Optimal, Multiplierless Implementations of the Discrete Wavelet Transform for Image Compression Applications

Kishore A. Kotteri

Thesis submitted to the Faculty of the
Virginia Polytechnic Institute and State University
(Virginia Tech)
in partial fulfillment of the requirements for the degree of

Master of Science
in
Electrical Engineering

Dr. A. E. Bell, Chair
Dr. B. D. Woerner
Dr. L. A. DaSilva

April 27, 2004
Blacksburg, Virginia

KEYWORDS: biorthogonal 9/7, canonical signed digit, discrete wavelet transform, field programmable gate array, filter bank, image compression, lifting, multiplierless, perfect reconstruction, quantization, simulated annealing, wavelets.

Optimal, Multiplierless Implementations of the Discrete Wavelet Transform for Image Compression Applications

Kishore A. Kotteri

(ABSTRACT)

The use of the discrete wavelet transform (DWT) for the JPEG2000 image compression standard has sparked interest in the design of fast, efficient hardware implementations of the perfect reconstruction filter bank used for computing the DWT. The accuracy and efficiency with which the filter coefficients are quantized in a multiplierless implementation impacts the image compression and hardware performance of the filter bank. A high precision representation ensures good compression performance, but at the cost of increased hardware resources and processing time. Conversely, lower precision in the filter coefficients results in smaller, faster hardware, but at the cost of poor compression performance. In addition to filter coefficient quantization, the filter bank structure also determines critical hardware properties such as throughput and power consumption.

This thesis first investigates filter coefficient quantization strategies and filter bank structures for the hardware implementation of the biorthogonal 9/7 wavelet filters in a traditional convolution-based filter bank. Two new filter bank properties—"no-distortion-mse" and "deviation-at-dc"—are identified as critical to compression performance, and two new "compensating" filter coefficient quantization methods are developed to minimize degradation of these properties. The results indicate that the best performance is obtained by using a cascade form for the filters with coefficients quantized using the "compensating zeros" technique. The hardware properties of this implementation are then improved by developing a cascade polyphase structure that increases throughput and decreases power consumption.

Next, this thesis investigates implementations of the lifting structure—an orthogonal structure that is more robust to coefficient quantization than the traditional convolution-based filter bank in computing the DWT. Novel, optimal filter coefficient quantization techniques are developed for a rational and an irrational set of lifting coefficients. The results indicate

that the best quantized lifting coefficient set is obtained by starting with the rational coefficient set and using a "lumped scaling" and "gain compensation" technique for coefficient quantization.

Finally, the image compression properties and hardware properties of the convolution and lifting based DWT implementations are compared. Although the lifting structure requires fewer computations, the cascaded arrangement of the lifting filters requires significant hardware overhead. Consequently, the results depict that the convolution-based cascade polyphase structure (with "z_1-compensated" coefficients) gives the best performance in terms of image compression performance and hardware metrics like throughput, latency and power consumption.

Acknowledgments

I would like to express my sincere gratitude to my advisor, Dr. Amy Bell. Her sound advice has been my beacon throughout my stay at Virginia Tech. I am grateful for her valuable feedback and suggestions, for they have greatly added to the value of this thesis. From her, I learnt how to take an idea from its abstract infancy, give it shape and put it down effectively on paper. I hope to achieve her level of organization and work ethic one day.

I would also like to thank Dr. Joan Carletta from the University of Akron for being patient with my numerous questions and comments. She made time from her busy schedule to help me get the results I needed to complete this work.

Many thanks to Dr. Brian Woerner and Dr. Luiz DaSilva for agreeing to serve on my committee and reviewing this work. I am also thankful to my DSPCL colleagues, Satyabrata Rout and Krishnaraj Varma. I will always cherish the lively discussions we had ranging from politics to religion to cricket. Even if many of these discussions did not come close to finding global solutions, they provided a welcome break from looking for artifacts in Lena. They made the DSPCL a fun place to work in.

Finally, I express my sincere gratitude to my parents. They made sure that my brother and I had the best of opportunities growing up. Their silent belief in my judgment, gave me the confidence to pursue a graduate degree. They form my support structure, and I know they will always be there for me.

to my parents and my brother

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Motivation

The Discrete Wavelet Transform (DWT) is the transform of choice at the heart of recent image compression algorithms. Adopted by the JPEG2000 image compression standard [1], it significantly outperforms algorithms based on other transforms, such as the discrete cosine transform, in terms of objective metrics as well as perceptual image quality [2]. The success of the DWT stems from its ease of computation and its inherent decomposition of an image into non-overlapping subbands that enables the design of efficient quantization algorithms and allows for incorporation of the human visual system. A DWT based image codec is a good choice for applications such as remote exploration, urban search and rescue operations and satellite imaging. These applications require transmission of still images from remote image acquisition devices to base stations. Images are compressed after acquisition to reduce the number of data bits that need to be transmitted back to the base station over a wired or wireless communication channel. A good hardware codec employed in these applications will have: low latency and high throughput if real-time operation is desired, low power consumption if working in an untethered, battery-driven environment, small hardware size, and most importantly, high fidelity for the reconstructed image after compression.

This thesis focuses on a field programmable gate array (FPGA) implementation of a DWT codec for the biorthogonal 9/7 wavelet. This wavelet has been shown to possess properties favorable for image compression; part I of the JPEG2000 standard specifies this wavelet for lossy compression. The DWT is typically computed using a perfect reconstruction (PR) filter bank. The performance of an FPGA implementation of the filter bank depends on the following two implementation design issues:

1. the filter bank structure and filter coefficient quantization; and,
2. the hardware architecture used to implement the filter bank structure.

The filter bank structure determines hardware metrics such as throughput and latency while filter coefficient quantization impacts the signal processing properties of the filter bank and determines its image compression performance. The hardware architecture of the filter bank determines properties such as latency and power consumption. This thesis investigates the first issue of filter bank implementation, namely, filter bank structure and coefficient quantization.

The image compression performance of the filter bank implementation critically depends on the two perfect reconstruction (PR) conditions: the no-distortion condition and the no-aliasing condition. When the irrational coefficients of the biorthogonal 9/7 wavelet filters are implemented in a floating point format, both PR conditions are satisfied and the filter bank gives perfect reconstruction under lossless compression. For fast hardware implementation on an FPGA, the filter coefficients are implemented in a multiplierless manner after representing them as sum-and-difference-of-powers-of-two (SPT). Multiplication is then achieved by shifting and adding. Thus, the filter coefficients have to be *quantized*, i.e. approximated by fixed point SPT representations. The number of non-zero terms in the SPT representation of a coefficient, denoted by T, provides an estimate of the hardware cost associated with implementing the coefficient. This quantization of the filter coefficients alters the no-distortion PR condition and, consequently, image compression performance is affected.

In general, the more non-zero terms (i.e. large T) used to represent the filter coefficients, the closer the quantized filter coefficients are to the unquantized coefficients, and the closer the quantized compression performance is to the unquantized performance. However, more non-zero terms means higher hardware cost. Conversely, fewer non-zero terms means lower hardware cost, but worse compression performance. Thus there is a trade-off between hardware cost and compression performance.

The filter bank structure also influences the performance of the filter bank. Cascade filter structures are more immune to coefficient quantization than direct structures and, in general result in better compression performance for the same T. Furthermore, polyphase structures operate at higher clock speeds than non-polyphase structures and hence result in better throughput. The lifting structure—an alternative to the traditional filter bank structure—offers the advantage of an orthogonal implementation that is more robust to coefficient quantization.

This thesis studies FPGA implementations of three filter bank structures: traditional non-polyphase structures, traditional polyphase structures, and lifting structures. All three structures are evaluated in terms of compression performance (using peak signal-to-noise ratio—PSNR) and various hardware metrics. For each structure, optimal quantized values are found for the filter coefficients. These coefficients enable the implementation of a fast, multiplierless DWT codec that generates the best possible PSNR performance for the given structure. The resulting comprehensive performance evaluation allows optimal choices for a biorthogonal 9/7 DWT implementation to be made based on the needs of the given application.

1.2 Previous Work

The basic issue in the design of a single multiplierless filter is the quantization of the ideal, real-valued coefficients as SPT. This must be accomplished while limiting the overall amount

of hardware, measured in terms of T (the total number of SPT terms in all of the quantized coefficients). Previous work in this area poses the problem as an optimization problem. Traferro et. al. use tabu search, optimizing the absolute error in magnitude of frequency response, weighted to favor particular frequency bands [3]. Lim et. al. use a measure of the sensitivity of the frequency response of a filter to a particular coefficient to decide how many of the T total terms to allocate to each individual coefficient, and then use an integer-programming algorithm [4]. Other optimization techniques such as mixed-integer linear programming [5, 6] are also used.

In the DWT, a filter bank is required rather than a single filter. Now the properties of the filter bank—and not the properties of a single filter—must be preserved after quantization. Mao et al. develop a method for the design of multiplierless perfect reconstruction FIR filter banks, but their method, which focuses on achieving low system delay, places no constraint on the amount of resulting hardware [7]. Akansu develops a method for the design of multiplierless perfect reconstruction quadrature mirror filters (QMFs), using quadratic programming to maximize the energy compaction performance [8]. Chen et al. propose an optimal T allocation scheme for finding multiplierless transform approximations; they employ a quasi-coordinate descent algorithm to minimize the mean-squared-error between the unquantized and quantized transform outputs. They illustrate their method using the discrete cosine transform [9]. Usevitch and Betancourt [10] develop conditions on the signal bit widths for fixed-point filter bank implementations so that the no-aliasing condition is met. In addition to no-aliasing, this thesis also addresses the no-distortion condition that is crucial to the quality of the compressed image. Kim and Li [11] address lossy and lossless image compression using biorthogonal wavelets with dyadic, rational filter coefficients. Although their methods are efficient from the hardware perspective, they cannot accommodate wavelet filters whose coefficients are not expressed as dyadic rational numbers; for example, their procedure could not implement the biorthogonal 9/7 wavelet filter.

Adam et. al. and Cheng et. al. seek to replace the original, real-valued 9/7 coefficients with rational or dyadic coefficients that are easier to implement in hardware [12, 13]. However,

these new wavelets do not have the superior image compression properties of the biorthogonal 9/7 wavelet, and result in lower PSNR values for compressed images. Unlike this previous work in filter bank design, this thesis focuses on how best to implement *any given* biorthogonal, infinite-precision FIR filter bank in a limited amount of fixed-point, high-speed digital hardware such that the filter bank properties are minimally impacted. The design process is driven by the perfect reconstruction requirements, rather than any optimization technique.

The lifting scheme [14, 15, 16] was developed by Sweldens as a technique to construct new wavelets and factor existing wavelets into smaller building blocks. Since then, this method has found favor for hardware implementations thanks to fewer required computations and the ability to perform in-place computations. Factorization of the biorthogonal 9/7 wavelets [17] results in six irrational lifting coefficients, that have to be quantized as SPT for hardware implementation. Tay and Guangjun et. al. show that rational coefficients can be used in place of the irrational coefficients with only slight changes in filter properties [18, 19]. These rational coefficients are easier to implement in hardware and give image compression performance almost identical to the irrational lifting coefficients. No published literature exists, that attempts to arrive at quantized values of the irrational biorthogonal 9/7 lifting coefficients or study the impact of lifting coefficient quantization on image compression performance. This thesis uses the perfect reconstruction conditions for quantization of the rational and irrational lifting coefficients and arrives at optimal quantized lifting values for the best possible image compression performance.

1.3 Significance of this Work

The related literature does not address the problem of quantizing filter or lifting coefficients for *any given* biorthogonal filter bank with the objective of keeping degradation of image compression performance to a minimum. This thesis evaluates the impact of coefficient quantization on the image compression performance of a filter bank from a signal processing

perspective, and uses the knowledge gleaned to drive new quantization techniques. The contributions of this thesis are as follows.

1. Identification of filter bank properties critical for image compression performance.
2. Derivation of two quantization methods, "gain compensation" and "zero compensation", for quantization of biorthogonal filter bank coefficients. The "zero compensation" method can also be applied to the quantization of any FIR filter.
3. Development of a polyphase filter bank structure that employs cascade form quantized coefficients, resulting in high throughput and good image compression properties.
4. Design of optimal quantization procedures for lifting coefficients.
5. Comparison of traditional convolution and lifting filter bank implementations on FPGA using hardware metrics such as latency, throughput, power consumption and hardware size.

The work done for this thesis has contributed to a number of publications. Development of the "gain compensation" and "zero compensation" quantization techniques is presented in [20]. Extension of zero compensation technique to quantization of FIR filters is treated in [21] and [22]. Hardware performance of polyphase implementations of the filter bank are evaluated in [23] and optimal quantization techniques for lifting coefficients are developed in [24].

1.4 Organization of this Thesis

The remainder of the thesis is organized as follows. Chapter 2 presents an overview of the DWT and its application to image compression. It also presents various ways to implement the traditional DWT and then introduces the figures of merit used to evaluate hardware implementations. Chapter 3 studies non-polyphase implementations of the biorthogonal 9/7

filter bank and derives two quantization techniques to improve the performance of quantized filter banks. Chapter 4 implements the filter bank in a polyphase form for increased throughput over the non-polyphase structures. Chapter 5 begins with a brief background on the lifting structure and continues to design and evaluate six different optimal quantized lifting implementations. In Chapter 6, the best non-polyphase, polyphase and lifting implementations from the previous chapters are compared in terms of hardware and image compression performance. Chapter 7 concludes with recommendations on the best filter bank structure and coefficient quantization scheme for an FPGA DWT implementation, and makes suggestions for future work.

Chapter 2

Background

A block diagram of a wavelet based image compression system is shown in Figure 2.1. At the heart of the analysis (or compression) stage of the system is the forward discrete wavelet transform (DWT). Here, the input image is mapped from a spatial domain, to a scale-shift domain. This transform separates the image information into octave frequency subbands. The expectation is that certain frequency bands will have zero or negligible energy content; thus, information in these bands can be thrown away or reduced so that the image is compressed without much loss of information.

The DWT coefficients are then quantized to achieve compression. Information lost during the quantization process cannot be recovered and this impacts the quality of the reconstructed image. Due to the nature of the transform, DWT coefficients exhibit spatial correlation, that are exploited by quantization algorithms like the embedded zero-tree wavelet (EZW) and set partitioning in hierarchical trees (SPIHT) for efficient quantization. The quantized coefficients may then be entropy coded; this is a reversible process that eliminates any redundancy at the output of the quantizer.

In the synthesis (or decompression) stage, the inverse discrete wavelet transform recovers the original image from the DWT coefficients. In the absence of any quantization the re-

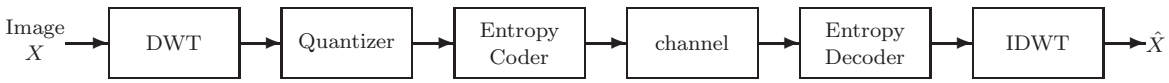


Figure 2.1: A wavelet based image compression system.

constructed image will be identical to the input image. However, if any information was discarded during the quantization process, the reconstructed image will only be an approximation of the original image. Hence this is called *lossy* compression. The more an image is compressed, the more information is discarded by the quantizer; the result is a reconstructed image that exhibits increasingly more artifacts. Certain integer wavelet transforms exist that result in DWT coefficients that can be quantized without any loss of information. These result in *lossless* compression, where the reconstructed image is an exact replica of the input image. However, compression ratios achieved by these transforms are small compared to lossy transforms (e.g. 4:1 compared to 40:1).

The remainder of this chapter explores the discrete wavelet transform, and the hardware implementation of the transform stage for computation of the 2D DWT for an image.

2.1 The Discrete Wavelet Transform

2.1.1 Multiresolution

The discrete wavelet transform decomposes the $L^2(\mathcal{R})$ space into a set of subspaces V_j, where,

$$\cdots \subset V_{-2} \subset V_{-1} \subset V_0 \subset V_1 \subset V_2 \cdots$$

and

$$\bigcup_{-\infty}^{\infty} V_j = L^2(\mathcal{R}); \qquad \bigcap_{-\infty}^{\infty} V_j = \{\emptyset\}.$$

Figure 2.2 illustrates these nested subspaces. Subspace V_j is spanned by the set of basis functions given by $\{2^{j/2}\phi(2^j t - k)|_{j,k\in\mathcal{Z}}\}$, which are derived from dyadic shifts and scales of

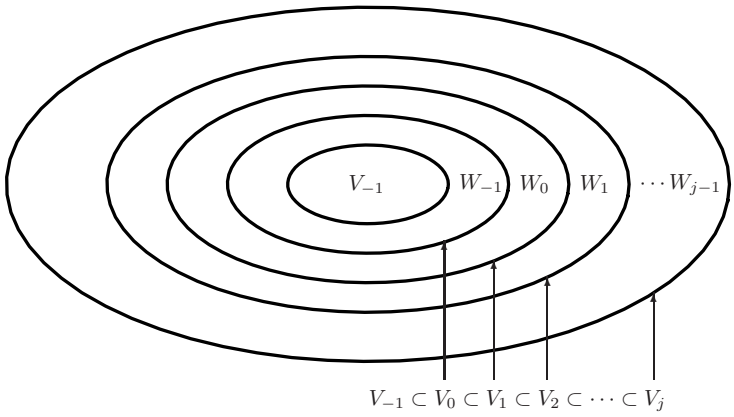


Figure 2.2: Nested subspaces spanned by scaling and wavelet functions.

a unit norm function $\phi(t)$. Every function $x(t) \in L^2(\mathcal{R})$ when mapped onto these subspaces, can be written as a linear combination of these basis functions as:

$$x(t) = \sum_{j=-\infty}^{\infty} \sum_{k=-\infty}^{\infty} a_{j,k}.2^{j/2}\phi(2^j t - k)$$

where,

$$a_{j,k} = \int_{-\infty}^{\infty} x(t).2^{j/2}\phi(2^j t - k)dt. \tag{2.1}$$

$\phi(t)$ is called the scaling function, and $a_{j,k}$ are the scaling coefficients. This however, is not an orthogonal expansion of $x(t)$ since subspaces V_j are nested. Also, the functions $\{2^{j/2}\phi(2^j t - k)|_{j,k\in\mathcal{Z}}\}$ are not orthogonal across scales, i.e.

$$< 2^{j/2}\phi(2^j t - k) \ , \ 2^{j'/2}\phi(2^{j'} t - k') > |_{k,k'\in\mathcal{Z}} \neq 0, \quad \text{when} \quad j \neq j',$$

and hence they do not form an orthogonal basis for $L^2(\mathcal{R})$. The nested subspaces implies that the subspace V_1 includes V_0. Thus $\phi(t)$ which resides in V_0, can be expressed in terms of the basis functions for V_1 as:

$$\phi(t) = \sqrt{2}\sum_k c(k)\phi(2t - k) \qquad \text{(Dilation equation)} \tag{2.2}$$

where $c(k)$ is the projection of $\phi(t)$ on the basis functions of V_1, and is given by

$$c(k) = \sqrt{2}\int_{-\infty}^{\infty} \phi(t)\phi(2t - k)dt.$$

Equation (2.2) is called the *Dilation Equation*.

To obtain an orthogonal decomposition of $L^2(\mathcal{R})$, we define "difference" spaces W_j as the complement of V_j in V_{j+1}. Figure 2.2 shows these difference spaces. Thus,

$$V_{j+1} = V_j \oplus W_j \qquad \text{and} \qquad V_j \cap W_j = \emptyset.$$

The W subspaces provide a decomposition of $L^2(\mathcal{R})$ into mutually orthogonal subspaces, and so,

$$W_j \perp W_{j'} \quad \text{if} \quad j \neq j', \quad \text{and} \quad \bigoplus_{j=-\infty}^{\infty} W_j = L^2(\mathcal{R}).$$

Subspace W_j is spanned by the family of basis functions $\{2^{j/2}\psi(2^j t - k)|_{j,k\in\mathcal{Z}}\}$, which are dyadic shifts and scales of the function $\psi(t)$. Every function $x(t) \in L^2(\mathcal{R})$ can be written as a linear combination of these basis functions:

$$x(t) = \sum_{j=-\infty}^{\infty} \sum_{k=-\infty}^{\infty} b_{j,k}.2^{j/2}\psi(2^j t - k) \qquad \text{(Synthesis)} \tag{2.3}$$

where,

$$b_{j,k} = \int_{-\infty}^{\infty} x(t).2^{j/2}\psi(2^j t - k)dt. \qquad \text{(Analysis)} \tag{2.4}$$

This is an orthogonal expansion of $x(t)$, since the subspaces W_j are orthogonal. $\psi(t)$ is called the wavelet function, and $b_{j,k}$ are the wavelet coefficients. Equations (2.3) and (2.4) together form the synthesis and analysis equations of the *Discrete Wavelet Transform* (DWT).

$\psi(t)$ resides in W_0, which in turn is a part of the V_1 subspace, and hence $\psi(t)$ can be expressed in terms of the basis functions of V_1 as:

$$\psi(t) = \sqrt{2}\sum_k d(k)\phi(2t - k) \qquad \text{(Wavelet equation)} \tag{2.5}$$

where $d(k)$ is the projection of $\psi(t)$ on the basis functions of V_1, and is given by

$$d(k) = \sqrt{2}\int_{-\infty}^{\infty} \psi(t)\phi(2t - k)dt.$$

Equation (2.5) is called the *Wavelet Equation*.

The nested subspaces in Figure 2.2 have basis functions derived from different *scales* of $\phi(t)$. Thus, mapping any function $x(t)$ on to these subspaces enables us to view this function at different *scales* or *resolutions*. At high scales (high values of j), wavelet coefficients ($b_{j,k}$) represent fine details of $x(t)$, while at low scales (low values of j) the wavelet coefficients measure coarser features of $x(t)$. This is the idea behind *multiresolution*.

From the way the subspaces W_j are defined, we have

$$\begin{aligned} V_M &= V_{M-1} \oplus W_{M-1} \\ &= V_{M-2} \oplus W_{M-2} \oplus W_{M-1} \\ &= V_{M-3} \oplus W_{M-3} \oplus W_{M-2} \oplus W_{M-1} \\ &= V_{M-n} \oplus W_{M-1} \oplus W_{M-2} \oplus W_{M-3} \oplus \cdots \oplus W_{M-n}. \end{aligned}$$

In general, every space V_M is the direct sum of $V_N|_{N<M}$ and $W_j|_{N\leq j<M}$:

$$V_M = V_N \oplus W_N \oplus W_{N+1} \oplus W_{N+2} \oplus \cdots \oplus W_{M-1}. \tag{2.6}$$

The DWT of any function $x(t) \in L^2(\mathcal{R})$, can have an infinite number of wavelet coefficients $b_{j,k}$, since the analysis equation (2.4) can be evaluated at an infinite number of scales j, and infinite number of shifts k. By virtue of equation (2.6) however, a function $x(t) \in V_M$ can be expressed in terms of the wavelet coefficients in the scales of interest ($N \leq j < M$), and the scaling coefficients at one coarse scale $N(< M)$ as follows:

$$x(t) = 2^{N/2} \sum_{k=-\infty}^{\infty} a_{N,k}\phi(2^N t - k) + \sum_{j=N}^{M-1} 2^{j/2} \sum_{k=-\infty}^{\infty} b_{j,k}\psi(2^j t - k). \tag{2.7}$$

If $x(t)$ is a finite duration signal, the limits on the shift variable k in the above expression are finite. Thus finite duration signals have a finite number of scaling and wavelet coefficients that define it.

2.1.2 Fast Wavelet Transform

Computation of the DWT involves computation of the integrals in equation (2.4). This is not a convenient method for computing the DWT. Mallat's fast wavelet transform (FWT) gives a discrete algorithm for computing the DWT coefficients using a two channel perfect reconstruction filter bank. The two channel filter bank shown in Figure 2.3 is used to compute the FWT. Filters $f[k]$ and $j[k]$ are related to the coefficients $c(k)$ and $d(k)$ in the dilation and wavelet equation by the relation:

$$c(k) = \sqrt{2}f[k] \qquad \text{and} \qquad d(k) = \sqrt{2}j[k].$$

Thus, the FIR low pass filter $f[k]$ corresponds to a finite duration scaling function $\phi(t)$, and the FIR high pass filter $j[k]$ corresponds to a finite duration wavelet function $\psi(t)$.

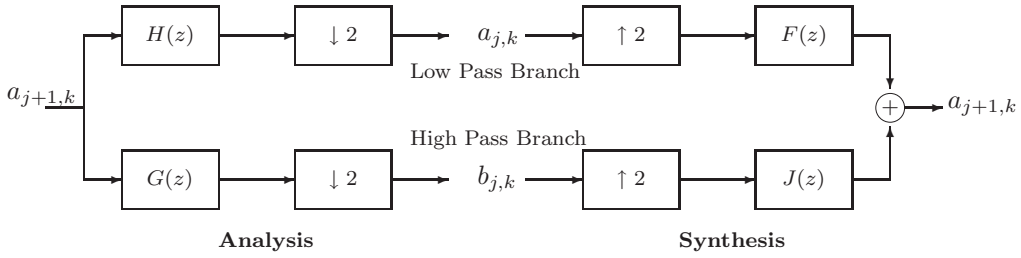


Figure 2.3: Fast wavelet transform using perfect reconstruction (PR) filter bank.

