VLSI ARCHITECTURES FOR THE 4-TAP AND 6-TAP 2-D DAUBECHIES WAVELET FILTERS USING ALGEBRAIC INTEGERS

A Thesis

Presented to

The Graduate Faculty of The University of Akron

In Partial Fulfillment

of the Requirements for the Degree

Master of Science

Shiva Kumar Madishetty

August, 2013

VLSI ARCHITECTURES FOR THE 4-TAP AND 6-TAP 2-D DAUBECHIES WAVELET FILTERS USING ALGEBRAIC INTEGERS

Shiva Kumar Madishetty

Thesis

Approved:

Advisor
Dr. Arjuna Madanayake

Faculty Reader
Dr. George C. Giakos

Faculty Reader
Dr. Hamid R. Bahrami

Faculty Reader
Dr. Dale H. Mugler

Department Chair
Dr. Alex De Abreu Garcia

Accepted:

Dean of the College
Dr. George K. Haritos

Dean of the Graduate School
Dr. George R. Newkome

Date

ABSTRACT

Discrete wavelet transforms (DWTs) are of special interest in signal and image processing due to their capability of signal decomposition, denoising, and event detection. This thesis proposes a novel algebraic integer (AI) based multi-encoding of Daubechies-4 and -6 2-D wavelet filters having error-free integer-based computation. Digital VLSI architectures employing parallel channels are proposed, physically realized and tested. The multi-encoded AI framework allows a multiplication-free and computationally accurate architecture. It also guarantees a noise-free computation throughput the multi-level multi-rate 2-D filtering operation. A *single* final reconstruction step (FRS) furnishes filtered and down-sampled image outputs in fixed-point, resulting in low levels of quantization noise.

Significant SNR and PSNR improvements were observed in favour of AI-based systems, when compared to 8-bit fixed-point schemes (six fractional bits). The designs are physically implemented for a 4-level 2-D decomposition using Daubechies-4 and -6 4-level VLSI architectures on a Xilinx Virtex-6 vcx240t-1ff1156 FPGA device and verified on an FPGA chip using an ML605 platform. A 45 nm CMOS synthesis shows improved clock frequencies for a supply voltage of 1.1 V.

ACKNOWLEDGEMENTS

With great pleasure I pay deep sense of gratitude and heartfelt thanks to my advisor **Dr. H. L. P. Arjuna Madanayake**, Asst. Professor, Department of Electrical and Computer Engineering for the outstanding guidance, help, support, constant encouragement and motivation throughout the progress of this work. It was really a great experience working under him and his guidance, which was of immense help in my thesis work without which it would have been an unachievable task.

I profusely thank Dr. Renato J. Cintra, Universidade Federal de Pernambuco, Brazil for his invaluable collaborative support and help in my thesis work.

I would like to express my sincere thanks to Dr. Vassil Dimitrov, University of Calgary, AB, Canada and Dr. Dale Mugler, The University of Akron for their great technical support and help.

My gratitude and immense respect to my advisory committee, Department of Electrical and Computer Engineering for the Teaching Assistantship, infrastructure, all other essential facilities and encouragement given to me during the project work.

I am thankful to my parents and friends for their love, motivation and continuous support. I thank THE ALMIGHTY for being with me throughout.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER I

INTRODUCTION

The field of discrete wavelet transforms (DWT) has been attracting substantial interest in part due to the wavelet analysis being capable of decomposing a signal into a particular set of basis functions equipped with good spectral properties [1–4]. Wavelet analysis has been used to detect system non-linearities by making use of its localization feature [5]. DWT-based multi-resolution analysis leads to both time and frequency localization [4, 6–11].

Indeed, wavelet filter banks establish a strong support for many signal processing systems [12–16]. Wavelets are employed in numerical analysis [17, 18], real-time processing [17], image compression and reconstruction [3, 15, 19–23, 23–28], pattern recognition [17], biomedicine [18, 20], approximation theory, computer graphics [29, 30] and image, video coding standards (H.265) [4, 22, 31–35]. Following the adoption of the bi-orthogonal 2.2 wavelet filters in the JPEG2000 standard [3, 6, 19, 28], much research effort has been employed on reducing computational and circuit complexities of DWT hardware architectures in VLSI systems [2, 17, 19, 36–42].

A particular class of DWT are the Daubechies wavelets [43]. They are well-suited and commonly used in image compression applications [3,12,24–26,37]. Herein

we refer to the Daubechies wavelets generated from 4- and 6-tap filter banks as Daub-4 and -6 wavelets, respectively. In particular, whereas the Daub-4 wavelets are often employed in applications where the signals are smooth and slowly varying, the Daub-6 wavelets are used for signals bearing abrupt changes, spikes, and having high undesired noise levels [17]. Daub-4 wavelets can be highly localized to smooth [2,26] and Daub-6 wavelets have found applications in medical imaging, such as wireless capsule endoscopy where images of fine details are regarded important [17, 43, 44].

Since wavelets can be associated to specific filter banks, practical wavelet analysis is achieved by means of sub-band coding [13, 34, 43, 45]. Sub-band coding is a basic filtering principle which splits a given signal in several frequency bands for subsequent encoding [8, 14]. In particular, 2-D multi-resolution analysis is obtained via sub-band coding [43, 45, 46].

Daubechies-4 and -6 tap filter coefficients are irrational numbers and cannot be represented *exactly* in standard finite precision number systems such as the two's complement fixed-point format. Therefore, fixed-point representations incur errors when employed in arithmetic processors. These errors, due to quantization, overflow, and underflow, propagate through the entire process of wavelet decomposition. Hence the results of the wavelet analysis possess a reduced signal-to-noise ratio.

Conventional 2-D Daubechies-4 and -6 filter banks employ 1-D filters as their building block. The 1-D Daubechies-4 filters are repeatedly applied to row- and column-wise operations to yield 2-D filtering [45]. This aggravates the above mentioned noise issue.

1.1 Research Publications

The following is a list of conference and journal research papers where this research has been published or in press to publish.

1.1.1 Conference Papers

1. **Madishetty, Shiva**; Madanayake, A.; Cintra, R.J.; Mugler, Dale; Dimitrov, V.S., "Error-free VLSI architecture for the 2-D Daubechies 4-tap filter using algebraic integers," Circuits and Systems (ISCAS), 2012 IEEE International Symposium on , vol., pp.1484,1487, 20-23 May 2012 doi: 10.1109/ISCAS.2012.6271528

2. **Madishetty, S. K.**; Madanayake, A.; Cintra, R. J. ; Tay, D. B. H.; Selvaraju, Murugesan; , "VLSI Implementation of Rationalized 1-D Orthogonal Daubechies and Bi-orthogonal wavelet filters" (In Progress, not submitted)

1.1.2 Journal Papers

1. **Madishetty, S.K.**; Madanayake, A.; Cintra, R.J.; Dimitrov, V.S.; Mugler, D.H., "VLSI Architectures for the 4-Tap and 6-Tap 2-D Daubechies Wavelet Filters Using Algebraic Integers," Circuits and Systems I: Regular Papers, IEEE Transactions on , vol.60, no.6, pp.1455,1468, June 2013 doi: 10.1109/TCSI.2012.2221171

2. **Madishetty, S. K.**; Madanayake, A.; Cintra, R. J.; Dimitrov, V. S.; Mugler, D. H.; , "Algebraic Integer Architecture with Minimum Adder count for the 2-D

Daubechies 4-Tap wavelet Filter Banks " Journal of Multidimensional systems and signal processing. (Accepted, In press)

3. **Madishetty, S. K.**; Madanayake, A.; Cintra, R. J.; Dimitrov, V. S.; , "Precise Algebraic Integer Architecture with Low Adder count for the 1-D/ 2-D Daubechies 6-Tap wavelet Filter Banks" Circuits and Systems I: Regular Papers, IEEE Transactions. (Submitted, Under Review)

4. Arjuna Madanayake; Renato J. Cintra; Nilanka Rajapaksha; Uma Potluri; **Shiva Madishetty**;Vassil Dimitrov; Fabio Bayer and Khan Wahid "Advanced VLSI Signal Processing Circuits using Number Theoretic Methods" (Accepted for Extended Abstract)

1.2 Contributions from External Collaborators

This research work has significant technical contributions from Dr. Madanayake's collaborators Dr. Renato J. Cintra from Universidade Federal de Pernambuco, Brazil and Dr. Vassil S. Dimitrov from University of Calgary, AB, Canada.

To emphasize collaborator's contributions in this work, the following shows a list of exhaustive numerical search results used in this work.

1. In Chapter 3, Section 3.3 discusses the final reconstruction step (FRS) using canonical signed digit (CSD) encoding and expansion factor methods (EFM)for

Daubechies 4-tap and 6-tap wavelet filter banks. Tables 3.3, 3.4 and 3.5 provide the data from the contributions.

2. In Chapter 5, Section 5.3 provides numerical search results for filter parametrization and optimization in Table 5.1.

3. In Chapter 5, Section 5.4 describes the FRS using CSD for optimized 1-D/ 2-D Daubechies 6-tap wavelet filter bank leading to low adder count designs. Table 5.3 offers the CSD encoding for 8-bit approximations of algebraic integer (AI) bases.

1.3 Thesis Outline

The rest of this research work is organized as follows:

Chapter 2 reviews fundamental algorithm behind wavelet decomposition known as subband coding and details significant research efforts on AI based DWTs. Also, it presents our proposed encoding scheme that addresses the computational noise injection issue of the wavelet decomposition.

Chapter 3 provides exhaustive analysis of AI based multi-encoded Daubechies 4-tap and 6-tap wavelet filter banks, final reconstruction step methods and FPGA implementation and results.

In chapter 4 we optimized the filter parameters to reduce arithmetic complexity and thereby yielding a low adder count architecture for 2-D Daubechies 4-tap filter bank. Exhaustive numerical search methods are used to optimize the filter parametrization and optimization.

Chapter 5 extends the optimized encoding to 1-D/ 2-D Daubechies 6-tap filter bank with FPGA implementation and results. The chosen filter optimization resulted in significant savings in adder counts as detailed in this chapter.

This thesis work has its conclusive remarks in Chapter 6 with feasible scope for future work to improve and stir innovative ideas in AI based wavelet filter implementation.

CHAPTER II

REVIEW OF SUBBAND CODING

2.1 Review of Sub-band Coding

Wavelet decomposition of input image data can be accomplished by sub-band coding. A 2-D finite impulse response (FIR) filter bank processes the input data resulting in an approximation and detail sub-images.

The input image $\mathbf{A}_{n-1}$ is of resolution $N \times N$ pixels; and it is input to a pair of low-pass (approximation) and high-pass (detail) filters $\mathbf{h}$ and $\mathbf{g}$, respectively. The filters operate column-wise on the image followed by dyadic down-sampling, i.e., only one of every two columns are retained. Then the same process is applied row-wise. The outputs are four sub-images $\mathbf{A}_n$, $\mathbf{Dv}_n$, $\mathbf{Dh}_n$, and $\mathbf{Dd}_n$, which represent the 2-D wavelet coefficients for the coarse approximation, vertical details, horizontal details, and diagonal details, respectively. This process is shown in Fig. 2.1 for one-level wavelet analysis via filter banks. Symbols $2\downarrow 1$ and $1\downarrow 2$ are used to denote the column-wise and row-wise down-sampling. respectively [47, pp. 6-26]. The resultant sub-images are all of size $N/2 \times N/2$, because of dyadic down-sampling.

These operations can be performed recursively [43,45]. The resulting approximation $\mathbf{A}_n$ can be re-submitted to the signal flow architecture shown in Fig. 2.1.

