Efficient Algorithms and Architectures with High Accuracy and Low Complexity for DSP Applications

A DISSERTATION
SUBMITTED TO THE FACULTY OF THE GRADUATE SCHOOL
OF UNIVERSITY OF MINNESOTA
BY

SANGHO YUN

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR THE DEGREE OF
DOCTOR OF PHILOSOPHY

GERALD E. SOBELMAN, ADVISOR

MAY, 2016

Acknowledgements

I am most grateful to my advisor Professor Gerald E. Sobelman, for his support, guidance and encouragement throughout the time of studying the entire Ph. D program and finally this thesis. His careful review and constructive feedback greatly improve the quality of this work.

I would like to express my sincere gratitude to my committee members, of Professor Marc Riedel, Professor Thomas A. Posbergh, and Professor Andrew M. Odlyzko for their insightful and valuable comments on my thesis.

I want to express my deepest appreciation and love to my parent for their unlimited love and support, for everything they have done for me through my life. Also, special thanks to my sister for always believing in me.

Lastly, I would like to extend my innermost appreciation to my beloved wife, Eunju Yang. She has dedicated herself astoundingly to support my Ph. D program. Without your support and encouragement, I would not have achieved this milestone. I also thank my sweet daughter; Leah Yun to be such a lovely kid. I truly love you forever.

Dedication

Dedicated to my parents

for their love, great devotion and sacrifice

Dedicated to my beloved wife, Eunju

for her love, endless support and encouragement

Dedicated to my sweet daughter, Leah

who makes me smile all day long

Abstract

In this thesis, we investigate efficient algorithms and architectures having high accuracy and low complexity for Digital Signal Processing (DSP) applications. We propose several design approaches to meet the requirements of real-time throughput and input-output latency, which are characteristic features of DSP applications. The approaches we propose include the use of efficient arithmetic representations, as well as creating structures with pipelining and parallel processing. The architectures also provide reduced hardware complexity by converting certain mathematical operations in the algorithms into simple DSP operations.

The main topics are addressed in this thesis are the following: A complex multiplier using fused arithmetic units, a Fast Fourier Transform (FFT) processor using Algebraic Integer (AI) encoding, and adaptive wavelet transforms based on the lifting scheme. These topics are discussed in terms of the mathematical algorithms involved and their hardware implementations. To provide quantitative comparisons with previous work, we conduct fixed-point simulations with appropriate data wordlengths and implement architectures using field programmable gate array (FPGA) and Very Large-Scale Integration (VLSI) based platforms. Our results show improvements in performance and/or hardware complexity for the algorithms that are considered in this thesis.

Table of Contents

List of Tables

List of Figures

Chapter. 1
Introduction

Digital Signal Processing (DSP) is the mathematical manipulation of information-bearing signals by the use of various algorithms. The objectives of DSP are to analyze, modify, enhance and/or extract useful information in a signal. Most of the signals that are usually encountered in real life are analog. More specifically, they are continuous-time signals having continuous variations in amplitude. In order to manipulate these signals with a DSP processor, the signals must first be sampled at regular intervals and quantized into digital form. The resulting discrete-time signals having discrete-valued amplitudes are merely sequences of finite-precision number. These sequences can be processed using simple mathematical operations such as addition, multiplication and time delay.

1. 1. Motivation

Prior to the 1960s, it was common that the technology for signal processing was almost exclusively based on analog manipulations. The rapid evolution into digital forms is a result of the significant advances in digital computers and microprocessors, which can be fabricated as integrated circuits with low cost. The advances were reinforced through important theoretical developments such as the Fast Fourier Transform (FFT) algorithm, parametric signal modeling, multi-rate sampling technique, multiphase FIR filters and wavelet analysis [1].

As the limitations of implementations are being continuously reduced by new technology and through the flexibility of a design combining software and hardware, a variety of sophisticated algorithms for DSP have been introduced. The fundamental DSP operations consist of convolution, correlation, filtering, transformation and modulation [2]. One of the most important properties that distinguishes DSP from other general purpose computations is the real-time throughput requirement. The hardware should be designed in order to meet the sampling rate of the application. This is important in order to meet the constraints of real-time processing, where the system executes the computations within the specified time and meets strict limits on the delay between input and output [3]-[4]. This thesis presents new algorithms and hardware design approaches for several important DSP applications.

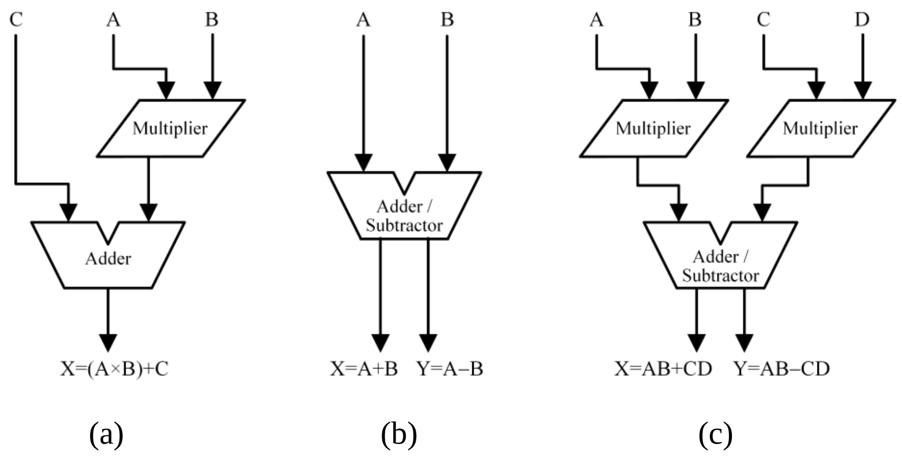


Figure 1.1 Block diagram of the floating-point fused arithmetic units. (a) FMA, (b) FAS, (c) FDP

1. 2. Background Overview

1. 2. 1 The Floating-Point Fused Arithmetic Unit

Since the 1980s, the development of the microprocessor made digital signal processing more versatile and powerful. The achievement of powerful, low-priced designs allows fixed-point and floating-point microprocessors to be widely used for DSP algorithms. These advances have provided a large opportunity for DSP applications to proliferate.

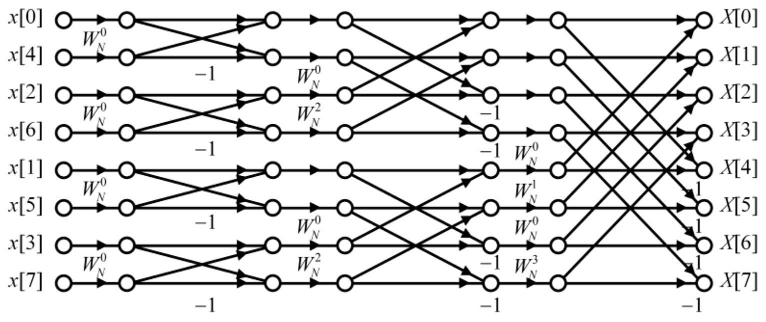
IBM introduced the floating-point fused multiply-add (FMA) unit on the RISC System 6000 (IBM RS/6000) chip in 1990 [5]. The FMA unit computes a floating-point multiplication followed by a floating-point addition. The operation is also known as a multiplier-accumulator or a MAC operation. When done with floating-point numbers in a single FMA operation (A×B)+C, it obtains the result of the operation with higher accuracy by performing a single final rounding, which characterizes the fused operation.

This advantage of the FMA lead to its incorporation in the IEEE 754-2008 floating-point standard [6]. Other fused floating-point arithmetic units have been proposed to provide additional capabilities. In particular, a floating-point fused add-subtract (FAS) unit [7] and a floating-point fused dot-product (FDP) unit [8], as shown in the Figure 1.1 along with the FMA, provide not only higher accuracy but also fast computational performance in many hardware designs. These floating-point fused arithmetic units can speed up and improve the accuracy of complex computations such as the dot product, FFT and matrix multiplication.

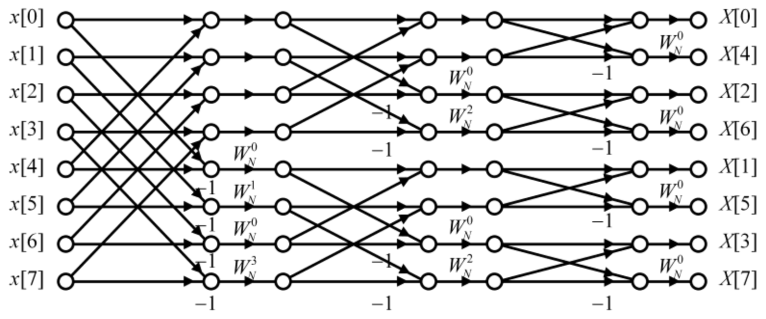
1. 2. 2 Fast Fourier Transform (FFT)

The Fast Fourier transform (FFT) is a well-known collection of algorithms to compute the discrete Fourier transform (DFT) and its inverse. An efficient algorithm for the computation of the DFT was proposed by Cooley and Tukey in 1965 [9] and many other researchers have developed other, alternative approaches. The FFT is one of the most important algorithms in signal processing and data analysis.

The FFT architecture mainly consists of multiple stages of butterfly processing units and associated control signals. In addition, complex multipliers are required for multiplying quantities by certain complex numbers known as twiddle factors. In some implementations, pre-computed twiddle factors are stored in a ROM and retrieved when needed [10]. However, the ROM has the drawback that it occupies a large area and has potentially high power consumption, particularly for large-length FFTs [11]. There are two basic classes of algorithms for implementing the FFT, decimation in time (DIT) and



(a)



(b)

Figure 1.2 Signal flow graph for 8-poiont Radix-2 FFT. (a) 8-point radix -2 DIT FFT, (b) 8-point radix -2 DIF FFT

decimation in frequency (DIF). Figure 1.2 shows the signal flow graphs for an 8-point, radix-2 DIT FFT and a DIF FFT, respectively.

1. 2. 3 Adaptive Wavelet Transform

The wavelet transform is similar to the Fourier transform in terms of decomposing the signals into basis functions. It uses functions that are localized in square-integrable (real- or complex- valued) space. The discrete wavelet transform (DWT) is a form of implementation that uses a discrete set of scaled and shifted functions. The DWT has had

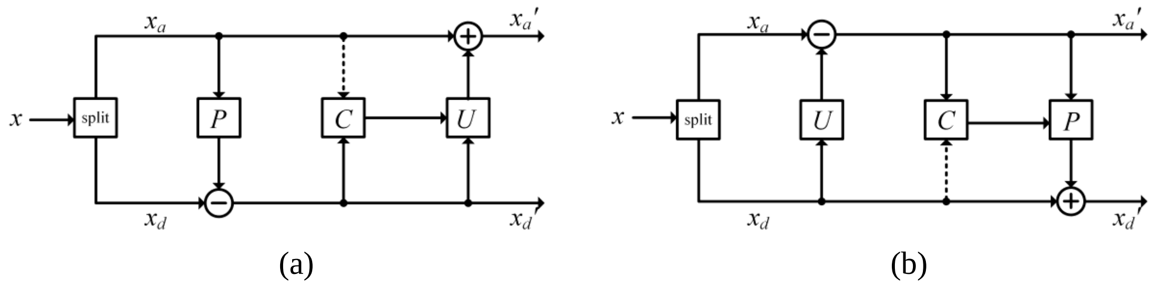


Figure 1.3 (a) Adaptive update-lifting scheme, (2) Adaptive prediction-lifting scheme.

numerous applications in signal, image and video processing. It has been used in the JPEG-2000 image standard [12] and in the MPEG-4 video standard [13], as well as in applications such as compression, denoising and watermarking. Moreover, the lifting-based wavelet scheme [14]-[15] can reduce the computational complexity compared to traditional approaches.

The adaptive directional lifting-based (ADL) wavelet transform has been proposed to achieve better reconstruction. Gerek and Çetin presented a 2-D modification of the lifting scheme for the 5/3 wavelet by considering diagonal prediction [16]. Ding *et al*. [17] proposed an ADL wavelet transform for image coding which incorporates directionally spatial predictions into the conventional 2-D lifting scheme. Liu *et al*. [18] proposed a weighted adaptive lifting (WAL)-based wavelet transform which was designed to further improve the performance of the ADL approach.

Some wavelet families, such as the Daub-4 wavelet with two vanishing moments and the Daub-6 wavelet with three vanishing moments, have irrational wavelet coefficients. In terms of an implementation, multiplications with approximations to the irrational coefficients lead to error accumulation in the final transform results, which may cause degradation of the image quality. Since algebraic integer quantization (AIQ) was first introduced by Cozzens and Finkelstein [19], it has been employed in the DWT to reduce these computational errors. By mapping irrational wavelet coefficients into vectors or arrays of integers, the computations in the wavelet decomposition can be achieved without errors in the resulting integer calculations.

1. 3. Summary of Thesis Contributions

This thesis focuses on the efficient algorithms and architectures having high accuracy and low complexity for DSP applications. The algorithms and architectures in this thesis are designed to meet real-time processing requirements. The specific contributions of this thesis are as follows:

Chapter 2: The design and implementation of a novel and efficient low-complexity floating-point complex multiplier is presented. The proposed TT-FDP unit occupies less area than two FDP units, which jointly perform the same arithmetic operation due to its smaller number of multiplier trees. By using the TT-FDP, the proposed complex multiplier occupies less area and exhibits reasonable average and variance values of the relative errors.

Chapter 3: An efficient architecture of the 64-point FFT processor using two-dimensional algebraic integer (AI) encoding is presented. The advantage of two-dimensional AI encoding is that the hardware complexity for multiplication is reduced since a multiplication can be replaced by a few simple shifts and additions. The ROMless FFT processor, which replaces the ROM for twiddle factors with a twiddle factor generator (TFG) using 2-D AI encoding, has less hardware complexity than an existing approach. The proposed architecture has an acceptable SNR and requires less hardware resources on an FPGA.

Chapter 4: An efficient architecture for the wavelet transform using the adaptive directional lifting (ADL) scheme with algebraic integers is introduced. The proposed architecture has low complexity and efficient hardware utilization without additional latency by applying parallel processing and by reordering the input data stream for real-time processing. To avoid accumulating errors from irrational coefficients in the lifting scheme, the proposed architecture uses algebraic integers with appropriate input scaling parameters and this technique leads to a reduced hardware complexity due to the elimination of multipliers.

Chapter 5: This chapter presents efficient algorithms and VLSI architectures for two new adaptive wavelet transforms, the Adaptive Curved Wavelet Transform (ACWT) and the Gradient-based ACWT (GACWT). It also proposes a novel associated image quality metric, GACWT-SSIM. ACWT uses diagonal edges in the predication step in addition to the standard vertical and horizontal directions in a two-dimensional wavelet transform,

and the GACWT also includes gradient features. These procedures can reduce the prediction error signal energy in the lower band, which leads to better image quality at lower bit rates. The proposed algorithms outperform the previous curved wavelet transform algorithm in terms of the quality of the highly compressed image through the suppression of ringing artifacts. The simulation results show that the proposed algorithms have better coding performance for several standard benchmark images with an average improvement for GACWT of 1.89 dB in PSNR and 0.0348 in the structural similarity measure compared to the existing approach. In a 45 nm CMOS technology, the synthesis results indicate that the proposed architectures can operate at a clock frequency of approximately 250 MHz to achieve a 4 Gb/s throughput.

Chapter. 2

A Low Complexity Floating-Point Complex Multiplier with a Three-term Dot-Product Unit

2. 1 Introduction

A fused multiply-add unit performs floating-point multiplication and addition in one step with a single rounding procedure. When designed and implemented inside a microprocessor, an FMA is known to be faster than a multiply operation followed by addition. The FMA operation is also included in the IEEE 754-2008 standard. It was first introduced as the multiply-add fused unit in the IBM POWER1 processor [5], and it has been utilized in many other signal processing and graphics applications. For example, an

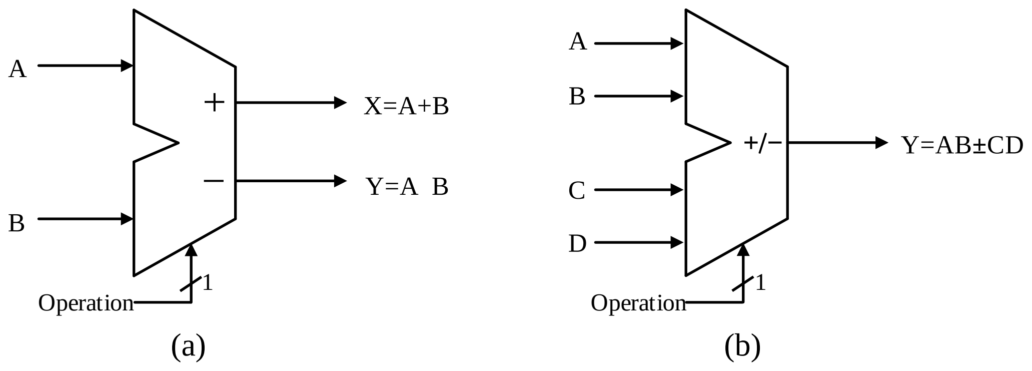


Figure 2.1 Conceptual diagrams of FAS [4] and FDP [5], (a) Fused Add-Subtract Unit, (b) Fused Dot-Product Unit.

FFT processor was designed using floating-point FMA instructions [20]. In addition, power-efficient FIR filters were proposed with the use of FMA units [21]

Other types of fused floating-point arithmetic units have also been proposed to provide additional capabilities. In particular, a floating-point fused add-subtract (FAS) unit [7] and a floating-point fused dot-product (FDP) unit [8] perform the following arithmetic operations:

$$\text{FAS}: \quad x = a + b \text{ and } \; y = a - b$$

$$\text{FDP}: \quad x = a \cdot b \pm c \cdot d$$

The FAS unit saves area and power consumption without any time overhead compared to the conventional parallel implementation for obtaining the final add and subtract results. The FDP unit has low latency and a substantial saving in area and power consumption when compared to a conventional parallel dot-product unit. In addition, the FDP produces more accurate results because only the final result is rounded, whereas three rounding operations are required in the parallel dot-product approach. Block diagrams of these two computational primitives are shown in Figure 2.1.

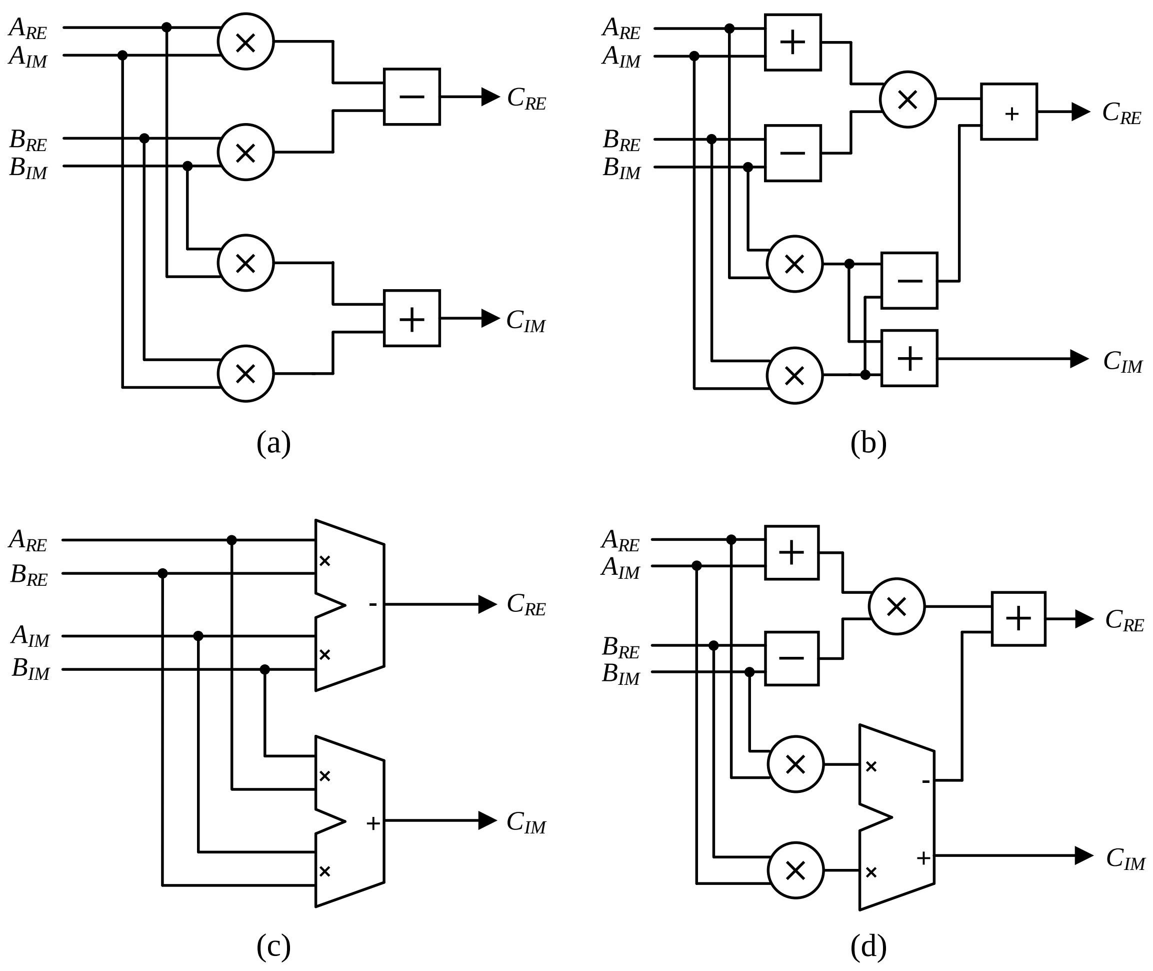


Figure 2.2 Diagrams of floating-point complex multipliers. (a) Conventional complex multiplier, (b) Golub complex multiplier, (c) Conventional complex multiplier with two FDP units, (d) Golub complex multiplier with a FAS unit.

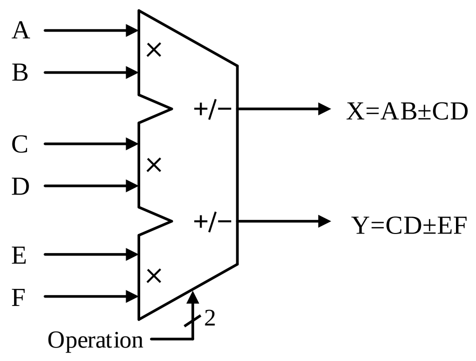


Figure 2.3 Conceptual diagram of Three-term Fused Dot-Product Unit.

This chapter presents the implementation of a new and more efficient floating-point complex multiplier for IEEE-754 single precision floating-point numbers. The remainder of this chapter is organized as follows. In section 2.2, we give a brief overview of previous work and introduce the proposed design. Section 2.3 provides the error analysis and section 2.4 gives the implementation results, including comparisons with previous designs. Finally, section 2.5 presents the conclusions.

2. 2 Design Approach

2. 2. 1 Previous Approaches for Complex Multiplication

There are two basic approaches to complex multiplication. The first one is the conventional method that uses four multiplications and two additions. The second is Golub's method requiring three multiplications and five additions [22]. In many implementations, addition is much less complex and considerably faster than multiplication. Because of the fewer number of multiplications than in the conventional method, Golub's method can reduce the complexity with a slight increase of latency. A complex multiplier design using two FDP units as shown in Figure 2.2(c) has been shown to have the lowest complexity, area and power dissipation among four implementations that were considered [23].

