalue estimation, investigated and analyzed the numerical error, implemented the proposed method and its competitor, and wrote the manuscript.

Contribution V: A method for computing matrix inversion and the determinant computation for matrices of particular size in the context of polarimetric synthetic aperture radar (PolSAR). The method is shown to reduce the overall arithmetic complexity compared to the usual approach based on Cholesky factorization. A data parallel software implementation using GPGPU is provided. This contribution addresses the Challenge II and III. It was submitted as:

- Fast Matrix Inversion and Determinant Computation for Polarimetric Synthetic Aperture Radar, **D. F. G. Coelho**, R. J. Cintra, A. C. Frery, V. Dimitrov - Computers & Geosciences, Elsevier, Jan 2018.

Dr. V. S. Dimitrov contributed with insightful suggestions about the algorithm design. Dr. A. Frery contributed with comments and contextualization. Dr. R. J. Cintra gave suggestions on the writing and comments about the implementation. I proposed the new fast algorithm for matrix inversion of the particular nature treated in the paper, investigated and analyzed the arithmetic complexity, implemented the proposed method and its competitor, and wrote the manuscript.

1.5 Organization

This thesis has six chapters. It is organized as follows.

Chapter 2 presents Contribution I. The proposed algebraic integer based numerical representation is introduced. The irrational quantities required by the DFT for some specific sequence sizes are encoded into the proposed representation and shown to have an error-free representation. An architecture for the DFT computation based on the proposed numerical representation is introduced and compared to previous works.

Chapter 3 shows the work associated with Contribution II using algebraic integers for a new numerical representation. The numerical representation is introduced and combined with an efficient algorithm for the proposition of a scheme for the computation of the DCT. The resulting architecture is implemented and compared to state-of-the-art methods using algebraic integers.

Chapter 4 describes the Contribution III using summation-by-parts formula for the efficient computation of the DCT. The method proposed in Chapter 4 is shown to reduce the number of multiplications required by the 8-point DCT computation to the theoretical lower bound [52]. Moreover, the proposed method outperforms state-of-the-art algorithms in terms of multiplications and additions for computing the DCT when the input signal has some particular characteristics found in signal processing applications.

Chapter 5 presents Contribution IV for the fast and accurate computation of the largest eigenvalue and its corresponding eigenvector for tridiagonal matrices. The method is shown to outperform

classical methods in terms of execution time and accuracy.

Chapter 6 presents the Contribution V for the fast matrix inversion and determinant computation. The method is based on a fast algorithm tailored for the specific matrix sizes that occur in the context of polarimetric synthetic aperture radar (PolSAR). This chapter provides arithmetic analysis of the proposed algorithm and compares it with the usual approach based on Cholesky factorization. It also presents data parallel software implementation for GPGPU.

Chapter 7 presents conclusions and research directions for future work. The Appendix A has a list of publications, together with additional works that the candidate has produced during his participation in the PhD program, but that are not included as chapters of this thesis. The Appendix B has a list of awards received by the candidate during his participation in the PhD program.

Chapter 2

DFT Computation using Gauss-Eisenstein Basis: FFT Algorithms and VLSI Architectures

2.1 Introduction

Fixed-point number representations are often employed in digital signal processing (DSP) architectures [53, 54, 14]. Nevertheless, such representations may not be the ideal approach when non-rational quantities are required to be represented [55]. Indeed, several quantities in common mathematical methods are not perfectly represented in usual finite binary representation. For instance, the roots of unity, which are employed in the discrete Fourier transform (DFT) computation, are not necessarily rational numbers [14]. Representing such quantities in fixed-point requires the adoption of compromise solutions involving truncation and/or rounding-off operations [56, 57]. Such approximations systematically introduce errors, which may propagate throughout a given computational architecture [57]. This fact results in an error floor, decreasing the system output signal-to-noise ratio [56].

Algebraic integer (AI) encoding technique provides a means to address this problem [57, 58]. Introduced by Cozzens and Finkelstein [59, 56], AI encoding consists of mapping possibly irrational numbers into integer vectors that can be processed in an error-free arithmetic framework. Indeed, literature presents several AI-based architectures. For instance, we may cite applications in: DSP systems based on Eisenstein residue number systems (RNS) [55]; row-parallel 8×8 2-D DCT architectures [57]; real orthogonal transform implementation using RNS [60]; and VLSI architectures for the 4- and 6-tap 2-D Daubechies wavelet filters [61].

Among the DSP methods, the DFT occupies a central position. The design of efficient DFT methods—collectively know as fast Fourier transforms (FFTs)—has constantly attracted research

community efforts [1]. Divide-and-conquer techniques, such as the Cooley-Tukey algorithm, constitute a particularly representative class of algorithms [14, 1]. However, the computational gains offered by this type of algorithm are highly dependent on the efficiency of small blocklength transforms [14], such as the 3-, 5-, 6-, and 7-point FFT algorithms [62, 63]. In [64], a collection of Winograd small FFT were unified in a single architecture; considered blocklenghts included 3, 4, 5, and 7. Such transformations find application in the 3780-point DFT required in the Chinese digital TV standard, which is based on orthogonal frequency-division multiplexing [64, 65, 66, 9, 10].

The 3-point DFT was also considered in the implementation of the 3GPP Long Term Evolution phone system [67]. The 6-point DFT was applied in optical communication applications in [68]. The 3- and 6-point DFT are required for the implementation of the 12-point DFT [69], which was previously considered in several contexts: (i) speech signal processing [70]; (ii) the hardware simplification and reduction of number of registers for the Chinese digital TV [71], (iii) the design and evaluation performance of 2.5 giga-sample per second receiver-on-a-chip [72], and (iv) electric power systems [73]. In [58], Dubois and Venetsanopoulos showed that representing the DFT coefficients over the lattice graph implied by elements 1 and $\omega = (-1 + j\sqrt{3})/2$ could be a venue for the efficient computation of a multiplication-free radix-3 FFT. Prakash and Rao proposed a fast algorithm for the 6-point DFT [74]. In [62], Suzuki *et al.* introduced a suit of algorithms for the 3-, 6-, and 12-point DFT. Based on Prakash-Rao design [74], White proposed a fast algorithm for 12-point DFT aiming at low power consumption [63].

The goal of this work is two-fold. First, we advance a new mixed-basis algebraic integer number representation [75] based on the Gaussian integers and Eisenstein integers [76]. We refer to the new structure as the Gauss-Eisenstein (G-E) number representation. We investigate the mathematical properties of the proposed structure, which aims at both error-free representation and computation. Fast algorithms for elementary arithmetic operations over G-E integers are also sought. Second, we examine the G-E number representation as a suitable framework for the low-complexity computation of the 3-, 6-, and 12-point DFT.

This chapter is structured as follows. In Section 2.2, the G-E number representation and structure are detailed and algorithms for coding and decoding G-E integers are proposed. In Section 2.3, fast algorithms based on G-E integers are introduced for the computation of 3-, 6-, and 12-point DFT. In Section 3.4, the final reconstruction step (FRS) block, which decodes the G-E representation into usual complex numbers, is detailed. Three different FRS architectures are presented and compared. Section 6.4 presents hardware implementations of the introduced methods. Conclusions and final remarks are drawn in Section 3.6.

2.2 Gauss-Eisenstein Representation

2.2.1 Mathematical Preliminaries

Definition 1 *A Gaussian integer is a complex number $a+bj$, where a and b are usual integers [76].*

Definition 2 *A Gauss-Eisenstein integer ζ is defined as $\zeta = z_1 + z_2\omega$, where z_1 and z_2 are Gaussian integers and $\omega = (-1+j\sqrt{3})/2$ is a primitive cube root of unity [76].*

Considering $z_1 = a+bj$ and $z_2 = c+dj$, the above definitions implies the following representation:

$$\zeta = a+bj+c\omega+dj\omega,$$

where a, b, c, and d are usual integers. The set of G-E integers can be interpreted as an algebraic extension of $\mathbb{Z}$ by adjoining the elements j and ω [77, p. 285]. We refer to the set of G-E integers as $\mathbb{Z}[j,\omega]$. Notice that the set $\{1, j, \omega, j\omega\}$ forms a basis for $\mathbb{Z}[j,\omega]$ over $\mathbb{Z}$.

We aim at representing complex numbers as a 4-tuple of integers characterized by a G-E integer. Before addressing the problem of converting a complex number into a G-E integer, we note that any complex number can be mapped into a G-E integer with arbitrary precision. This can be demonstrated by adapting the proof of Proposition 3 of [59]. In the next two sub-sections, we address the problem of converting a G-E integer into a usual complex number and vice-versa.

2.2.2 Decoding

Mapping a given G-E integer into its associate complex number can be directly accomplished according to:

$$\zeta = \left(a - \frac{c}{2} - \frac{d}{2}\sqrt{3}\right) + j\left(b + \frac{c}{2}\sqrt{3} - \frac{d}{2}\right).$$

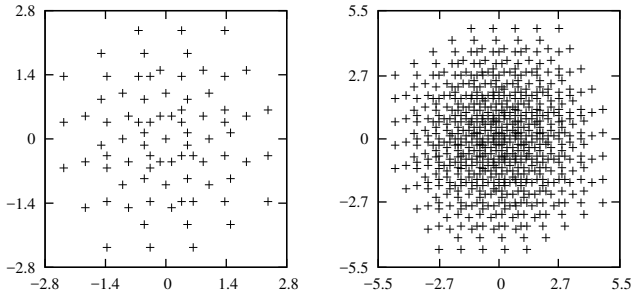
We refer to this operation as G-E decoding. The decoding operation requires the quantity $\sqrt{3}$, which cannot be represented with infinite precision in usual computer arithmetic, such as fixed- or floating-point arithmetic. Thus, the accuracy of the decoding operation depends on how precise $\sqrt{3}$ is represented. Consequently, the user has complete control of the decoding accuracy based on the numerical representation of a single element. The decoding operation is also referred to as the final reconstruction step and is given a comprehensive analysis in Section 3.4.

2.2.3 Encoding

Although a complex number can be represented as a G-E integer with arbitrary precision, in practical terms, the integer elements a, b, c and d are bounded by the maximum value of the considered computer arithmetic. Therefore, let us consider G-E integers a, b, c and d such that their magnitudes are no greater than M, where M is a positive integer. This restriction implies a subset of $\mathbb{Z}[j,\omega]$ containing $(2M+1)^4$ elements described as: $\mathbb{Z}[j,\omega]_M = \{a+bj+c\omega+dj\omega : |a|, |b|, |c|, \text{ and } |d| \leq M\}$. Fig. 2.1 depicts the scatter plots associated to $\mathbb{Z}[j,\omega]_1$ and $\mathbb{Z}[j,\omega]_2$. Notice that the number of elements in $\mathbb{Z}[j,\omega]_M$ dramatically increases as the value of M becomes larger.

The elements of $\mathbb{Z}[j,\omega]_M$ are complex numbers that can be given an error-free G-E representation. For example, it can be found by inspection that the complex number $z = 1+\sqrt{3}j$ is precisely mapped into $\mathbb{Z}[j,\omega]_2$ as $2+0j+2\omega+0j\omega$.

Let z be the complex number that can be ideally mapped into the G-E integer ζ. Now we aim at deriving an algorithm for the encoding operation over $\mathbb{Z}[j,\omega]_M$. For such, we propose the following procedure:



(a) $M = 1$ (b) $M = 2$

Figure 2.1: Representable points of $\mathbb{Z}[j,\omega]_M$ for $M = \{1,2\}$.

1. Approximate z to the nearest Gaussian integer by means of the rounding-off function: $\bar{z} = \text{round}(\Re(z)) + j \cdot \text{round}(\Im(z))$, where $\Re(\cdot)$ and $\Im(\cdot)$ represent the real and imaginary parts of its complex argument, respectively. This corresponds to the following G-E integer: $\bar{\zeta} = \bar{z} + 0\,\omega + 0\,j\omega$. The initial approximation $\bar{z}$ has an error given by $\bar{e} = z - \bar{z}$, whose magnitude is bounded by $\sqrt{2}/2$;

2. Find the G-E integer $\hat{\varepsilon}$ that best approximates the complex number $\bar{e}$ for a given prescribed M. This approximation can be obtained from a lookup table that stores pre-calculated G-E integers over $\mathbb{Z}[j,\omega]_M$ with magnitude less than $\sqrt{2}/2$;

3. Compute $\hat{\zeta} = \bar{\zeta} + \hat{\varepsilon}$, which is the final encoded G-E. The encoding error is given by $\varepsilon = \zeta - \hat{\zeta}$.

Fig. 2.2 illustrates the relationships among the above-mentioned quantities over the Cartesian coordinate system. To illustrate the proposed procedure, we adopted $M = 32$ and encoded some roots of the unity as shown in Table 2.1.

Input signals are often derived from the analog-to-digital conversion of real physical phenomena [1]. Thus, the input data is already of the form $z = m + rj$, $m, r \in \mathbb{Z}$, which can be trivially encoded as a G-E integer according to:

$$\zeta = m + rj + 0\omega + 0j\omega. \tag{2.1}$$

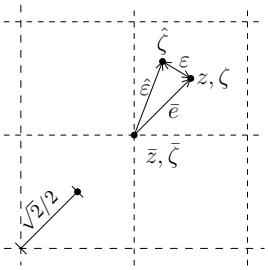


Figure 2.2: Encoding scheme. Squares have unitary size.

Table 2.1: G-E representation of the roots of unity

Root	a	b	c	d	Error
$e^{-\frac{2\pi j}{1}}$	1	0	0	0	0
$e^{-\frac{2\pi j}{2}}$	-1	0	0	0	0
$e^{-\frac{2\pi j}{3}}$	-1	0	-1	0	0
$e^{-\frac{2\pi j}{4}}$	0	-1	0	0	0
$e^{-\frac{2\pi j}{5}}$	9	26	-19	21	$5.6 \cdot 10^{-3}$
$e^{-\frac{2\pi j}{6}}$	0	0	-1	0	0
$e^{-\frac{2\pi j}{7}}$	-16	-30	17	-29	$9.7 \cdot 10^{-3}$
$e^{-\frac{2\pi j}{9}}$	-10	22	-25	2	$8.1 \cdot 10^{-3}$
$e^{-\frac{2\pi j}{12}}$	0	-1	0	-1	0

2.3 DFT over G-E Integers

The DFT maps an N-point input signal $x[n]$, $n = 0, 1, \ldots, N-1$, into another N-point signal $X[k]$, $k = 0, 1, \ldots, N-1$, according to the following expression:

$$X[k] = \sum_{n=0}^{N-1} x[n] \cdot W_N^{kn}, \tag{2.2}$$

where $W_N = e^{\frac{-2\pi j}{N}}$ is the Nth root of unity.

For $N = 3, 6, 12$, the proposed G-E encoding can represent W_N exactly (cf. Table 2.1), namely $W_3 = -1 - \omega$, $W_6 = -\omega$, and $W_{12} = -j - \omega j$. Although the multiplication of two arbitrary G-E requires at least six multiplications [78, pg. 170, 174], the trivial encoding of the roots of unity suggests the suitability of the G-E representation for the proposal of multiplierless DFT algorithms for $N = 3, 6, 12$. Hereafter, we adopted the general case where the input signal is a sequence of Gaussian integers. Such data can be trivially encoded into G-E representation, as shown in (2.1). As a particular case, we also aim at addressing input sequences of usual integers.

2.3.1 3-point DFT

Let $x[n] = a_n + b_n j + 0\omega + 0\, j\omega$ be the G-E encoded input signal of a quantized complex sequence. For $N = 3$, we have that $W_3 = -1 + \omega$, $W_3^2 = \omega$, and $W_3^4 = W_3$. Thus, from (2.2), we obtain:

$$\begin{aligned}
X[0] &= a_0 + a_1 + a_2 + (b_0 + b_1 + b_2) j + 0\omega + 0\, j\omega, \\
X[1] &= a_0 - a_1 + (b_0 - b_1) j + (a_2 - a_1)\omega + (b_2 - b_1) j\omega, \\
X[2] &= a_0 - a_2 + (b_0 - b_2) j + (a_1 - a_2)\omega + (b_1 - b_2) j\omega.
\end{aligned}$$

Therefore, over the G-E representation, the 3-point DFT could be computed without any multiplication. In Fig. 2.3, we propose a fast algorithm for such operation which demands 10 additions for complex integer input and 5 additions for real integer input. Fig. 2.4 depicts the associated signal flow graph for the proposed algorithm.

Input: $x[n] = a_n + b_n j$, $a_n, b_n \in \mathbb{Z}$, for $n = 0,1,2$
Output: $X[k] = A_k + B_k j + C_k \omega + D_k j\omega$, $A_k, B_k, C_k, D_k \in \mathbb{Z}$, for $n,k = 0,1,2$

Auxiliary additions: $s_0 = a_2 - a_1$; $s_1 = b_2 - a_1$

Compute each component:
$A_0 = a_0 + a_1 + a_2$; $B_0 = b_0 + b_1 + b_2$; $C_0 = D_0 = 0$
$A_1 = a_0 - a_1$; $B_1 = b_0 - b_1$; $C_1 = s_0$; $D_1 = s_1$
$A_2 = a_0 - a_2$; $B_2 = b_0 - b_2$; $C_2 = -s_0$; $D_2 = -s_1$

Figure 2.3: G-E based 3-point FFT for complex input.

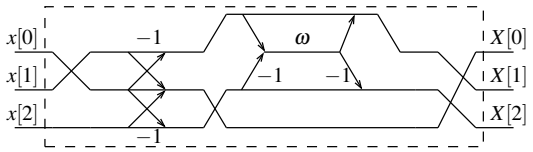


Figure 2.4: Proposed 3-point G-E FFT signal flow graph.

2.3.2 6- and 12-point DFT

Considering the prime factor algorithm (PFA) [14], the calculation of the 6- and 12-point DFT can be performed by means of multiple instantiations of the 3-point DFT [14, 1]. For that purpose, we present the 2- and 4-point DFT computation over the G-E representation in Fig. 2.5, which require 4 and 16 real additions for complex input; and 2 and 8 additions for purely integer input. The associate algorithms for the 6- and 12-point DFT are shown in Fig. 2.6 and 2.7, respectively.

2.3.3 Final Reconstruction Step

The FRS performs two key operations: (i) mapping the G-E DFT 4-tuple output $X[k] \in \mathbb{Z}[j,\omega]_M$ to $X_{\text{dec}}[k] \in \mathbb{C}$ and (ii) computing multiplication by the constant $\sqrt{3}/2$ present in the G-E basis

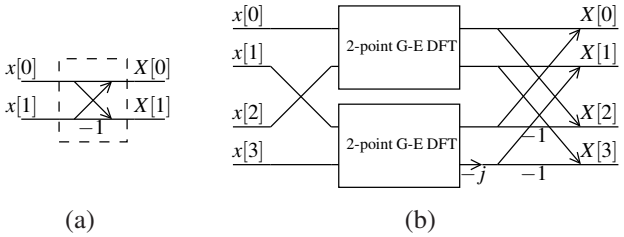


Figure 2.5: SFG for the (a) 2- and (b) 4-point G-E DFT.

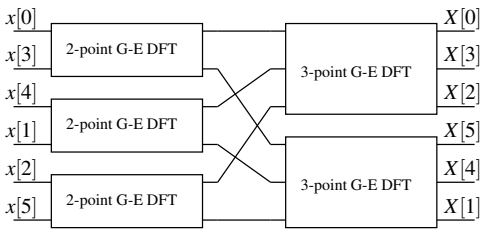


Figure 2.6: Proposed 6-point G-E FFT signal flow graph.

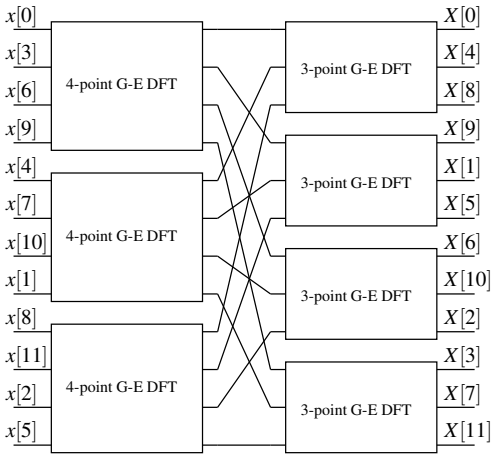


Figure 2.7: Proposed 12-point G-E FFT signal flow graph.

element ω. Because all discussed G-E algorithms rely on the fundamental 3-point DFT block, the FRS is derived for this particular block only.

In Fig. 2.8, we propose a G-E decoding algorithm for the 3-point DFT core. The decoding of $X[0]$ is effortless; only $X[1]$ and $X[2]$ are submitted to actual computation. Notice also that the 3-point decoder is not a collection of three separate G-E decoders. Interrelated and intermediate computations are taken into consideration to minimize the overall arithmetic cost, which consists of two multiplications by a real constant; six real additions; and two bit-shifting operations. The arithmetic cost of the 6- and 12-point DFT FRS are, respectively, two and four times the cost of the 3-point DFT FRS.

2.3.4 Arithmetic Complexity Assessment

Being an algebraic integer method, the G-E computation is capable of error-free computation up to the FRS. In particular, the proposed 3-, 6-, and 12-point DFT algorithms over the G-E representa-

Input: $X[0] = A_0 + B_0 j$, $X[1] = A_1 + B_1 j + C_1 \omega + D_1 j\omega$,
$X[2] = A_2 + B_2 j - C_1 \omega - D_1 j\omega$
Output: $X_{\text{dec}}[k] \in \mathbb{C}$, for $k = 0, 1, 2$.

Auxiliary quantities:
$t_0 = \frac{\sqrt{3}}{2} C_1$; $t_1 = \frac{\sqrt{3}}{2} D_1$; $s_0 = -C_1/2 - t_1$; $s_1 = t_0 - D_1/2$;

Compute each component:
$X_{\text{dec}}[0] = A_0 + B_0 j$
$X_{\text{dec}}[1] = (A_1 + s_0) + (B_1 + s_1) j$
$X_{\text{dec}}[2] = (A_2 - s_0) + (B_2 - s_1) j$

Figure 2.8: FRS for G-E 3-point FFT output for complex input.

tion are also capable of multiplierless computation up to the FRS. Table 2.2 compares the proposed schemes in terms of the required number of real multiplications and additions with several competing designs for the case of complex input signal. Operation counts are reported according to each computing stage: (i) input encoding, (ii) computation core, and (iii) output decoding (FRS, if AI encoded). Total counts are included. To expand the scope of our comparisons, we extended the algorithm by Qureshi *et al.* [64]—originally proposed for the 3-point DFT only, to the 6- and 12-point DFT. For such extension, we employed the PFA. Similarly, we also extended the DFT algorithm proposed in [58] The arithmetic complexity for these algorithms are shown in Table 2.2 in *italic*. The cells with "–" indicate that the approach described in the respective reference cannot be extended to higher sizes or are not provided by the authors. For instance, the method in [74] and [58] cannot be extended to 12-point sequences within an error-free paradigm. The work in [63] concentrates on 12-point sequences and does not provide algorithms for 3- or 6-point sequences. Notice that even considering the computation at the FRS for 6- and 12-point the multiplicative complexities of the proposed method are at least 50–73% smaller than the best error-free algorithm previously reported in literature [62, 63]. For the 3-point DFT, the proposed algorithm achieves the multiplicative complexity of the best scheme in literature. Although the design in [64] demands fewer addition operations, its 3-point DFT and its extensions for 6- and 12-point sequences are not error-free. This is a trade-off between complexity and accuracy, whose choice depend on the ap-

Table 2.2: Arithmetic complexity assessment of the 3-, 6-, and 12-point DFT algorithms for complex input case

Method		$N=3$		$N=6$		$N=12$	
		Mult.	Add.	Mult.	Add.	Mult.	Add.
Dubois *et al.* [58]	Enc.	3	3	*6*	*6*	–	–
	Core	0	14	*0*	*42*	–	–
	Dec.	3	3	*6*	*6*	–	–
	Total	6	20	*12*	*54*	–	–
Prakash *et al.* [74]	Enc.	–	–	6	6	–	–
	Core	–	–	0	42	–	–
	Dec.	–	–	6	6	–	–
	Total	–	–	12	54	–	–
Suzuki *et al.* [62]	Enc.	–	–	–	–	–	–
	Core	4	12	8	36	36	104
	Dec.	–	–	–	–	–	–
	Total	4	12	8	36	36	104
White *et al.* [63]	Enc.	–	–	–	–	12	12
	Core	–	–	–	–	6	106
	Dec.	–	–	–	–	12	12
	Total	–	–	–	–	30	130
Qureshi *et al.* [64]	Enc.	–	–	–	–	–	–
	Core	2	6	*4*	*24*	*8*	*72*
	Dec.	–	–	–	–	–	–
	Total	2	6	*4*	*24*	*8*	*72*
Proposed	Enc.	0	0	0	0	0	0
	Core	0	10	0	32	0	88
	FRS	2	6	4	16	8	24
	Total	2	16	4	48	8	112

plication. In Section 6.4 we show that the proposed G-E encoding scheme leads to improvements in the signal-to-noise ratio.

2.4 FRS Designs

In this section, we propose three different designs for the FRS. Each design is based on (i) the Dempster-McLeod minimum adder representation [79]; (ii) the expansion factor method [45, 80] using the approach detailed in [81]; and (iii) the addition aware quantization proposed in [82].

Table 2.3: Fast algorithms for integer multiplication

m	$y = m \cdot x$
222	$v_1 = (2^3 - 1) \cdot x$ $y = 2 \cdot (2^4 \cdot v_1 - 1)$
3547	$v_1 = (2^9 - 1) \cdot x$ $v_2 = 2 \cdot (2^4 - 1) \cdot x$ $y = (2^3 - 1) \cdot v_1 - v_2$
1774	$v_1 = (2^7 + 1) \cdot x$ $v_2 = (2^7 - 1) \cdot x$ $y = 2^3 \cdot v_2 - v_1$
7094	$v_1 = 2 \cdot (2^9 - 1) \cdot x$ $v_2 = 2^2 \cdot (2^4 - 1) \cdot x$ $y = (2^3 - 1) \cdot v_1 - v_2$

2.4.1 Dempster-McLeod Method

The Dempster-McLeod representation for constant integer multiplication is an improvement over canonical signed digit (CSD) encoding [79]. Its main goal is the binary representation of numerical quantities with a small number of adders/subtractors, rendering them suitable for integer multiplication. Considering 8- and 12-bit wordlengths, the quantity $\sqrt{3}/2$ (cf. Algorithm in Fig. 2.8) possesses the following dyadic approximation, respectively: $\frac{\sqrt{3}}{2} \approx \frac{222}{2^8}$ (8-bit) and $\frac{\sqrt{3}}{2} \approx \frac{3547}{2^{12}}$ (12-bit). The absolute errors implied by the above approximations are less than 10^{-2} and 10^{-4}, respectively. The required powers of two can be implemented effortlessly by means of bit-shifting operations. Thus, we focus on the integer multiplications by 222 and 3547, which can be computed efficiently according to a sequence of additions and bit-shifting operations, as suggested in [79]. As a consequence, we have the algorithms shown in Table 2.3, requiring 2/3 and 4/4 addition/bit-shifting operations, respectively. In both cases, an extra bit-shifting is required for the power-of-two denominator of the proposed dyadic rationals.

2.4.2 Expansion Factor Method

Another approach for the FRS design is based on the expansion factor method [80, 45]. Capable of furnishing a scaled spectrum, this method has been previously considered for deriving discrete

transform approximations [83, 57] and for designing FRS blocks associated to wavelet analysis architectures based on algebraic integers [81].

Let $\boldsymbol{\zeta} = \begin{bmatrix} 1 & j & \omega & j\omega \end{bmatrix}^\top$ be the G-E basis vector. In this context, a expansion factor is the real number $\alpha^* > 1$ that satisfies the minimization problem

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

where $\|\cdot\|$ is the Euclidean norm. Obtaining a closed-form solution for above optimization problem is non-trivial task due to analytical intractability of the round-off function [84, 85]. However, exhaustive search methods can provide the sought solution [86]. Adopting the search space $\alpha \in (1, 128]$ and a numerical precision of 10^{-4}, we obtain $\alpha^* = 112.0022$. Therefore, we have that: $\alpha^* \cdot \begin{bmatrix} 1 & j & \omega & j\omega \end{bmatrix} \approx \begin{bmatrix} 112 & 112j & -56+97j & -97-56j \end{bmatrix}$. The quantity α^* scales the required basis elements to integer values which can be given simple dyadic rational representations. Thus, the approximate dyadic rational basis effects an scaled output. In the following, we consider the inputs to the algorithm in Fig. 2.8 and the CSD representation. For the decoding of $X[0]$, the term $(A_0 + B_0 j)$ must be scaled by 112, which demands 2/4 additions/bit-shifting operations. Additionally, we have that: $X[1] = (A_1 + B_1 j) + (C_1 + D_1 j)\omega$ and $X[2] = (A_2 + B_2 j) - (C_1 + D_1 j)\omega$, where $\alpha^* \omega \approx (-56 + 97j)$. Thus, the terms $(A_1 + B_1 j)$ and $(A_2 + B_2 j)$ are also scaled by 112, requiring 4/8 additions/bit-shifting operations. Decoding $X[1]$ and $X[2]$ requires the product $(C_1 + D_1 j) \cdot (-56 + 97j)$ which demands 6/8 additions/bit-shifting operations. Therefore, a total of 12/20 additions/bit-shifting operations are needed.