Given a function $x(t) \in V_{j+1}$, the scaling coefficients at scale $j+1$ are given by equation (2.1)

$$a_{j+1,k} = \int_{-\infty}^{\infty} x(t).2^{(j+1)/2}\phi(2^{j+1}t - k)dt.$$

When scaling coefficients $a_{j+1,k}$ are passed through the two channel filter bank shown in Figure 2.3, the filtering and downsampling operations yield the scaling and wavelet coefficients at scale j at the output of the lowpass and highpass branches. Thus,

$$a_{j,k} = \sum_l h(l-2k)a_{j+1,l} \qquad \text{and} \qquad b_{j,k} = \sum_l g(l-2k)a_{j+1,l}. \tag{2.8}$$

On the synthesis side, scaling and wavelet coefficients at scale j are upsampled and filtered to give the scaling coefficients at scale $j+1$:

$$a_{j+1,k} = \sum_l [f(k-2l)a_{j,l} + j(k-2l)b_{j,l}]. \tag{2.9}$$

Equations (2.8) and (2.9) together constitute the forward and inverse fast wavelet transform. Thus Mallat's algorithm establishes the relation between scaling coefficients ($a_{j+1,k}$) at a fine scale ($j+1$), and the scaling ($a_{j,k}$) and wavelet ($b_{j,k}$) coefficients at the next coarse scale (j). Passing $a_{j,k}$ through the analysis section of a filter bank again will yield the scaling and wavelet coefficients at the next coarse scale ($j-1$).

Thus starting with the scaling coefficients $a_{M,k}$ of a signal $x(t) \in V_M$, application of equation (2.8) $(M-N)$ times gives rise to a tree shaped filter bank structure that is $(M-N)$ levels deep, which yields wavelet coefficients at scales $N \leq j < M$, and the scaling coefficients at scale N, enabling a complete representation of the signal as shown in equation (2.7).

There are two methods of finding the scaling coefficients at the finest scale $a_{j+1,k}$ needed to start the FWT. First, $a_{j+1,k}$ can be computed using equation (2.4) by evaluating integrals. Second, $a_{j+1,k}$ can be set equal to $x[n]$ which is a sampled version of $x(t)$. This second method is commonly used to start the FWT.

2.1.3 Orthogonal and Biorthogonal DWT

In an orthogonal DWT filter bank, synthesis lowpass and highpass filters are related by alternating flip as follows:

$$j(n) = (-1)^n f(N-n) \quad \leftrightarrow \quad J(z) = -z^{-N} F(-z^{-1}).$$

The analysis filters are simply time reversals of the synthesis filters:

$$h(n) = f(-n) \quad \leftrightarrow \quad H(z) = F(z^{-1})$$

$$g(n) = j(-n) \quad \leftrightarrow \quad G(z) = J(z^{-1}).$$

Thus all filters in an orthogonal filter bank, can be defined by just one filter—the lowpass analysis filter $H(z)$.

An FIR filter bank cannot have both orthogonal and symmetric filters with length greater than 2. In image compression applications it is essential to have symmetric filters for efficient handling of image boundaries. Hence instead of using the orthogonal DWT, the biorthogonal DWT is used where the input function is mapped onto biorthogonal subspaces. Here, the wavelet function used in the analysis stage is different from the one used in the synthesis stage. The FIR filters are no longer orthogonal but they are symmetric, which is handy for image compression. The analysis and synthesis equations for the biorthogonal DWT of any $x(t) \in L^2(\mathcal{R})$ are

$$\tilde{a}_{j,k} = \int x(t) 2^{j/2} \tilde{\phi}(2^j t - k) dt, \qquad \tilde{b}_{j,k} = \int x(t) 2^{j/2} \tilde{\psi}(2^j t - k) dt \quad \text{(Analysis)}$$

$$x(t) = 2^{N/2} \sum_k \tilde{a}_{N,k} \phi(2^N t - k) + \sum_{j=N}^{M-1} 2^{j/2} \sum_k \tilde{b}_{j,k} \psi(2^j t - k). \quad \text{(Synthesis)}$$

$\{\phi_{i,j}(t), \psi_{i,j}(t)\}$ and $\{\tilde{\phi}_{i,j}(t), \tilde{\psi}_{i,j}(t)\}$ are the biorthogonal basis functions. $\tilde{a}_{j,k}$ and $\tilde{b}_{j,k}$ are the scaling and wavelet coefficients respectively; together they form the biorthogonal DWT coefficients of $x(t)$. The DWT starts at some finest scale $j = M$ and stops at some coarsest scale $j = N$. $M - N$ is the number of levels of decomposition in the biorthogonal DWT of $x(t)$.

Mallat's fast wavelet transform (FWT) can be used for computing the biorthogonal DWT coefficients. The lowpass analysis filter $h[n] \leftrightarrow H(z)$ corresponds to the analysis scaling function $\tilde{\phi}(t)$, the highpass analysis filter $g[n] \leftrightarrow G(z)$ corresponds to the analysis wavelet function $\tilde{\psi}(t)$, the lowpass synthesis filter $f[n] \leftrightarrow F(z)$ corresponds to the synthesis scaling function $\phi(t)$, the highpass analysis filter $j[n] \leftrightarrow J(z)$ corresponds to the synthesis wavelet function $\psi(t)$.

The forward and inverse biorthogonal FWT is given by:

$$\tilde{a}_{j,k} = \sum_l h(l - 2k) \tilde{a}_{j+1,l}, \qquad \tilde{b}_{j,k} = \sum_l g(l - 2k) \tilde{a}_{j+1,l} \qquad \text{(FWT)}$$

$$\tilde{a}_{j+1,k} = \sum_l [f(k-2l)\tilde{a}_{j,l} + j(k-2l)\tilde{b}_{j,l}]. \qquad \text{(IFWT)}$$

In a PR filter bank, the reconstructed scaling coefficients $\tilde{a}_{j+1,k}$ will be equal to the scaling coefficients input to the filter bank (to within a shift d), if the biorthogonal filters satisfy the following perfect reconstruction (PR) conditions:

$$F(z)H(z) + J(z)G(z) = 2z^{-d} \quad \text{(no-distortion)} \tag{2.10}$$

$$F(z)H(-z) + J(z)G(-z) = 0 \quad \text{(no-aliasing)} \tag{2.11}$$

In a biorthogonal filter bank, the no-aliasing PR condition is satisfied by design, by choosing the analysis and synthesis filters such that

$$\begin{aligned} g(n) = (-1)^n f(n) \quad &\leftrightarrow \quad G(z) = F(-z) \\ j(n) = -(-1)^n h(n) \quad &\leftrightarrow \quad J(z) = -H(-z). \end{aligned} \tag{2.12}$$

Since the highpass filters are related to the lowpass filters by equation (2.12), the no-distortion PR condition is determined by the frequency response of only the lowpass filters $H(z)$ and $F(z)$. Thus the biorthogonal filter bank is represented by two filters $H(z)$ and $F(z)$ (compared to just $H(z)$ in the orthogonal filter bank).

2.2 2-D Discrete Wavelet Transform

A 2-D separable discrete wavelet transform is equivalent to two consecutive 1-D transforms. For an image, a 2-D DWT is implemented as a 1-D row transform followed by a 1-D column transform. Transform coefficients are obtained by projecting the 2-D input image $x(u, v)$ onto a set of 2-D basis functions that are expressed as the product of two 1-D basis shown in equation (2.13).

$$\phi(u, v) = \phi(u)\phi(v)$$

$$\psi_1(u, v) = \psi(u)\phi(v)$$

$$\psi_2(u,v) = \phi(u)\psi(v)$$
$$\psi_3(u,v) = \psi(u)\psi(v) \tag{2.13}$$

The 2-D DWT (analysis) can be expressed as the set of equations (2.14). The scaling function $\phi(u,v)$ and wavelets $\psi_1(u,v)$, $\psi_2(u,v)$ and $\psi_3(u,v)$ (and the corresponding transform coefficients $X(N,j,m)$, $X^{(1)}(N,j,m)$, $X^{(2)}(N,j,m)$ and $X^{(3)}(N,j,m)$) correspond to different subbands in the decomposition. $X(N,j,m)$ are the coarse coefficients that constitute the LL subband. The $X^{(1)}(N,j,m)$ coefficients contain the vertical details and correspond to the LH subband. The $X^{(2)}(N,j,m)$ coefficients contain the horizontal details and correspond to the HL subband. The $X^{(3)}(N,j,m)$ coefficients represent the diagonal details in the image and constitute the HH subband. Thus a single-level decomposition at scale $(N+1)$ has four subbands of coefficients as shown in Figure 2.4.

$$\begin{aligned}
X(N,j,m) &= \int\int x(u,v)2^N\phi(2^Nu-j)\phi(2^Nv-m)dudv \Rightarrow LL \\
X^{(1)}(N,j,m) &= \int\int x(u,v)2^N\psi(2^Nu-j)\phi(2^Nv-m)dudv \Rightarrow LH \\
X^{(2)}(N,j,m) &= \int\int x(u,v)2^N\phi(2^Nu-j)\psi(2^Nv-m)dudv \Rightarrow HL \\
X^{(3)}(N,j,m) &= \int\int x(u,v)2^N\psi(2^Nu-j)\psi(2^Nv-m)dudv \Rightarrow HH
\end{aligned} \tag{2.14}$$

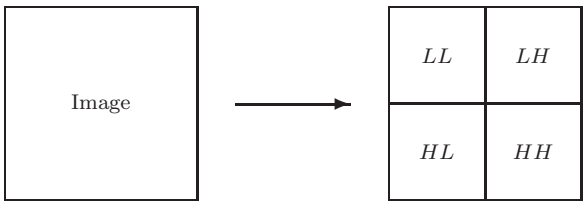


Figure 2.4: Single-level 2-D wavelet decomposition (analysis).

The synthesis bank performs the 2-D IDWT to reconstruct $x(u,v) \in V_M$. 2-D IDWT is given in equation (2.15).

$$x(u,v) = \sum_j \sum_m X(N,j,m)2^N\phi(2^Nu-j)\phi(2^Nv-m)$$

$$+\sum_{i=N}^{M-1}\sum_{j}\sum_{m}[X^{(1)}(i,j,m)2^i\psi(2^iu-j)\phi(2^iv-m)$$
$$+X^{(2)}(i,j,m)2^i\phi(2^iu-j)\psi(2^iv-m)$$
$$+X^{(3)}(i,j,m)2^i\psi(2^iu-j)\psi(2^iv-m)] \quad (2.15)$$

A single stage of a 2-D filter bank is shown in Figure 2.5. First, the rows of the input image are filtered by the highpass and lowpass filters. The outputs from these filters are downsampled by two, and then the columns of the outputs are filtered and downsampled to decompose the image into four subbands. The synthesis stage performs upsampling and filtering to reconstruct the original image.

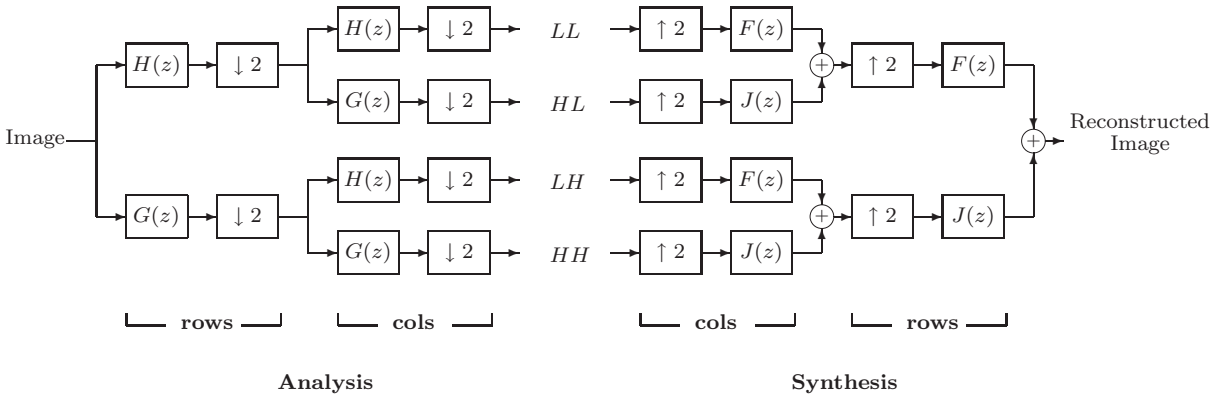


Figure 2.5: One level filter bank for computation of 2-D DWT and IDWT.

Multiple levels of decomposition are achieved by iterating the analysis stage on only the LL band. For l levels of decomposition, the image is decomposed into $3l+1$ subbands. Figure 2.6 shows the filter bank structure for three levels of decomposition of an input image, and Figure 2.7 shows the subbands in the transformed image.

Figure 2.8 shows the three-level decomposition of the image "Lighthouse" using the biorthogonal 9/7 wavelet. Subband decomposition separates high frequency details in the image (edges of fence, grass, telescope) from the low frequency information (sky, wall of the lighthouse). Vertical details appear in the LH^* subbands, horizontal details appear in HL^* subbands and diagonal details can be found in the HH^* subbands.

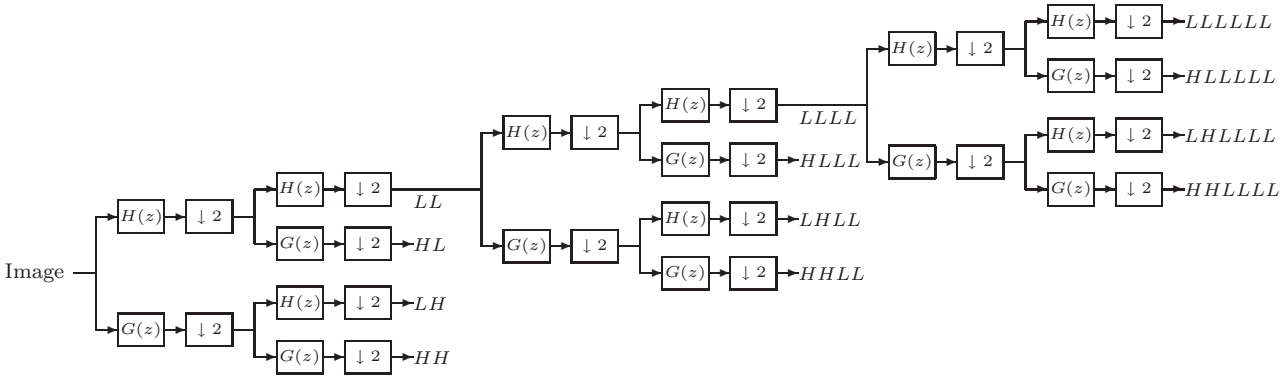


Figure 2.6: Analysis section of a three level 2-D filter bank.

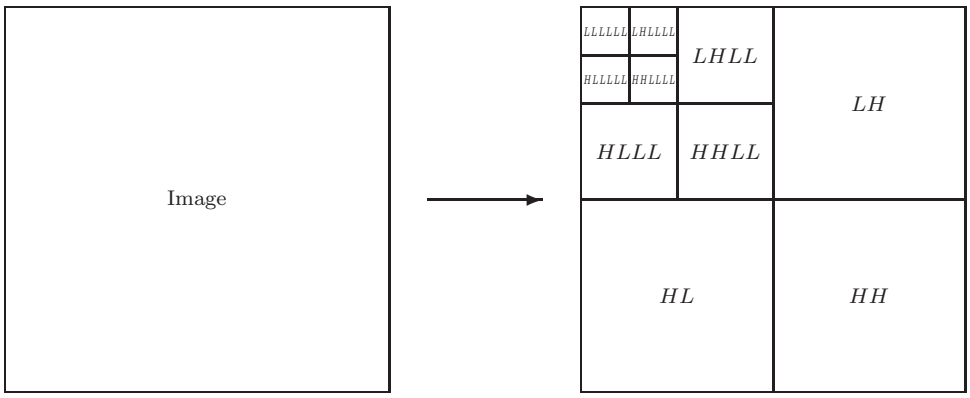


Figure 2.7: Three level wavelet decomposition of an image.

The histograms for coefficients in all subbands shown in Figure 2.8, illustrate the advantage of transform coding. The original 512×512 lighthouse image had few zero pixel values. The three level DWT of the image separated the energy content in the image into ten non-overlapping frequency subbands. There are still 512×512 DWT coefficients, but most of these coefficients (especially in the HL^*, LH^* and HH^* subbands) are zero, or have values very close to zero. These coefficients can be efficiently coded in the quantizer and entropy coder, resulting in significant compression. The coefficient distribution for the $LLLLLL$ subband resembles that of the original image. This is indicative of the fact that the $LLLLLL$ band is just an approximation of the original image at a coarser resolution. Coefficients in the $LLLLLL$ band have the highest variance (energy) suggesting that in this image most of the energy is contained at low frequencies. This indeed is the case for most natural images.

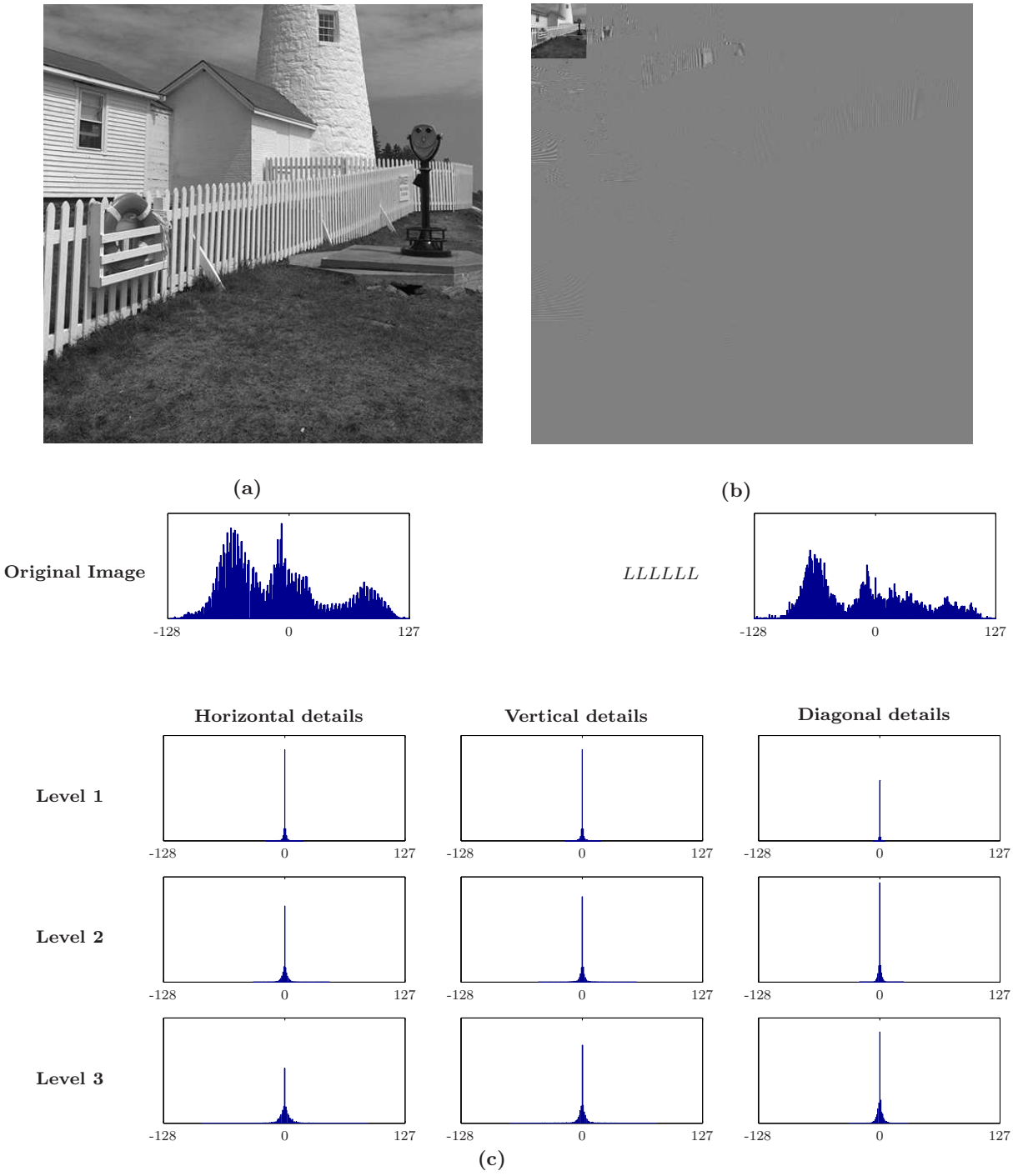


Figure 2.8: Three level decomposition of 'lighthouse' using biorthogonal 9/7.

2.3 Non-expansive DWT and Symmetric Extension

Consider one level of a filter bank as shown in Figure 2.3. If the input signal has length L, and the filters $H(z)$ and $G(z)$ have lengths L_h and L_g respectively, then after downsampling, the length of the highpass and lowpass coefficients are $(L + L_h - 1)/2$ and $(L + L_g - 1)/2$ respectively. The total length of the output sequence is $L - 1 + (L_h + L_g)/2$; the number of output samples is more than the number of input samples. This is the *expansive* DWT. For multiple levels of decomposition, the number of output samples keeps increasing. For example, a 5-level expansive DWT of a 512×512 image using the biorthogonal 9/7 wavelet with $L_h = 9$ and $L_g = 7$ results in about 5.5% more DWT samples compared to image samples. This is not desirable for compression, since we are looking to reduce the number of samples to be encoded.

One way to avoid this is to use circular convolution in place of linear convolution for computing the output of the filters. With circular convolution, an input of length L convolved with a filter of length L_h ($< L$), will give an output of length L. After downsampling, the total length of the output samples (highpass and lowpass) will be equal to L resulting in a *non-expansive* transform. However, circular convolution, which is equivalent to periodic extension of the input signal, results in aliasing at the boundaries of the input signal, resulting in border artifacts. For lossy compression these artifacts are undesirable because:

1. they require more bits to code; and,
2. they do not represent any real information present in the input signal, instead they are spurious features resulting from the method used to perform the transform.

The bits used to code this artifact could be better used coding information present in the original image.

To avoid artifacts resulting from periodic extension, the input signal is *symmetric-periodically* (or just *symmetrically*) extended at the borders. When no compression is desired, the original

signal can be reconstructed perfectly from its non-expansive DWT coefficients when the input signal is symmetrically extended and the wavelet filters used to compute the DWT coefficients are symmetric (biorthogonal wavelets).

It has been shown that for image compression, the non-expansive DWT with symmetric extension outperforms non-expansive DWT with periodic extension in terms of subjective quality [25]. Thus image compression algorithms like JPEG2000, compute a 2-D non-expansive DWT of the input image by symmetrically extending the rows and columns of the input and filtering using biorthogonal filters.

2.4 Hardware Implementation of the 2-D DWT

2.4.1 Implementation Choices

A wavelet based image compression system will have a 2-D DWT core implemented on FPGA or as an application specific integrated circuit (ASIC). A typical 2-D DWT core will consist of a single level 2-D filter bank, memory modules for storing image and DWT data, and control logic for routing and scheduling. A single level 2-D filter bank may be implemented in one of the following three ways:

1. Time Domain Convolution (TDC): This approach derives its name from the fact that the filtering operation is implemented in the time domain using convolution. The lowpass and highpass filter coefficients are quantized for implementation in hardware. Quantized values of these coefficients influence the performance of the image compression system. There are two ways of implementing TDC—(a) lowpass and highpass filters of the biorthogonal wavelet are directly implemented in hardware; and (b) analysis polyphase matrix of the biorthogonal wavelet is factored to obtain a sequence of short "lift" and "update" filters, which are then implemented in hardware. The second method (called *lifting*) has the advantage of an orthogonal analysis and synthesis

structure, which makes it relatively immune to filter coefficient quantization. TDC is an intuitive approach to implementation of the filter bank, and has the advantage of being computationally inexpensive.

2. Fast Fourier Transform (FFT): This approach computes the output of the lowpass and highpass branches of the filter bank in the frequency domain. For symmetric extension, the input data is "folded" about its last sample giving a symmetric signal that has twice the number of samples as the input signal. DFTs of the filters ($H(\omega)$ and $G(\omega)$) are multiplied by a DFT of the symmetrically extended input data ($X_{sym}(\omega)$), and an IDFT of the product is computed to obtain the circularly convolved outputs. $H(\omega)$ and $G(\omega)$ can be pre-computed and quantized for hardware implementation, while $X_{sym}(\omega)$ needs to be computed for all input signals using FFT. Compared to TDC, FFT operates on signals that have double the number of samples. This, added to the fact that the FFT method requires implementation of the FFT, IFFT algorithms and involves complex multiplications, makes it computationally expensive and infeasible.

3. Time Domain Matrix (TDM): This approach implements the filtering operation in time domain using matrix operations. The symmetrically extended input signal is left-multiplied with a sparse matrix, whose rows are comprised of dyadic circular shifts of the lowpass and highpass filter coefficients. Filter coefficients need to be quantized for hardware implementation similar to TDC. The TDM method offers the advantage of improved performance over TDC and FFT for orthogonal wavelets due to an additional step in the synthesis stage that recovers samples lost to maintain a non-expansive transform [26]. For biorthogonal wavelets however, TDM offers the same performance as TDC and FFT, without the need for the additional step to recover lost samples. Implementation of a biorthogonal filter bank in hardware using TDM is as computationally expensive as TDC.