2. 2. 2 Low Complexity Complex multiplier with Three-term Dot-Product Unit

Multiplication of complex numbers is commutative and associative, and the distributive law can also be applied. So we can expand the operation using these properties as follows:

$$\begin{aligned} C &= (A_{RE} + A_{IM} i) \cdot (B_{RE} + B_{IM} i) \\ &= (A_{RE} B_{RE} - A_{IM} B_{IM}) + (A_{RE} B_{IM} + A_{IM} B_{RE}) i \\ &= C_{RE} + C_{IM} i \end{aligned} \tag{2.1}$$

$$\begin{aligned} C_{RE} &= A_{RE} B_{RE} - A_{IM} B_{IM} \\ &= \underline{\underline{(B_{RE} + B_{IM}) A_{RE}}} - (A_{RE} + A_{IM}) B_{IM} \end{aligned} \tag{2.2}$$

$$\begin{aligned} C_{IM} &= A_{RE} B_{IM} + A_{IM} B_{RE} \\ &= \underline{\underline{(B_{RE} + B_{IM}) A_{RE}}} + (A_{IM} - A_{RE}) B_{RE} \end{aligned} \tag{2.3}$$

The result of complex multiplication has two terms, the real term C_{RE} and the complex term C_{IM}. Each expanded equation shown in equation (2.2) and (2.3) contains the same double-underlined term as an intermediate value. The complex multiplier with two terms can be implemented by combining two equations into one unit to reduce the hardware complexity [24]. Figure 2.4(a) shows the diagram of a complex multiplier which uses only three real multiplications. An FDP uses two multiplications and an addition. For a complex multiplication, the result is obtained in parallel with two independent FDP units as shown in Figure 4(b). The overlapped multiplying operations in each of two FDPs can be merged to form the proposed three-term fused dot-product

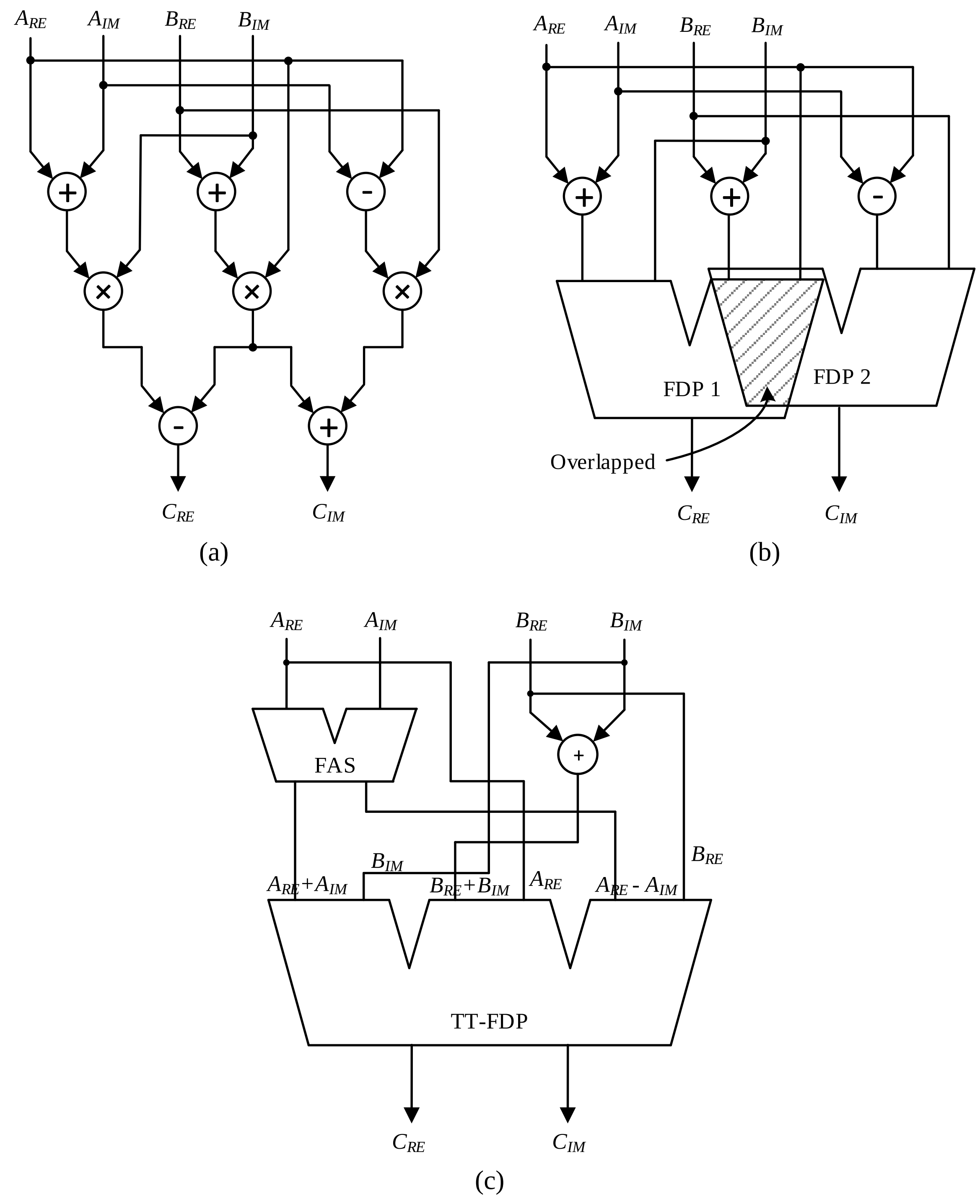


Figure 2.4 Diagram of a new complex multiplier. (a) Complex multiplier using primitive floating-point arithmetic units, (b) Complex multiplier using two FDPs, (c) Complex multiplier using a three-term FDP unit.

(TT-FDP) unit. A TT-FDP unit performs the following arithmetic operations and is depicted in the conceptual diagram shown in Figure 2.3:

$$\text{TT-FDP}: \quad x = a \cdot b \pm c \cdot d \quad \text{and} \quad y = c \cdot d \pm e \cdot f$$

The TT-FDP unit can be implemented by modifying and merging two FDP units. An FAS unit, which is another fused arithmetic unit, can perform add and subtract operations simultaneously. Based on the above two units, a complex multiplier has been developed to reduce the hardware cost and power consumption, as shown in Figure 2.4(c).

Figure 2.5 presents the hardware architecture of the TT-FDP unit. A TT-FDP unit is composed of three multiplier trees and is used to calculate two output values. The number of multiplier trees is highly significant because each multiplier tree occupies a large portion of the area. The hardware has fewer multiplier trees to perform two floating dot-product operations than two FDP units, which contain a total of four multiplier trees. Thus, we can expect to have an overall lower complexity by using equation (2.2) and (2.3) combined instead of equation (2.1) for complex multiplication.

2. 3 Error Analysis

In this section the issue of accuracy is considered. Here, we only examine a single term which is a monomial in a complex multiplication. Multiplying two complex numbers is accomplished in a manner similar to multiplying two binomials and the monomials of the binomial result are independent of each other.

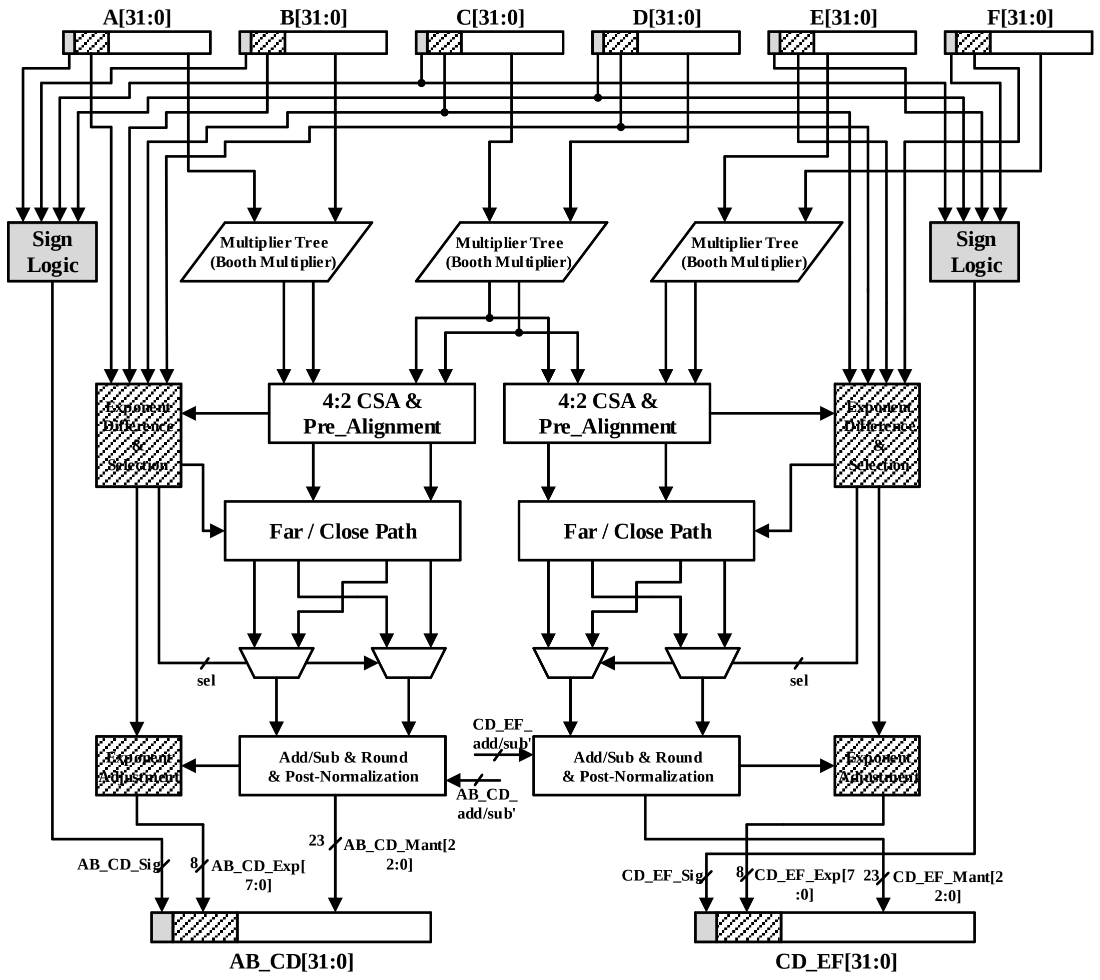
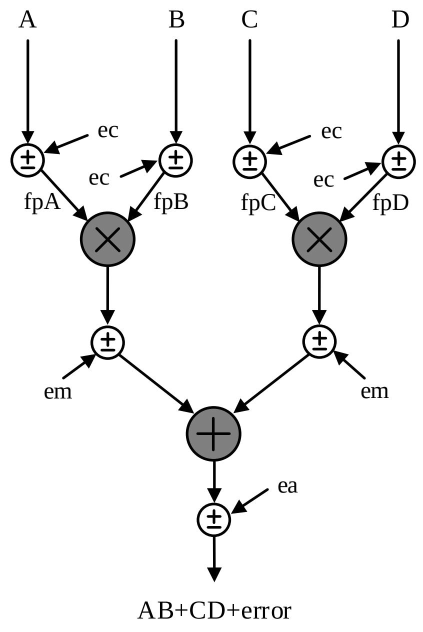
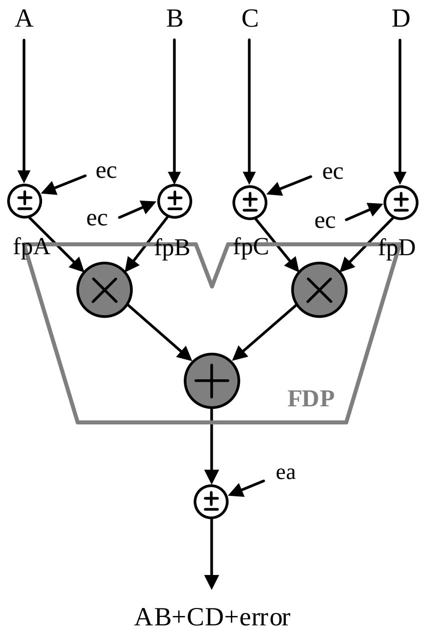


Figure 2.5 Block diagram of a Three-term Fused Dot-Product Unit.

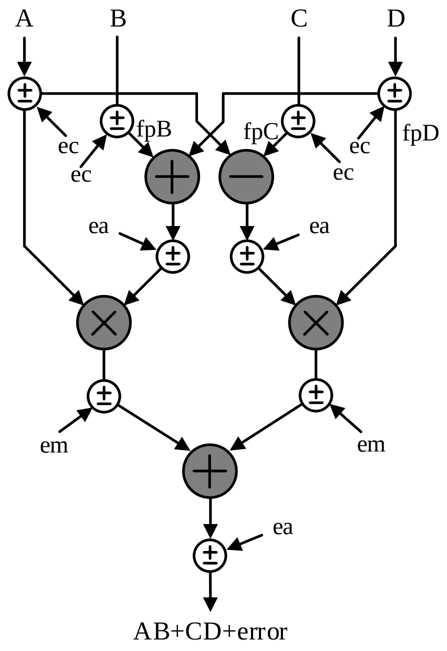
Several error analysis models were implemented in Matlab. The first model (a) is a conventional complex multiplier which consists of two FPMs and an FPA. The model (c) consists of two FPMs and three FPAs. Models (a) and (c) are identical in mathematical structure because they have the same number of rounding operations. Models (b) and (d) have an FDP unit, so there will be less round-off error from addition. The model (b) is an



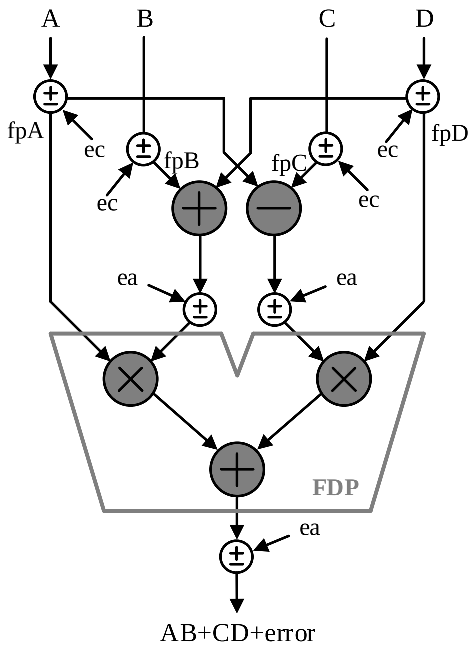
(a)



(b)



(c)



(d)

Figure 2.6 Various error models of floating-point complex multipliers.

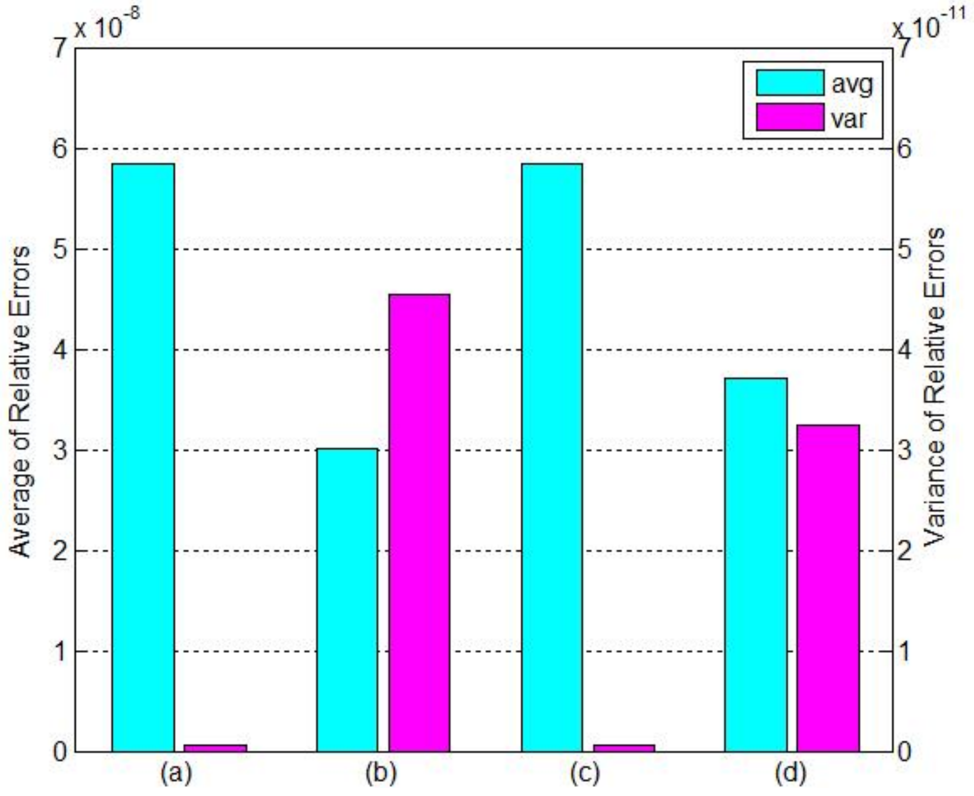


Figure 2.7 Average and variance of relative errors.

FDP unit and the model (d) is proposed for a low complexity floating-point complex multiplier, which is composed of 2 FPMs and an FDP. The model (d) computes the same function as one of the outputs of the TT-FDP. (The other output function has the same error properties). Each of the input sets is generated from 32-bit binary numbers, and input values may have special cases such as Zero, Denormals, Positive/Negative Infinity, SNaN and QNaN. The total number of simulation input sets used is 5 million and the round-to-nearest-even scheme was used.

Table 2.1 Average and Variance of the Relative Errors

	(a)	**(b)**	**(c)**	**(d)**
Average	5.8628×10^{-8}	3.0232×10^{-8}	5.8628×10^{-8}	3.7232×10^{-8}
Variance	5.6434×10^{-13}	4.5463×10^{-11}	5.6440×10^{-13}	3.2536×10^{-11}

Table 2.1 shows the average value and variance for the relative errors. The model (b) has the lowest average and highest variance among the four models. The model (d) has a slightly higher average error but it has a lower error variance than model (b).

2. 4 Implementation and Synthesis Results

To quantify the area savings obtained by using the proposed TT-FDP unit, the following fused floating-point units were implemented:

(a) Conventional complex multiplier with floating-point adders and floating-point multipliers	Figure 2.2 (a)
(b) Complex multiplier with two FDP units	Figure 2.2 (c)
(c) Golub complex multiplier	Figure 2.2 (b)
(d) Proposed complex multiplier with TT-FDP	Figure 2.2 (c)

All units were modeled in synthesizable Verilog HDL and simulated to verify their functionality with 5 million randomly generated numbers in IEEE-754 single precision

Table 2.2 Implementation Results of Floating-point Arithmetic Units

Units	Area (μm^2)	Latency (ns)	Power (mW)
FPA	2,765	1.18	1,093
FPM	6,807	1.40	4,638
FAS	2.779	1.18	1,529
FDP	11,043	2.25	7,802
TT-FDP	13,202	2.27	12,217

Table 2.3 Implementation results of Complex Multipliers

Units	Area		Latency		Power	
	μm^2	%	ns	%	mW	%
(a)	32,758	175	2.58	75	21,542	146
(b)	22,086	118	2.25	65	15,604	106
(c)	34,246	183	4.76	138	19,982	136
(d)	18,746	100	3.45	100	14,732	100

format. After complete verification of the design functionality, they were then synthesized using a 45-nm standard-cell library optimized for a 1.1V supply voltage.

Table 2.2 shows the implementation results of floating-point arithmetic units used in complex multiplication. The TT-FDP unit occupies an area of 13,202 μm^2, while two FDP units that perform the same arithmetic operation as the TT- FDP occupy more area than a single TT-FDP. The proposed complex multiplier unit was implemented as shown in Figure 4 and compared with the other multipliers presented in [23]. Table 2.3 shows the implementation results for the complex multipliers. The final implementation of the proposed design was optimized to save area by combining the exponent control unit and the sign unit, and by removing some redundant glue logic. The proposed complex multiplier using the TT-FDP achieves a reduction in area of 43% and 16% compared to the conventional complex multiplier and the complex multiplier with two FDPs, respectively.

2. 5 Conclusion

This chapter has presented the design and implementation of a novel and efficient low-complexity floating-point complex multiplier. The proposed TT-FDP unit occupies less area than two FDP units (which jointly perform the same arithmetic operation) because of its smaller number of multiplier trees. By using the TT-FDP, the proposed complex multiplier occupies less area and exhibits quite reasonable average and variance values of the relative errors.

Chapter. 3

A Low Complexity FFT Processor using Two-Dimensional Algebraic Integer Encoding for WLAN Applications

3. 1 Introduction

The Fast Fourier Transform (FFT) and Inverse Fast Fourier Transform (IFFT) are key components of the physical layer (PHY) processing in an orthogonal frequency division multiplexing (OFDM) system. For the IEEE wireless LAN standards 802.11a/g/n/ac, the FFT/IFFT size is determined by the number of subcarriers present in the OFDM transmission. Specifically, a 64-point FFT/IFFT is required, which arises from having 64 subcarriers spaced 312.5 KHz apart, where the symbol time $T_s = 3.2\ \mu s$ [25].

A high throughput rate and low latency are desired in a real-time OFDM system, as are low power operation and low hardware complexity. In order to meet the requirements, several different FFT architectures have been proposed, such as the multi-path delay commutator (MDC) and the single-path delay feedback (SDF) structures used with radix-2 and radix-4 algorithms. [26]-[27]

The FFT architecture consists of multiple stages of butterfly processing units and their associated control signals. In addition, complex multipliers are utilized for multiplying a stage's inputs or outputs with appropriate twiddle factors. Pre-computed twiddle factors may be stored in a ROM and retrieved when applicable, and many implementations have used a ROM-based design [10]. However, the ROM has the drawback that it occupies a large area and it may also have a high power consumption, particularly for large-length FFTs [11].

Cozzens and Finkelstein proposed a parallel algorithm for computation of the Discrete Fourier Transform (DFT) via computations using algebraic-integer quantization (AIQ) [19]. Wahid *et. al*. presented an error-free architecture for an 8×8 Discrete Cosine Transform (DCT) and its inverse using two-dimensional AIQ [28]. AI encoding schemes have also been applied to other applications such as the Discrete Hartley Transform (DHT) [29] and the Discrete Wavelet Transform (DWT) [30]. The advantage of algebraic integer encoding is that the hardware complexity for multiplication is low because a multiplication can be replaced by a few simple shifts and additions.

In this chapter, we present a low-complexity 64-point FFT processor for Wireless Local Area Network (WLAN) applications using two-dimensional algebraic integer encoding. Specifically, a radix-2^2 FFT implementation with a ROMless twiddle factor generator (TFG) using 2-D AI encoding is proposed. The rest of this chapter is organized as follows. Section 3.2 gives a brief overview of the necessary background for this work. Section 3.3 describes the proposed 64-point radix-2^2 FFT architecture. In section 3.4, the simulation results are presented, and the hardware implementation results are given in section 3.5. Finally, the conclusions are stated in section 3.6.

3. 2 Preliminary

3. 2. 1 The Radix-2^2 FFT Algorithm

An FFT is an algorithm for computing the DFT and its inverse that reduces the number of algebraic operations required, such as multiplications and additions. Many algorithms have been proposed for reducing computational complexity [2-3]. In particular, the radix-2^2 algorithm has the lower complexity of radix-4 algorithms while retaining the simpler radix-2 butterfly structure [10].

The DFT of length N is defined as follows:

$$X(k) = \sum_{n=0}^{N-1} x(n) W_N^{nk} \qquad 0 \le k \le N-1 \tag{3.1}$$

where W_N denotes $e^{-j\pi/N}$, the N-th primitive root of unity.

We can derive the radix-2^2 algorithm by applying a 3-dimensional linear index map,

$$n = \left\langle \frac{N}{2} n_1 + \frac{N}{4} n_2 + n_3 \right\rangle_N \qquad 0 \le k_1, k_2 \le 1 \tag{3.2}$$

$$k = \langle k_1 + 2k_2 + 4k_3 \rangle_N \qquad 0 \le n_3, k_3 \le N/4 - 1$$

(3.3)

The common factor algorithm (CFA) takes the form of

$$\begin{aligned} X(k_1+2k_2+4k_3) &= \sum_{n_3=0}^{N/4-1} \sum_{n_2=0}^{1} \sum_{n_1=0}^{1} x(\frac{N}{2}n_1 + \frac{N}{4}n_2 + n_3) W_N^{nk} \\ &= \sum_{n_3=0}^{N/4-1} \sum_{n_2=0}^{1} B_{N/2}^{k_1}(\frac{N}{4}n_2 + n_3)\, W_N^{(N/4 n_2 + n_3)(k_1+2k_2+4k_3)} \\ &= \sum_{n_3=0}^{N/4-1} \left[H_{N/4}(n_3,k_1,k_2) W_N^{n_3(k_1+2k_2)} \right] W_{N/4}^{n_3k_3} \end{aligned} \qquad (3.4)$$

where a secondary butterfly structure $H(n_3, k_1, k_2)$ can be expressed as

$$H(n_3,k_1,k_2) = x(n_3) + (-1)^{k_1} x(n_3 + \frac{N}{2}) + (-j)^{k_1+2k_2}[x(n_3 + \frac{N}{4}) + (-1)^{k_1} x(n_3 + \frac{3N}{4})] \qquad (3.5)$$

Equation (3.4) involves many trivial multiplications by powers of ($-j$) and (-1), which can be implemented by real-imaginary swapping and/or sign inversion. Taking advantage of such trivial multiplications leads to less hardware complexity than would be needed using general complex multipliers.