2.4.3 Addition Aware Quantization Method

A third approach for the FRS design is based on the addition aware quantization proposed in [82]. The addition aware quantization optimally identifies the binary representation with the minimum number of additions as the number of fractional bits varies [82]. We applied the method to $\sqrt{3}$ then we performed a bit-shift to obtain $\sqrt{3}/2$. Considering the 10- and 12-bit wordlengths, the addition aware quantization furnishes the following representations: $\sqrt{3} \approx \frac{1774}{2^{10}}$ (10-bit) and $\sqrt{3} \approx \frac{7094}{2^{12}}$

Table 2.4: Complete arithmetic complexity comparison for the FFT including the FRS

Method	$N=3$		$N=6$		$N=12$	
	Add.	Shifts	Add.	Shifts	Add.	Shifts
Dempster-McLeod (8 bits)	20	10	56	20	128	40
Dempster-McLeod (12 bits)	24	12	64	24	144	48
Expansion Factor	28	20	62	40	160	80
Addition Aware (10 bits)	22	10	60	20	136	40
Addition Aware (12 bits)	24	14	64	28	144	56

(12-bit), with associated errors magnitude of 3.7×10^{-4} and 1.2×10^{-4}. The fast algorithms for multiplication by these quantities are shown in Table 2.3.

2.5 Hardware Implementation and Results

Complete arithmetic complexity for designs with considered FRS methods are shown in Table 2.4 for 3-, 6-, and 12-point DFTs. The 12-point DFT suits as a comprehensive case for the proposed architecture requiring all introduced design aspects. Thus, we selected it for digital hardware implementation. The 12-point DFT was physically implemented using a Xilinx Virtex-6 XC6VLX240T field programmable gate array (FPGA) device. The digital design consists of (i) FRS sections based on the 12-bit Dempster-McLeod approach, (ii) the expansion factor method with $\alpha^* = 112.0022$, and (iii) the 12-bit addition aware quantization methods. Input signals to the 12-point DFT FPGA realizations were chosen as real valued, which allowed simple G-E encoding operation (cf. (2.1)). The wordlengths at each point in the SFGs have full precision starting with wordlength L at the input ports. The constant integer multiplications in the FRS implementations are realized using hardwired bit-shift and addition operations. The implemented digital designs were fine-grain pipelined to achieve the maximum speed of operation. The process of introducing pipeline stages invariably leads to additional chip area (for integrated circuits) and additional programmable logic resources (for FPGAs). The 12-point Winograd DFT was implemented for comparison at 14- and 16-bit constant multiplication precision.

We selected these wordlengths in the constant multiplication because smaller wordlengths would provide a higher level or error, making it unfavorable to the Winograd algorithm. That is, the G-E algorithm employs a smaller wordlentgh when compared to the Winograd FFT for the same error. This design requires 96 adders and 16 multipliers.

2.5.1 FPGA implementation

The DFT architectures were tested on the FPGA device using stepped hardware co-simulation in an ML605 evaluation platform. The on-chip tests assumed input wordlength of $L = 8$ bits. Table 2.5 shows the consumption of programmable logic resources in the XC6VLX240T device for all proposed algorithm designs and for the Winograd designs with 14 bits and 16 bits multiplication constant precision. Overall hardware cost is higher in the Winograd architecture when compared with the proposed methods. The expansion factor FRS method showed the highest overall hardware cost in proposed methods. The maximum speed of operation for the expansion factor FRS, Dempster-McLeod FRS, and addition aware FRS proposed designs; and Winograd with 14- and 16-bit multiplication precision were 302, 254, 257, 258, and 240 MHz, respectively. The corresponding throughputs are: 3.62×10^9, 3.05×10^9, 3.08×10^9, 3.10×10^9, and 2.88×10^9 coefficients/s.

In Table 2.7, the proposed method with G-E algorithm and expansion factor FRS is compared with FPGA implementations of FFT methods available in the literature. Throughput per area for FPGA implementations is calculated as the ratio of the operating frequency in MHz per the number of occupied slices. According to such figure of merit the proposed method G-E algorithm and expansion factor FRS outperforms the design detailed in [87, 88, 8], but not the one in [89]. However, the proposed method requires no multipliers, eliminating the need for digital signal processing elements (DSP48E on Virtex-6). On the other hand, the methods in [87], [88], and [89] require 16, 128, and 72 DSP48E slices.

Table 2.5: Resource consumption of Xilinx FPGA for the FFT including the FRS

Method	Slices	Slice Regs.	Slice LUTs	IOBs
Dempster-McLeod	831 (2%)	2164 (1%)	2904 (1%)	553 (76%)
Expansion factor	853 (2%)	2924 (1%)	2746 (1%)	689 (95%)
Addition aware	833 (2%)	2156 (1%)	2905 (1%)	553 (76%)
Winograd (14 bits)	1224 (3%)	2409 (1%)	3681 (2%)	721 (100%)
Winograd (16 bits)	1254 (3%)	2421 (1%)	3988 (2%)	737 (102%)

2.5.2 ASIC synthesis and results

The designs were synthesized for application specific integrated circuits (ASIC) using the Cadence RTL compiler version v09.10-s242-1 for 0.18μm CMOS technology. The Austria Micro Systems (AMS) standard-cell library hitkit version 4.11 was employed in synthesis with optimization goal set to maximize the clock speed. The ASIC implementation was performed for an operating voltage of 1.8 V up to synthesis in order compare performance of proposed architectures and Winograd architecture. The architecture showing better performance was then separated for full ASIC implementation with place and route. The maximum speed of operation obtained in synthesis for input wordlength of 8 bits, for the expansion factor FRS, Dempster-McLeod FRS, and addition aware FRS proposed designs and Winograd design at 14- and 16-bit precision were 1.23, 1.20, 1.19, 0.90, and 0.85 GHz, respectively. The corresponding throughputs are: 14.76×10^9, 14.40×10^9, 14.28×10^9, 10.80×10^9, and 10.20×10^9coefficients/s. Table 2.6 shows the average percentage error (Er), area (A), dynamic power consumption (D_p), critical path delay (T), area-time complexity (AT^m), $m = 1, 2$, and the metric $M = AT^2 \cdot Er \cdot D_p$ for all considered architectures. The metric M is an indicator of accuracy and hardware cost; being lower values desirable. The average percentage errors were computed relative to the Matlab output. Results show that the proposed DFT design with the expansion factor FRS operates with higher throughput than the DFT design with the Dempster-McLeod FRS, addition aware FRS, or Winograd designs, in both FPGA implementation and ASIC synthesis. Considering the metric M, the proposed 12-point DFT architecture with expansion factor FRS outperformed the 14- and 16-bit Winograd architectures for

$L = 8$; whereas for $L = 16$, the 16-bit Winograd architecture performed better with the trade-off of reduced throughput.

The proposed 12-point DFT architecture with the FRS based on the expansion factor method was submitted to 0.18μm ASIC implementation with place and route using AMS Encounter digital implementation libraries. The maximum speed of operation for implemented architecture with input wordlength of 8 bits is 505 MHz with real-time throughput of 6.06×10^9 coefficients/s. The summary of the ASIC implementation is shown in Table 2.7 along with other ASIC/FPGA implementations of FFT/DFT algorithms for comparisons purposes. In Table 2.7, normalized metrics are used in order to achieve a fair comparison with ASIC implementation with different FFT sizes. The normalized 2nd-order area-time complexity (AT^2_{norm}) for the FFT size and technology and is given by

$$AT^2_{norm} = \frac{A \cdot T^2}{N \cdot (s/0.18)^2},$$

where s represents the semiconductor processing technology in μm and N the FFT sequence size.

From the architectures proposed in Tsai *et al.* [87], the 64-point FFT architecture is the one showing the highest signal to quantization error (SQNR) and thus is chosen for comparison. For Kuo *et al.* [90] variable length FFT architecture, we choose the highest FFT points related parameters for comparison since the reported core area is fixed. From FFT architectures in Saponara *et al.* [8], we choose the architecture optimized for maximum throughput and from architectures in Saenz *et al.* [88] we choose 16-point FFT architecture with parallel input and output for comparison.

The hardware cost metric $AT^2_{norm} \cdot D_p$ shows a lower cost to proposed method compared to other ASIC methods, except for Tsai *et al.* [87] FFT implementation. However, the proposed method has a higher maximum frequency metric and a better signal to quantization error (SQNR) as well, which exceed the SQNR in Tsai *et al.* [87] in about 30 dB.

Table 2.6: Average percentage magnitude error and resource consumption in ASIC hardware for the FFT including the FRS

Method	Word-length (L)	Avg. percentage error (Er)	Dynamic power (D_p, mW)	Area (A, μm^2)	Critical delay (T, ns)	AT ($\mu m^2 \cdot ns$)	AT^2 ($\mu m^2 \cdot ns^2$)	$M = AT^2 \times Er \times D_p$ ($\mu m^2 \cdot ns^2 \cdot mW$)
Expansion factor	8	4.266×10^{-3}	409	591349	0.81	478993	387984	676952
	16	1.784×10^{-4}	685	1057336	0.88	930456	818801	100060
Dempster McLeod	8	6.172×10^{-3}	493	738063	0.83	612592	508452	1547114
	16	2.273×10^{-3}	823	1265482	0.88	1113624	979989	1833245
Addition aware	8	6.215×10^{-3}	529	790599	0.84	664103	557847	1834052
	16	2.265×10^{-3}	869	1331069	0.89	1184651	1054340	2075241
Winograd (14-bit)	8	5.717×10^{-3}	290	497773	1.11	552528	613306	1016819
	16	2.268×10^{-3}	462	828383	1.21	1002343	1212836	1270828
Winograd (16-bit)	8	5.196×10^{-3}	289	524997	1.18	619496	731006	1097711
	16	1.246×10^{-4}	478	863570	1.30	1122641	1459433	86922

Table 2.7: Comparison summary of hardware utilization for the FFT including the FRS

	Tsai [87]	Kuo [90]	Maharatna [91]	Tran [92]	Saenz [88]	Mamatha[89]	Saponara [8]	**Proposed**
FFT size	64	64-2048	64	64	16	28	128	**12**
Technology	$0.09\mu m$ / Vertex 4	$0.35\mu m$	$0.25\mu m$	$0.18\mu m$	Vertex 6	Vertex 5	Vertex 4	**$0.18\mu m$ / Vertex 6**
FFT algorithm	Radix-2/4/8	Cached FFT Architecture	2D Structure of 8-point FFTs	Radix-2	Radix-2	Cyclic Convolution	Radix-4 (Fully parallel)	**Proposed G-E**
SQNR (dB)	55	40 (with AWGN)	NA	NA	NA	NA	27.2	**85**
Max. frequency (MHz)	413 / 65	80	26	NA	NA	224.9	NA	**505 / 302**
Op. freq. (MHz)	413 / 40	80	20	50	25	224.9	55	**505 / 302**
Area (A, mm^2)	0.408	6.76	6.8	0.34	NA	NA	NA	**0.74**
Power (D_p, mW)	147	126-574	41	21.43	NA	NA	NA	**192**
AT^2_{norm} $(mm^2 \cdot ns^2)$	0.149	0.136	137.7	2.125	NA	NA	NA	**0.242**
$AT^2_{norm} \cdot D_p$ $(mm^2 \cdot ns^2 \cdot mW)$	21.98	78.30	5645.70	45.54	NA	NA	NA	**46.42**
No. of occupied slices	7549	NA	NA	NA	2048	1102	20000	**2924**
MHz/No. of slice	0.0053	NA	NA	NA	0.0122	0.2040	0.0027	**0.1033**
No.of DSP48Es	16	NA	NA	NA	128	72	NA	**0**

2.6 Chapter Summary

In this chapter, a new representation for complex numbers was introduced as well as a suit of fast algorithms for the 3-, 6-, and 12-point DFT computation. The new proposed algorithms are error-free and multiplierless up to the FRS. The arithmetic complexity of the proposed methods was favorably compared with several competing algorithms. The 12-point G-E DFT algorithm was physically realized using a Xilinx Virtex-6 XC6VLX240T FPGA device at a real-time clock frequency of 254 MHz for FRS sections based on the Dempster-McLeod technique, 257 MHz for addition aware method, and 305 MHz for the expansion factor method at $\alpha^* = 112.0022$, respectively.

The 12-point G-E DFT hardware was mapped, placed and routed to $0.18\mu m$ CMOS technology employing Austria Micro Systems (AMS) standard-cell library (hitkit version 4.11) with place and route at 1.8 V DC operating voltage, using Cadence Encounter RTL tools, resulting in maximum clock rate of 505 MHz for expansion factor based FRS. The proposed 12-point DFT architecture with expansion factor FRS outperforms the Winograd-based scheme for 14- and 16-bit multiplier constant precision and 8-bit input wordlength.

Chapter 3

Error-Free Computation of 8-point DCT Based on the Loeffler Factorization and Algebraic Integers

3.1 Introduction

Discrete transforms plays a major role in the signal and image processing theory and applications. In particular, the Karhunen-Loève transform (KLT) offers optimal decorrelation properties when applied to Markov-I random signals [45]. The KLT is also optimal in the mean square error (MSE) sense for representing truncated data. Nevertheless, the KLT matrix depends on the input signal, which limits its practical application. Introduced by Ahmed *et al.*, the discrete cosine transform (DCT) is asymptotically equivalent to the the KLT [93] and is capable of closely approximating the KLT—even for small blocklengths [45]. Because of this, the DCT had been frequently employed in image processing techniques, such as image compression [94], noise reduction [95], and watermarking methods [96, 97]. In particular, the 8-point DCT is adopted in several image and video compression standards [98], including JPEG [99], MPEG-1 [100], H.264 [101], and HEVC [102].

Numerous efficient algorithms for the 8-point DCT computation have been proposed [45]. A particularly successful method is the Loeffler DCT algorithm [103] which is capable of computing the 8-point at minimal multiplicative cost [104, 105, 103] and is often regarded as a reference method for comparison and analysis of 8-point DCT algorithms. In [59, 56], Cozzens and Finkelstein proposed a method based on algebraic integer (AI) representation for the computation of the 2^n-point discrete Fourier transform. Algebraic integer representation has the distinction of being capable of exact computation, numerical calculation without error propagation, and arbitrary precision [57]. Besides the DFT, AI methods have been applied for the DCT computation [106, 107, 108, 109]. Recently, Shen *et al.* [110] introduced a two-dimensional algebraic

encoding representation for the Loeffler [103]. Shen's design could not attain an error-free architecture, due to the partial encoding of selected rotational blocks required by the Loeffler DCT. As a consequence, the resulting design requires a number of multiplications.

The goal of this chapter is the introduction of a *fully* error-free AI-based scheme for the 8-point DCT computation by means of the Loeffler algorithm. Moreover, we aim at a multiplication-free architecture, where only simple additions and bit-shifting operations are necessary. For such, a suitable AI-based representation for the DCT multipliers is sought. Resulting algorithms are designed in a multiplication-free framework. We also aim at proposing decoding architectures for mapping AI encoded quantities back to usual fixed point arithmetic. For resource comparison purposes, the proposed algorithm and the Shen's algorithm [110], have been realized in hardware along with proposed decoding architectures.

This chapter unfolds as follows. Section 3.2 details the mathematical description of the introduced algebraic integer representation. Section 3.3 describes the application of the proposed AI scheme for the multiplierless computation of the Loeffler DCT. Section 3.4 addresses practical methods for AI decoding by means of the final reconstruction step. Section 3.5 presents VLSI implementations of the proposed schemes with comparisons. Section 3.6 concludes the chapter.

3.2 Algebraic Integer Representation

3.2.1 Mathematical Preliminaries

An algebraic integer is any root of a monic polynomial with integer coefficients [76, p. 178]. In [59, 56], Cozzens and Finkelstein proposed the use of a ring of algebraic integers for numerical representation. Such representation was based on the subring of the field of complex numbers generated by the primitive root of the polynomial $\omega^{R/2}+1=0$, where R is a power of two. The implied subring is denoted by $\mathbb{Z}[\omega]$, where $\omega = e^{2\pi j/R}$, and the set $\{1,\omega,\omega^2,\ldots,\omega^{R/2-1}\}$ forms an integral basis for $\mathbb{Z}[\omega]$ [59, 111].

3.2.2 8-point DCT AI Basis

The 8-point DCT is a linear orthogonal transformation given by [45, 94]:

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

where $\beta_0 = 1/\sqrt{2}$ and $\beta_k = 1$, for $k = 1,2,\ldots,7$. Our goal is to characterize a ring suitable for the computation of the 8-point DCT computation. Since the 8-point DCT requires the quantities $\cos[\pi k(2n+1)/16]$, $n,k = 0,1,\ldots,N-1$ [45], then, for $R = 32$, the irreducible polynomial $\omega^{16}+1$ furnishes the necessary algebraic structure. Thus, we have the ring $\mathbb{Z}\left[e^{j\pi/16}\right]$, which spans the following subset of complex numbers:

$$\mathbb{Z}\left[e^{j\pi/16}\right] = \left\{ z \in \mathbb{C} : z = \sum_{k=0}^{15} a_k \cdot \left(e^{j\frac{\pi}{16}}\right)^k, \quad \text{for } a_k \in \mathbb{Z} \right\}.$$

However, because the DCT kernel is real, we focus on the purely real elements of $\mathbb{Z}\left[e^{j\pi/16}\right]$. Hence, we impose that $\Im\{z\} = \Im\left\{\sum_{k=0}^{15} a_k \cdot \sin(k\pi/16)\right\} = 0$, where $\Im\{\cdot\}$ denotes the imaginary part operator. This condition is equivalent to $a_k + a_{16-k} = 0$ [112, p. 72]. Therefore, because $\cos(k\pi/16) = \cos((N-k)\pi/16)$, the purely real elements of $\mathbb{Z}\left[e^{j\pi/16}\right]$ can be represented as:

$$x = a_0 + 2\cdot\sum_{k=1}^{7} a_k \cdot \cos\left(k\frac{\pi}{16}\right). \tag{3.1}$$

Thus, a set that spans x over $\mathbb{Z}$ is

$$\mathbf{Z} = \{1, c_1, c_2, c_3, c_4, c_5, c_6, c_7\},$$

where $c_k = 2\cos(k\pi/16)$, for $k = 1,2,\ldots,7$. Notice that $\text{span}(\mathbf{Z}) \subset \mathbb{Z}\left[e^{j\pi/16}\right]$, where $\text{span}(\cdot)$ is the vector space generated by linear combination of the elements of $\mathbf{Z}$.

3.2.3 Encoding and Decoding

The AI encoding of a given real number x over a particular basis is denoted by:

$$f_{\text{enc}}(x;\boldsymbol{\zeta}) = \mathbf{x},$$

where $\boldsymbol{\zeta}$ is a vector with the basis elements, $\mathbf{x} = \begin{bmatrix} a_0 & a_1 & a_2 & a_3 & a_4 & a_5 & a_6 & a_7 \end{bmatrix}^\top$ is a vector of integers representing the encoding of x over the AI domain, and $^\top$ denotes transposition. For the discussed set $\mathbf{Z}$, we have

$$\boldsymbol{\zeta} = \begin{bmatrix} 1 & c_1 & c_2 & c_3 & c_4 & c_5 & c_6 & c_7 \end{bmatrix}^\top. \tag{3.2}$$

The decoding operation is furnished by the dot product operation [57]:

$$f_{\text{dec}}(\mathbf{x}; \boldsymbol{\zeta}) = \mathbf{x}^\top \cdot \boldsymbol{\zeta} = a_0 + \sum_{i=1}^{7} a_k \cdot c_k = \hat{x}, \tag{3.3}$$

In the Appendix 3.7.1, we show that the above presentation is dense and can provide arbitrary precision, i.e., it is always possible to determine an integer vector $\mathbf{x}$ such that $|x - \hat{x}| < \varepsilon$, for any $\varepsilon > 0$. For example, the real number $x = 1 + \sqrt{3}$ can be represented by $\begin{bmatrix} 1 & -2 & 1 & -1 & 2 & 1 & 2 & 0 \end{bmatrix}^\top$ or $\begin{bmatrix} 3 & 0 & 1 & 1 & -3 & -2 & 3 & 1 \end{bmatrix}^\top$, considering representation errors of 7.07×10^{-5} and 7.10×10^{-6}, respectively. However, some particular real numbers are well-suited for the proposed basis, allowing error-free representation: $|x - \hat{x}| = 0$. For instance, the real number $x = 1 - \sqrt{2 + \sqrt{2 + \sqrt{2}}}$ has the following error-free representation over the discussed basis:

$$f_{\text{enc}}(x; \boldsymbol{\zeta}) = \begin{bmatrix} 1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}^\top.$$

In fact, by construction, the basis elements and their combinations can be given error-free integer representation. Table 3.1 displays the proposed encoding of the basis elements.

In actual applications, the input data is discrete and usually quantized [113] in the form of an integer or a dyadic rational [45]. Therefore, input data can be understood as a sequence of integers [14]. In this particular but realistic case, the proposed encoding can be trivially performed. Indeed, any integer quantity $m \in \mathbb{Z}$ can be represented by [57]:

$$f_{\text{enc}}(m; \boldsymbol{\zeta}) = \begin{bmatrix} m & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}^\top. \tag{3.4}$$

Thus, under above assumptions, the encoding operation is effortless. In the current work, only trivial encoding is required.

Table 3.1: Encoding of AI basis elements

x	$f_{\text{enc}}(x;\boldsymbol{\zeta})$
1	$[1\ 0\ 0\ 0\ 0\ 0\ 0\ 0]^\top$
c_1	$[0\ 1\ 0\ 0\ 0\ 0\ 0\ 0]^\top$
c_2	$[0\ 0\ 1\ 0\ 0\ 0\ 0\ 0]^\top$
c_3	$[0\ 0\ 0\ 1\ 0\ 0\ 0\ 0]^\top$
c_4	$[0\ 0\ 0\ 0\ 1\ 0\ 0\ 0]^\top$
c_5	$[0\ 0\ 0\ 0\ 0\ 1\ 0\ 0]^\top$
c_6	$[0\ 0\ 0\ 0\ 0\ 0\ 1\ 0]^\top$
c_7	$[0\ 0\ 0\ 0\ 0\ 0\ 0\ 1]^\top$

3.2.4 Arithmetic Operations

In this subsection, we discuss the basic arithmetic operations over the proposed AI representation. Only addition/subtraction and multiplication are analyzed here, which are the only elementary operations required for the DCT evaluation via the Loeffler DCT algorithm. Because the AI representation consists of an array of coefficients linked to a particular basis, the addition/subtraction operations follow usual vector sum rule. Let $\mathbf{u} = \begin{bmatrix} u_0 & u_1 & u_2 & u_3 & u_4 & u_5 & u_6 & u_7 \end{bmatrix}^\top$ and $\mathbf{v} = \begin{bmatrix} v_0 & v_1 & v_2 & v_3 & v_4 & v_5 & v_6 & v_7 \end{bmatrix}^\top$ be two AI encoded numbers. Thus, $\mathbf{w} = \begin{bmatrix} w_0 & w_1 & w_2 & w_3 & w_4 & w_5 & w_6 & w_7 \end{bmatrix}^\top = \mathbf{u} + \mathbf{v}$ satisfies $w_k = u_k + v_k$, where $k = 0, 1, \ldots, 7$.

Considering the multiplication operation, we restrict our analysis to operations involving encoded basis elements (cf. Table 3.1). In fact, only this type of multiplication is present in the Loeffler algorithm. Thus, considering (3.1) and trigonometric identities [112, p. 72], the results shown in Table 3.2 are established. Notice that all required multiplications are trivial in the sense that only element permutations and additions are required. In terms of hardware realization, these operations can be implemented by wiring and adder blocks.

For quantized real input, the multiplication can be trivially computed. Indeed, the multiplication between an integer m and a basis element is simply the basis element itself scaled by m. Thus,

Table 3.2: Quantities required by Loeffler algorithm for 8-point DCT and their respective products by an arbitrary algebraic integer

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

for instance, we have that $f_{\text{enc}}(m\cdot c_1;\boldsymbol{\zeta}) = m\cdot f_{\text{enc}}(c_1;\boldsymbol{\zeta}) = m\cdot \begin{bmatrix}0 & 1 & 0 & 0 & 0 & 0 & 0 & 0\end{bmatrix}^\top = \begin{bmatrix}0 & m & 0 & 0 & 0 & 0 & 0 & 0\end{bmatrix}^\top$.

3.2.5 Representable Interval

Practical considerations require a limited dynamic range for the coefficients a_k, $k = 1,2,\ldots,7$. Therefore, representable numbers are limited to the elements defined in the following subset:

$$\left\{ x\in\mathbb{R} : x = a_0 + \sum_{k=1}^{7} a_k\cdot c_k, \quad |a_k|\le M \right\},$$

where M is the largest representable integer. In Appendix 3.7.2, we show that the largest representable number in the proposed AI representation is:

$$x_{\max} = \frac{\sin(\frac{15\pi}{32})}{\sin(\frac{\pi}{32})}\cdot M \approx 10.2\cdot M.$$

Thus, representable numbers are confined to the interval $[-x_{\max}, x_{\max}]$.

3.3 AI-based 8-point Loeffler DCT

3.3.1 Loeffler DCT Multiplicands

The original Loeffler DCT algorithm has the 4-stage signal flow graph (SFG) shown in Figure 3.1 [103]. An initial inspection reveals that the Loeffler DCT requires the following six

multiplicands: $\{c_1, \sqrt{2}c_2, c_3, c_4, c_5, \sqrt{2}c_6, c_7\}$. Most of these constants are basis elements of the discussed AI representation as shown in Table 3.1. After cascading the multiplicands through the SFG, a more detailed analysis shows that only six multiplicands are required by the Loeffler algorithm: $c_4 \cdot c_2$, $c_4 \cdot c_6$, $c_4 \cdot c_3$, $c_4 \cdot c_5$, $c_4 \cdot c_1$, and $c_4 \cdot c_7$. Notice that $c_4 = \sqrt{2}$.

Either by considering the multiplication rules in Table 3.2 or by employing standard trigonometric rules, we maintain that $c_i \cdot c_k = c_{i+k} + c_{i-k}$, for any $i, k \in \mathbb{Z}$ and $c_i = -c_{16-i}$, for $i = 8, 9, \ldots, 16$. Therefore, the above six multiplicands can be written as additive combinations of the AI basis:

$$\begin{aligned} c_4 \cdot c_2 &= c_2 + c_6, & c_4 \cdot c_6 &= c_2 - c_6, \\ c_4 \cdot c_3 &= c_1 + c_7, & c_4 \cdot c_5 &= c_1 - c_7, \\ c_4 \cdot c_1 &= c_3 + c_5, & c_4 \cdot c_7 &= c_3 - c_5. \end{aligned} \tag{3.5}$$

For instance, we obtain:

$$f_{\text{enc}}(c_4 \cdot c_2; \boldsymbol{\zeta}) = \begin{bmatrix} 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 \end{bmatrix}^\top,$$
$$f_{\text{enc}}(c_4 \cdot c_6; \boldsymbol{\zeta}) = \begin{bmatrix} 0 & 0 & 1 & 0 & 0 & 0 & -1 & 0 \end{bmatrix}^\top.$$

Thus, all necessary quantities for the Loeffler DCT computation possess simple, multiplierless representations over the introduced representation.

The set of equations (3.5) reveals another relevant fact. When written as as sums of basis elements the Loeffler multiplicands do not require the basis element c_4. This means that the ring implied by the Cozzens-Finkelstein formalism results in an overcomplete basis with more elements than necessary for the DCT computation. Thus, we propose the following reduced AI basis for the Loeffler DCT:

$$\mathbf{Z}' = \{1, c_1, c_2, c_3, c_5, c_6, c_7\} \subset \mathbf{Z}.$$

For the sake of clarity, we maintain the 8-point representation. However, considering actual implementations only the seven relevant components should be taken into account.

Additionally, the original Loeffler DCT algorithm contains rotation blocks (butterfly sections with multiplications) that require four multiplications via direct computation. This cost can be

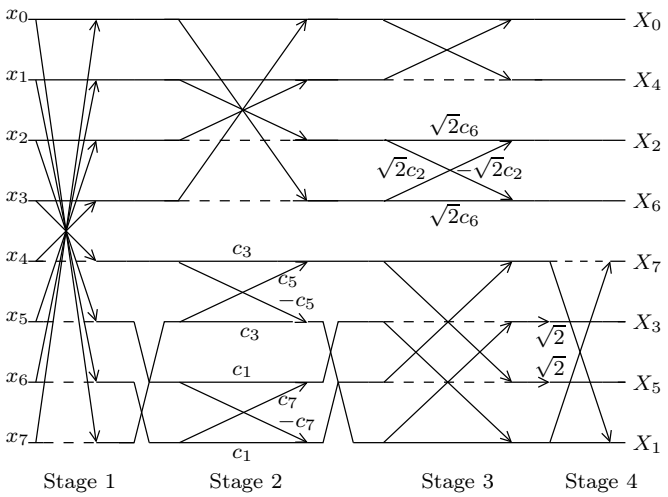


Figure 3.1: Loeffler algorithm for the 8-point DCT computation, where dashed lines represent product by -1.

reduced to three multiplications as shown in [14, 103]. Despite the fact that such multiplication saving scheme is particularly successful for the computation over usual arithmetic, this is not the case for the proposed AI-based representation. Indeed, all multiplications in the rotation blocks are trivial over the AI-based encoding. Therefore, we compute the rotation blocks directly.