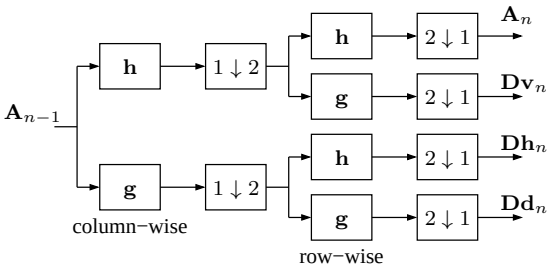


Figure 2.1: Diagram of a single application of the 2-D wavelet filter bank.

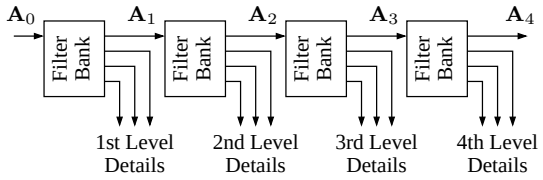


Figure 2.2: Recursive application of the 2-D wavelet filter bank.

As a result, after each iteration a coarser approximation can be achieved. Let the original image to be analyzed be denoted by $\mathbf{A}_0$. Fig. 2.2 shows the recursive diagram of the multi-level wavelet analysis. After each set of filter banks, a coarser approximation $\mathbf{A}_n$, $n = 1, 2, 3, 4$, is furnished. Each level also produces the detail information.

In this work, we focus on the computation of the coarser approximations $\mathbf{A}_n$, $n \geq 1$. The topmost branch of the signal flow shown in Fig. 2.1 computes the approximation data. Detail data $\mathbf{Dv}_n$, $\mathbf{Dh}_n$, and $\mathbf{Dd}_n$ are normally discarded or thresholded in data compression applications [43].

The 2-D FIR filter bank based on the Daub-4 and -6 filter bank is of particular relevance [2, 45]. Let the low-pass filter associate to these filter banks be

denoted as $\mathbf{h}^{(\text{Daub-4})}$ and $\mathbf{h}^{(\text{Daub-6})}$, respectively. These particular filters possess irrational quantities as shown below [2, 3, 6, 43, 48]:

$$\mathbf{h}^{(\text{Daub-4})} = \frac{1}{4\sqrt{2}} \left[1+\sqrt{3} \quad 3+\sqrt{3} \quad 3-\sqrt{3} \quad 1-\sqrt{3} \right]^{\top},$$

$$\mathbf{h}^{(\text{Daub-6})} = \frac{1}{16\sqrt{2}} \begin{bmatrix} 1+\sqrt{10}+\sqrt{5+2\sqrt{10}} \\ 5+\sqrt{10}+3\sqrt{5+2\sqrt{10}} \\ 10-2\sqrt{10}+2\sqrt{5+2\sqrt{10}} \\ 10-2\sqrt{10}-2\sqrt{5+2\sqrt{10}} \\ 5+\sqrt{10}-3\sqrt{5+2\sqrt{10}} \\ 1+\sqrt{10}-\sqrt{5+2\sqrt{10}} \end{bmatrix},$$

where the superscript $^{\top}$ denotes transposition.

2.2 The Problem of Fixed-Point Errors

Filter banks associated to Daubechies wavelets have irrational coefficients whose representation in fixed-point requires truncation or rounding off [2,6,16]. Such approximations introduce representation errors which propagate through a given filter bank. Moreover, the longer the required filter bank is, the greater the computational error may become. This process effects a lower obtained signal-to-noise of the resulting data.

2.3 Prior Art on AI based DWT

One way of addressing the computational noise injection is to employ a number representation based on algebraic integers (AI) [6]. Pioneered by Cozzens and Finkelstein [3, 49–51], algebraic integer (AI) quantization has been employed in several signal processing schemes, including wavelet and discrete cosine transform analysis [1, 17, 19, 29, 52–54].

In the AI representation, irrational numbers involved in the DWT process are encoded into integers associated to a given AI basis. The computing architecture is replaced with a parallel channel model [52]. After computation, resulting encoded numbers are mapped back into fixed-point representation in the final reconstruction step (FRS). The FRS is the only possible source of computational error in a given AI-based framework.

AI encoding can address the computational noise injection in wavelet analysis systems [6]. A significant advantage of the AI encoding is its capability of mapping the required irrational wavelet coefficients into vectors or arrays of integers. Therefore, wavelet decomposition can be performed without errors in a vectorial framework consisting exclusively of integer operations. Thus, the irrational coefficients of the Daubechies filters can be represented into integers, according to a selected AI basis [3, 6, 55].

The design of digital architectures for the 1-D Daub-4 and -6 filters were

pioneered by Wahid and Dimitrov in the recent past. Importantly, the 2-D architectures proposed by Wahid *et al.* [1–3, 6, 19] require intermediate reconstruction steps that map the AI encoded transform coefficients back into fixed-point format. These are 1-D DWT architectures that compute the 2-D DWT by repeated use of a 1-D AI-encoded architecture. Some AI-based 2-D Daubechies-4 implementations are archived in literature. In [2, 31], such architecture was realized on FPGA and VLSI technology for low-complexity low-power applications.

However, in all published AI-based architectures for the DWT, a reconstruction step is present between row-wise and column-wise computations. Such intermediate decoding-encoding operations lead to quantization noise being injected to the corresponding output 2-D image signal. In a sense, this defeats the purpose of AI encoding.

In order to address the above-mentioned noise injection problem, we propose a novel 2-D AI architecture for the computation of 2-D Daubechies-4 wavelet filters based on a new multi-encoding method for the subband coding of images. In the proposed architecture, all computations are entirely performed over the integers *without any FRS in intermediate calculations*. A single FRS decodes the resulting computations into fixed-point representation.

This approach could lead to arbitrarily low levels of uncorrelated and uncoupled quantization noise in the final output 2-D image. This intermediate reconstruction step is located after the first application of the transform (say, along rows) before submitting the resulting data to the next (say, column-wise) stage. In other

words, it is at the transposition stage between the application of the two series of 1-D transform. Such intermediate reconstruction step injects quantization noise and introduces transfer-function response errors. When multi-level decompositions are attempted, the problem is compounded because of repeated applications of the intermediate reconstruction stages at each level of filtering [3,6,19,31]. Errors incurred in the intermediate reconstructions mitigate the benefits of using AI encoding for 2-D multi-level DWTs.

This is an outstanding problem in the current literature which we identify and correct in the present contribution.

2.4 Proposed Encoding Scheme

We correct above described issue by proposing a multi-encoding method that possesses error-free computation across the 2-D decomposition levels. In our method, the reconstruction step appears only once, at the final level of decomposition and filtering [56]. Unlike the schemes described in [2,3,6,31], our scheme operates *entirely* over the AI representation—up to a single and final reconstruction block—without any intermediate reconstruction steps. Thus, the FRS is the *only* possible source of computational errors.

In view of the above, we propose a new AI-based architecture for sub-band coding of images using 2-D Daub-4 and -6 wavelet filters. The AI quantization approach leads to an architecture possessing a parallel channel structure [52]. Input

data is successively wavelet decomposed over several levels according to application requirements.

The single FRS employs constant coefficient multipliers based on canonical signed digit (CSD) representation, offering low circuit complexity. This architecture facilitates very low levels of uncorrelated and uncoupled quantization noise in the final decomposed image data.

CHAPTER III

AI BASED 4-TAP AND 6-TAP 2-D DAUBECHIES WAVELET FILTER BANKS

3.1 Introduction

In this chapter, we propose a new multi-encoding technique that achieves exact computation of multi-level 2-D Daubechies wavelet transforms using algebraic integer (AI) encoding. Compared to existing AI designs in literature [1–3,6,17,19,31], the proposed design can compute wavelet image approximations entirely over integer fields and with a *single* FRS in a purely AI based 2-D architecture. The design avoids the need of intermediate reconstruction steps.

Moreover, the proposed architecture is sought to be multiplier-free. Such design facilitate accuracy, speed, relatively smaller area on chip as well as cost of design. The new design is multi-encoded and multi-rate, operating over AI with no intermediate reconstruction steps. In this framework, error-free computations can be performed until the final FRS. Our architecture emphasizes on quality of output image and speed by trading complexity and power consumption for accuracy.

This chapter unfolds as follows. Section 3.2 translates the the mathematical formalism of AI encoding into the 2-D sub-band coding context. Wavelet sub-band coding using multi-encoding with AI bases are provided for multi-level decompo-

sition, considering both Daub-4 and -6 filter banks. The final reconstruction step (FRS) procedure for the proposed analyses are described in Section 3.3. Based on the expansion factor method [57, p. 274], alternative FRS schemes were also sought for the Daub-6 case. Field programmable gate array (FPGA) implementation results, hardware resource consumption, and power consumptions are provided in Section 3.4 for both 4- and 6-tap filters.

We also compare published 1-D and 2-D DWT architectures with the proposed architectures. Maximum operating frequency, signal-to-noise ratio (SNR) and peak-signal-to-noise ratio (PSNR) figures are sought using the proposed designs operating in fixed-point. Concluding remarks are given in Section 3.5.

3.2 AI-based Daubechies-4 and -6 Scaling Filters

3.2.1 Mathematical Background

An algebraic integer is a real or complex number that is a root of a monic polynomial with integer coefficients [51,54,58]. Algebraic integers can be employed to define encoding mappings which can precisely represent particular irrational numbers by means of usual integers. Considering the roots of the monic polynomials $x^2 - 3$, $x^2 - 10$, and $x^4 - 10x^2 - 15 = 0$ we can extend the set of integers $\mathbb{Z}$ by including the algebraic integer $\zeta = \sqrt{3}$, $\zeta_1 = \sqrt{10}$ and $\zeta_2 = \sqrt{5 + 2\sqrt{10}}$. Doing so, a given

quantity y can possibly be represented as

$$y = a + b \cdot \zeta,$$

$$y = c + d \cdot \zeta_1 + e \cdot \zeta_2 + f \cdot \zeta_1\zeta_2.$$

where a, b, c, d, e, and f are integers. Sets $\{1, \zeta\}$ and $\{1, \zeta_1, \zeta_2, \zeta_1\zeta_2\}$ constitute two bases for AI encoding. Notice that these two bases are adequate for representing the 4- and 6-tap Daubechies filter coefficients. Thus, taking apart quantities $1/\beta_1 = 4\sqrt{2}$ and $1/\beta_2 = 16\sqrt{2}$ as scaling factors, the Daub-4 and -6 filter coefficients can be represented as

$$\mathbf{h}^{(\text{Daub-4})} = \begin{bmatrix} 1+\zeta & 3+\zeta & 3-\zeta & 1-\zeta \end{bmatrix}^\top,$$

$$\mathbf{h}^{(\text{Daub-6})} = \begin{bmatrix} 1+\zeta_1+\zeta_2 \\ 5+\zeta_1+3\zeta_2 \\ 10-2\zeta_1+2\zeta_2 \\ 10-2\zeta_1-2\zeta_2 \\ 5+\zeta_1-3\zeta_2 \\ 1+\zeta_1-\zeta_2 \end{bmatrix}.$$

Therefore, these unnormalized low-pass FIR filters of 4-tap/6-tap can be

split into separate filters given by:

$$\mathbf{h}^{(\text{Daub-4})} = \mathbf{h_1} + \zeta \cdot \mathbf{h}_\zeta, \tag{3.1}$$

$$\mathbf{h}^{(\text{Daub-6})} = \mathbf{h_1}' + \zeta_1 \cdot \mathbf{h}_{\zeta_1} + \zeta_2 \cdot \mathbf{h}_{\zeta_2}, \tag{3.2}$$

where

$$\begin{aligned}
\mathbf{h_1} &= \begin{bmatrix} 1 & 3 & 3 & 1 \end{bmatrix}^\top, \\
\mathbf{h}_\zeta &= \begin{bmatrix} 1 & 1 & -1 & -1 \end{bmatrix}^\top, \\
\mathbf{h_1}' &= \begin{bmatrix} 1 & 5 & 10 & 10 & 5 & 1 \end{bmatrix}^\top, \\
\mathbf{h}_{\zeta_1} &= \begin{bmatrix} 1 & 1 & -2 & -2 & 1 & 1 \end{bmatrix}^\top, \\
\mathbf{h}_{\zeta_2} &= \begin{bmatrix} 1 & 3 & 2 & -2 & -3 & -1 \end{bmatrix}^\top.
\end{aligned}$$

Therefore, the Daub-4 and -6 filter bank analysis can be separated into two/three structures. This facilitates a two/four integer channel structure, where the integer coefficient filters $\mathbf{h_1}$ and $\mathbf{h}_\zeta$; and $\mathbf{h_1}'$, $\mathbf{h}_{\zeta_1}$ and $\mathbf{h}_{\zeta_2}$ are considered. All implied computations are necessarily over an integer field.