3. 2. 2 ROMless Twiddle Factor Generation (TFG)

The main advantage of using a multidimensional AI encoding is that it has a sparser representation than single-variable AI encoding. From an exhaustive analysis, a simple encoding for the twiddle factors using two variables α and β has been found. Taking $\alpha = cos(\frac{\pi}{32})$ and $\beta = sin(\frac{\pi}{32})$ we have the following relations:

$$W_N^{n_3(k_1+2k_2)}\Big|_{N=64} = e^{-j\frac{2\pi n_3\ k_1+2k_2}{64}} = \left[cos(\frac{\pi}{32}) - j\,sin(\frac{\pi}{32})\right]^{n_3\ k_1+2k_2}$$

$$= \left[\alpha - j\beta\right]^{n_3\ k_1+2k_2} \qquad 0 \le n_3 \le 15,\ 0 \le k_1 \le 1,\ 0 \le k_2 \le 1 \qquad (3.6)$$

$$\alpha = cos(\frac{\pi}{32}) = \frac{1}{2}\sqrt{2+\sqrt{2+\sqrt{2+\sqrt{2}}}} \qquad (3.7)$$

$$\beta = sin(\frac{\pi}{32}) = cos(\frac{15\pi}{32}) = \frac{1}{2}\sqrt{2-\sqrt{2+\sqrt{2+\sqrt{2}}}} \qquad (3.8)$$

Two elements used in an arbitrary twiddle factor are $cos(\frac{n\pi}{32})$ and $sin(\frac{n\pi}{32})$, and these functions are related to α and β through various identities. We can write expressions for the cosine functions appearing in the twiddle factors in the first quadrant as follows:

$$cos(\frac{\pi}{32}) = \alpha$$

$$cos(\frac{2\pi}{32}) = \alpha^2 - \beta^2$$

$$cos(\frac{3\pi}{32}) = \alpha^3 - 3\alpha\beta^2$$

$$cos(\frac{4\pi}{32}) = \alpha^4 - 6\alpha^2\beta^2 + \beta^4$$

$$cos(\frac{5\pi}{32}) = \alpha^5 - 10\alpha^3\beta^2 + 5\alpha\beta^4$$

$$cos(\frac{6\pi}{32}) = \alpha^6 - 15\alpha^4\beta^2 + 15\alpha^2\beta^4 - \beta^6$$

$$cos(\frac{7\pi}{32}) = \alpha^7 - 21\alpha^5\beta^2 + 35\alpha^3\beta^4 - 7\alpha\beta^6 + \beta^7$$

It is noted that the values in the second quadrant can be computed by using the precomputed values in the first quadrant as follows:

$$\sqrt{2}\,cos(\frac{8\pi}{32}) = 1$$
$$\sqrt{2}\,cos(\frac{9\pi}{32}) = cos(\frac{\pi}{32}) - \beta$$
$$\sqrt{2}\,cos(\frac{10\pi}{32}) = cos(\frac{2\pi}{32}) - 2\alpha\beta$$
$$\sqrt{2}\,cos(\frac{11\pi}{32}) = cos(\frac{3\pi}{32}) - 3\alpha^2\beta + \beta^3$$
$$\sqrt{2}\,cos(\frac{12\pi}{32}) = cos(\frac{4\pi}{32}) - 4\alpha^3\beta + 4\alpha\beta^3$$
$$\sqrt{2}\,cos(\frac{13\pi}{32}) = cos(\frac{5\pi}{32}) - 5\alpha^4\beta + 10\alpha^2\beta^3 - \beta^5$$
$$\sqrt{2}\,cos(\frac{14\pi}{32}) = cos(\frac{6\pi}{32}) - 6\alpha^5\beta + 20\alpha^3\beta^3 - 6\alpha\beta^5$$
$$\sqrt{2}\,cos(\frac{15\pi}{32}) = cos(\frac{7\pi}{32}) - 21\alpha^2\beta^5 + 35\alpha^4\beta^3 - 7\alpha^6\beta + \beta^7 = \sqrt{2}\beta$$

The remainder of the functions from *cos*(17π/32) through *cos*(31π/32) in the third and fourth quadrants are easily computed by taking into account the symmetry and periodicity of trigonometric functions. In addition, the values of *sin*(*n*π/32), which arise from the imaginary parts of the twiddle factors, may be obtained by taking into account appropriate trigonometric properties.

The set of coefficients a_{ij}, where $0 \le i, j \le 7$, that will multiply the various of powers of α and β may be represented in the matrix form shown in Table I. The specific values of these coefficients will be detailed in the following subsection.

Table 3.1 Algebraic Integer Coefficient Representation

$$\begin{array}{c|cccccccc} & \multicolumn{8}{c}{j} \\ \hline i & a_{00} & a_{10} & a_{20} & a_{30} & a_{40} & a_{50} & a_{60} & a_{70} \\ & a_{01} & a_{11} & a_{21} & a_{31} & a_{41} & a_{51} & a_{61} & a_{71} \\ & a_{02} & a_{12} & a_{22} & a_{32} & a_{42} & a_{52} & a_{62} & a_{72} \\ & a_{03} & a_{13} & a_{23} & a_{33} & a_{43} & a_{53} & a_{63} & a_{73} \\ & a_{04} & a_{14} & a_{24} & a_{34} & a_{44} & a_{54} & a_{64} & a_{74} \\ & a_{05} & a_{15} & a_{25} & a_{35} & a_{45} & a_{55} & a_{65} & a_{75} \\ & a_{06} & a_{16} & a_{26} & a_{36} & a_{46} & a_{56} & a_{66} & a_{76} \\ & a_{07} & a_{17} & a_{27} & a_{37} & a_{47} & a_{57} & a_{67} & a_{77} \end{array}$$

3. 2. 3 Multidimensional Algebraic Integer Encoding

We only need to consider *cos*($n\pi/32$) for the twiddle factors in the first and second quadrants, i.e. for $1 \le n \le 15$, which corresponds to equation (3.6) where k_1=1 and k_2=0, since the other twiddle factors can be obtained from them through trigonometric identities. The two-dimensional AI encodings of these cosine functions, which appear in equation (3.9), are given in Table 3.2. As an example, *cos*($7\pi/32$) can be represented exactly as $\alpha^7 - 21\alpha^5\beta^2 + 35\alpha^3\beta^4 - 7\alpha\beta^6 + \beta^7$, where $\alpha = cos(\pi/32)$ and $\beta = sin(\pi/32)$, as shown in Equations (3.7) and (3.8), respectively.

$$\text{Re } W_N^{n_3(k_1+2k_2)}\Big|_{0\le n_3\le 15, k_1=1,\, k_2=0} = cos(\frac{n\pi}{32})\Big|_{0\le n\le 15} = f(\alpha, \beta) = \sum_{i=0}^{7}\sum_{j=0}^{7} a_{ij}\alpha^i\beta^j \qquad (3.9)$$

Table 3.2 2-D Algebraic Integer Encoding for $cos(nx/32)|_{0\le x\le 15}$

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

$$cos(\frac{\pi}{32})=\alpha$$

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

$$cos(\frac{2\pi}{32})$$

$$\begin{bmatrix} 0&0&0&1&0&0&0&0\\ 0&0&0&0&0&0&0&0\\ 0&-3&0&0&0&0&0&0\\ 0&0&0&0&0&0&0&0\\ 0&0&0&0&0&0&0&0\\ 0&0&0&0&0&0&0&0\\ 0&0&0&0&0&0&0&0\\ 0&0&0&0&0&0&0&0 \end{bmatrix}$$

$$cos(\frac{3\pi}{32})$$

$$\begin{bmatrix} 0&0&0&0&1&0&0&0\\ 0&0&0&0&0&0&0&0\\ 0&0&-6&0&0&0&0&0\\ 0&0&0&0&0&0&0&0\\ 1&0&0&0&0&0&0&0\\ 0&0&0&0&0&0&0&0\\ 0&0&0&0&0&0&0&0\\ 0&0&0&0&0&0&0&0 \end{bmatrix}$$

$$cos(\frac{4\pi}{32})$$

$$\begin{bmatrix} 0&0&0&0&0&1&0&0\\ 0&0&0&0&0&0&0&0\\ 0&0&0&-10&0&0&0&0\\ 0&0&0&0&0&0&0&0\\ 0&5&0&0&0&0&0&0\\ 0&0&0&0&0&0&0&0\\ 0&0&0&0&0&0&0&0\\ 0&0&0&0&0&0&0&0 \end{bmatrix}$$

$$cos(\frac{5\pi}{32})$$

$$\begin{bmatrix} 0&0&0&0&0&0&1&0\\ 0&0&0&0&0&0&0&0\\ 0&0&0&0&-15&0&0&0\\ 0&0&0&0&0&0&0&0\\ 0&0&15&0&0&0&0&0\\ 0&0&0&0&0&0&0&0\\ -1&0&0&0&0&0&0&0\\ 0&0&0&0&0&0&0&0 \end{bmatrix}$$

$$cos(\frac{6\pi}{32})$$

$$\begin{bmatrix} 0&0&0&0&0&0&0&1\\ 0&0&0&0&0&0&0&0\\ 0&0&0&0&0&-21&0&0\\ 0&0&0&0&0&0&0&0\\ 0&0&0&35&0&0&0&0\\ 0&0&0&0&0&0&0&0\\ 0&-7&0&0&0&0&0&0\\ 1&0&0&0&0&0&0&0 \end{bmatrix}$$

$$cos(\frac{7\pi}{32})$$

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

$$\sqrt{2}\,cos(\frac{9\pi}{32})$$

$$\begin{bmatrix} 0&0&1&0&0&0&0&0\\ 0&-2&0&0&0&0&0&0\\ -1&0&0&0&0&0&0&0\\ 0&0&0&0&0&0&0&0\\ 0&0&0&0&0&0&0&0\\ 0&0&0&0&0&0&0&0\\ 0&0&0&0&0&0&0&0\\ 0&0&0&0&0&0&0&0 \end{bmatrix}$$

$$\sqrt{2}\,cos(\frac{10\pi}{32})$$

$$\begin{bmatrix} 0&0&0&1&0&0&0&0\\ 0&0&-3&0&0&0&0&0\\ 0&-3&0&0&0&0&0&0\\ 1&0&0&0&0&0&0&0\\ 0&0&0&0&0&0&0&0\\ 0&0&0&0&0&0&0&0\\ 0&0&0&0&0&0&0&0\\ 0&0&0&0&0&0&0&0 \end{bmatrix}$$

$$\sqrt{2}\,cos(\frac{10\pi}{32})$$

$$\begin{bmatrix} 0&0&0&0&1&0&0&0\\ 0&0&0&-4&0&0&0&0\\ 0&0&-6&0&0&0&0&0\\ 0&4&0&0&0&0&0&0\\ 1&0&0&0&0&0&0&0\\ 0&0&0&0&0&0&0&0\\ 0&0&0&0&0&0&0&0\\ 0&0&0&0&0&0&0&0 \end{bmatrix}$$

$$\sqrt{2}\,cos(\frac{12\pi}{32})$$

$$\begin{bmatrix} 0&0&0&0&0&1&0&0\\ 0&0&0&0&-5&0&0&0\\ 0&0&0&-10&0&0&0&0\\ 0&0&10&0&0&0&0&0\\ 0&5&0&0&0&0&0&0\\ -1&0&0&0&0&0&0&0\\ 0&0&0&0&0&0&0&0\\ 0&0&0&0&0&0&0&0 \end{bmatrix}$$

$$\sqrt{2}\,cos(\frac{13\pi}{32})$$

$$\begin{bmatrix} 0&0&0&0&0&0&1&0\\ 0&0&0&0&0&-6&0&0\\ 0&0&0&0&-15&0&0&0\\ 0&0&0&20&0&0&0&0\\ 0&0&15&0&0&0&0&0\\ 0&-6&0&0&0&0&0&0\\ -1&0&0&0&0&0&0&0\\ 0&0&0&0&0&0&0&0 \end{bmatrix}$$

$$\sqrt{2}\,cos(\frac{14\pi}{32})$$

$$\begin{bmatrix} 0&0&0&0&0&0&0&1\\ 0&0&0&0&0&0&-7&0\\ 0&0&0&0&0&-21&0&0\\ 0&0&0&0&35&0&0&0\\ 0&0&0&35&0&0&0&0\\ 0&0&-21&0&0&0&0&0\\ 0&-7&0&0&0&0&0&0\\ 1&0&0&0&0&0&0&0 \end{bmatrix}$$

$$\sqrt{2}\,cos(\frac{15\pi}{32})$$

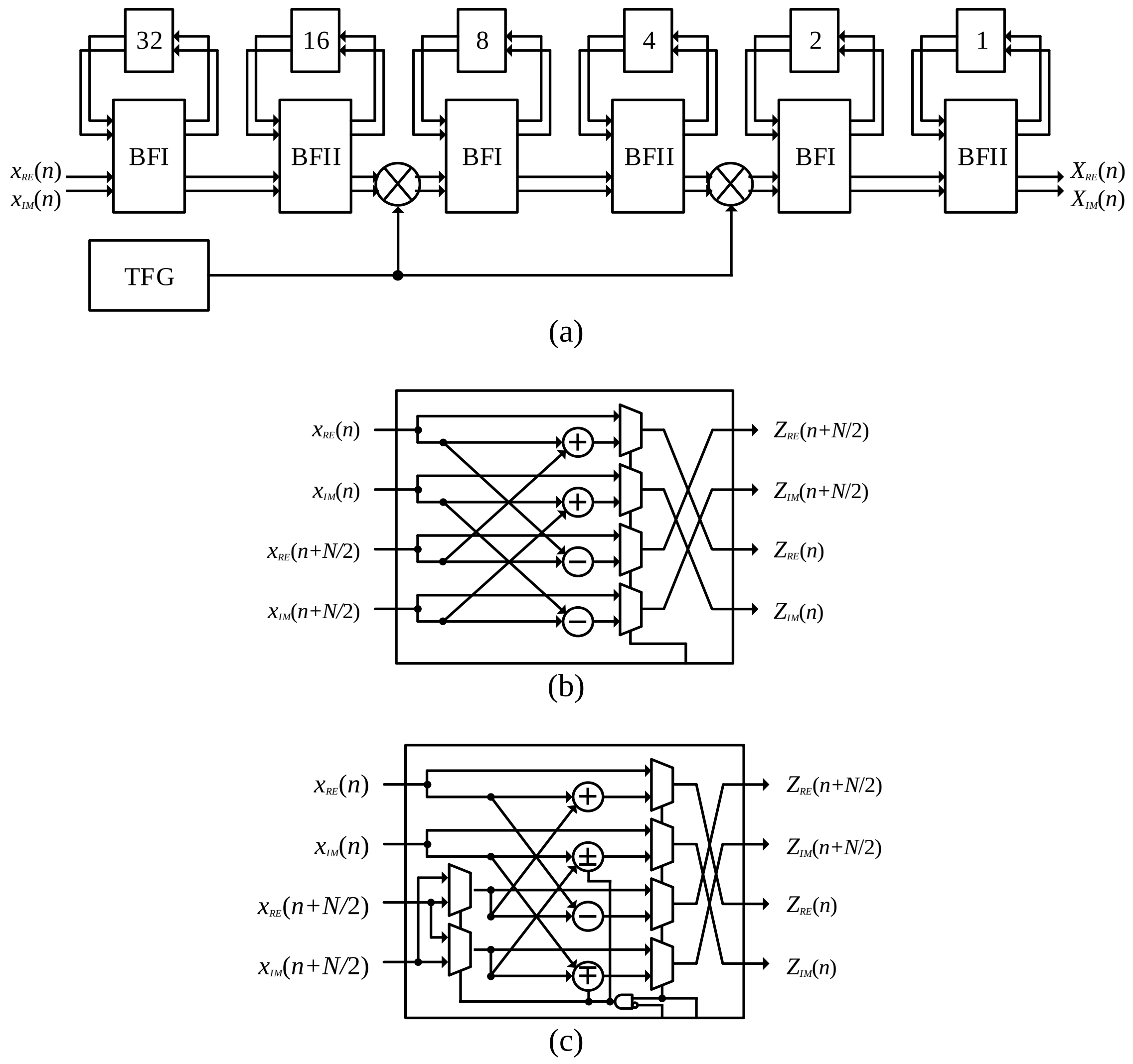


Figure 3.1 64-point FFT architecture (R2^2SDF).

3. 3 Proposed FFT Architecture

The radix-2^2 algorithm has been described in the above section. The proposed architecture uses a TFG instead of a pre-computed ROM-based look-up table. A block

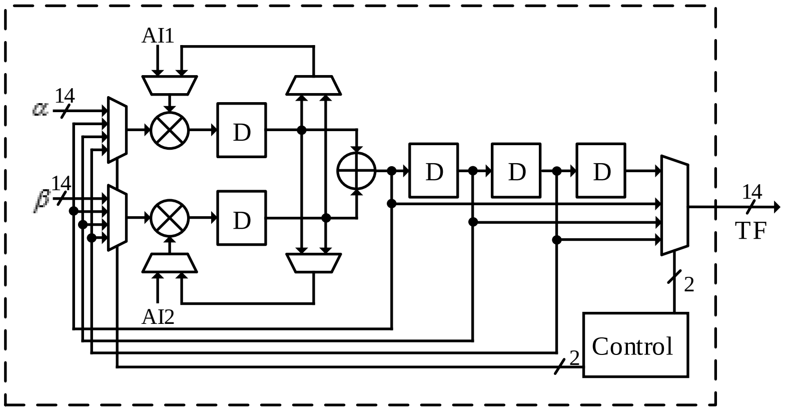


Figure 3.2   Twiddle factor generator (TFG) structure.

diagram of the proposed 64-point FFT architecture is given in Figure 3.1 [10]. The proposed architecture is composed of two types of butterfly units, complex multipliers and a TFG. The design of the TFG is shown in Figure 3.2. The internal wordlengh for the variables α and β was determined set a 14 bits, which was determined through the simulations described in the following section.

3. 4   Simulation and Evaluation

Before implementing the hardware architecture, the appropriate wordlength was determined by fixed-point simulations. Figure 3.3 depicts the simulation results for the SNR as a function of the internal wordlength of the FFT processor. The simulation was repeated 10,000 times for each input SNR in order to obtain an accurate average output SNR result. Based on the simulation results, a wordlength of 14 bits was selected.

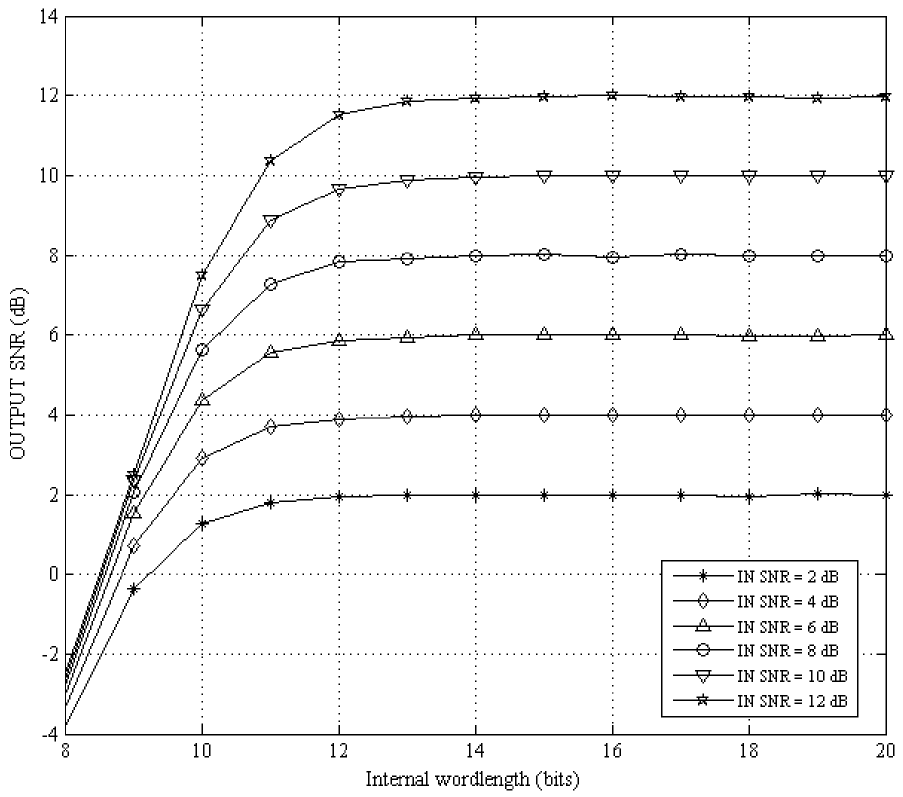


Figure 3.3 Simulation result for SNR versus the internal wordlength.

3. 5 Implementation and Synthesis Results

The proposed FFT processor was designed in Verilog HDL and simulated to verify its functionality. The proposed architecture has been synthesized using appropriate time and area constraints. Table 3.3 compares several FFT implementations. The synthesis step was carried out on the Xilinx Spartan 3E FPGA board in order to be able to compare with previous implementations. As shown in Table 3.3, the proposed 64-point FFT processor

Table 3.3
Comparison of the 64-Point FFT Architecture using FPGA

	Mehra *et al.* [31]	**Ouerhani *et al.* [32]**	**Kumar *et al.* [33]**	**Proposed**
Target	Spartan 3	Spartan 3E	Spartan 3E	Spartan 3E
Wordlength	10	18	14	14
Slices	2,155	758	552	326
Slice LUTs	3,073	NA	1058	563
Slice FF	2,063	1080	695	442
BRAMs	4	0	NA	0
Max. Freq. (MHz)	550	170	102	312

requires less hardware cost than previous FPGA implementations. It is noted that a ROMless FFT processor was achieved and thus there is no necessity of using block RAMs, as the TFG is implemented with combinational logic.

3. 6 Conclusion

An efficient architecture of a 64-point FFT processor for WLAN applications using two-dimensional algebraic integer encoding has been presented. A radix-2^2 algorithm and a 2-dimensional AI encoding can effectively reduce the hardware complexity by minimizing the number of complex multipliers required. The proposed architecture has an acceptable SNR for the application and it has a maximum operating frequency of 312 MHz according to the FPGA synthesis result.

Chapter. 4

Adaptive Directional Lifting Wavelet Transform Architecture using Algebraic Integers

4. 1 Introduction

The discrete wavelet transform (DWT) has been widely used in many areas of signal, image and video processing. It has been applied in the JPEG-2000 image standard [12] and in the MPEG-4 video standard [13], as well as in applications such as compression, denoising and watermarking. Moreover, the lifting-based wavelet scheme [14]-[15] can reduce the computational complexity compared to traditional approaches.

The 2-D DWT can be carried out as a separable transform, which splits the 2-D DWT into 1-D transforms in the horizontal and vertical directions. The approach using separable transforms is convenient for designing filters and for computing the wavelet signals because of its conceptual simplicity. However, the main drawback of using separable transforms is that vanishing moments of the high-pass filters are placed in only two directions (i.e., along the rows and columns) in the image. Edges along other directions can spread the energy into high-frequency subbands, which occurs in images having a large number of curves and oblique lines. As a result, an alternative approach may provide improved performance.

The adaptive directional lifting (ADL) wavelet transform has been proposed for achieving better reconstruction [16]-[18],[34]-[35]. Gerek and Çetin presented a 2-D modification of the lifting scheme for the 5/3 wavelet by considering diagonal prediction [16]. Ding *et al*. [17] proposed an ADL wavelet transform for image coding which incorporates directional prediction into the conventional 2-D lifting scheme. Liu *et al*. [18] propose a weighted adaptive lifting (WAL) wavelet transform which is designed to further improve the performance of the ADL approach.

Some wavelet families such as the Daub-4 wavelet with two vanishing moments and the Daub-6 wavelet with three vanishing moments have irrational wavelet coefficients. In a practical implementation, multiplications with approximations to the irrational coefficients lead to error accumulation in the final transform results, which may cause degradation of the image quality. Since algebraic integer quantization (AIQ) was first introduced by Cozzens and Finkelstein [19], it has been employed in the DWT to reduce computational errors. By mapping irrational wavelet coefficients into vectors or arrays of integers, the computations in wavelet decompositions can be performed without errors in the integer calculations.

In this chapter, we present an adaptive directional wavelet transform based on the lifting scheme that uses algebraic integers. The resulting architecture has less hardware complexity and lower latency than previous ADL designs. In addition, it provides better performance in terms of the PSNR and SSIM image quality metrics. The remainder of this chapter is organized as follows. Section 4.2 gives a brief overview of the necessary background for this work. Section 4.3 describes the proposed architecture. The performance results are presented in section 4.4, and the hardware implementation results are given in section 4.5. Finally, our conclusions are stated in section 4.6.