3.3.2 Real Quantized Input Data

Considering real quantized input, the AI-encoded data can be represented according to $\begin{bmatrix} x_i & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}^\top$, $x_i \in \mathbb{Z}$, $i = 0, 1, \ldots, 7$. Therefore, all three rotation blocks in the AI-based Loeffler DCT have their input data encoded in the format described in (3.4). In this case, the multiplication rules can be dramatically simplified. Considering the input data $\begin{bmatrix} u_0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}^\top$ and $\begin{bmatrix} v_0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}^\top$, Figure 3.2(a) depicts the resulting operations. For the multiplication by $\sqrt{2}$, the input data is always in the format $\begin{bmatrix} 0 & u_1 & 0 & u_3 & 0 & u_5 & 0 & u_7 \end{bmatrix}^\top$, for $u_1, u_3, u_5, u_7 \in \mathbb{Z}$. Figure 3.2 details the required multiplication rules.

The application of the proposed encoding scheme into the 8-point Loeffler DCT furnishes the multiplierless algorithm shown in Figure 3.3. The total arithmetic cost of the proposed algorithm consists of 20 additions. In fact, the algorithm output is a scaled DCT spectrum with scaling factor

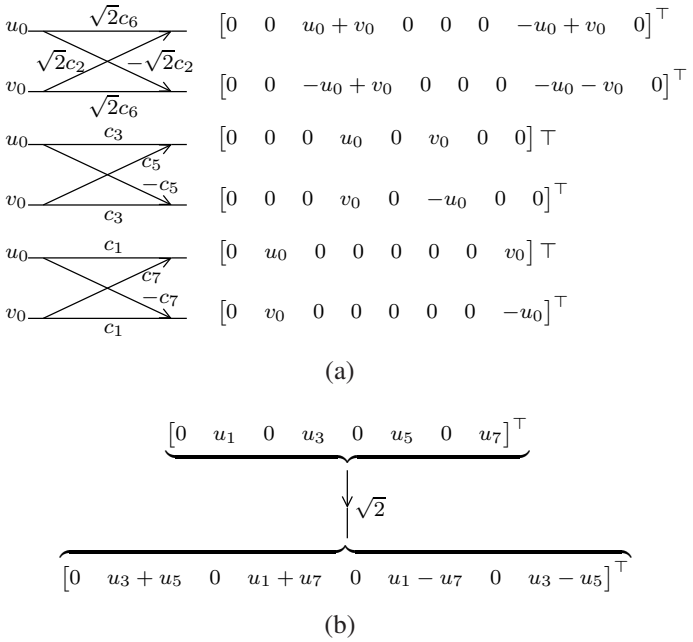


Figure 3.2: Multiplication rules required by the Loeffler DCT over AI: (a) rotation block computation and (b) multiplication by $\sqrt{2}$.

Table 3.3: Multiplicative complexity comparison for 8-point DCT computation

Algorithm	Precision	Multiplications		Additions
		Real	Integer	
Loeffler [103]	Finite	11	0	29
Shen *et al.* [110]	Finite	0	3	46
Proposed	Arbitrary	0	0	20

of 2. If necessary, the scaling can be removed by simple bit-shifting or it can be embedded in the decoding stage. Thus, such scaling does not contribute to any increase of the computational cost.

Table 3.3 compares the proposed method with the original Loeffler algorithm [103] and Shen *et al.* [110] in terms of precision and arithmetic cost over the AI representation.

3.3.3 Other AI-based Representation Schemes

Table 3.4 summarizes a comparison among competing AI-based encoding schemes related to the DCT computation. The proposed scheme has the distinction of providing canonical representation to all elements required by the DCT computation (cf. Table 3.1). Thus, over the introduced

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:

$$\begin{array}{ll} A_0 = x_0 + x_7 & A_4 = x_3 - x_4 \\ A_1 = x_1 + x_6 & A_5 = x_2 - x_5 \\ A_2 = x_2 + x_5 & A_6 = x_1 - x_6 \\ A_3 = x_3 + x_4 & A_7 = x_0 - x_7 \end{array}$$

Additions in Stage 2:

$$\begin{array}{ll} B_0 = A_0 + A_3 & B_2 = A_1 - A_2 \\ B_1 = A_1 + A_2 & B_3 = A_0 - A_3 \end{array}$$

Additions in Stage 3:

$$\begin{array}{ll} C_0 = B_0 + B_1 & C_2 = B_2 + B_3 \\ C_1 = B_0 - B_1 & C_3 = B_2 - B_3 \end{array}$$

Additions in Stage 4:

$$\begin{array}{ll} D_0 = -A_5 + A_6 & D_2 = A_4 + A_7 \\ D_1 = A_4 - A_7 & D_3 = -A_5 - A_6 \end{array}$$

Output:

$$\begin{array}{lcccccccc} \mathbf{X}_0 = [2C_0 & 0 & 0 & 0 & 0 & 0 & 0 & 0]^\top \\ \mathbf{X}_1 = [0 & -D_3 & 0 & D_2 & 0 & -D_1 & 0 & D_0]^\top \\ \mathbf{X}_2 = [0 & 0 & C_2 & 0 & 0 & 0 & -C_3 & 0]^\top \\ \mathbf{X}_3 = [0 & -D_1 & 0 & D_3 & 0 & D_0 & 0 & D_2]^\top \\ \mathbf{X}_4 = [2C_1 & 0 & 0 & 0 & 0 & 0 & 0 & 0]^\top \\ \mathbf{X}_5 = [0 & D_2 & 0 & -D_0 & 0 & D_3 & 0 & D_1]^\top \\ \mathbf{X}_6 = [0 & 0 & -C_3 & 0 & 0 & 0 & -C_2 & 0]^\top \\ \mathbf{X}_7 = [0 & -D_0 & 0 & -D_1 & 0 & -D_2 & 0 & -D_3]^\top \end{array}$$

Figure 3.3: The 8-point DCT algorithm for a real quantized input sequence.

representation the DCT can be computed with trivial multiplications by $\{0, \pm 1\}$.

3.4 Final Reconstruction Step

The AI decoding is mathematically described in (3.3) and it is performed at the final reconstruction step (FRS) block, which maps the AI represented quantities into usual fixed point representation. In this section, we consider two methods for AI decoding: (i) the canonical-signed-digit (CSD) method [116] and (ii) the expansion factor method [80, 45]. The CSD method is suitable for exact spectrum computation, whereas the expansion factor method grants a scaled spectrum.

Table 3.4: Encoding scheme for 8-point DCT computation

Method	Encoding	Coefficients	
		Domain	Dimension
Dimitrov *et al.* [114]	Exact	$\mathbb{Z}$	8
Fu *et al.* [115]	Exact	$\mathbb{Z}$	8
Dimitrov *et al.* [106]	Non-exact	$\mathbb{Z}$	4
Dimitrov *et al.* [107]	Non-Exact	$\mathbb{Z}$	4
Shen *et al.* [110]	Non-exact	$\mathbb{Z}$	4
Proposed	Exact	$\{-1,0,1\}$	7

Table 3.5: 12-bit CSD encoding of the AI bases elements

Basis element	CSD Encoding
c_0	01.0000000000
c_1	10.0000$\bar{1}$0$\bar{1}$000
c_2	10.00$\bar{1}$0$\bar{1}$00100
c_3	10.$\bar{1}$0101010$\bar{1}$0
c_4	10.$\bar{1}$0$\bar{1}$0101000
c_5	01.00100$\bar{1}$0001
c_6	01.0$\bar{1}$0001000$\bar{1}$
c_7	00.10$\bar{1}$001000$\bar{1}$

3.4.1 Canonical Signed Digit Method

The CSD representation is capable of representing fixed point numbers according to a combination of powers-of-two with a minimum number of adders [45, 116]. Therefore, the multiplications required in the decoding operation described in (3.3) can be performed with few additions and bit-shifting operations.

Considering 12-bit representation, Table 3.5 shows the CSD representation of the proposed AI basis elements $\boldsymbol{\zeta}$ as in (3.2), where 1 and $\bar{1}$ represent additive and subtractive powers of two, respectively [116]. The basis element c_4 was included for completeness; however, it is not necessary for the decoding stage as discussed in Section 3.3.1. Table 3.6 shows the minimum and maximum absolute error of the CSD representation for basis elements at several wordlengths N.

Table 3.6: CSD encoding error

N	Error	
	min	max
5	$1.52 \cdot 10^{-2}$	$1.11 \cdot 10^{-1}$
6	$1.52 \cdot 10^{-2}$	$4.86 \cdot 10^{-2}$
7	$4.01 \cdot 10^{-3}$	$2.41 \cdot 10^{-2}$
8	$1.77 \cdot 10^{-3}$	$1.54 \cdot 10^{-2}$
9	$6.33 \cdot 10^{-4}$	$7.55 \cdot 10^{-3}$
10	$1.03 \cdot 10^{-4}$	$3.65 \cdot 10^{-3}$
11	$1.03 \cdot 10^{-4}$	$1.77 \cdot 10^{-3}$
12	$1.03 \cdot 10^{-4}$	$8.3 \cdot 10^{-4}$

3.4.2 Expansion Factor

The expansion factor method consists of finding a real number $\alpha^* > 1$ such that $\alpha^* \cdot \boldsymbol{\zeta}$ is close to an integer vector. This allows decoding by means of integer arithmetic operations according to:

$$\alpha^* \cdot f_{\text{dec}}(\mathbf{x}; \boldsymbol{\zeta}) = \mathbf{x}^\top \cdot (\alpha^* \cdot \boldsymbol{\zeta}),$$

where $\alpha^* \cdot \boldsymbol{\zeta}$ is approximately a vector of integers. On the other hand, the decoded quantities are scaled by α^*; thus this method offers the scaled DCT. The expansion factor α^* satisfies the following optimization problem:

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

Above problem is non-linear and an algebraic closed-form solution may not be simple. Thus, we resort to exhaustive search methods to numerically solve (3.6). Because the basis component c_4 is not required for decoding (cf. Section 3.3.1), we solve (3.6) considering the reduced vector $\boldsymbol{\zeta}' = \begin{bmatrix} 1 & c_1 & c_2 & c_3 & c_5 & c_6 & c_7 \end{bmatrix}^\top$ instead of $\boldsymbol{\zeta}$.

For instance, considering the search space $[0, 1023]$ (10-bit wordlength) and a step size of 10^{-2},

Table 3.7: Scale factors and maximum/minimum relatives error

N	α^*	$\zeta - \frac{\text{round}(\alpha^* \cdot \zeta)}{\alpha^*}$	
		$\min(\cdot)$	$\max(\cdot)$
5	25.98	$5.27 \cdot 10^{-4}$	$7.18 \cdot 10^{-3}$
6	25.98	$5.27 \cdot 10^{-4}$	$7.18 \cdot 10^{-3}$
7	107.10	$7.44 \cdot 10^{-5}$	$2.01 \cdot 10^{-3}$
8	107.10	$7.44 \cdot 10^{-5}$	$2.01 \cdot 10^{-3}$
9	341.01	$4.84 \cdot 10^{-13}$	$3.06 \cdot 10^{-4}$
10	341.01	$4.84 \cdot 10^{-13}$	$3.06 \cdot 10^{-4}$
11	1844.96	$2.13 \cdot 10^{-7}$	$7.38 \cdot 10^{-5}$
12	1844.96	$2.13 \cdot 10^{-7}$	$7.38 \cdot 10^{-5}$

we obtain a optimal value of $\alpha^* = 719.86$, leading to

$$341.01 \cdot \boldsymbol{\zeta} = \begin{bmatrix} 341.01 \\ 668.92\ldots \\ 630.10\ldots \\ 567.08\ldots \\ 482.26\ldots \\ 378.91\ldots \\ 261.00\ldots \\ 133.06\ldots \end{bmatrix} \approx \begin{bmatrix} 341 \\ 669 \\ 630 \\ 567 \\ 482 \\ 379 \\ 261 \\ 133 \end{bmatrix}.$$

Notice that the element 482, which corresponds to c_4, is never used and is not submitted to implementation. Table 3.7 presents optimal expansion factors for several wordlengths N with precision of 10^{-2}. Minimum and maximum relative errors are also shown.

3.5 VLSI Implementation

The proposed algorithm was initially modeled and tested in MATLAB Simulink using Xilinx block set. Two separate designs were used to realize both the CSD and the expansion factor approach. For hardware comparison purposes, the competing AI algorithm proposed in [110] was also realized.

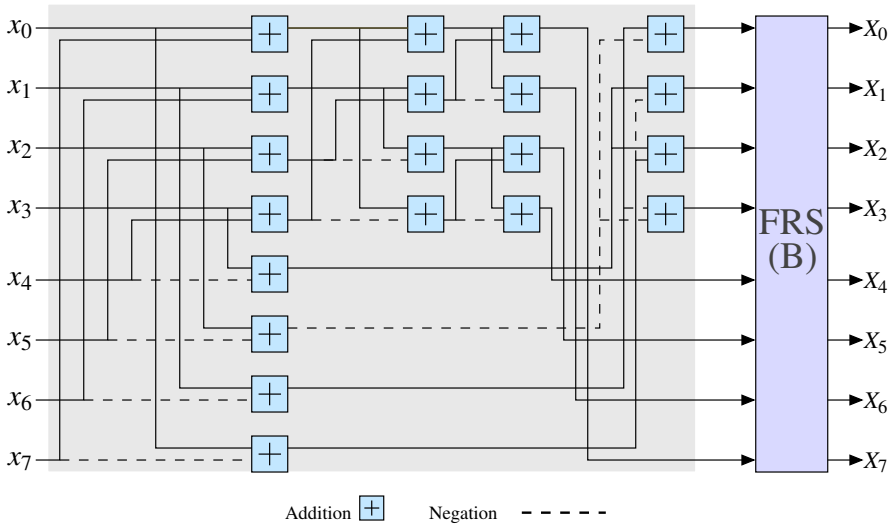


Figure 3.4: Digital architecture of the Loeffler algorithm for the 8-point DCT computation.

All three methods were designed using bit-shifting operations and reinterpretation blocks. Hence no multipliers were needed in implementation of the FRS. Figure 3.4 depicts the architecture for the proposed 1-D Loeffler DCT. Figure 3.5 displays the FRS implementation of hardware using zero multipliers.

3.5.1 Hardware Implementation and Results

Architectures for the CSD, expansion factor, and Shen's methods were physically realized on Xilinx Virtex-6 XC6VSX475T-1FF1759 ROACH2 Board[117]. A single layer of pipelining was included in all architectures for increased throughput. Realizations were designed using 32-bit software registers, both on input and output. The CSD method was physically realized considering the 12-bit representation depicted in Table 3.5; whereas the expansion factor method was implemented considering the discussed 10-bit representation. Shen's algorithm was implemented using the 16-bit CSD method as provided in [110]. All three designs were evaluated for hardware complexity and real-time performance using metrics such as configurable logic blocks (CLB) and flip-flop (FF) count, critical path delay (T_{cpd}) in ns, and maximum operating frequency (F_{max}) in MHz. Values were obtained from the Xilinx FPGA synthesis and place-route tools by accessing the `xflow.results` report file. Table 3.8 depicts the FPGA hardware resource consumption for

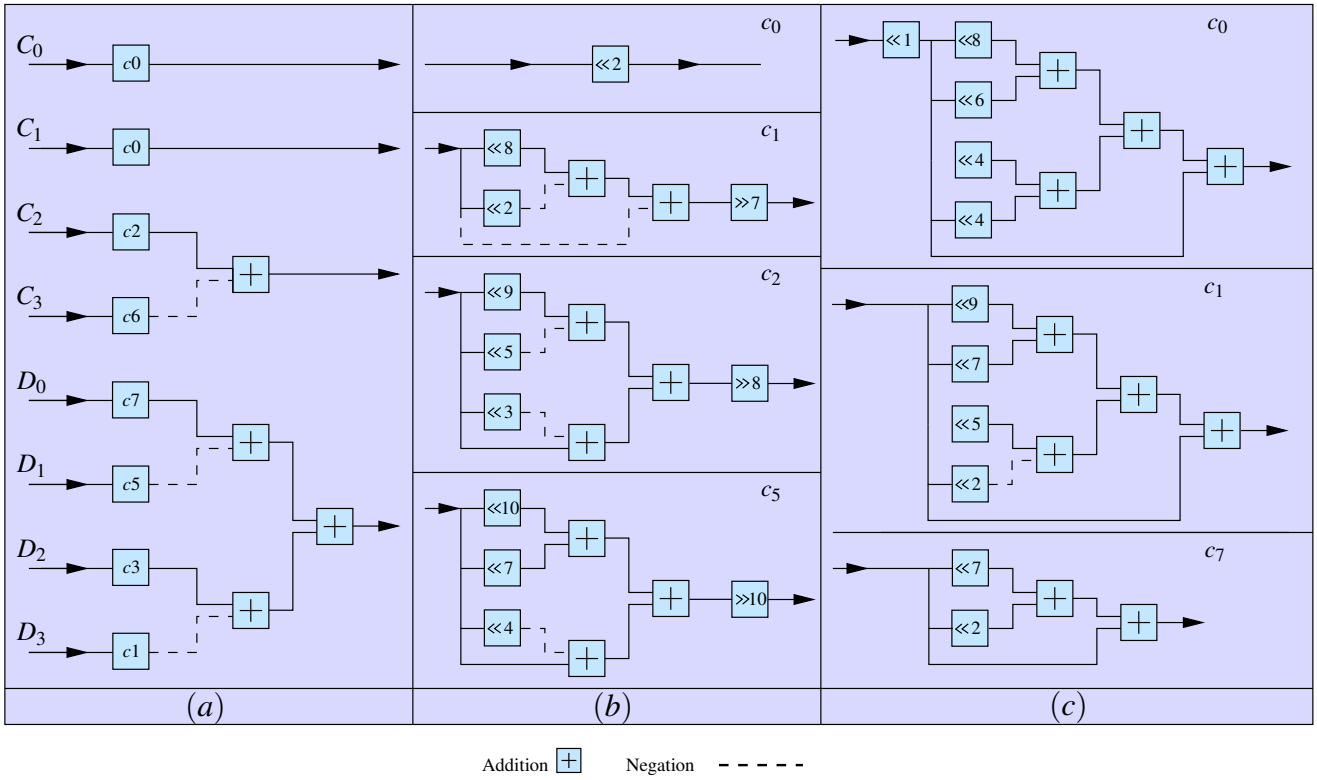


Figure 3.5: Digital implementation of several outputs using the FRS: (a) computation of the final outputs $\mathbf{X}_0, \mathbf{X}_1, \ldots, \mathbf{X}_7$; (b) computation of $c_0, c_1, \ldots, c_7$ using the CSD method as shown in Table 3.5, and (c) computation of $c_0, c_1, \ldots, c_7$ using the expansion factor method.

Table 3.8: Hardware resource consumption using Xilinx Virtex-6 XC6VSX475T 1FF1759 device

FRS Method	CLB	FF	T_{cpd} (ns)	F_{max} (MHz)
CSD Method	1573	1572	9.958	100.42
EF Method	1565	1455	9.960	100.40
Shen's Method [110]	1573	1890	17.196	58.15

Table 3.9: Hardware resource consumption for CMOS 18nm ASIC synthesis

FRS Method	Area (mm^2)	AT	AT^2	T_{cpd} (ns)	F_{max} (MHz)	D_p (mW/MHz)
CSD Method	0.628	1.879	5.625	2.993	334	1.60
EF Method	0.776	2.584	8.610	3.310	302	2.18
Shen's Method[110]	0.989	4.007	16.238	4.052	246	4.01

the CSD, the expansion factor and Shen's algorithm designs.

The ASIC implementation was realized by porting the hardware description language code to 18 nm CMOS technology and was subjected to synthesis using the AMS Encounter Digital Implementation (EDI) libraries. Libraries for the best case scenario were used in getting the synthesis results with gate voltage of 1.8 V. The adopted figures of merit for the ASIC synthesis were: area (A) in mm^2, area-time complexity (AT) in $\text{mm}^2 \cdot \text{ns}$, area-time-squared complexity (AT^2) in $\text{mm}^2 \cdot \text{ns}^2$, dynamic ($D_p$) power consumption in mW/MHz, critical path delay (T_{cpd}) in ns, and maximum operating frequency (F_{max}) in MHz. Results for all three methods are displayed in Table 3.9.

3.6 Chapter Summary

An algebraic integer encoding scheme capable of precise numerical representation was proposed. The encoding is based on an integer vector representation of the 8-point DCT kernel multiplicands. The proposed scheme paves the way for a fully error-free AI-based Loeffler DCT implementation with null multiplicative complexity. A fast algorithm that requires only 20 additions was also introduced; outperforming competing methods. Two procedures for AI decoding (FRS block) were

suggested. Proposed architectures were physically realized in a ROACH2 board, which consists of a Virtex-6 VSX475T FPGA, and ported to 18 nm ASIC synthesis using AMS standard cell libraries. Both FPGA and ASIC realizations have operating frequencies of 100 MHz and 334 MHz, respectively.

3.7 Appendix

3.7.1 Dense Representation

Consider ζ as in (3.2) and let x be an arbitrary real number. For each $\varepsilon > 0$, there are a_k and $b_k \in \mathbb{Z}$, $k = 1, 2, \ldots, 7$, such that:

$$|x - b_k - a_k \cdot c_k| < \sqrt{\varepsilon/7}.$$

Now let $a_0 = \sum_{k=1}^{7} b_k$. Thus, we have that

$$|x - a_0 - \sum_{k=1}^{7} a_k c_k|^2 \leq \sum_{k=1}^{7} |x - b_k - a_k c_k|^2 < \sum_{k=1}^{7} \sqrt{\varepsilon/7}^2 < \varepsilon.$$

As consequence, the proposed AI representation is dense over the set of real numbers.

3.7.2 Largest Representable Number

Considering (3.1), we notice that x is a conical combination [118, 119] of $|a_k| \leq M$ for $k = 0, 1, \ldots, 7$. Thus, the maximum value of x is obtained by maximizing the coefficients a_k, i.e., $a_k = M$, for all k. Therefore, we obtain:

$$x_{\max} = M + 2M \sum_{k=0}^{7} \cos\left(\frac{k\pi}{16}\right).$$

Considering the Dirichlet kernel formula [120], we have $x_{\max} = \frac{\sin(\frac{15\pi}{32})}{\sin(\frac{\pi}{32})} \cdot M$.

Chapter 4

Efficient Computation of the 8-point DCT via Summation by Parts

4.1 Introduction

Discrete transforms play a central role in signal processing. Noteworthy methods include trigonometric transforms—such as the discrete Fourier transform (DFT) [1], discrete Hartley transform (DHT) [1], discrete cosine transform (DCT) [45], and discrete sine transform (DST) [45]—as well as the Haar and Walsh-Hadamard transforms [121]. Among these methods, the DCT has been applied in several practical contexts: noise reduction [95], watermarking methods [122], image/video compression techniques [45], and harmonic detection [45], to cite a few. In fact, when processing signals modeled as a stationary Markov-1 type random process, the DCT behaves as the asymptotic case of the optimal Karhunen–Loève transform in terms of data decorrelation [45]. This approximation holds true when the correlation coefficient of the related stochastic process tends to the unit, which is the case for many real signals—specially images [45]. Moreover, the recent increase in image/video processing demand for consumer electronics [123] and big data manipulation [124] emphasizes the necessity for fast and efficient DCT computation [125].

As a consequence, the 8-point DCT is adopted in several image and video coding schemes [98], such as JPEG [99], MPEG-1 [99], H.264 [101], HEVC [102], AVS China [126, p. 61], and VP-10 [126, p. 165]. Aiming at minimizing the computational cost of the DCT evaluation, a number of fast algorithms for the 8-point DCT have been proposed, including Chen's DCT algorithm [127], Lee method [128], Loeffler algorithm [103], Feig-Winograd DCT factorization [129], and the Arai DCT [130].

Multiplication operations as required by DCT and others discrete-time transforms can be im-

plemented via long sequences of additions, bit-shifting operation, and sign changes [131]. Thus, algorithms that require multiplications often have higher computational costs [14]. Therefore, above-mentioned methods were developed in order to reduce the overall number of multiplications [45]. The Arai DCT is particularly useful because it furnishes a scaled version of the DCT spectrum. In some applications such as harmonic detection [132, 133] and JPEG-like image compression [99, 98], the scaled DCT is often a sufficient tool. This is because in these contexts only the relative value of the spectrum is necessary. Therefore, part of the cost of computing the DCT can be avoided [45].

Among the fundamental mathematical tools, we separate the summation-by-parts technique [26, p. 54], which is the discrete-time counterpart for the well-known integration-by-parts method [134, p. 144]. Although applied in several contexts such as computational physics for approximate second derivatives [135], approximations of the linear advection-diffusion equation in computational fluid dynamics [136], and rapid calculation of slow converging series in electromagnetic problems [137], it has been particularly overlooked by the signal processing community. Early attempts to employ it as a numerical analysis tool are due to Boudreaux-Bartels and collaborators in the context of the DFT computation [138] and the evaluation of Fourier coefficients errors calculations [139].

The aim of this chapter is to propose a new fast algorithm for the 8-point DCT computation based on the summation-by-parts formula for periodic signals [26, 140]. The introduced method is sought to achieve the theoretical minimal multiplicative complexity for the exact DCT computation [105, 52]. Moreover, to further minimize computational costs, the proposed algorithm is also sought to provide a scaled version of the DCT spectrum [45]. The proposed algorithm finds application in some important problems, such as feature detection, where DC level may not be relevant [141, 142]. Also, it can be applied to scenarios where input signal is natively accumulated (integrated) [1, p. 19]. This situation occurs in face recognition problems, where usual algorithms require data to be integrated [143, 144].

This chapter is organized as follows. In Section 4.2, we furnish the mathematical background for the summation-by-parts technique and the DCT. Considering matrix formalism, we detail the proposed algorithm for the DCT in Section 4.3. In Section 4.4, the introduced method is assessed in terms of its computational complexity and comparisons with competing algorithms are shown. Section 4.5 brings final comments and remarks.

4.2 Mathematical Background

4.2.1 Summation-by-parts

The summation-by-parts technique is the discrete-time equivalent of the integration-by-parts method [26]. Let $x[n]$ and $y[n]$ be two discrete-time signals. The summation-by-parts prescribes that [26, 138]:

$$\begin{aligned}\sum_{n=0}^{N-1} x[n]y[n] =& x[N]y[N] - x[0]y[0] \\ &- \sum_{n=0}^{N-1} \left(\sum_{i=0}^{n-1} x[i] \right) \cdot \Delta y[n],\end{aligned}$$

where Δ denotes the forward difference operator given by $\Delta y[n] \triangleq y[n+1] - y[n]$ [26]. Above expression can be simplified with the assumption of the following additional weak conditions. Admitting that the considered signals are periodic with period N, it was established in [140] that:

$$\sum_{n=0}^{N-1} x[n] \cdot y[n] = - \sum_{n=0}^{N-1} \left(\sum_{i=0}^{n-1} x[i] \right) \cdot \Delta y[n]. \tag{4.1}$$

The above condition is not too restrictive. Indeed discrete-time Fourier analysis often assume that the input signals are periodic [1, 45, 94]. In particular, the DCT can be obtained as the solution to the harmonic oscillation problem [45].

The expression $\sum_{n=0}^{N-1} x[n] \cdot y[n]$ can be interpreted as a discrete-time transformation. Let $x[n]$ be the input signal to be transformed and $y[n] = \ker[n,k]$ be a given discrete transformation kernel for the kth transform-domain component. Therefore, we have that:

$$X[k] = \sum_{n=0}^{N-1} x[n] \cdot \ker[n,k], \quad k = 0, 1, \ldots, N-1, \tag{4.2}$$

Table 4.1: Common discrete transform kernels

Transform	$\ker[n,k]$
DFT [1]	$\exp\left(-j\frac{2\pi nk}{N}\right)$
DHT [1]	$\operatorname{cas}\left(\frac{2\pi nk}{N}\right)$
DCT [45]	$\cos\left(\frac{\pi(2n+1)k}{2N}\right)$
DST [45]	$\sin\left(\frac{\pi}{N}(k+\frac{1}{2})(n+\frac{1}{2})\right)$

where $X[k]$ is the transformed output signal. Table 4.1 summarizes common transformation kernels. Therefore, applying (4.1) into (4.2) yields the following expression for the transform-domain components:

$$\begin{aligned} X[k] &= -\sum_{n=0}^{N-1}\left(\sum_{i=0}^{n} x[i]\right)\cdot\Delta\ker[n,k] \\ &= -\sum_{n=0}^{N-1} z[n]\cdot\Delta\ker[n,k], \quad k=0,1,\ldots,N-1, \end{aligned} \tag{4.3}$$

where $z[n]=\sum_{i=0}^{n} x[i]$, for $n=0,1,\ldots,N-1$. Comparing (4.2) with (4.3), we notice that the original transform expression was re-written into an alternative form where both the input data and the kernel function were processed. Notice that $z[n]$ is the output of an accumulator system for input signal $x[n]$ [1]; whereas $\Delta\ker[n,k]$ derives from a forward difference system for input signal $\ker[n,k]$ [1]. Although the forward difference system is not causal, this fact poses no difficulty to above formalism. This is because $\ker[n,k]$ is not a random real-time sequence—but a deterministic sequence whose values are known *a priori* [131, p. 7].