Notice that a usual integer m can be effortlessly represented in either basis:

$$m = m + 0 \cdot \zeta,$$

$$m = m + 0 \cdot \zeta_1 + 0 \cdot \zeta_2 + 0 \cdot \zeta_1\zeta_2.$$

This is relevant for encoding image pixel values, which are integers. In practical terms, this means that no circuitry for encoding integer input data is necessary. AI based Daub-4 and -6 filter structures are shown in Fig. 3.2 and Fig. 3.3. These filters possess zero initial condition.

3.2.2 2-D Filtering

We now provide the mathematical framework to describe the operation of the proposed AI-based multi-level encoding design. The following notation is adopted in this work. Let $\mathbf{C}$ be an $N \times N$ matrix with columns $\mathbf{c}_j$, $j = 0, 1, \ldots, N-1$, $\mathbf{C} = \begin{bmatrix} \mathbf{c}_0 & \mathbf{c}_1 & \mathbf{c}_2 & \cdots & \mathbf{c}_{N-1} \end{bmatrix}$ and $\mathbf{v}$ be an N-point column vector. The operation $\oslash$ is defined according to:

$$\begin{aligned} \mathbf{v} \oslash \mathbf{C} &\triangleq (2\downarrow 1) \begin{bmatrix} \mathbf{v} * \mathbf{c}_0 & \mathbf{v} * \mathbf{c}_1 & \mathbf{v} * \mathbf{c}_2 & \cdots & \mathbf{v} * \mathbf{c}_{N-1} \end{bmatrix} \\ &= \begin{bmatrix} \mathbf{v} * \mathbf{c}_0 & \mathbf{v} * \mathbf{c}_2 & \cdots & \mathbf{v} * \mathbf{c}_{N-2} \end{bmatrix}, \end{aligned}$$

where $*$ is the convolution operation. Analogously, operation $\ominus$ is given by:

$$\mathbf{v} \ominus \mathbf{C} \triangleq \left(\mathbf{v} \oslash \mathbf{C}^\top\right)^\top.$$

In other words, $\ominus$ and $\oslash$ are the filtering operations along the rows and columns of a given image, respectively, followed by a dyadic down-sampling stage.

AI-based Daub-4 DWT Decomposition

For the Daub-4 one-level decomposition, we have:

$$\beta_1{}^2 \cdot \mathbf{A}_1 = \mathbf{h}^{(\text{Daub-4})} \ominus \mathbf{h}^{(\text{Daub-4})} \oslash \mathbf{A}_0, \tag{3.3}$$

where $\mathbf{A}_0$ is the input image of integer pixel values. Substituting (3.1) into (3.3), we obtain:

$$\begin{aligned}
\beta_1{}^2 \cdot \mathbf{A}_1 =& (\mathbf{h_1} + \zeta \cdot \mathbf{h}_\zeta) \ominus (\mathbf{h_1} + \zeta \cdot \mathbf{h}_\zeta) \oslash \mathbf{A}_0 \\
=& \mathbf{h_1} \ominus \mathbf{h_1} \oslash \mathbf{A}_0 + \zeta \cdot \mathbf{h}_\zeta \ominus \mathbf{h_1} \oslash \mathbf{A}_0 \\
&+ \zeta \cdot \mathbf{h_1} \ominus \mathbf{h}_\zeta \oslash \mathbf{A}_0 + \zeta^2 \cdot \mathbf{h}_\zeta \ominus \mathbf{h}_\zeta \oslash \mathbf{A}_0.
\end{aligned}$$

Notice that $\zeta^2 = 3$. Thus, we obtain:

$$\beta_1{}^2 \cdot \mathbf{A}_1 = \mathbf{A}_1{}^{(1)} + \zeta \cdot \mathbf{A}_1{}^{(\zeta)}, \tag{3.4}$$

where $\mathbf{A}_1{}^{(1)}$ and $\mathbf{A}_1{}^{(\zeta)}$ are given in Table 4.3.

The operations described above are illustrated in Fig. 3.1(a). The combinational block A is exploited to compute $\mathbf{A}_1{}^{(1)}$ and $\mathbf{A}_1{}^{(\zeta)}$ from the AI filter bank.

Table 3.1: Daub-4 Decompositions

Level	Base	Expression
1	$\mathbf{A}_1{}^{(1)}$	$\mathbf{h_1} \ominus \mathbf{h_1} \oplus \mathbf{A}_0 + 3 \cdot \mathbf{h}_\zeta \ominus \mathbf{h}_\zeta \oplus \mathbf{A}_0$
	$\mathbf{A}_1{}^{(\zeta)}$	$\mathbf{h}_\zeta \ominus \mathbf{h_1} \oplus \mathbf{A}_0 + \mathbf{h_1} \ominus \mathbf{h}_\zeta \oplus \mathbf{A}_0$
2	$\mathbf{A}_2{}^{(1)}$	$\mathbf{h_1} \ominus \mathbf{h_1} \oplus \mathbf{A}_1{}^{(\zeta)} + 3 \cdot \mathbf{h}_\zeta \ominus \mathbf{h}_\zeta \oplus \mathbf{A}_1{}^{(\zeta)} + \mathbf{h}_\zeta \ominus \mathbf{h_1} \oplus \mathbf{A}_1{}^{(1)} + \mathbf{h_1} \ominus \mathbf{h}_\zeta \oplus \mathbf{A}_1{}^{(1)}$
	$\mathbf{A}_2{}^{(\zeta)}$	$\mathbf{h_1} \ominus \mathbf{h_1} \oplus \mathbf{A}_1{}^{(\zeta)} + 3 \cdot \mathbf{h}_\zeta \ominus \mathbf{h}_\zeta \oplus \mathbf{A}_1{}^{(\zeta)} + \mathbf{h}_\zeta \ominus \mathbf{h_1} \oplus \mathbf{A}_1{}^{(1)} + \mathbf{h_1} \ominus \mathbf{h}_\zeta \oplus \mathbf{A}_1{}^{(1)}$

Table 3.2: Daub-4 Decompositions (Continued)

Level	Base	Expression
n	$\mathbf{A}_n{}^{(1)}$	$\mathbf{h_1} \ominus \mathbf{h_1} \oplus \mathbf{A}_{n-1}{}^{(1)} + 3 \cdot \mathbf{h}_\zeta \ominus \mathbf{h}_\zeta \oplus \mathbf{A}_{n-1}{}^{(1)} + 3 \cdot \mathbf{h}_\zeta \ominus \mathbf{h_1} \oplus \mathbf{A}_{n-1}{}^{(\zeta)} + 3 \cdot \mathbf{h_1} \ominus \mathbf{h}_\zeta \oplus \mathbf{A}_{n-1}{}^{(\zeta)}$
	$\mathbf{A}_n{}^{(\zeta)}$	$\mathbf{h_1} \ominus \mathbf{h_1} \oplus \mathbf{A}_{n-1}{}^{(\zeta)} + 3 \cdot \mathbf{h}_\zeta \ominus \mathbf{h}_\zeta \oplus \mathbf{A}_{n-1}{}^{(\zeta)} + \mathbf{h}_\zeta \ominus \mathbf{h_1} \oplus \mathbf{A}_{n-1}{}^{(1)} + \mathbf{h_1} \ominus \mathbf{h}_\zeta \oplus \mathbf{A}_{n-1}{}^{(1)}$

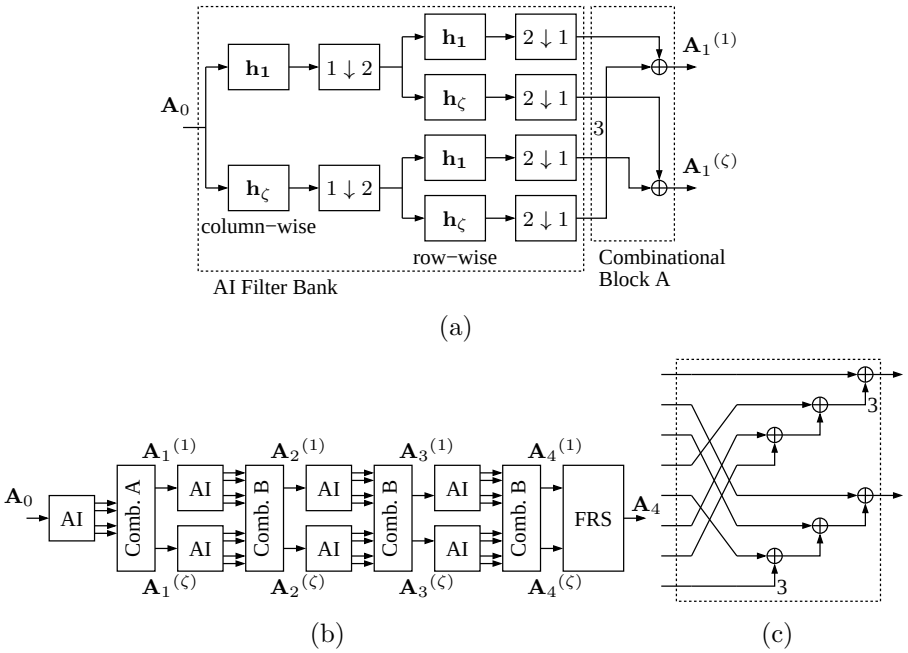


Figure 3.1: (a) Single AI filter bank decomposition, (b) multi-level AI filter bank with final reconstruction step, and (c) combinational block B.

The resulting filtered images decomposition $\mathbf{A}_1{}^{(1)}$ and $\mathbf{A}_1{}^{(\zeta)}$ necessitate only integer arithmetic to be rendered. Multi-level analysis follows the same algorithm, i.e., multiplications by the AI base ζ are never explicitly performed.