We choose TDC for implementation of the filter bank, due to its simplicity and low computational complexity. Direct and lifting forms of TDC are evaluated for performance after

coefficient quantization.

2.4.2 Figures of Merit

Several important figures of merit are used to measure the performance and efficiency of an image compression system implemented in hardware. *Peak signal-to-noise ratio (PSNR)* is a commonly used quantitative measure that compares the fidelity of a compressed image with its original. High PSNR values indicate compressed images that are very similar to the original images. For an 8-bit grayscale $R \times C$ image x, the PSNR of a compressed image $\hat{x}$ is computed in decibels using the relation

$$PSNR = 10 \log_{10} \left(\frac{255^2}{mse} \right)$$

where,

$$mse = \frac{1}{RC} \sum_{r=0}^{R-1} \sum_{c=0}^{C-1} |x(r,c) - \hat{x}(r,c)|^2.$$

Objective measures such as PSNR do not indicate *subjective quality* at low PSNR values and high compression ratios. Subject evaluation of a reconstructed image requires visual inspection of the image for artifacts, and measures perceptual quality of the reconstructed image.

Several metrics are used to evaluate hardware performance. The first is *hardware cost*. Filters in the TDC implementation are multiplierless, and all multiplications are performed using shifts and additions after approximating each filter coefficient as a sum or difference of powers of two (SPT). Before a filter is implemented, hardware cost is estimated in terms of T, the total number of non-zero terms used when writing all the filter coefficients in SPT format. T is related to hardware cost because it represents the number of terms that must be added to perform the multiplication; the hardware for multiplication has $T - 1$ word adders. A subset of SPT representation called canonical signed digit (CSD) form is used for representation of coefficients. No two adjacent bits are non-zero in CSD form. The

canonical form is optimal in terms of how many non-zero digits are required to represent a given number. For example, the CSD form of -7 has two terms, $0\overline{1}001$ ($-7 = -8+1$), rather than three terms, $0\overline{111}$ ($-7 = -4-2-1$)[1]. On average, a canonical representation uses two-thirds as many non-zero digits as a binary representation, so the CSD form can significantly reduce the number of adders required in hardware. However, for some numbers there may be another form that is just as good as the CSD form; +3, for example, can be written using two non-zero digits either canonically, $010\overline{1}$ ($3 = 4-1$), or non-canonically, 0011 (3=$2+1$). In such cases, the CSD form is used unless another form has the same number of non-zero digits and is better in the sense that the terms to be added together have fixed point formats that match more closely (so that the word adders used for the addition need not be as wide).

Once a filter is implemented in hardware, hardware cost is measured directly in terms of the *number of logic cells used*. *Throughput*, or the number of data outputs per second that the filter can generate relates to how long it takes to process an image. *Latency*, or the number of clock cycles that it takes from the time that an input enters the system to the time that the corresponding output exits the system, is also an important measure of hardware performance. For remote, untethered applications where battery power is at a premium, *energy* consumed by the hardware to process an image also becomes critical.

2.5 JPEG2000

The JPEG2000 standard uses the scalar wavelet transform for image compression compared to the discrete cosine transform (DCT) used by JPEG. Part I of the JPEG2000 standard supports LeGall 5/3 wavelet for reversible (lossless) compression, and the Cohen, Daubechies and Feauveau (CDF) 9/7 wavelet (commonly known as biorthogonal 9/7) for irreversible (lossy) compression [1]. The symmetry of the biorthogonal 9/7 filters and the fact that they

[1]A $\overline{1}$ in the binary or SPT representation of any value indicates that the corresponding power of two should be negated.

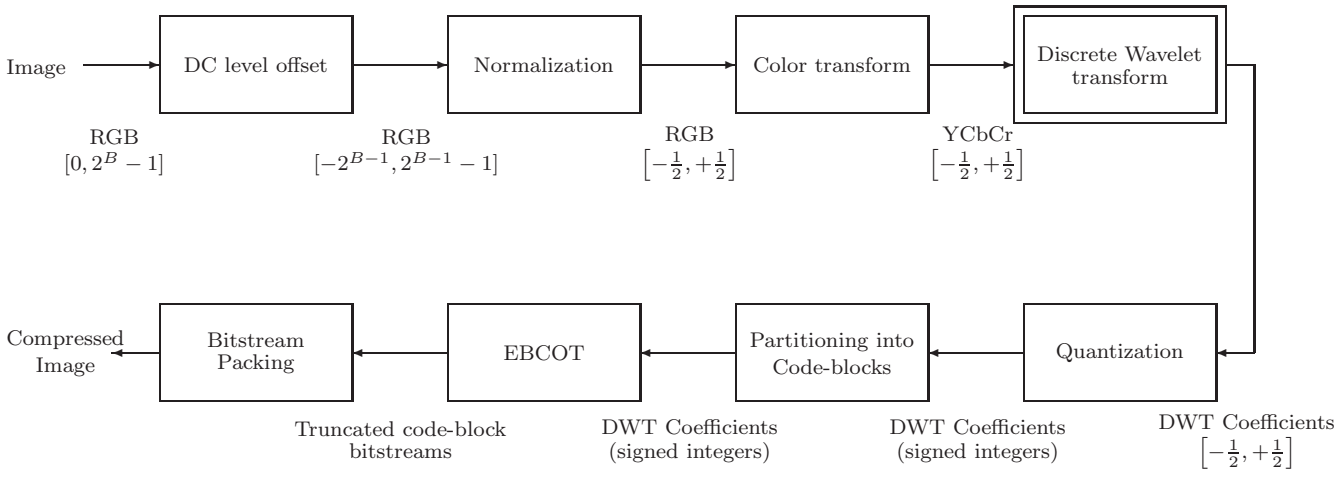


Figure 2.9: Block diagram of the JPEG2000 coding algorithm.

are "nearly" orthogonal [27], makes it a good candidate for image compression applications. Figure 2.9 shows the block diagram of a JPEG2000 encoder. The input image undergoes DC level-shifting and normalization so that all pixels lie within the range $[-\frac{1}{2}, \frac{1}{2}]$. The image is then transformed from the RGB domain to the YCbCr domain. 2D DWT of each plane (luminance, chrominance-blue and chrominance-red) is then computed. The EBCOT (Embedded Block Coding with Optimal Truncation) algorithm is then used to code the DWT coefficients.

In this thesis we study the DWT block of the JPEG2000 encoder. TDC implementations of the biorthogonal 9/7 discrete wavelet transform on field programmable gate arrays (FPGAs) are examined. The DC gain of the filters used in this thesis is different from the gain of the filters specified in the JPEG2000 standard. This however is a minor difference and can be addressed by simple scaling of the DWT coefficients by powers of two after they have been computed.

The next chapter examines filter structure and coefficient quantization issues for direct implementation of TDC.

Chapter 3

Filter Structure and Quantization

Filter coefficients are quantized before implementation in fast hardware; this changes the frequency response of the filter. When the quantized filter is part of a filter bank, this change impacts perfect reconstruction properties of the filter bank, and degrades image compression performance. If the filter coefficients are quantized such that the filter bank properties are preserved, then the degradation in compression performance will be minimal. Filter structure also plays an important role in the performance of the filter bank in the presence of coefficient quantization. Some filter structures are more immune to coefficient quantization than others and can be used to minimize performance degradation in the filter bank. This chapter derives and investigates two filter structures and two "compensating" filter coefficient quantization methods for improving the performance of multiplierless, quantized filter banks. Rather than using an optimization technique to guide the design process, the new methods utilize the perfect reconstruction requirements of the filter bank. The design method is easily extended to any biorthogonal perfect reconstruction filter bank with finite length filters.

The performance of a hardware implementation of a biorthogonal filter bank depends on how well the two PR conditions —equations (2.10) and (2.11), and the linear phase characteristic are preserved. Linear phase and the no-aliasing condition can be preserved by ensuring that each filter remains symmetric after coefficient quantization. The magnitude response

Table 3.1: Unquantized biorthogonal 9/7 wavelet lowpass filter coefficients.

n	$h_9(n)$	$f_7(n)$
0	0.03782845550726	-0.06453888262870
1	-0.02384946501956	-0.04068941760916
2	-0.11062440441844	0.41809227322162
3	0.37740285561283	0.78848561640558
4	0.85269867900889	0.41809227322162
5	0.37740285561283	-0.04068941760916
6	-0.11062440441844	-0.06453888262870
7	-0.02384946501956	
8	0.03782845550726	

of the lowpass and highpass branches of the filter bank determine whether the no-distortion condition is satisfied.

The unquantized nine-tap lowpass analysis filter of the biorthogonal 9/7 wavelet, $H_9(z)$, has four zeros at $z = -1$ and four complex zeros in the right-half z-plane; the symmetry and realness of $h_9(n)$ dictate that these eight zeros occur in complex conjugate and reciprocal pairs. Similarly, the unquantized seven-tap lowpass synthesis filter, $F_7(z)$, has four zeros at $z = -1$ and two real (reciprocal) zeros in the right-half z-plane. The unquantized lowpass filter (LPF) coefficients, $h_9(n)$ and $f_7(n)$, are listed in Table 3.1 [28]; these coefficients are normalized such that their sum equals $\sqrt{2}$.

The filters can be expressed in their factored form as

$$\begin{aligned} H_9(z) &= k_9(z+1)^4(z-z_1)(z-\frac{1}{z_1})(z-z_1^*)(z-\frac{1}{z_1^*}) \\ F_7(z) &= k_7(z+1)^4(z-z_2)(z-\frac{1}{z_2}) \end{aligned}$$

where the zero $z_1 = 0.28409629819231 + j0.24322822591087$ dictates the placement of the quartet of complex conjugate zeros of reciprocal magnitude in the right hand plane for $H_9(z)$, and the zero $z_2 = 0.32887591778548$ dictates the placement of the pair of real zeros of reciprocal magnitude in the right hand plane for $F_7(z)$. The filter gains are $k_9 = 0.03782845550726$ and $k_7 = -0.06453888262870$.

There are three primary structures for FIR filters: direct, cascade and lattice. The symmetry in the biorthogonal 9/7 filters results in at least one lattice coefficient equal to one; thus, the lattice structure cannot be employed. This chapter examines the direct and cascade implementations of the 9/7 filters.

3.1 Direct Implementation

Direct form implementations of the two LPFs require that the sixteen coefficients be quantized. The total number of SPT used (T) must be kept to a minimum in order to achieve hardware efficiency and speed. The challenge is to allocate a given T across the coefficients such that compression performance is maximized.

The first method begins with a mostly uniform distribution of T across the sixteen coefficients; most coefficients are allotted the same number of SPT terms, and the remaining SPT terms are allocated to those coefficients that are most different (in terms of percent different) from their unquantized values. This kind of allocation is referred to as *mostly uniform allocation*. Performance of the filter bank is then improved, not by investigating different ways to optimally divvy up T, but by exploiting the relationship between the quantized filters and the no-distortion perfect reconstruction (PR) condition.

3.1.1 Direct Form without Gain Compensation

The first method uses the mostly uniform allocation scheme. Table 3.2 depicts one quantization of the two LPFs for $T = 32$; the quantized filters are denoted by $H_9'(z)$ and $F_7'(z)$. The distribution of T is done so as to maintain symmetry. In this direct form, there is no change to the original filter gains (beyond the slight change in gain that is inevitable when the coefficients are quantized), so no separate gain factor K is required.

The no-distortion PR condition, equation (2.10), involves the sum of the cascaded LPFs and

Table 3.2: Quantized filter coefficients for the direct implementation with $T = 32$ (with and without gain compensation). Values are shown in both decimal and SPT format.

	Direct without Gain Compensation				Direct with Gain Compensation			
	$H'_9(z)$		$F'_7(z)$		$H'_9(z)$		$F'_7(z)$	
0	0.03515625	0.000010010	-0.0625	$0.000\overline{1}0$	0.03515625	0.00001001	-0.0625	$0.000\overline{1}0000$
1	-0.0234375	$0.00000\overline{11}00$	-0.03125	$0.0000\overline{1}$	-0.0234375	$0.0000\overline{1}001$	-0.03125	$0.0000\overline{1}000$
2	-0.1171875	$0.00\overline{1}000100$	0.4375	$0.100\overline{1}0$	-0.125	$0.00\overline{1}00000$	0.4375	$0.100\overline{1}0000$
3	0.376953125	0.011000001	0.78125	0.11001	0.375	0.01100000	0.78515625	0.11001001
4	0.8125	0.110100000	0.4375	$0.100\overline{1}0$	0.8125	0.11010000	0.4375	$0.100\overline{1}0000$
5	0.376953125	0.011000001	-0.03125	$0.0000\overline{1}$	0.375	0.01100000	-0.03125	$0.0000\overline{1}000$
6	-0.1171875	$0.00\overline{1}000100$	-0.0625	$0.000\overline{1}0$	-0.125	$0.00\overline{1}00000$	-0.0625	$0.000\overline{1}0000$
7	-0.0234375	$0.00000\overline{11}00$			-0.0234375	$0.0000\overline{1}001$		
8	0.03515625	0.000010010			0.03515625	0.00001001		
K	-		-		-		-	-
G	-		-		-		1.0166015625	1.0000010001

cascaded HPFs. Since the HPFs are related to the LPFs by equation (2.12), PR can be evaluated in terms of only the LPFs. The implication is that better compression performance can be expected when the product of the quantized LPFs, $H'_{LPB}(z) = H'_9(z)F'_7(z)$, more closely resembles the product of the unquantized LPFs, $H_{LPB}(z) = H_9(z)F_7(z)$. (Here the subscript LPB stands for the "Low Pass Branch" of the filter bank which loosely comprises the cascade of the analysis and synthesis LPFs (see Figure 2.3)). Furthermore, since the preponderance of energy in most images is concentrated at low frequencies, a close resemblance between $|H'_{LPB}(\omega)|$ and $|H_{LPB}(\omega)|$ is more critical in the passband, and in particular at $\omega = 0$ (DC). This observation forms the basis for the second design method.

3.1.2 Direct Form with Gain Compensation

In this method, a few SPT terms are reserved for a compensating gain factor G, the sole purpose of which is to match $|H'_{LPB}(\omega)|$ to $|H_{LPB}(\omega)|$ at DC. First, the unreserved SPT terms are allocated using the mostly uniform technique. A compensating gain factor is then

computed; since $|H_{LPB}(\omega)|_{\omega=0}$ is 2, the compensating gain is given by

$$G = \frac{2}{|H'_{LPB}(\omega)|_{\omega=0}}.$$

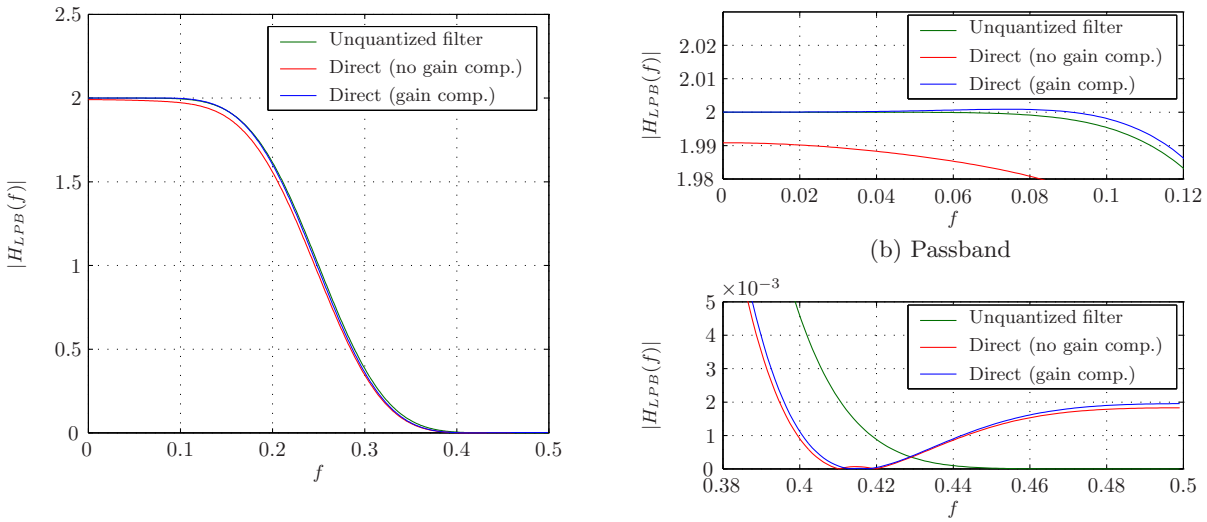
The compensating gain G is quantized using the reserved SPT terms, and incorporated with the $F'_7(z)$ filter. The result is that the shape of the quantized magnitude response is retained, but its values at low frequencies are much closer to the unquantized magnitude response. Figure 3.1 shows a close-up of the direct form with "gain compensation" $|H'_{LPB}|$ at low and high frequencies. $|H'_{LPB}|$ approximates $|H_{LPB}|$ closely in the passband; however, the deviation from zero at $\omega = \pi$ is more pronounced with the compensating gain factor.

Table 3.2 lists the direct form with "gain compensation" quantized coefficients for $T = 32$. The coefficients of the two methods are not always identical; three SPT terms must be reserved for the compensating gain, so the "with gain compensation" case can only allocate twenty-nine SPT terms to the coefficients.

Figure 3.2 shows the pole zero plots for the unquantized and direct form quantized LPFs with and without "gain compensation". In a direct form quantized filter, the location of each zero depends on all of the coefficients; so for both of these direct form methods, all of the zeros are perturbed from their original locations. (The zeros still occur as complex conjugates and reciprocals since symmetry and realness are preserved after quantization). In particular, the four zeros in the left hand plane are no longer at $z = -1$. This translates into DC leakage of the synthesis HPF ($J(z)$ in Figure 2.3) and non-smooth synthesis functions. Consequently, images compressed and reconstructed using the direct form implementation (with or without gain compensation) will depict the checkerboarding artifact.

3.2 Cascade Implementation

Consider the cascade form for $H'_9(z)$ and $F'_7(z)$ where each filter is the cascade of two sections: section one captures the four zeros at $z = -1$ and section two captures the remaining zeros.



(a) (b) Passband (c) Stopband

Figure 3.1: (a) Comparison of magnitude response of LPB using unquantized, original filters and $T = 32$ direct implementation with and without gain compensation. (b) and (c) zooms in on the passband and stopband respectively.

In this form, the 9/7 wavelet lowpass filters can be written as

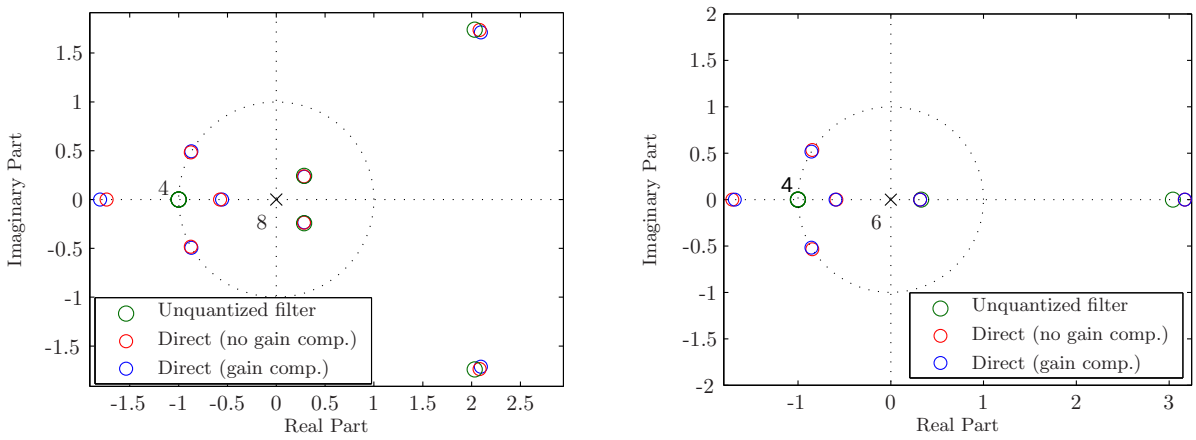
$$H_9(z) = k_9(z^4 + 4z^3 + 6z^2 + 4z + 1) \cdot (z^4 + a_1 z^3 + a_2 z^2 + a_1 z + 1) \tag{3.1}$$

$$F_7(z) = k_7(z^4 + 4z^3 + 6z^2 + 4z + 1) \cdot (z^2 + a_3 z + 1) \tag{3.2}$$

where k_9 and k_7 are as before and

$$\begin{aligned} a_1 &= -(z_1 + z_1^* + \frac{1}{z_1} + \frac{1}{z_1^*}) = -4.63046362055955, \\ a_2 &= (z_1 z_1^* + \frac{1}{z_1 z_1^*} + \frac{z_1^*}{z_1} + \frac{z_1}{z_1^*} + 2) = 9.59748437683150, \\ a_3 &= -(z_2 + \frac{1}{z_2}) = -3.36953637943799. \end{aligned}$$

This structure preserves symmetry in all of the sections, and offers the advantage of separating out the four zeros at $z = -1$ into their own section, so that they are unaffected by quantization in the rest of the filter. Serendipitously, only six SPT terms are required to implement the four zeros at $z = -1$ *exactly*; the five coefficients are $\{1, 4, 6, 4, 1\}$. The following four design methods use this cascade structure with a fixed first section.



(a) $H_9(z)$ (b) $F_7(z)$

Figure 3.2: Comparison of location of zeros for the direct form quantized LPFs with and without gain compensation. (a) 9-tap $H_9(z)$ and (b) 7-tap $F_7(z)$.

3.2.1 Cascade Form without Gain Compensation

For this method, six SPT terms are reserved for the first section exactly implementing the four zeros at $z = -1$. The coefficients of the second section are normalized so that its first and last coefficients are exactly "1", thereby requiring only one SPT term each. Because of this normalization a quantized, normalizing gain factor, K, must be included—and a few SPT terms are reserved for this purpose. All remaining SPT terms are allocated to the remaining coefficients in the second sections using the mostly uniform scheme.

The normalizing gain factors are quantized to $k_9' = 0.0390625 = 2^{-5} + 2^{-7}$ and $k_7' = -0.0625 = -2^{-4}$. The analysis LPF requires $T = 2$ for its normalizing gain factor. The synthesis LPF normalizing gain can be implemented as a simple shift; it requires no SPT terms and no real hardware. Thus, the gain and section one coefficients require $T = 8$ in $H_9'(z)$ and $T = 6$ in $F_7'(z)$ (refer to Table 3.3).

The quantized values of the three remaining coefficients, a_1', a_2' and a_3', determines performance of the filter bank. The cascade form without gain compensation uses uniform distribution of the remaining T across these three coefficients. Table 3.3 lists one quantiza-

Table 3.3: Quantized filter coefficients for the cascade implementation with $T = 32$ (with and without gain compensation). Values are shown in both decimal and SPT format.

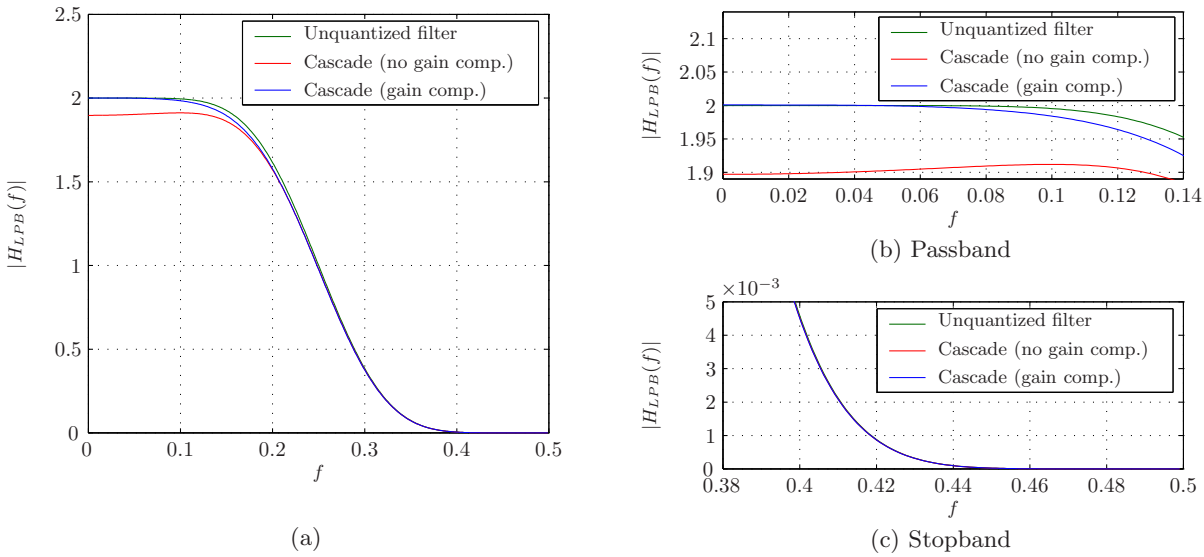
	Cascade without Gain Compensation				Cascade with Gain Compensation			
	$H_9'(z)$		$F_7'(z)$		$H_9'(z)$		$F_7'(z)$	
Section 1	1	001	1	001	1	001	1	001
	4	100	4	100	4	100	4	100
	6	110	6	110	6	110	6	110
	4	100	4	100	4	100	4	100
	1	001	1	001	1	001	1	001
Section 2	1	0001.0000	1	01.0000	1	0001.0	1	01.00
	-4.625	$0\overline{1}00.\overline{1}0\overline{1}0$	-3.3125	$\overline{11}.0\overline{1}0\overline{1}$	-4.5	$0\overline{1}00.\overline{1}$	-3.25	$\overline{11}.0\overline{1}$
	9.5625	$1010.\overline{1}001$	1	01.0000	9.5	$1010.\overline{1}$	1	01.00
	-4.625	$0\overline{1}00.\overline{1}0\overline{1}0$			-4.5	$0\overline{1}00.\overline{1}$		
	1	0001.0000			1	0001.0		
K	0.0390625	0.0000101	-0.0625	(simple shift)	0.0390625	0.0000101	-0.0625	(simple shift)
G	-		-		-		1.0244140625	1.0000011001

tion of the two LPFs for $T = 32$ for this method. Symmetry is maintained in both filters and, unlike the direct implementation, the zeros remain at $z = -1$ (refer to Figure 3.4).