Moreover, if $x[n]$ possesses null mean, then the following expression holds true:

$$z[N-1]=\sum_{i=0}^{N-1} x[i]=0 \quad \text{and} \quad X[0]=0.$$

For trigonometric transforms, above condition implies $X[0]=0$ (null DC value). Therefore, (4.3) can be simplified and written as:

$$X[k]=-\sum_{n=0}^{N-2} z[n]\cdot\Delta\ker[n,k], \quad k=1,2,\ldots,N-1. \tag{4.4}$$

Above summation ranges from 0 to $N-2$. This means that the transformation matrix linked to (4.3) has dimension $(N-1)\times(N-1)$. This fact contrasts with the original transformation matrix,

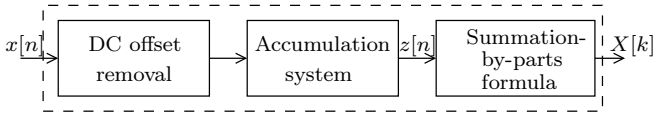


Figure 4.1: Block diagram of the proposed architecture.

which has size $N \times N$. Thus, the summation-by-parts effected a dimension reduction of transform computation. As a consequence, the computational cost of associate algorithms is expected to be reduced.

Figure 4.1 depicts the overall diagram for the transform computation based on the summation-by-parts formula, when $x[n]$ is assumed to be an arbitrary signal. Notice that, if N is a power of two, both the DC removal block and the accumulation system are multiplierless operations.

4.2.2 Discrete Cosine Transform

The DCT is a linear transformation that maps an N-point discrete-time signal $x[n]$ into another N-point discrete-time signal $X[k]$ according to the following relationship [103]:

$$X[k] = \frac{4}{\sqrt{N}}\,\alpha_k \sum_{n=0}^{N-1} x[n] \cos\left(\frac{\pi(2n+1)k}{2N}\right), \tag{4.5}$$

where $k = 0, 1, \ldots, N-1$, $\alpha_0 = 1/\sqrt{2}$, and $\alpha_k = 1$, for $k > 0$. The above expression can be given a compact format by means of matrix representation. Indeed, considering signals $x[n]$ and $X[k]$ in column vector format as $\mathbf{x} = \begin{bmatrix} x[0] & x[1] & \cdots & x[N-1] \end{bmatrix}^\top$ and $\mathbf{X} = \begin{bmatrix} X[0] & X[1] & \cdots & X[N-1] \end{bmatrix}^\top$, we have that:

$$\mathbf{X} = \mathbf{C}_N \cdot \mathbf{x}, \tag{4.6}$$

where $\mathbf{C}_N$ is the DCT matrix, whose (k,n)-entry is given by $\frac{4}{\sqrt{N}}\alpha_k \cos\left(\frac{\pi(2n+1)k}{2N}\right)$. For $N = 8$, we have the following transformation matrix:

$$\mathbf{C} \triangleq \mathbf{C}_8 = \sqrt{2} \cdot \begin{bmatrix} c_4 & c_4 & c_4 & c_4 & c_4 & c_4 & c_4 & c_4 \\ c_1 & c_3 & c_5 & c_7 & -c_7 & -c_5 & -c_3 & -c_1 \\ c_2 & c_6 & -c_6 & -c_2 & -c_2 & -c_6 & c_6 & c_2 \\ c_3 & -c_7 & -c_1 & -c_5 & c_5 & c_1 & c_7 & -c_3 \\ c_4 & -c_4 & -c_4 & c_4 & c_4 & -c_4 & -c_4 & c_4 \\ c_5 & -c_1 & c_7 & c_3 & -c_3 & -c_7 & c_1 & -c_5 \\ c_6 & -c_2 & c_2 & -c_6 & -c_6 & c_2 & -c_2 & c_6 \\ c_7 & -c_5 & c_3 & -c_1 & c_1 & -c_3 & c_5 & -c_7 \end{bmatrix},$$

where $c_k = \cos(k\pi/16)$, for $k = 1, 2, \ldots, 7$. This particular definition of the DCT is adopted in [145, 146, 147], having been considered to derive the well-known Loeffler DCT algorithm [103]. Notice that $\sqrt{2} \cdot c_4 = 1$; therefore the DC component $X[0]$ is evaluated without multiplications [103]. In [52], Heideman introduces an in-depth mathematical analysis of the multiplicative complexity of major discrete transforms. A result from multiplicative complexity theory that is relevant to the current work is the following. If the transform blocklength is a power of two, $N = 2^r$, $r = 1, 2, \ldots$, then the minimum multiplicative complexity $\mu(N)$ of the DCT has a general formula given by $\mu(2^r) = 2^{r+1} - r - 2$ [52, Theorem 6.3, p. 117]. For $N = 8$, we obtain $\mu(2^3) = 11$.

4.3 DCT Computation via Summation-by-parts

In this section, we apply the summation-by-parts technique to propose an alternative computation of the 8-point DCT. Next we analyze the resulting expressions in order to derive a fast algorithm by means of matrix factorization.

4.3.1 Matrix Formalism

To facilitate the development of the sought DCT fast algorithm, we re-cast the summation-by-part formula into matrix formalism. But, first, the forward difference operator needs to be adapted to manipulate matrices. Let $\mathbf{M}$ be a square matrix. Then $\Delta\mathbf{M}$ is the matrix resulted from cyclically applying the forward difference operator to each row of $\mathbf{M}$.

Therefore, considering the summation-by-parts formula shown in (4.4), we have that (4.6) can be written according to:

$$\mathbf{X} = \Delta\mathbf{C} \cdot \mathbf{z},$$

where the N-point vector $\mathbf{X} = \begin{bmatrix} X[0] & X[1] & \cdots & X[N-1] \end{bmatrix}^\top$ represents the DCT spectrum, $\mathbf{z} =$

$\begin{bmatrix} z[0] & z[1] & \cdots & z[N-1] \end{bmatrix}^\top$ is the accumulated input signal, and

$$\Delta\mathbf{C} = \sqrt{2}\times$$
$$\begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ c_3-c_1 & c_5-c_3 & c_7-c_5 & -2c_7 & c_7-c_5 & c_5-c_3 & c_3-c_1 & 2c_1 \\ c_6-c_2 & -2c_6 & c_6-c_2 & 0 & c_2-c_6 & 2c_6 & c_2-c_6 & 0 \\ -c_3-c_7 & c_7-c_1 & c_1-c_5 & 2c_5 & c_1-c_5 & c_7-c_1 & -c_3-c_7 & 2c_3 \\ -2c_4 & 0 & 2c_4 & 0 & -2c_4 & 0 & 2c_4 & 0 \\ -c_1-c_5 & c_1+c_7 & c_3-c_7 & -2c_3 & c_3-c_7 & c_1+c_7 & -c_1-c_5 & 2c_5 \\ -c_2-c_6 & 2c_2 & -c_2-c_6 & 0 & c_2+c_6 & -2c_2 & c_2+c_6 & 0 \\ -c_5-c_7 & c_3+c_5 & -c_1-c_3 & 2c_1 & -c_1-c_3 & c_3+c_5 & -c_5-c_7 & 2c_7 \end{bmatrix}.$$

Considering the sum-to-product identities [112, p. 72] and symmetry identities [26], the entries of the matrix $\Delta\mathbf{C}$ can be given a multiplicative form, as shown below:

$$\Delta\mathbf{C} = 2\sqrt{2}\cdot\begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ s_1s_2 & s_1s_4 & s_1s_6 & s_1 & s_1s_6 & s_1s_4 & s_1s_2 & s_7 \\ s_2s_4 & s_2 & s_2s_4 & 0 & -s_2s_4 & -s_2 & -s_2s_4 & 0 \\ s_3s_6 & s_3s_4 & -s_3s_2 & -s_3 & -s_3s_2 & s_3s_4 & s_3s_6 & s_5 \\ s_4 & 0 & -s_4 & 0 & s_4 & 0 & -s_4 & 0 \\ s_5s_6 & -s_5s_4 & -s_5s_2 & s_5 & -s_5s_2 & -s_5s_4 & s_5s_6 & s_3 \\ s_6s_4 & -s_6 & s_6s_4 & 0 & -s_6s_4 & s_6 & -s_6s_4 & 0 \\ s_7s_2 & -s_7s_4 & s_7s_6 & -s_7 & s_7s_6 & -s_7s_4 & s_7s_2 & s_1 \end{bmatrix},$$

for $s_k = \sin(k\pi/16)$, $k = 1,2,\ldots,7$.

If the input signal possesses null mean, then we have that $X[0] = 0$ and $z[N-1] = 0$. Therefore, the first row and last column of $\Delta\mathbf{C}$ can be neglected. It follows that only the remaining 7×7 submatrix is necessary for the computation of $\mathbf{X}$. This particular matrix is given by:

$$\widetilde{\mathbf{C}} = 2\sqrt{2}\cdot\begin{bmatrix} s_1s_2 & s_1s_4 & s_1s_6 & s_1 & s_1s_6 & s_1s_4 & s_1s_2 \\ s_2s_4 & s_2 & s_2s_4 & 0 & -s_2s_4 & -s_2 & -s_2s_4 \\ s_3s_6 & s_3s_4 & -s_3s_2 & -s_3 & -s_3s_2 & s_3s_4 & s_3s_6 \\ s_4 & 0 & -s_4 & 0 & s_4 & 0 & -s_4 \\ s_5s_6 & -s_5s_4 & -s_5s_2 & s_5 & -s_5s_2 & -s_5s_4 & s_5s_6 \\ s_6s_4 & -s_6 & s_6s_4 & 0 & -s_6s_4 & s_6 & -s_6s_4 \\ s_7s_2 & -s_7s_4 & s_7s_6 & -s_7 & s_7s_6 & -s_7s_4 & s_7s_2 \end{bmatrix}.$$

Notice that $\widetilde{\mathbf{C}}$ is sufficient for the computation of all DCT coefficients—except the DC level.

4.3.2 Scaling Matrix

An examination of matrix $\widetilde{\mathbf{C}}$ shows repeated multiplicands along its rows. Thus, the following factorization is obtained:

$$\widetilde{\mathbf{C}} = \mathbf{S}\cdot\begin{bmatrix} s_2 & s_4 & s_6 & 1 & s_6 & s_4 & s_2 \\ s_4 & 1 & s_4 & 0 & -s_4 & -1 & -s_4 \\ s_6 & s_4 & -s_2 & -1 & -s_2 & s_4 & s_6 \\ 1 & 0 & -1 & 0 & 1 & 0 & -1 \\ s_6 & -s_4 & -s_2 & 1 & -s_2 & -s_4 & s_6 \\ s_4 & -1 & s_4 & 0 & -s_4 & 1 & -s_4 \\ s_2 & -s_4 & s_6 & -1 & s_6 & -s_4 & s_2 \end{bmatrix},$$

where $\mathbf{S} = 2\sqrt{2} \cdot \mathrm{diag}(s_1, s_2, s_3, s_4, s_5, s_6, s_7)$. The above expression tells us that the matrix $\mathbf{S}$ contributes only with scaling factors to the actual DCT computation. When considering applications that require only a scaled version of the DCT—such as harmonic detection [133] and color enhancement [148]—the computational cost of $\mathbf{S}$ can be disregarded. Additionally, in the context of image compression, diagonal matrices can be directly absorbed into the quantization matrix; representing no extra computation [149, 107].

Notice also that the scaling factor $2\sqrt{2}s_4 = 2$ is a trivial multiplication [14] that can be implemented via a simple bit-shifting operation. Therefore, the computational cost of the scaling matrix $\mathbf{S}$ is actually only six—not seven—multiplications.

4.3.3 Fast Algorithm

Now our aim is to provide a sparse matrix factorization to $\widetilde{\mathbf{C}}$. Because $\widetilde{\mathbf{C}}$ is a highly symmetrical matrix, factorization methods based on butterfly structures can be directly applied [1, 45, 14]. Therefore, we obtain the following factorization:

$$\widetilde{\mathbf{C}} = \mathbf{S} \cdot \mathbf{P} \cdot \mathbf{M} \cdot \mathbf{A},$$

where

$$\mathbf{P} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 \end{bmatrix},$$

$$\mathbf{A} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & -1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & -1 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & -1 \end{bmatrix},$$

and

$$\mathbf{M} = \begin{bmatrix} s_2 & s_4 & s_6 & 1 & 0 & 0 & 0 \\ s_6 & s_4 & -s_2 & -1 & 0 & 0 & 0 \\ s_6 & -s_4 & -s_2 & 1 & 0 & 0 & 0 \\ s_2 & -s_4 & s_6 & -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & s_4 & 1 & s_4 \\ 0 & 0 & 0 & 0 & -1 & 0 & 1 \\ 0 & 0 & 0 & 0 & s_4 & -1 & s_4 \end{bmatrix}.$$

Matrix $\mathbf{P}$ is a simple permutation matrix, which represents no computational cost. In terms of hardware, $\mathbf{P}$ translates into simple wiring. Matrix $\mathbf{A}$ is an additive matrix consisting of the usual

butterfly stage present in decimation-in-frequency algorithms [14]. The remaining matrix $\mathbf{M}$ is block-diagonal and still contains mathematical redundancies due to its symmetrical nature. Considering the matrix factorizations for fast algorithm design described in [14], the following expression can be obtained:

$$\mathbf{M} = \mathbf{M}_1 \cdot \mathbf{M}_2 \cdot \mathbf{M}_3 \cdot \mathbf{M}_4,$$

where

$$\mathbf{M}_1 = \begin{bmatrix} 1 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & -1 & 0 & 0 & 0 \\ 1 & 0 & -1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 \end{bmatrix},$$

$$\mathbf{M}_2 = \begin{bmatrix} s_2 & s_6 & 0 & 0 & 0 & 0 & 0 \\ s_6 & -s_2 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 & -1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & -1 \end{bmatrix},$$

$$\mathbf{M}_3 = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & s_4 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & s_4 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix},$$

$$\mathbf{M}_4 = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & -1 \end{bmatrix}.$$

The block-diagonal matrix $\mathbf{M}_2$ contains a rotation block $\begin{bmatrix} s_2 & s_6 \\ s_6 & -s_2 \end{bmatrix}$, which can be further decomposed [45]. Thus, we have:

$$\mathbf{M}_2 = \mathbf{R}_1 \cdot \mathbf{R}_2 \cdot \mathbf{R}_3,$$

where

$$\mathbf{R}_1 = \begin{bmatrix} 0 & s_6 - s_2 & 1 & 0 & 0 & 0 & 0 & 0 \\ s_2 + s_6 & 0 & -1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix},$$

$$\mathbf{R}_3 = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 & -1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & -1 \end{bmatrix},$$

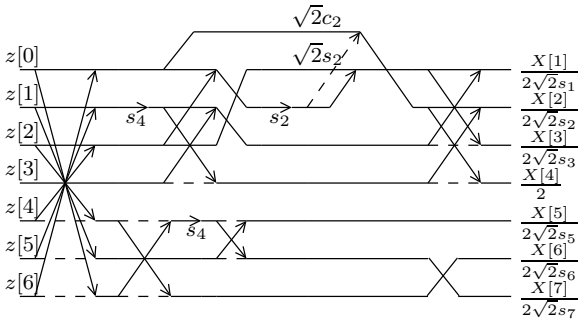


Figure 4.2: SFG for the proposed algorithm. Dashed lines represents multiplication by -1.

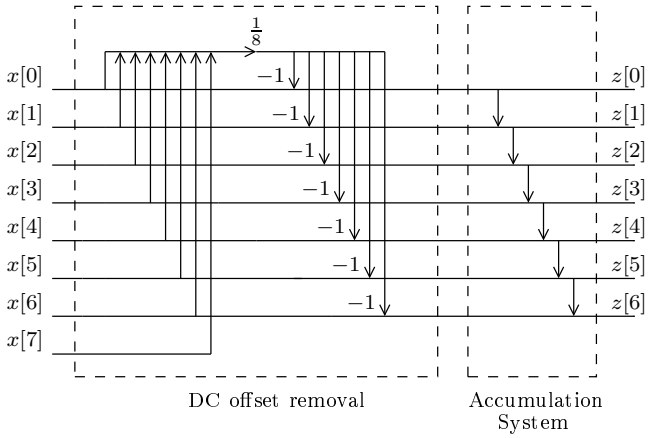


Figure 4.3: SFG for pre-processing stage consisting of the DC component removal coupled with accumulation system.

and $\mathbf{R}_2 = \text{diag}(1,1,s_2,1,1,1,1,1)$.

By means of usual trigonometric manipulations, we notice that the required multiplicands in $\mathbf{R}_1$ satisfy: $s_2 + s_6 = \sqrt{2}\,c_2$ and $s_6 - s_2 = \sqrt{2}\,s_2$. Finally, the complete sparse matrix factorization is given by:

$$\widetilde{\mathbf{C}} = \mathbf{S} \cdot \mathbf{P} \cdot \mathbf{M}_1 \cdot \mathbf{R}_1 \cdot \mathbf{R}_2 \cdot \mathbf{R}_3 \cdot \mathbf{M}_3 \cdot \mathbf{M}_4 \cdot \mathbf{A}.$$

The signal flow graph (SFG) for the proposed algorithm is shown in Figure 4.2.

4.4 Computational Complexity Assessment and Comparison

In this section, we assess the computational complexity of the proposed algorithm as measured by its arithmetic cost. For such, we evaluate the additive and multiplicative count required by the

introduced method.

4.4.1 Results

We distinguish the following scenarios for the input data: (i) arbitrary signal; (ii) null mean signal; (iii) accumulated signal; and (iv) null mean and accumulated signal. Scenario (i) is the most general case. Thus, there is less redundancy to be exploited aiming at the minimization of computational cost. Scenario (i) is commonly found in image compression applications [45, 94]. Scenario (ii) is pertinent in the context of feature detection, for instance, where DC level may not be relevant [141, 142]. Scenario (iii) represents the case where the input signal is natively accumulated (integrated). This situation occurs in face recognition problems, where usual algorithms require data to be integrated [143, 144]. Scenario (iv) is a combination of above last two cases. Notice that Scenario (ii) does not require the DC offset removal block (cf. Figure 4.1); Scenario (iii) does not require the accumulation system but needs a DC removal block; and Scenario (iv) requires neither the DC offset removal block nor the accumulation system.

Table 4.2 compares the proposed method with several prominent techniques for the 8-point DCT evaluation under all discussed scenarios. The details of the arithmetic complexity assessment of each considered technique is found in [45]. For each scenario, we emphasized in bold the best results. Because Arai method [130] and the proposed algorithm are scaled DCT methods, we show the multiplicative complexity of the non-scaled computation as well. In parenthesis, when applicable, we furnish the full multiplicative complexity of the scaled methods, when the scaling factors are considered. The multiplicative complexity for Loeffler, Lee, and Chen DCT algorithms are limited to the non-scaled case, because these methods do not admit scaled computation. The proposed algorithm could outperform *all* competing method under Scenarios (ii), (iii), and (iv). Although the proposed method requires five multiplications to compute the scaled DCT—the same as the Arai method—it demands fewer scaling multiplications. Scenario (i) is not ideally suitable for the proposed algorithm because it benefits of the DC level removal and accumulated input signal. However, even in this case, the introduced algorithm could achieve the theoretical minimum

Table 4.2: Comparison for non-trivial products and additions for 8-point DCT algorithms

Algorithm	Scaled?	Scenarios	Mult.	Additions
Loeffler [103]	No	(i)	**11**	**29**
		(ii)	11	26
		(iii)	11	36
		(iv)	11	33
Lee [128]	No	(i)	12	29
		(ii)	11	26
		(iii)	12	36
		(iv)	11	33
Chen *et al.* [127]	No	(i)	13	26
		(ii)	12	23
		(iii)	13	33
		(iv)	12	30
Arai *et al.* [130]	Yes	(i)	**5** (13)	**28**
		(ii)	5 (12)	25
		(iii)	5 (13)	35
		(iv)	5 (12)	32
Proposed	Yes	(i)	5 (11)	39
		(ii)	**5** (**11**)	**25**
		(iii)	**5** (**11**)	**30**
		(iv)	**5** (**11**)	**19**

of multiplicative complexity.

4.4.2 Discussion

For Scenario (i), the DC offset removal block requires seven additions to compute the mean value and then seven subtractions to remove the DC level from input samples $x[0], x[1], \ldots, x[6]$. Thus, a total of 14 additions are necessary. The accumulation system requires a total of six additions (cf. (4.4)). Thus, the DC removal block combined with accumulation system requires a total of 20 additions. Figure 4.3 details the inner structure of each of the above-mentioned blocks. Multiplicative costs are concentrated in matrices $\mathbf{S}$, $\mathbf{R}_1$, $\mathbf{R}_2$, and $\mathbf{M}_3$. They account for 11 non-trivial multiplications, which is the minimum multiplicative complexity for the 8-point DCT [105, 52]. Matrices $\mathbf{M}_1$, $\mathbf{R}_1$, $\mathbf{R}_3$, $\mathbf{M}_4$, and $\mathbf{A}$ contribute with 19 additions. Thus, under Scenario (i), the proposed algorithm requires a total of 39 additions. As we demonstrate below, this is the only scenario

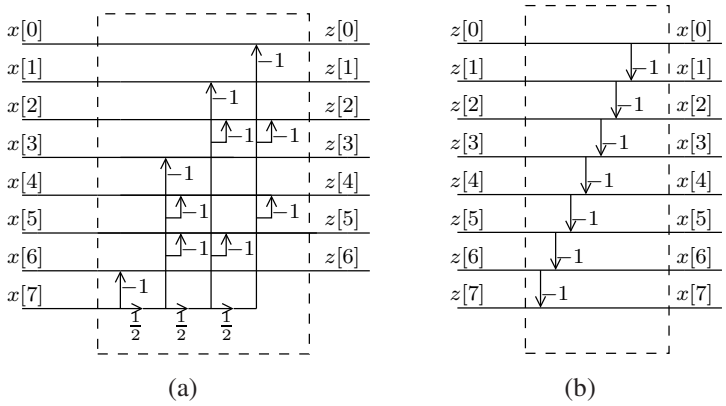


Figure 4.4: SFG of (a) the DC removal block and (b) the finite difference operator, both for accumulated input signal.

where the proposed method cannot outperform the most relevant method for the 8-point DCT computation. Nevertheless, even in this scenario, the proposed method attained the theoretical minimal multiplicative complexity, which is a relevant figure of merit [14]. In comparison with competing methods, the extra required computation consists of additions only.

Under Scenario (ii), the proposed algorithm does not demand the DC removal block but only one instantiation of an accumulation system (Figure 4.3, rightmost subblock). Therefore, a total of 13 additions in savings is attained when compared to Scenario (i). Consequently, considering Scenario (ii), the proposed algorithm requires only 25 additions. Usual methods presented in literature can be adapted for Scenario (ii). However, when compared to the costs under Scenario (i), such methods could save only three additions, instead of seven additions for the computation of the DC level. This is particularly true for both Loeffler [103] and Lee [128] algorithms which demand the same number of multiplications as the proposed method when exact spectrum is required. Additionally, the proposed algorithm can furnish a scaled version of the DCT spectrum at the same performance required by the Arai algorithm [130].

Under Scenario (iii), the proposed method does not demand the accumulation system, but still requires the DC removal block. In this case, the DC removal block must be adapted to process accumulated input signals. Figure 4.4(a) shows the modified DC removal block, which requires 10 additions to generate the sought sequence $z[n]$. In contrast, traditional methods for DCT

computation when faced with accumulated signals have to pre-process the input data through a difference system. Figure 4.4(b) shows the SFG for a difference system, which is completely *eliminated* in the proposed algorithm.

Scenario (iv) is a combination of Scenarios (ii) and (iii); being the most appropriate scenario for the proposed method. In this case, the proposed algorithm does not require any pre-processing stage, being the input data directly applicable. Thus, the proposed method is limited the operations described in the architecture shown in Figure 4.2. Under this scenario, usual methods requires a pre-processing stage consisting of a difference system (Figure 4.4(b)), which demands seven extra additions.

The proposed algorithm can be extended for different sequence sizes, which can benefit multiple-blocklength encoding methods such as HEVC [102, 45]. Such extension can be derived from (4.3) assuming an arbitrary length N. Considering the DCT kernel (Table 4.1) and applying the forward difference operator, we obtain:

$$\Delta \ker[n,k] = 2\alpha_k \sin\left(\frac{k\pi}{2N}\right) \sin\left(\frac{k\pi(n+1)}{N}\right),$$

where α_k is given as in (4.5) and $n,k = 1,2,\ldots,N-1$. Despite the similarity with the DST kernel, the above expression contains a post-multiplicative factor $2\alpha_k \sin(k\pi/2N)$ that can be separated into a diagonal matrix. Thus the design of the associate fast algorithm becomes equivalent to the problem of factorizing an $(N-1)\times(N-1)$ matrix with entries given by $\sin(k\pi(n+1)/N)$, $n,k = 1,2,\ldots,N-1$. Such factorization can be accomplished by means of the techniques detailed in [14] and the approach described in this chapter, where the 8-point DCT is a showcase.

The proposed 1D DCT algorithm can be readily extended to the two dimensional (2D) case, thus being suitable for image processing applications. Indeed, because of DCT kernel separability, the 2D DCT can be computed by successive calls of the 1D DCT. This can be achieved in a two-step procedure: (i) computation of the 1D DCT of each row of a given input image; and (ii) computation of the 1D DCT of each column of the image derived from step (i). The resulting image is the 2D DCT of the original image. Therefore, in principle, any 1D DCT algorithm can be

immediately extended to the 2D case without any additional modification [45].

Regarding the DCT computation over 2D domain, different image representation schemes can be employed in order to reduce the overall execution time. The works on [150, 151, 152, 153] consider a different image representation based on slice intensity representation (ISR). Although this representation can be useful for some scenarios such as pattern recongnition [154, 155, 156], we attain to the usual image representation adopted by the signal processing community [14, 45, 121].

4.5 Chapter Summary

In this chapter, we proposed a new method for the 8-point DCT computation based on the summation-by-parts formula. A matrix formalism was furnished and its arithmetic complexity was assessed.

The proposed method attained the theoretical multiplicative complexity lower bound for the computation of the *exact* DCT as detailed in [52]. The theoretical minimum is consists of 11 multiplications. *Per se*, achieving the minimum of 11 multiplications is not a trivial task. Not many DCT algorithms are capable of such, being the Loeffler DCT the most popular method in literature to achieve the minimum [103].

Moreover, the introduced method could also compute the scaled DCT with only *five* multiplications, matching the well-known Arai DCT [130]. Thus the proposed method can simultaneously match both Loeffler and Arai DCT in terms of multiplicative complexity. Not only it could attain the multiplicative complexity minimum, but it could also outperform several competing methods in three different scenarios when we also consider the additive complexity as a secondary comparison metric. In fact, it outperformed several popular DCT algorithms, when the input signal is assumed to possess null mean or/and it is an integrated (accumulated) signal.

The proposed design can be understood as a fundamental mathematical block to be considered for software and hardware realizations. Additionally, the field of integer and approximate

transforms can benefit of the proposed scheme [157, 158]. Indeed, the design of extremely low-complexity approximation methods relies on exact algorithms. Therefore, exact, algebraically precise methods can be a useful resource; even when an approximation could be enough [159].

Particularly, the proposed method is an alternative to Arai algorithm [130] for selected scenarios. The suggested method is suitable in the context of feature detection, where DC level may not be relevant [142] as well as in situations where traditional algorithms require data to be integrated, such as in face recognition problems [143, 144]. In future works, we aim at applying the summation-by-parts formula to different transforms at various blocklengths.

Chapter 5

Efficient Computation of Tridiagonal Matrices Largest Eigenvalue

5.1 Introduction

Matrix computations are relevant for a wide range of scientific problems [46]. In some applications, the matrices involved in the calculations have a specific format or distribution of non-zero elements [46], such as in signal processing [14] and geophysics [50]. A relevant type of matrix is the tridiagonal matrix. Tridiagonal matrices frequently occurs in problems related to the solution of linear systems associated with discretization of partial differential equations as proposed by J. Crank and P. Nicholson [160, 161, 162].

Some particular applications requires the computation of the associated eigenvalues with the tridiagonal matrices defining a linear system [46]. Frequently, the matrices involved in such a problem are symmetric, whose analytic expression for its eigenvalues and eigenvectors are known [160, 161].

Nevertheless, the involved tridiagonal matrices are not necessarily symmetric in some other applications. Some relevant quantities associated with linear systems defined by asymmetric tridiagonal matrices have been targeted in recent works [163, 164, 162].