Further decompositions are similarly computed. In particular, the 2nd level decomposition is formulated below:

$$\beta_1{}^2 \cdot \mathbf{A}_2 = \mathbf{h}^{(\text{Daub-4})} \ominus \mathbf{h}^{(\text{Daub-4})} \oslash \mathbf{A}_1.$$

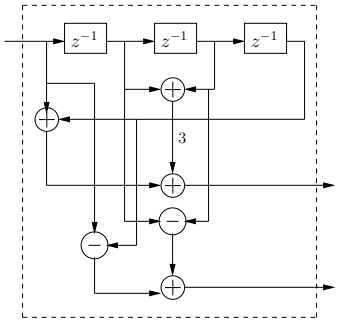


Figure 3.2: Daub-4 AI Filter Structure

Applying (3.4) into above expression, we proceed as follows:

$$\begin{aligned}\beta_1{}^4 \cdot \mathbf{A}_2 =& \mathbf{h}^{(\text{Daub-4})} \ominus \mathbf{h}^{(\text{Daub-4})} \oplus \left(\mathbf{A}_1{}^{(1)} + \zeta \cdot \mathbf{A}_1{}^{(\zeta)}\right) \\ =& \mathbf{A}_2{}^{(1)} + \zeta \cdot \mathbf{A}_2{}^{(\zeta)}\end{aligned}$$

where approximations $\mathbf{A}_2{}^{(1)}$ and $\mathbf{A}_2{}^{(\zeta)}$ have their fully expanded forms given in Table 4.3.

Indeed, the above manipulation can be similarly applied to the remaining approximation levels. Thus the nth level approximation is furnished by:

$$\beta_1{}^{2n} \cdot \mathbf{A}_n = \mathbf{A}_n{}^{(1)} + \zeta \cdot \mathbf{A}_n{}^{(\zeta)}, \quad n \geq 2,$$

where $\mathbf{A}_n{}^{(1)}$ and $\mathbf{A}_n{}^{(\zeta)}$ are shown in Table 4.3.

Notice that the required multiplications by 3 shown in Table 4.3 can be easily realized by a bit-shift operation and addition, i.e. $3 \cdot m = (m \ll 1) + m$, where m is an integer.

The above multi-level analyses for Daub-4 DWT filters are depicted in Fig. 3.1(b). Expressions for level 2 and n shown in Table 4.3 induce the implementation of the combinational block B, as shown in Fig. 3.1(b). The architecture of this block is detailed in Fig. 3.1(c). Fig. 3.1(b) also shows the FRS block, which is detailed in the next section.

AI-based Daub-6 DWT Decomposition

In a similar fashion, the Daub-6 filter bank can be put into the AI formalism. Considering Fig. 2.1, we can derive the following expression:

$$\beta_2{}^2 \cdot \mathbf{A}_1 = \mathbf{h}^{(\text{Daub-6})} \ominus \mathbf{h}^{(\text{Daub-6})} \oplus \mathbf{A}_0,$$

where $\mathbf{A}_0$ is the input image of integer pixel values.

Invoking (3.2), we obtain:

$$\begin{aligned}
{\beta_2}^2 \cdot \mathbf{A}_1 =& (\mathbf{h_1}' + \zeta_1 \cdot \mathbf{h}_{\zeta_1} + \zeta_2 \cdot \mathbf{h}_{\zeta_2}) \\
& \ominus (\mathbf{h_1}' + \zeta_1 \cdot \mathbf{h}_{\zeta_1} + \zeta_2 \cdot \mathbf{h}_{\zeta_2}) \oplus \mathbf{A}_0 \\
=& \mathbf{h_1}' \ominus \mathbf{h_1}' \oplus \mathbf{A}_0 + \zeta_1 \cdot \mathbf{h}_{\zeta_1} \ominus \mathbf{h_1}' \oplus \mathbf{A}_0 \\
& + \zeta_2 \cdot \mathbf{h}_{\zeta_2} \ominus \mathbf{h_1}' \oplus \mathbf{A}_0 + \mathbf{h_1}' \ominus \zeta_1 \cdot \mathbf{h}_{\zeta_1} \oplus \mathbf{A}_0 \\
& + \zeta_1 \cdot \mathbf{h}_{\zeta_1} \ominus \zeta_1 \cdot \mathbf{h}_{\zeta_1} \oplus \mathbf{A}_0 \\
& + \zeta_2 \cdot \mathbf{h}_{\zeta_2} \ominus \zeta_1 \cdot \mathbf{h}_{\zeta_1} \oplus \mathbf{A}_0 \\
& + \mathbf{h_1}' \ominus \zeta_2 \cdot \mathbf{h}_{\zeta_2} \oplus \mathbf{A}_0 + \zeta_1 \cdot \mathbf{h}_{\zeta_1} \ominus \zeta_2 \cdot \mathbf{h}_{\zeta_2} \oplus \mathbf{A}_0 \\
& + \zeta_2 \cdot \mathbf{h}_{\zeta_2} \ominus \zeta_2 \cdot \mathbf{h}_{\zeta_2} \oplus \mathbf{A}_0.
\end{aligned}$$

By grouping the relevant terms, we obtain:

$${\beta_2}^2 \cdot \mathbf{A}_1 = {\mathbf{A}_1}^{(1)} + \zeta_1 \cdot {\mathbf{A}_1}^{(\zeta_1)} + \zeta_2 \cdot {\mathbf{A}_1}^{(\zeta_2)} + \zeta_1\zeta_2 \cdot {\mathbf{A}_1}^{(\zeta_1\zeta_2)}, \tag{3.5}$$

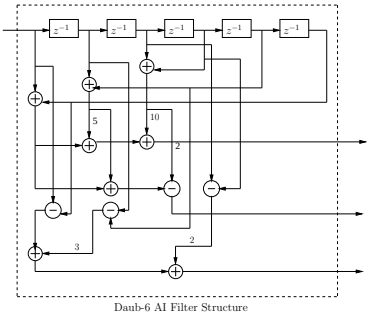


Figure 3.3: Daub-6 AI Filter Structure

Table 3.3: Daub-6 Decompositions

Level	Base	Expression
1	${\mathbf{A}_1}^{(1)}$	$\{\mathbf{h_1}' \ominus \mathbf{h_1}' + 10\mathbf{h}_{\zeta_1} \ominus \mathbf{h}_{\zeta_1} + 5\mathbf{h}_{\zeta_2} \ominus \mathbf{h}_{\zeta_2}\} \oplus \mathbf{A}_0.$
	${\mathbf{A}_1}^{(\zeta_1)}$	$\{\mathbf{h}_{\zeta_1} \ominus \mathbf{h_1}' + \mathbf{h_1}' \ominus \mathbf{h}_{\zeta_1} + 2\mathbf{h}_{\zeta_2} \ominus \mathbf{h}_{\zeta_2}\} \oplus \mathbf{A}_0$
	${\mathbf{A}_1}^{(\zeta_2)}$	$\{\mathbf{h}_{\zeta_2} \ominus \mathbf{h_1}' + \mathbf{h_1}' \ominus \mathbf{h}_{\zeta_2}\} \oplus \mathbf{A}_0$
	${\mathbf{A}_1}^{(\zeta_1\zeta_2)}$	$\{\mathbf{h}_{\zeta_1} \ominus \mathbf{h}_{\zeta_2} + \mathbf{h}_{\zeta_2} \ominus \mathbf{h}_{\zeta_1}\} \oplus \mathbf{A}_0$

where ${\mathbf{A}_1}^{(1)}$, ${\mathbf{A}_1}^{(\zeta_1)}$, ${\mathbf{A}_1}^{(\zeta_2)}$, and ${\mathbf{A}_1}^{(\zeta_1\zeta_2)}$ are given in Table 3.2.

The error free integer operations described above are illustrated in Fig. 3.4(a). The combinational block C is employed in order to furnish ${\mathbf{A}_1}^{(\zeta_1)}$, ${\mathbf{A}_1}^{(\zeta_2)}$, and ${\mathbf{A}_1}^{(\zeta_1\zeta_2)}$ from the AI filter bank. The level 2 decomposition follows similar manipulations, as detailed below:

$$ {\beta_2}^2 \cdot \mathbf{A}_2 = \mathbf{h}^{(\text{Daub-6})} \ominus \mathbf{h}^{(\text{Daub-6})} \oplus \mathbf{A}_1. $$

Calling (3.5), we derive the following expression:

$$ {\beta_2}^4 \cdot \mathbf{A}_2 = {\mathbf{A}_2}^{(1)} + \zeta_1 \cdot {\mathbf{A}_2}^{(\zeta_1)} + \zeta_2 \cdot {\mathbf{A}_2}^{(\zeta_2)} + \zeta_1\zeta_2 \cdot {\mathbf{A}_2}^{(\zeta_1\zeta_2)}, $$

where the approximations ${\mathbf{A}_2}^{(1)}$, ${\mathbf{A}_2}^{(\zeta_1)}$, ${\mathbf{A}_2}^{(\zeta_2)}$, and ${\mathbf{A}_2}^{(\zeta_1\zeta_2)}$ are given in Table 3.2.

The general result for the n level decomposition is shown in Table 3.2.

Table 3.4: Daub-6 decompositions (Continued)

Level	Base	Expression
2	$\mathbf{A_2}^{(1)}$	$\{\mathbf{h_1}' \ominus \mathbf{h_1}' + 10 \cdot \mathbf{h}_{\zeta_1} \ominus \mathbf{h}_{\zeta_1} + 5 \cdot \mathbf{h}_{\zeta_2} \ominus \mathbf{h}_{\zeta_2}\} \oplus \mathbf{A_1}^{(1)} + 10 \cdot \{\mathbf{h}_{\zeta_1} \ominus \mathbf{h_1}' + \mathbf{h_1}' \ominus \mathbf{h}_{\zeta_1} + 2 \cdot \mathbf{h}_{\zeta_2} \ominus \mathbf{h}_{\zeta_2}\} \oplus \mathbf{A_1}^{(\zeta_1)} + 5 \cdot \mathbf{A_1}^{(\zeta_2)} \oplus \{\mathbf{h}_{\zeta_2} \ominus \mathbf{h_1}' + \mathbf{h_1}' \ominus \mathbf{h}_{\zeta_2} + 4 \cdot \mathbf{h}_{\zeta_1} \ominus \mathbf{h}_{\zeta_2} + 4 \cdot \mathbf{h}_{\zeta_2} \ominus \mathbf{h}_{\zeta_1}\} + \{4 \cdot \mathbf{h}_{\zeta_2} \ominus \mathbf{h_1}' + 10 \cdot \mathbf{h}_{\zeta_2} \ominus \mathbf{h}_{\zeta_1}\} 5 \cdot \oplus \mathbf{A_1}^{(\zeta_1\zeta_2)}$
	$\mathbf{A_2}^{(\zeta_1)}$	$\{\mathbf{h}_{\zeta_1} \ominus \mathbf{h_1}' + \mathbf{h_1}' \ominus \mathbf{h}_{\zeta_1} + 2 \cdot \mathbf{h}_{\zeta_2} \ominus \mathbf{h}_{\zeta_2}\} \oplus \mathbf{A_1}^{(1)} + \{\mathbf{h_1}' \ominus \mathbf{h_1}' + 10 \cdot \mathbf{h}_{\zeta_1} \ominus \mathbf{h}_{\zeta_1} + 5 \cdot \mathbf{h}_{\zeta_2} \ominus \mathbf{h}_{\zeta_2}\} \oplus \mathbf{A_1}^{(\zeta_1)} + \{2 \cdot \mathbf{h}_{\zeta_2} \ominus \mathbf{h_1}' + 2 \cdot \mathbf{h_1}' \ominus \mathbf{h}_{\zeta_2} + 5 \cdot \mathbf{h}_{\zeta_1} \ominus \mathbf{h}_{\zeta_2} + 5 \cdot \mathbf{h}_{\zeta_2} \ominus \mathbf{h}_{\zeta_1}\} \oplus \mathbf{A_1}^{(\zeta_2)} + \{\mathbf{h}_{\zeta_1} \ominus \mathbf{h}_{\zeta_2} + \mathbf{h}_{\zeta_2} \ominus \mathbf{h}_{\zeta_1}\} \oplus \mathbf{A_1}^{(\zeta_1\zeta_2)}$
	$\mathbf{A_2}^{(\zeta_2)}$	$\{\mathbf{h_1}' \ominus \mathbf{h}_{\zeta_2} + \mathbf{h}_{\zeta_2} \ominus \mathbf{h_1}'\} \oplus \mathbf{A_1}^{(1)} + \{\mathbf{h}_{\zeta_1} \ominus \mathbf{h}_{\zeta_2} + \mathbf{h}_{\zeta_2} \ominus \mathbf{h}_{\zeta_1}\} \oplus \mathbf{A_1}^{(\zeta_1)} + \mathbf{A_1}^{(\zeta_2)} \oplus \{\mathbf{h_1}' \ominus \mathbf{h_1}' + 10 \cdot \mathbf{h}_{\zeta_1} \ominus \mathbf{h}_{\zeta_1} + 5 \cdot \mathbf{h}_{\zeta_2} \ominus \mathbf{h}_{\zeta_2}\} + \{\mathbf{h}_{\zeta_1} \ominus \mathbf{h_1}' + \mathbf{h_1}' \ominus \mathbf{h}_{\zeta_1} + 2 \cdot \mathbf{h}_{\zeta_2} \ominus \mathbf{h}_{\zeta_2}\} \oplus 10 \cdot \mathbf{A_1}^{(\zeta_1\zeta_2)}$
	$\mathbf{A_2}^{(\zeta_1\zeta_2)}$	$\{\mathbf{h}_{\zeta_2} \ominus \mathbf{h}_{\zeta_1} + \mathbf{h}_{\zeta_1} \ominus \mathbf{h}_{\zeta_2}\} \oplus \mathbf{A_1}^{(1)} + \{\mathbf{h}_{\zeta_2} \ominus \mathbf{h_1}' + \mathbf{h_1}' \ominus \mathbf{h}_{\zeta_2}\} \oplus \mathbf{A_1}^{(\zeta_1)} + \mathbf{A_1}^{(\zeta_2)} \oplus \{\mathbf{h}_{\zeta_1} \ominus \mathbf{h_1}' + \mathbf{h_1}' \ominus \mathbf{h}_{\zeta_1} + 2 \cdot \mathbf{h}_{\zeta_2} \ominus \mathbf{h}_{\zeta_2}\} + \{\mathbf{h_1}' \ominus \mathbf{h_1}' + 10 \cdot \mathbf{h}_{\zeta_1} \ominus \mathbf{h}_{\zeta_1} + 5 \cdot \mathbf{h}_{\zeta_2} \ominus \mathbf{h}_{\zeta_2}\} \oplus \mathbf{A_1}^{(\zeta_1\zeta_2)}$