3.2.2 Cascade Form with Gain Compensation

For this method, as before, six SPT terms are reserved for each of the first sections. Again, the quantized, normalizing gain factor, K, is included—two SPT terms are reserved for this purpose.

As in the "direct form with gain compensation" method, the compensating gain factor G is computed, the purpose of which is to ensure a better fit of frequency response at DC. G is quantized and incorporated with the $F_7'(z)$ filter. Figure 3.3 compares the magnitude responses of the cascade form with and without "gain compensation". For both implementations, $|H_{LPB}'|$ is indistinguishable from the unquantized magnitude response at high frequencies due to the unperturbed zeros at $z = -1$ (Figure 3.4). At low frequencies and especially around DC, "gain compensation" provides a close approximation to the unquantized



(a)

(b) Passband

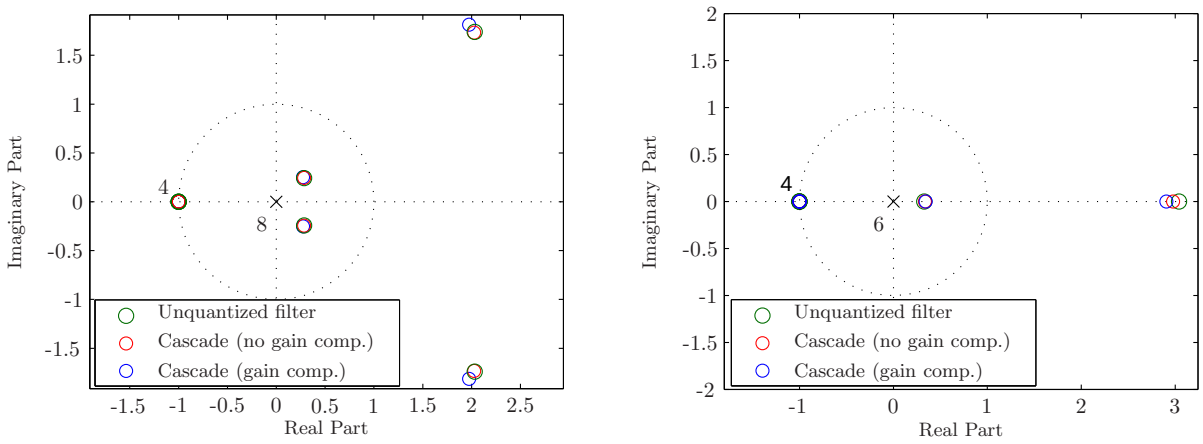
(c) Stopband

Figure 3.3: (a) Comparison of magnitude response of LPB using unquantized, original filters and $T = 32$ cascade implementation with and without gain compensation. (b) and (c) zooms in on the passband and stopband respectively.

magnitude response. Although cascade with "gain compensation" closely matches $|H_{LPB}|$ at DC and very low frequencies, the deviation in the passband for $\omega \in (0.4, 0.8)$ is larger than in the direct form with "gain compensation". Table 3.3 shows these cascade form with "gain compensation" quantized coefficients for $T = 32$.

3.2.3 Compensating Zeros

Instead of using a compensating gain factor, the perfect reconstruction (PR) conditions can be used to directly to obtain quantized filters that give better performance for the same T. Now, the values of a'_1, a'_2 and a'_3 shape $|H'_{LPB}|$ instead of G. There are two possible design methods based on this observation. Both rely on the fact that in the cascaded implementation, a'_1 and a'_2 determine the location of the quartet of zeros in the right hand plane of the low pass analysis filter $H'_9(z)$, while a'_3 determines the location of the pair of zeros in the right hand plane of the low pass synthesis filter $F'_7(z)$.



(a) $H_9(z)$ (b) $F_7(z)$

Figure 3.4: Comparison of location of zeros for the cascade form quantized LPFs with and without gain compensation.

Cascade Form With z_2 Compensation. Here, a_1 and a_2 are first quantized to a_1' and a_2'; correspondingly, the zero z_1 moves to z_1' (with the other three zeros of the complex quartet moving accordingly). This fixes the frequency response of the quantized lowpass analysis filter $H_9'(z)$. It is possible to choose a new location z_2' for z_2 such that the deviation of $|H_{LPB}'|$ from the unquantized magnitude response is minimized. For example, if z_1' is closer to the unit circle than z_1, then to compensate for this effect in the cascaded magnitude response, z_2' should be chosen such that it is further from the unit circle than z_2.

The unquantized and quantized magnitude responses of the LPB of the filter bank are given by

$$H_{LPB}(z) = \hat{H}(z)(z^4 + a_1 z^3 + a_2 z^2 + a_1 z + 1) \cdot (z^2 + a_3 z + 1) \tag{3.3}$$

$$H_{LPB}'(z) = \hat{H}'(z)(z^4 + a_1' z^3 + a_2' z^2 + a_1' z + 1) \cdot (z^2 + a_3' z + 1) \tag{3.4}$$

where $\hat{H}(z) = k_9 k_7 (z+1)^8$ and $\hat{H}'(z) = k_9' k_7' (z+1)^8$. a_3' is the new value of a_3 that places z_2' at a compensating location. Ideally, the LPB quantized and unquantized transfer functions should be equal; i.e.

$$H_{LPB}'(z) = H_{LPB}(z). \tag{3.5}$$

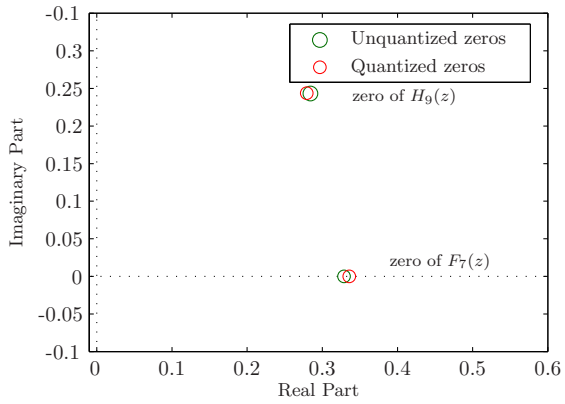


Figure 3.5: Illustration of zero movement for $T = 32$ cascade form with z_1 compensation. As z_2 (the real zero belonging to $F_7(z)$) moves closer to the unit circle after quantization of a_3, the compensating location for z_1 (the zero belonging to $H_9(z)$) is further away from the unit circle. Unquantized vector lengths: $|z_1| = 0.37399288298886$, $|z_2| = 0.32887591778548$. Quantized vector lengths: $|z_1'| = 0.37066793950141$, $|z_2'| = 0.33596061410765$.

Since $k_9'k_7' \approx k_9k_7$, this equality becomes

$$\begin{aligned}(z^4 + a_1z^3 + a_2z^2 + a_1z + 1)(z^2 + a_3z + 1)\\ = (z^4 + a_1'z^3 + a_2'z^2 + a_1'z + 1)(z^2 + a_3'z + 1).\end{aligned} \tag{3.6}$$

Solving equation (3.6) at $\omega = 0$ ($z = 1$), gives the compensating value of a_3'. This value is then quantized with the remaining available T.

Cascade Form With z_1 Compensation. Alternatively, a_3 can be first quantized to a_3', so that z_2 moves to z_2' (with its reciprocal zero moving accordingly). Now, z_1' does the compensating; Figure 3.5 illustrates that as z_2 moves closer to the unit circle after quantization of a_3, the compensating location for z_1 is further away from the unit circle. Equation (3.6) is solved for two unknowns: a_1' and a_2', by choosing two frequencies at which H_{LPB}' should match H_{LPB}. In addition to $\omega = 0$, a passband frequency closer to the passband edge, $\omega = 0.2\pi$, is chosen. The computed a_1' and a_2' are quantized with the remaining available SPT terms. If unquantized values of a_1', a_2' and a_3' are used, "z_1 compensation" will provide a more accurate $|H_{LPB}'|$ than "z_2 compensation" since "z_1 compensation" fits $|H_{LPB}|$ at two

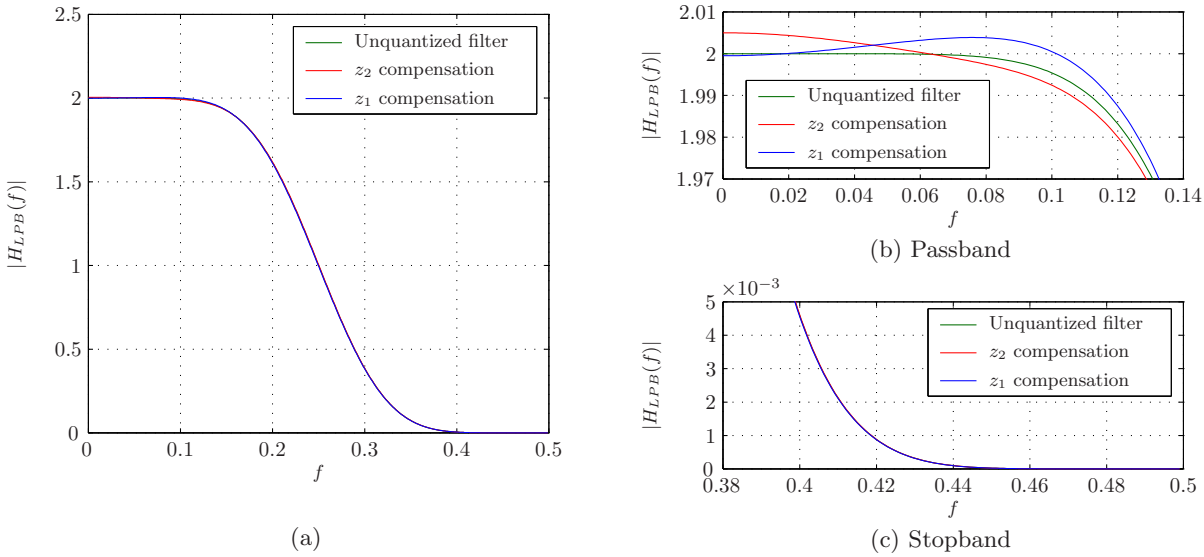


Figure 3.6: (a) Comparison of magnitude response of LPB using unquantized, original filters and $T = 32$ cascade implementation with z_2 and z_1 compensation. (b) and (c) zooms in on the passband and stopband respectively.

frequencies instead of one.

Figure 3.6 illustrates the effect of using compensating zeros; a close-up shows how "z_1 compensation" and "z_2 compensation" attempt to match $|H_{LPB}|$ in the passband. The deviation of both curves from the unquantized magnitude response at the matching frequencies occurs because of the subsequent quantization of the compensating coefficients. Because the zeros at $z = -1$ remain fixed in the cascade structure, both methods generate magnitude responses indistinguishable from the unquantized magnitude response at high frequencies. Table 3.4 lists these cascade form "z_1 compensation" and "z_2 compensation" quantized coefficients for $T = 32$.

Direct Form Revisited. This compensating zeros method could also be used on the direct form implementation. For instance, the seven coefficients in $F_7'(z)$ could be quantized and the nine compensating coefficients in $H_9'(z)$ could be computed. This would involve the solution of five linear equations resulting from the selection of five frequencies at which to fit $|H_{LPB}'|$ to $|H_{LPB}|$. This ability to fit the magnitude responses at five frequencies would

Table 3.4: Quantized filter coefficients for the cascade implementation with $T = 32$ (two compensating-zero cases). Values are shown in both decimal and SPT format.

	Cascade with z_2 Compensation				Cascade with z_1 Compensation			
	$H_9'(z)$		$F_7'(z)$		$H_9'(z)$		$F_7'(z)$	
Section 1	1	001	1	001	1	001	1	001
	4	100	4	100	4	100	4	100
	6	110	6	110	6	110	6	110
	4	100	4	100	4	100	4	100
	1	001	1	001	1	001	1	001
Section 2	1	0001.000	1	001.00000000	1	0001.0000	1	01.0000
	-4.625	$0\bar{1}00.\bar{1}0\bar{1}$	-3.42578125	$\bar{1}00.10010011$	-4.625	$0\bar{1}00.\bar{1}0\bar{1}0$	-3.3125	$\bar{1}\bar{1}.0\bar{1}0\bar{1}$
	9.5	$1010.\bar{1}00$	1	001.00000000	9.6875	$1010.0\bar{1}0\bar{1}$	1	01.0000
	-4.625	$0\bar{1}00.\bar{1}0\bar{1}$			-4.625	$0\bar{1}00.\bar{1}0\bar{1}0$		
	1	0001.000			1	0001.0000		
K	0.0390625	0.0000101	-0.0625	(simple shift)	0.0390625	0.0000101	-0.0625	(simple shift)
G	-		-		-		-	

seem like it ought to provide superior performance over the cascade form with compensating zeros. However, the cascade form will still outperform the direct form for two reasons. First, the four zeros will no longer be at $z = -1$ in the quantized direct form (with or without compensation). Second, there is no expectation that any of the direct form quantized coefficients will be convenient from the SPT perspective (like several of the coefficients in the cascade form). For example, in the $T = 32$ cascade "z_1 compensation" case, twenty-two SPT terms are allocated before compensation, leaving ten SPT terms to represent the three compensating coefficients ($\frac{10}{3} = 3.33$ SPT term per coefficient). However, in the $T = 32$ direct "without gain compensation" case, eleven SPT terms are allocated to $F_7'(z)$, leaving twenty-one SPT terms to represent the nine compensating coefficients ($\frac{21}{9} = 2.33$ SPT term per coefficient)—one full SPT term less per coefficient than in the cascade case. Thus, the quantization of the compensating coefficients will be more severe for the direct implementation for a given T. Simulations showed that this "direct with compensating zeros" approach resulted in PSNR values 4-5dB less than the "direct with gain compensation" method which, in turn, is always outperformed by the "cascade with z_1 compensation"

method.

Extensions to Other Biorthogonal Wavelets. The compensating zeros method for quantizing filter banks can be extended to any biorthogonal filter bank. First, the analysis and synthesis LPFs are each divided into two cascaded sections: the first section is comprised of the zeros at $z = -1$ and the second section is comprised of the remaining zeros in the filter. Keeping the complex conjugate and reciprocal zeros together in the second cascade section ensures that the filters are real and symmetric. Next, the coefficients in the second section of the shorter of the two filters (analysis LPF and synthesis LPF) is quantized. Finally, equation (3.5) is solved for the compensating coefficients of the second section of the longer filter. Equation (3.5) is solved at as many points on the unit circle as there are unknown symmetric coefficients in the second cascade section of the longer filter.

3.3 Results

The DWT of three 8-bit grayscale images was computed using the two-channel filter bank where each set of the six quantized filter coefficients in Tables 3.2, 3.3 and 3.4 were used, in turn, to replace the unquantized, biorthogonal 9/7 wavelet filter coefficients. A non-expansive, five-level transform with symmetric extension was computed; SPIHT was employed for quantizing the transform coefficients and no entropy coder was used [29]. The images were examined for perfect reconstruction (i.e. a 1:1 compression ratio) and at four compression ratios (8:1, 16:1, 32:1, and 64:1).

3.3.1 Compression Performance

PSNR Performance. Table 3.5 compares the performance of the six implementations in terms of peak signal-to-noise ratio (PSNR); these results were generated in software on a general purpose personal computer. Each method utilized a total of $T = 32$ SPT terms

Table 3.5: PSNR comparison of the six quantization methods (all with $T = 32$). Numbers shown for the quantized filters indicate PSNR degradation with respect to unquantized PSNR values. Negative numbers indicate PSNR performance worse than the unquantized filter, while positive numbers indicate PSNR performance better than the unquantized filters.

Image	Compression Ratio	Unquantized	Quantized					
			Direct		Cascade		Cascade	
			no gain	with gain	no gain	with gain	z_2 comp.	z_1 comp.
Lena	1:1	58.61	-22.58	-9.65	-35.59	-14.11	-9.83	-1.55
	8:1	39.83	-5.50	-1.12	-16.87	-1.13	-0.46	-0.08
	16:1	36.71	-3.69	-0.82	-13.84	-0.58	-0.22	-0.01
	32:1	33.59	-2.23	-0.55	-10.90	-0.34	-0.13	-0.03
	64:1	30.54	-1.25	-0.38	-8.16	-0.14	-0.06	+0.04
Barbara	1:1	58.64	-23.52	-10.77	-36.72	-15.05	-11.12	-2.18
	8:1	35.75	-3.54	-0.72	-13.98	-0.51	-0.29	+0.00
	16:1	30.82	-1.46	-0.22	-9.38	-0.30	-0.09	-0.11
	32:1	27.04	-0.77	-0.26	-6.23	-0.14	-0.06	-0.07
	64:1	24.36	-0.32	-0.13	-4.29	-0.11	-0.03	-0.06
Fingerprint	1:1	58.92	-24.08	-11.84	-37.73	-16.05	-11.58	-1.55
	8:1	35.27	-3.36	-0.55	-14.22	-0.78	-0.24	-0.16
	16:1	30.39	-1.20	+0.04	-9.62	-0.08	+0.24	+0.18
	32:1	26.63	-0.65	-0.18	-6.46	-0.11	+0.01	-0.06
	64:1	23.53	-0.20	-0.08	-4.22	-0.03	+0.00	+0.01

— PSNR degradation < 0.2dB
— 0.2dB < PSNR degradation < 2dB
— 2dB < PSNR degradation

for the two quantized LPFs $H_9'(z)$ and $F_7'(z)$. For all compression ratios and all images, the direct form with gain compensation outperforms the direct form without gain compensation; similarly for the cascade form with and without gain compensation. The two compensating zeros methods exhibit the best performance in terms of PSNR. The z_1 compensation method outperforms the z_2 compensation method at 1:1 and 8:1, but not at the higher compression ratios. This is due to the z_1 method's ability to fit $|H_{LPB}|$ at two frequencies instead of one. The cascade with z_1 compensation method resulted in PSNR values within 0.16dB of the unquantized PSNR values across all three images and all four compression ratios.

These results also indicate that the PSNR differences between the methods are larger at lower compression ratios and smaller at higher compression ratios. This is because the quantization of the filter coefficients becomes relatively less important compared to the quantization of the transform coefficients at the lower bit rates.

Subjective Performance. Although the PSNR results indicate relatively similar performance at low bit rates across the four best methods (e.g. differences of 0.01–0.55dB at 32:1 for direct with gain, cascade with gain, cascade with z_2 compensation and cascade with z_1 compensation), the subjective results exhibit important differences. The cascade form implementation will, by design, always maintain the four zeros at $z = -1$ while the direct form implementation will not. This translates into DC leakage of the synthesis HPF $J(z)$ and non-smooth synthesis functions. Consequently, images compressed and reconstructed using the direct form implementation (with or without gain compensation) depict the checkerboarding artifact. Consider the Goldhill image compressed at 32:1 using the unquantized filters in Figure 3.7(a) (PSNR=30.13dB). When the same image is compressed at 32:1 using the direct with gain compensation quantized filters, Figure 3.7(b), the checkerboard is easily observed at the top of the image; however, the PSNR is not significantly degraded (29.90dB). The cascade with gain compensation and cascade with z_1 compensation methods generate reconstructed images that are indistinguishable from the unquantized filter reconstruction at 32:1 (for example, see Figure 3.7(c) for z_1 compensation compression, PSNR=30.00dB).

Just as spurious visual characteristics are sometimes not translated into significant decreases in PSNR, sometimes large differences in PSNR are not evident in subjective evaluations. For instance, the cascade without gain compensation PSNR results in Table 3.5 are substantially lower than all of the other filters; however, the reconstructed image looks very similar to the other filters—with one minor difference. Although the edges and features of the reconstructed image are virtually identical to the other synthesized images, the cascade without gain compensation reconstruction looks "washed out" (i.e. lighter or over-exposed) as can be seen in Figure 3.7(d). $|H'_{LPB}|$ for the cascade without gain compensation method has a similar shape to the unquantized magnitude response (no distortion), but it underestimates

(a)
Unquantized filters;
(PSNR=30.13dB).

(b)
Quantized direct filters with gain compensation;
(PSNR=29.90dB).

(c)
Quantized cascade filters with z_1 compensation;
(PSNR=30.00dB).

(d)
Quantized cascade filters without gain compensation;
(PSNR=21.52dB).

Figure 3.7: Goldhill compressed at 32:1 using different $T = 32$, quantized filters.

Table 3.6: Filter property comparison of the six implementations ($T = 32$).

Filter Property	Unquantized	Quantized					
		Direct		Cascade		Cascade	
		no gain	with gain	no gain	with gain	z_2 comp.	z_1 comp.
Deviation at DC	0	9.2e-3	-3.8e-5	1.0e-1	-8.1e-4	-5.0e-3	4.9e-4
No Distortion MSE	0	4.0e-3	5.1e-4	5.9e-3	9.1e-4	1.4e-5	2.8e-5
Alias Cancellation MSE	0	0	0	0	0	0	0
Accuracy	4/4	0/0	0/0	4/4	4/4	4/4	4/4
Smoothness (analysis)	1.41	N/A	N/A	1.40	1.53	1.37	1.46
Smoothness (synthesis)	2.12	N/A	N/A	2.08	2.04	2.16	2.08
Coding Gain	9.71	10.57	9.86	11.21	10.09	9.67	9.69

the ideal response (lower gain).

Quantized Filter Properties. Table 3.6 illustrates that the deviation at DC for cascade without gain compensation is worse by at least two orders of magnitude than the other methods. (positive values for the deviation at DC indicate underestimation). Conversely, significant negative values for deviation at DC resulted in reconstructed images that were darker than the originals.

Of all the properties described in Table 3.6, the no distortion property best distinguishes good PSNR performance (the deviation at DC is part of this property). Moreover, the combination of the no distortion and synthesis smoothness properties best predicts good subjective quality. (In Table 3.6 , the no distortion and alias cancellation properties describe the filters' ability to meet the two perfect reconstruction conditions, equations (2.10)-(2.11)); accuracy is given by the number of zeros at $z = -1$ in the LPFs; smoothness is evaluated by examining the eigenvalues of the autocorrelation matrices of the LPFs [30]; and, the coding gain is calculated using the unified coding gain estimate in [31].

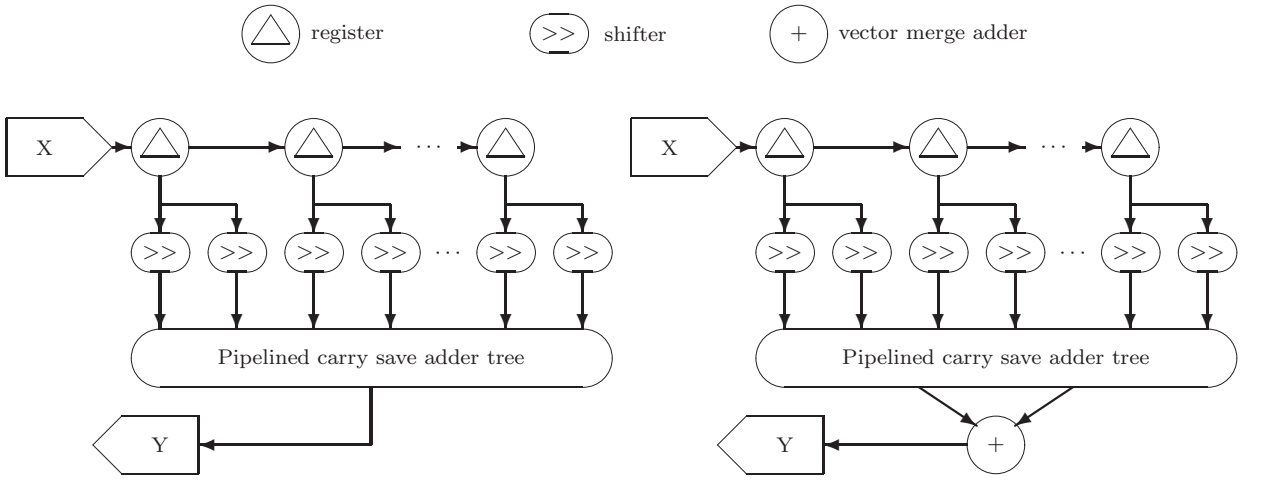


Figure 3.8: Direct form hardware architectures for the FIR filter.