There are several methods for solving eigenvalue problems. One may cite QR factorization methods [46], Schur decomposition [51], Jacobi diagonalization [165] and Krylov methods [51, 166]. Each of the several existing methods are better suitable for particular type of matrices. Iterative methods, such as Krylov space methods [46], are a possible choice when there the matrix involved have sparsity pattern that can be exploited. A particular Krylov space technique is the Power method. The Power method [46] requires a number of matrix-vector multiplications along

some additional floating-point operations that allows the matrix structure to be exploited.

In this work, we propose a way of speeding up the estimation of the largest eigenvalue of a tridiagonal matrix. The speedup is achieved by considering the square of the tridiagonal matrix of interest with the Power method. The adoption of the matrix square in the Power method provides a substantial reduction in the total number of iterations to about a half. The decrease of the total number of iterations can potentially reduces the overall computational error due to the computer finite precision arithmetic operations. In order to further reduce the overall arithmetic complexity, we propose a fast algorithm for computing tridiagonal matrix square.

Several works consider the case of dense matrix-matrix multiplication, such as the Coppersmith-Winograd algorithm [167] and its improved versions [168, 169]. These methods reduce the number of multiplications at the expense of a high number of additions, making them possess a very high multiplicative constant associated with the asymptotic complexity [170]. But because the matrices treated here have a particular tridiagonal structure and we are interested in the square, we resort to derive an algorithm that reduce both the number of additions and multiplication operations with floating-point quantities. Moreover, some works have considered the problem of computing the eigenvalues of symmetric tridiagonal matrices. The work in [171] proposed a method for partitioning tridiagonal symmetric matrices into several small symmetric tridiagonal matrices. The eigenvalues of the matrix of interest in [171] are computed through the solution of the characteristic polynomial of the several generated small tridiagonal symmetric matrices. Although there is some overhead associated with the partitioning and the original method was not numerically stable, later works considered some improvements [172, 173, 174, 175] turning it stable. In this work, we consider the case where the tridiagonal matrices can asymmetric.

The chapter unfolds as follows. Section 5.2 introduces the proposed method for fast evaluation of the largest eigenvalue of tridiagonal matrices. We evaluate the number of iterations of the proposed method compared to the Power method in worst scenario possible. Still on this section, we present an error analysis and show that the proposed method does not increase the computa-

tional error associated with the largest eigenvalue estimation compared to the Power method. In Section 5.3, we introduce a fast algorithm for the computation of tridiagonal matrix square. Implementation details related to compressed storage formats are discussed. In Section 5.4, we provide experiment results testifying the efficiency of the proposed method. We also provide comparison of the proposed squaring algorithm when compared to BLAS level 3 matrix multiplication implemented on a well-known scientific library. Section 5.5 has concluding remarks.

This chapter substantially improves the previously reported (MatLab only) findings from [176]. It also contains new results on various matrix encodings, different performance metrics and offers an extended error analysis.

5.2 Power Method Improvement for Tridiagonal Matrices

Let $\mathbf{A}$ be a square matrix of size $n \times n$ with real entries. The well-known Power method [46, 51, 166] is an iterative process that has as output the largest eigenvalue $\lambda_{\max}$ and its associated eigenvector $\mathbf{v}_{\lambda_{\max}}$ of a particular matrix $\mathbf{A}$. It requires an initial guess or approximation $\mathbf{v}_{\text{init}}$ for the eigenvector of the input matrix $\mathbf{A}$ associated with its largest eigenvalue. Every iteration of the Power method requires a matrix-vector operation along some additional arithmetic operations.

In general, the matrix-vector product requires n^2 real multiplications plus $n(n-1)$ additions. In particular, if the matrix is tridiagonal, each matrix-vector product demands only $3n-2$ real multiplications with $2n-2$ additions. Because of the vector scaling in each iteration, the Power method demands more n divisions in each iteration [46, 51]. At total, the Power method requires roughly $6n-4$ floating-point operations per iteration.

Let now $\mathbf{A}$ also be a tridiagonal matrix, i.e., $[\mathbf{A}]_{i,j} = 0$ if $|i-j| > 1$. Let us assume that the execution of the Power method [46, 51] applied to $\mathbf{A}$ requires a total of s steps for reaching a user prescribed precision ε. Because of the matrix-vector and divisions in each iteration, the Power method requires approximately $s(6n-4)$ real arithmetic operations in order to estimate $\lambda_{\max}$.

The square of tridiagonal matrices are pentadiagonal matrices of the same size. The straight-

forward computation of $\mathbf{A}^2$ requires $9n-10$ real multiplication plus $4n-4$ additions. We propose to use $\mathbf{A}^2$ instead of $\mathbf{A}$ to the estimate $\lambda_{\max}$ along with a fast algorithm in order to compute $\mathbf{A}^2$.

Using $\mathbf{A}^2$ instead $\mathbf{A}$ in the Power method requires at most $(s+1)/2$ steps, if s is odd. We show this result in the following theorem.

Theorem 1 *Let $\mathbf{A}$ be an $n \times n$ real matrix. If the Power method for estimating the largest eigenvalue λ_{max} associated with $\mathbf{A}$ with an user prescribed precision ε requires s iterations, the Power method with the same precision with the substitution of $\mathbf{A}$ by its square $\mathbf{A}^2$ will require at most $\lfloor s/2 \rfloor$ iterations.*

Proof: Let the sequence of estimates for $\mathbf{v}_{\lambda_{max}}$ and λ_{max} using the Power method be $\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_s$ and $\lambda_1, \lambda_2, \ldots, \lambda_s$, respectively. The kth eigenvector estimate can be represented as a function of the $k-1$ and $k-2$th estimates as follows:

$$\begin{aligned}\mathbf{v}_k &= \mathbf{A}\mathbf{v}_{k-1}/\lambda_{k-1} \\ &= \mathbf{A}\left(\mathbf{A}\mathbf{v}_{k-2}/\lambda_{k-2}\right)/\lambda_{k-1} \\ &= \mathbf{A}^2\mathbf{v}_{k-2}/\left(\lambda_{k-2}\lambda_{k-1}\right).\end{aligned}$$

Repeating the process $k-2$ times and setting $k = s$, we have that

$$\mathbf{v}_s = \mathbf{A}^s\mathbf{v}_{init}/\left(\prod_{i=1}^{s}\lambda_i\right).$$

If s is even, we can write

$$\mathbf{v}_s = \left(\mathbf{A}^2\right)^{s/2}\mathbf{v}_{init}/\left(\prod_{i=1}^{s}\lambda_i\right).$$

Otherwise, we can write

$$\mathbf{v}_s \approx \mathbf{v}_{s+1} = \left(\mathbf{A}^2\right)^{(s+1)/2}\mathbf{v}_{init}/\left(\prod_{i=1}^{s+1}\lambda_i\right),$$

where $\lambda_{s+1} = \|\mathbf{A}\mathbf{v}_s/\lambda_s\|_\infty$. Note that if s is even, substituting $\mathbf{A}$ by $\mathbf{A}^2$ in the Power method, we achieve convergence in exactly $s/2$ iterations. The worst occurs when s is odd, when the convergence is achieved in $(s+1)/2$ iterations. In either case, it corresponds to $\lfloor s/2 \rfloor$. □

Note that above theorem is verified assuming that we compute the Power method in a machine with infinite precision. Obviously, in a finite precision machine such as digital computers, this is verified in approximate form.

5.2.1 Worst Case Analysis for the Number of Iterations

We adopt the arithmetic complexity as figure of merit in order to show when the proposed method is advantageous over the usual Power method. Every matrix-vector multiplication for a tridiagonal matrix $\mathbf{A}$ of size n requires $5n-4$ floating-point operations. Besides that, every scaling of a vector of size n demands more n floating-point operations. Thus, if the usual Power method requires s steps for convergence, it will demand a total of $s(6n-4)$ floating-point operations.

For the proposed method, the total number of operations will be given by the arithmetic operations required to compute $\mathbf{A}^2$ plus the number of operations required to perform the matrix-vector product in each iteration and the respective vector scaling. In order to compute $\mathbf{A}^2$, we need $13n-14$ arithmetic operations. Since the squared matrix $\mathbf{A}^2$ is pentadiagonal, the matrix-vector product operation in each operation will require $5n-6$ multiplications and $4n-6$ additions, resulting in $9n-12$ operations per iteration for each matrix-vector operation. Using the result established by Theorem 1, we know that in the worst case, the number of iterations for the Power method using $\mathbf{A}^2$ will be $(s+1)/2$. The vector scaling accounts for more n floating-point divisions in each iteration. Thus, the total arithmetic complexity for the proposed method in the worst case is $(s+1)(10n-12)/2+13n-14$.

In order to the proposed method be advantageous over the usual approach, we need that the total number of operations required by the proposed method be smaller than the number of operations required by the usual Power method. This happens when

$$s(6n-4) \geq \frac{s+1}{2}(10n-12)+13n-14.$$

This is achieved when s is greater than a minimum threshold $s_{\min}$

$$s \geq s_{\min} = \left\lceil \frac{18n-20}{n+2} \right\rceil. \tag{5.1}$$

In the limit, we have that $s_{\min} = 18$. Note that this limit is reached even for relative small matrices, e.g., $n = 10^2$. This is of particular importance because even with small sized matrices, the proposed method can be advantageous.

Note that the computational complexity of the Power method is not entirely dependent only on arithmetic operations. In fact, each iteration requires the computation of the infinity norm of a vector of size n. Due its nature, the infinity norm requires a search over all the elements of its vector argument. Obviously, this search requires no arithmetic operation, but it does have a computational overhead associated with it due to memory access. By reducing the total number of iterations, we decrease the number of searches over the eigenvector estimates. This reduction contributes to further speedup of the proposed method compared to the usual Power method.

Also, the computation of the matrix square when $\mathbf{A}$ is tridiagonal is not a problem for memory management. In fact, because $\mathbf{A}^2$ is pentadiagonal, its storage may requires just a few kylo bytes even for relatively large matrices.

5.2.2 Error Analysis

In order to evaluate the numerical error imposed by the proposed method, we follow the methodology in [47] and the notation described in [177, 51]. Let $\mathbf{v}_k$ and λ_k be the kth estimates for $\mathbf{v}_{\lambda_{\max}}$ and $\lambda_{\max}$, respectively, where $\mathbf{v}_0 = \mathbf{A}\mathbf{v}_{\text{init}}$ and $\lambda_0 = \|\mathbf{v}_0\|_\infty$ and $k = 1, 2, \ldots, s$. Let also δ and θ represent the averaged percentage error due to the product and division of two floating-point numbers, respectively. Because of the numerical roundoff errors due to machine finite precision, the computed value $fl(\mathbf{v}_0)$ of the matrix-vector product between $\mathbf{A}$ and the vector $\mathbf{v}_{\text{init}}$ and the first eigenvalue estimate λ_0 are approximately $fl(\mathbf{A}\mathbf{v}_{\text{init}}) = (1+\delta)\mathbf{v}_0$ and $fl(\lambda_0) = (1+\delta)\lambda_0$. The second estimate for $\mathbf{v}_{\lambda_{\max}}$ and $\lambda_{\max}$ are given by $fl(\mathbf{v}_1) = (1+\delta)^2(1+\theta)\mathbf{v}_1$ and $fl(\lambda_1) = (1+\delta)^2(1+\theta)\lambda_1$, respectively. Note that in each iteration, we have a percentage contribution of about $(1+\delta)(1+\theta)$ to the computational error due to roundoff operations. For each iteration of the Power method, we have a matrix-vector product and a division by a non-null real quantity $\lambda_k = \|\mathbf{v}_k\|_\infty$. If the usual Power method converge for the user specified tolerance within s steps, we have that

$fl(\mathbf{v}_s) = (1+\delta)^s(1+\theta)^{(s-1)}\mathbf{v}_s$ and $fl(\lambda_s) = (1+\delta)^s(1+\theta)^{(s-1)}\lambda_s$. These expressions imply in the following approximations:

$$\|fl(\mathbf{v}_s) - \mathbf{v}_s\| \approx \|s(\delta+\theta)\mathbf{v}_s\|,$$

and

$$\|fl(\lambda_s) - \lambda_s\| \approx \|s(\delta+\theta)\lambda_s\|,$$

where we assume $s(\delta+\theta) << 1$ and $\|\cdot\|$ can represent any vector norm [51].

According Theorem 1, the proposed method achieve the user prescribed precision in at most $(s+1)/2$ steps. However, because the matrix square is computed and the error is propagated through the subsequent calculations, we have that $fl(\mathbf{A}^2) = (1+\delta)\mathbf{A}^2$. As consequence, in each iteration, we have a averaged relative contribution of $(1+\delta)^2(1+\theta)$ to the computational error due to roundoff operations. This is contrasting with the $(1+\delta)(1+\theta)$ contribution furnished by the usual Power method. However, because the proposed method achieve convergence in at most $(s+1)/2$ iterations–that is about a half–we have that

$$fl(\tilde{\mathbf{v}}_{(s+1)/2}) = (1+\delta)^{2(s+1)/2}(1+\theta)^{(s-1)/2}\tilde{\mathbf{v}}_{(s+1)/2},$$

and

$$fl(\tilde{\lambda}_{(s+1)/2}) = (1+\delta)^{2(s+1)/2}(1+\theta)^{(s-1)/2}\tilde{\lambda}_{(s+1)/2}.$$

The computational error is then approximately

$$\|fl(\tilde{\mathbf{v}}_{(s+1)/2}) - \tilde{\mathbf{v}}_{(s+1)/2}\| \approx \left\|s(\delta+\theta/2)\tilde{\mathbf{v}}_{(s+1)/2}\right\|, \tag{5.2}$$

and

$$\|fl(\tilde{\lambda}_{(s+1)/2}) - \tilde{\lambda}_{(s+1)/2}\| \approx \left\|s(\delta+\theta/2)\tilde{\lambda}_{(s+1)/2}\right\|. \tag{5.3}$$

Because the proposed method achieve convergence with at most $(s+1)/2$, we can assume that $\tilde{\lambda}_{(s+1)/2} \approx \lambda_s$ and $\tilde{\mathbf{v}}_{(s+1)/2} \approx \mathbf{v}_s$. Restricting ourselves to the l^2 norm for (5.2) and (5.3), we obtain

$$\|fl(\tilde{\mathbf{v}}_{(s+1)/2}) - \tilde{\mathbf{v}}_{(s+1)/2}\|_2 \lesssim \|fl(\mathbf{v}_s) - \mathbf{v}_s\|_2 \tag{5.4}$$

and

$$\|fl(\tilde{\lambda}_{(s+1)/2}) - \tilde{\lambda}_{(s+1)/2}\|_2 \lesssim \|fl(\lambda_s) - \lambda_s\|_2, \tag{5.5}$$

where $\lesssim$ denotes approximately smaller than. Therefore, both the computational error of the proposed method for estimating both the eigenvalue $\lambda_{\max}$ and its correspondent eigenvector $\mathbf{v}_{\lambda_{\max}}$ have the same magnitude of error of the usual Power method [46].

5.3 Fast Algorithm for Tridiagonal Matrix Squaring

The method for speeding up the Power method is based in the squaring of the tridiagonal matrix under analysis. In order to further reduce the arithmetic complexity, we provide a fast algorithm for computing the square of a tridiagonal matrix. First, let us have a look on the 3×5 matrix block centred at the a_i, the element at position (i,i), with $2 < i < n-1$:

$$\begin{bmatrix} c_{i-2} & a_{i-1} & b_{i-1} & 0 & 0 \\ 0 & c_{i-1} & a_i & b_i & 0 \\ 0 & 0 & c_i & a_{i+1} & b_{i+1} \end{bmatrix}.$$

The non-zero elements of the ith row of squared matrix $\mathbf{A}^2$ are

$$\begin{aligned} \left[\mathbf{A}^2\right]_{i,i-2} &= c_{i-2}c_{i-1}, \\ \left[\mathbf{A}^2\right]_{i,i-1} &= a_{i-1}c_{i-1} + a_i c_{i-1}, \\ \left[\mathbf{A}^2\right]_{i,i} &= b_{i-1}c_{i-1} + a_i^2 + b_i c_i, \\ \left[\mathbf{A}^2\right]_{i,i+1} &= a_i b_i + a_{i+1} b_i, \\ \left[\mathbf{A}^2\right]_{i,i+2} &= b_i b_{i+1}. \end{aligned}$$

The direct computation of these elements requires 9 multiplications and 4 additions. The number of real multiplications can be reduced to 6 for each of the lines $3,4,\ldots,n-2$ as follows.

The second and fourth non-null element of ith row can be represented as

$$(a_{i-1} + a_i)c_{i-1} \qquad \text{and} \qquad (a_i + a_{i+1})b_i.$$

This saves only 2 multiplications per row. If we write the expression for the diagonal element in the $i-1$th row, we would have

$$b_{i-2}c_{i-2} + a_{i-1}^2 + b_{i-1}c_{i-1}.$$

It is noteworthy the fact that the product $b_{i-1}c_{i-1}$ occurs both in the $i-1$th and ith diagonal elements of $\mathbf{A}^2$. This saves one more multiplication for each of the rows $3,4,\ldots,n-2$.

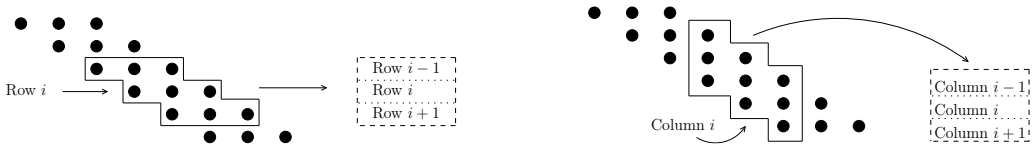
The extreme rows $1,2,n-1$ and n have to be treated separately. For the first two lines of $\mathbf{A}^2$, we have

$$\begin{bmatrix} a_1^2+b_1c_1 & a_1b_1+a_2b_1 & b_1b_2 & 0 & 0 \ldots 0 \\ a_1c_1+a_2c_1 & b_1c_1+a_2^2+b_2c_2 & a_2b_2+a_3b_2 & b_2b_3 & 0 \ldots 0 \end{bmatrix}$$

The elements at positions $(1,2)$, $(2,1)$ and $(2,3)$ can be written as $(a_1+a_2)b_1$, $(a_1+a_2)c_1$ and $(a_2+a_3)b_2$, respectively. This saves more 3 multiplications. Also, since the term b_1c_1 appear both in the first and second diagonal element, we can save another multiplication. By adopting this strategy, we need only 9 multiplications to compute the first two rows of $\mathbf{A}^2$. The same strategy can be employed for the last two rows of $\mathbf{A}^2$, saving more four multiplications.

While the direct computation requires $13n-14$ arithmetic operations, the method outlined above achieve the same result with only $9n-8$ floating-point operations. Note that this is possible only because this strategy reduces both the number of additions and multiplications. This is not a common situation in methods for fast algorithms, since the reduction of one type of arithmetic operation usually implies in the increase of others (e.g., reduction in the number of multiplications implies in the increase of additions [14]).

Among all the compressed storage formats, the compressed row storage (CRS) and compressed column storage (CCS) formats are the most well known. For every input matrix in CRS (CCS) format that we are interested in computing the matrix square, we read its rows (columns) at a time for computing the elements in the corresponding row (column) in the squared matrix. The CRS (CCS) format represents each matrix with three vectors: the vector `p` that store the non-zero elements of the sparse matrix; the vector `i` that store the column (row) positions of the elements in `p`; and the vector `j` that stores the index of the elements on each row (column) on the vector `p` [46].



(a) Reading for CRS format. (b) Reading for CCS format.

Figure 5.1: A graphical representation of how the data tridiagonal matrices in CRS and CCS formats are read.

The number of non-zero elements of that particular matrix is denoted `nz`.

In order to access the row (column) elements, we read the elements and store them in a temporary 3×3 matrix. The rows (columns) of $\mathbf{A}$ are stored in the rows of the temporary 3×3 matrix and the elements of each row (column) of $\mathbf{A}^2$ are computed as the rows (column) of $\mathbf{A}$ are read. This scheme is represented in Figure 5.1(a) and Figure 5.1(b) for CRS and CCS formats, respectively. Because of the way we represent compressed storage formats in current computers and the proposed method for computing the square of a tridiagonal matrix, the provided implementation works both for CRS and CCS formats[1] and it is described in a high level manner in Algorithm 1.

We denote the vectors `p`, `i`, and `j` representing a particular matrix $\mathbf{M}$ as $\mathbf{M}$`->p`, $\mathbf{M}$`->i`, $\mathbf{M}$`->j` and the number of non-zeros as $\mathbf{M}$`->nz`. We denote the insertion of a real element a in a particular sparse compressed matrix $\mathbf{M}$ at row and column (column and row) (i, j) as insert$(\mathbf{M}, a, i, j)$ and its description can be found at [46]. Because the matrix elements in compressed format representation are stored in contiguous memory, the Algorithm 1 will have a low cache miss rate. This is specially important for high performance applications.

5.4 Computer Simulation

The proposed method for estimation of largest eigenvalue was implemented in a Core i7 microprocessor with 8GB of RAM memory. We adopted C/C+ language and the GNU Scientific Library (GSL) for implementation of the routines needed for estimating the largest eigenvalue of a tridia-

[1]The C/C++ implementation of the proposed tridiagonal matrix squaring can be found at *https://github.com/diegofgcoelho/tridiagonal*.

Algorithm 1: The pseudocode for the fast tridiagonal matrix squaring algorithm.

Input: Square tridiagonal compressed matrix $\mathbf{A}$ of size n
Output: Square pentadiagonal compressed matrix $\mathbf{B} = \mathbf{A}^2$ of size n

Define temporary matrix $\mathbf{M} = [M_{ij}] = 0$ for $i, j = 1, 2, 3$

Initialize $\mathbf{B}$ as a null square matrix of order n

Do $M_{ij} \leftarrow A_{ij}$ for $i, j = 1, 2$ and 3

Compute the auxiliary quantities
$m_1 \leftarrow M_{21} \cdot M_{12}$ $\quad m_2 \leftarrow M_{31} \cdot M_{23}$
$s_1 \leftarrow M_{11} + M_{22}$ $\quad s_2 \leftarrow M_{22} + M_{32}$

Compute the elements on row (column) 1
insert$(\mathbf{B}, M_{11}^2 + m_1, 1, 1)$
insert$(\mathbf{B}, s_1 \cdot M_{12}, 1, 2)$
insert$(\mathbf{B}, M_{12} \cdot M_{23}, 1, 3)$

Compute the elements on row (column) 2
insert$(\mathbf{B}, s_1 \cdot M_{21}, 2, 1)$
insert$(\mathbf{B}, m_1 + M_{22}^2 + m_2, 2, 2)$
insert$(\mathbf{B}, s_2 \cdot M_{23}, 2, 3)$
insert$(\mathbf{B}, M_{23} \cdot M_{33}, 2, 4)$

for $i = 3, 4, \ldots, n-2$ **do**
  Update temporary matrix $\mathbf{M}$:
  $\mathbf{M}_{1\cdot} \leftarrow \mathbf{M}_{2\cdot}$ $\quad \mathbf{M}_{2\cdot} \leftarrow \mathbf{M}_{3\cdot}$ $\quad \mathbf{M}_{3\cdot} \leftarrow \mathbf{A}_{i\cdot}$
  Compute the auxiliary quantities for row (column) i
  $s_1 \leftarrow s_2$ $\quad s_2 \leftarrow M_{32} + M_{22}$
  $m_1 \leftarrow m_2$ $\quad m_2 \leftarrow M_{23} \cdot M_{31}$
  Compute the elements on row (column) i
  insert$(\mathbf{B}, M_{11} \cdot M_{21}, i, i-2)$
  insert$(\mathbf{B}, s_1 \cdot M_{21}, i, i-1)$
  insert$(\mathbf{B}, m_1 + M_{22}^2 + m_2, i, i)$
  insert$(\mathbf{B}, s_2 \cdot M_{23}, i, i+1)$
  insert$(\mathbf{B}, M_{23} \cdot M_{33}, i, i+2)$
end for

Update temporary matrix $\mathbf{M}$:
$\mathbf{M}_{1\cdot} \leftarrow \mathbf{M}_{2\cdot}$ $\quad \mathbf{M}_{2\cdot} \leftarrow \mathbf{M}_{3\cdot}$ $\quad \mathbf{M}_{3\cdot} \leftarrow \mathbf{A}_{n\cdot}$
Compute the auxiliary quantities
$m_1 \leftarrow m_2$ $\quad m_2 \leftarrow M_{23} \cdot M_{32}$
$s_1 \leftarrow s_2$ $\quad s_2 \leftarrow M_{22} + M_{33}$

Compute the elements on row $n-1$
insert$(\mathbf{B}, M_{11} \cdot M_{21}, n-1, n-3)$
insert$(\mathbf{B}, s_1 \cdot M_{21}, n-1, n-2)$
insert$(\mathbf{B}, m_1 + M_{22}^2 + m_2, n-1, n-1)$
insert$(\mathbf{B}, s_2 \cdot M_{23}, n-1, n)$

Compute the elements on row n
insert$(\mathbf{B}, M_{21} \cdot M_{32}, n, n-2)$
insert$(\mathbf{B}, s_2 \cdot M_{32}, n, n-1)$
insert$(\mathbf{B}, m_2 + M_{33}^2, n, n)$

Table 5.1: Average number of iterations and time for Power method execution tridiagonal matrices of sizes $100, 300, 500, 700, 1000$ with replicates of size 1000 using Kac–Sylvester test matrices.

Matrix	Power Method			Proposed Method			Improvement
n	μ_i	μ_t	ε	μ_i	μ_t	ε	Factor
100	174.00	1.09	$3.43 \cdot 10^{-3}$	86.00	0.57	$4.81 \cdot 10^{-4}$	1.92
300	525.00	7.23	$7.92 \cdot 10^{-4}$	263.00	3.83	$5.47 \cdot 10^{-4}$	1.89
500	874.00	17.49	$4.79 \cdot 10^{-4}$	437.00	9.22	$5.57 \cdot 10^{-4}$	1.90
700	1222.00	36.00	$7.00 \cdot 10^{-4}$	611.00	18.83	$5.58 \cdot 10^{-4}$	1.91
1000	1735.00	76.60	$7.02 \cdot 10^{-4}$	868.00	39.93	$5.54 \cdot 10^{-4}$	1.92

gonal matrix and its associated eigenvector. The initial guesses were set using usual uniform pseudorandom number generator through the function `gsl_rng_uniform` with the Mersenne Twister algorithm [178]. For fair comparison, both methods were executed with the same initial guess. We adopted the Sylvester–Kac test matrix [179] for comparing the Power method and the proposed method performance. We used matrix sizes ranging from 100 to 1000 with 1000 replicates. The eigenvalues and eigenvectors of Sylvester–Kac matrices are known, and then used to compare the numerical errors of the usual Power method and the proposed method for the largest eigenvalue estimation. Table 5.1 shows the average number (μ_i) of iterations considering an user prescribed precision of 10^{-3}. Also, Table 5.1 shows average time required by each method in milliseconds (μ_t), the averaged numerical error of the estimated eigenvalues and improvement factors based on the ratio of averaged times μ_t. We adopted the CCS format in the simulations for generating the data in Table 5.1. Note that the average number of iteration μ_i of the proposed method for all considered sizes are about a half of the usual Power method. Accordingly, the average time μ_t required by the proposed method is also about a half of the time required by the usual Power method. Note that the numerical error induced by the proposed method for estimating $\lambda_{\max}$ is in average slightly smaller than the usual Power method. This can be justified by the error analysis in Section 5.2.2, specifically, the expressions in (5.4) and (5.5). Considering the same parameters, we performed the simulations with randomly generated tridiagonal matrices as well. The entries

Table 5.2: Average number of iterations and time for Power method execution tridiagonal matrices of sizes $100, 300, 500, 700, 1000$ with replicates of size 1000 using random generated tridiagonal matrices.

Matrix	Power Method			Proposed Method			Improvement
n	μ_i	μ_t (ms)	ε	μ_i	μ_t (ms)	ε	Factor
100	38.00	0.23	$8.97 \cdot 10^{-4}$	19.00	0.13	$7.73 \cdot 10^{-4}$	1.73
300	71.00	1.16	$9.67 \cdot 10^{-4}$	35.00	0.64	$9.67 \cdot 10^{-4}$	1.81
500	94.00	2.22	$9.99 \cdot 10^{-4}$	47.00	1.25	$9.48 \cdot 10^{-4}$	1.77
700	84.00	2.86	$9.82 \cdot 10^{-4}$	42.00	1.61	$9.09 \cdot 10^{-4}$	1.78
1000	113.00	5.24	$9.46 \cdot 10^{-4}$	56.00	2.91	$9.46 \cdot 10^{-4}$	1.80

Table 5.3: Total time in milliseconds for matrix squaring for the BLAS level 3 matrix multiplication `gsl_spmatrix_dgemm` for CCS format, the proposed fast algorithm considering CRS and CCS formats, respectively, for random generated matrices of sizes $100, 300, 500, 700, 1000$ with replicates of size 1000.