Table 3.5: Daub-6 decompositions (Continued)

Level	Base	Expression
n	$\mathbf{A}_n{}^{(1)}$	$\mathbf{h_1}' \ominus \mathbf{h_1}' + 10 \cdot \mathbf{h}_{\zeta_1} \ominus \mathbf{h}_{\zeta_1} + 5 \cdot \mathbf{h}_{\zeta_2} \ominus \mathbf{h}_{\zeta_2}\} \oplus \mathbf{A}_{n-1}{}^{(1)} + \{40 \cdot \mathbf{h}_{\zeta_2} \ominus \mathbf{h_1}' + 100 \cdot \mathbf{h}_{\zeta_1} \ominus \mathbf{h}_{\zeta_2}\} \oplus \mathbf{A}_{n-1}{}^{(\zeta_1\zeta_2)}$ $+5 \cdot \mathbf{A}_{n-1}{}^{(\zeta_2)} \oplus \{\mathbf{h}_{\zeta_2} \ominus \mathbf{h_1}' + \mathbf{h_1}' \ominus \mathbf{h}_{\zeta_2} + 4 \cdot \mathbf{h}_{\zeta_1} \ominus \mathbf{h}_{\zeta_2} + 4 \cdot \mathbf{h}_{\zeta_2} \ominus \mathbf{h}_{\zeta_1}\}$ $+\{\mathbf{h}_{\zeta_1} \ominus \mathbf{h_1}' + \mathbf{h_1}' \ominus \mathbf{h}_{\zeta_1} + 2 \cdot \mathbf{h}_{\zeta_2} \ominus \mathbf{h}_{\zeta_2}\}10 \cdot \oplus \mathbf{A}_{n-1}{}^{(\zeta_1)}$
	$\mathbf{A}_n{}^{(\zeta_1)}$	$\{\mathbf{h}_{\zeta_1} \ominus \mathbf{h_1}' + \mathbf{h_1}' \ominus \mathbf{h}_{\zeta_1} + 2 \cdot \mathbf{h}_{\zeta_2} \ominus \mathbf{h}_{\zeta_2}\} \oplus \mathbf{A}_{n-1}{}^{(1)} + \{\mathbf{h_1}' \ominus \mathbf{h_1}' + 10 \cdot \mathbf{h}_{\zeta_1} \ominus \mathbf{h}_{\zeta_1} + 5 \cdot \mathbf{h}_{\zeta_2} \ominus \mathbf{h}_{\zeta_2}\} \oplus \mathbf{A}_{n-1}{}^{(\zeta_1)} +$ $\{2 \cdot \mathbf{h}_{\zeta_2} \ominus \mathbf{h_1}' + 2 \cdot \mathbf{h_1}' \ominus \mathbf{h}_{\zeta_2} + 5 \cdot \mathbf{h}_{\zeta_1} \ominus \mathbf{h}_{\zeta_2} + 5 \cdot \mathbf{h}_{\zeta_2} \ominus \mathbf{h}_{\zeta_1}\} \oplus \mathbf{A}_1{}^{(\zeta_2)} + \{\mathbf{h}_{\zeta_1} \ominus \mathbf{h}_{\zeta_2} + \mathbf{h}_{\zeta_2} \ominus \mathbf{h}_{\zeta_1}\} \oplus \mathbf{A}_{n-1}{}^{(\zeta_1\zeta_2)}$
	$\mathbf{A}_n{}^{(\zeta_2)}$	$\{\mathbf{h}_{\zeta_1} \ominus \mathbf{h}_{\zeta_2} + \mathbf{h}_{\zeta_2} \ominus \mathbf{h}_{\zeta_1}\} \oplus \mathbf{A}_{n-1}{}^{(1)} + \{\mathbf{h}_{\zeta_1} \ominus \mathbf{h}_{\zeta_2} + \mathbf{h}_{\zeta_2} \ominus \mathbf{h}_{\zeta_1}\} \oplus \mathbf{A}_{n-1}{}^{(\zeta_1)} + \mathbf{A}_{n-1}{}^{(\zeta_2)} \oplus$ $\{\mathbf{h_1}' \ominus \mathbf{h_1}' + 10 \cdot \mathbf{h}_{\zeta_1} \ominus \mathbf{h}_{\zeta_1} + 5 \cdot \mathbf{h}_{\zeta_2} \ominus \mathbf{h}_{\zeta_2}\} + \{\mathbf{h}_{\zeta_1} \ominus \mathbf{h_1}' + \mathbf{h_1}' \ominus \mathbf{h}_{\zeta_1} + 2 \cdot \mathbf{h}_{\zeta_2} \ominus \mathbf{h}_{\zeta_2}\} \oplus \mathbf{A}_{n-1}{}^{(\zeta_1\zeta_2)}$
	$\mathbf{A}_n{}^{(\zeta_1\zeta_2)}$	$\{\mathbf{h}_{\zeta_2} \ominus \mathbf{h_1}' + \mathbf{h_1}' \ominus \mathbf{h}_{\zeta_2}\} \oplus \mathbf{A}_{n-1}{}^{(1)} + \{\mathbf{h}_{\zeta_1} \ominus \mathbf{h}_{\zeta_2} + \mathbf{h}_{\zeta_2} \ominus \mathbf{h}_{\zeta_1}\} \oplus \mathbf{A}_{n-1}{}^{(\zeta_1)} + \mathbf{A}_{n-1}{}^{(\zeta_2)} \oplus$ $\{\mathbf{h_1}' \ominus \mathbf{h_1}' + 10 \cdot \mathbf{h}_{\zeta_1} \ominus \mathbf{h}_{\zeta_1} + 5 \cdot \mathbf{h}_{\zeta_2} \ominus \mathbf{h}_{\zeta_2}\} + \{\mathbf{h}_{\zeta_1} \ominus \mathbf{h_1}' + \mathbf{h_1}' \ominus \mathbf{h}_{\zeta_1} + 2 \cdot \mathbf{h}_{\zeta_2} \ominus \mathbf{h}_{\zeta_2}\}10 \cdot \oplus \mathbf{A}_{n-1}{}^{(\zeta_1\zeta_2)}$

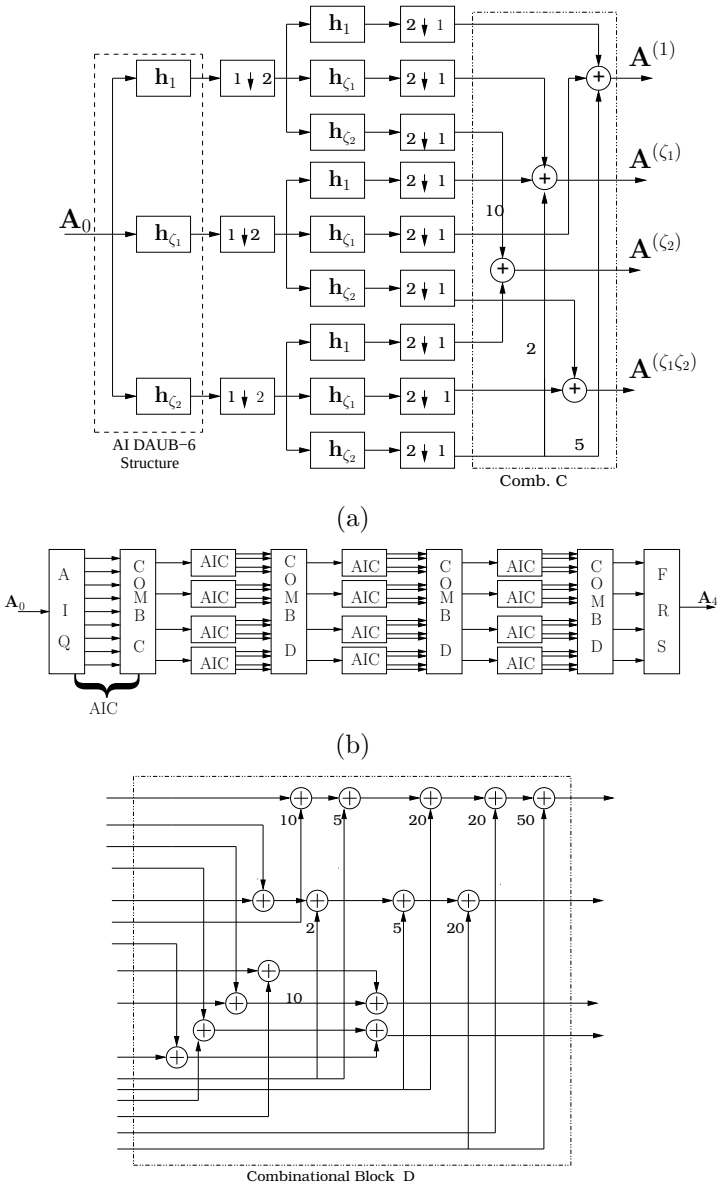


Figure 3.4: (a) Single AI filter bank decomposition, (b) multi-level AI filter bank with final reconstruction step, and (c) combinational block D.