4. 2 Preliminaries

In this section, we briefly review the lifting-based DWT scheme, the adaptive lifting scheme and the use of algebraic integers.

4. 2. 1 Lifting Based DWT Scheme

A finite impulse response (FIR) filter can be represented using a lifting scheme [14]. The original lifting scheme was introduced by in Sweldens [15] in order to construct filter banks for the discrete wavelet transform. The lifting scheme decomposes the discrete wavelet filters into consecutive lifting steps. Let $H(z)$ and $G(z)$ be a pair of wavelet analysis filters, where $H(z)$ is a low-pass filter and $G(z)$ is a high-pass filter.

$$H(z)=\sum_{n=k_l}^{k_h} h_n z^{-n} \qquad G(z)=\sum_{n=k_l}^{k_h} g_n z^{-n} \tag{4.1}$$

By splitting the filter coefficients into even and odd index parts, the filters can be rewritten as:

$$H(z)=H_e(z^2)+z^{-1}H_o(z^2) \quad G(z)=G_e(z^2)+z^{-1}G_o(z^2) \tag{4.2}$$

and the corresponding polyphase matrix *P*(*z*) is defined as

$$P(z)=\begin{bmatrix} H_e(z) & H_o(z) \\ G_e(z) & G_o(z) \end{bmatrix} \tag{4.3}$$

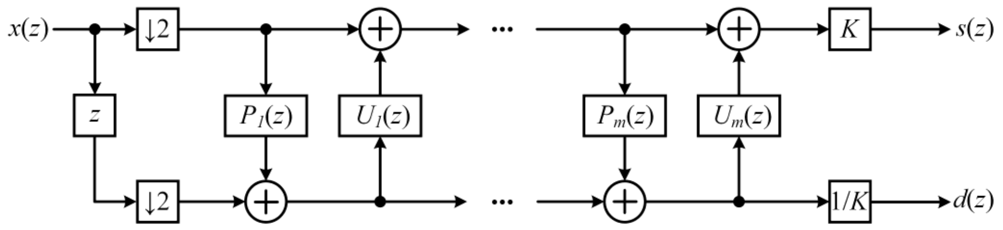
The polyphase matrix *P*(*z*) can be factorized into a sequence of upper and lower triangular matrices multiplied by a constant diagonal matrix using the Euclidean algorithm:

$$P(z)=\begin{bmatrix} k & 0 \\ 0 & 1/k \end{bmatrix} \prod_{i=1}^{m} \begin{bmatrix} 1 & s_i(z) \\ 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 \\ t_i(z) & 1 \end{bmatrix} \tag{4.4}$$

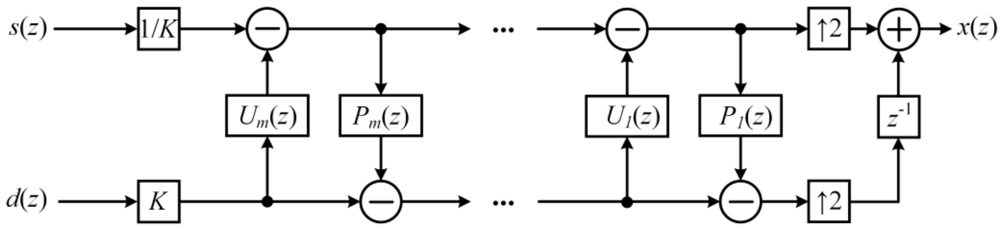
where *k* is the normalization factor, and $s_i(z)$ and $t_i(z)$ are the Laurent polynomials which correspond to the update and prediction operators of the lifting steps. The series of upper and lower triangular matrices can be obtained from the filter banks by factorization. Figure 4.1 represents the different steps of the forward and inverse wavelet transform employing the lifting scheme.

4. 2. 2 Adaptive Wavelet Transform via Lifting Scheme

Claypoole *et al.* [35] proposed an adaptive wavelet transform using the lifting scheme in which they focused on the spatial domain of the transform when designing the predictor.



(a) Forward LWT – analysis process.



(b) Inverse LWT – synthesis process.

Figure 4.1 Block diagram of the lifting wavelet transform.

By using a high-order predictor in the smooth region and a low-order predictor near the edges, the scheme can minimize errors in the presence of edges. The nonlinearity comes from the quantization which is needed to ensure an integer-to-integer transform. Due to the nature of the lifting implementation, perfect reconstruction is guaranteed in spite of the nonlinearity.

Gerek and Çetin presented a 2-D modification to prediction in the lifting scheme of the 5/3 wavelet by considering diagonal prediction [16]. A small value can be obtained in the filtered residual of the prediction when the difference between neighboring pixels is small. Four other nearest diagonal neighbors may be closer in value to the given pixel. Therefore, exploiting the four nearest diagonal pixels may be a better choice in the prediction filter. Perfect reconstruction is also assured in this scheme in the absence of quantization because of the symmetric property of the lifting implementation. However,

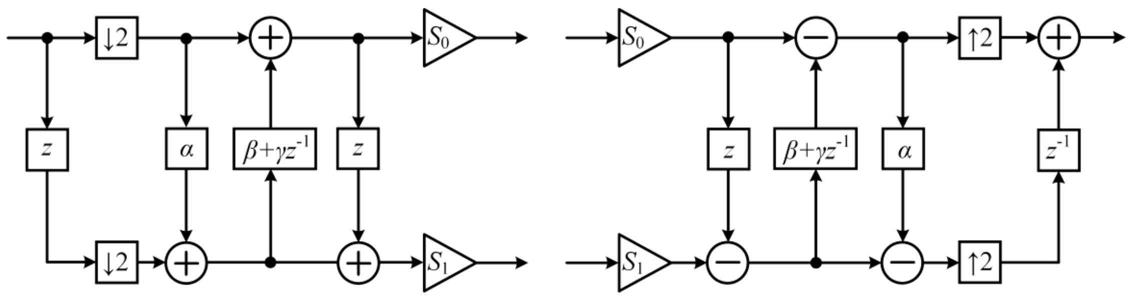


Figure 4.2 Block diagram of the lifting wavelet transform using algebraic integers.

it is necessary to keep track of the prediction choices if the transform coefficients are quantized. Thus, it remains to consider an efficient quantization approach in the lifting scheme in terms of having smaller errors in reconstruction at the synthesis step if the transform coefficients are quantized.

4. 2. 3 Lifting Wavelet Transform using Algebraic Integers

The algebraic integer quantization (AIQ) technique was first introduced by Cozzens *et al.* [10] in order to eliminate round-off errors in the DWT. Madishetty *et al*. [12] proposed an approach using wavelet based subband coding to accomplish error-free calculations from exact representations of Daub-4 and Daub-6 wavelet filter coefficients with algebraic integers. Balakrishnan *et al*. [37] developed a novel lifting scheme based DWT using the AIQ technique. The general block diagram of the lifting scheme using AIQ is illustrated in Figure 4.2. The advantage using the AIQ based DWT is that it requires fewer resources and performs better than traditional techniques in DWT implementations. Inexact representation of the irrational values of the filter coefficients causes error accumulation which degrades the quality of image reconstruction at the synthesis stage.

Mapping these irrational coefficients to appropriate algebraic integers and quantization in the subband coding algorithm can not only minimize the approximation error but can also achieve lower hardware cost while maintaining the image quality.

4. 3 Architecture

The proposed architecture is organized as a two-parallel structure using pipelining and reordering of the input data stream to achieve high throughput and reduced latency.

4. 3. 1 Two-Dimensional Wavelet Transform via Adaptive Directional Lifting

The main idea of the adaptive prediction algorithm is finding the minimum difference of directional gradients which are adjacent to the eight immediately neighboring pixels. A similar interpolation scheme is used in the prediction part of a lifting stage in the discrete wavelet transform. The lifting scheme in the DWT is carried out using single-line processing, and the polyphase transform matrix can be written as

$$\begin{aligned}\begin{bmatrix} s(z) \\ d(z) \end{bmatrix} &= \begin{bmatrix} k & 0 \\ 0 & 1/k \end{bmatrix} \begin{bmatrix} 1 & U(z) \\ 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 \\ -P(z) & 1 \end{bmatrix} \begin{bmatrix} x_e(z) \\ x_o(z) \end{bmatrix} \\ &= \begin{bmatrix} k & 0 \\ 0 & 1/k \end{bmatrix} \begin{bmatrix} 1-P(z)U(z) & U(z) \\ -P(z) & 1 \end{bmatrix} \begin{bmatrix} x_e(z) \\ x_o(z) \end{bmatrix}\end{aligned} \tag{4.5}$$

We begin with a simple example of a wavelet transform to illustrate the architecture. As its name indicates, the 5/3 wavelet [14] has five scaling filter coefficients for decomposition, which are h_0=[−1/8, 1/4, 3/4, 1/4, −1/8], and three scaling filter coefficients for reconstruction, which are h_1=[−1/2, 1, −1/2]. It is noted that the implementation is very efficient since the filter coefficients involve powers of 2 that are easily implemented using shifters. The polyphase matrix of the 5/3 wavelet is given by:

$$\begin{bmatrix} 1-\frac{1}{8}(1+z)(1+z^{-1}) & \frac{1}{4}(1+z^{-1}) \\ -\frac{1}{2}(1+z) & 1 \end{bmatrix} \tag{4.6}$$

In the lifting implementation, we need to consider the delay elements z_v^{-1} and z_h^{-1} which operate in the vertical and horizontal directions, respectively. This leads to a different factorization using the following polyphase matrices for 45º and 135º prediction:

$$\begin{bmatrix} 1-\frac{1}{8}(z_v^{-1}+z_v\cdot z_h)(1+z_h^{-1}) & \frac{1}{4}(1+z_h^{-1}) \\ -\frac{1}{2}(z_v^{-1}+z_v\cdot z_h) & 1 \end{bmatrix} \tag{4.7}$$

Polyphase matrix of 45º prediction

$$\begin{bmatrix} 1-\frac{1}{8}(z_v+z_v^{-1}\cdot z_h)(1+z_h^{-1}) & \frac{1}{4}(1+z_h^{-1}) \\ -\frac{1}{2}(z_v+z_v^{-1}\cdot z_h) & 1 \end{bmatrix} \tag{4.8}$$

Polyphase matrix of 135º prediction

The use of the ADL wavelet decomposition was presented in [16]. The probability of the horizontal case in the prediction of $x_0(m, 2n)$ is less than half of the sum of the other probabilities of $x_{45}(m, 2n)$ and $x_{135}(m, 2n)$. Therefore, using the alternative diagonal predictions $x_{45}(m, 2n)$, $x_{135}(m, 2n)$ rather than the horizontal one can lead to making a better decision in the adaptive prediction. An important aspect of ADL is determining the differences among these predictions. While the predictions may be influenced by quantization effects, degradation of the reconstructed wavelet image quality was not observed for a compression ratio below 16:1 in an 8 bpp original image. Hence, the ADL wavelet transform is suitable for moderate compression ratios and it is robust under quantization.

4. 3. 2 Proposed Architecture

A diagonal approximation signal $A_{45}(m, n)$ uses the two most recently computed differences between the detail signals $D_{45}(m+1, n-1)$ and $D_{45}(m-1, n+1)$. The very next approximation signal $A_{45}(m, n+2)$ is furnished by the next detail signals $D_{45}(m+1, n+1)$ and $D_{45}(m-1, n+3)$. The structural drawback of the architecture is that the sequence of detail signals $D_{45}(m+1, n-1)$ and $D_{45}(m+1, n+1)$ are not computed and loaded into registers simultaneously. The sample inputs needed in the detail signal $D_{45}(m+1, n-1)$ are

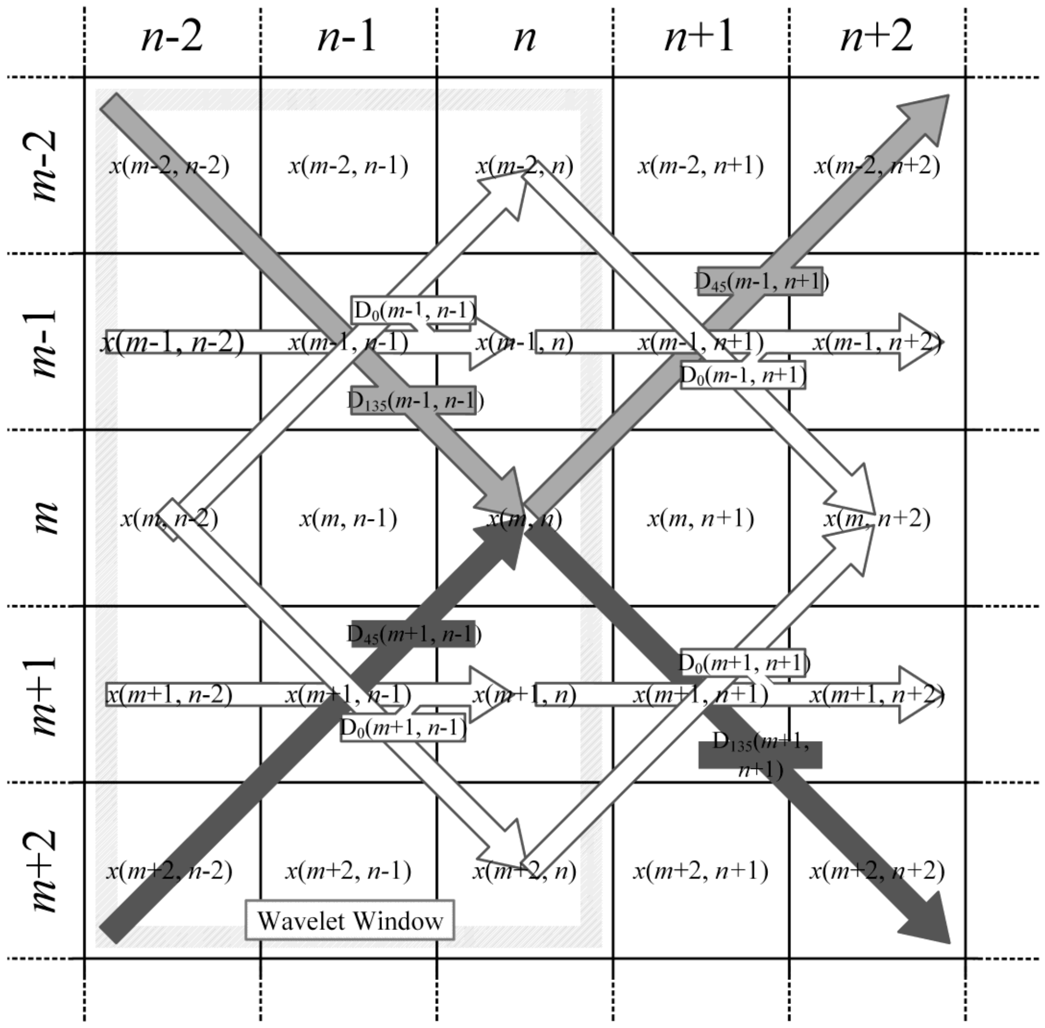
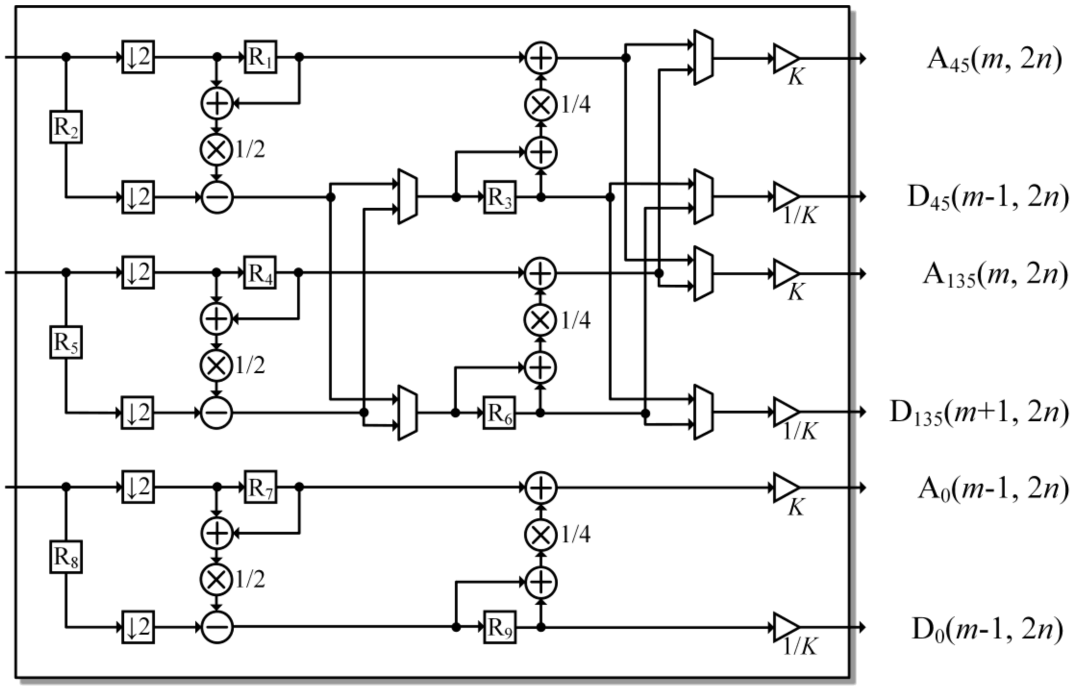


Figure 4.3 Image fragment of an adaptive directional segment of a 5/3 wavelet.

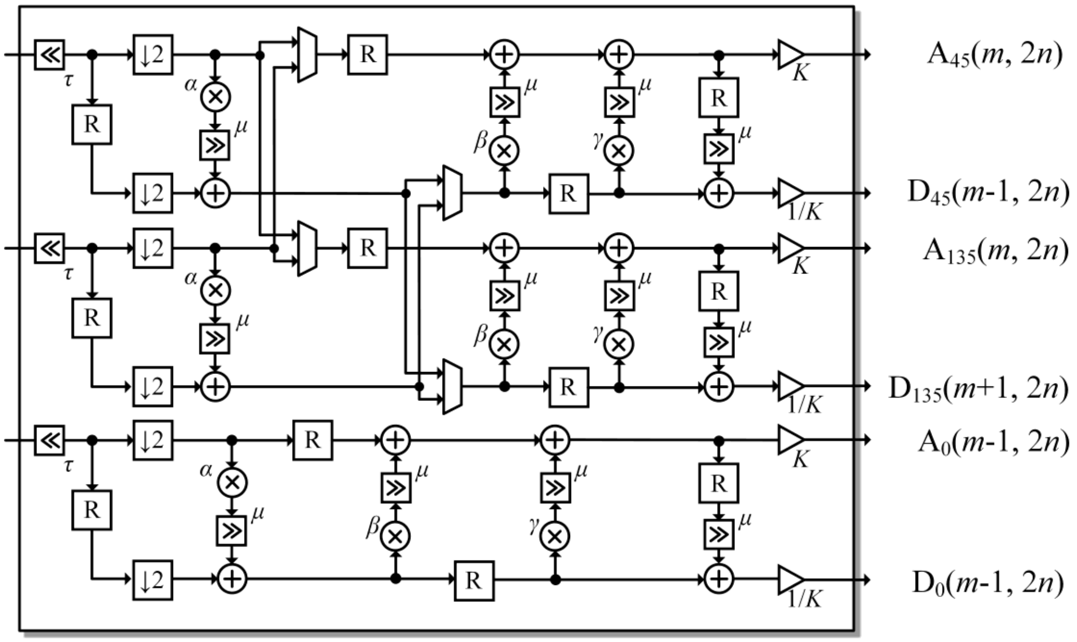
$x(m+2, n-2)$, $x(m+1, n-1)$ and $x(m, n)$. These input streams are not continuous with the sample inputs $x(m+2, n)$, $x(m+1, n+1)$ and $x(m+2, n+2)$ for $D_{45}(m+1, n+1)$. This is also the case for the directional approximation signal $A_{135}(m, n)$. However, we can manipulate the input stream to prevent discontinuities. A continuous stream can be realized by feeding the input streams in a zigzag pattern, as shown in Figure 4.3. The delay elements z_v^{-1} and z_h^{-1} are utilized to perform multiline processing. The other input stream, which is perpendicular to the first, is fed into the architecture in a zigzag order as well. In a similar fashion, the other lifting transform design is applied to a single Lifting Wavelet Transform (LWT) architecture organized as two-parallel processing. Also, the input steams are modified to be carried out as two even and odd processes. In addition, the horizontal lifting block is exploited to compute the prediction step along the angle of 0º. The architectures of the proposed adaptive directional lifting wavelet transform for the 5/3 and the Daub-4 wavelets are illustrated in Figure 4.4. The AIQ technique was employed in the architecture of ADL for the Daub-4 wavelet to minimize accumulating errors caused by three irrational coefficients in the lifting scheme. The other structural implementation is designed in the same manner as the ADL architecture for the 5/3 wavelet.

The block diagram of the proposed ADL architecture is shown in Figure 4.5. The architecture consists of two LWT blocks and three comparators. The comparators are used to select the minimum of the wavelet analysis signals. Each LWT block computes three detail signals of adaptive orientation, where the details are D_0, D_{45} and D_{135}. The total hardware needed to implement an LWT block for the 5/3 wavelet is 6 multipliers and 9 adders whereas for the Daub-4 wavelet it is 9 multipliers and 12 adders. Note that the computation of D_{90} is not needed because it will be taken account of by column operations in the decomposition step.

Table 4.1 shows the timing diagram of the proposed ADL architecture for the 5/3 wavelet without pipelining. Pipelining was included to increase the clock speed and the



(a)



(b)

Figure 4.4 Block diagram of a single LWT structure for the proposed ADL architecture. (a) Adaptive directional lifting architecture for the 5/3 wavelet, (b) Adaptive directional lifting architecture for the Daub-4 wavelet.

Table 4.1 Timing Diagram of the Proposed ADL Architecture for the 5/3 Wavelet.