Matrix size n	GSL function `gsl_spmatrix_dgemm`	Proposed Algorithm for CRS	Proposed Algorithm for CCS
100	0.650	0.287	0.261
300	1.322	0.555	0.499
500	2.110	0.840	0.780
700	3.101	1.140	1.091
1000	5.005	1.539	1.408

of the input tridiagonal matrices were generated considering the Mersenne Twister pseudorandom number generator. The Table 5.2 shows the results for such a simulation, where now ε represents the relative norm of the difference between the last consecutive eigenvector estimates. Note that the numerical values on Table 5.2 agree with the findings on Table 5.1 and the theoretical results.

We also performed a comparison solely based on the matrix squaring algorithms for CCS format against the BLAS level 3 operation implementation of sparse matrix multiplications through the routine `gsl_spmatrix_dgemm`. Table 5.3 shows the total time required for the implementations considering several matrix sizes with 1000 replicates. Note that the implementation of the

proposed fast algorithm executes in less than a half of the `gsl_spmatrix_dgemm`. This is more than what we expected and might be due to the overhead reduction

5.5 Chapter Summary

In this chapter we proposed an improved way to estimate the dominant eigenvalue of arbitrary tridiagonal matrices. The method is based on the computation of tridiagonal matrix squaring and its fast algorithm. The proposed fast algorithm for matrix squaring reduces both the number of multiplications and additions, circumstances not always found in numerical problems requiring fast algorithms [14]. We have shown that the computation of the matrix square provides faster convergence for the Power method, thus reducing the total number of floating-point operations. The proposed method was implemented in C/C++ language and compared to an standard matrix multiplication routine in the C/C++ scientific library GSL. We verified that the proposed method achieve convergence in about a half of the number of iterations and the time required by the original Power method [46].

Future works include the extension of the present work to the estimation of the largest eigenvalue and its corresponding eigenvector for pentadiagonal matrices and general band-limited matrices. Future works may also include a parallel implementation of the proposed tridiagonal matrix squaring algorithm and the use of Weilandt deflation.

Chapter 6

Fast Matrix Inversion and Determinant Computation for Polarimetric Synthetic Aperture Radar

6.1 Introduction

Microwave remote sensing is basilar as it provides complementary information to that provided by classical sensors which perceive the optical spectrum. Longer wavelengths in the 1 cm to the 1 m can penetrate clouds and other adverse atmospheric conditions, as well as canopies and soils. There are passive and active microwave sensors; the latter carry their own source of illumination, and can be either imaging or non-imaging. Radar devices belong to the former. They transmit a radio signal and detect the returned echo (backscatter), with which an image is formed. Several signal processing techniques are used to enhance the spatial resolution of such imagery. In particular, the use of synthetic antennas leads to synthetic aperture radar (SAR) imaging; as well as to polarimetric SAR (PolSAR) imaging which employs polarized signals

The statistical properties of PolSAR were studied in the context of optical polarimetry [180]. The simplest case occurs when the backscatter is constant. In such a case, the targets are characterized by a scattering (or Sinclair) matrix $\mathbf{S}$, which describes dependence of its scattering properties on the polarization. The scattering matrix is defined on the horizontal (H) and vertical (V) basis as

$$\mathbf{S}' = \begin{bmatrix} S_{\mathrm{HH}} & S_{\mathrm{HV}} \\ S_{\mathrm{VH}} & S_{\mathrm{VV}} \end{bmatrix}, \tag{6.1}$$

where each element is a complex value, representing the amplitude and the phase of the scattered signal. Reciprocity $S_{\mathrm{HV}} = S_{\mathrm{VH}}$ holds for most natural targets, so one may use the scattering vector $\mathbf{S} = \begin{bmatrix} S_{\mathrm{HH}} & S_{\mathrm{HV}} & S_{\mathrm{VV}} \end{bmatrix}^{\top}$ without loss of information, where $^{\top}$ represents transposition [181].

More often than not, these single-look complex data are processed in order to improve the

signal-to-noise ratio. Multilook fully polarimetric data are formed as

$$Z^{(N)} = \frac{1}{N}\sum_{\ell=1}^{N} \mathbf{S}(\ell)^{\mathsf{H}}\,\mathbf{S}(\ell), \tag{6.2}$$

where $^{\mathsf{H}}$ denotes the conjugate transposition and ℓ indexes theoretically independent looks of the same scene. This 3×3 complex matrix is positive Hermitian with real entries in the diagonal.

As noted by Torres *et al.* [39] and the references therein, many techniques for PolSAR image processing and analysis rely on the statistical properties of $Z^{(N)}$. The density of several models (Wishart [182], K [183], G^0 [184], G [185], Kummer-U [186] to name a few) depends solely on a few parameters: the covariance matrix, the number of looks L and, sometimes, texture descriptors. The covariance matrix is the expected value of $Z^{(N)}$, namely $\mathbf{Z} = \mathrm{E}\{Z^{(N)}\}$, and it enters the expressions only through its determinant and its inverse.

Moreover, divergences among these models (Kullback-Leibler, Hellinger, Rényi, Bhattacharya, Triangular, Harmonic [187]), test statistics (likelihood ratio) [42], and classification rules [188] only require the computation of $\det(\mathbf{Z})$, $\mathbf{Z}^{-1}$, and $\det(\mathbf{Z}^{-1})$. A typical Uninhabited Aerial Vehicle Synthetic Aperture Radar (UAVSAR) image may have $10^4\times 4\cdot 10^4$ pixels, where each pixel is represented by a 3×3 Hermitian matrix as in (6.2). As illustrative example, if one relies on Nonlocal Means Filter approach based on stochastic distances [39], at each sear window, which are typically large, e.g. 23×23 pixels, one needs to compute 23^2 inverse and 23^2 determinant operations for each pixel. This scales to a long computing time as it results in a total of $23^2\cdot 4\cdot 10^8 \approx 2.1\cdot 10^{11}$ inverse matrix and determinant calculations *per image*. Such amount of data is likely to increase with the incoming availability of sensors with finer resolution. Thus one reaches the conclusion that accurate and fast procedures are of paramount importance for dealing with PolSAR data.

The Cholesky factorization is the most popular numerical analysis method for the direct solution of linear algebra tasks involving positive definite dense matrices [51, 46, 189]. It is also the algorithm of choice for matrix inversion and determinant calculation in the context of image classification of PolSAR images [39]. In this chapter, we propose a fast algorithm for the computation of

matrix inverse and determinant of 3×3 Hermitian matrices in the context of PolSAR image classification that outperforms the Cholesky factorization. The introduced algorithm is proven to reduce the overall arithmetic complexity associated with the matrix inversion and determinant calculation when compared with the Cholesky factorization approach. Such lower arithmetic cost results in a reduction of the computation time to about a half of the Cholesky based method. The proposed algorithm and the matrix inversion and determinant calculation based on Cholesky factorization are implemented in a general purpose graphical processing unit (GPGPU) using C/C++ and open computing language (OpenCL), which are tools that have been used for accelerating a several of algorithms in geoscience and remote sensing [190, 191, 192, 193]. The proposed algorithm and the method based on Cholesky factorization are tested using both simulated and measured PolSAR data [194].

The chapter unfolds as follows. Section 6.2 introduces notation and preliminary considerations. The Cholesky factorization is reviewed and described as a means for matrix inversion and determinant computation. In Section 6.3, the proposed method and its fast algorithm is introduced. Section 6.3 brings the arithmetic complexity assessment of the proposed method compared with the Cholesky factorization approach. Section 6.4 has implementation considerations including software and hardware comments. Experiment results shows the effectiveness of the proposed method. Final comments and suggested directions for future works are in Section 6.5.

6.2 Mathematical Review

6.2.1 Preliminaries and Notation

The type of matrix that occurs in the PolSAR problem is the 3×3 correlation matrix with complex entries. Being a correlation matrix, it inherits the Hermitian property [46]. Therefore, it can be represented according to:

$$\mathbf{A} = \begin{bmatrix} a & b & c \\ \bar{b} & d & e \\ \bar{c} & \bar{e} & f \end{bmatrix},$$

where a, b, and c are real quantities; b, c, and e are complex numbers; and the overbar $\bar{}$ denotes the complex conjugation. Because $\mathbf{A}$ is Hermitian, in actual implementations, only the upper or lower triangular part of the matrix is stored for computation.

The (i, j) cofactor of the matrix $\mathbf{A}$ is the determinant of the submatrix formed with the elimination of the ith row and jth column times $(-1)^{i+j}$ [46]. The adjoint matrix is computed according to the following [46]:

$$\tilde{\mathbf{A}} = \begin{bmatrix} a_c & b_c & c_c \\ \bar{b}_c & d_c & e_c \\ \bar{c}_c & \bar{e}_c & f_c \end{bmatrix},$$

where the element in position (i, j) represents the cofactor of the element (j, i) in the original matrix $\mathbf{A}$ and are given by

$$\begin{aligned} a_c &= d \cdot f - |e|^2, \\ b_c &= c \cdot \bar{e} - b \cdot f, \\ c_c &= b \cdot e - c \cdot d, \\ d_c &= a \cdot f - |c|^2, \\ e_c &= c \cdot \bar{b} - a \cdot e, \\ f_c &= a \cdot d - |b|^2. \end{aligned} \tag{6.3}$$

The adjoint matrix $\tilde{\mathbf{A}}$ is also Hermitian; thus a_c, d_c, and f_c are real numbers.

6.2.2 Cholesky Factorization

Cholesky factorization is applied in many numerical problems [195, 196, 197]. Let $\mathbf{A}$ be the input matrix we are interested into inverting and computing the determinant. Cholesky factorization decomposes an input matrix into the product $\mathbf{L} \cdot \mathbf{L}^{\mathsf{H}}$, where $\mathbf{L}$ is a lower triangular matrix and $^{\mathsf{H}}$ represents the transpose conjugate operation. Let $l_{i,j}$ be the (i, j) entry of $\mathbf{L}$. The Cholesky factorization is based on the following relations between the elements of $\mathbf{A}$ and $\mathbf{L}$:

$$
\begin{aligned}
|l_{0,0}|^2 &= a, \\
l_{1,0} \cdot l_{0,0} &= \bar{b}, \\
l_{2,0} \cdot l_{0,0} &= \bar{c}, \\
|l_{1,1}|^2 + |l_{1,0}|^2 &= d, \\
l_{2,1} \cdot l_{1,1} + l_{1,0} \cdot \bar{l}_{2,0} &= e, \\
|l_{2,2}|^2 + |l_{2,0}|^2 + |l_{2,1}|^2 &= f,
\end{aligned}
$$

where $|\cdot|$ returns the magnitude of its complex argument [26].

Once matrix $\mathbf{L}$ is derived, its inverse can be directly obtained from the following expressions:

$$
\mathbf{L}^{-1} = \begin{bmatrix}
\frac{1}{l_{0,0}} & 0 & 0 \\
\frac{1}{l_{1,1}} \cdot \left(1 - \frac{l_{1,0}}{l_{0,0}}\right) & \frac{1}{l_{1,1}} & 0 \\
\frac{1}{l_{2,2}} \cdot \left(\frac{l_{2,1} \cdot l_{1,0}}{l_{1,1} \cdot l_{0,0}} - \frac{l_{2,0}}{l_{0,0}}\right) & -\frac{l_{2,1}}{l_{1,1}} \frac{1}{l_{2,2}} & \frac{1}{l_{2,2}}
\end{bmatrix}
$$

Given $\mathbf{L}^{-1}$, the inverse of the input matrix $\mathbf{A}$ is furnished by: $\mathbf{A}^{-1} = (\mathbf{L}^{-1})^{\mathsf{H}} \cdot \mathbf{L}^{-1}$. The determinant of $\mathbf{A}$, $\det(\mathbf{A})$, is also available as a by-product of the discussed factorization [46]. The detailed procedure for inverting 3×3 matrices based on Cholesky factorization is described in Algorithm 6.1.

Input: $a, d, f \in \mathbb{R}$ and $b, c, e \in \mathbb{C}$
Output: $a_i, d_i, f_i \in \mathbb{R}$ and $b_i, c_i, e_i \in \mathbb{C}$ and $\det(\mathbf{A})$

Compute $\mathbf{L}$ and auxiliary variables:

$$l_{0,0} \leftarrow \sqrt{a} \qquad v_0 \leftarrow 1/l_{0,0}$$
$$l_{1,0} \leftarrow \bar{b} \cdot v_0 \qquad l_{2,0} \leftarrow \bar{c} \cdot v_0$$
$$l_{1,1} \leftarrow \sqrt{d - |l_{1,0}|^2} \qquad v_1 \leftarrow 1/l_{1,1}$$
$$l_{2,1} \leftarrow (e - l_{1,0} \cdot \bar{l}_{2,0}) \cdot v_1 \qquad l_{2,2} \leftarrow \sqrt{f - |l_{2,0}|^2 - |l_{2,1}|^2}$$
$$v_2 \leftarrow 1/l_{2,2}$$

Compute $\mathbf{L}^{-1} = \{t_{i,j}\}$:

$$t_{0,0} \leftarrow v_0 \qquad t_{1,0} \leftarrow -l_{1,0} \cdot v_0 \cdot v_1$$
$$t_{1,1} \leftarrow v_1 \qquad t_{2,0} \leftarrow v_2 \cdot v_0 \cdot (l_{2,1} \cdot l_{1,0} v_1 - l_{2,0})$$
$$t_{2,1} \leftarrow -l_{2,1} \cdot v_1 \cdot v_2 \qquad t_{2,2} \leftarrow v_2$$

Compute the determinant: $\det(\mathbf{A}) \leftarrow (l_{0,0} \cdot l_{1,1} \cdot l_{2,2})^2$

Compute $\mathbf{A}^{-1}$:

$$a_i \leftarrow t_{0,0}^2 + |t_{1,0}|^2 + |t_{2,0}|^2 \qquad b_i \leftarrow \bar{t}_{1,0} \cdot t_{1,1} + \bar{t}_{2,0} \cdot t_{2,1}$$
$$c_i \leftarrow \bar{t}_{2,0} \cdot t_{2,2} \qquad d_i \leftarrow t_{1,1}^2 + |t_{2,1}|^2$$
$$e_i \leftarrow \bar{t}_{2,1} \cdot t_{2,2} \qquad f_i \leftarrow t_{2,2}^2$$

return $a_i, b_i, c_i, d_i, e_i, f_i$ and $\det(\mathbf{A})$

Figure 6.1: The matrix inversion and determinant calculation based on the Cholesky factorization.

6.3 Fast Inversion and Determinant Computation

6.3.1 Proposed Fast Algorithm

From matrix theory [198, p. 59], we know that

$$\mathbf{A}^{-1} = \frac{1}{\det(\mathbf{A})}\tilde{\mathbf{A}} \tag{6.4}$$

and

$$\det(\mathbf{A}) = a \cdot a_c + \bar{b} \cdot b_c + \bar{c} \cdot c_c. \tag{6.5}$$

Notice that (6.5) is a direct consequence of the fact that the determinant of a matrix can be obtained by the summation of any of the row or column elements weighted by their corresponding cofactors [32]. Notice that the first term in (6.5), aa_c, is purely real. Also notice that because $\mathbf{A}$ is Hermitian, its determinant is real-valued [46]. Therefore, the term $(\bar{b} \cdot b_c + \bar{c} \cdot c_c)$ must be real. As a consequence, we calculate only the real part of $\bar{b}b_c + \bar{c}c_c$ in order to compute $\det(\mathbf{A})$ as follows:

$$\begin{aligned} \det(\mathbf{A}) =& a \cdot a_c + \Re(b) \cdot \Re(b_c) + \Im(b) \cdot \Im(b_c) \\ &+ \Re(c) \cdot \Re(c_c) + \Im(c) \cdot \Im(c_c), \end{aligned} \tag{6.6}$$

where $\Re(\cdot)$ and $\Im(\cdot)$ return the real and imaginary parts of its complex argument, respectively.

Expressions (6.3), (6.4), and (6.6) can be combined to derive a fast algorithm for the computation of $\mathbf{A}^{-1}$ and $\det(\mathbf{A})$. Algorithm 6.2 details the efficient procedure. As input data, it requires only the upper triangular part of the input matrix $\mathbf{A}$. The proposed algorithm returns (i) the upper triangular part of $\mathbf{A}^{-1}$, which can be fully reconstructed due to the Hermitian property and (ii) inverse and determinant. The quantity $1/\det(\mathbf{A})$, which appears in all the aforementioned densities and divergences, is readily available as a by-product of Algorithm 6.2 under the auxiliary variable t.

6.3.2 Complexity Assessment and Discussion

Algorithm 6.1 requires the computation of the squared norm of six complex quantities. This demands 12 multiplications and six additions of real quantities [14]. Besides these operations, Al-

Input: $a, d, f \in \mathbb{R}$ and $b, c, e \in \mathbb{C}$
Output: $a_i, d_i, f_i \in \mathbb{R}$ and $b_i, c_i, e_i \in \mathbb{C}$ and $\det(\mathbf{A})$

Compute cofactors:

$$\begin{aligned} a_i &\leftarrow d \cdot f - |e|^2 & b_i &\leftarrow c \cdot \bar{e} - b \cdot f \\ c_i &\leftarrow b \cdot e - c \cdot d & d_i &\leftarrow a \cdot f - |c|^2 \\ e_i &\leftarrow c \cdot \bar{b} - a \cdot e & f_i &\leftarrow a \cdot d - |b|^2 \end{aligned}$$

Compute determinant:

$$\begin{aligned} \det(\mathbf{A}) \leftarrow a \cdot a_i + \Re(b) \cdot \Re(b_i) &+ \Im(b) \cdot \Im(b_i) \\ + \Re(c) \cdot \Re(c_i) + \Im(c) \cdot \Im(c_i) & \end{aligned}$$

Compute the inverse of the determinant: $t \leftarrow 1 / \det(\mathbf{A})$

Compute the $\mathbf{A}^{-1}$ entries:

$$\begin{aligned} a_i &\leftarrow t \cdot a_i & b_i &\leftarrow t \cdot b_i \\ c_i &\leftarrow t \cdot c_i & d_i &\leftarrow t \cdot d_i \\ e_i &\leftarrow t \cdot e_i & f_i &\leftarrow t \cdot f_i \end{aligned}$$

return $a_i, b_i, c_i, d_i, e_i, f_i$ and $\det(\mathbf{A})$

Figure 6.2: Proposed algorithm for fast matrix inversion and determinant calculation.

Table 6.1: Operation Counts in Terms of Real Arithmetic Operations with Real Inputs x and y

Method	$x \cdot y$	$x + y$	$1/x$	$\sqrt{x}$	Total
Cholesky	59	24	3	3	89
Proposed	37	26	1	0	64

gorithm 6.1 demands 44 multiplications, 12 additions, three inversions, three squarings, and three square roots of real quantities. This accounts for a total of 89 operations as shown in Table 6.1. Note that the proposed Algorithm 6.2 outperforms Algorithm 6.1 not only in the total count of operations. In fact, it reduces the arithmetic cost for all operations listed in Table 6.1, except for the number of additions which requires only two extra additions.

In contrast, the proposed Algorithm 6.2 requires four multiplications and two additions for the computation of a_i, d_i, and f_i, each. It also requires ten multiplications and six additions for the computation of b_i, c_i, and e_i, each. The computation of $\det(\mathbf{A})$ and its inverse demands five multiplications, four additions, and only one inversion. Because of that, Algorithm 6.2 requires a total of 37 multiplications, 26 additions, and one division of real quantities. A total of 64 operations; 28% less than the usual Cholesky factorization approach. Above quantities are summarized in Table 6.1.

6.4 Implementation and Results

6.4.1 Material and Equipment

We implemented the proposed Algorithm 6.2 and the Cholesky factorization based approach using open computing language (OpenCL) with the C application interface (API) [199]; being compiled for a general purpose graphical processing unit (GPGPU). The device used is a NVIDIA GeForce 940M, which contains three computing units and allows the data to be scheduled through the maximum of three different dimensions. Each dimension allows the maximum use of 1024, 1024, and 64 work items at the same time, respectively. The device has a global memory of 2 GB. The host side was implemented in C/C++ language. The machine used has a Core i7 microprocessor

and 8 GB of random access memory (RAM) and runs a Linux operating system with distribution Ubuntu version 16.04 LTS.

The communication between the proposed method and popular platforms for image processing, such as Matlab, ENVI, and NEST can be performed by means of interface capabilities offered by each different vendor. For Matlab, this can be accomplished by mex functions. ENVI or NEST can resort to direct calls of the compiled C/C++ code to interface with the proposed algorithm.

6.4.2 Data Setup

We separated simulated and measured data for numerical analysis.

Simulated Data

Simulated data were generated from random matrices obtained with the Eigen C/C++ library [200] through the method `Eigen::MatrixXcd::Random`. The real and imaginary parts of the matrix coefficients are drawn from the uniform distribution on the closed interval $[-1, 1]$. The obtained matrices are then converted into Hermitian matrices by computing the product of itself with the transpose conjugate [46].

Measured Data

For the measured data, we considered the data available in [194]. Such data were acquired by the Airborne Synthetic Aperture Radar (AIRSAR) sensor over San Francisco, CA, in fully polarimetric mode and L-band. The approximate spatial resolution is $10\,\mathrm{m} \times 10\,\mathrm{m}$. The data were stored in `.rdata` format, original from R language [201]. For recovering the data, we employ RInside C/C++ library [202], which allows the binding of R and C/C++ applications. The data were converted to the appropriate types in C/C++ and then sent to the GPGPU using OpenCL scheduling runtime routines. The data used for this experiment has a total of 450×600 pixels.

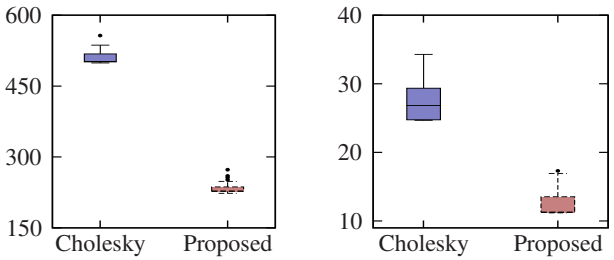


Figure 6.3: Boxplots for the computing time (in milliseconds) of the Cholesky based approach and the proposed method using simulated data (left) and measured data (right).

6.4.3 Methodology and Results

The OpenCL implementations of Algorithm 6.1 and Algorithm 6.2 were executed using all the compute units and their respective work items as a single dimension [199, pg. 41][203]. The GPGPU device was allowed to manage and use the preferred work-group size [199]. The computation was performed with 1000 replicates in order to reduce the operating system fluctuations when estimating the speedup.

For the simulated data computation, we considered 5 469 354 matrices for each replicate. This amount of matrices was carefully selected to use the GPGPU device at the limit imposed by its available memory for allocation, and corresponds to an image of approximately 2340×2340 pixels. Indeed, GPGPU finite registers (buffers) are dedicated to data interchange with the host device and the buffer size sets an upper bound to the maximum allocated memory. This is a limiting factor in the high performance applications. For measured data, we submitted the 270 000 matrices associated to the selected 450×600 pixel image.

Table 6.2 displays descriptive statistics of the computing time required by Algorithm 6.1 and Algorithm 6.2 for simulated and measured data. We also provide the ratio between the measures as figure of merit for the speed gain. Figure 6.3 displays boxplots of the runtimes for the Cholesky based approach and the proposed method both for measured and simulated data. The time statistics for simulated data are larger when compared to the results from measured data, because simulated and measured data sets have different number of matrices. However, the ratios of the computation time statistics for the simulated and measure data are very close.

Table 6.2: Run-time execution statistics: minimum (t_{min}), average (t_{avg}), maximum (t_{max}), and standard deviation (t_{std}) for matrix inversion and determinant evaluation. Measurements in milliseconds

Time Statistics	Simulated Data (5.5×10^6 matrices)			Measured Data (2.7×10^5 matrices)		
	Cholesky	**Proposed**	Gain	Cholesky	**Proposed**	Gain
t_{min}	498.89	223.06	2.23	24.68	11.18	2.21
t_{avg}	507.54	231.51	2.19	28.35	12.98	2.18
t_{max}	556.79	273.08	2.03	62.60	28.59	2.18
t_{std}	9.34	6.11	–	2.55	1.44	–

6.4.4 Discussion

Although Table 6.1 suggests an improvement of $89/64 \approx 1.4$, i.e., roughly a 28% reduction in complexity, the actual improvement gain or complexity savings is much larger. Indeed, to correctly evaluate the speed improvements, one must take into account that inversion and square-root computations demand more GPU clock cycles when compared with simple additions or multiplications. In particular, reducing the number of calls for the square root operations proves to be significant. This is because, in modern processors, the square root operation is not computed directly but through the calculation of the inverse square root [204], which requires calling a division operation at the end of the calculation in order to obtain the sought result [205, 206, 207]. Therefore, the total impact of the arithmetic reduction in the proposed algorithm is not just the ratio of the counting of arithmetic operations of the Cholesky based approach and the proposed fast algorithm. Rather, this ratio is a conservative lower bound. Results from Table 6.2 confirm this analysis, showing that the speedup gain is more than double.

6.5 Chapter Summary

In this chapter, we reviewed the Cholesky factorization method for the inversion and determinant calculation of 3×3 Hermitian matrices in the context of PolSAR image processing. As the main contribution, we proposed a fast algorithm for the computation of matrix inverse and determinant computation for such matrices. The proposed algorithm is based on the computation of the adjoint

matrix and careful examination of the quantities involved. The derived algorithm was compared to the usual Cholesky based factorization approach and shown to significantly reduce the total number of arithmetic operations. The proposed algorithm and the Cholesky based approach were both implemented in GPGPU using OpenCL and C/C++. The proposed approach proved to reduce the total computation time in about a half when compared to the Cholesky based approach with real data; in other words, the proposed method offers a 120.7% improvement in speed, at least. Future works may include the investigation of the stability of the proposed approach in numerical terms and the combination of several matrices into one inversion and recovering with permuted multiplications.

Chapter 7

Conclusions and Future Works

7.1 Thesis Summary

The popularity of consumer electronics and the technological and scientific advancements in the last few decades caused a burst in the demand for efficient data processing in the most varying applications. This demand requires hardware devices and software solutions that are capable of processing a huge amount of data efficiently in terms of (i) time; (ii) accuracy; and (iii) energy. This is particularly important in applications with highly intensive computational steps, such as image and video compression [45, 98, 208], spectral analysis [209, 210, 211, 212], and matrix computation [51, 167, 168, 162].

This thesis proposed efficient methods that address the challenges I, II, and III in several problems in the context of digital signal processing and matrix-based computations. Specifically, this thesis offers the following contributions:

- The proposal of a non-conventional numerical representation for the DFT computation that provides error-free arithmetic up to a final stage. The proposed numerical representation is based on algebraic integer theory. The heavy computation required for reconstructing the quantities encoded in the proposed representation back to usual fixed-point is deferred to the end of the algorithm. Because of that, the source of errors are controlled and the proposed architecture can provide better numerical accuracy than standard methods.

- The proposal of a fast algorithm for the DCT computation based on algebraic integers. The proposed algorithm is able to compute the components of the DCT without error up to the final stage. This drastically reduces the numerical error in-

curred in the subsequent computations and allows a selective choice of precision that can be easily controlled by modifying only the final part of the algorithm.

- The proposition of a fast algorithm for the DCT computation based on summation-by-parts formula. The algorithm is shown to reduce the overall arithmetic complexity of the DCT computation, reaching the theoretical lower bound for the number of multiplications. It is shown that the proposed algorithm outperforms state-of-the-art methods when the input signal has some specific characteristics often found in common applications in signal processing problems.

- The proposal of a method for the fast computation of the largest eigenvalue and its corresponding eigenvector of tridiagonal matrices. The method is based on the computation of the square of the input matrix. Along with the introduced method, a fast algorithm for the matrix squaring for tridiagonal matrices is also proposed. It is shown that not only the computation of the largest eigenvalue and its eigenvector is faster than the usual methods, but it also improves accuracy by reducing the number of arithmetic operations that can strongly contribute to numerical errors.

- The proposal of a method for the efficient matrix inversion and determinant computation for specific sized matrices in the context of PolSAR image classification. The proposed method outperforms the classical method adopted for matrix inversion and determinant computation in that context. Along with the introduced method, a data parallel implementation using GPGPU is provided. The efficiency of the method is not only analyzed theoretically in terms of arithmetic complexity, but it is also verified in actual implementation with exhaustive simulations to testify its effectiveness.

In total, this thesis contributed to the state-of-the-art algorithms for efficient methods in digital signal processing and matrix-based computations. The proposed efficient methods were shown to

improve hardware or software implementation metrics such as computing time, energy consumption and accuracy.

7.2 Future Works

Considering the contributions in this thesis, there are several research directions that can be pursued as follows.

7.2.1 Larger Size DFT Computation Using Gauss-Eisenstein Based Representation

It was addressed in Chapter 2 the computation of DFT using algebraic integer theory. The DFT sizes addressed there were focused on the building block sizes 3-, 6-, and 12-point DFT. This is because the proposed numerical representation based on Gaussian and Eisenstein algebraic integers can exactly represent the unity-of-roots required for the computation of the DFT of sizes 3-, 6-, and 12-point sequences.