Fig. 3.4(b) depicts the full scheme of a four-level Daub-6 decomposition. The structure of combinatorial block D stems from the expressions shown in Table 3.2 for level 2 and n decompositions. Fig. 3.4(c) details this stage.

3.3 Final Reconstruction Step

The proposed AI-based wavelet analyses based on Daub-4 and -6 filter banks are computed entirely over extended integer fields. However, the resulting AI encoded approximations ${\mathbf{A}_n}^{(1)}$, ${\mathbf{A}_n}^{(\zeta)}$, ${\mathbf{A}_n}^{(\zeta_1)}$, ${\mathbf{A}_n}^{(\zeta_2)}$, and ${\mathbf{A}_n}^{(\zeta_1\zeta_2)}$ must be converted back to standard fixed-point representation. This is required in order to interface the resulting approximation sub-images with conventional real time systems. Decoding operations for both Daub-4 and -6 consist of explicitly performing the following computations, respectively:

$$\mathbf{A}_n = \frac{{\mathbf{A}_n}^{(1)} + \zeta \cdot {\mathbf{A}_n}^{(\zeta)}}{\beta_1^{2n}}, \tag{3.6}$$

$$\mathbf{A}_n = \frac{1}{\beta_2^{2n}} \left({\mathbf{A}_n}^{(1)} + \zeta_1 \cdot {\mathbf{A}_n}^{(\zeta_1)} + \zeta_2 \cdot {\mathbf{A}_\mathbf{n}}^{(\zeta_2)} + \zeta_1 \cdot \zeta_2 \cdot {\mathbf{A}_\mathbf{n}}^{(\zeta_1 \cdot \zeta_2)} \right). \tag{3.7}$$

Fortunately, the factors $1/\beta_1^{2n}$ and $1/\beta_2^{2n}$ are always a power of two, which can be conveniently realized with bit-shift operation. The above decoding operations are realized at the FRS blocks depicted in Fig. 3.1(b) and 3.4(b), respectively.

Therefore, the only possible source of errors in the proposed architectures

for Daub-4 and -6 are the multiplication by AI basis elements:

$$\zeta = \sqrt{3} \approx 1.73205080756888\ldots,$$

$$\zeta_1 = \sqrt{10} \approx 3.16227766016838\ldots,$$

$$\zeta_2 = \sqrt{5 + 2\sqrt{10}} \approx 3.36519766437824\ldots,$$

$$\zeta_1\zeta_2 = \sqrt{50 + 20\sqrt{10}} \approx 10.6416893961141\ldots.$$

We propose two approaches for the FRS design: (i) CSD representation and (ii) expansion factor method.

3.3.1 CSD Approximation

The FRS can be directly implemented by approximating the required irrationals in (3.6) and (3.7) into rationals. A possibility is employing CSD representation.

Table 3.3 displays CSD encodings for ζ and Table 3.4 shows encoding for ζ_1, ζ_2, and $\zeta_1\zeta_2$ for several word lengths as well as the associate relative errors. CSD encoding requires only bit-shifters and adders/subtracters.

3.3.2 Expansion Factor Method

Expansion factors are scaling constants usually employed in the design of approximate discrete transforms [33, 53]. In [57, p. 274], Britanak *et al.* survey the topic in this context. Recently this methodology was extended and adapted to the design of final reconstruction blocks related to AI based architectures [56].

Table 3.6: CSD Encoding for ζ

Word length	CSD Encoding	% Error
8 bit	$2 - 2^{-2} - 2^{-6}$	1.33
10 bit	$2 - 2^{-2} - 2^{-6} - 2^{-9}$	0.021
12 bit	$2 - 2^{-2} - 2^{-6} - 2^{-9}$	0.021
14 bit	$2 - 2^{-2} - 2^{-6} - 2^{-9} - 2^{-12}$	0.0086
16 bit	$2 - 2^{-2} - 2^{-6} - 2^{-9} - 2^{-12} - 2^{-13}$	0.0028

Table 3.7: CSD Encoding for ζ_1, ζ_2, and $\zeta_1\zeta_2$

AI	Word length (bits)	Encoding	Absolute Relative Error
ζ_1	8	$2^2 - 1 + 2^{-3} + 2^{-6}$	0.0068
	10	$2^2 - 1 + 2^{-3} + 2^{-6} + 2^{-8}$	0.0056
	12	$2^2 - 1 + 2^{-3} + 2^{-5} + 2^{-7} + 2^{-10}$	0.00087
	14	$2^2 - 1 + 2^{-3} + 2^{-5} + 2^{-7} + 2^{-12}$	0.00064
	16	$2^2 - 1 + 2^{-3} + 2^{-5} + 2^{-7} + 2^{-14}$	0.00058
ζ_2	8	$2^2 - 2^{-1} - 2^{-3} - 2^{-6}$	0.0017
	10	$2^2 - 2^{-1} - 2^{-3} - 2^{-7}$	0.00059
	12	$2^2 - 2^{-1} - 2^{-3} - 2^{-7} - 2^{-10}$	0.00030
	14	$2^2 - 2^{-1} - 2^{-3} - 2^{-7} - 2^{-10} - 2^{-12}$	0.00022
	16	$2^2 - 2^{-1} - 2^{-3} - 2^{-7} - 2^{-10} - 2^{-12}$	0.00011
$\zeta_1\zeta_2$	8	$2^3 + 2 + 2^{-1} + 2^{-3} + 2^{-4}$	0.0043
	10	$2^3 + 2 + 2^{-1} + 2^{-3} + 2^{-6}$	0.0001
	12	$2^3 + 2 + 2^{-1} + 2^{-3} + 2^{-6}$	0.0001
	14	$2^3 + 2 + 2^{-1} + 2^{-3} + 2^{-6} + 2^{-10}$	0.000008
	16	$2^3 + 2 + 2^{-1} + 2^{-3} + 2^{-6} + 2^{-10}$	0.000008

An expansion factor is simply a constant that simultaneously scales a given set of real numbers into integer values. In practical terms, only approximate integers at a given error tolerance are sought.

In mathematical terms, we have the following structure. Let the AI elements ζ_1, ζ_2, and $\zeta_1\zeta_2$ constitute a vector $\boldsymbol{\zeta} = \begin{bmatrix} \zeta_1 & \zeta_2 & \zeta_1\zeta_2 \end{bmatrix}^\top$. An expansion factor is a real number $\alpha^* > 1$ that satisfies the following minimization problem [57, p. 274]:

$$\alpha^* = \arg\min_{\alpha>1} \|\alpha \cdot \boldsymbol{\zeta} - \text{round}(\alpha \cdot \boldsymbol{\zeta})\|, \tag{3.8}$$

where $\|\cdot\|$ returns the Euclidean norm and round$(\cdot)$ is the rounding-off function. Resulting integer approximations are given by $m_1 = \text{round}(\alpha^* \cdot \zeta_1)$, $m_2 = \text{round}(\alpha^* \cdot \zeta_2)$, and $m_3 = \text{round}(\alpha^* \cdot \zeta_1\zeta_2)$.

Now, we can recast (3.7) according to:

$$\mathbf{A}_n \approx \mathbf{A}_n{}^{(1)} + \frac{1}{\alpha^*}\left(m_1 \cdot \mathbf{A}_n^{(\zeta_1)} + m_2 \cdot \mathbf{A}_n^{(\zeta_2)} + m_3 \cdot \mathbf{A}_n^{(\zeta_1 \cdot \zeta_2)}\right).$$

Notice that the above expression in parentheses can be evaluated by means of integer arithmetic, which requires simple additions and bit-shift operations in hardware. As a consequence, only a single non-integer multiplication by $1/\alpha^*$ is required.

As posed above, (3.8) is a non-linear, unconstrained optimization problem. Its intractability indicates the application of computational search. In this case, we must impose a constraint to the search space.

Table 3.8: FRS Based on the Expansion Factor Method

Expansion Factor		49.3336	78.7465	192.2623	271.0093	463.2723	734.2817
AI Encoded Base	m_1	156	249	608	857	1465	2322
	m_2	166	265	647	912	1559	2471
	m_3	525	838	2046	2884	4930	7814
Approximate Error		0.00020	0.00002	0.01182	0.00730	0.00520	0.00457

Thus, for $1 < \alpha \leq 10^3$ with a precision of 10^{-4}, we could obtain five distinct solutions for (3.8). These values are listed in Table 3.5. The scaling factor choice depends on the specific application in question, resource constraints, and the accepted error tolerance.

For example, taking $\alpha^* = 49.3336$, we obtain:

$$\alpha^* \cdot \begin{bmatrix} \zeta_1 \\ \zeta_2 \\ \zeta_1\zeta_2 \end{bmatrix} = \begin{bmatrix} 156.00654\ldots \\ 166.01731\ldots \\ 524.99284\ldots \end{bmatrix} \approx \begin{bmatrix} 156 \\ 166 \\ 525 \end{bmatrix} = \begin{bmatrix} m_1 \\ m_2 \\ m_3 \end{bmatrix}.$$

Above particular scaling leads to percent relative errors of 0.0042, 0.0104, and 0.0014 in ζ_1, ζ_2, $\zeta_1\zeta_2$, respectively. We used the CSD representation for Daub-4 filters and both CSD representation and the expansion factor method for Daub-6.

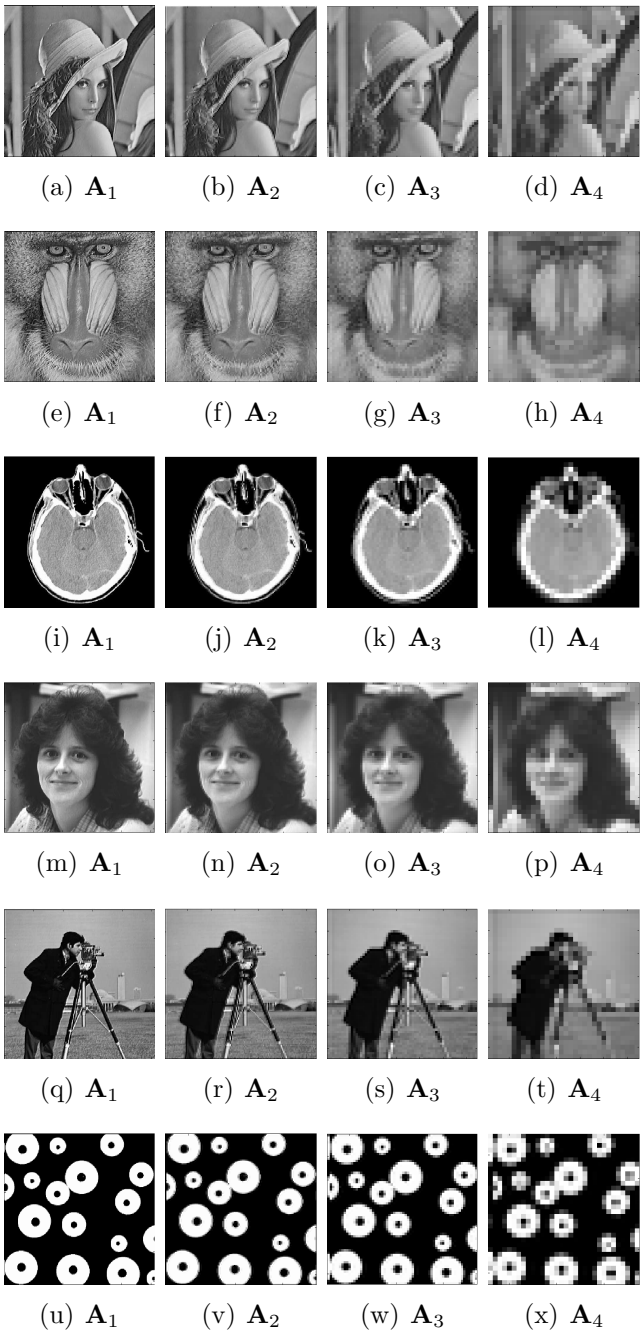
The expansion factor method is expected to offer better results for larger basis. Indeed, the Daub-4 architecture could not benefit from the expansion factor method since its basis contains only one non-unity element: ζ. However, because the AI basis related to the Daub-6 scheme has three non-unity elements (ζ_1, ζ_2,

$\zeta_1\zeta_2$), the expansion factor method could lead to useful architectures in the FRS following computational search algorithms for suitable integer combinations. In the next section we provide measurement results concerning the expansion factor method.