CP	R_1	R_2	R_3	R_4	R_5	R_6	R_7	R_8	R_9	VALID
$N+0$	$x(m, n-2)$	$x(m-1, n-1)$	–	$x(m, n-2)$	$x(m-1, n-1)$	–	$x(m-1, n-2)$	$x(m-1, n-1)$	–	–
$N+1$	$x(m-2, n)$	$x(m-1, n+1)$	$D_{135}(m-1, n-1)$	$x(m-2, n)$	$x(m-1, n+1)$	$D_{45}(m-1, n-1)$	$x(m-1, n)$	$x(m-1, n+1)$	$D_0(m-1, n-1)$	–
$N+2$	$x(m, n+2)$	$x(m-1, n+3)$	$D_{45}(m-1, n+1)$	$x(m, n+2)$	$x(m-1, n+3)$	$D_{135}(m-1, n+1)$	$x(m-1, n+2)$	$x(m-1, n+3)$	$D_0(m-1, n+1)$	$A_0(m-1, n)$ $A_{45,135}(m, n)$
$N+3$	$x(m-2, n+4)$	$x(m-1, n+5)$	$D_{135}(m-1, n+3)$	$x(m-2, n+4)$	$x(m-1, n+5)$	$D_{45}(m-1, n+3)$	$x(m-1, n+4)$	$x(m-1, n+5)$	$D_0(m-1, n+3)$	$A_0(m-1, n+2)$ $A_{45,135}(m, n+2)$
$N+4$	$x(m, n+6)$	$x(m-1, n+7)$	$D_{45}(m-1, n+5)$	$x(m, n+6)$	$x(m-1, n+7)$	$D_{135}(m-1, n+5)$	$x(m-1, n+6)$	$x(m-1, n+7)$	$D_0(m-1, n+5)$	$A_0(m-1, n+4)$ $A_{45,135}(m, n+4)$
$N+5$	$x(m-2, n+8)$	$x(m-1, n+9)$	$D_{135}(m-1, n+7)$	$x(m-2, n+8)$	$x(m-1, n+9)$	$D_{45}(m-1, n+7)$	$x(m-1, n+8)$	$x(m-1, n+9)$	$D_0(m-1, n+7)$	$A_0(m-1, n+6)$ $A_{45,135}(m, n+6)$

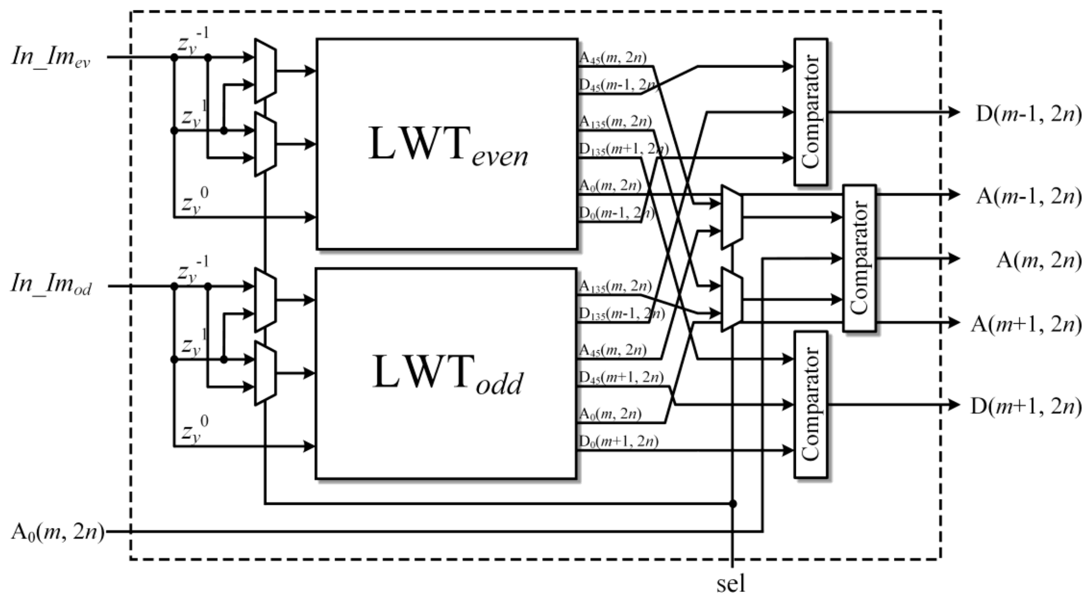


Figure 4.5 Block diagram of the proposed ADL architecture.

sample speed in the actual implementation. By inserting appropriate registers in the wavelet transform structure, the critical path delay can be reduced to $T_{\text{mult}}+T_{\text{add}}$, where T_{mult} is the time taken for multiplication and T_{add} is the time needed for addition. Thus, it

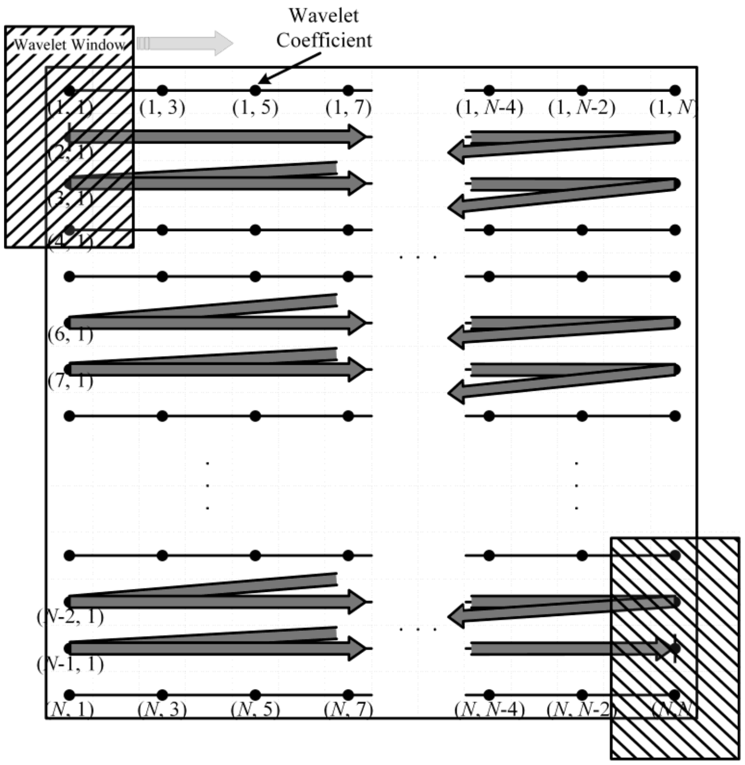


Figure 4.6 Traversal of the proposed ADL wavelet transform coefficients.

takes 2 clock cycles to form a valid approximation signal. The input stream is fed into the architecture at a rate of one 8-bit sample per clock cycle. The proposed ADL wavelet architecture operates in two-parallel form in order to reduce the latency and to avoid interruptions of the input stream.

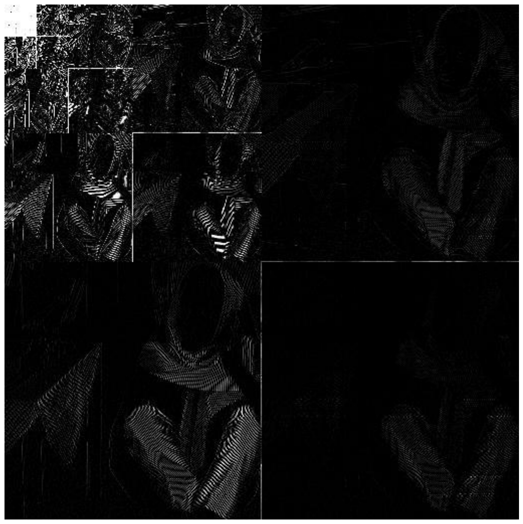
The traversal of the proposed ADL wavelet coefficients is shown in Figure 4.6. The shaded box indicates the region of the wavelet window. The window scans from left to right along the 1-D sequence of image pixels. The wavelet window computes detail signals along the horizontal direction by skipping over the adjacent pixel. The alternate diagonal prediction at angles of 45º and 135º can be computed in a single wavelet window as well. As noted, the architecture is processed in two-parallel fashion in order to reduce the latency. Thus, the LWT can be split into two parts as described above and these windows are overlapped in the processing. Note that the two windows compute mutually distinct signals from each other. The next scanning point will be four pixels below the previous starting point, which is the center of the window region. Accordingly, two wavelet windows are scanning the image and computing prediction signals simultaneously.

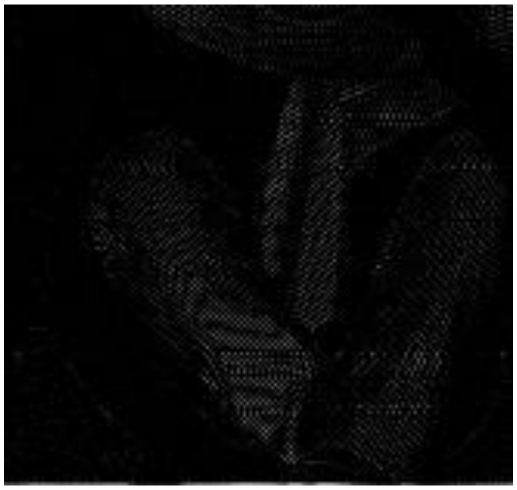
4. 4 Experimental Results

In order to evaluate the performance of the 2-D ADL wavelet, we will examine the coding performance for image compression. In particular, we have applied embedded coding based on the zero-tree method. The zero-tree method gives a compact description of the location of significant values during the embedded encoding and decoding processes. The image compression algorithms used are EZW (embedded zero-tree wavelet) and SPIHT (set partitioning in hierarchical tree), which are based on embedded bit-plane encoding [39]-[40]. Zero-tree based compression algorithms such as EZW and SPIHT use the statistical properties of the tree structure in order to obtain the locations of the significant coefficients.

Figure 4.7(c) and (d) show the portion of the vertical subband for the *Barbara* image at the level of the fourth original 5/3 wavelet decomposition and directionally adaptive 5/3 wavelet decomposition, respectively. The sub-image in Figure 4.7(c) from the

(a) Original


(a) 4-level decomposition


(c) Enlarged conventional wavelet
H = 1.8415 σ = 117.1135

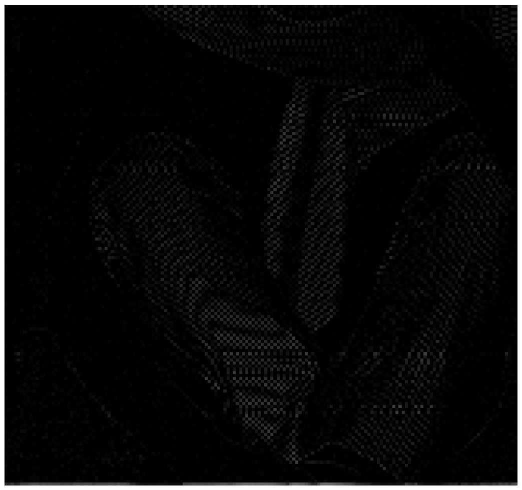

(d) Enlarged ADL wavelet
H = 1.7683 σ = 100.1475

Figure 4.7 Approximation sub-images of the wavelet decomposition.

conventional wavelet transform has an entropy H = 1.8415 and a variance σ = 117.1135, while that in Figure 4.7(d) from the ADL wavelet transform has an entropy H = 1.7683 and a variance σ = 100.1475. The directions of the conventional wavelet transform are limited to horizontal and vertical, while the adaptive directional wavelet transform has two additional directions at the diagonal angles. In the latter scheme, sharp edges laying side by side on a diagonal can have reductions in high-band signal energy. Thus, substantial energy reduction in the information of the decomposed coefficients that bit-plane needs to encode can bring better compression results, as Figure 4.7 demonstrates.

Wang *et al.* [41] proposed a heuristic weighting model as an image quality metric, which was combined with the structural similarity (SSIM) value that is based on properties of the human visual system. SSIM is computed with normalized signals $(x-\mu_x)/\sigma_x$ and $(y-\mu_y)/\sigma_y$, where μ_x and μ_y are the means of x and y, respectively, and σ_x and σ_y are the sample variances of x and y, respectively. Note that the correlation between $(x-\mu_x)/\sigma_x$ and $(y-\mu_y)/\sigma_y$ is equivalent to the correlation coefficient between x and y. SSIM can be expressed as

$$\mathrm{SSIM}(x,y)=\frac{(2\mu_x\mu_y+C_1)(2\sigma_{xy}+C_2)}{(\mu_x^2+\mu_y^2+C_1)(\sigma_x^2+\sigma_y^2+C_2)} \tag{4.9}$$

where σ_{xy} is the covariance of x and y. Local adaptation with the ADL wavelet transform can provide a better approximation by effectively reducing the correlation over dependencies found in discontinuities between adjacent pixels. This means that we can obtain higher covariance σ_{xy} in the numerator of the SSIM formula using the ADL wavelet transform.

Tables 4.2 and 4.3 show the comparison results for PSNR and SSIM using the 5/3 wavelet, where EZW and SPIHT were applied in the coding algorithm. We employed bit-plane encoding of the transform domain coefficients and we used nine standard benchmark images, all of size 512×512 with 8-bit grayscale pixel values [42]. A decomposition level of 4 was selected. Single iterative lossy compression in EZW and SPIHT was used to allow a precise comparison between the adaptive wavelet and non-adaptive wavelet methods. The images having many sharp edges show slightly better PSNR and SSIM values than the others. This is because ADL can consider diagonal pixels along a diagonal direction as well as adjacent pixels along the horizontal and vertical directions. Situations having a greater chance of choosing a diagonal interpolation lead to higher PSNR and SSIM values. The images with contrast stretches have a higher value of SSIM compared to PSNR, which is computed using the mean squared error. Also, the benchmark images having high edge contrast and a limited range of gray scale values have higher SSIM values, such as for *House* and *Mandrill*.

Tables 4.4 and 4.5 show the comparison results for PSNR and SSIM using the Daub-4 wavelet. There are irrational coefficients in the ADL wavelet using Daub-4. For comparison purposes, EZW and SPIHT were again used as the compression algorithms. Table 4.6 shows that the proposed wavelet transform based on ADL with algebraic integers has better coding performance for the standard benchmark images, with an improvement of up to 3.98 dB in PSNR (*Peppers* at 4.0 bpp) compared to the conventional lifting wavelet using algebraic integers.

Table 4.2 PSNR/SSIM Performance for EZW (5/3 Wavelet).

Image	**BPP (Bits per Pixel)**				
	0.25	**0.5**	**1.0**	**2.0**	**4.0**
Lena	28.51 0.4264	31.85 0.5667	33.77 0.6791	34.78 0.8258	35.38 0.9498
Cameraman	28.21 0.2855	32.49 0.4771	34.18 0.6547	34.62 0.8360	34.71 0.9284
Mandrill	22.95 0.4438	25.88 0.6776	28.61 0.8482	29.78 0.9366	30.00 0.9556
Barbara	23.51 0.4252	26.29 0.5930	28.50 0.7175	29.76 0.8285	30.09 0.9392
Peppers	28.93 0.3835	31.53 0.4872	32.89 0.5906	34.61 0.7834	36.17 0.9382
House	32.75 0.4762	39.06 0.6446	41.65 0.8466	42.48 0.9354	42.62 0.9697
Stream and Bridge	22.47 0.3909	24.46 0.5844	26.80 0.7528	29.33 0.8867	30.66 0.9551
Boat	25.22 0.3655	28.92 0.5433	30.98 0.6772	32.27 0.8284	33.11 0.9461
Woman	27.26 0.3974	29.65 0.5311	31.55 0.6505	32.93 0.8100	33.66 0.9402

Table 4.3 PSNR/SSIM Performance for SPIIHT (5/3 Wavelet).

Image	**BPP (Bits per Pixel)**				
	0.25	**0.5**	**1.0**	**2.0**	**4.0**
Lena	26.31 0.4126	28.95 0.4955	32.51 0.6148	32.20 0.7820	34.46 0.9066
Cameraman	25.83 0.2755	30.38 0.3981	31.23 0.5862	32.59 0.7911	32.83 0.8874
Mandrill	21.42 0.4297	23.98 0.5993	25.98 0.7819	28.27 0.8899	27.81 0.9163
Barbara	21.91 0.4134	24.58 0.5159	25.89 0.6485	28.31 0.7803	27.76 0.9030
Peppers	26.90 0.3696	29.32 0.4128	29.80 0.5260	31.58 0.7412	33.87 0.8941
House	30.99 0.4658	36.53 0.5721	38.44 0.7830	38.65 0.8906	40.06 0.9282
Stream and Bridge	20.58 0.3781	22.75 0.5104	24.80 0.6886	27.50 0.8405	28.48 0.9125
Boat	23.17 0.3488	27.15 0.4661	28.54 0.6130	29.29 0.7791	30.47 0.9088
Woman	24.77 0.3896	27.54 0.4603	29.69 0.5887	30.59 0.7658	30.64 0.9012

Table 4.4 PSNR/SSIM Performance for EZW (Daub-4 Wavelet).

Image	**BPP (Bits per Pixel)**				
	0.25	**0.5**	**1.0**	**2.0**	**4.0**
Lena	24.17 0.3510	26.35 0.4549	28.13 0.5661	29.78 0.7397	31.36 0.8914
Cameraman	23.23 0.2245	25.48 0.3545	27.13 0.5152	28.64 0.7042	30.11 0.8363
Mandrill	21.09 0.3130	23.34 0.4831	25.56 0.6447	27.49 0.7770	29.07 0.8676
Barbara	21.61 0.3403	24.34 0.5026	26.73 0.6357	28.59 0.7521	30.16 0.9028
Peppers	24.57 0.3311	26.73 0.4084	28.54 0.5084	30.36 0.7268	32.08 0.9177
House	26.13 0.3308	28.62 0.4890	30.43 0.6726	32.11 0.7959	33.75 0.9021
Stream and Bridge	20.88 0.3122	23.07 0.4639	25.37 0.6306	27.56 0.7910	29.31 0.8898
Boat	22.74 0.2942	25.17 0.4420	27.07 0.5694	28.78 0.7540	30.39 0.9000
Woman	24.14 0.3288	26.46 0.4431	28.48 0.5603	30.30 0.7191	31.99 0.8979

Table 4.5 PSNR/SSIM Performance for SPIIHT (Daub-4 Wavelet).

Image	**BPP (Bits per Pixel)**				
	0.25	**0.5**	**1.0**	**2.0**	**4.0**
Lena	21.66 0.3375	24.38 0.3749	26.52 0.4905	27.54 0.6865	28.70 0.8406
Cameraman	21.70 0.2132	23.80 0.2783	25.19 0.4464	26.90 0.6525	27.52 0.7859
Mandrill	19.79 0.3038	21.01 0.4118	23.47 0.5690	25.19 0.7217	26.25 0.8120
Barbara	20.35 0.3252	22.61 0.4315	25.07 0.5627	25.77 0.6940	27.41 0.8503
Peppers	23.22 0.3181	24.64 0.3352	26.33 0.4345	27.87 0.6735	28.88 0.8680
House	24.07 0.3144	25.88 0.4094	28.10 0.6024	29.25 0.7417	30.96 0.8500
Stream and Bridge	18.99 0.2956	21.75 0.3838	23.59 0.5585	25.24 0.7367	26.63 0.8353
Boat	20.49 0.2823	23.29 0.3620	24.75 0.5029	26.98 0.6977	27.68 0.8468
Woman	22.24 0.3195	24.29 0.3687	25.85 0.4922	27.75 0.6649	29.34 0.8479

Table 4.6 Comparison of PSNR Performance (Daub-4 Wavelet).

Image	**BPP (Bits per Pixel)**									
	0.25		**0.5**		**1.0**		**2.0**		**4.0**	
	†L.S.	**‡ADL**	**†L.S.**	**‡ADL**	**†L.S.**	**‡ADL**	**†L.S.**	**‡ADL**	**†L.S.**	**‡ADL**
Barbara	20.53	21.61	22.64	24.34	23.97	26.73	25.74	28.59	29.01	30.16
Lena	23.24	24.17	24.92	26.35	26.69	28.13	28.54	29.78	30.54	31.36
Mandrill	21.09	21.09	21.99	23.34	23.36	25.56	25.44	27.49	28.36	29.07
Peppers	23.98	24.57	24.28	26.73	25.19	28.54	27.05	30.36	28.11	32.08
Cameraman	22.48	23.23	23.94	25.48	25.62	27.13	27.76	28.64	29.15	30.11

† Lifting Wavelet using AIQ ‡ Adaptive Directional Lifting Wavelet using AIQ
Compression Algorithm: EZW, Single Iteration

4. 5 Hardware Implementation Results

Logic synthesis was performed using a 45-nm CMOS standard cell library that is optimized for a 1.1V supply voltage. Table 4.7 shows the implementation results of the proposed ADL wavelet architecture along with several performance comparisons. The proposed ADL architecture for the 5/3 wavelet operates at an approximate clock frequency of 250 MHz, has an approximate 4 Gb/s throughput and requires about 18% fewer equivalent gates than previous architectures. The reason that the gate count is reduced is because the architecture for the 5/3 wavelet is entirely multiplier-free due to their replacements by shift operations. In addition, the proposed ADL architecture for the Daub-4 wavelet has the same processing rate and throughput as for the 5/3 wavelet. By taking advantage of the lifting scheme, the gate count (in terms of 2-input NAND gate equivalents) used in the proposed architecture for the Daub-4 wavelet is 18.7K gates,

Table 4.7 Implementation and Performance Comparisons of 2D-DWT Architectures

	ANDRA ***et al.*** [19]	**ANGELOPOULOU** ***et al.*** [20]	**DIA** ***et al.*** [21]	**LIU** ***et al.*** [22]	**SEO** ***et. al.*** [23]	**YAMAUCHI** ***et. al.*** [24]	**BALAKRISHNAN** ***et al.*** [13]	**MADISHETTY** ***et. al.*** [12], [25]		**PROPOSED**	
Wavelet	9/7, 5/3 L.S.	5/3 L.S.	5/3 L.S.	9/7, 5/3 L.S.	9/7, 5/3 L.S.	9/7, 5/3	Daub-4 L.S.	Daub-4		5/3 L.S.	Daub-4 L.S.
Technology	CMOS 0.18um	Xilinx XC4VLX15	Xilinx XCV600E	UMC 0.18 μm	Hynix 0.35 μm	CMOS 0.25 μm	Xilinx XCV300E	Xilinx XC6VC X240T	CMOS 45 nm	CMOS 45 nm	
Image Size	512 × 512	512 × 512	256 × 256	512 × 512	512 × 512	720 × 480	512 × 512	512 × 512		512 × 512	
Input Precision	8 bits	8 bits	8 bits	8 bits	8 bits	8 bits	8 bits	8 bits		16 bits	
Gate count / (Slices)	†15.6k	(293)	(1835)	†28k	‡89k	N/A	(282)	(426)	§360k	†12.7k	†18.7k
Throughput	1.6 Gb/s	123 fps	187 fps	23.29 fps	1.2 Gb/s	30 fps	N/A	2.3 Gb/s	4.2 Gb/s	4 Gb/s	
Max. Freq. (MHz)	200	172.4	108	100	150	27.4	N/A	282.5	523.6	250	
AI Encoding	No	YES	NO	NO	NO	NO	YES	YES		YES	YES
Decomposition Level	5	4	3	5	N/A	5	3	4		4	4

†2-input NAND gate equivalents ‡ASIC logic gate count §ASIC Gate Count (AGC) generated by Cadence Encounter® RTL Compiler

which is considerably reduced compared to the previous architecture implemented in the same technology [49].

4. 6 Conclusion

We have proposed an efficient architecture for the 2-D wavelet transform using the adaptive directional lifting (ADL) scheme with algebraic integers. The proposed architecture has low complexity and efficient hardware utilization without additional latency by applying parallel processing and by reordering the input data stream. To avoid accumulating errors from irrational coefficients in the lifting scheme, the proposed architecture uses algebraic integers with appropriate input scaling parameters. This technique leads to a reduced hardware complexity due to the elimination of multipliers. The results show that the proposed wavelet transform based on ADL using AIQ has better coding performance for several standard benchmark images with an improvement of up to 3.98 dB in PSNR compared to previous results. The proposed architecture has been designed and synthesized in a 45-nm CMOS standard cell technology. The synthesis results show that the proposed architecture for the Daub-4 wavelet can operate at a clock frequency of approximately 250 MHz and has a favorable complexity compared to previous designs.

Chapter. 5

Gradient-based Adaptive Curved Wavelet Transform and a New Structural Similarity Image Quality Metric

5. 1 Introduction

Over the last two decades the Discrete Wavelet Transform (DWT) has become one of the most widely used tools in the area of signal, image and video processing. It is used in the JPEG-2000 and MPEG-4 standards, as well as in other applications such as image compression, denoising and watermarking. Moreover, the lifting-based wavelet scheme [15] can reduce the computational complexity of the hardware implementation.

One of the most powerful applications of the wavelet transform is for entropy coding in image compression. Two of its important advantages are excellence in energy compaction and suitability for image processing based on the Human Visual System (HVS). The former advantage leads to image compression, which features the multiresolution capability of energy compaction with reconstructed images at high compression ratios. In transform-based image compression, entropy coding reduces the redundancy in the bit stream and enables an image to be recovered with acceptable quantization errors. Algorithms such as the Embedded Zerotree Wavelet (EZW) [39] and Set Partitioning In Hierarchical Trees (SPIHT) [40] achieve excellent performance by exploiting the statistical properties of the tree structure. However, transformed data from an image having a large number of curves and obliques can spread the energy into high-frequency subbands. As a result, ringing artifacts can occur around the edges at low coding bit rates. To overcome this issue, a variety of two-dimensional Adaptive Directional Lifting-based (ADL) wavelet decompositions have been proposed. A modification of the lifting scheme for the 5/3 wavelet using diagonal prediction has been developed [16] and another scheme which incorporates directional prediction into the conventional lifting scheme was proposed [17]. In addition, Wang *et al*. [50] proposed the Curved Wavelet Transform (CWT) and its associated image coder.

The quality of compression results for the wavelet transform are evaluated using various criteria, such as Peak Signal-to-Noise Ratio (PSNR) and Mean Squared Error (MSE). While these traditional metrics are still widely used, they are often inconsistent with subjective human perception. To overcome this inconsistency, structural similarity

(SSIM) was proposed as an Image Quality Assessment (IQA) metric [41]. Rather than being based on a distance between two images, SSIM measures the correlation between two images by comparing three components in the spatial domain, i.e., luminance, contrast and structure, and it has been shown to have good consistency with the HVS. Many variations and modifications of SSIM have been proposed in order to improve the assessment accuracy. A multiscale SSIM (MS-SSIM) approach was proposed that includes variations in image resolution and viewing conditions [51], and an information content weighted SSIM (IW-SSIM) was defined in [52]. Also, using a Sobel mask for edge detection in [53], researchers have proposed a gradient similarity measure to improve the quality assessment [54]–[58].

The contributions of this chapter are two-fold. First, efficient algorithms and VLSI architectures for two novel adaptive wavelet transforms are proposed. Second, we suggest a new IQA metric inspired by these transforms and demonstrate its effectiveness by numerical experiments. The remainder of this chapter is organized as follows. Section 5.2 gives a brief overview of the necessary background for this work. Section 5.3 describes the proposed algorithms. The experimental results are presented in section 5.4, and the hardware implementation results are given in section 5.5. Finally, our conclusions are stated in section 5.6.