3.3.2 Hardware Performance

The hardware performance of the filters was evaluated by synthesizing the filters for an Altera FPGA. The experimental procedure is as follows. Given the desired SPT coefficients (from Tables 3.2, 3.3, and 3.4), filter synthesis software, written in C, generates a synthesizable VHDL description of the filter. VHDL was written in a structural style, and care was taken to ensure that hardware features such as carry chains are fully exploited. The Altera Quartus II version 2.1 software package is used for logic synthesis, placement and routing on an EP20K1000EFC672-1X Altera FPGA, a member of the APEX family. The filter designs are verified by comparing the output of a Quartus simulation of the implemented filter with the expected output from Matlab. The Quartus software is used to analyze critical path delays, to determine hardware size, measured here in terms of number of Altera logic cells, and to estimate power consumption for a sample set of stimuli that include 500 random patterns, along with 120 other patterns useful for verification. All energy estimates are done at a common operating frequency of 66.67MHz as described in Appendix A.

Filters were synthesized with enough precision on intermediate signals so that no bits of

information are ever lost. Fixed-point, two's complement adders are used, but the number of integer and fraction bits for each hardware adder are chosen so that no round-off or overflow can occur. Thus, the fixed-point hardware gives precisely the same results that infinite precision hardware would give *given the quantized coefficients.* A major advantage of this approach is that all hardware implementation effects are fully considered by analyzing the effects of quantizing the coefficients; there is no need to do a numerical analysis of the computations themselves. Data formats for signals are represented using the notation (n, l), where n is the total number of bits, including the sign bit, and l is the weight of the least significant bit. For example, if a signal has format $(12, -2)$, it has twelve bits, with the least significant bit in the 2^{-2}'s place, and so it has ten bits of integer and two bits of fraction.

In creating a hardware implementation of a direct or cascade filter, consideration must be given to the hardware architecture. Figure 3.8 shows the architectural variations considered. In all cases, a chain of registers is used to shift the data in, and the data is shifted in accordance with the filter coefficients before being summed. For example, if one of the filter coefficients were $0.46875 = 2^{-2} - 2^{-5}$, the corresponding data word would go to two shifters and be shifted two and five places, respectively, before being summed. The main difference between the variations is in how the summing is done. The first variation, shown in part (a) of Figure 3.8, uses a pipelined tree of ripple carry adders (RCAs). This tree has depth that grows logarithmically with T, the number of terms to be added (and the number of non-zero digits in the SPT representations of all the filter coefficients); depth is $\lceil \log_2(T) \rceil$, and plays a role in the latency of the filter. The delay of a ripple carry adder grows linearly with the bit width of the adder, and the throughput of the RCA tree depends on the bit width of the widest ripple carry adder in the tree.

The other architectural variations replace the RCA tree with a pipelined tree of carry save adders (CSAs), as shown in part (b) of Figure 3.8. Carry save addition propagates two separate partial results, rather than one complete sum. Thus, each carry save adder (CSA) in the tree takes three data words in, and produces two words at the output (denoted a sum word and a carry word). This tree's depth is $\lceil \log_{1.5}(T) \rceil$. By splitting the sum and carry

information into two separate words to be dealt with independently, the carry save adder has the highly desirable property that its delay is simply the delay of a full adder, independent of the bit width of the signals being added. This means that the throughput of the CSA tree is high, and independent of the bit widths of the intermediate signals. The fact that the CSA tree is deeper than a similar RCA tree means that these architectural variations have higher latency than the first, but latency is easily tolerated, and less important than throughput, for image processing applications.

The CSA tree produces two outputs that must be summed, or "vector-merged", using conventional techniques to produce the final filter output. For the second architectural variation, a simple ripple carry adder (RCA) is used for the vector merge. Here, the vector merge adder is the slowest element in the system, and therefore the throughput of the entire system will depend on the delay of this adder. The third architectural variation uses a parallelized adder instead to increase the throughput; a number of RCAs, operating round-robin style, work in parallel to better keep up with the high throughput of the CSA tree. The result is a system with higher throughput, but also higher hardware cost.

Direct filters with no gain compensation are implemented exactly as shown in Figure 3.8; for cascade filters, each section is implemented as shown, and the sections are cascaded together. Gains, whether for a direct or cascade filter, are implemented as an additional cascaded section, as though a gain were an additional filter with a single coefficient.

Table 3.7 compares the three architectural variations for a single filter design, the $T = 32$ direct filters without gain compensation, whose coefficients are shown in Table 3.2. All variations produce precisely the same mathematical results, and give the same PSNR when used for image compression. From the table, it is easy to see that the variations provide a trade-off between hardware cost and hardware performance. The architecture with a simple RCA tree is small, with low latency, but relatively low in throughput. Using a CSA tree with a parallelized vector merge adds significantly to size and latency, but also significantly improves throughput. These comments apply in general, and so in comparing the filter

Table 3.7: Comparison of three different hardware architectures for $T = 32$ direct filters without gain compensation. Filter coefficients are shown on the left-hand side of Table 3.2.

	RCA tree		CSA tree with RCA vector merge		CSA tree with parallelized vector merge	
	$H_9'(z)$	$F_7'(z)$	$H_9'(z)$	$F_7'(z)$	$H_9'(z)$	$F_7'(z)$
number SPT terms	21	11	21	11	21	11
size (logic cells)	574	307	648	349	850	465
latency (clock cycles)	5	4	8	6	12	10
input data format	(8,0)	(8,0)	(8,0)	(8,0)	(8,0)	(8,0)
output data format	(18,-9)	(14, -5)	(18,-9)	(14, -5)	(18,-9)	(14, -5)
clock rate (MHz)	68.68	102.72	107.55	114.98	199.16	204.12
energy (mJ)	6.37		6.14		6.78	

Table 3.8: Comparison of hardware metrics for design methods for $T = 32$. All filters use a CSA tree with an RCA vector merge.

	Direct no gain		Direct with gain		Cascade no gain		Cascade with gain		Cascade z2 comp.		Cascade z1 comp.	
	$H_9'(z)$	$F_7'(z)$	$H_9'(z)$	$F_7'(z)$	$H_9'(z)$	$F_7'(z)$	$H_9'(z)$	$F_7'(z)$	$H_9'(z)$	$F_7'(z)$	$H_9'(z)$	$F_7'(z)$
SPT terms	21	11	17	15	20	12	17	15	19	13	20	12
size (logic cells)	648	349	519	473	677	387	554	563	669	470	690	387
latency (clocks)	8	6	7	9	13	10	13	14	13	10	13	10
input format	(8,0)	(8,0)	(8,0)	(8,0)	(8,0)	(8,0)	(8,0)	(8,0)	(8,0)	(8,0)	(8,0)	(8,0)
output format	(18,-9)	(14,-5)	(17,-8)	(27,-18)	(20,-11)	(17,-8)	(18,-8)	(25,-16)	(19,-10)	(21,-12)	(21,-11)	(17,-8)
clock rate (MHz)	107.55	114.98	103.57	78.31	86.25	128.75	97.97	71.05	86.50	77.82	94.02	128.75
energy (mJ)	6.15		6.18		6.34		6.45		6.44		6.36	

design methods, results are presented for the middle variation (CSA tree with an RCA vector merge), keeping in mind that the other variations can be used to meet lower cost or higher throughput requirements.

Table 3.8 shows the hardware performance for the quantized 9/7 LPFs, synthesized using the six proposed design methods. All filters share $T = 32$ SPT terms between the low pass analysis filter $H_9'(z)$ and the low pass synthesis filter $F_7'(z)$. The cascade form filters are slightly larger than the direct form filters, and have higher latency; this is to be expected, because breaking a filter into sections implies a larger total number of filter coefficients, and therefore more registers. Adding up the number of logic cells required for $H_9'(z)$ and $F_7'(z)$, the two direct form systems require 997 and 992 logic cells, respectively, while the cascade

form systems require between 1064 and 1139 logic cells. All filters take in inputs of data format (8,0), i.e., eight bit two's complement integers. They vary in terms of their output data formats, depending on the precision of the coefficients used; for example, output formats for $F_7'(z)$ range from fourteen bits for the "direct without gain compensation" method, for which the coefficients span only five bit positions, to twenty-five bits for the "cascade with gain compensation" method, which has a gain that alone spans eleven bit positions. A larger span implies internal signals of larger bit width, since no information is lost due to truncation or round off. The clock rate results also indicate system throughput, since the filter takes in one input and produces one output at every clock. The results show that clock rate is largely related to the bit widths of the output signal (with some variation attributable to differences in routing choices made by the Quartus synthesis software); this is to be expected, since the critical path in the architecture is through the vector merge adder, and its delay is proportional to the output bit width. The cascade form filters also require more energy than the direct form filters to compute the DWT. This is a consequence of the higher number of logic elements and wider output bit widths in the cascade form filters.

3.4 Summary

This chapter addressed the filter structure and filter coefficient quantization design issues for a fast, multiplierless hardware implementation of the traditional (convolution based) DWT filter bank. A cascade form implementation is the better choice compared to the the direct form implementation since it leaves the LPF zeros at $z = -1$ unperturbed, avoiding the checkerboarding artifact in compressed images. Two compensation techniques—"gain compensation" and "zero compensation"—were developed to reduce degradation of the quantized filter bank properties. The "z_1 compensation" technique yields the best filter bank performance in terms of compression performance. From a hardware perspective, the direct form implementations were slightly smaller, had lower power consumption and had smaller latency than the cascade form implementations. Among the cascade form filters, z_1-compensated

filters resulted in the fastest hardware with low energy consumption.

The next chapter seeks to improve the performance of "z_1 compensation" by examining a polyphase implementation.

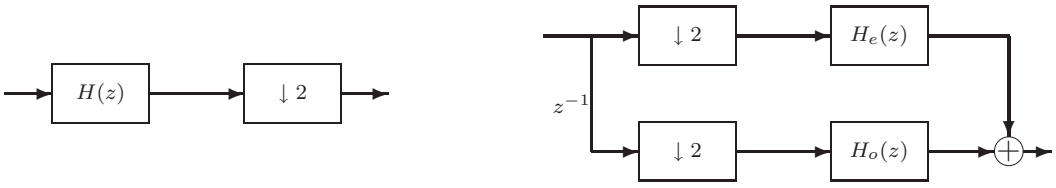
Chapter 4

Polyphase Implementation

The TDC based filter bank structure as shown in Figure 2.3 is inefficient in terms of data throughput, regardless of whether the filters are implemented in the direct form or as a cascade of sections. In the analysis stage, the downsampling operation follows the filtering, and half the samples just computed by the filters are discarded. Thus, the output rate of the analysis stage is half the input rate. In the synthesis stage, the upsampling operation precedes the filtering operation. Here, the filters operate at double the input rate; however, because upsampling inserts zeros in the input data, at any given time, half of the multipliers in the synthesis filters are multiplying by zero. Thus, half of the mathematical operations are wasted. The filter bank in Figure 2.3 is referred to as a *non-polyphase structure*. This chapter presents a *polyphase structure* that can avoid wasted operations and double filter bank throughput.

4.1 Polyphase Structures

The basic block of the analysis side of a filter bank is a filter $h[n] = (h_0, h_1, h_2, h_3, h_4, h_5, ...) \leftrightarrow H(z)$ followed by a downsampling operation, as shown in Figure 4.1a. Separating the odd



(a) Non-polyphase structure

(b) Polyphase structure

Figure 4.1: Direct form filter, analysis side.

and even powers in the transfer function we have

$$\begin{aligned} H(z) &= h_0 + h_1 z^{-1} + h_2 z^{-2} + h_3 z^{-3} + h_4 z^{-4} + ... \\ &= (h_0 + h_2 z^{-2} + h_4 z^{-4} + ...) \\ &\quad + z^{-1}(h_1 + h_3 z^{-2} + h_5 z^{-4} + ...) \\ &= H_e(z^2) + z^{-1} H_o(z^2) \end{aligned} \tag{4.1}$$

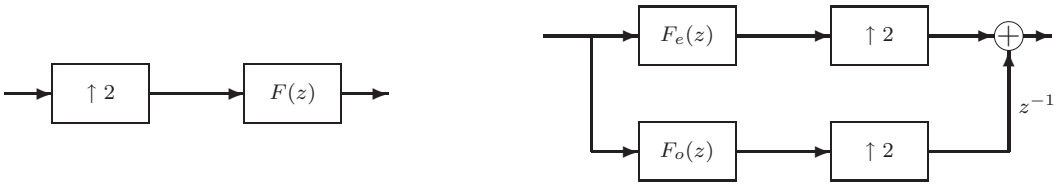
where

$$\begin{aligned} H_e(z) &\leftrightarrow (h_0, h_2, h_4, ...) \quad \text{(even phase of } h[n]) \\ H_o(z) &\leftrightarrow (h_1, h_3, h_5, ...) \quad \text{(odd phase of } h[n]) \ . \end{aligned}$$

Once the even and odd powers have been separated, the order of the downsampling operation and the filter can be exchanged, according to the first Noble identity [30]. The resulting polyphase filter structure, shown in Figure 4.1b, is computationally equivalent to the original structure in Figure 4.1a, but much more efficient: rather than filtering the full input stream, and then downsampling the results, it downsamples the input stream so that only the necessary computations are done.

A similar transformation applies to the synthesis side of the filter bank. The synthesis filter $f[n] = (f_0, f_1, f_2, f_3, f_4, ...) \leftrightarrow F(z)$ is divided into its even- and odd-indexed coefficients:

$$\begin{aligned} F_e(z) &\leftrightarrow (f_0, f_2, f_4, ...) \quad \text{(even phase of } f[n]) \\ F_o(z) &\leftrightarrow (f_1, f_3, f_5, ...) \quad \text{(odd phase of } f[n]) \ . \end{aligned}$$



(a) Non-polyphase structure (b) Polyphase structure

Figure 4.2: Direct form filter, synthesis side.

Again, the resulting polyphase structure is equivalent to, but with twice the throughput of, the original structure. Both structures are shown in Figure 4.2.

The polyphase structure can be applied to both direct and cascade filter banks. The following subsections describe its application to the direct structure, to the cascade structure, and to a hybrid direct-cascade structure.

4.1.1 Direct Polyphase Structure

For a polyphase, direct implementation of the biorthogonal 9/7 filter bank, each filter is split into its even and odd phases. Each phase will still be symmetric. Applying the polyphase idea does not change the coefficients at all, so a direct form polyphase implementation (direct-poly) requires the same total number of SPT terms as the direct form without polyphase (direct-no-poly). The advantage of a direct-poly implementation, is that the throughput is doubled with no significant change in hardware cost. The direct-poly and direct-no-poly structures produce precisely the same data outputs, and therefore have the same effect on image compression quality. In particular, this implies that the direct-poly implementation will still have the same problems as the direct-no-poly implementation: the zeros at $z = -1$ will not be preserved, and reconstructed images will exhibit the checkerboard artifact.

The direct-poly structure of one stage of the filter bank in Figure 2.3 is shown in Figure 4.3.

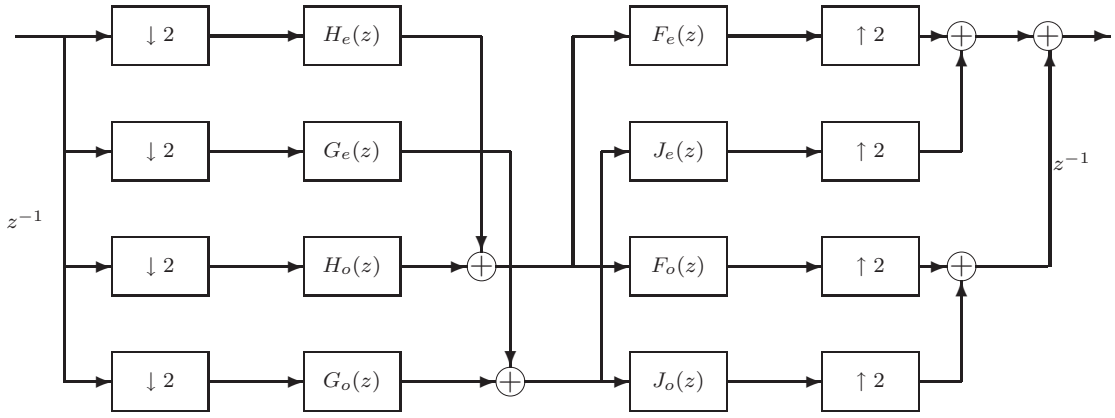


Figure 4.3: Direct-poly structure for the filter bank.

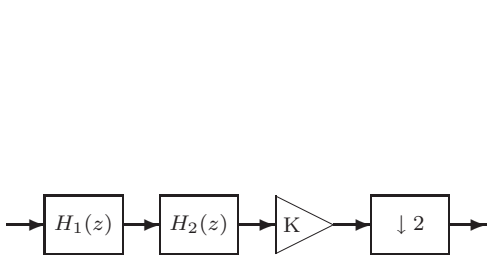
4.1.2 Cascade Polyphase Structure

The cascade filter structure puts the zeros at $z = -1$ in a separate filter section so as to ensure that they are not perturbed, even after coefficient quantization. Figure 4.4a shows the cascade structure (two sections $H_1(z)$ and $H_2(z)$ and a gain of K) followed by the downsampling operation.

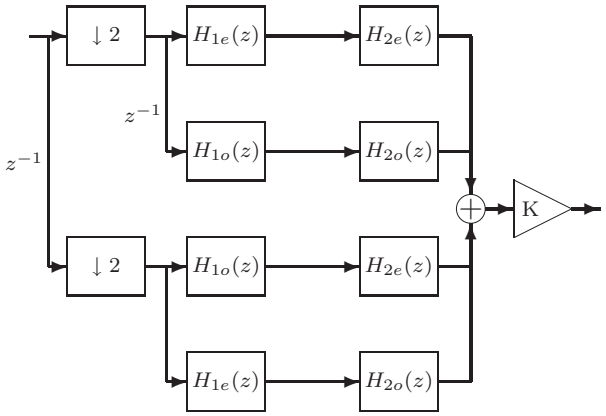
To apply the polyphase idea to the cascade structure, the filter is rewritten in so that the Noble identity can be used to move the downsampling operation to the filter input. Starting with the polyphase representation of both cascaded sections, $H_1(z)$ and $H_2(z)$, the equations are rearranged such that the delays are first grouped and factored out, and then the Noble identity is employed:

$$\begin{aligned} H(z) &= H_1(z).H_2(z) \\ &= [H_{1e}(z^2) + z^{-1}H_{1o}(z^2)].[H_{2e}(z^2) + z^{-1}H_{2o}(z^2)] \\ &= H_{1e}(z^2)H_{2e}(z^2) + \\ &\quad z^{-1}.[H_{1o}(z^2)H_{2e}(z^2) + H_{2o}(z^2)H_{1e}(z^2)] + \\ &\quad z^{-2}H_{1o}(z^2)H_{2o}(z^2). \end{aligned} \tag{4.2}$$

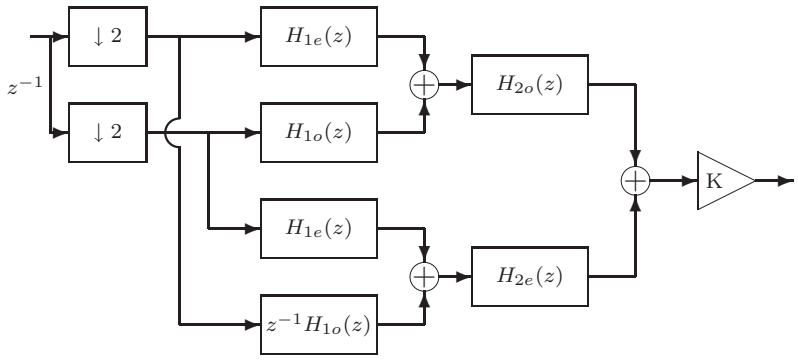
The result is the cascade-polyphase form in Figure 4.4b. This structure doubles the through-



(a) Non-polyphase cascade structure



(b) Non-optimal polyphase structure



(c) Cascade polyphase structure

Figure 4.4: Cascade form filter, analysis lowpass filter.

put of the original cascade-no-poly form while still preserving the zeros at $z = -1$. The structure in Figure 4.4b can be further simplified by absorbing the delay following the down-sampling in the second branch into $H_{1o}(z)$. The two branches containing $H_{2e}(z)$ can then be combined into one branch and the two branches containing $H_{2o}(z)$ can be combined into another branch to give the structure in Figure 4.4c [32].

The structure in Figure 4.4c represents the analysis LPF $H(z)$ of the filter bank, which is split into two sections—$H_1(z)$ containing the four zeros at $z = -1$ and $H_2(z)$ containing the remaining zeros in the right half of the z-plane. Since each polyphase section of first cascade filter $H_1(z)$ appears twice in the structure, the cascade polyphase implementation requires

more hardware than the original non-polyphase cascade form. This extra hardware, however can be used to implement the analysis HPF $G(z)$ of the filter bank as shown next.

The cascade form for the biorthogonal 9/7 lowpass filters are (equations (3.1) and (3.2)),

$$\begin{aligned} H(z) &= k_9 H_1(z) \cdot H_2(z) \qquad \text{and} \\ F(z) &= k_7 F_1(z) \cdot F_2(z), \end{aligned}$$

where

$$H_1(z) \leftrightarrow \{1, 4, 6, 4, 1\}, \qquad H_2(z) \leftrightarrow \{1, a_1, a_2, a_1, 1\},$$

$$F_1(z) \leftrightarrow \{1, 4, 6, 4, 1\}, \qquad F_2(z) \leftrightarrow \{1, a_3, 1\}.$$

The analysis highpass filter $G(z)$ is related to the synthesis lowpass filter $F(z)$ by $G(z) = F(-z)$; the cascade form of $G(z)$ can thus be written as,

$$G(z) = k_7 G_1(z) \cdot G_2(z),$$

where

$$G_1(z) \leftrightarrow \{1, -4, 6, -4, 1\} \quad \text{and} \quad G_2(z) \leftrightarrow \{1, -a_3, 1\}.$$

Splitting $G_1(z)$ into its polyphase sections, we see that

$$G_{1e}(z) = H_{1e}(z) \quad \text{and} \quad G_{1o}(z) = -H_{1o}(z).$$

Thus the cascade polyphase structure for the analysis highpass filter can be obtained from the lowpass filter structure in Figure 4.4c, by simply scaling the branches containing $H_{1o}(z)$ by -1, and using the polyphase sections $G_{2e}(z)$ and $G_{2o}(z)$ in place of $H_{2e}(z)$ and $H_{2o}(z)$ respectively.

The cascade polyphase structure for the analysis lowpass and highpass filters can be combined to give the structure in Figure 4.5—a unified cascade polyphase structure for the analysis side of the biorthogonal 9/7 filter bank [32]. Since the polyphase sections belonging to $H_1(z)$ are used by both—$H(z)$ and $G(z)$, the total SPT required for this cascade-polyphase structure is the same as that required for the non-polyphase cascade form.

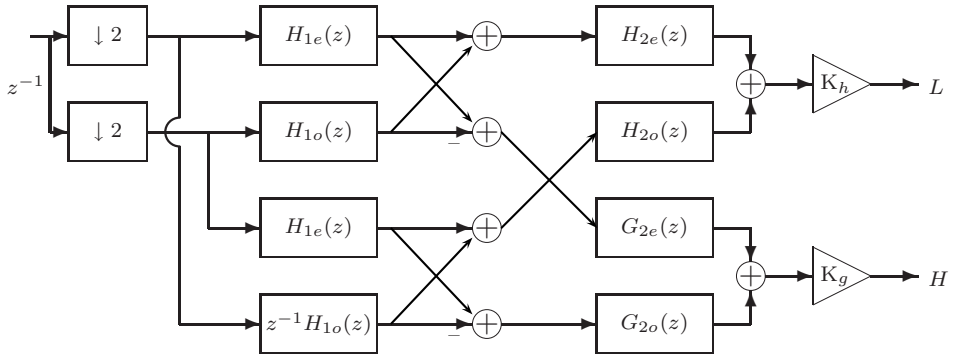


Figure 4.5: Cascade polyphase structure for analysis side.

4.1.3 Direct-Cascade Polyphase Structure

The direct form coefficients in Table 3.2 (with and without gain compensation), do not preserve the zeros at $z = -1$. As a result subjective quality of images compressed using the direct-poly structure is inferior to that obtained using the cascade-poly structure. The cascade structure has the coefficients that maintain zeros at $z = -1$, and can be used to derive the direct-poly coefficients.

Starting with the quantized coefficients from the "cascade with z_1 compensation" method shown in Table 3.3, the coefficients of the first and second sections are convolved together to make a single section. This single section is mathematically equivalent to the original cascade, but can be implemented in a direct form polyphase implementation. The normalizing gain factor, K, remains a separate stage. The resulting coefficients are shown in Table 4.1. This structure is referred to as the direct-cascade-poly form, since it is equivalent in image compression performance to the cascade, but is a direct polyphase structure.