The application of the same numerical representation for other DFT sizes does not generate error-free architectures as the one proposed in Chapter 2. In fact, the unity-of-roots for DFT for other sequence sizes, such as 24-point sequences, are required to be approximated by the closest Gauss-Eisenstein integer. Although an architecture using the numerical representation introduced in Chapter 2 for larger DFT sizes is not error-free anymore (up to the final reconstruction step), the errors are not amplified and can be *easily* controlled. A possible direction in this research line would be the investigation of the use of the same algebraic integer representation proposed in Chapter 2 for the computation of larger DFT sequences and its hardware implementation.

7.2.2 Multidimensional DCT Computation Using Algebraic Integers

Chapter 3 addressed how to compute the one dimensional (1D) DCT using algebraic integers. The DCT is also often computed in scenarios where the input signals are multidimensional, e.g., image (2D) and video (3D). A possible direction of this work is to investigate the use of the numerical

representation proposed in Chapter 3 in the computation of multidimensional DCT.

7.2.3 Discrete Transforms Computation With Summation-by-Parts Formula

Chapter 4 proposed the use of summation-by-parts formula to compute the 1D DCT. Although the 1D DCT is a relevant discrete transform and has been used in many scenarios, the same approach could be used for computing a multidimensional DCT, in particular, the 2D DCT in scenarios of feature detection, where the mean value is not relevant, and the summation-by-parts formula can be applied. A different direction could be the application of summation-by-parts formula for the computation of different transforms such as DFT and DST.

Another direction, yet, is the merging of the summation-by-parts formula for the 1D DCT with motion estimation in video codecs and computing the 3D DCT. This is an interesting approach because the motion estimation in video codecs can be computed using the difference of two subsequent image blocks. If the 3D DCT for the video compression is implemented using the DCT kernel splitting, the 1D DCT component across the time can be computed using a fast algorithm where the input signal is natively the difference of the sequence we are interested in computing the DCT. In this case, the input would be the difference of the pixels in neighbouring blocks. This may be a potential direction because an algorithm using summation-by-parts formula may outperform state-of-the-art methods when the input sequence is the output of a forward difference operator.

7.2.4 Fast Eigenvalue Estimation of Band Matrices

Chapter 5 proposed a method for the fast and accurate estimation of tridiagonal matrices largest eigenvalue and its associated eigenvector. The method is based on the matrix squaring of a tridiagonal matrix and its proposed fast algorithm. The same insight can be used for band diagonal matrices. This is because using the square of a matrix in the Power method reduces the total number of iterations in almost a half, as demonstrated in Chapter 5. The computation of the squaring of a general matrix, however, may defeat the purpose of speeding up the overall largest eigenvalue estimation. But if the matrix is band diagonal, the complexity of squaring a matrix is drastically

reduced. Additionally, the use of fast algorithms for the computation of matrix square, tailored specifically for band diagonal matrices, can further improve the method. Following the same sketch of the proof of Theorem 1 and the demonstration on Section 5.2.2 in Chapter 5, one can show that the accuracy is also improved when using such approach for computing the largest eigenvalue and its eigenvector considering band diagonal matrices.

7.2.5 Fast Matrix Inversion and Determinant Computation in Radar Image Classification Problems

Chapter 6 introduced a fast algorithm for matrix inversion and determinant computation in image classification schemes based on data modelling with the complex Wishart distribution. The particular case attacked in this contribution is constrained to matrices of order 3×3. A more general scheme and sometimes adopted in radar image classification using the complex Wishart distribution requires the inversion and determinant computation of matrices of sizes $3k \times 3k$, where $k = 2, 3, \ldots$. A possible future direction is attacking the problems where matrix inversion and determinant computation are required for larger matrices.

Another possible direction is the investigation of hardware oriented architectures for the method proposed in Chapter 6. Considering the arithmetic complexity reduction in the proposed method in relation to the usual approach based on Cholesky factorization, a hardware implementation, either on FPGA or final ASIC design, would furnish a higher advantage for the proposed method than the speed up offered by software implementation.

Bibliography

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

[2] W. J. Tompkins, Ed., *Biomedical Digital Signal Processing: C-language Examples and Laboratory Experiments for the IBM PC*. Upper Saddle River, NJ, USA: Prentice-Hall, Inc., 1993.

[3] T. L. Rusch, R. Sankar, and J. E. Scharf, "Signal processing methods for pulse oximetry," *Computers in Biology and Medicine*, vol. 26, no. 2, pp. 143–159, Mar. 1996.

[4] G. Wang, Q. Ding, and Z. Hou, "Independent component analysis and its applications in signal processing for analytical chemistry," *Trends in Analytical Chemistry (TrAC)*, vol. 27, no. 4, pp. 368–376, Apr. 2008.

[5] D. Gust, J. Andreasson, U. Pischel, T. A. Moore, and A. L. Moore, "Data and signal processing using photochromic molecules," *Chemical Communications*, vol. 48, pp. 1947–1957, Jan. 2012.

[6] D. H. Evans and W. N. McDicken, *Doppler Ultrasound: Physics, Instrumentation and Signal Processing*, 2nd ed. Wiley, Feb. 2000.

[7] W. E. Cleland and E. G. Stern, "Signal processing considerations for liquid ionization calorimeters in a high rate environment," *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment*, vol. 338, no. 2, pp. 467–497, Jan. 1994.

[8] S. Saponara, M. Rovini, L. Fanucci, A. Karachalios, G. Lentaris, and D. Reisis, "Design and comparison of FFT VLSI architectures for SoC telecom applications with different fle-

xibility, speed and complexity trade-offs," *Circuits, Systems, and Signal Processing*, vol. 31, no. 2, pp. 627–649, 2012.

[9] F. Camarda, J.-C. Prevotet, and F. Nouvel, "Implementation of a reconfigurable fast Fourier transform application to digital terrestrial television broadcasting," in *International Conference on Field Programmable Logic and Applications (FPL)*, Prague, Czech Republic, 2009, pp. 353–358.

[10] Z.-X. Yang, Y.-P. Hu, C.-Y. Pan, and L. Yang, "Design of a 3780-point IFFT processor for TDS-OFDM," *IEEE Transactions on Broadcasting*, vol. 48, no. 1, pp. 57–61, 2002.

[11] A. Grinsted, J. C. Moore, and S. Jevrejeva, "Application of the cross wavelet transform and wavelet coherence to geophysical time series," *Nonlinear Processes in Geophysics*, 2004.

[12] E. Foufoula-Georgiou and P. Kumar, Eds., *Wavelets in Geophysics*. Academic Press, 1994.

[13] J. Clearbout, *Fundamentals of Geophysical Data Processing: with Applications to Petroleum Prospecting*. Department of Geophysics, Stanford University: Mc-Graw Hill, 1976.

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

[15] J. G. Proakis and D. K. Manolakis, *Digital Signal Processing*. Upper Saddle River, NJ, USA: Pearson Prentice Hall, 2007.

[16] R. E. Blahut, *Algebraic Codes for Data Transmission*. Cambridge University Press, 2003.

[17] R. M. Rangayyan, *Biomedical Image Analysis*. CRC Press, 2005.

[18] D. F. G. Coelho, R. M. Rangayyan, and V. S. Dimitrov, "Detection of neovacularization near the optic disk due to diabetic retinopathy," in *European Signal Processing Conference (EUSIPCO)*. IEEE, Aug. 2016, pp. 2040–2044.

[19] K. Ogata, *Discrete-time Control Systems*. Upper Saddle River, NJ, USA: Prentice-Hall, Inc., 1987.

[20] N. Keshava and J. F. Mustard, "Spectral unmixing," *IEEE Signal Processing Magazine*, pp. 44–57, Jan. 2002.

[21] Z. Chen, *Reservoir Simulation - Mathematical Techniques in Oil Recovery*. Society for Industrial and Applied Mathematics (SIAM), 2007.

[22] T. Ertekin, J. H. Abou-Kassem, and G. R. King, *Basic Applied Reservoir Simulation*. Society of Petroleum Engineers, 2000.

[23] J. Stein, *Digital Signal Processing: A Computer Science Approach*, ser. Telecomunications and Signal Processing. Wiley, Sep. 2000.

[24] J. E. Savage, *Models of Computation: Exploring the Power of Computing*, 1st ed. Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc., 1997.

[25] S. Winograd, *Arithmetic Complexity of Computations*. SIAM, 1980.

[26] R. L. Graham, D. E. Knuth, and O. Patashnik, *Concrete Mathematics*. Reading, MA: Addison-Wesley Publishing Company, 1989.

[27] M. Bl aser and B. Chokaev, "Algebras of minimal multiplicative complexity," in *IEEE 27th Conference on Computational Complexity*, Jun. 2012, pp. 224–234.

[28] S. V. Fedorenko, "The discrete fourier transform over a finite field with reduced multiplicative complexity," in *IEEE International Symposium on Information Theory Proceedings*, Jul. 2011, pp. 1200–1204.

[29] N. Hockert and K. Compton, "Improving floating-point performance in less area: Fractured floating point units (FFPUs)," *Journal of Signal Processing Systems*, vol. 67, no. 1, pp. 31–46, Apr. 2012.

[30] P. M. Seidel, "How to half the latency of IEEE compliant floating-point multiplication," in *24th Euromicro Conference*, vol. 1, Aug. 1998, pp. 329–332 vol.1.

[31] J. D. Bruguera and T. Lang, "Floating-point fused multiply-add: reduced latency for floating-point addition," in *17th IEEE Symposium on Computer Arithmetic (ARITH)*, Jun. 2005, pp. 42–51.

[32] D. S. Watkins, *Fundamentals of Matrix Computations*, ser. Pure and Applied Mathematics: A Wiley Series of Texts, Monographs and Tracts. Wiley, 2004.

[33] A. Fog, "Lists of instruction latencies, throughputs and micro-operation breakdowns for Intel, AMD and VIA CPUs," http://www.agner.org/optimize/instruction_tables.pdf, Apr. 2017.

[34] Intel Corporation, "Intel® 64 and IA-32 architectures optimization reference manual," Jun. 2016.

[35] A. A. Wahba and H. A. H. Fahmy, "Area efficient and fast combined binary/decimal floating point fused multiply add unit," *IEEE Transactions on Computers*, vol. 66, no. 2, pp. 226–239, Feb. 2017.

[36] N. Brunie, "Modified fused multiply and add for exact low precision product accumulation," in *IEEE 24th Symposium on Computer Arithmetic (ARITH)*, Jul. 2017, pp. 106–113.

[37] C. H. Roth, Jr. and L. K. John, *Digital Systems Design Using VHDL*. Cengage, 2017.

[38] K. Fatahalian, J. Sugerman, and P. Hanrahan, "Understanding the efficiency of GPU algorithms for matrix-matrix multiplication," in *Proceedings of the ACM Special Interest Group on Computer Graphics and Interactive Techniques (SIGGRAPH) and European Association for Computer Graphics (EUROGRAPHICS) Conference on Graphics Hardware*. New York, NY, USA: ACM, 2004, pp. 133–137.

[39] L. Torres, S. J. S. Sant'Anna, C. da Costa Freitas, and A. C. Frery, "Speckle reduction in polarimetric SAR imagery with stochastic distances and nonlocal means," *Pattern Recognition*, vol. 47, no. 1, pp. 141–157, Jan. 2014.

[40] G. Casella and R. L. Berger, *Statistical Inference*, 2nd ed. Duxbury Advanced Series, 2002.

[41] J. S. Lee, D. L. Schuler, R. H. Lang, and K. J. Ranson, "K-distribution for multi-look processed polarimetric SAR imagery," in *Proc. IEEE IGARSS*, vol. 4, Pasadena, CA, Aug. 1994, pp. 2179–2181.

[42] K. Conradsen, A. A. Nielsen, J. Schou, and H. Skriver, "A test statistic in the complex Wishart distribution and its application to change detection in polarimetric SAR data," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 41, no. 1, pp. 4–19, Jan. 2003.

[43] J. Fan and J. Wang, "A two-phase fuzzy clustering algorithm based on neurodynamic optimization with its application for polSAR image segmentation," *IEEE Transactions on Fuzzy Systems*, vol. 26, no. 1, pp. 72–83, Feb. 2018.

[44] D. Li, Y. Zhang, and F. Zhu, "PolSAR image fast classification based on random similarity," in *Progress In Electromagnetics Research Symposium - Spring (PIERS)*, May 2017, pp. 883–888.

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

[46] G. H. Golub and C. F. V. Loan, *Matrix Computations*, 3rd ed. Baltimore, MD, USA: Johns Hopkins University Press, 1996.

[47] J. H. Wilkinson, *Rounding Errors in Algebraic Processes*. Dover Publications Inc., 1963.

[48] J. Liu, X. Lu, and F. Meng, "An adaptive regularization method for ill-conditioned problem," in *Fifth International Joint Conference on Computational Sciences and Optimization*, Jun. 2012, pp. 45–47.

[49] A. Ert urk, M. D. Iordache, and A. Plaza, "Hyperspectral change detection by sparse unmixing with dictionary pruning," in *7th Workshop on Hyperspectral Image and Signal Processing: Evolution in Remote Sensing (WHISPERS)*, Jun. 2015, pp. 1–4.

[50] R. L. Parker, *Geophysical Inverse Theory*. Princeton University Press, 1994.

[51] Å. Björck, *Numerical Methods in Matrix Computations*, ser. Texts in Applied Mathematics. Springer International Publishing, 2014.

[52] M. T. Heideman, *Multiplicative Complexity, Convolution, and the DFT*, C. S. Burrus, Ed. New York: Springer-Verlag, 1988.

[53] P. K. Meher, "A new convolutional formulation of the DFT and efficient systolic implementation," in *IEEE Region 10 Conference (TENCON)*, 2005, pp. 1–5.

[54] ——, "Memory-based computation of inner-product for digital signal processing applications," in *International Symposium on Electronic System Design (ISED)*. Bhubaneswar, India: IEEE, 2010, pp. 95–100.

[55] V. Dimitrov, G. A. Jullien, and W. C. Miller, "Eisenstein residue number system with applications to DSP," in *Proceedings of the 40th Midwest Symposium on Circuits and Systems*, vol. 2, Sacramento, CA, Aug. 1997, pp. 675–678.

[56] 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*, vol. 33, no. 4, p. 9, Jul. 1987.

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

[58] E. Dubois and A. Venetsanopoulos, "The generalized discrete Fourier transform in rings of algebraic integers," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. 28, no. 2, pp. 169–175, Apr. 1980.

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

[60] V. S. Dimitrov, G. A. Jullien, and W. C. Miller, "A residue number system implementation of real orthogonal transforms," *IEEE Transactions on Signal Processing*, vol. 46, pp. 563–570, Mar. 1998.

[61] S. K. Madishetty, A. Madanayake, R. J. Cintra, and V. S. Dimitrov, "Precise VLSI architecture for AI based 1-D/ 2-D Daub-6 Wavelet filter banks with low adder-count," *IEEE Transactions on Circuits and Systems I*, vol. 61, no. 7, pp. 1984–1993, Feb. 2014.

[62] Y. Suzuki, T. Sone, and K. Kido, "A new FFT algorithm of radix 3, 6 and 12," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. 34, no. 2, pp. 380–383, Apr. 1986.

[63] S. A. White, "A radix-12 FFT building block," *Conference Record of Thirty-Fourth Asilomar Conference on Signal, Systems and Computers*, vol. 1, pp. 320–324, 1993.

[64] F. Qureshi, M. Garrido, and O. Gustafsson, "Unified architecture for 2, 3, 4, 5, and 7-point DFTs based on Winograd Fourier transform algorithm," *Electronics Letters*, vol. 49, no. 5, pp. 348–349, 2013.

[65] H. Yang, K. Chen, S. Xie, M. Jing, Y. Zhiyi, and X. Zeng, "Efficient implementation of 3780-point FFT on a 16-core processor," in *IEEE 10th International Conference on ASIC (ASICON)*, 2013, pp. 1–4.

[66] X. Yang, W. Du, J. Yu, R. Lv, and Z. Yang, "A novel 3780-point FFT," in *IEEE International Conference on Wireless Communications, Networking and Information Security (WCNIS)*, Beijing, China, 2010, pp. 178–182.

[67] J. Lofgren and P. Nilsson, "On hardware implementation of radix 3 and radix 5 FFT kernels for LTE systems," in *The Nordic Microelectronics Conference (NORCHIP)*, Lund, Sweden, Nov. 2011, pp. 1–4.

[68] H. X. Huang, L. R. Liu, Y. Z. Yin, and L. Y. Zhao, "Fast parallel complex discrete Fourier transforms using a multichannel optical correlator," *Optics Communications*, vol. 68, no. 6, pp. 408–411, Nov. 1988.

[69] S. A. White, "Radix-12 DFT/FFT building block," Aug. 1993, US Patent 5,233,551.

[70] H. F. Silverman, "An introduction to programming the Winograd Fourier transform algorithm (WFTA)," *IEEE Transactions on Acoustics, Speech and Signal Processing*, vol. 25, no. 2, pp. 152–165, 1977.

[71] E. Chen, Y. Chen, Y. Wang, C. Chen, and X. Zeng, "A multi-core mapping implementation of 3780-point FFT," in *International SoC Design Conference (ISOCC)*, Jeju Island, South Korea, Nov. 2012, pp. 289–292.

[72] C.-I. H. Chen, K. George, W. McCormick, J. B. Y. Tsui, S. L. Hary, and K. M. Graves, "Design and performance evaluation of a 2.5-GSPS digital receiver," *IEEE Transactions on Instrumentation and Measurement*, vol. 54, no. 3, pp. 1089–1099, 2005.

[73] I. D. Lu and P. Lee, "Use of mixed radix FFT in electric power system studies," *IEEE Transactions on Power Delivery*, vol. 9, no. 3, pp. 1276–1280, 1994.

[74] S. Prakash and V. V. Rao, "A new radix-6 FFT algorithm," *IEEE Transactions on Acoustics, Speech and Signal Processing*, vol. 29, no. 4, pp. 939–941, 1981.

[75] D. F. G. Coelho, R. J. Cintra, L. C. Souza, A. Madanayake, and V. S. Dimitrov, "Representação numérica em inteiros de Gauss-Eisenstein," in *Anais do XXX Simpósio Brasileiro de Telecomunicações*. Brasília, DF, Brasil: Sociedade Brasileira de Telecomunicações, Sep. 2012.

[76] G. H. Hardy, E. M. Wright, and A. Wiles, *An introduction to the theory of numbers*, 6th ed., R. Heath-Brown and J. Silverman, Eds. New York, NY: Oxford University Press, Sep. 2008.

[77] J. Fraleigh, *A First Course in Abstract Algebra*, 7th ed., V. J. Katz, Ed. University of Rhode Island: Pearson, 2003.

[78] S. Winograd, "Some bilinear forms whose multiplicative complexity depends on the field of constants," *Mathematical Systems Theory*, vol. 10, no. 1, pp. 169–180, 1976.

[79] A. G. Dempster and M. D. Macleod, "Constant integer multiplication using minimum adders," *IEEE Transactions on Circuits and Systems II: Analog and Digital Signal Processing*, vol. 141, no. 5, pp. 407–413, 1994.

[80] G. Plonka, "A global method for invertible integer DCT and integer wavelet algorithms," *Applied and Computational Harmonic Analysis*, vol. 16, no. 2, pp. 90–110, Mar. 2003.

[81] S. K. Madishetty, A. Madanayake, R. J. Cintra, V. S. Dimitrov, and D. H. Mugler, "VLSI architectures for the 4-Tap and 6-Tap 2-D Daubechies wavelet filters using algebraic integers," *IEEE Transactions on Circuits and Systems I*, vol. 60, no. 6, pp. 1455–1468, 2013.

[82] O. Gustafsson and F. Qureshi, "Addition aware quantization for low complexity and high precision constant multiplication," *IEEE Signal Processing Letters*, vol. 17, no. 2, pp. 173–176, 2010.

[83] H. Malvar, "Fast computation of discrete cosine transform through fast Hartley transform," *Electronics Letters*, vol. 22, no. 7, pp. 352–353, 1986.

[84] T. M. Apostol, *Calculus*, 2nd ed., G. Springer, Ed. Pasadena, CA: John Wiley & Sons, Inc., Sep. 1966, vol. 1.

[85] ——, *Calculus*, 2nd ed., G. Springer, Ed. Pasadena, CA: John Wiley & Sons, Inc., Sep. 1968, vol. 2.

[86] J. T. Manassah, *Elementary Mathematical and Computational Tools for Electrical and Computer Engineers Using Matlab* ®. New York, NY: CRC Press, 2001.

[87] P.-Y. Tsai, C.-W. Chen, and M.-Y. Huang, "Automatic IP generation of FFT/IFFT processors with word-length optimization for MIMO-OFDM systems," *EURASIP Journal on Advances in Signal Processing*, vol. 2011, pp. 1–15, Jan. 2011.

[88] S. J. Saenz, J. J. Raygoza P., E. C. Becerra A., S. O. Cisneros, and J. R. Dominguez, "FPGA design and implementation of radix-2 fast Fourier transform algorithm with 16 and 32 points," in *IEEE International Autumn Meeting on Power, Electronics and Computing (ROPEC)*, Nov. 2015, pp. 1–6.

[89] I. Mamatha, J. N. Raj, S. Tripathi, and T. S. B. Sudarshan, "Systolic architecture implementation of 1D DFT and 1D DCT," in *IEEE International Conference on Signal Processing, Informatics, Communication and Energy Systems (SPICES)*, Feb. 2015, pp. 1–5.

[90] J.-C. Kuo, C.-H. Wen, C.-H. Lin, and A.-Y. Wu, "VLSI design of a variable-length FFT/IFFT processor for OFDM-based communication systems," *EURASIP Journal on Advances in Signal Processing*, vol. 2003, pp. 1306–1316, Jan. 2003.

[91] K. Maharatna, E. Grass, and U. Jagdhold, "A 64-point Fourier transform chip for high-speed wireless LAN application using OFDM," *IEEE Journal of Solid-State Circuits*, vol. 39, no. 3, pp. 484–493, Mar. 2004.

[92] T. H. Tran, S. Kanagawa, D. P. Nguyen, and Y. Nakashima, "ASIC design of MUL-RED Radix-2 pipeline FFT circuit for 802.11ah system," in *IEEE Symposium in Low-Power and*

High-Speed Chips (COOL CHIPS XIX), Yokohama, Japan, Apr. 2016, pp. 1–3.

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

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

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

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

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

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

[99] G. K. Wallace, "The JPEG still picture compression standard," *IEEE Transactions on Consumer Electronics*, vol. 38, no. 1, pp. 18–34, Feb. 1992.

[100] N. Roma and L. Sousa, "Efficient hybrid DCT-domain algorithm for video spatial downscaling," *EURASIP Journal on Advances in Signal Processing*, vol. 2007, no. 1, p. 057291, Sep. 2007.

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

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

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

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

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

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

[107] ——, "On the error-free computation of fast cosine transform," *Information Theories and Applications*, vol. 12, no. 4, pp. 321–327, 2005.

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

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

[110] Y. Shen and H. Oh, "Pipelined implementation of AI-based Loeffler DCT," *IEICI Electronics Express*, vol. 10, no. 12, pp. 1–7, May 2013.

[111] H. Pollard and H. G. Dimamond, *The Theory of Algebraic Numbers*, 2nd ed., ser. The Carus Mathematical Monographs. The Mathematical Association of America, 1975.

[112] M. Abramowitz and I. A. Stegun, *Handbook of Mathematical Functions with Formulas, Graphs, and Mathematical Tables*, ninth dover printing, tenth gpo printing ed. New York, NY: Dover, 1964.

[113] A. V. Oppenheim, R. W. Schafer, and J. R. Buck, *Discrete-time signal processing*, 2nd ed., A. V. Oppenheim, Ed. Upper Saddle River, NJ: Prentice-Hall, Inc., 1999, vol. 1.

[114] V. Dimitrov, G. A. Jullien, and W. C. Miller, "A new DCT algorithm based on encoding algebraic integers," in *Proceedings of the 1998 IEEE International Conference on Acoustics, Speech and Signal Processing*, vol. 3. Seattle, WA: IEEE, May 1998, pp. 1377–1380.

[115] M. Fu, V. Dimitrov, and G. A. Jullien, "An efficient technique for error-free algebraic-integer encoding for high performance implementation of the DCT and IDCT," in *Proceedings of IEEE International Symposium on Circuits and Systems*, vol. 4, Sydney, Australia, May 2001, pp. 906–909.

[116] R. D. Tocci, N. Widmer, and G. Moss, *Digital Systems: Principles and Applications*, 11st ed. Prentice-Hall, Inc., Jul. 2010.

[117] ROACH2-CASPER. Available at https://casper.berkeley.edu/wiki/ROACH-2_Revision_2. Last visit on Jun 30, 2015.

[118] J.-B. Hiriart-Urruty and C. Lemaréchal, *Convex Analysis and Minimization Algorithms II, Advanced Theory and Bundle Methods*, ser. Grundlehren der mathematischen Wissenschaften. Berlin, New-york: Springer-Verlag, 1996.

[119] M. W. Jeter, *Mathematical Programming : An Introduction to Optimization*, ser. Monographs and textbooks in pure and applied mathematics. New York: M. Dekker, 1986, index.

[120] G. B. Folland, *Fourier Analysis and Its Applications*. Seattle, WA: American Mathematical Society, 2000.

[121] N. Ahmed and K. R. Rao, *Orthogonal Transforms for Digital Signal Processing*. New York, NY: Springer-Verlag, 1975.

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

[123] A. A. Chien and V. Karamcheti, "Moore's law: The first ending and a new beginning," *Computer*, vol. 46, no. 12, pp. 48–53, 2013.

[124] M. R. Wigan and R. Clarke, "Big data's big unintended consequences," *Computer*, vol. 46, no. 6, pp. 46–53, 2013.

[125] L. Ji and L. X. Ming, "New DCT computation algorithm for VLIW architecture," in *6th International Conference on Signal Processing*, vol. 1, 2002, pp. 41–44.

[126] K. R. Rao, D. N. Kim, and J. J. Hwang, *Video coding standards: AVS China, H.264/MPEG-4 PART 10, HEVC, VP6, DIRAC and VC-1*. Arlington, TX, USA: Springer, 2014.

[127] H. W.-H. Chen, C. Smith, and S. Fralick, "A fast computational algorithm for the discrete cosine transform," *IEEE Transactions on Communications*, vol. 25, no. 9, pp. 1004–1009, Jan. 2003.

[128] B. Lee, "A new algorithm to compute the discrete cosine transform," *IEEE Transactions on Acoustics, Speech and Signal Processing*, vol. 32, no. 6, pp. 1243–1245, Dec. 1984.

[129] E. Feig and S. Winograd, "Fast algorithms for the discrete cosine transform," *IEEE Transactions on Signal Processing*, vol. 40, no. 9, pp. 2174–2193, Sep. 1992.

[130] Y. Arai, T. Agui, and M. Nakajima, "A fast DCT-SQ scheme for images," *IEICE Transactions*, vol. E71, no. 11, pp. 1095–1097, Nov. 1988.

[131] R. W. Hamming, *Digital Filters*, 3rd ed. Hertfordshire, UK: Prentice Hall International Ltd., 1989.

[132] X. Limin, X. Weisheng, and Y. Youling, "A fast harmonic detection method based on recursive DFT," in *8th International Conference on Electronic Measurement and Instruments (ICEMI)*. Xi'an, China: IEEE, Aug. 2007, pp. 3–972.

[133] E. Zheng, Z. Liu, and L. Ma, "Study on harmonic detection method based on FFT and wavelet transform," in *2nd International Conference on Signal Processing Systems (ICSPS)*, vol. 3, 2010.

[134] T. M. Apostol, *Mathematical Analysis*. Reading, MA: Addison-Wesley Publishing Company, 1981.

[135] K. Mattssona and J. NordstrÃűma, "Summation by parts operators for finite difference approximations of second derivatives," *Journal of Computational Physics*, vol. 199, no. 20, pp. 503–540, Sep. 2004.

[136] K. Mattsson, "Boundary procedures for summation-by-parts operators," *Journal of Scientific Computing*, vol. 18, no. 1, pp. 133–153, Feb. 2003.

[137] J. R. Mosig and A. A. Melcon, "The summation-by-parts algorithm - A new efficient technique for the rapid calculation of certain series arising in shielded planar structures," *IEEE Transactions on Microwave Theory and Techniques*, vol. 50, no. 1, pp. 215–218, Jan. 2002.

[138] G. Boudreaux-Bartels and T. W. Parks, "Discrete Fourier transform using summation by parts," in *IEEE International Conference on Acoustics, Speech, and Signal Processing*, vol. 12, Apr. 1987, pp. 1827–1830.

[139] G. Boudreaux-Bartels, D. W. Tufts, P. Dhir, G. Sadasiv, and G. Fischer, "Analysis of errors in the computation of Fourier coefficients using the arithmetic Fourier transform (AFT) and summation by parts (SBP)," in *IEEE International Conference on Acoustics, Speech, and Signal Processing*, vol. 2, May 1989, pp. 1011–1014.

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

[141] A. K. Jain, *Fundamentals of Digital Image Processing*, T. Kailath, Ed. University of California, Davis: Prentice Hall Information and System Sciences Series, 1989.