5. 2 Gradient-based Adaptive Curved Wavelet Transform (GACWT)

The LeGall 5/3 wavelet, also called the bi-orthogonal CDF 5/3 wavelet, has a wide range of applications due to its rational filter coefficients, which make it more convenient for real-time hardware implementation. We can improve its coding performance using two new proposed algorithms, as described in subsections 5.2.2 and 5.2.3 below.

5. 2. 1 Curved Wavelet Transform (CWT)

The orientations of edges and lines in an image are taken into account in several image coding methods. The Curved Wavelet Transform (CWT) [50] performs wavelet filtering along curves and exploits geometric features that are parallel to edges and lines in an image to be compressed. This provides energy compactness in image coding, and its stronger coding ability has been demonstrated. In the CWT, the allowed orientations for vertical curves in an image are selected from the five candidates $\pm\pi/4$, $\pm\pi/8$ and 0, as shown in Figure 5.1 [50]. The CWT was used to construct a new image coder named EBCOT, which is compatible with the JPEG2000 code-steam syntax [50][59]. Similarly, the wavelet transforms in [34][60] and the directionlets in [19] also exploit directional features in the image.

To optimize the coding bit rate, determination of the best curve is necessary while maintaining a low computational cost. Rate-distortion optimization can also provide good

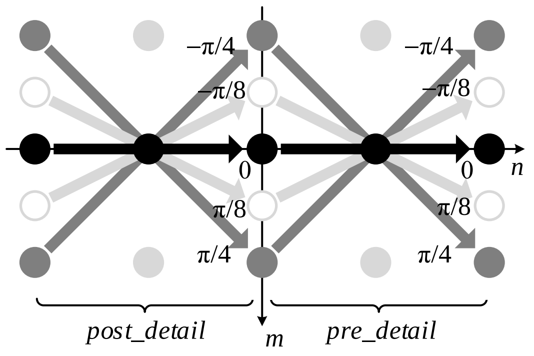


Figure 5.1 Allowed orientations of horizontal detail signals for adaptive curves.

performance but requires considerable computational cost [41]. In the remainder of this section, we develop two new approaches that address these issues.

5. 2. 2 Proposed Adaptive Curved Wavelet Transform (ACWT)

In [50], the authors considered a special case of the CWT in which two or more curves intersect. In that case, the selection of the curve among the candidates is based on the positions of adjacent pixels. Figure 5.2 shows pairs of high-pass coefficients associated with a single low-pass coefficient. The vertical distance between the *pre_detail* and *post_detail* signals can range from 0 to 4, assuming a unit distance between pixels. In the figure, details signals are illustrated by using dark gray and light gray arrows. At each pixel, there are five possible choices of orientation for the detail signals, namely –π/4, –π/8, 0, π/8 and π/4. Therefore, we need to consider a total of twenty-five cases for each adjacent pair of detail signals.

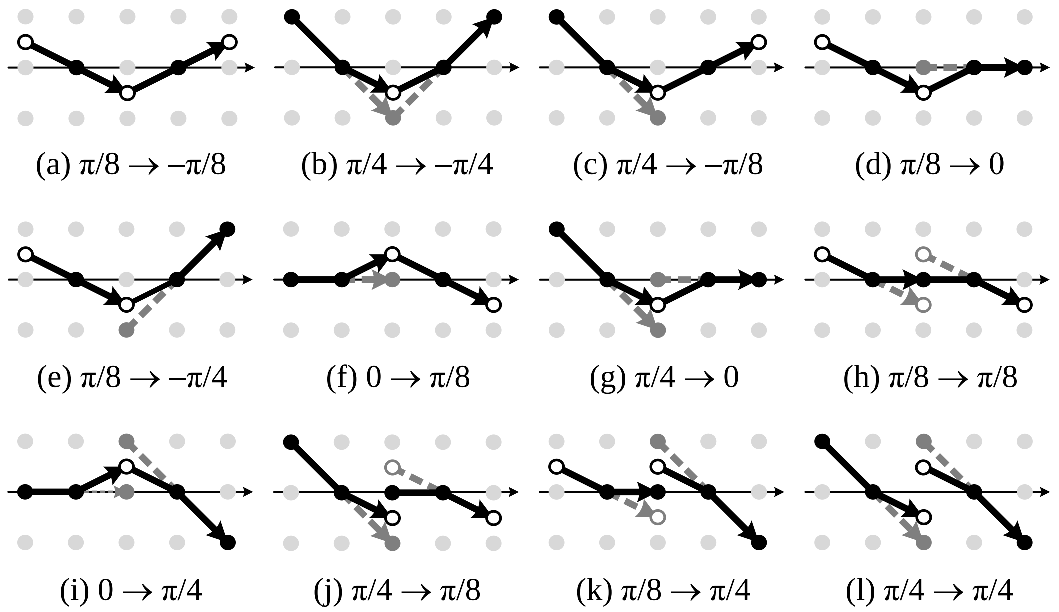


Figure 5.2 Proposed patterns for the curves of the ACWT.

Two adjacent detail signals having the same absolute value but opposite signs, or which are both zero, give a continuous approximation. On the other hand, two detail signals having different angles would be discontinuous, which could result in ringing artifacts in the compressed image. These artifacts become more pronounced at higher compression rates. Our approach minimizes these artifacts by enhancing the directional continuity of the detail signals. We performed a set of exhaustive simulations, considering all possible combinations of adjacent detail signals and their results on the image compression. The results of these simulations lead to the patterns shown in Figure 5.2. These specific patters were found to give the best image quality results when evaluated over a set of 9 representative images. Note that the patterns in the figure are taken to be mirror-image symmetric with respect to the horizontal axis at $m = 0$. Thus, for example, the case $-\pi/4 \rightarrow \pi/4$ can be inferred from case (b). Figure 5.3 illustrates portions of the vertical curves that are determined by the standard CWT and by the proposed ACWT algorithm for the *Barbara* image. The pseudocode for this algorithm is as follows:

Algorithm 1 Adaptive Curved Wavelet Transform (ACWT)

```
     Input:  in_sig
     Output: appr_sig, det_sig
 1:  pre_det ← 0
 2:  while(in_sig != EOF)
 3:     det_sig ← high_pass_filtering(in_sig)
 4:     post_det ← min(det_sig)
 5:     switch(pre_det, post_det)
 6:         case (d), (e):
 7:             update(pre_det)
 8:         case (c), (f):
 9:             update(post_det)
10:         case (b), (g), (h), (i), (j), (k), (l)
11:             update(pre_det, post_det)
12:         otherwise
13:             no_update(pre_det, post_det)
14:     endswitch
15:     appr_sig ← low_pass_filtering(pre_det, post_det)
16:     det_sig ← post_det
17:     pre_det ← post_det
18:  endwhile
```

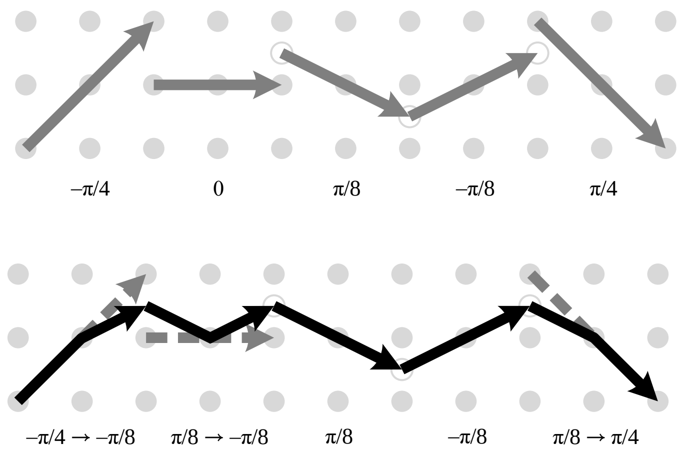


Figure 5.3 Detail signals from the high-pass filter. (a) Detail signals for ADWT, (b) Adaptive curves for ACWT.

5. 2. 3 Proposed Gradient-based ACWT (GACWT)

Opportunities for further performance improvements remain. First, salt-and-pepper noise on the curves, which is manifested as sparsely occurring white and black pixels, may be present. Second, it is possible that lines having very high or low pixel values may cross the curves. These features can cause the curves to go astray.

To address these issues, we propose a slightly modified approach. Specifically, we apply edge detection filters in the gradient domain to avoid those cases that lead to energy spreading near edges. This enhancement leads to greater continuity in the adaptive curves and allows for better coding performance. We allow the use of either Sobel or Prewitt filters to provide additional flexibility. Also, threshold scaling factors of 0.25 and 0.75 are taken from [62]–[63] but they have been quantized for efficiency in a hardware implementation. The resulting modifications to Algorithm 1 are underlined in the following gradient-based ACWT (GACWT) algorithm:

Algorithm 2 Gradient-based ACWT (GACWT)

```
    Input: in_sig
    Output: appr_sig, det_sig
 1: pre_det ← 0
 2: while(in_sig != EOF)
 3:   det_sig ← high_pass_filtering(in_sig)
 4:   post_det ← min(det_sig)
 5:   switch(pre_det, post_det)
 6:     case (d), (e):
 7:       update(pre_det)
 8:     case (c), (f):
 9:       update(post_det)
10:     case (b), (g), (h), (i), (j), (k), (l)
11:       update(pre_det, post_det)
12:     otherwise
13:       no_update(pre_det, post_det)
14:   endswitch
15:   p_post_det ← prewitt(post_det)
16:   s_post_det ← sobel(post_det)
17:   post_det ← grad_sig_update(pre_det, post_det, p_post_det, s_post_det)
18:   appr_sig ← low_pass_filtering(pre_det, post_det)
19:   det_sig ← post_det
20:   pre_det ← post_det
21: endwhile
```

```
function grad_sig_update($A$, $B$, $C$, $D$)
    if !($A \times 0.25 \le C \le A \times 0.75$)
        $C \leftarrow B$
    end
    if !($A \times 0.25 \le D \le A \times 0.75$)
        $D \leftarrow B$
    end
    if $((|B| \le |A|) \le (|C| \le |A|))$ & $((|B| \le |A|) \le (|D| \le |A|))$
        $grad\_sig \leftarrow B$
    else
        if $((|C| \le |A|) \le (|B| \le |A|))$ & $((|C| \le |A|) \le (|D| \le |A|))$
            $grad\_sig \leftarrow C$
        else
            $grad\_sig \leftarrow D$
        end
    end
        return $grad\_sig$
endfunction
```

5. 3 GACWT-SSIM

5. 3. 1 SSIM

Wang *et al*. proposed a heuristic weighting model as an image quality metric, using the SSIM value that is based on properties of the HVS [41]. SSIM is defined as:

$$\mathrm{SSIM}(r,\ d) = [l(r,d)]^{\alpha} \cdot [c(r,d)]^{\beta} \cdot [s(r,d)]^{\gamma} \tag{5.1}$$

$$= \frac{(2\mu_r\mu_d + C_1)(2\sigma_{rd} + C_2)}{(\mu_r^2 + \mu_d^2 + C_1)(\sigma_r^2 + \sigma_d^2 + C_2)} \tag{5.2}$$

where *r* and *d* denote the reference and distorted images, and σ_{rd} is the covariance of the two images. SSIM is computed with normalized signals $(r-\mu_r)/\sigma_r$ and $(d-\mu_d)/\sigma_d$, where μ_r and μ_d are the means of *r* and *d*, respectively, and σ_r and σ_d are the sample variances of *r* and *d*, respectively. Note that the correlation between $(r-\mu_r)/\sigma_r$ and $(d-\mu_d)/\sigma_d$ is equivalent to the correlation coefficient between *r* and *d*.

5. 3. 2 SSIM employing Gradient-based Methods

It was reported that SSIM is less capable of predicting quality for images having white noise or those which are badly blurred [57]. To address these limitations, the gradient-based SSIM (GSSIM) was proposed in [53] by considering contrast and structure in the gradient map. This approach relies on characteristics of the HVS, which is very sensitive to edge regions.

Several other modifications have been introduced to further improve the performance. In [54], the gradient information matrix and its eigenvalues were utilized to reflect the information about the geometric structure of an image. The Mean Geometric Structural Distortion (MGSD) was proposed which uses interpolation of the edge direction [55]. In [56], the Gradient Similarity (GS) scheme is proposed that captures the features of images by measuring contrast–structural changes with the gradient similarity. A metric which calculates the structural similarity based on an Edge Detection Histogram (EDH) was proposed in [58] and demonstrated that the distribution of edge orientations was well matched with the structural information.

The edge features in the gradient domain do not change the structural content in the decomposed image. Hence, it is important to evaluate the effectiveness of the features on the edge/gradient similarity with a more elaborate IQA metric.

5. 3. 3 GACWT-SSIM

CW-SSIM has been shown to be a useful measure for images in the complex wavelet transform domain [64]. The metric value in DWT-SSIM is calculated directly from each wavelet decomposition band [65]. The idea of the proposed wavelet-based SSIM is to minimize the effect of noise factors which can distort the image. The proposed GACWT-SSIM is analogous to previous wavelet-based SSIM metrics by using wavelet decomposition bands. However, it is distinct from them by not only applying the proposed adaptive wavelet decomposition but also by observing the subbands at each level. The proposed IQA metric can be evaluated by means of the GACWT-SSIM at each decomposition level, and is defined as:

$$\text{GACWT-SSIM}(r,\, d) = [l(r,d)]^{\alpha} \cdot [c_g(r,d)]^{\beta} \cdot [s_g(r,d)]^{\gamma} \tag{5.3}$$

$$= \frac{\sum_{i=1}^{M} \text{GACWT-SSIM}_i(r,d)}{M} \tag{5.4}$$

5. 4 Experimental Result

5. 4. 1 ACWT in Image Coding

In this subsection, we examine the coding performance of the proposed ACWT. The image compression algorithms in this experiment are EZW and SPIHT, which are based on embedded bit-plane encoding [39]–[40]. Zerotree based compression algorithms such as EZW and SPIHT use the statistical properties of the tree structure in order to obtain the locations of the significant coefficients. Reducing discontinuities in the signals gives improvements in image quality.

We employed bit-plane encoding in the transform domain coefficients and used nine standard benchmark images, all of size 512×512 with 8-bit grayscale pixel values [42]. A decomposition level of 4 was selected and single lossy compression was employed. The resulting PSNR and SSIM values are shown for EZW in Table 5.1 and for SPIHT in Table 5.2.

5. 4. 2 GACWT-SSIM for Image Quality Assessment

Simulations were conducted to verify the validity of the proposed GACWT-SSIM and to compare its performance with existing metrics. The evaluation was conducted based on the database developed by the Laboratory for Image & Video Engineering (LIVE) [66]. The full database contains 982 images, with 203 reference images and 779 distorted images. We used a logistic function, which is recommended in [67], in order to perform

Table 5.1 PSNR/SSIM Performance (EZW)

Image	**BPP (Bits per Pixel)**				
	0.25	**0.5**	**1.0**	**2.0**	**4.0**
Lena	29.63	32.85	35.02	36.25	36.75
	0.4366	0.5927	0.7023	0.8559	0.9911
Cameraman	29.33	33.49	35.43	36.09	36.08
	0.3586	0.4988	0.6780	0.8682	0.9699
Mandrill	24.07	26.88	29.86	31.26	31.37
	0.4570	0.6976	0.8780	0.9710	0.9929
Barbara	24.63	27.29	29.75	31.24	31.46
	0.4386	0.6124	0.7444	0.8641	0.9784
Peppers	30.05	32.53	34.14	36.08	37.54
	0.3955	0.5099	0.6201	0.8135	0.9745
House	33.87	40.06	42.90	43.96	44.00
	0.4926	0.6648	0.8707	0.9640	0.9971
Stream and Bridge	23.59	25.46	28.05	30.81	32.03
	0.4065	0.6029	0.7778	0.9166	0.9909
Boat	26.34	29.92	32.23	33.75	34.48
	0.3760	0.5686	0.7000	0.8594	0.9875
Woman	28.38	30.65	32.80	34.41	35.04
	0.4126	0.5563	0.6792	0.8458	0.9809

Table 5.2 PSNR/SSIM Performance (SPIHT)

Image	**BPP (Bits per Pixel)**				
	0.25	**0.5**	**1.0**	**2.0**	**4.0**
Lena	27.97	30.46	35.01	35.32	37.47
	0.4221	0.5201	0.6424	0.8149	0.9505
Cameraman	27.49	31.89	33.73	35.72	35.83
	0.3850	0.4226	0.6111	0.8223	0.9316
Mandrill	23.08	25.49	28.48	31.40	30.81
	0.4374	0.6247	0.8084	0.9235	0.9586
Barbara	23.57	26.09	28.39	31.44	30.76
	0.4215	0.5419	0.6739	0.8113	0.9469
Peppers	28.56	30.83	32.30	34.71	36.88
	0.3792	0.4391	0.5519	0.7724	0.9361
House	32.65	38.04	40.94	41.78	43.06
	0.4741	0.5985	0.8091	0.9227	0.9716
Stream and Bridge	22.24	24.26	27.30	30.63	31.49
	0.3887	0.5348	0.7155	0.8736	0.9574
Boat	24.83	28.66	31.04	32.42	33.48
	0.3570	0.4908	0.6374	0.8113	0.9506
Woman	26.43	29.05	32.19	33.71	33.65
	0.3970	0.4844	0.6163	0.7976	0.9446

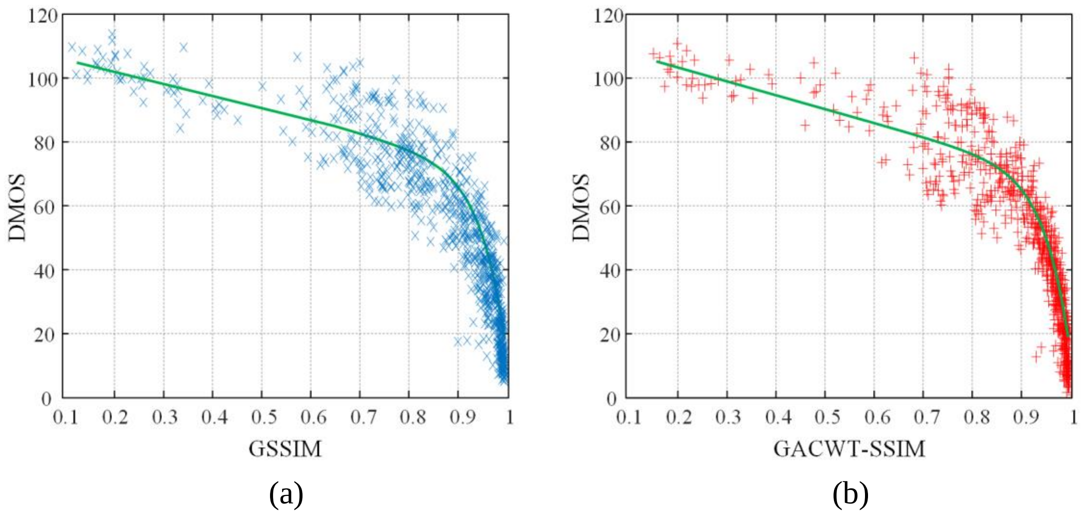


Figure 5.4 Performance comparison of IQA models on LIVE dataset. (a) GSSIM, (b) GACWT.

nonlinear regression analyses between the objective and subjective scores. This function is defined as:

$$f(x) = \frac{a_1 - a_2}{1 + \exp\left[-(x - a_3)/a_4\right]} + a_2 \tag{5.5}$$

where x is the score obtained from the objective IQA metric. The parameters a_1 through a_4 are found numerically using a nonlinear regression process to minimize the least square error between the subjective Difference Mean Opinion Score (DMOS) and the objective score $f(x)$. Figure 5.4 shows the scatter plots for GSSIM and for the proposed GACWT-SSIM, where the green curve is the function $f(x)$. It can be observed that the proposed metric is more consistent with the subjective scores than is GSSIM.

Table 5.3 Performance Comparison of Various IQA Models on LIVE Database

Model	SRCC	KRCC	PLCC	RMSE
PSNR	0.8756	0.6865	0.8723	13.3597
SSIM [41]	0.9479	0.7963	0.9449	8.9455
MS-SSIM [51]	0.9513	0.8044	0.9489	8.6188
IW-SSIM [52]	0.9567	0.8175	0.9522	8.3473
GSSIM [53]	0.9180	0.8013	0.9207	10.7400
MGSD [55]	0.9561	0.8150	0.9512	8.4327
GS [56]	0.9554	0.8131	0.9437	9.0376
GW-SSIM [58]	0.9598	0.8234	0.9551	8.0954
Proposed	0.9536	0.8140	0.9479	8.5285

After the nonlinear regression, we carried out the experiment by following four evaluation criteria to measure objective models: 1) Spearman Rank-order Correlation Coefficient (SRCC), 2) Kendall Rank-order Correlation Coefficient (KRCC), 3) Pearson Linear Correlation Coefficient (PLCC), and 4) Root-Mean-Squared Error (RMSE). Note that for the criteria based on a correlation coefficient (SRCC, KRCC and PLCC), a larger value means better accuracy, while the opposite is the case for RMSE. The performance comparison of various IQA models on the LIVE dataset is shown in Table 5.3. While some of the other gradient-based metrics have slightly better values, the proposed metric outperforms SSIM, MS-SSIM and GSSIM.