The number of SPT terms required for the direct-cascade-poly structure depends on the particular coefficients; in general, the convolved coefficients for the direct-cascade-poly structure will require more terms than the original cascade-no-poly coefficients from which they were derived. For the "z_1 compensation" coefficients, the cascade-no-poly structure required $T = 32$ SPT terms and the direct-cascade-poly structure required $T = 45$. The

Table 4.1: Direct-cascade-poly coefficients obtained by convolving the "z_1 compensation" coefficients in Table 3.3, $T = 45$.

n	$H'(z)$		$F'(z)$	
0	1	00001.0000	-1	$000\overline{1}$.0000
1	-0.625	00000.$\overline{1}0\overline{1}0$	-0.6875	0000.$\overline{1}0\overline{11}$
2	-2.8125	$000\overline{11}$.$010\overline{1}$	6.25	0110.0100
3	10.375	01010.0110	11.875	$1100.00\overline{1}0$
4	23.125	$1100\overline{1}$.0010	6.25	0110.0100
5	10.375	01010.0110	-0.6875	0000.$\overline{1}0\overline{11}$
6	-2.8125	$000\overline{11}$.$010\overline{1}$	-1	$000\overline{1}$.0000
7	-0.625	00000.$\overline{1}0\overline{1}0$		
8	1	00001.0000		
K	0.0390625	0.0000101	0.0625	(simple shift)
G	-		-	

direct-cascade-poly design method gives the high throughput of a polyphase structure with a moderate increase in hardware cost, while preserving image compression quality by leaving the zeros at $z = -1$ unperturbed.

4.2 Results

Polyphase and non-polyphase structures are implemented using the architecture shown in Figure 3.8b, where each filter coefficient is implemented using a carry save adder (CSA) tree and a vector merge adder. Results involve four different combinations of filter coefficient quantization technique, form and structure: a direct form with gain compensation (from Table 3.2) in both non-polyphase and polyphase structures (direct-no-poly and direct-poly), a cascade form with z_1 compensation (from Table 3.3) in non-polyphase and polyphase structures (cascade-no-poly and cascade-poly), and a direct form that effectively implements cascade with z_1 compensation in a polyphase structure (direct-cascade-poly).

From an image compression performance point of view, since the direct-no-poly and direct-

Table 4.2: Hardware metrics for polyphase implementations of the filter banks.

	Non-Polyphase		Polyphase		
	direct-no-poly	cascade-no-poly	direct-poly	cascade-poly	direct-cascade-poly
SPT terms	32	32	32	32	45
size (logic cells)	992	1077	1143	1371	1415
latency (clock cycles)	9	13	19	26	21
full rate (MHz)	78.31	94.02	143.67	111.70	130.45
half rate (MHz)	-	-	71.83	55.85	65.22
Energy (mJ)	6.18	6.36	7.83	4.81	8.17
approx format, s_l	(17, -8)	(21, -11)	(17, -8)	(21, -11)	(21, -11)
details format, d_l	(27, -18)	(17, -8)	(27, -18)	(17, -8)	(17, -8)

poly are mathematically equivalent, both give the same results as shown in Table 3.5 under "direct with gain compensation". Similarly, the cascade-no-poly, cascade-poly and direct-cascade-poly systems, all give the results noted under "cascade with z_1 compensation". The structures using cascade coefficients result in significantly better qualitative results since there is no checkerboarding; the checkerboard artifact is easily visible with the direct structures at most compression ratios.

Hardware performance of the four systems was evaluated as in Chapter 3. Table 4.2 summarizes the hardware performance of the filter banks. The polyphase filters operate at a higher rate than the non-polyphase versions; for example, the direct polyphase filter bank has a full rate of 144 MHz, while the non-polyphase version has a full rate of only 78 MHz. The difference in rate between the non-polyphase filters and the polyphase filters can be accounted for by considering which hardware must run at the full rate of speed. In the non-polyphase filters, where downsampling occurs after the analysis filters and upsampling occurs before the synthesis filters, practically all of the hardware (i.e., all the individual filters) must run at the full rate. In essence, the non-polyphase analysis hardware computes twice as many outputs as needed, and then throws half of them away during downsampling. In the polyphase filters, where downsampling occurs before the analysis filters and upsampling occurs after the synthesis filters, only the downsampling and upsampling hardware must run at the full

rate; the filters themselves run at the slower half rate. The polyphase filters perform only half as many computations as their non-polyphase counterparts, since they perform only the computations that are needed. Since the filters are by far the most complicated part of the system, with the longest critical paths, a polyphase system that requires that they run only at half speed will be able to use a faster system clock.

The cascade-poly structure is faster than the cascade non-poly structure, but does not show the almost doubling of throughput exhibited by the direct-poly structure. Of the three structures derived from the cascade coefficients, the direct-cascade-poly structure has the highest throughput; but this comes at the cost of larger hardware, since it requires 45 SPT terms compared to 32 SPT terms for the two other structures. Note that in hardware, the polyphase implementations do not show an exact doubling of throughput compared to the non-polyphase structures, as one would expect. This is because in an FPGA implementation, throughput not only depends on whether each filter has been implemented as polyphase, but also on a number of timing parameters such as gate delays, latency of the vector merge adders and routing delays, and the interaction between these parameters.

The energy required for computing a 5-level DWT of a 512×512 image (see Appendix A) is reported in the last row of Table 4.2. The direct non-polyphase structure requires less energy than the direct polyphase structure, while the cascade polyphase structure requires less energy than the cascade non-polyphase structure. Among the polyphase structures, the cascade-poly structure is the most energy efficient. This is because, of the 18 coefficients needed to implement the cascade-poly structure, only 4 coefficients (a_2, and 2 instances for a_1 for $H(z)$, and a_3 for $G(z)$) have wide bit widths. All other coefficients have compact bit widths, resulting in smaller, low-power adders. In contrast, the direct-cascade-poly structure has coefficients with wide bit widths as seen in Table 4.1.

4.3 Summary

The polyphase structure of the filter bank can be used to increase throughput of the filter bank. In an FPGA implementation the actual increase in throughput depends on a number of hardware timing parameters, and may not always equal the theoretical doubled throughput. The best image compression performance is obtained by adopting the cascade form for filters and quantizing coefficients using z_1 compensation. The cascade-poly structure results in a low-power polyphase implementation using the cascade coefficients. The direct-cascade-poly structure gives the same image compression performance at higher throughput, but at the cost of larger hardware and higher energy consumption.

The next chapter studies "lifting"—an alternative polyphase implementation for computing the DWT.

Chapter 5

Lifting Implementation

The polyphase and non-polyphase implementations of the biorthogonal 9/7 DWT in Chapters 3 and 4 employ Mallat's pyramid algorithm based on the perfect reconstruction filter bank shown in Figure 2.3. This chapter examines the "lifting" approach to computing the DWT. Lifting provides an efficient alternative to the traditional convolution filter bank implementation. This method not only requires fewer computations than the conventional method, but it also operates in polyphase—i.e. the output rate of the analysis stage is the same as the input rate, and hence there are no wasted computations.

5.1 Lifting Structure

Consider the signals in a two-channel PR filter bank shown in Figure 5.1. Splitting the input $X(z)$ into its even and odd phases, and using the polyphase forms for the analysis filters,

$$\begin{aligned} H(z) &= H_e(z) + z^{-1}H_o(z) \qquad \text{and} \\ G(z) &= G_e(z) + z^{-1}G_o(z), \end{aligned}$$

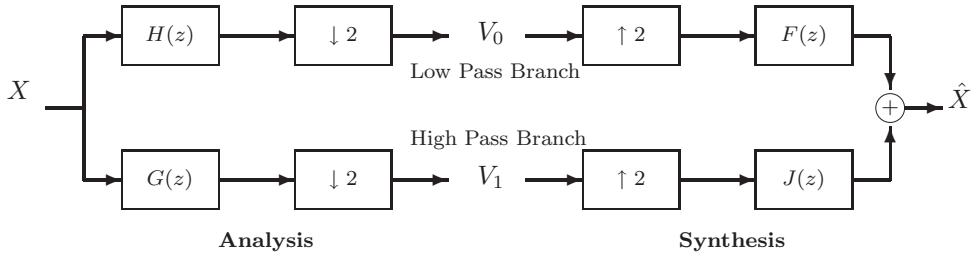


Figure 5.1: A two-channel (PR) filter bank.

the analysis stage of the filter bank can be expressed in the matrix form as

$$\begin{bmatrix} V_0(z) \\ V_1(z) \end{bmatrix} = \begin{bmatrix} H_e(z) & H_o(z) \\ G_e(z) & G_o(z) \end{bmatrix} \begin{bmatrix} X_e(z) \\ z^{-1}X_o(z) \end{bmatrix}. \tag{5.1}$$

Using the polyphase forms for the filters on the synthesis side, the reconstructed signal can be expressed as:

$$\hat{X}(z) = \begin{bmatrix} z^{-1} & 1 \end{bmatrix} \begin{bmatrix} F_o(z^2) & J_o(z^2) \\ F_e(z^2) & J_e(z^2) \end{bmatrix} \begin{bmatrix} V_0(z^2) \\ V_1(z^2) \end{bmatrix}. \tag{5.2}$$

Perfect reconstruction will be obtained when [33],

$$\mathbf{F}_p(z)\mathbf{H}_p(z) = z^{-d} \begin{bmatrix} 0 & 1 \\ z^{-1} & 0 \end{bmatrix}, \tag{5.3}$$

where,

$$\mathbf{H}_p(z) = \begin{bmatrix} H_e(z) & H_o(z) \\ G_e(z) & G_o(z) \end{bmatrix}, \qquad \text{and}$$

$$\mathbf{F}_p(z) = \begin{bmatrix} F_o(z) & J_o(z) \\ F_e(z) & J_e(z) \end{bmatrix},$$

are the analysis and synthesis polyphase matrices respectively, of the two-channel filter bank. When the polyphase matrices satisfy the PR condition of equation (5.3), the reconstructed signal is $\hat{x}(n) = x(n - n_0)$, where $n_0 = 2d + 2$. The polyphase form of the filter bank is shown in Figure 5.2.

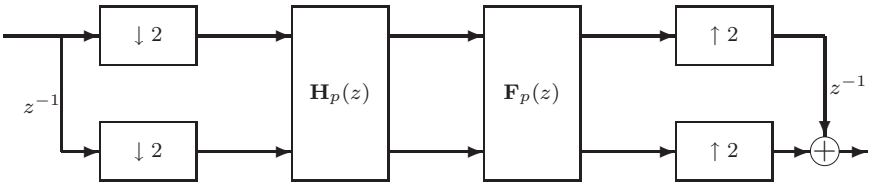


Figure 5.2: Filter bank using the analysis and synthesis polyphase matrices.

For the biorthogonal 9/7 filters, the polyphase matrices are [1]:

$$\mathbf{H}_p(z) = \begin{bmatrix} h_4(z^2+z^{-2})+h_2(z+z^{-1})+h_0 & h_3(z^2+z^{-1})+h_1(z+1) \\ g_1(1+z)+g_3(z^2+z^{-1}) & g_0 z+g_2(z^2+1) \end{bmatrix} \quad \text{and}$$

$$\mathbf{F}_p(z) = \begin{bmatrix} f_1(z+1)+f_3(z^{-1}+z^2) & j_0+j_2(z^{-1}+z)+j_4(z^{-2}+z^2) \\ f_0+f_2(z+z^{-1}) & j_1(1+z^{-1})+j_3(z+z^{-2}) \end{bmatrix}. \tag{5.4}$$

The filters $(H(z), G(z))$ constitute a complementary filter pair; i.e. their analysis polyphase matrix has determinant 1. Thus, it can be decomposed using the Euclidean algorithm for Laurent polynomials [17], so that

$$\mathbf{H}_p(z)^{\mathbf{T}} = \prod_{i=1}^{m} \begin{bmatrix} 1 & s_i(z) \\ 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 \\ t_i(z) & 1 \end{bmatrix} \begin{bmatrix} K & 0 \\ 0 & 1/K \end{bmatrix}.$$

This factorization is not unique, and there are a number of $\{s_i(z), t_i(z)\}$ pairs that lead to the same analysis and synthesis polyphase matrices. For odd length symmetric wavelet filters whose lengths differ by 2, such as the biorthogonal 9/7 wavelet filters, the polyphase matrix

[1] The biorthogonal 9/7 filters are given by [34]:

$$\begin{aligned} H(z) &= h_0 + h_1(z+z^{-1}) + h_2(z^2+z^{-2}) + h_3(z^3+z^{-3}) + h_4(z^4+z^{-4}) \\ G(z) &= g_0 z + g_1(z^2+1) + g_2(z^3+z^{-1}) + g_3(z^4+z^{-2}) \\ F(z) &= f_0 + f_1(z+z^{-1}) + f_2(z^2+z^{-2}) + f_3(z^3+z^{-3}) \\ J(z) &= j_0 z + j_1(1+z^{-2}) + j_2(z+z^{-3}) + j_3(z^2+z^{-4}) + j_4(z^3+z^{-5}). \end{aligned}$$

where,

$h_0 = j_0 = 0.85269867900889$ $\quad f_0 = g_0 = 0.78848561640558$
$h_1 = -j_1 = 0.37740285561283$ $\quad f_1 = -g_1 = 0.41809227322162$
$h_2 = j_2 = -0.11062440441844$ $\quad f_2 = g_2 = -0.04068941760916$
$h_3 = -j_3 = -0.02384946501956$ $\quad f_3 = -g_3 = -0.06453888262870$
$h_4 = j_4 = 0.03782845550726$

Table 5.1: Lifting coefficients for the biorthogonal 9/7 wavelet filters.

	Irrational	Rational
α	-1.58613434...	-3/2
β	-0.0529801185...	-1/16
γ	0.882911076...	4/5
δ	0.443506852...	15/32
ζ	1.14960439...	$4\sqrt{2}/5$

can be factored such that $\{s_i(z), t_i(z)\}$, are two-tap symmetric filters. These filters find favor for hardware implementation as they reduce the number of multipliers to be implemented.

Factoring the biorthogonal 9/7 polyphase matrix results in the form [17]:

$$\begin{aligned}\mathbf{H}_p(z)^{\mathbf{T}} &= \begin{bmatrix} 1 & \alpha(1+z^{-1}) \\ 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 \\ \beta(1+z) & 1 \end{bmatrix} \begin{bmatrix} 1 & \gamma(1+z^{-1}) \\ 0 & 1 \end{bmatrix} \\ &\quad \begin{bmatrix} 1 & 0 \\ \delta(1+z) & 1 \end{bmatrix} \begin{bmatrix} \zeta & 0 \\ 0 & 1/\zeta \end{bmatrix} \end{aligned} \tag{5.5}$$

where, the floating point values of α, β, γ, δ and ζ (called the lifting coefficients) are listed in Table 5.1 under "Irrational".

Implementation of the analysis side of the biorthogonal 9/7 filter bank using the lifting scheme is shown in Figure 5.3 (i.e. implementation of the matrix $\mathbf{H}_p(z)$). The analysis side of the filter bank is represented by the operations:

$$\begin{aligned} s_l^{(0)} &= x_{2l} \\ d_l^{(0)} &= x_{2l-1} \\ d_l^{(1)} &= d_l^{(0)} + \alpha(s_l^{(0)} + s_{l-1}^{(0)}) \\ s_l^{(1)} &= s_l^{(0)} + \beta(d_l^{(1)} + d_{l+1}^{(1)}) \\ d_l^{(2)} &= d_l^{(1)} + \gamma(s_l^{(1)} + s_{l-1}^{(1)}) \\ s_l^{(2)} &= s_l^{(1)} + \delta(d_l^{(2)} + d_{l+1}^{(2)}) \\ s_l &= \zeta s_l^{(2)} \end{aligned}$$

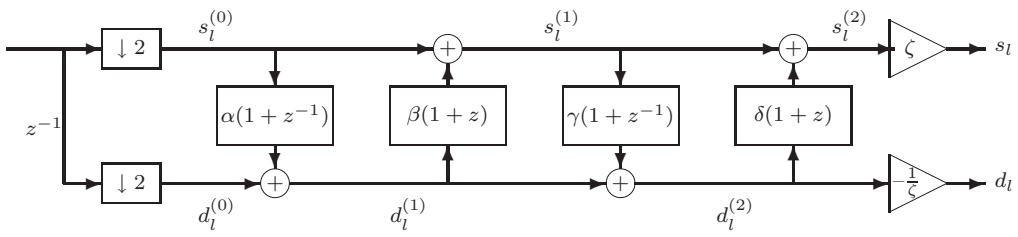


Figure 5.3: Lifting implementation of the analysis side of the biorthogonal 9/7 filter bank.

$$d_l \;=\; -d_l^{(2)}/\zeta. \tag{5.6}$$

The lowpass and highpass filtering operations have been replaced by 4 two-tap "lift" and "update" filters and a scaling step at the end. The synthesis side of the filter bank simply inverts the scaling, and reverses the sequence of the lifting and update steps. Figure 5.4 shows the synthesis side of the filter bank using lifting. Figures 5.3 and 5.4 show that if the scaling factors ζ and $1/\zeta$ are ignored and the synthesis stage immediately follows the analysis stage (i.e. no compression), analysis is *exactly* inverted regardless of the quantized values used for α, β, γ and δ. Thus, the lifting structure is inherently *orthogonal*, giving it a significant advantage over convolution where perfect reconstruction is not guaranteed after the coefficients are quantized.

The irrational lifting coefficients in Table 5.1 have infinitely long binary representations; for a fast hardware implementation these coefficients must be quantized, or approximated in fixed-point. As before, the more SPT terms used to represent the lifting coefficients (the higher the value of T), the closer the quantized lifting coefficients are to the unquantized coefficients—

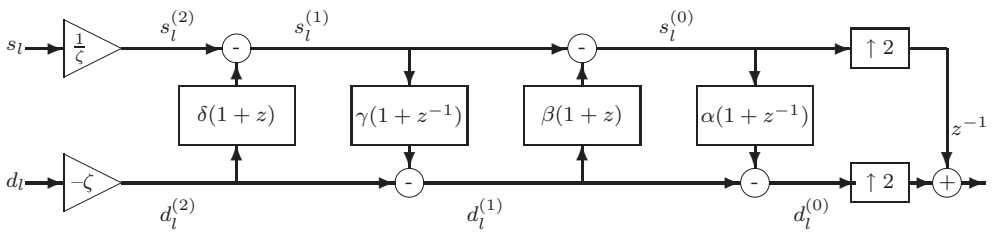


Figure 5.4: Lifting implementation of the synthesis side of the biorthogonal 9/7 filter bank.

and consequently, the closer the PSNR values achieved by the fixed-point hardware are to the PSNR values obtained with unquantized coefficients.

The unquantized 9/7 analysis lowpass and highpass filters have four zeros at $z = -1$ and 1 respectively. These zeros determine the smoothness of the analysis and synthesis scaling functions and prevent DC leakage in the analysis highpass filter. Tay and Guangjun et. al. independently derived a set of rational lifting coefficients by moving two zeros of the analysis lowpass filter away from $z = -1$ [18, 19]. These coefficients are listed in Table 5.1 under "Rational". The rational lifting coefficients generate PSNR values almost exactly equal to the irrational coefficient PSNR values [19]. The rational coefficients are also more easily quantized than the irrational; all except γ and ζ have finite binary representations and can be represented exactly—with no approximation—with a small T.

5.2 Lifting Coefficient Quantization

5.2.1 Quantization Objectives

Hardware implementation of the lifting structure requires quantization of the six lifting coefficients—α, β, γ, δ, ζ and $1/\zeta$. Quantization of the six lifting coefficients changes the magnitude response of the analysis filters. As seen in Chapter 3, good compression performance requires that the magnitude response of the quantized analysis filters ($H'(z)$ and $G'(z)$) closely approximate the magnitude response of the unquantized 9/7 filters ($H(z)$ and $G(z)$). Quantized values of α, β, γ and δ determine the shape of the magnitude responses; whereas, quantized values of ζ and $1/\zeta$, determine the DC gain and Nyquist gain of $H'(z)$ and $G'(z)$ respectively.

Quantization techniques presented in the following subsections, for both the irrational and rational lifting coefficients, attempt to:

1. perfectly reconstruct the DC component ($|H'(0)||G'(\pi)| = 2$),

2. prevent DC leakage and avoid the checkerboarding artifact at low bit rates ($|G'(0)| = 0$), and

3. minimize the mean-squared-error (MSE) of the magnitude responses of the quantized lowpass and highpass filters ($|H'(\omega)|$ and $|G'(\omega)|$).

5.2.2 Quantization of Irrational Coefficients

Lifting coefficient quantization can be posed as a problem of allocating a fixed number T of SPT terms to the six coefficients (α, β, γ, δ, ζ and $1/\zeta$) while retaining acceptable subjective quality for the reconstructed image. Here, "acceptable" subjective quality is defined by the absence of the checkerboarding artifact. (For all approaches, the T used is the smallest number of terms that did not result in checkerboarding.) The following three approaches were developed for quantizing the irrational lifting coefficients.

5.2.2.1 Mostly Uniform Allocation (MUA)

Here all six lifting coefficients are allotted the same base number of SPT terms and then the largest magnitude coefficients are allotted extra terms. Once the number of terms for a given coefficient is decided, the unquantized coefficient is quantized to the closest SPT coefficient with that number of terms. Three SPT terms were used as the base, and three extra terms were allotted to α and ζ, for a total of $T = 21$ terms. Quantized lifting coefficients obtained with lower values of T resulted in checkboarding; $T = 21$ was the smallest value that yielded images free from checkerboarding. Table 5.2 lists these coefficients under "irrational: MUA".

5.2.2.2 Exhaustively Searched Allocation (ESA)

All possible allocations of a fixed number T of SPT terms to the six lifting coefficients are examined. For each allocation, α, β, γ and δ are quantized to the closest SPT coefficient

Table 5.2: Quantized lifting coefficients for biorthogonal 9/7 wavelet filters.

	Quantized irrational coefficients			Quantized rational coefficients		
	MUA T=21	ESA T=19	SA T=19	MUA T=20	MUA-LS T=19	MUA-LSGC T=21
α	-1.5859375	-1.59375	-1.5546875	-1.5	-1.5	-1.5
β	-0.052734375	-0.0546875	-0.0546875	-0.0625	-0.0625	-0.0625
γ	0.8828125	0.8828125	0.85546875	0.7998046875	0.7998046875	0.7998046875
δ	0.44140625	0.4453125	0.4453125	0.46875	0.46875	0.46875
ζ	1.1484375	1.140625	1.1328125	1.13134765625	0.7998046875	0.7998046875
$-1/\zeta$	-0.87109375	-0.876708984375	-0.8828125	-0.8837890625	-1.25	-1.25030517578125
Lumped scaling	N/A	N/A	N/A	N/A	$\sqrt{2}$	$\sqrt{2}$

with the allotted number of terms; together, they determine the shape of the magnitude response of the quantized filters. Next, a gain factor ζ_{comp} is computed such that the DC gain of the analysis LPF equals $\sqrt{2}$; this gain compensates for the change in DC gain due to quantization of α, β, γ and δ. Quantized versions of the gain factors are then obtained by approximating ζ_{comp} and $1/\zeta_{comp}$ using the terms allotted to them. The following cost function, CF_{ESA}, is used to evaluate the quantized coefficient sets:

$$CF_{ESA} = MSE_h + MSE_g + dev\_dc, \tag{5.7}$$

where

$$\begin{aligned} MSE_h &= \frac{1}{\pi}\int_0^{\pi}(|H(\omega)| - |H'(\omega)|)^2 \mathrm{d}\omega, \\ MSE_g &= \frac{1}{\pi}\int_0^{\pi}(|G(\omega)| - |G'(\omega)|)^2 \mathrm{d}\omega, \\ dev\_dc &= |2 - |H'(0)||G'(\pi)||. \end{aligned}$$

CF_{ESA} measures the deviation of $H'(z)$ from $H(z)$, the deviation of $G'(z)$ from $G(z)$, and the degree to which an image's DC component is reconstructed. The smallest number of SPT terms for which the reconstructed images did not exhibit checkerboarding was $T = 19$. The coefficient set judged to be best resulted in the smallest CF_{ESA} using 19 terms. Table 5.2 lists the resulting coefficients under "irrational:ESA".

5.2.2.3 Simulated Annealing (SA)

Simulated annealing searches for the best quantized coefficient set given a *domain* for each coefficient [35][36]. The domain denoted $\mathcal{D}(B, T_{max})$ is the set of all real numbers between 2 and -2 whose SPT representation is up to B bits wide and has up to T_{max} non-zero terms. SA has the advantage that constraints can be placed on the bit widths of the lifting coefficients by fixing B; however, there is no direct control over the total T for the set.

Coefficients are quantized in two passes. In the first pass, SA iteratively searches for the best four-coefficient set $\{\alpha, \beta, \gamma, \delta\}$. For this pass, scaling factors ζ and $1/\zeta$ are set to 1, and the quality of a candidate set is measured by the cost function, CF_{SA}. The cost function uses the MSE between the quantized and unquantized magnitude responses. MSE in the stopband is weighted more heavily than MSE in the passband to ensure that $G'(z)$ has a magnitude response close to 0 at $z = 1$.

$$CF_{SA} = 0.75(MSE_{stopband}) + 0.25(MSE_{passband}),$$

where

$$\begin{aligned} MSE_{stopband} &= \frac{2}{\pi}\int_{\pi/2}^{\pi}(|H(\omega)| - |H'(\omega)|)^2 d\omega + \\ &\quad \frac{2}{\pi}\int_{0}^{\pi/2}(|G(\omega)| - |G'(\omega)|)^2 d\omega, \\ MSE_{passband} &= \frac{2}{\pi}\int_{0}^{\pi/2}(|H(\omega)| - |H'(\omega)|)^2 d\omega + \\ &\quad \frac{2}{\pi}\int_{\pi/2}^{\pi}(|G(\omega)| - |G'(\omega)|)^2 d\omega \end{aligned}$$

In the second pass, SA uses the best quantized $\{\alpha, \beta, \gamma, \delta\}$ from the first pass and iteratively searches for the best quantized scaling factors ζ and $1/\zeta$. For this pass, the cost function CF_{ESA} from equation (5.7) is used.