3.4 FPGA Implementation and Results

The architectures for Daub-4 and -6 filter banks were implemented on Xilinx Virtex xc6vcx240t-1ff1156 device using the ML605 evaluation board. The designs were tested with six different standard images obtained from [59]. Gray 512×512 images images *Woman*, *Cameraman*, and *Reflection* to the Daub-4 filter banks whereas *Mandrill*, *Lena*, and *CT head* were submitted to the Daub-6 filter banks. Hardware results were verified with MATLAB. Fig. 5.6 displays hardware results from the Xilinx FPGA for the Daub-4 and -6 filter banks. Table 3.6 shows a performance comparison among proposed Daub-4 and -6 architectures for single level decomposition of 8-bit *Lena* image.

For comparison, we devised a version of the proposed system that operates over fixed-point arithmetic instead of AI-based arithmetic. For such, we employed 8 bits for word size with 6 fractional bits. In this case, the required filter banks were implemented by quantizing the exact filter coefficients into the fixed-point representation. Notice that the fixed-point scheme incurs coupled quantization noise, whereas the AI-based architecture is immune to this source of contamination. Fig. 3.6 shows the results for the fixed-point design.



(a) $\mathbf{A}_1$ (b) $\mathbf{A}_2$ (c) $\mathbf{A}_3$ (d) $\mathbf{A}_4$

(e) $\mathbf{A}_1$ (f) $\mathbf{A}_2$ (g) $\mathbf{A}_3$ (h) $\mathbf{A}_4$

(i) $\mathbf{A}_1$ (j) $\mathbf{A}_2$ (k) $\mathbf{A}_3$ (l) $\mathbf{A}_4$

(m) $\mathbf{A}_1$ (n) $\mathbf{A}_2$ (o) $\mathbf{A}_3$ (p) $\mathbf{A}_4$

(q) $\mathbf{A}_1$ (r) $\mathbf{A}_2$ (s) $\mathbf{A}_3$ (t) $\mathbf{A}_4$

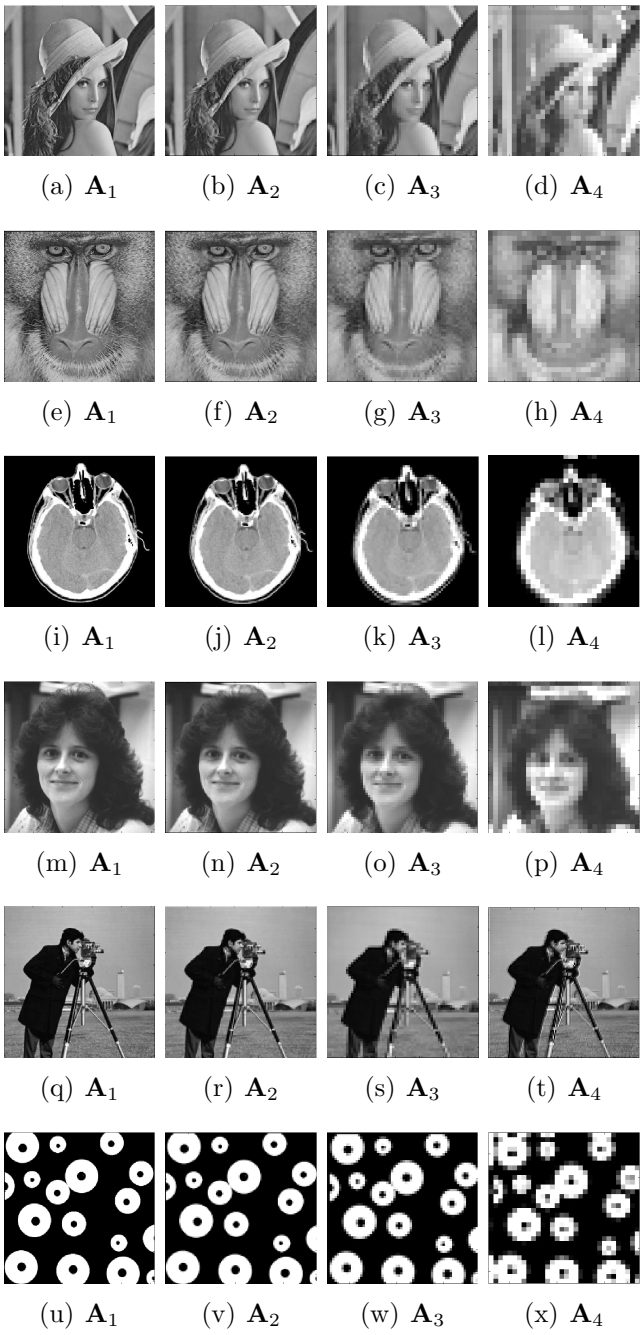
(u) $\mathbf{A}_1$ (v) $\mathbf{A}_2$ (w) $\mathbf{A}_3$ (x) $\mathbf{A}_4$

Figure 3.5: Approximation sub-images $\mathbf{A}_1$, $\mathbf{A}_2$, $\mathbf{A}_3$, and $\mathbf{A}_4$ obtained from on-chip physical verification on a Virtex-6 vcx240t-1ff1156.

Table 3.9: Comparison of Daub-4 and -6 Performances †

Aspect	Daub-4	Daub-6
FPGA	Xilinx Virtex-6	Xilinx Virtex-6
Target Device	vcx240t-1ff1156	vcx240t-1ff1156
Max. Freq. (MHz)	442.47	274.72
CPD (ns)	2.26	3.64
Dynamic Power (mW)	38	57
PSNR (dB)	66.82	68.12
# Adders	32	61
Registers	258	765
AT ($\times 10^{-6}$)	1.62	6.99
AT^2 ($\times 10^{-15}$)	3.66	49.20
Throughput	1 ip/ op	1 ip/ op

† Measured for single level decomposition with 8-bit input data



(a) $\mathbf{A}_1$ (b) $\mathbf{A}_2$ (c) $\mathbf{A}_3$ (d) $\mathbf{A}_4$

(e) $\mathbf{A}_1$ (f) $\mathbf{A}_2$ (g) $\mathbf{A}_3$ (h) $\mathbf{A}_4$

(i) $\mathbf{A}_1$ (j) $\mathbf{A}_2$ (k) $\mathbf{A}_3$ (l) $\mathbf{A}_4$

(m) $\mathbf{A}_1$ (n) $\mathbf{A}_2$ (o) $\mathbf{A}_3$ (p) $\mathbf{A}_4$

(q) $\mathbf{A}_1$ (r) $\mathbf{A}_2$ (s) $\mathbf{A}_3$ (t) $\mathbf{A}_4$

(u) $\mathbf{A}_1$ (v) $\mathbf{A}_2$ (w) $\mathbf{A}_3$ (x) $\mathbf{A}_4$

Figure 3.6: Approximation sub-images $\mathbf{A}_1$, $\mathbf{A}_2$, $\mathbf{A}_3$, and $\mathbf{A}_4$ obtained from FP scheme (8-bit word length and 6 fractional bits) Daubechies 4- and 6-tap wavelet filters.

3.4.1 Resource Consumption and Figures of Merit

Tables 3.7 and 3.8 list resource consumption for the Daub-4 and -6 filter banks. Monitored resources include: the number of slice registers, the look-up table (LUT) count, and the number of configurable logic blocks (CLB).

Critical path delays (CPD), the maximum operating frequency, area-time product (AT), and AT^2 were selected as figures of merit, being also reported in Tables 3.7 and 3.8.

The AT product is a standard performance metric in digital hardware designs. It refers to chip-area and speed (maximum frequency) of the design. Lower AT values indicate a higher speed of operation. In an FPGA, the area (A) is provided by the number of slice LUTs used for logic given by the FPGA design tool called XFLOW and the time is simply the critical path delay. Quantity AT^2 is useful, when clock speed is the driving factor of design optimization, for high-throughput realizations.

Table 3.11 shows the estimated power consumption for the Daub-4 and -6 filter banks.

Xilinx power analyzer (XPA) was employed to analyze the power consumption on Xilinx FPGA Virtex-6 device. The quiescent (static) power dissipation is a combined effect of standby and leakage power (dominant) dissipations [66]. At 40 nm process technology static power dominates dynamic power. Dynamic power represents the fluctuating power as the design runs and is the sum of short-circuit and capacitive (switching of logic cells) power dissipations. Leakage and standby currents

Table 3.10: Resource Consumption for Proposed Daub-4 Architecture

Resource	Word length				
	8 bit	10 bit	12 bit	14 bit	16 bit
Slices	2482	2718	3020	3362	3669
LUTs	8119	9438	10495	11652	12887
CLB	3916	4486	4879	5371	5825
CPD (ns)	3.54	3.91	4.28	4.83	5.44
AT ($\times 10^{-5}$)	1.38	1.75	2.08	2.59	3.16
AT^2 ($\times 10^{-13}$)	4.90	6.85	8.93	12.52	17.23
Max. Freq. (MHz)	282.50	255.80	233.10	207.00	183.80

Table 3.11: Resource Consumption for Proposed Daub-6 Architecture

Resource	Word length				
	8 bit	10 bit	12 bit	14 bit	16 bit
Slices ($\times 10^3$)	17.64	19.55	20.99	22.03	23.47
LUTs ($\times 10^3$)	47.08	52.85	58.56	63.21	68.72
CLB ($\times 10^3$)	21.97	24.30	26.58	28.84	30.46
CPD (ns)	7.04	7.44	8.02	8.67	8.96
Area-time ($\times 10^{-5}$)	15.4	18.0	21.3	25.0	27.2
Area-time^2 ($\times 10^{-13}$)	10.7	13.4	17.1	21.6	24.4
Max. Freq. ($\times$100 MHz)	1.464	1.39	1.327	1.264	1.218

Table 3.12: Comparison of Proposed Architectures with Existing 2-D DWT Architectures

	Tze-Yun *et al.* [60]	Marino *et al.* [61]	Po-Chih *et al.* [62]	Bing-Fei *et al.* [35]	Hongyu *et al.* [63]	Kishore *et al.* [64]	Zhang *et al.* [65]	Proposed Daub-4	Proposed Daub-6
LR [1]	No	No	No	No	No	No	No	Yes	Yes
Wavelet	Daub-4	N/A	9/7	5/3, 9/7	9/7	5/3, 9/7	9/7	Daub-4	Daub-6
Structure	RPA[2]	MRPA[3]	RPA	RPA	RPA	Lifting	Lifting	See Fig. 3.1	See Fig. 3.4
CQ[4] Noise	Yes	Yes	Yes	Yes	Yes	Yes	Yes	No	No
Multiplierless	Yes	No	No	No	No	No	No	Yes	Yes
H. U.[5]	100%	N/A	N/A	100%	50–70%	N/A	N/A	∼100%	∼100%