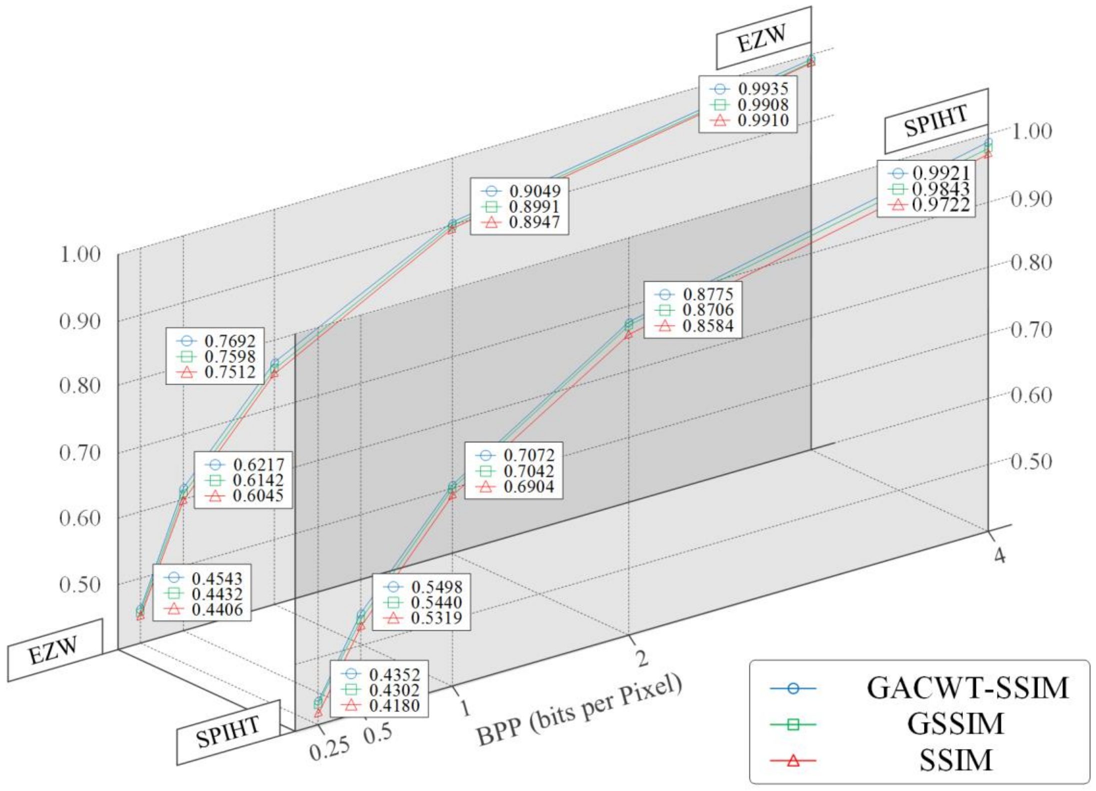
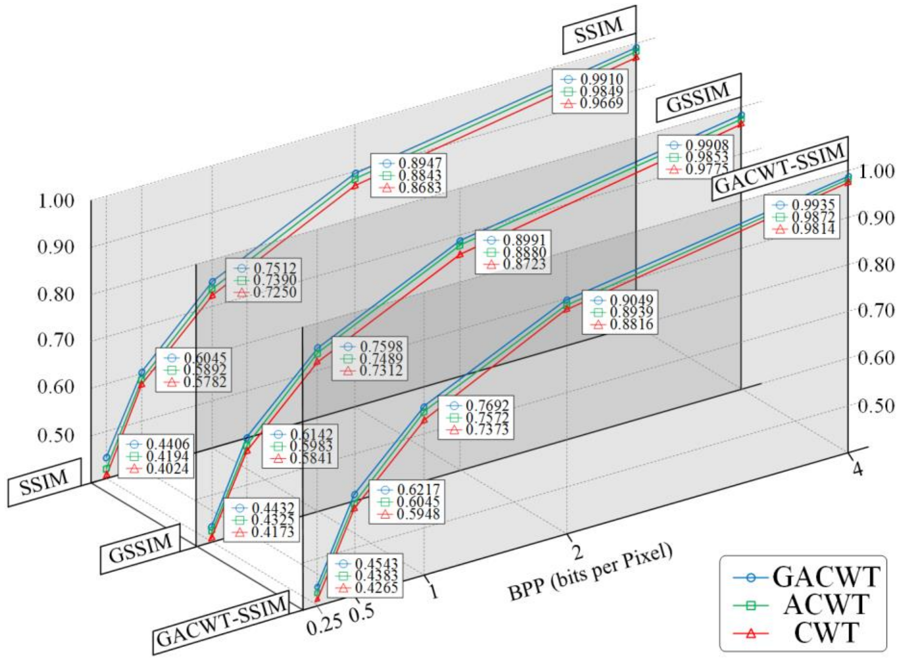


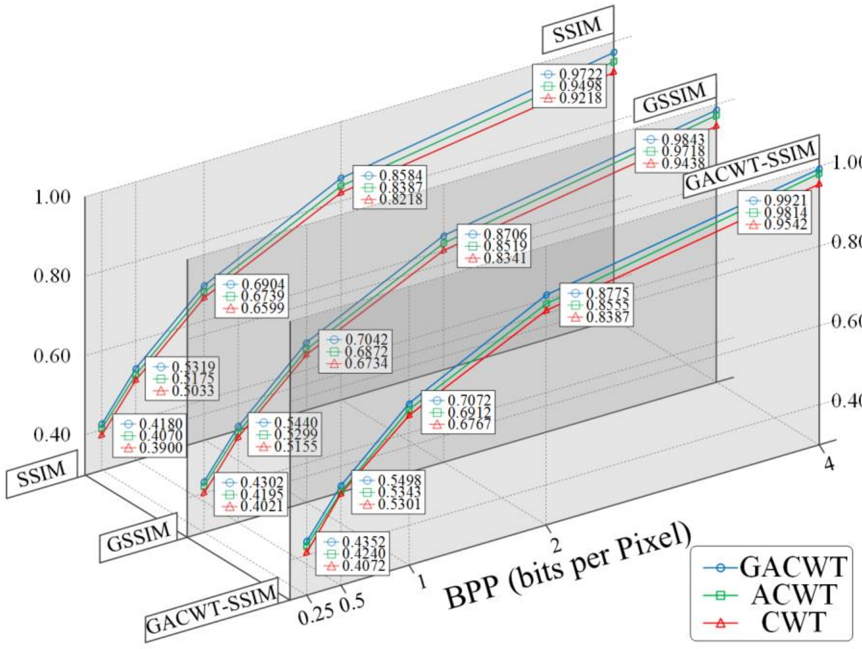
Figure 5.5 Various SSIMs comparison in GACWT using EZW/SPIHT algorithm

5. 4. 3 GACWT in Image Coding

For the evaluation of GACWT, the same experimental procedure that was used for ACWT was applied. The results for the proposed GACWT are summarized in Table 5.4, where for each entry the upper-left value is for EZW and the lower-right one is for SPIHT. All of the values in this table have been calculated using the proposed GACWT-SSIM metric. The images having many edges show slightly higher SSIM values than the others. This is because energy reduction along the curves in the transformed image using GACWT provide additional fidelity in the fine quantization. This is particularly evident in the higher SSIM values for the images having distinct contrast stretches, such as



(a)



(b)

Figure 5.6 (a) Gradient-based SSIMs vs. BPP in EZW algorithm, (b) Gradient-based SSIMs vs. BPP in SPIHT algorithm.

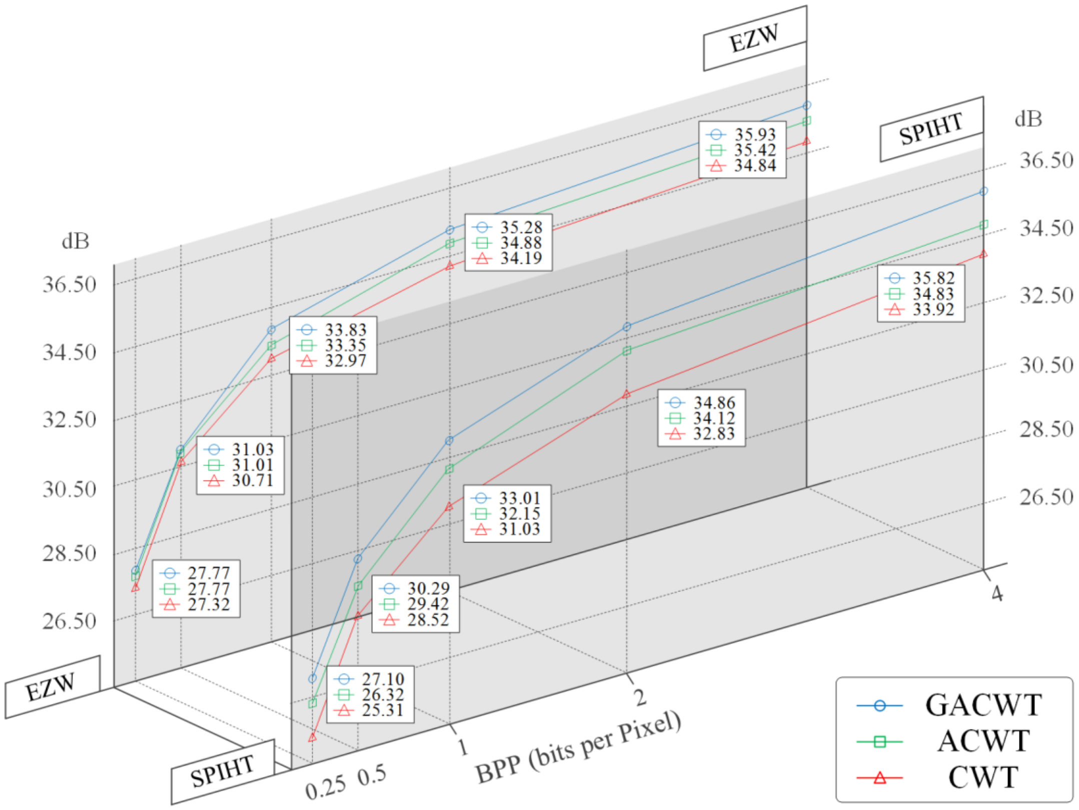


Figure 5.7 PSNR curves in EZW/SPIHT algorithm

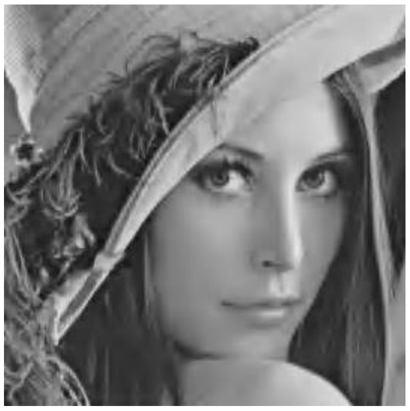
Cameraman, *House* and *Boat*. Figure 5.5 shows the coding performance of GACWT with three different image quality metrics, in each case using the average value for the nine decoded images at various bit rates for both the EZW and SPIHT coders. Figure 5.6 shows the coding performance for three different wavelet transforms, as measured by three different image quality metrics. Figure 5.7 shows the average PSNR of the nine decoded images at various bit rates for three different wavelet transforms. Substantial energy reduction in the high-band for GACWT at edges increases the coding

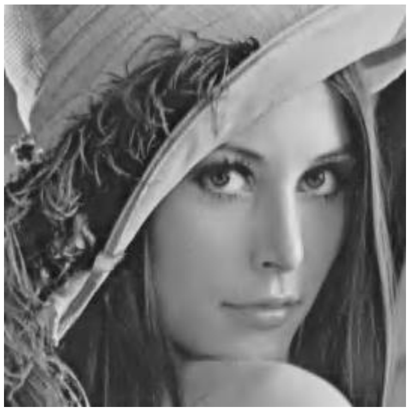
Table 5.4 PSNR/SSIM Performance (EZW)

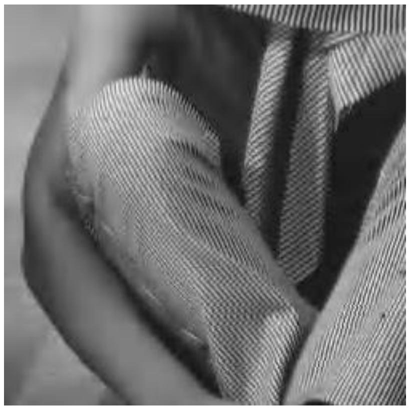
Image	**BPP (Bits per Pixel)**				
	0.25	**0.50**	**1.00**	**2.00**	**4.00**
Lena	0.5047	0.6254	0.7335	0.8787	0.9993
	0.4531	0.5529	0.6721	0.8493	0.9964
Cameraman	0.4257	0.5076	0.6979	0.8871	0.9901
	0.3722	0.4653	0.6489	0.8615	0.9791
Mandrill	0.4838	0.7500	0.9228	0.9968	0.9964
	0.4743	0.6508	0.8403	0.9636	0.9982
Barbara	0.4666	0.6516	0.7689	0.8750	0.9875
	0.4552	0.5713	0.7124	0.8575	0.9837
Peppers	0.4194	0.5275	0.6297	0.8339	0.9853
	0.4105	0.4757	0.5935	0.8073	0.9830
House	0.5241	0.7192	0.9228	0.9949	0.9986
	0.5112	0.6202	0.8333	0.9566	0.9975
Stream and Bridge	0.4296	0.6423	0.8163	0.9424	0.9954
	0.4219	0.5625	0.7444	0.9096	0.9962
Boat	0.3952	0.5900	0.7276	0.8754	0.9912
	0.3902	0.5305	0.6699	0.8528	0.9928
Woman	0.4395	0.5820	0.7036	0.8603	0.9895
	0.4282	0.5190	0.6500	0.8393	0.9862

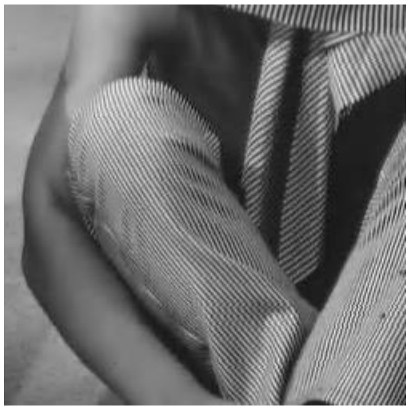
performance. This is reflected in the higher PSNR and similarity measure values in Figures 5.6 and 5.7, respectively.

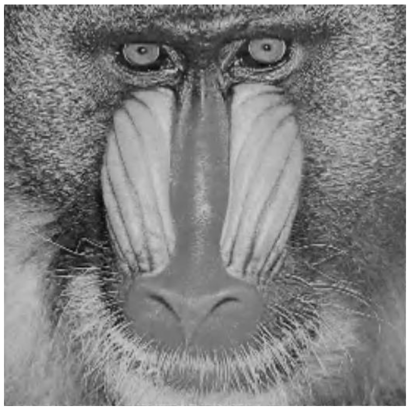
Magnified portions of images decoded with a compression ratio of 16:1 (i.e., 0.5 bpp) using EZW are shown in Figure 5.8. It can be observed that GACWT provides finer image quality than ACWT. Edge details using GACWT are more accurate and more clearly distinguishable compared to those using ACWT, especially for the outline of *Lena*'s hat and the stripes in *Barbara*'s pants. In addition, the outline of *Mandrill*'s eyes and nose using GACWT are well preserved, even for highly compressed images at a low bit rate. In general, it is easier for GACWT to discern the directionality of the stripes and to preserve the shape of sharp edges in the image compared to ACWT. The proposed


(a)


(b)


(c)


(d)


(e)

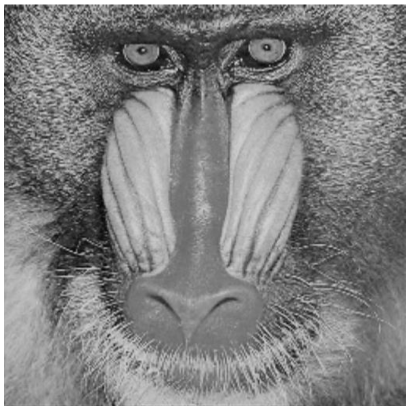

(f)

Figure 5.8 Zoom of image compressed using EZW coder at 0.5 bpp, (a) *Lena* – ACWT (b) *Lena* – GACWT (c) *Barbara* – ACWT (d) *Barbara* – GACWT (e) *Mandrill* – ACWT (f) *Mandrill* – GACWT.

GACWT algorithm preserves the image features and improves the image quality in the edge adaptation of the gradient prediction with less ringing artifacts compared to those from the ACWT algorithm.

5. 5 Architecture and Synthesis Results

The hardware implementation of the GAWCT is organized as a two-parallel structure using pipelining and reordering of the input data stream to achieve high throughput and reduced latency.

5. 5. 1 Proposed Architecture

As noted earlier, we use two edge detection filters, namely Sobel and Prewitt. The first and third of the row-wise elements in each horizontal filter, namely ±2 for Sobel and ±1 for Prewitt, are implemented using an adder/subtractor and a shifter, which only incurs a small hardware cost. Furthermore, the operations can be achieved in a regular lifting-based structure without additive latency since the approximation signals indicate a delay by one clock cycle in updating the detail signals, as shown in the timing diagram of Figure 5.9.

5. 5. 2 Hardware Implementation and Synthesis Result

The block diagram of the proposed GACWT architecture is shown in Figure 5.10. The architecture consists of two curved lifting-based wavelet transform (CLWT) blocks, four signal comparators, two gradient filters and a control signal block. Figure 5.11 illustrates the input signal patterns for the proposed GACWT. The two input signal patterns are symmetric with respect to the horizontal line *m* and each takes part in a single input stream in the two-parallel structure. Table 5.5 lists the control signals for the CLWTs and Figure 5.12 shows the state transition diagram for generating the control signals. The comparators are used to select the signal that results in the minimum value. Each CLWT block, shown in Figure 5.13, computes five detail and approximation signals of the adaptive orientation, with the allowed orientations of $\pm\pi/4$, $\pm\pi/8$ and 0. The block diagram is redrawn using two layers of functional blocks in Figure 5.13 (c). The functional blocks in the upper layer, shown in Figure 5.13 (a), compute detail and approximation signals along the horizontal and diagonal up-right direction. The functional bock in the lower layer, shown in Figure 5.13 (b), computes the signals along the diagonal down-right. Wiring from the upper layer to the lower layer are detailed in Figure 5.13 (b). The total hardware needed to implement each CLWT block is 11 multipliers and 21 adders. Note that the two CLWT blocks are mirror symmetric with respect to a horizontal dashed line shown in Figure 5.10. The UpdateCtrlGen block between the two CLWT blocks generates all of the controls signals for comparing, updating high-pass detail signals and selecting one of the gradient filters, as described in Algorithm 2.

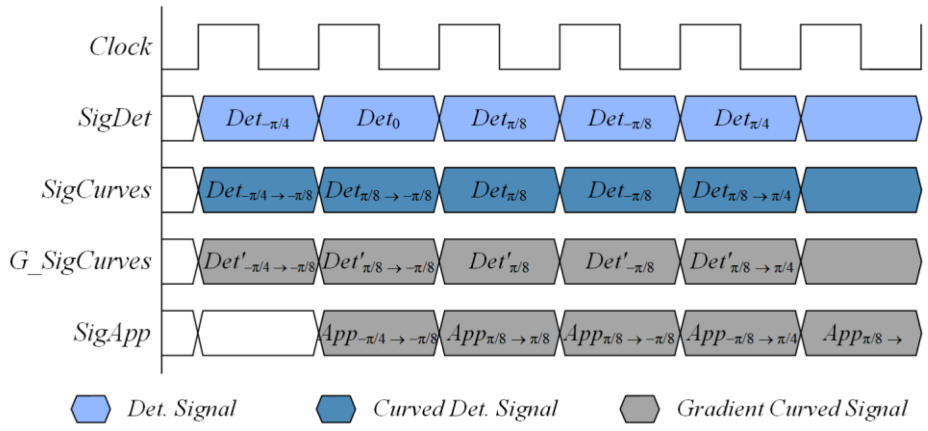


Figure 5.9 Timing diagram of decomposed signals for GACWT – Detail signal and approximation signal.

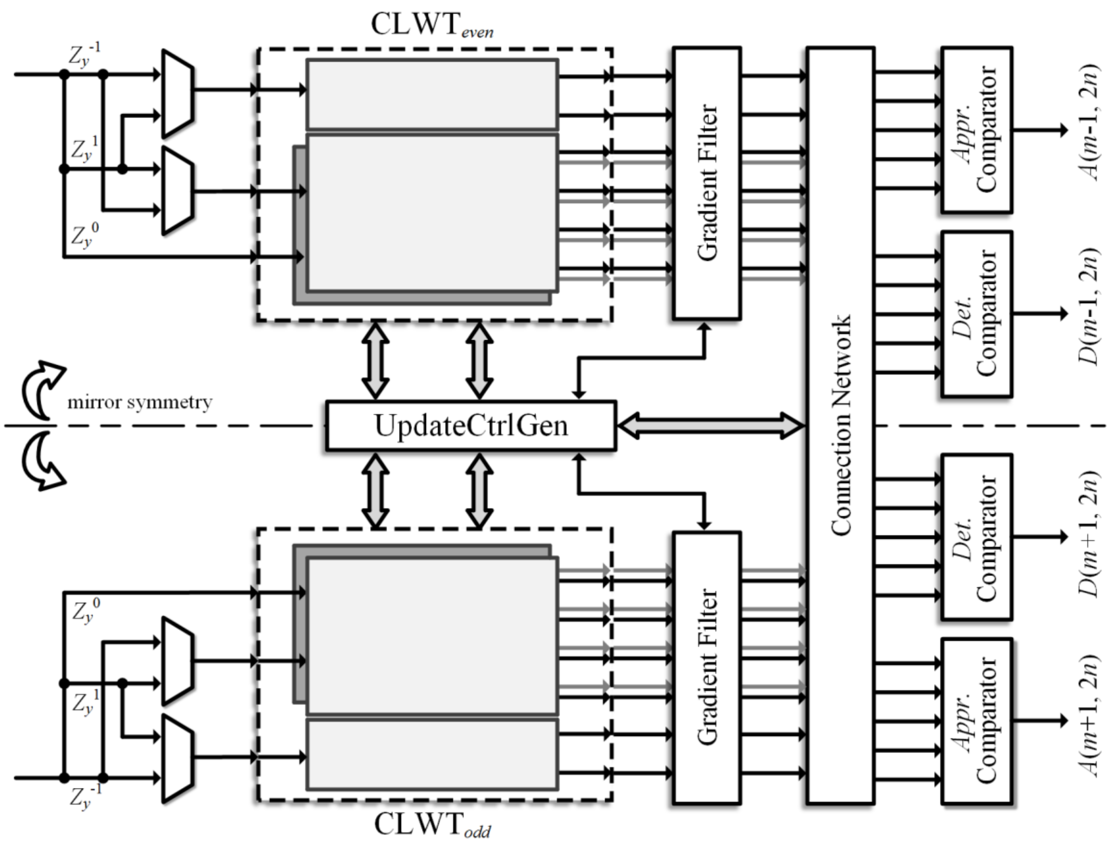
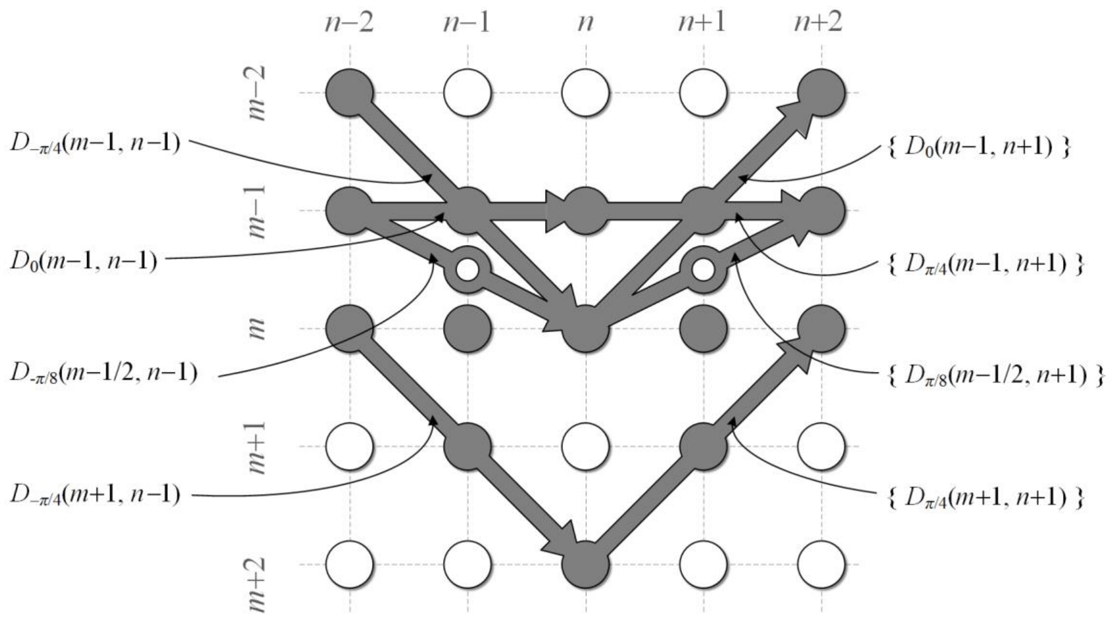
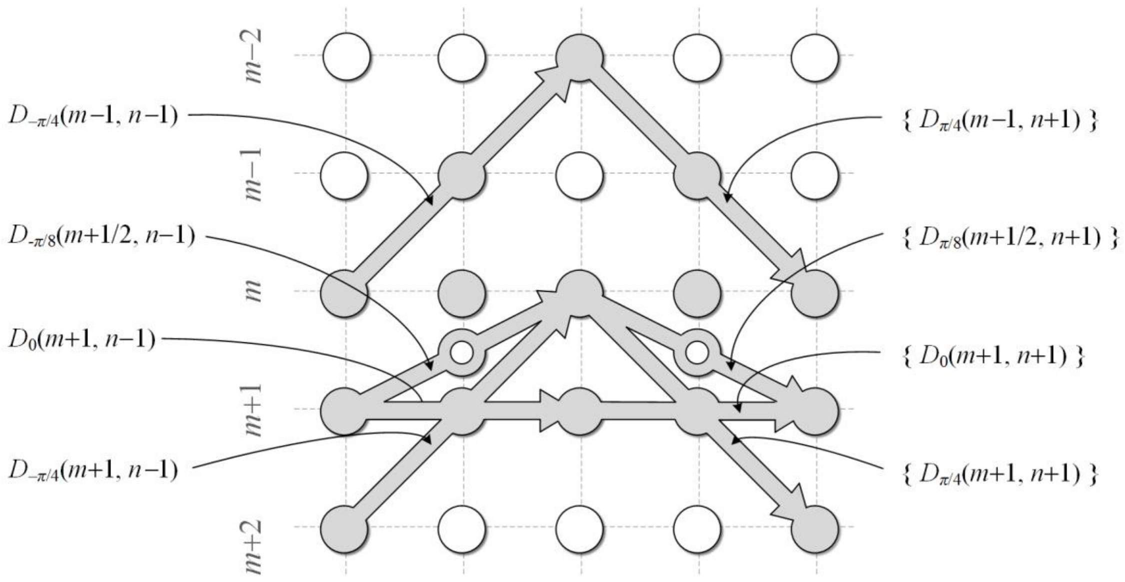


Figure 5.10 Block diagram of the proposed GACWT architecture.



(a)



(b)

Figure 5.11 Input signal patterns for the GACWT. (a) Pattern for CLWT$_{even}$, (b) Pattern for CLWT$_{odd}$.