Optimal quantized lifting coefficients were obtained using $\mathcal{D}(9,4)$ for $\{\alpha, \beta, \gamma, \delta\}$ and $\mathcal{D}(8,4)$ for the scaling factors. Using smaller bit widths or fewer non-zero digits resulted in checkerboarding. The optimal lifting coefficient set used a total of $T = 19$ SPT terms; Table 5.2 lists these coefficients under "irrational:SA".

5.2.3 Quantization of Rational Coefficients

Of the rational lifting coefficients depicted in Table 5.1, α, β and δ can be represented exactly in SPT with a small number of terms, but γ, ζ and $1/\zeta$ have infinitely long SPT representations. Hence the fixed-point quantization of γ, ζ and $1/\zeta$ require approximations. The following three approaches were developed for quantizing these three coefficients.

5.2.3.1 Mostly Uniform Allocation (MUA)

The mostly uniform allocation scheme described above is again used for quantizing γ, ζ and $1/\zeta$. This scheme resulted in coefficients that required a total of $T = 20$. These coefficients are listed in Table 5.2 under "rational:MUA".

5.2.3.2 With Lumped Scaling (MUA-LS)

The rational gain factors ζ and $1/\zeta$ include factors of $\sqrt{2}$ and $1/\sqrt{2}$, respectively. Implementation of these factors in hardware can be avoided by lumping them together. For example, one 2-D DWT stage includes both row and column filtering and two factors are required. The $\sqrt{2}$ factors can be defered until 2 such factors can be combined into one factor of 2, 1/2 or 1. Multiplication and division by 2 can then be easily implemented as bit shifts.

With lumped scaling, γ and ζ are both equal to 0.8, which has an infinitely long binary (and SPT) representation. Therefore, γ and ζ must be approximated by quantization; the other coefficients, including $1/\zeta = 1.25$, are represented *exactly*. γ and ζ are quantized to γ' and

ζ' by allocating the remaining Ts to each coefficient. A total of $T = 19$ terms are used for the set and the coefficients are labeled "rational:MUA-LS" in Table 5.2.

5.2.3.3 With Lumped Scaling and Gain Compensation (MUA-LSGC)

This method uses lumped scaling to avoid implementation of the $\sqrt{2}$ factors as before, but modifies $1/\zeta$ from its original value of 1.25 in order to compensate for the change in DC gain caused by quantization of γ. γ and ζ are quantized as in MUA-LS, and a new value of $1/\zeta$, $[1/\zeta]_{\text{comp}}$, is computed using the quantized value of ζ; i.e.

$$[1/\zeta]_{\text{comp}} = \frac{1}{\zeta_{\text{quantized}}}.$$

$[1/\zeta]_{\text{comp}}$ is then quantized using the terms allotted to it. A total of $T = 21$ terms are used and the coefficients are labeled "rational:MUA-LSGC" in Table 5.2.

5.3 Results

5.3.1 Compression Performance

Table 5.3 compares the PSNR performance of the six quantized coefficient designs. Among the quantized irrational designs, quantized coefficients obtained using the ESA method performed best; among the quantized rational designs, all three designs performed equally well under compression, but at 1:1 the gain compensation technique employed in MUA-LSGC outperforms MUA-LS and MUA. Among all the quantized coefficient designs, the rational coefficients quantized with lumped scaling and gain compensation (MUA-LSGC) yield the best PSNR performance.

Table 5.4 shows the filter properties for the six lifting implementations. Among the quantized irrational coefficients, the quantized set obtained using ESA has the smallest deviation-at-DC (1.96e-5) and no-distortion MSE (5.8e-11). Among the quantized rational coefficients the

Table 5.3: PSNR performance of the six quantized lifting coefficient designs. Numbers shown for the quantized filters indicate PSNR degradation with respect to unquantized PSNR values. Negative numbers indicate PSNR performance worse than the unquantized filter, while positive numbers indicate PSNR performance better than the unquantized filters.

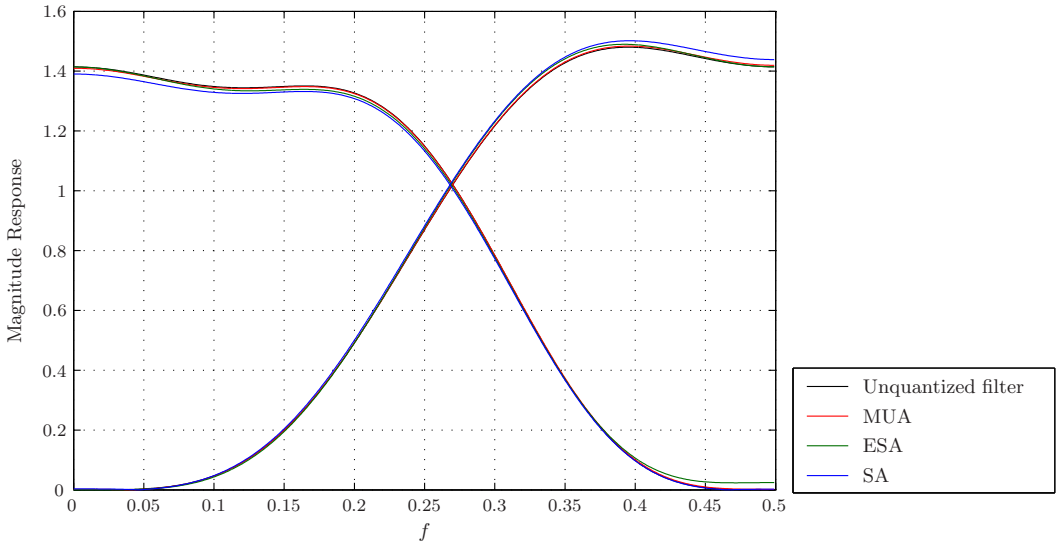
		Unquantized	Quantized irrational coefficients			Quantized rational coefficients		
			MUA T=21	ESA T=19	SA T=19	MUA T=20	MUA-LS T=19	MUA-LSGC T=21
Aerial2	1:1	58.75	-2.62	+0.01	-0.08	-0.40	-1.34	+0.05
	8:1	32.76	-0.01	+0.00	-0.01	+0.00	+0.00	+0.00
	32:1	27.99	-0.01	+0.00	-0.02	-0.01	-0.01	-0.01
	100:1	24.79	-0.01	-0.01	-0.04	-0.01	-0.01	-0.02
Bike	1:1	58.70	-4.17	+0.00	-0.18	-0.67	-2.12	+0.05
	8:1	32.83	-0.02	-0.01	-0.03	-0.01	-0.01	-0.01
	32:1	24.81	-0.02	-0.01	-0.03	+0.00	+0.00	+0.00
	100:1	21.24	-0.02	-0.03	-0.14	-0.03	-0.04	-0.04
Woman	1:1	58.68	-3.20	+0.00	-0.13	-0.45	-1.53	+0.06
	8:1	33.68	+0.00	+0.01	+0.02	+0.01	+0.01	+0.01
	32:1	27.29	-0.01	-0.01	-0.07	-0.01	-0.01	-0.01
	100:1	24.51	-0.02	-0.03	-0.07	-0.03	-0.03	-0.03

— PSNR degradation < 0.06dB
— 0.06dB < PSNR degradation < 0.5dB
— 0.5dB < PSNR degradation

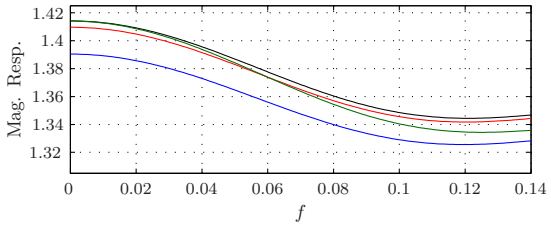
Table 5.4: Filter properties for the six quantized lifting coefficient designs.

Filter Property	Unquantized	Quantized irrational coefficients			Quantized rational coefficients		
		MUA T=21	ESA T=19	SA T=19	MUA T=20	MUA-LS T=19	MUA-LSGC T=21
Deviation at DC	0	-8.0e-4	2.0e-5	-1.2e-4	2.5e-4	4.9e-4	3.9e-7
No Distortion MSE	0	6.3e-7	5.8e-11	1.5e-8	6.5e-8	2.4e-7	1.4e-14
Alias Cancellation MSE	0	0	0	0	0	0	0
Accuracy	4/4	0/0	0/0	0/0	0/0	0/0	0/0
Smoothness (analysis)	1.41	1.39	1.31	1.41	1.52	1.52	1.52
Smoothness (synthesis)	2.12	2.04	1.96	2.02	2.04	2.04	2.04
Coding Gain	9.71	9.69	9.70	9.70	9.71	9.71	9.70

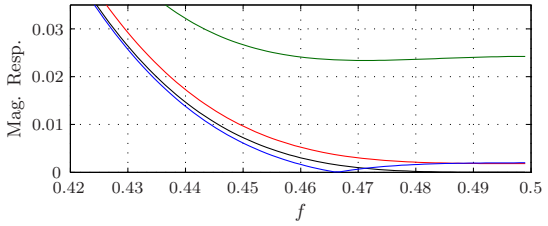
MUA-LSGC coefficients exhibit the lowest deviation-at-DC (3.8e-7) and no-distortion MSE (1.4e-14). This explains the superior performance of ESA and MUA-LSGC compared to the



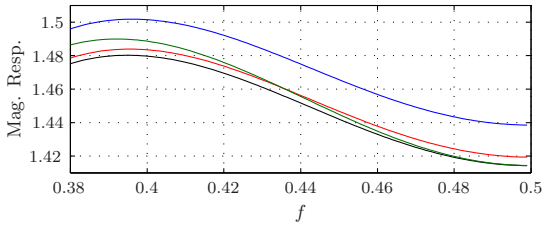
(a) Lowpass and highpass analysis filters



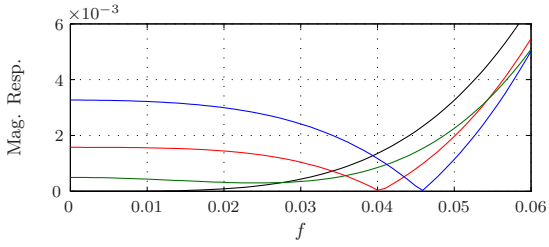
(b) Passband of lowpass filter



(c) Stopband of lowpass filter



(d) Passband of highpass filter

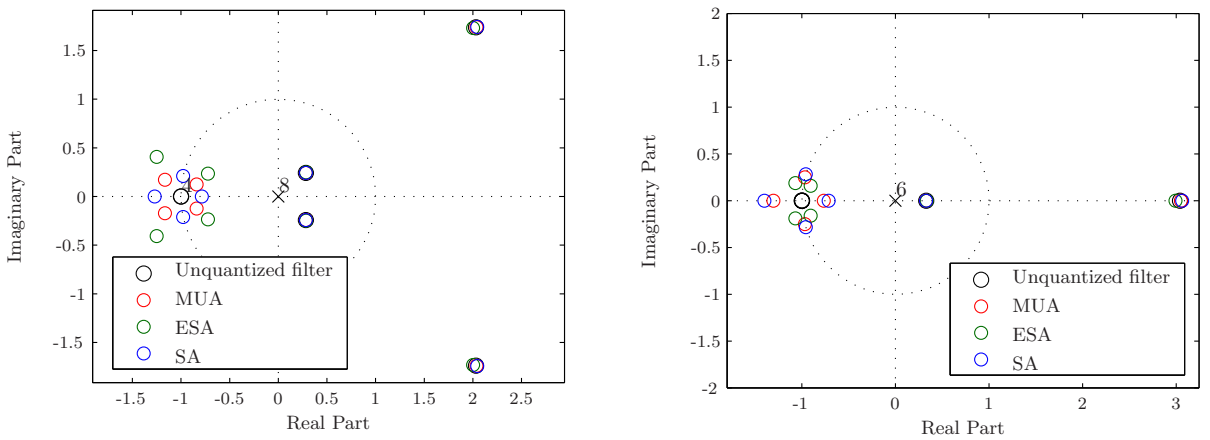


(e) Stopband of highpass filter

Figure 5.5: Comparison of magnitude response of analysis filters using unquantized, original filters and filters obtained using quantized irrational lifting coefficients.

other quantized coefficients. All six designs resulted in similar smoothness of the analysis and synthesis scaling functions; moreover, the coding gain of all six designs (9.69-9.71) was almost identical to the unquantized coding gain (9.71).

Figure 5.5 compares the magnitude response of the lowpass and highpass analysis filters using

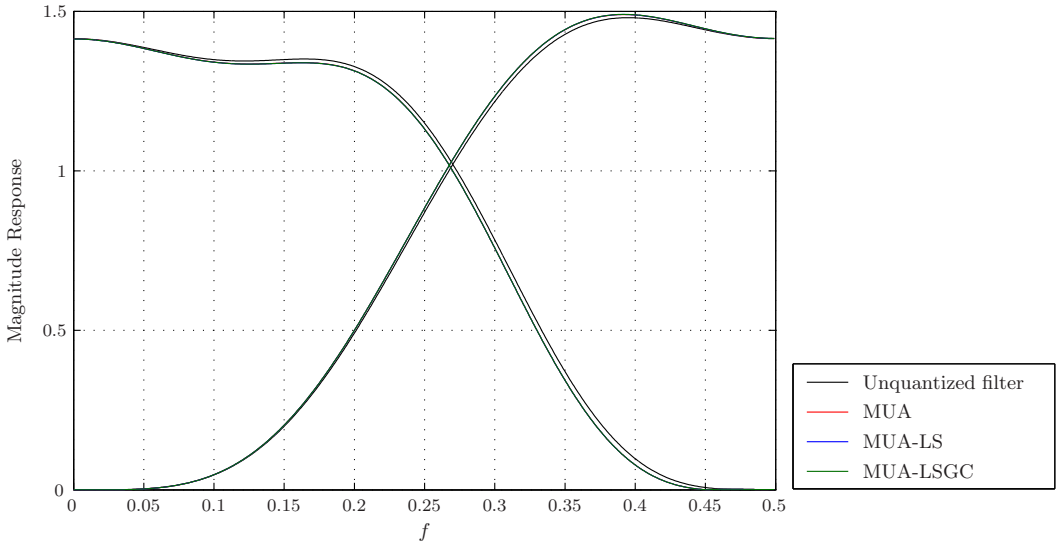


(a) $H(z)$ (b) $F(z)$

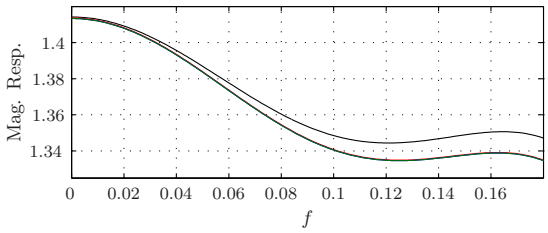
Figure 5.6: Comparison of location of zeros between unquantized and quantized irrational lifting coefficients. (a) 9-tap $H(z)$ and (b) 7-tap $F(z)$.

the unquantized and quantized irrational lifting coefficients. Among the three quantized coefficient sets, the coefficients obtained using ESA gives the closest match to the unquantized magnitude response in the passband of the analysis filters. The magnitude responses for the unquantized filters and ESA, is practically indistinguishable between the frequencies [0, 0.04] for the lowpass filter and [0.46, 0.5] for the highpass filter. Similarly, the ESA method gives the best match to the unquantized magnitude response in the stopband for the highpass analysis filter. Although the value of the magnitude response at $f = 0$ is not zero, the value is very small and there is no significant DC leakage through the highpass analysis filter. In the lowpass stopband frequencies, the ESA method yields the worst match to the unquantized magnitude response. For natural images however, since a very small fraction of the energy occurs at high frequencies around 0.5, high frequency leakage through the lowpass filter does not have a significant impact on the reconstructed image.

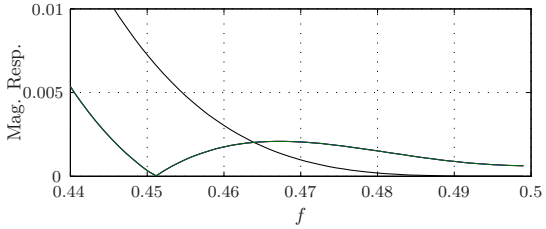
Pole zero plots for the unquantized biorthogonal 9/7 lowpass filters and the filters resulting from the three quantized irrational coefficient sets are shown in Figure 5.6. The quantized filters do not have zeros at $z = -1$; instead they are distributed around $z = -1$ in a manner so that the magnitude response at $f = 0.5$ is zero.



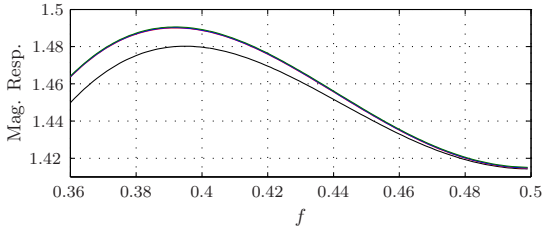
(a) Lowpass and highpass analysis filters



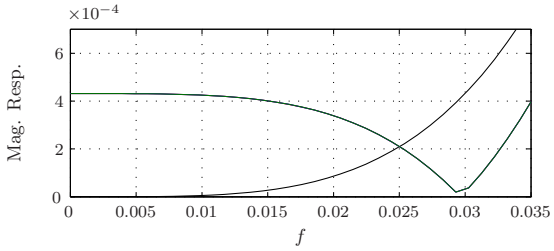
(b) Passband of lowpass filter



(c) Stopband of lowpass filter



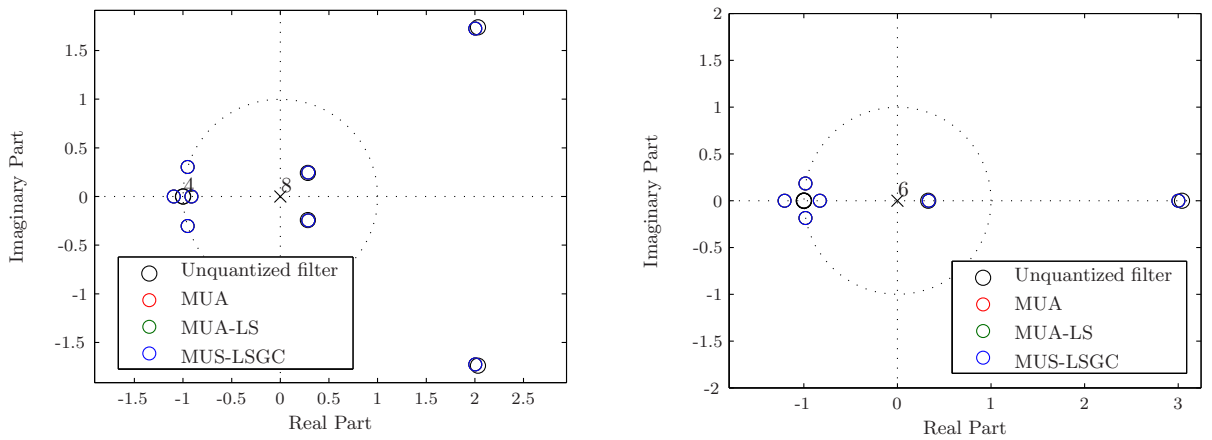
(d) Passband of highpass filter



(e) Stopband of highpass filter

Figure 5.7: Comparison of magnitude response of analysis filters using unquantized, original filters and filters obtained using quantized rational lifting coefficients.

Figures 5.7 and 5.8 show the magnitude responses and pole-zero plots for filters obtained using the three quantized rational coefficient sets. The three quantized magnitude responses are indistinguishable. All quantized rational implementations have a small a non-zero value (4.1e-4) for the analysis highpass magnitude response at $f = 0$. The pole-zero plots show



(a) $H(z)$ (b) $F(z)$

Figure 5.8: Comparison of location of zeros between unquantized and quantized rational lifting coefficients. (a) 9-tap $H(z)$ and (b) 7-tap $F(z)$.

that the quantized lowpass analysis and synthesis filters do not have zeros at $z = -1$, but they are distributed around $z = -1$ in a manner so that the magnitude response at $f = 0.5$ is zero.

Comparing PSNR results for the lifting implementation in Table 5.3 to PSNR results for the convolution implementation in Table 3.5, it can be seen that the lifting implementation performs better in terms of image compression performance. This is due to the increased robustness of lifting to coefficient quantization arising from the inherent orthogonality of the structure. Although this orthogonality should have ensured perfect reconstruction (PSNR = ∞) at 1:1, all filters in Table 5.3 (with quantized and unquantized lifting coefficients) show finite PSNR values at 1:1. There are two reasons for this—(a) DWT coefficients are rounded to integers after the 5-level analysis stage and before quantization, introducing irretrievable loss of information in the DWT coefficients before the synthesis stage, and (b) in the case of the quantized filters, the product of the quantized ζ and quantized $1/\zeta$ is not exactly equal to 1.

Table 5.5: Hardware performance of the six quantized lifting coefficient designs.

	Quantized irrational coefficients			Quantized rational coefficients		
	MUA	ESA	SA	MUA	MUA-LS	MUA-LSGC
SPT Terms	21	19	19	20	19	21
size (logic cells)	2705	2284	2441	1981	1958	2123
Latency(clock cycles)	53	51	51	47	49	49
input format	(8, 0)	(8, 0)	(8, 0)	(8, 0)	(8, 0)	(8, 0)
f_{max}(MHz)	60.90	72.59	70.29	78.55	75.74	63.63
Energy (mJ)	6.45	6.27	6.17	5.63	5.61	5.79
approx format, s_l	(47, -38)	(41, -32)	(45, -36)	(40, -31)	(39, -30)	(39, -30)
details format, d_l	(40, -31)	(40, -31)	(38, -29)	(34, -25)	(27, -17)	(39, -29)

5.3.2 Hardware Performance

Table 5.5 depicts the hardware performance of the six quantized coefficient designs. The maximum possible operating frequency f_{max} is obtained via timing analysis, and indicates system throughput. The energy shown is that required to compute a five-level, two-dimensional DWT of a 512×512 block of random pixels. The table also shows the fixed-point formats of the approximation and detail coefficients produced; the notation is (n, f): n is the total number of bits, including the sign, and 2^f is the weight of the least significant bit.

The quantized rational designs are smaller, faster, and require less energy than the quantized irrational designs. The primary reason is that the rational coefficients have narrower bit widths, which implies narrower adders that are faster and require less energy. (For example, the first coefficient, -1.5, requires only three bits.) In the lifting structure, intermediate signals grow in bit width after each lifting stage, with the amount of growth determined by the corresponding coefficient. In the rational designs, the bit width grows more slowly, with bit widths near the input of the system being particularly small. These smaller bit widths imply narrower adders, and smaller hardware. Smaller hardware translates into lower energy consumption. Throughput is a consequence of the width of the largest adder in the system, so lower bit widths at the output make for faster, higher performance systems.

5.4 Summary

The lifting implementation is a polyphase implementation of the filter bank. The inherent orthogonality in the lifting structure makes it more immune to coefficient quantization compared to traditional convolution methods. This chapter investigated various quantization schemes for the lifting coefficients of the biorthogonal 9/7 wavelets. Coefficient quantization was accomplished such that the magnitude responses of the quantized analysis filters were minimally impacted. The MUA-LSGC quantization method yields optimal, quantized lifting coefficients for the biorthogonal 9/7 wavelet filters. This method began with the unquantized, rational coefficients previously reported. These coefficients offer the best image compression performance (in terms of PSNR) with a small, fast hardware implementation. The energy required by MUA-LSGC to compute the DWT of an image is lower than any of the quantized coefficient designs that began with the irrational lifting coefficients.

The next chapter compares the two TDC implementation approaches introduced in Chapters 3-5: convolution and lifting.

Chapter 6

Convolution versus Lifting

The last three chapters addressed the hardware implementation of the DWT using two different approaches. The first approach, called the "convolution" approach, uses a filter bank for computing the DWT, and can employ either a non-polyphase or polyphase structure. The best image compression performance for a hardware implementation based on the convolution approach was obtained in Chapter 3 by using a cascade form for the filters, and using z_1 compensation for filter coefficient quantization. In Chapter 4 the throughput of the convolution-based filter bank was increased by using a polyphase structure for the filter bank. The second approach to computing the DWT is "lifting". This approach requires fewer computations than convolution and also operates in polyphase. In Chapter 5, optimal quantized values for two sets of lifting coefficients were arrived at for good image compression performance. This chapter compares the best non-polyphase, polyphase and lifting implementations, and evaluates them on the basis of image compression and hardware performance.

6.1 Candidates

Coefficient quantization plays a crucial role in the image compression performance of both—convolution and lifting implementations of the DWT. The filter and filter bank structures influence hardware metrics such as throughput, latency and power consumption of the implementation. The best approaches to coefficient quantization preserve the two important filter bank properties, no-distortion MSE and deviation-at-DC, for image compression performance. We now compare the convolution approach and the lifting approach, using the "best" quantized coefficient set obtained for each approach. The "best" set of quantized coefficients is the set that requires the least number T of SPT terms, while retaining reasonable image compression performance. Here "reasonable" is defined by the absence of checkerboarding in the compressed image and PSNR degradation of no more than 0.1dB at compression ratios ranging from 1:1 to 100:1.