[142] Y. Wang, C.-S. Chua, and Y.-K. Ho, "Facial feature detection and face recognition from 2D and 3D images," *Pattern Recognition Letters*, vol. 23, no. 10, pp. 1191–1202, 2002.

[143] Elboher and M. Werman, "Efficient and accurate Gaussian image filtering using running sums," in *12th International Conference on Intelligent Systems Design and Applications (ISDA)*, 2012, pp. 897–902.

[144] P. Viola and M. Jones, "Rapid object detection using a boosted cascade of simple features," in *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, vol. 1, 2001.

[145] H. El-Banna, A. A. El-Fattah, and W. Fakhr, "An efficient implementation of the 1D DCT using FPGA technology," in *Engineering of Computer-Based Systems, 2004. Proceedings. 11th IEEE International Conference and Workshop on the*, 2004, pp. 356–360.

[146] S. J. Shan, C. Chen, and E. Yang, "High performance 2-D IDCT for image/video decoding based on FPGA," in *2012 International Conference on Audio, Language and Image*

Processing (ICALIP), 2012, pp. 33–38.

[147] H. K. C.-K. Fong and W.-K. Cham, "LLM integer cosine transform and its fast algorithm," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 22, no. 6, pp. 844–854, 2012.

[148] Mukherjee and S. K. Mitra, "Enhancement of color images by scaling the DCT coefficients," *IEEE Transactions on Image Processing*, vol. 17, no. 10, pp. 1783–1794, 2008.

[149] S. Bouguezel, M. O. Ahmad, and M. N. S. Swamy, "A multiplication-free transform for image compression," in *2nd International Conference on Signals, Circuits and Systems*, Monastir, TN, 2008, pp. 1–4.

[150] G. A. Papakostas, D. E. Koulouriotis, and E. G. Karakasis, *Image Processing*. INTECH, Sep. 2009, ch. 2, Efficient 2-D DCT Computation from an Image Representation Point of View.

[151] G. A. Papakostas, E. G. Karakasis, and D. E. Koulouriotis, "On accelerating the computation of 2-D discrete cosine transform in image processing," in *International Conference on Signals and Electronic Systems*, Sep. 2008, pp. 7–10.

[152] ——, "Efficient and accurate computation of geometric moments on gray-scale images," *Pattern Recognition*, vol. 41, no. 6, pp. 1895–1904, 2008.

[153] ——, "Exact and speedy computation of legendre moments on binary images," in *Proceedings of the Eight International Workshop on Image Analysis for Multimedia Interactive Services (WIAMIS)*. Washington, DC, USA: IEEE Computer Society, 2007, pp. 48–48.

[154] M. Hassaballah, A. A. Abdelmgeid, and H. A. Alshazly, *Image Features Detection, Description and Matching*. Cham: Springer International Publishing, 2016, pp. 11–45.

[155] M. Sayyouri, A. Hmimid, and H. Qjidaa, "A fast computation of hahn moments for binary and gray-scale images," in *IEEE International Conference on Complex Systems (ICCS)*, Nov. 2012, pp. 1–6.

[156] A. Hmimid, M. Sayyouri, and H. Qjidaa, "A fast method for reconstruction of binary and gray-scale images by the tchebichef moments," in *Colloquium in Information Science and Technology*, Oct. 2012, pp. 106–111.

[157] S. Bouguezel, M. O. Ahmad, and M. N. S. Swamy, "Binary discrete cosine and hartley transforms," *IEEE Transactions on Circuits and Systems I*, vol. 60, no. 4, pp. 989–1002, Apr. 2013.

[158] ——, "A low-complexity parametric transform for image compression," in *IEEE International Symposium on Circuits and Systems*, Rio de Janeiro, BR, May 2011, pp. 2145–2148.

[159] C. J. Tablada, F. M. Bayer, and R. J. Cintra, "A class of DCT approximations based on the Feig-Winograd algorithm," *Signal Processing*, 2015.

[160] J. Crank and E. Nicolson, "A practical method for numerical evaluation of solutions of partial differential equations of the heat-conduction type," *Mathematical Proceedings of the Cambridge Philosophical Society*, vol. 43, pp. 50–67, 1947.

[161] G. Meurant, "A review on the inverse of symmetric tridiagonal and block tridiagonal matrices," *Journal on Matrix Analysis and Applications*, vol. 13, no. 3, pp. 707–728, Jul. 1992.

[162] A. D. A. Hadj and M. Elouafi, "A fast numerical algorithm for the inverse of a tridiagonal and pentadiagonal matrix," *Applied Mathematics and Computation*, vol. 202, no. 2, pp. 441–445, Aug. 2008.

[163] M. El-Mikkawy and A. Karawia, "Inversion of general tridiagonal matrices," *Applied Mathematics Letters*, vol. 19, no. 8, pp. 712–720, Aug. 2006.

[164] Y. Huang and W. F. McColl, "Analytical inversion of general tridiagonal matrices," *Journal of Physics A: Mathematical and General*, vol. 30, no. 22, p. 7919, Nov. 1997.

[165] J. Demmel and K. Veselić, "Jacobi's method is more accurate than QR," *SIAM Journal on Matrix Analysis and Applications*, vol. 13, no. 4, pp. 1204–1245, 1992.

[166] P. Arbenz, "Lecture notes on solving large scale eigenvalue problems," http://people.inf.ethz.ch/arbenz/ewp/Lnotes/lsevp.pdf, Apr. 2016.

[167] D. Coppersmith and S. Winograd, "Matrix multiplication via arithmetic progressions," *Journal of Symbolic Computation*, vol. 9, no. 3, pp. 251–280, Mar. 1990, special issue on computational algebraic complexity.

[168] A. M. Davie and A. J. Stothers, "Improved bound for complexity of matrix multiplication," *Proceedings of the Royal Society of Edinburgh Section A: Mathematics*, vol. 143, no. 2, pp. 351–369, 2013.

[169] F. L. Gall, "Powers of tensors and fast matrix multiplication," in *Proceedings of the 39th International Symposium on Symbolic and Algebraic Computation (ISSAC)*. New York, NY, USA: ACM, 2014, pp. 296–303.

[170] S. Robinson, "Toward an optimal algorithm for matrix multiplication," http://www.siam.org/pdf/news/174.pdf, Nov. 2005, SIAM News.

[171] J. J. M. Cuppen, "A divide and conquer method for the symmetric tridiagonal eigenproblem," *Numerische Mathematik*, vol. 36, no. 2, pp. 177–195, Jun. 1980.

[172] A. S. Krishnakumar and M. Morf, "Eigenvalues of a symmetric tridiagonal matrix: A divide-and-conquer approach," *Numerische Mathematik*, vol. 48, no. 3, pp. 349–368, May 1986.

[173] J. J. Dongarra and D. C. Sorensen, "A fully parallel algorithm for the symmetric eigenvalue problem," *SIAM Journal on Scientific and Statistical Computing*, vol. 8, no. 2, pp. 139–154, 1987.

[174] M. Gu and S. C. Eisenstat, "A stable and efficient algorithm for the rank-one modification of the symmetric eigenproblem," *SIAM Journal on Matrix Analysis and Applications*, vol. 15, no. 4, pp. 1266–1276, 1994.

[175] ——, "A divide-and-conquer algorithm for the symmetric tridiagonal eigenproblem," *SIAM Journal on Matrix Analysis and Applications*, vol. 16, no. 1, pp. 172–191, 1995.

[176] D. F. G. Coelho and V. S. Dimitrov, "Fast estimation of tridiagonal matrices largest eigenvalue," in *The 30th Annual IEEE Canadian Conference on Electrical and Computer Engineering*. Windsor, ON, Canada: IEEE, Apr. 2017.

[177] N. J. Higham, *Accuracy and Stability of Numerical Algorithms*, ser. Applied Mathematics. University of Manchester, Manchester, England: SIAM, 2002.

[178] M. G. *et al.*, "GNU scientific library reference manual, 3rd ed." https://www.gnu.org/software/gsl/manual/gsl-ref.pdf, iSBN: 0954612078.

[179] R. Oste and J. V. der Jeugt, "Tridiagonal test matrices for eigenvalue computations: Two-parameter extensions of the clement matrix," *Journal of Computational and Applied Mathematics*, vol. 314, pp. 30–39, Apr. 2017.

[180] J. W. Goodman, *Statistical Optics*. Stanford, California, USA: Wiley Series in Pure & Applied Optics, 1985.

[181] S. N. Anfinsen, A. P. Doulgeris, and T. Eltoft, "Goodness-of-fit tests for multilook polarimetric radar data based on the Mellin transform," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 49, no. 7, pp. 2764–2781, Jul. 2011.

[182] N. R. Goodman, "Statistical analysis based on a certain multivariate complex Gaussian distribution (an introduction)," *Annals of Mathematical Statistics*, vol. 34, no. 1, pp. 152–177, Mar. 1963.

[183] S. H. Yueh, J. A. Kong, J. K. Jao, R. T. Shin, H. A. Zebker, T. L. Toan, and H. Ottl, "K-distribution and polarimetric terrain radar clutter," *Progress in Electromagnetics Research*, vol. PIER 03, pp. 237–275, 1990.

[184] A. C. Frery, H. J. Muller, C. C. F. Yanasse, and S. J. S. Sant'Anna, "A model for extremely heterogeneous clutter," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 35, no. 3, pp. 648–659, May 1997.

[185] C. C. Freitas, A. C. Frery, and A. H. Correia, "The polarimetric $\mathscr{G}$ distribution for SAR data analysis," *Environmetrics*, vol. 16, no. 1, pp. 13–31, Feb. 2005.

[186] A. P. Doulgeris, "An automatic $\mathscr{U}$-distribution and Markov random field segmentation algorithm for PolSAR images," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 53, no. 4, pp. 1819–1827, Apr. 2015.

[187] A. D. C. Nascimento, R. J. Cintra, and A. C. Frery, "Hypothesis testing in speckled data with stochastic distances," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 48, no. 1, pp. 373–385, Jan. 2010.

[188] H. Skriver, "Crop classification by multitemporal C- and L-band single- and dual-polarization and fully polarimetric SAR," *IEEE Transactions on Geoscience and Remote Sensing*, 2012.

[189] T. S. Shores, *Applied Linear Algebra and Matrix Analysis*. Lincoln, NE: Springer, 2007.

[190] N. Lukac and B. Zalik, "GPU-based roofs' solar potential estimation using liDAR data," *Computers & Geosciences*, vol. 52, no. Supplement C, pp. 34–41, 2013.

[191] M. Steinbach and R. Hemmerling, "Accelerating batch processing of spatial raster analysis using GPU," *Computers & Geosciences*, vol. 45, no. Supplement C, pp. 212–220, 2012.

[192] X. Li, T. Huang, D.-T. Lu, and C. Niu, "Accelerating experimental high-order spatial statistics calculations using GPUs," *Computers & Geosciences*, vol. 70, no. Supplement C, pp. 128–137, 2014.

[193] X. Li, C. Song, S. López, Y. Li, and J. F. López, "Fast computation of bare soil surface roughness on a fermi GPU," *Computers & Geosciences*, vol. 82, no. Supplement C, pp. 38–44, 2015.

[194] The Polarimetric SAR Data Processing and Educational Tool, "PolSARpro database," https://earth.esa.int/web/polsarpro/home, information retrieved on Nov 08, 2017.

[195] F. Aquilante, P.-Å. Malmqvist, T. B. Pedersent, A. Ghosh, and B. O. Roos, "Cholesky decomposition–based multiconfiguration second-order perturbation theory (CD-CASPT2): Application to the spin-state energetics of Co^{III} (diiminato)(NPh)," *Journal of Chemical Theory and Computation*, vol. 5, no. 4, pp. 694–702, Apr. 2008.

[196] S. Wilson, "Universal basis sets and Cholesky decomposition of the two-electron integral matrix," *Computer Physics Communications*, vol. 58, no. 1, pp. 71–81, 1990.

[197] D. S. Kershaw, "The incomplete Cholesky–conjugate gradient method for the iterative solution of systems of linear equations," *Journal of Computational Physics*, vol. 26, no. 1, pp. 43–65, 1978.

[198] G. A. F. Seber, *A Matrix Handbook for Statisticians*, ser. Wiley Series in Probability and Statistics. Hoboken, NJ: John Wiley & Sons, Inc., 2008.

[199] Advanced Micro Devices, "Introduction to OpenCL™ programming," May 2010.

[200] G. Guennebaud, B. Jacob *et al.*, "Eigen v3," http://eigen.tuxfamily.org, 2010.

[201] R Development Core Team, *R: A Language and Environment for Statistical Computing*, R Foundation for Statistical Computing, Vienna, Austria, 2008, http://www.R-project.org/.

[202] D. Eddelbuettel, *RInside*. New York, NY: Springer New York, 2013, pp. 127–137.

[203] Advanced Micro Devices, "AMD APP SDK OpenCL^{TM} user guide," Aug. 2015.

[204] B. Parhami, *Computer Arithmetic: Algorithms and Hardware Designs*, 2nd ed. Oxford, UK: Oxford University Press, 2010.

[205] A. Hasnat, T. Bhattacharyya, A. Dey, S. Halder, and D. Bhattacharjee, "A fast FPGA based architecture for computation of square root and inverse square root," in *Devices for Integrated Circuit (DevIC)*, Mar. 2017, pp. 383–387.

[206] E. Libessart, M. Arzel, C. Lahuec, and F. Andriulli, "A scaling-less Newton–Raphson pipelined implementation for a fixed-point inverse square root operator," in *15th IEEE International New Circuits and Systems Conference (NEWCAS)*, Jun. 2017, pp. 157–160.

[207] A. H. Karp and P. Markstein, "High-precision division and square root," *ACM Transactions on Mathematical Software*, vol. 23, no. 4, pp. 561–589, Dec. 1997.

[208] Joint Collaborative Team on Video Coding (JCT-VC). (2013) HEVC references software documentation. Fraunhofer Heinrich Hertz Institute.

[209] D. Brandwood, *Fourier Transforms in Radar and Signal Processing*. Artech House, Inc, 2003.

[210] R. J. Cintra, "An integer approximation method for discrete sinusoidal transforms," *Circuits, Systems, and Signal Processing*, vol. 30, no. 6, pp. 1481–1501, 2011.

[211] M. H. Hayes, *Statistical Digital Signal Processing and Modeling*, T. VenGraitis, Ed. New York, NY: John Wiley & Sons, Inc., 1996.

[212] E. D. Kolaczyk, "Methods for analyzing certain signals and images in astronomy using Haar wavelets," in *Asilomar Conference on Signals, Systems & Computers*, vol. 1, 1997, pp. 80–84.

Appendix A

List of Publications

The following works were published during the candidate participation in the PhD program:

- DFT Computation using Gauss-Eisenstein Basis: FFT Algorithms and VLSI Architectures, **D. F. G. Coelho**, R. J. Cintra, N. Rajapaksha, G. J. Mendis, A. Madanayake, V. S. Dimitrov - Transactions on Computers, IEEE, Vol 66, No. 8, pp. 1442–1448, Mar 2017;

- Error-Free Computation of 8-point DCT Based on the Loeffler Factorization and Algebraic Integers, **D. F. G. Coelho**, R. J. Cintra, S. Kulasekera, A. Madanayake, V. S. Dimitrov - Signal Processing, IET, Vol 10, No. 6, pp. 633–640, July 2016;

- Efficient Computation of the 8-point DCT via Summation by Parts, **D. F. G. Coelho**, R. J. Cintra, V. S. Dimitrov - Journal of Signal Processing Systems, Springer, pp. 1–10, Aug 2017;

- Efficient Computation of Tridiagonal Matrices Largest Eigenvalue, **D. F. G. Coelho**, V. Dimitrov, L. Rakai - Journal of Computational and Applied Mathematics, Elsevier, Vol 330, pp. 268–275, Mar 2018;

- Fast estimation of tridiagonal matrices eigenvalue, **D. F. G. Coelho**, V. Dimitrov - IEEE 30th Canadian Conference of Electrical and Computer Engineering (CCECE), Windsor, ON, Canada, pp. 1–4 Apr 2017;

- A Parallel Method for the Computation of Matrix Exponential Based on Truncated Neumann Series, V. Dimitrov, V. Ariyarathna, D. F. G. Coelho, L. Rakai, A. Madanayake, R. J. Cintra - IEEE 24th Symposium on Computer Arithmetic (ARITH), London, UK, pp. 35–42, Jul 2017;

- Detection of Neovacularization Near the Optic Disk Due to Diabetic Retinopathy, D. F. G. Coelho, R. M. Rangayyan, V. S. Dimitrov - European Signal Processing Conference, Budapest, Hungary, pp. 2040–2044, Aug 2016.

Besides the seven published results, the following works are submitted or under review:

- Fast Matrix Inversion and Determinant Computation for Polarimetric Synthetic Aperture Radar, **D. F. G. Coelho**, R. J. Cintra, A. C. Frery, V. Dimitrov - Computers & Geosciences, Elsevier, Jan 2018.

- Computation of 2D 8×8 DCT Based on the Loeffler Factorization Using Algebraic Integer Encoding, **D. F. G. Coelho**, S. Nimmalapalli, V. S. Dimitrov, A. Madanayake, R. J. Cintra, A. Tisserand - under review in Transactions on Computers, IEEE, Oct 2017.

- Low-complexity DCT Approximations Based on the Loeffler Matrix Parametrization for Image and Video Coding, **D. F. G. Coelho**, R. J. Cintra, F. M. Bayer, S. Kulasekera, A. Madanayake, P. A. M. Oliveira, T. L. T. Silveira, R. S. Oliveira, V. S. Dimitrov - submitted to Transactions on Circuits and Systems for Video Technology, IEEE, Dec 2017.

The author of this thesis also had the opportunity to contribute to the advancement of different scientific and technological problems during his participation in the PhD program. Some of the above-mentioned works were not included in the thesis as part of the main document because either (i) the author of this thesis was not the main author of the resulting papers or (ii) the resulting papers were not fitting into the thesis theme. The author of this thesis still finds to be worth to mention those works given that the purpose of a PhD student is to tackle scientific problems, no matter what is its nature.

Appendix B

List of Awards

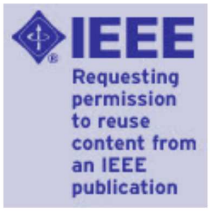
The following awards were received by the candidate during his participation in the PhD program:

- **Academic Excellence Award:** Award for the excellent academic performance in the courses in the graduate program as a PhD student at University of Calgary (2017);
- **Academic Productivity Award:** Award for the outstanding productivity regarding scientific research and publications in highly respected periodicals and conferences in the area of Electrical and Computer Engineering (2017);
- **Electrical and Computer Engineering Open Scholarship Award:** Award for the top 15 students in the Electrical and Computer Engineering Department at the University of Calgary considering research quality and academic performance (2017);
- **Eyes High International Doctoral Scholarship:** Scholarship offered to students that have research with potential impact in line with the University of Calgary long term goals. In particular, the research topics of my PhD work have impact in the design and implementation of devices with reduced energy consumption (eco-friendly devices) (2017);

Appendix C

List of Permissions

The following pages include the copyright permission letters for reusing the contents of published IEEE, IET, Springer, and Elsevier papers in this thesis.



Title: DFT Computation Using Gauss-Eisenstein Basis: FFT Algorithms and VLSI Architectures

Author: Diego F. G. Coelho

Publication: Computers, IEEE Transactions on

Publisher: IEEE

Date: 1 Aug. 2017

Thesis / Dissertation Reuse

The IEEE does not require individuals working on a thesis to obtain a formal reuse license, however, you may print out this statement to be used as a permission grant:

Requirements to be followed when using any portion (e.g., figure, graph, table, or textual material) of an IEEE copyrighted paper in a thesis:

1) In the case of textual material (e.g., using short quotes or referring to the work within these papers) users must give full credit to the original source (author, paper, publication) followed by the IEEE copyright line © 2011 IEEE.
2) In the case of illustrations or tabular material, we require that the copyright line © [Year of original publication] IEEE appear prominently with each reprinted figure and/or table.
3) If a substantial portion of the original paper is to be used, and if you are not the senior author, also obtain the senior author's approval.

Requirements to be followed when using an entire IEEE copyrighted paper in a thesis:

1) The following IEEE copyright/ credit notice should be placed prominently in the references: © [year of original publication] IEEE. Reprinted, with permission, from [author names, paper title, IEEE publication title, and month/year of publication]
2) Only the accepted version of an IEEE copyrighted paper can be used when posting the paper or your thesis on-line.
3) In placing the thesis on the author's university website, please display the following message in a prominent place on the website: In reference to IEEE copyrighted material which is used with permission in this thesis, the IEEE does not endorse any of [university/educational entity's name goes here]'s products or services. Internal or personal use of this material is permitted. If interested in reprinting/republishing IEEE copyrighted material for advertising or promotional purposes or for creating new collective works for resale or redistribution, please go to http://www.ieee.org/publications_standards/publications/rights/rights_link.html to learn how to obtain a License from RightsLink.

If applicable, University Microfilms and/or ProQuest Library, or the Archives of Canada may supply single copies of the dissertation.

BACK CLOSE WINDOW

Ref: DFC/Permission/SPR0175

FAO: Diego F. G. Coelho

4 April 2018

Permission to reproduce IET content

Dear Sirs,

Error-free computation of 8-point discrete cosine transform based on the Loeffler factorisation and algebraic integers ("the Material") to be used in the thesis 'Advanced Methods for Efficient Digital Signal Processing and Matrix-based Computations', by Diego F. G. Coelho, to be published under the University of Calgary ("the Work")

The Institution of Engineering and Technology ("the IET") hereby grants to Diego F. G. Coelho ("DFC") non-exclusive permission to use the Material in the Work subject to the terms set out below:

Territory:	Worldwide
Languages:	All languages
Term:	Life of the Work
Media:	All print and electronic media how known or hereafter devised
Credit:	Coelho, D. F. G., Cintra, R. J., Kulasekera, S., et al.: 'Error-free computation of 8-point discrete cosine transform based on the Loeffler factorisation and algebraic integers', IET Signal Processing, 2016, 10, 6, pp. 633-640
Fee:	None

The permission granted by this letter may not be licensed or assigned without the prior written consent of the IET.

Neither party shall be liable to the other for indirect, incidental, special or consequential damages arising out of or in relation to this agreement (unless such liability cannot be excluded or limited by law).

The parties agree that this letter constitutes the entire agreement between them relating to the use of the Material by DFC and supersedes all previous agreements, understandings and arrangements between them, whether in writing or oral in respect of its subject matter.

No variation of this agreement shall be valid or effective unless it is in writing, refers to this agreement and is duly signed or executed by, or on behalf of, each party.

This letter and any dispute or claim arising out of, or in connection with, it, its subject matter or formation (including non-contractual disputes or claims) shall be governed by, and construed in accordance with, the laws of England and Wales and the parties irrevocably

agree that the courts of England and Wales shall have exclusive jurisdiction to settle any dispute or claim arising out of, or in connection with, this letter, its subject matter or formation (including non-contractual disputes or claims).

Please sign and return the enclosed copy of this letter to confirm your agreement to it.

Yours faithfully

James Sutherland
Permissions Officer

We confirm our agreement to the terms set out above

Signed:
Print name: Diego F. G. Coelho

Date: April 4, 2018

SPRINGER NATURE

Title:	Efficient Computation of the 8-point DCT via Summation by Parts
Author:	D. F. G. Coelho, R. J. Cintra, V. S. Dimitrov
Publication:	Journal of Signal Processing Systems
Publisher:	Springer Nature
Date:	Jan 1, 2017

Order Completed

Thank you for your order.

This Agreement between Mr. Diego Coelho ("You") and Springer Nature ("Springer Nature") consists of your license details and the terms and conditions provided by Springer Nature and Copyright Clearance Center.

Your confirmation email will contain your order number for future reference.

printable details

License Number	4321430112679
License date	Apr 03, 2018
Licensed Content Publisher	Springer Nature
Licensed Content Publication	Journal of Signal Processing Systems
Licensed Content Title	Efficient Computation of the 8-point DCT via Summation by Parts
Licensed Content Author	D. F. G. Coelho, R. J. Cintra, V. S. Dimitrov
Licensed Content Date	Jan 1, 2017
Licensed Content Volume	90
Licensed Content Issue	4
Type of Use	Thesis/Dissertation
Requestor type	non-commercial (non-profit)
Format	print and electronic
Portion	full article/chapter
Will you be translating?	no
Circulation/distribution	<501
Author of this Springer Nature content	yes
Title	Mr. Diego Coelho Thesis
Instructor name	Diego Coelho
Institution name	UofC
Expected presentation date	Apr 2018
Attachment	
Requestor Location	Mr. Diego Coelho 165 Rua Maria Ramos Recife, PE 52221-130 Brazil Attn: Mr. Diego Coelho
Billing Type	Invoice
Billing address	Mr. Diego Coelho 165 Rua Maria Ramos

Recife, Brazil 52221-130
Attn: Mr. Diego Coelho

Total 0.00 USD

ORDER MORE CLOSE WINDOW

Title: Efficient computation of tridiagonal matrices largest eigenvalue

Author: Diego F.G. Coelho,Vassil S. Dimitrov,L. Rakai

Publication: Journal of Computational and Applied Mathematics

Publisher: Elsevier

Date: 1 March 2018

Gmail **Diego Coelho**

Authorization for the Use of Papers in Thesis

2 messages

Diego Coelho Thu, Apr 5, 2018 at 9:26 AM
To: Vassil Dimitrov

Dear Prof. Vassil,

I am writing to you in connection to the papers we have published/submitted together as a result of our collaboration and my PhD thesis. I would like to use them as part of my dissertation, therefore, I am asking your permission for the use of the following paper:

Fast Matrix Inversion and Determinant Computation for Polarimetric Synthetic Aperture Radar, D. F. G. Coelho, R. J. Cintra, A. C. Frery, V. Dimitrov - Computers & Geosciences, Elsevier, Jan 2018.

Proper acknowledgement is giving on my thesis for the collaboration.

If you agree, you can just reply with an 'yes'. I need your response to attach it to the thesis.

--
Best,
Diego F. G. Coelho.

Vassil Dimitrov Thu, Apr 5, 2018 at 10:16 AM
To: Diego Coelho

Yes, I do authorize this!

Vassil
[Quoted text hidden]

Gmail Diego Coelho

Authorization for the Use of Papers in Thesis

Diego Coelho Thu, Feb 15, 2018 at 11:20 AM
To: Renato J Cintra
Cc: Vassil Dimitrov

Dear Prof. Cintra,

I am writing to you in connection to the papers we have published/submitted together as a result of our collaboration and my PhD thesis. I would like to use them as part of my dissertation, therefore, I am asking your permission for the use of the following papers:

DFT Computation using Gauss-Eisenstein Basis: FFT Algorithms and VLSI Architectures, D. F. G. Coelho, R. J. Cintra, N. Rajapaksha, G. J. Mendis, A. Madanayake, V. S. Dimitrov - Transactions on Computers, IEEE, Vol 66, No. 8, pp. 1442–1448, Mar 2017;

Error-Free Computation of 8-point DCT Based on the Loeffler Factorization and Algebraic Integers, D. F. G. Coelho, R. J. Cintra, S. Kulasekera, A. Madanayake, V. S. Dimitrov - Signal Processing, IET, Vol 10, No. 6, pp. 633–640, July 2016;

Fast Matrix Inversion and Determinant Computation for Polarimetric Synthetic Aperture Radar, D. F. G. Coelho, R. J. Cintra, A. C. Frery, V. Dimitrov - Computers & Geosciences, Elsevier, Jan 2018.

Proper acknowledgement is giving on my thesis for the collaboration.

If you agree, you can just reply with an 'yes'.

--
Best,

4/5/2018 Gmail - Authorization for the Use of Papers in Thesis

Gmail Diego Coelho

Authorization for the Use of Papers in Thesis

R J Cintra Thu, Feb 15, 2018 at 1:54 PM
To: Diego Coelho
Cc: Vassil Dimitrov

Diego:

I authorize all of them.

BR

RJC
[Quoted text hidden]

Gmail **Diego Coelho**

Authorization for the Use of Papers in Thesis

Diego Coelho Thu, Feb 15, 2018 at 11:26 AM
To: "Alejandro C. Frery"
Cc: Vassil Dimitrov

Dear Prof. Frery,

I am writing to you in connection to the papers we have submitted together as a result of our collaboration and my PhD thesis. I would like to use them as part of my dissertation, therefore, I am asking your permission for the use of the following paper:

Fast Matrix Inversion and Determinant Computation for Polarimetric Synthetic Aperture Radar, D. F. G. Coelho, R. J. Cintra, A. C. Frery, V. Dimitrov - Computers & Geosciences, Elsevier, Jan 2018.

Proper acknowledgement is giving on my thesis for the collaboration.

If you agree, you can just reply with an 'yes'.

Best,
Diego F. G. Coelho.

4/5/2018 Gmail - Authorization for the Use of Papers in Thesis

Gmail **Diego Coelho**

Authorization for the Use of Papers in Thesis

Alejandro C. Frery Thu, Feb 15, 2018 at 2:23 PM
To: Diego Coelho

Dear Diego,

Sure, yes.

Alejandro
[Quoted text hidden]