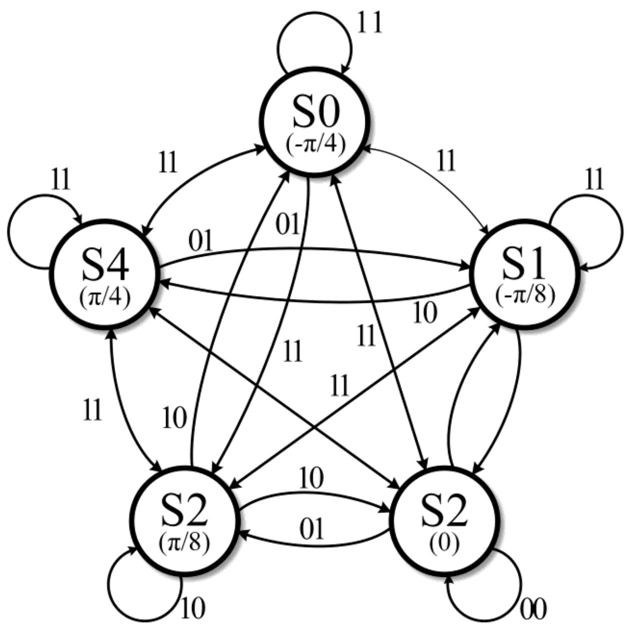


Figure 5.12 State transition diagram for GACWT control signals.

Table 5.5 Control Signals for CLWTs

post_det. / ***pre_det.***	**−π/4**	**−π/8**	**0**	**π/8**	**π/4**
−π/4	1 / 1	1 / 1	1 / 1	0 / 1	1 / 1
−π/8	1 / 1	1 / 1	1 / 0	0 / 0	0 / 1
0	1 / 1	0 / 1	0 / 0	0 / 1	1 / 1
π/8	1 / 0	0 / 0	1 / 0	1 / 1	1 / 1
π/4	1 / 1	0 / 1	1 / 1	1 / 1	1 / 1

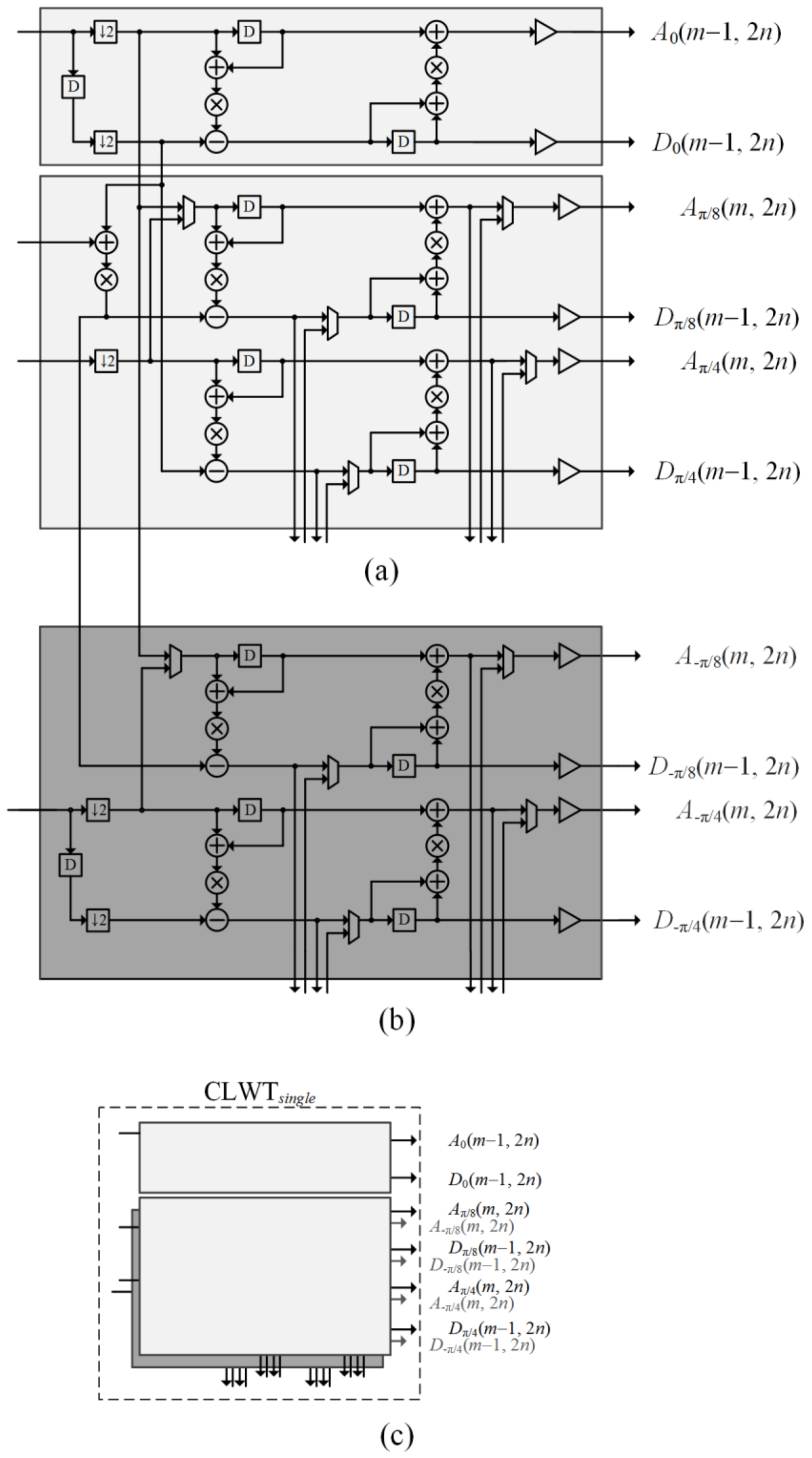


Figure 5.13 A single curved lifting-based wavelet transform (CLWT). (a) Upper layer, (b) Lower layer, (c) CLWT Double-layered representation.

Table 5.6 Implementation Comparison of Adaptive Wavelet Architecture

Resource	**ACWT**	**GACWT**
Wavelet	5/3 LeGall	5/3 LeGall
Technology	CMOS 45nm	CMOS 45nm
Image Size	512 × 512	512 × 512
† IP / CR / TP (bits / MHz / GB/s)	8 / 250 / 4	8 / 250 / 4
Gate Count	§ 28.63K (-)	§ 29.38K (+2.62%)

† Input Precision (IP) / Clock Rate (CR) / Throughput (TP)
§ 2-input NAND gate equivalents (occupies 1.88 μm^2/NAND2)

The synthesis results for the two proposed designs are given in Table 5.6. Logic synthesis was done using a 45 nm CMOS standard cell library that is optimized for a 1.1V supply voltage. The total number of gates (in terms of 2-input NAND gate equivalents) used in the proposed ACWT and GACWT architectures are 28.63K and 29.38K, respectively. The only difference between the architectures for ACWT and GACWT is that the latter contains the two additional gradient filter blocks and their associated control signals. From pre-layout simulation, both of the proposed architectures operate at a clock frequency of 250 MHz and have a data processing throughput rate of 4 Gb/s.

5. 6 Conclusion

Two new image coding algorithms, ACWT and GACWT, and their associated hardware architectures have been presented. In addition, a new metric for image quality assessment has been proposed. Using adaptation along the curves in the wavelet transform can reduce the prediction error signal energy in the lower band and it leads to good image quality for lower bit-rate image coding. The proposed adaptive algorithms, which consider the connectivity of neighboring approximation signals while keeping the direction of the wavelet curves, outperform the previous curved wavelet transform algorithm in terms of the quality of a highly compressed image through the suppression of ringing artifacts. The simulation results show better coding performance for several standard benchmark images, with a corresponding improvement in the PSNR and in the similarity measure values.

Chapter. 6
Conclusion and Future Work

In this dissertation we have studied efficient algorithms and architectures having high accuracy and low complexity for a range of DSP applications. Our main focus was to develop several new algorithms and architectures in order to meet the requirements of real-time throughput and input-output latency, which distinguishes DSP from other general purpose computations. The three main areas addressed in this thesis are summarized below.

In Chapter 2, we proposed a new design for a low-complexity floating-point complex multiplier for DSP applications. The design uses a three-term dot-product unit that is more efficient than a previous two-term fused dot-product unit. Comparisons with a primitive fused adder-subtract unit, a dot-product unit and combinations of these primitive units were also given. The synthesis results show a reduction in area and in power consumption as compared to a previous complex multiplier using two fused dot-product units.

In Chapter 3, an efficient architecture of a 64-point radix-2^2 FFT processor for WLAN applications using two-dimensional algebraic integer encoding was proposed. A radix-2^2 FFT having the simple butterfly structure of a radix-2 FFT and the low multiplicative complexity of a radix-4 FFT was utilized. Wordlength optimization was studied, and a ROMless FFT was designed and synthesized using an FPGA.

Finally, efficient algorithms and architectures for two-dimensional adaptive wavelet transforms were presented in Chapters 4 and 5. An efficient architecture of the wavelet transform for the adaptive directional lifting (ADL) scheme in image coding using algebraic integers was proposed. To avoid accumulating errors from irrational coefficients in the ADL scheme, the proposed approach replaces the coefficients by algebraic integers with appropriate input scaling parameters. This technique reduces the hardware complexity by minimizing multiplications. The simulation results show that the proposed method has better coding performance for several standard benchmark images. Also, two new adaptive curved wavelet transform image coding algorithms, ACWT and GACWT, and their associated hardware architectures were presented, and an associated image quality metric was proposed. The new adaptive algorithms outperform the previous curved wavelet transform algorithm in terms of the quality of a highly compressed image through the suppression of ringing artifacts.

For future work, there are additional opportunities to apply the proposed algorithms to other real-time computations that arise in image and video processing. For example, the algorithm for the determination of the adaptive curves can be applied to the directionlet transform [19] when finding the local dominant directions in the spatial image segments. Also, the techniques for input stream manipulation can be applied to quincunx wavelets, which are based on diagonal interpretation by quincunx sampling.

Bibliography

[1] A. V. Oppenheim and R. W. Schafer, "Discrete-time signal processing," 3rd Ed, Prentice-Hall, 2009.

[2] E. C. Ifeachor and B. W. Jervis, "Digital signal processing," 1st Ed, Prentice-Hall, 2002.

[3] S. M. Kuo and W. Gan, "Digital signal processing: architecture, implementations and applications." Pearson Education, Inc. Upper Saddle River, NJ, 2005.

[4] K. K. Parhi, "VLSI digital signal processing system: design and implementation," John Wiley and Sons, Inc., New York, NY, 1999

[5] R. K. Montoye, E. Hokenek and S. L. Runyon, "Design of the IBM RISC System/6000 floating-point execution unit," *IBM Journal of Research & Development*, vol. 34 (1), pp. 59-70, Jan. 1990.

[6] "IEEE standard 754 for binary floating-point arithmetic"

[7] H. H. Saleh, E. E. Swartzlander, "A floating-point fused add-subtract unit," *IEEE The Fifty-first Midwest Symp. Circuits and Syst.* (*MWSCAS*), pp. 519-522, Aug. 2008.

[8] H. H. Saleh, E. E. Swartzlander, "A floating-point fused dot-product unit," *IEEE International Conference on Computer Design* (*ICCD*), pp. 427-431, Oct. 2008.

[9] J. W. Cooley and J. W. Tukey, "An algorithm for the machine calculation of complex Fourier series," *Mathematics of computations*, vol. 19, no. 90, pp. 297-301, Apr. 1965.

[10] J. Chi and S. Chen "An efficient FFT twiddle factor generator," in *Proc. IEEE European Signal Process. Conf.* (*URASIP*), pp. 1533-1536, Sep. 2004.

[11] L. Cheng and Y. Zhong, "A new ROMless twiddle factor generator for radix-2 1024-point FFT," in *Proc. IEEE Int. Conf. ASIC* (*ICASIC*), vol. 2, pp. 828-831, Oct. 2003.

[12] ISO/IEC FCD 15444-1: 2000, "JPEG 2000 image coding system," 2000.

[13] ISO/IEC FCE 14496-2, Information Technology - "Coding of Audio-Visual Objects," 2004.

[14] I. Daubechies and W. Sweldens, "Factoring wavelet transforms into lifting steps," *J. Fourier Anal. Appl.*, vol. 4, no. 3, pp. 247-269, 1998.

[15] W. Sweldens, "Lifting scheme: a new philosophy in biorthogonal wavelet constructions," in Proc. SPIE 2569, *Wavelet Applications in Signal and Image Process. III*, vol. 2569, pp. 68-79, Sep. 1995.

[16] O. N. Gerek and A. E. Cetin, "A 2-D orientation-adaptive prediction filter in lifting structures for image coding," *IEEE Trans. Image Process.*, vol. 15, no. 1, pp. 106-111, Jan. 2006.

[17] W. Ding, F. Wu, X. Wu, S. Li, and H. Li, "Adaptive directional lifting-based wavelet transform for image coding," *IEEE Trans. Image Process.*, vol. 16, no. 2, pp. 416-427, Feb. 2007.

[18] Y. Liu and K. N. Ngan, "Weighted adaptive lifting-based wavelet transform for image coding," *IEEE Trans. Image Process.*, vol. 17, no. 4, pp. 500-511, Apr. 2008.

[19] J. H. Cozzens and L. A. Finkelstein, "Computing the discrete Fourier transform using residue number systems in a ring of algebraic integers," *IEEE Trans. Inf. Theory*, vol. IT-31, no. 5, pp. 580-588, Sep. 1985.

[20] D. Takahashi, "A radix-16 FFT algorithm suitable for multiply-add instruction based on Goedecker method," in *Proc. IEEE Int. Conf. Acoust., Speech, Signal Process.* (*ICASSP*), vol. 2, pp. II-845-848, Apr. 2003.

[21] E. Quinnell, E. E. Swartzlander and C. Lemonds, "Bridged floating-point fused multiply-add design," *IEEE Trans. on Very Large Scale Integr.* (*VLSI*) Syst., vol. 16, no. 12, pp. 1727-1731, Dec. 2008.

[22] J. W. Hartwell, "A procedure for implementing the fast Fourier transform on small computers," *IBM Journal of Research and Development*, vol. 15, pp. 355-363, Sep. 1971.

[23] E. E. Swartzlander and H. H. Saleh, "Floating-point Implementation of Complex Multiplication," in *Proc. IEEE Conf. Record of the Forty-Third Asilomar Conf. Signals, Syst. Comput.*, pp. 926-029, Nov. 2009.

[24] R. G. Lyons, Understanding Digital Signal Processing, 3rd Ed, Prentice-Hall, 2010.

[25] IEEE 802.11 Working Group, "IEEE 802.11-2007: Wireless LAN medium access control (MAC) and physical layer (PHY) specifications," Jun. 2007.

[26] Y. Lin, H. Liu and C. Lee, "A 1-GS/s FFT/IFFT processor for UWB applications," *IEEE J. Solid-State Circuits*, vol. 40, no. 8, pp. 1726-1735, Aug. 2005

[27] H. Lee and M. Shin, "A high-speed four-parallel Radix-24 FFT/IFFT processor for UWB applications." *in Proc. IEEE Int. Symp. Circuits Syst.* (*ISCAS*), Seattle, WA, USA, 2008, pp. 960-963.

[28] K. Wahid, V. Dimitrov and G. Jullien, "Error-free computation of 8×8 2D DCT and IDCT using two-dimensional algebraic integer quantization," in *Proc. 17th IEEE Symp. Comput. Arith.* (*ARITH*), pp. 214-221, Jun. 2005.

[29] V. Dimitrov and R. Baghaie, "Computing discrete Hartley transform using algebraic integers," *Asilomar Conf. Signals, Syst. Comput,* (ACSSC), pp. 1351-1355, Oct. 1999.

[30] K. Wahid, V. Dimitrov, G. Jullien and W. Badawy, "Error-free computation of Daubechies wavelets for image compression applications," *IEEE Electron Lett.*, Vol. 39, pp. 428-429, Mar. 2003.

[31] R. Mehra and P. Kataria, "FPGA Based area efficient 64-point FFT using MAC algorithm," *Int. J. Recent Trends Elect. Eng.* (*IJRTE*), vol 3, 2013.

[32] Y. Ouerhani, M. Jridi and A. Alfalou, "Implementation techniques of high-order FFT into low-cost FPGA," *IEEE 54th Int. Midwest Symp. Circuits. Syst.* (*MWSCAS*), pp. 1-4, Aug. 2011.

[33] A. Kumar, U. Tripathi, R. Verma and M. Mishra, "64- point Radix-4 FFT butterfly realization using FPGA," *Int. J. Eng. Innovative Tech.* (*IJEIT*), vol. 4, pp. 57-60, Oct. 2014.

[34] C. Chang and B. Girod, "Directional-adaptive discrete wavelet transform for image compression," *IEEE Trans. Image Process*., vol. 16, no. 5, pp. 1289-1302, May 2007.

[35] R. L. Claypoole, G. Davis, W. Sweldens and R. G. Baraniuk, "Nonlinear wavelet transforms for image coding via lifting," *IEEE Trans. Image Process*., vol. 12, no. 12, pp. 1449-1459, Dec. 2003.

[36] 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 Trans. Circuits Syst*. I, Reg. Papers, vol. 60, no. 6, pp. 1455-1468, Jun. 2013.

[37] P. Balakrishnan and K. Wahid, “A novel lifting algorithm for Daubechies wavelets using algebraic integers.” in *Proc. 24th IEEE Canadian Conf. Electr. Comput. Eng.* (*CCECE*), pp. 1-4, May 2013.

[38] D. Le Gall and A. Tabatabai, “Sub-band coding of digital images using symmetric short kernel filters and arithmetic coding techniques,” in *Proc. IEEE Int. Conf. on Acoust, Speech, Signal Process.* (ICASSP), New York, NY, USA, 1988, pp. 761-764.

[39] J. M. Shapiro, “Embedded image coding using zerotrees of wavelet coefficients,” *IEEE Trans. Signal Process.*, vol. 41, no. 12, pp. 3445-3462, Dec. 1993.

[40] A. Said and W. A. Pearlman, “A new, fast and efficient image codec based on set partitioning,” *IEEE Trans. Circuits Syst. Video Technol.*, vol. 6, no. 3, pp. 243-250, Jun. 1996.

[41] Z. Wang, A. C. Bovik, H. R. Sheikh and E. P. Simoncelli, “Image quality assessment: from error visibility to structural similarity,” *IEEE Trans. Image Process.*, vol. 13, no. 4, pp. 600–612, Apr. 2004.

[42] The USC-SIPI Image Database [Online]. Available: http://sipi.usc.edu/database/database.php

[43] K. Andra, C. Chakrabarti and T. Acharya, “A VLSI architecture for lifting-based forward and inverse wavelet transform*,” IEEE Trans. Signal Process.*, vol. 50, no. 4, pp. 966-977, Apr. 2002.

[44] M. E. Angelopoulou, P. Y. K. Cheung, K. Masselos and Y. Andreopoulous “Implementation and comparison of 5/3 Lifting 2D Discrete Wavelet Transform Computation Schedules on FPGAs*” J. Signal Process. Syst.*, vol. 51, no. 1, pp. 3-21, Apr. 2008.

[45] D. Dia, M. Zeghid, T. Saidani, M. Atri, B. Bouallegue, M. Machhout and R. Tourki, “Multi-level discrete wavelet transform architecture design,” in *Proc. World Congress on Engineering* (WCE), vol. I, London, U.K., 2009, pp. 191-195.

[46] L. Liu, H. Meng and M. Zhang, "An ASIC implementation of lifting-based 2-D discrete wavelet transform," in *Proc. IEEE Asia Pacific Conf. Circuits Syst.* (*APCCAS*), Singapore, 2006, pp. 271-274.

[47] Y. Seo and D. Kim, "VLSI architecture of line-based lifting wavelet transform for motion JPEG2000," *IEEE J. Solid-State Circuits*, vol. 42, no. 2, pp. 431-440, Feb. 2007.

[48] H. Yamauchi, S.Okada, K. Taketa, T. Ohyama, Y. Matsuda, T. Mori, S. Okada, T. Watanabe, Y. Matsuo, Y. Yamada, T. Ichikawa and Y. Matsushita, “Image processor capable of block-noise-free JPEG2000 compression with 30 frames/s for digital camera applications,” in *Proc. IEEE Int. Solid-State Circuits Conf.* (ISSCC), San Francisco, CA, USA, 2003, vol. 1, pp. 45-477.

[49] S. K. Madishetty, “VLSI architecture for the 4-tap and 6-tap 2-D Daubechies wavelet filters using algebraic integers,” M.S. thesis, Dept. Elect. and Comput. Eng., Univ. Akron, OH, USA, Aug. 2013.

[50] D. Wang, L. Zhang, A. Vincent and F. Speranza, “Curved wavelet transform for image coding,” *IEEE Trans. Image Process.*, vol. 15, no. 8, pp. 2413-2421, Aug. 2006

[51] Z. Wang, E. P. Simoncelli, and A. C. Bovik, “Multiscale structural similarity for image quality assessment,” in *Proc. IEEE Asilomar Conf. Signals, Systems, and Computers*, vol. 2, pp. 1398–1402, Pacific Grove, CA, Nov. 2003.

[52] Z. Wang and Q. Li, “Information content weighting for perceptual image quality assessment,” *IEEE Trans. Image Process*., vol. 20, no. 5, pp. 1185–1198, May 2011.

[53] G. H. Chen, C. L. Yang and S. L. Xie, "Gradient-based structural similarity for image quality assessment," in *Proc. IEEE ICIP*, pp. 2929–2932, Atlanta, GA, Oct. 2006.

[54] D. Kim and R. Park, "New image quality metric using the Harris response," *IEEE Signal Process. Letters*, vol. 16, no. 7, pp. 616–619, Jul. 2009.

[55] G. Cheng, J. C. Huang, C. Zhu, Z. Liu and L. Cheng, "Perceptual image quality assessment using a geometric structural distortion model," in *Proc. IEEE ICIP*, pp. 325–328, Hong Kong, China, Sep. 2010.

[56] A. Liu, W. Lin and M. Narwaria, "Image quality assessment based on gradient similarity," *IEEE Trans. Image Process.*, vol. 21, no. 4, pp. 1500–1512, Apr. 2012.

[57] X. Chen, R. Zhang, and S. Zheng, "Image quality assessment based on local edge direction histogram," in *Proc. IEEE IASP*, pp. 108–112, Hubei, China, Oct. 2011.

[58] Q. Li, Y. Fang and W. Lin, "Gradient-weighted structural similarity for image quality assessments,' in *Proc. IEEE ISCAS*, pp. 2165–2168, Lisbon, Portugal, May 2015.

[59] D. Zheng, L. Zhang, D. Wang and A. Vincent, "Rate distortion optimized curved determination for curved wavelet image coding," in *Proc. IEEE ICIP*, pp. 2829-2832, Cairo, Egypt, Nov. 2009.

[60] J. Huang, G. Cheng, Z. Liu, C. Zhu and B. Xiu, "Synthetic aperture radar image compression using tree-structured edge-directed orthogonal wavelet packet transform," *AEÜ Int. J. Electron. Commun.* 66(3), pp. 195-203, Mar. 2012

[61] D. Jayachandra and A. Makur, "Directionlets using in-phase lifting for image representation," *IEEE Trans. Image Process.*, vol. 23, no. 1, pp. 240–249, Jan. 2014.

[62] D. L. Donoho, “De-noising by soft-thresholding,” *IEEE Transactions on Information Theory*, vol. 41, no. 3, pp. 613–627, May 1995.

[63] R. D. da Silva, R. Minetto, W. R. Schwartz and H. Pedrini, “Adaptive edge-preserving image denoising using wavelet transforms” *Pattern. Anal. Appl.* 16(4), pp. 1–14, 2012.

[64] M. P. Sampat, Z. Wang, S. Gupta, A. C. Bovik, and M. K. Markey, “Complex wavelet structural similarity: A new image similarity index,” *IEEE Trans. Image Process.*, vol. 18, no. 11, pp. 2385–2401, Nov. 2009.

[65] C. L. Yang, W. R. Gao and L. M. Po, “Discrete wavelet transform-based structural similarity for image quality assessment,” in *Proc. IEEE ICIP*, pp. 377-380, San Diego, CA. Oct. 2008.

[66] H. R. Sheikh, Z. Wang, L. Cormack, and A. C. Bovik, “LIVE image quality assessment database release 2,” 2005, available: http://live.ece.utexas.edu/research/quality.

[67] Video Quality Experts Group (VQEG), Final report from the video quality experts group on the validation of objective models of video quality assessment, phase II, 2003, [Online] available: http://www.vqeg.org