6.1.1 Convolution Approach

6.1.1.1 Non-polyphase Implementation

The convolution approach requires quantization of the highpass and lowpass analysis filter coefficients. As seen in Chapter 3, the best quantization technique for biorthogonal filter coefficients is the z_1 compensation method. Chapter 3 illustrated the performance of a set of quantized coefficients with $T = 32$. Although 32 SPT terms provided reasonable performance at high compression ratios, it is too restrictive for reasonable performance under perfect reconstruction (1:1). To improve performance at 1:1, an additional 10 SPT terms are allocated to coefficients of the second cascade section in the analysis filters. The quantized coefficients and their SPT representations are listed in Table 6.1.

Quantized coefficient values are arrived at with z_1 compensation, with a_3 being quantized first, and quantized values of a_1 and a_2 used to offset changes in the magnitude response of

Table 6.1: Optimal quantized coefficients for convolution-based implementation ($T = 42$).

	$H_9'(z)$		$F_7'(z)$	
Section 1	1	001	1	001
	4	100	4	100
	6	110	6	110
	4	100	4	100
	1	001	1	001
Section 2	1	0001.000000000000	1	0001.0000000000
	-4.630615234375	$0\bar{1}00.\bar{1}0\bar{1}000\bar{1}01001$	-3.36962890625	$\bar{1}001.0100010\bar{1}0\bar{1}$
	9.59765625	$1010.\bar{1}010\bar{1}0010000$	1	0001.0000000000
	-4.630615234375	$0\bar{1}00.\bar{1}0\bar{1}000\bar{1}01001$		
	1	0001.000000000000		
K	0.0390625	0.0000101	-0.0625	(simple shift)
G	-		-	

the lowpass branch arising from quantization of a_3.

6.1.1.2 Polyphase Implementation

The non-polyphase structure suffers from wasted computations and low throughput. A polyphase implementation can be used to increase the throughput of the filter bank. The cascade-poly structure derived in Chapter 4, (Figure 4.5) can be used to implement the $T = 42$ cascade coefficients in Table 6.1. This structure combines the increased throughput of the polyphase structure with the superior compressed image quality of the cascade structure and z_1 compensated coefficients.

6.1.2 Lifting Approach

6.1.2.1 Irrational Coefficients

The lifting approach requires quantization of six lifting coefficients. Three coefficient quantization techniques were described in Chapter 5 for quantization of irrational lifting coefficients

obtained from factoring the biorthogonal 9/7 analysis polyphase matrix. The best quantized coefficient set was obtained using the *Exhaustively Searched Allocation* (ESA) method. This method examined all possible allocations of a given T across the six lifting coefficients, and chose the allocation that yielded the best quantized match to the unquantized lowpass and highpass magnitude responses. The ESA coefficients, shown in Table 5.2, required $T = 19$ and are chosen to represent the lifting approach using irrational coefficients.

6.1.2.2 Rational Coefficients

Chapter 5 discussed three methods for quantizing the rational lifting coefficients for the biorthogonal 9/7 that were reported previously. In addition to giving comparable image compression performance as the irrational coefficients, these coefficients also have a convenient SPT representation for implementation in hardware. Coefficients that have finite SPT representation, are represented exactly in hardware without approximation. The best compression performance was obtained using *Mostly Uniform Allocation* with *Lumped Scaling* and *Gain Compensation* (MUA-LSGC). With lumped scaling, the $\sqrt{2}$ factors in the scaling coefficients were combined to result in a single scaling factor of either 2, $\frac{1}{2}$ or 1 at the end of one level of 2D filtering. Gain compensation is used to ensure near perfect reconstruction of the DC component of the image. The MUA-LSGC coefficients, shown in Table 5.2, required $T = 21$ and are chosen to represent the lifting approach using rational coefficients.

6.2 Results

The two convolution designs and the two lifting designs were used to compute the five-level, non-expansive, symmetric extension DWT of three eight-bit grayscale images from the ITU standard digitized image set [37]. Perfect reconstruction and three compression ratios (8:1, 32:1 and 100:1) are examined.

Digital hardware for one level of analysis for each structure was implemented on an Altera

Table 6.2: PSNR and hardware performance of convolution and lifting based implementations.

		Unquant.	Convolution		Lifting	
			Non-poly (T=42)	Poly (T=42)	ESA (T=19)	MUA-LSGC (T=21)
Aerial2	1:1	58.75	58.70	58.70	58.76	58.80
	8:1	32.76	32.75	32.75	32.76	32.76
	32:1	27.99	27.98	27.98	27.99	27.98
	100:1	24.79	24.86	24.86	24.78	24.77
Bike	1:1	58.70	58.63	58.63	58.70	58.75
	8:1	32.83	32.81	32.81	32.82	32.82
	32:1	24.81	24.85	24.85	24.80	24.81
	100:1	21.24	21.42	21.42	21.21	21.20
Woman	1:1	58.68	58.62	58.62	58.68	58.74
	8:1	33.68	33.59	33.59	33.69	33.69
	32:1	27.29	27.43	27.43	27.28	27.28
	100:1	24.51	24.56	24.56	24.48	24.48
size (#logic elements)			1780	1913	2284	2123
f_{max}(MHz)			56.38	93.63	72.59	63.63
Latency(clocks)			15	26	51	49
Energy (mJ)			7.89	5.36	6.27	5.79
approx format, s_l			(29, -19)	(29, -19)	(41, -32)	(39, -30)
details format, d_l			(24, -15)	(24, -15)	(40, -31)	(39, -29)

Apex EP20K1000EFC672-1X FPGA. A multiplierless architecture is used; computations are organized in a fully pipelined tree of carry save adders with a ripple carry adder at the base. The Quartus II v2.1 software package is used for synthesis, placement and routing of structural VHDL descriptions of the architectures. The fixed-point formats are chosen to avoid truncation and round-off, given the quantized values of the coefficients.

6.2.1 Compression Performance

Table 6.2 compares the PSNR performance of the four implementations. All implementations give PSNR values, no more than 0.1dB worse than the unquantized PSNR values. The lifting implementations require fewer T for achieving this PSNR performance compared to the

Table 6.3: Filter properties for convolution and lifting based implementations.

Filter Property	Unquantized	Convolution		Lifting	
		Non-poly T=42	Poly T=42	ESA T=19	MUA-LSGC T=21
Deviation at DC	0	-2.2e-5	-2.2e-5	2.0e-5	3.9e-7
No Distortion MSE	0	5.0e-9	5.0e-9	5.8e-11	1.4e-14
Alias Cancellation MSE	0	0	0	0	0
Accuracy	4/4	4/4	4/4	0/0	0/0
Smoothness (analysis)	1.41	1.41	1.41	1.31	1.52
Smoothness (synthesis)	2.12	2.12	2.12	1.96	2.04
Coding Gain	9.71	9.70	9.70	9.70	9.70

convolution implementations. This is in part due to the fact that the lifting implementation has fewer coefficients to quantize than the convolution implementation. The orthogonality of the lifting structure also ensures high PSNR values in the presence of coefficient quantization.

Filter properties for the four implementations are shown in Table 6.3. The lifting implementations have better no-distortion-MSE and deviation-at-DC characteristics than the convolution implementations. This explains the better performance of lifting compared to convolution at perfect reconstruction (1:1) and low compression ratios. The two lifting implementations have lower values for smoothness of the analysis filters. Smoothness is related to the number of zeros at $z = 1$ for the analysis lowpass filter, which in turn is related to the vanishing order of the wavelet. Vanishing order impacts the compaction property of the wavelets; higher the number of zeros at $z = 1$, higher is the vanishing order and consequently better is the compaction property and compression efficiency of the wavelet. The lifting implementations have a lower vanishing order than the convolution implementations, and hence at high compression ratios (100:1), convolution implementations yield better PSNR than lifting implementations.

6.2.2 Hardware Performance

Table 6.2 depicts the hardware performance of the four implementations. The maximum possible operating frequency f_{max} is obtained via timing analysis and indicates system throughput. The energy required by the filters to perform the computations of a five-level, two-dimensional DWT for a 512×512 image is calculated as described in Appendix A. The convolution designs resulted in smaller hardware and smaller latencies compared to lifting, despite the fact that lifting required lower T than convolution. This demonstrates that although T can serve as an estimate for hardware cost before FPGA implementation, the actual hardware cost depends on a number of factors such as bit widths of the filter coefficients and the architecture of the filter bank. The lifting structure employs a number of filters that are in cascade, and also have wider output bit widths for the approximate and detail coefficients. These factors contribute to making the lifting structure bigger and slower.

Lifting and convolution polyphase implementations have higher throughput than the convolution non-polyphase implementation. The lifting structure exhibits lower throughput than the convolution polyphase implementation due to the large bit widths of some of the lifting coefficients; throughput could be improved at the expense of latency by further pipelining the hardware. The convolution polyphase implementation also requires lower energy than any of the other implementations for computation of the DWT. This is because compared to the non-polyphase implementation, it performs half the number of computations resulting in lower average power consumption. The larger hardware required for lifting structures means that they consume slightly more energy compared to the convolution polyphase structure. The lifting implementations resulted in approximation and detail coefficients with larger bit widths compared to the convolution implementation. This is because in the convolution implementation, the output bit width depends only on the widest coefficient in the filter; in lifting, the output bit width is proportional to the sum of the bit widths of all lifting coefficients. Bit width of output coefficients becomes a concern for multiple levels of DWT decomposition, where growing bit widths at the end of each stage warrants coefficient

truncation between levels, leading to image compression performance degradation.

6.3 Summary

This chapter compared the convolution and lifting approaches to computing the DWT. The optimal quantized coefficients for each implementation were selected to synthesize non-polyphase, polyphase and lifting structures on an FPGA. All these designs resulted in PSNR performance that was within 0.1dB of the unquantized PSNR performance at all compression ratios. However, the cascade-polyphase structure resulted in smallest hardware with lowest energy consumption and highest throughput.

Chapter 7

Conclusion

7.1 Concluding Remarks

This thesis treated the issue of filter coefficient quantization and filter bank structure for a fast, multiplierless implementation of the biorthogonal 9/7 DWT with image compression applications. Quantization of filter bank coefficients is known to impact perfect reconstruction properties and degrade image compression performance of the filter bank while filter bank structure determines hardware metrics such as throughput, latency and power-consumption for the implementation. This work determines which filter bank structure is best suited for an FPGA implementation, and arrives at an optimal set of quantized coefficients for this structure.

Properties of the filter bank that are critical for image compression performance were identified in chapter 3. The "no-distortion-MSE" and "deviation-at-DC" properties determine PSNR values for the compressed image. Additionally, it is important for the highpass analysis filter to have magnitude response of zero at $f = 0$, so as to avoid checkerboarding in the compressed image. The cascade form was chosen for filter implementation, since it preserves zeros at $z = 1$ for the highpass analysis filter. Two techniques—"gain compensation" and

"zero compensation" were developed for quantization of cascade filter coefficients. These techniques attempt to preserve the no-distortion-MSE and deviation-at-DC properties of the filter bank by quantizing the synthesis filters so that it offsets the degradation caused by quantization of the analysis filters. The best image compression performance was obtained using a cascade form for the filters with quantized "z_1 compensation" coefficients.

Chapter 4 looked at polyphase structures that aimed to improve efficiency of the traditional filter bank by avoiding wasted computations. A cascade-polyphase structure was derived, that combined the fast throughput of the polyphase structure with the superior compression performance of the cascade form and "z_1 compensation" coefficients, resulting in fast, low-power implementation. A faster polyphase alternative can be obtained by convolving out the coefficients of the cascaded filters, and implementing the resulting coefficients in a direct-polyphase structure, but this implementation has higher power requirements compared to the cascade-polyphase structure. Results in chapter 4 also showed that an FPGA implementation of the polyphase structure cannot achieve the theoretical doubled throughput that one expects in a polyphase structure. This is because throughput not only depends on the polyphase implementation of filters, but also on various FPGA timing parameters and their interactions.

Chapter 5 looked at the lifting approach to computing the DWT. Since its development for building new wavelets, lifting has found favor for implementation of the DWT, because of its need for fewer computations, ability to support in-place computations and polyphase operation. Two sets of lifting coefficients were examined—an irrational set obtained from factorization of the biorthogonal 9/7 wavelet polyphase matrix and a rational set proposed previously. Three quantization methods were developed to quantize the irrational and rational coefficients each. The quantization methods were driven by the no-distortion-MSE and deviation-at-DC properties identified in chapter 3 to obtain optimal quantized coefficients. The lifting structure is more immune to coefficient quantization than the convolution based polyphase and non-polyphase structures due to its inherent orthogonality. The best quantized set was obtained starting with the rational lifting coefficients and using lumped scaling

and gain compensation to quantize coefficients.

In chapter 6, the convolution and lifting approaches to computing the DWT were compared. Best quantized coefficient sets were chosen to represent non-polyphase, polyphase and lifting implementations. The cascade-polyphase structure using z_1 compensated coefficients outperformed non-polyphase and lifting implementations in terms of throughput and power consumption. This was despite the fact that lifting was theoretically shown to require fewer computations in terms of multiplications and additions compared to the convolution approach. This goes to show that when implementing on an FPGA, a number of factors such as implementation structure and bit width of quantized coefficients play a role in determining the final throughput and energy requirements of the implementation. Though the lifting structure only uses two-tap symmetric filters, these filters were arranged in a cascaded manner. Growing bit widths of intermediate coefficients, meant that additional bits had to be kept track of after every filtering operation resulting in big, slow hardware.

To summarize, the cascade-polyphase structure is the best choice for implementing the DWT on FPGA, in terms of throughput and energy. Coefficients for this structure can be quantized using the z_1 compensation method. If faster hardware with lower latency is desired for real-time applications, the cascade section coefficients can be convolved out to obtain direct form coefficients, to be used in a direct-polyphase implementation. This however comes at the cost of higher power-consumption. In applications where perfect reconstruction of images at 1:1 is essential, lifting is a better choice than convolution. The orthogonality of the lifting structure provides good PSNR performance even under coefficient quantization at the cost of bigger, slower hardware.

Work done in this thesis also shows that the number of SPT terms T, used for the filter coefficients is not the only indicator of hardware cost. Within a given implementation (i.e. lifting or polyphase), T serves as an estimate for hardware cost before implementation. However, final hardware cost, measured in terms of number of logic elements, depends on the structure being implemented (lifting, polyphase etc.) and bit widths of filter coefficients

and intermediate signals.

7.2 Future Work

The focus of this thesis was on the filter structure and coefficient quantization aspect of DWT implementation. Considerable amount of work has been done in the hardware architecture aspect of 2D DWT implementation, and it continues to be a field for future research. Some of the areas for future work are:

1. Further improvements in hardware performance can be obtained by addressing hardware architecture issues such as pipelining, placement and routing and memory access. FPGA implementations of the cascade-polyphase and the rational lifting implementation can be tweaked and optimized to improve hardware performance.

2. A complete 2D DWT implementation will need to address issues related to memory access for reading and writing of DWT coefficients and intermediate results. A good implementation will require few memory accesses, and use fast internal cache memory for intermediate results.

3. Multiple levels of DWT computation present the problem of growing signal bit widths. Starting with 8-bit image data, the input to each successive DWT level will have wider bit widths, making it impractical to save all bits of precision in memory till the final decomposition level. Intermediate DWT coefficients will have to be truncated after every level, so that data read from and written to memory will have fixed bit widths. For the lifting implementation, where bit widths grow after every filter within a single lifting stage, intermediate signals may have to be truncated within a level. Truncation of DWT coefficients and other intermediate values during the computation of the DWT further impacts PSNR performance, and presents another interesting topic for further study.

Appendix A

Energy Computation for Filter Implementations

Energy numbers reported in this thesis estimate that required for the computation of a 5-level DWT of a 512×512 image. All energy estimates are made at an operating frequency of 40MHz, unless specified otherwise. Hardware for the filters that comprise the analysis side of the convolution or lifting based implementation is synthesized to obtain a system that operates at 40MHz. The hardware is then used to compute the 5-level DWT for 500 blocks of random patterns along with 120 other patterns useful for verification; each block is 512×512 in size. The Quartus II version 2.1 software package reports an estimate of average power throughout the run. The processing time in milliseconds for computing the 5-level DWT of the image is then computed as follows:

$$PT(ms) \quad = \quad \sum_{n=0}^{4} 3 \cdot \frac{M}{2^{n+1}} \cdot \left(\frac{M}{2^n} + M_{\text{ext}} + L \right) \cdot \frac{1}{f} \cdot 10^{-3} \tag{A.1}$$

where, $M \times M$ is the image size in pixels ($M = 512$ here), M_{ext} is the number of pixels required for symmetric extension ($M_{\text{ext}} = 9$ here), L is the latency in clock cycles, f is the frequency in Hz at which samples are fed to the analysis side and PT is the processing time in milli-seconds.

The processing time is then multiplied by average power to get an estimate of energy consumed by the hardware implementation.

Bibliography

[1] *ITU-T Recommendation T.800. JPEG2000 image coding system - Part 1*, ITU Std., July 2002. [Online]. Available: http://www.itu.int/ITU-T/

[2] A. Skodras, C. Christopoulis, and T. Ebrahimi, "The JPEG2000 still image compression standard," *IEEE Signal Processing Mag.*, vol. 18, no. 5, pp. 36–58, September 2001.

[3] S. Traferro, F. Capparelli, F. Piazza, and A. Uncini, "Efficient allocation of power of two terms in FIR digital filter design using tabu search," in *Proc. IEEE Int'l. Symposium on Circuits and Systems*, vol. 3, 1999, pp. 411–414.

[4] Y. Lim, R. Yang, D. Li, and J. Song, "Signed power-of-two term allocation scheme for the design of digital filters," *IEEE Trans. Circuits Syst. II*, vol. 46, no. 5, pp. 577–584, May 1999.

[5] Y. Lim and S. Parker, "FIR filter design over a discrete power-of-two coefficient space," *IEEE Trans. Acoust., Speech, Signal Processing*, vol. ASSP-31, pp. 583–591, June 1983.

[6] N. Cho and S. Lee, "Optimal design of finite precision FIR filters using linear programming with reduced constraints," *IEEE Trans. Signal Processing*, vol. 46, no. 1, pp. 195–199, January 1998.

[7] J. Mao, S. Chan, W. Liu, and K. Ho, "Design and multiplier-less implementation of a class of two-channel PR FIR filterbanks and wavelets with low system delay," *IEEE Trans. Signal Processing*, vol. 48, no. 12, pp. 3379–3394, December 2000.

[8] A. Akansu, "Multiplierless PR quadrature mirror filters for subband image coding," *IEEE Trans. Image Processing*, vol. 5, no. 9, pp. 1359–1363, September 1996.

[9] Y. Chen, S. Oraintara, T. D. Tran, K. Amaratunga, and T. Q. Nguyen, "Multiplierless approximation of transforms using lifting scheme and coordinate descent with adder constraint," in *IEEE Proc. Int'l Conf. Acoust., Speech, Sig. Proc. (ICASSP)*, vol. 3, 2002, pp. 3136–3139.

[10] B. E. Usevitch and C. L. Betancourt, "Fixed-point error analysis of two-channel perfect reconstruction filter banks with perfect alias cancellation," *IEEE Trans. Circuits Syst. II*, vol. 46, no. 11, pp. 1437–1440, November 1999.

[11] H. Kim and C. C. Li, "Lossless and lossy image compression using biorthogonal wavelet transforms with multiplierless operations," *IEEE Trans. Circuits Syst. II*, vol. 45, no. 8, pp. 1113–1118, August 1998.

[12] M. Adam and F. Kossentini, "Reversible integer-to-integer wavelet transforms for image compression: performance evaluation and analysis," *IEEE Trans. Image Processing*, vol. 9, no. 6, pp. 1010–1024, 2000.

[13] L. Cheng, D. Liang, and Z. Zhang, "Popular biorthogonal wavelet filters via a lifting scheme and its application in image compression," *IEE Proc. Vision, Image and Signal Processing*, vol. 150, no. 4, pp. 227–232, August 2003.

[14] W. Sweldens, "The lifting scheme: A new philosophy in biorthogonal wavelet constructions," in *Proc. SPIE Int. Soc. Opt. Eng.*, vol. 2569, 1995, pp. 68–79.

[15] ——, "The lifting scheme: A custom-design construction of biorthogonal wavelets," *Appl. Comput. Harmon. Anal.*, vol. 3, no. 2, pp. 186–200, April 1996.

[16] ——, "The lifting scheme: A construction of second generation wavelets," *SIAM J. Math. Anal.*, vol. 29, no. 2, pp. 511–546, March 1998.

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

[18] D. Tay, "A class of lifting based integer wavelet transform," in *Proc. IEEE Int'l Conference on Image Processing*, vol. 1, 2001, pp. 602–605.

[19] Z. Guangjun, C. Lizhi, and C. Huowang, "A simple 9/7-tap wavelet filter based on lifting scheme," in *Proc. IEEE Int'l Conference on Image Processing*, vol. 2, 2001, pp. 249–252.

[20] K. A. Kotteri, A. E. Bell, and J. E. Carletta, "Design of multiplierless, high-performance, wavelet filter banks with image compression applications," *IEEE Trans. Circuits Syst. I*, vol. 51, no. 3, pp. 483–494, March 2004.

[21] ——, "Quantized FIR filter design using compensating zeros," *IEEE Signal Processing Mag.*, vol. 20, no. 6, pp. 60–67, November 2003.

[22] ——, "Quantized FIR filter design: A collaborative project for digital signal processing and digital design courses," in *Proc. ASEE Annual Conference and Exposition*, June 2004.

[23] ——, "Polyphase structures for multiplierless biorthogonal filter banks," in *Proc. IEEE Int'l Conf. Acoustics, Speech, and Signal Processing*, May 2004.

[24] S. Barua, K. A. Kotteri, A. E. Bell, and J. E. Carletta, "Optimal quantized lifting coefficients for the 9/7 wavelet," in *Proc. IEEE Int'l Conf. Acoustics, Speech, and Signal Processing*, May 2004.

[25] M. J. T. Smith and S. L. Eddins, "Analysis/synthesis techniques for subband image coding," *IEEE Trans. Acoust., Speech, Signal Processing*, vol. 38, no. 8, pp. 1446–1456, August 1990.

[26] S. Rout, "Orthogonal vs. biorthogonal wavelets for image compression," Master's thesis, Virginia Polytechnic Institute and State University, 2003.

[27] B. E. Usevitch, "A tutorial on modern lossy wavelet image compression: foundations of JPEG2000," *IEEE Signal Processing Mag.*, vol. 18, no. 5, pp. 22–35, September 2001.

[28] M. Antonini, M. Barlaud, P. Mathieu, and I. Daubechies, "Image coding using wavelet transform," *IEEE Trans. Image Processing*, vol. 1, no. 2, pp. 205–220, April 1992.

[29] A. Said and W. A. Pearlman, "A new, fast, and efficient image codec based on set partitioning in hierarchical trees," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 6, no. 3, pp. 243–250, June 1996.

[30] G. Strang and T. Nguyen, *Wavelets and Filter Banks*, 1st ed. Wellesley MA: Wellesley-Cambridge Press, 1996.

[31] J. Katto and Y. Yasuda, "Performance evaluation of subband coding and optimization of its filter coefficients," in *Proc. SPIE Vis. Comm. Image Proc.*, vol. 1605, Boston, MA, November 1991, pp. 95–106.

[32] C.-T. Huang, P.-C. Tseng, and L.-G. Chen, "VLSI architecture for discrete wavelet transform based on B-spline factorization," in *IEEE Workshop on Sig. Proc. Sys.*, Aug 2003, pp. 346–350.

[33] P. P. Vaidyanathan, *Multirate systems and filter banks*. Englewood Cliffs, N.J.: Prentice-Hall, Inc., 1993.

[34] I. Daubechies, *Ten Lectures on Wavelets*, 1st ed. Philadelphia PA: SIAM, 1992.

[35] N. Benvenuto, M. Marchesi, and A. Uncini, "Applications of simulated annealing for the design of special digital filters," *IEEE Trans. Signal Processing*, vol. 40, no. 2, pp. 323–332, February 1992.

[36] A. Corona, M. Marchesi, C. Martini, and S. Ridella, "Minimizing multimodal functions of continuous variables with the simulated annealing algorithm," *ACM Trans. Math. Software*, vol. 13, no. 3, pp. 262–280, September 1987.

[37] *ITU-T Recommendation T.24. Standardized digitized image set*, ITU Std., June 1998. [Online]. Available: http://www.itu.int/ITU-T/

Vita

Kishore A. Kotteri was born on June 1, 1976 in Mumbai, India. He graduated with distinction from Victoria Jubilee Technical Institute, University of Mumbai in July 1997 with a Bachelor of Engineering degree in Electronics Engineering. Faced with the option of joining the industry or pursuing higher studies after graduation, his passion for programming led him to join TATA Consultancy Services (TCS) as a Systems Engineer. During his stint with TCS, he was involved in many projects, in various capacities, rising up the ranks to IT Analyst. Four years later, his desire for graduate studies rekindled, he enrolled in the Masters program in Electrical Engineering at Virginia Tech. During his studies at Virginia Tech he worked as a research assistant in the Digital Signal Processing and Communications Laboratory.

Kishore's interests lay in the fields of digital signal processing, image processing and communications. He is an avid reader and cricket enthusiast.