[1] Locally Reproduced [2] Recursive Pyramid Algorithm (RPA) [3] Modified Recursive Pyramid Algorithm (MRPA) [4] Coupled Quantization (CQ) [5] Hardware Utilization (HU)

Table 3.13: Comparison of proposed architectures with existing 2-D DWT architectures (Continued)

DWT/ IDWT	DWT/ IDWT	DWT	DWT	DWT	DWT	DWT	DWT	DWT	DWT
SNR/ PSNR [6]	N/A	N/A	N/A	N/A	69.14/ 74.85 [7]	N/A	39 [8]	65.36/ 66.82 [9]	64.18/68.12 [9]
Logic cells	N/A	N/A	N/A	N/A	879	N/A	986	426	1040
Throughput	N/A	1 ip/ op	2 ip/ op	1 ip/ op	2 ip/ op	1 ip/op	1 ip/ op	1 ip/ op	1 ip/ op
Technology	Xilinx XC2V4000	N/A	N/A	CMOS 0.25um	Virtex-II XC2V250	CMOS 0.18um	Xilinx xc2v500	Xilinx xc6vcx240t	Xilinx xc6vcx240t
Max. Freq. (MHz)	N/A	N/A	N/A	200	50	200	98	282.50	146.42
AI encoding	No	No	No	No	No	No	No	Yes	Yes

[6] For 8-bit Lena Image [7] Using 16-bit multipliers [8] PSNR (SNR is N/A) [9] 8-bit FRS

Table 3.14: Memory Requirements

	FIFO requirements	
Level	Daub-4	Daub-6
1	$4+3N$	$9+45\frac{N}{2}$
$n \geq 2$	$16+24\frac{N}{2^n}$	$36+\frac{90N}{2^n}$

do exist in digital circuits and are reported for the entire FPGA. Dynamic power consumption is associated only with the logic of the design under test. Therefore, reported static power for FPGAs can be several Watts and has limited usefulness as a metric. Table 4.9 lists the dynamic power consumption for the single level decomposition. The quiescent power reported for FPGAs are for the entire chip, not just the relevant parts of the particular design being tested.

The memory requirement expressed as 1-deep FIFO elements count for the Daub-4 and -6 schemes are given in the Table 3.10 as a function of image size N in pixels and number of decomposition levels n.

The SNR and peak PSNR were adopted as figures of merit. Table 3.12 provides these quantities for standard input images from [59] for the Daub-4 and -6 architectures. Both fixed-point and AI encoding schemes were considered, for 8-bit input word length. Table 3.6 shows a comparative performance analysis for the proposed Daub-4 and -6 designs.

Table 3.15: Daub-4 and Daub-6 ISE XPower Results

Word length	Power (Watt)							
	Daub-4				Daub-6			
	Clock net	Quiescent	Dynamic	Total	Clock net	Quiescent	Dynamic	Total
8	0.035	2.573	0.164	2.737	0.097	4.423	0.339	4.762
10	0.038	2.573	0.184	2.757	0.112	4.423	0.348	4.771
12	0.044	2.574	0.201	2.784	0.124	4.424	0.360	4.784
14	0.049	2.574	0.226	2.796	0.137	4.424	0.378	4.802
16	0.055	2.575	0.237	2.812	0.145	4.425	0.408	4.833

3.4.2 Comparison with Existing Methods

A significant amount of work is published on 1-D and 2-D DWT VLSI architectures [1–3, 6, 19, 31, 63, 67–69]. In particular, the designs proposed [3, 6] address the Daub-4 and -6 wavelet analysis. Also detailed data is reported in [3, 6] allowing us to derive meaningful comparisons.

Considering 8-bit input word length, the obtained SNR and PSNR values for proposed architectures, were roughly 30–40% higher than the 1-D and 2-D DWT architectures described in [3, 6].

Among the FRS approaches we have mentioned, we used canonical signed digit (CSD) approximation for comparison. Moreover, we compared the proposed architectures with several prominent VLSI 2-D DWT designs archived in literature. In particular, we separated the following works: [35, 60–65]. Table 3.9 shows the comparison results.

The proposed architectures are also compared with recently published AI based DWT architectures. Table 4.9 shows the comparison results. Notice that

Table 3.16: SNR and PSNR for Daub-4 and Daub-6 Filter Banks Based on Fixed-point and AI Encoding

Wavelet Filter	Scheme Used	Measured Aspect	Original Image (512×512)	Approximate Sub-image			
				$\mathbf{A}_1$ (256×256)	$\mathbf{A}_2$ (128×128)	$\mathbf{A}_3$ (64×64)	$\mathbf{A}_4$ (32×32)
Daub-4	Fixed-point	SNR (dB)	Cameraman	40.03	33.35	31.72	31.90
			Woman	42.28	36.79	38.84	34.52
			Reflection	38.18	34.64	26.98	25.03
		PSNR (dB)	Cameraman	43.03	39.38	34.69	34.54
			Woman	44.05	38.79	39.84	36.75
			Reflection	42.21	39.80	30.21	32.08

Table 3.17: SNR and PSNR for Daub-4 and Daub-6 Filter Banks Based on Fixed-point and AI Encoding (Continued)

Wavelet Filter	Scheme Used	Measured Aspect	Original Image (512 × 512)	Approximate Sub-image			
				$\mathbf{A}_1$ (256 × 256)	$\mathbf{A}_2$ (128 × 128)	$\mathbf{A}_3$ (64 × 64)	$\mathbf{A}_4$ (32 × 32)
Daub-4	AI Encoding	SNR (dB)	Cameraman	63.26	61.27	57.18	52.83
			Woman	64.16	60.53	51.37	59.86
			Reflection	66.53	61.29	43.72	52.71
		PSNR (dB)	Cameraman	72.20	64.33	60.82	56.21
			Woman	71.12	66.27	67.64	64.47
			Reflection	68.38	66.71	56.18	54.61

Table 3.18: SNR and PSNR for Daub-4 and Daub-6 Filter Banks Based on Fixed-point and AI Encoding (Continued)

Wavelet Filter	Scheme Used	Measured Aspect	Original Image (512 × 512)	Approximate Sub-image $\mathbf{A}_1$ (256 × 256)	$\mathbf{A}_2$ (128 × 128)	$\mathbf{A}_3$ (64 × 64)	$\mathbf{A}_4$ (32 × 32)
Daub-6	Fixed-point	SNR (dB)	Mandrill	37.78	33.72	33.20	32.21
			Lena	38.75	33.19	31.23	29.80
			CT head	38.18	34.64	26.98	25.03
		PSNR (dB)	Mandrill	39.36	35.57	35.27	33.32
			Lena	41.13	35.46	35.69	32.21
			CT head	43.44	39.35	33.54	37.28

Table 3.19: SNR and PSNR for Daub-4 and Daub-6 Filter Banks Based on Fixed-point and AI Encoding (Continued)

Wavelet Filter	Scheme Used	Measured Aspect	Original Image (512×512)	Approximate Sub-image			
				$\mathbf{A}_1$ (256×256)	$\mathbf{A}_2$ (128×128)	$\mathbf{A}_3$ (64×64)	$\mathbf{A}_4$ (32×32)
Daub-6	AI Encoding	SNR (dB)	Mandrill	61.45	59.81	56.63	53.76
			Lena	64.18	59.12	58.14	55.76
			CT head	62.53	58.71	53.82	52.14
		PSNR (dB)	Mandrill	64.28	62.86	60.28	58.03
			Lena	68.12	64.23	63.68	61.28
			CT head	66.45	62.12	55.17	58.74

the Daub-6 FRS design based on the expansion factor method could offer a 7% improvement in the clock frequency when compared to the design based on CSD representation.

To compare with other architectures, PSNR values presented in Table 4.9, for proposed Daub-4 and -6 architectures, were obtained by employing reconstructions between column and row transforms, whereas the PSNR values in Table 3.12 are entirely 2-D based with *single* final reconstruction step. All values mentioned in Table 4.9 are for the CSD representation considering 8-bit equivalent word size, unless it is specifically mentioned that we employed the expansion factor method. The comparison is provided in Table 4.9.

The proposed architectures are entirely multiplier free with no coupled quantization noise; possess low levels of both uncorrelated and uncoupled quantization noise; and offer the maximum frequency of operation among others. Since the design is speed optimized using fine-grain pipelining and parallel architectures, it is not anticipated to yield advantages in terms of power and area. In a sense, we traded the speed (maximum frequency) for power and resources.

3.5 Conclusion

We proposed a multi-encoded AI-based 2-D wavelet filter bank architecture capable of arbitrarily high numerical accuracy. The introduced design employs AI-based arithmetic which is (i) error-free, (ii) defined over integers, and (iii) free of multiplications.

Table 3.20: Comparison of Proposed Architectures with Existing AI Based DWT Architectures

	Wahid *et al.* [1]	Wahid *et al. [6]*	Wahid *et al.* [19]	Wahid *et al. [3]*	Wahid *et al.* [31]	Wahid *et al.* [17]	Gustafsson *et al.* [67]	Proposed Daub-4	Proposed Daub-6
LR[1]	No	No	No	No	No	No	No	Yes [2]	Yes [2]
Wavelet	Daub-4/6	Daub-4/6	Daub-6	Daub-4/6	Daub-8	Daub-4/6	Daub-6	Daub-4	Daub-6
Architecture	1-D/ 2-D	1-D	1-D	1-D	1-D	1-D	1-D	2-D	2-D
IRS[3]	Yes	Yes	Yes	Yes	Yes	Yes	N/A	No	No
HC[4]	10/ 18	10/ 25	25	N/A	N/A	9/ 18	18	10	21

[1] Locally Reproduced (LR) [2] Measured [3] Intermediate Reconstruction Step (IRS)
[4] Hardware Complexity (HC) (# adders) * Employing reconstruction between row and column.
[†] Taken from PSNR plots in [1, p. 1264] [‡] At 50 MHz [17] [§] At 442.47 MHz [*] At 274.72 MHz

Table 3.21: Comparison of proposed architectures with existing AI based DWT architectures (Continued)

Registers[5]	200/ 494	N/A	N/A	N/A	186	115/196	N/A	258	765
Logic cells[5]	248/680	N/A	N/A	N/A	518	106/254	N/A	426	1040
PSNR (dB)[6]	38 †(Daub-4) 39 †(Daub-6)	50.49 (Daub-4) 49.87 (Daub-6)	N/A	22.26	90	N/A	N/A	54.64⋆	57.12⋆
FRS[7]	CSD	CSD	Booth	Booth	Booth	CSD	CSD	CSD	CSD/EFM[8]
Throughput	1 ip/ op	1 ip/ op	N/A	N/A	N/A	2 ip/op	1 ip/ op	1 ip/ op	1 ip/ op
DP[9](mW)[5]	15.94/ 22.29	N/A	N/A	N/A	4.8	4.51‡	N/A	38§	57*
Technology	Xilinx VirtexE	N/A	N/A	N/A	Xilinx VirtexE	CMOS 0.18um	N/A	Xilinx xc6vcx240t	Xilinx xc6vcx240t
MF [10](MHz)	148.21/119.57	N/A	N/A	N/A	71	100	N/A	282.50	146.42/ 157.16[6]

[5] For single level decomposition [6] For 8-bit Lena Image [7] Final Reconstruction Step (FRS)
[8] Expansion Factor Method (EFM) [9] Dynamic Power (DP) [10] Maximum Frequency (MF)
⋆ Employing reconstruction between row and column.
† Taken from PSNR plots in [1, p. 1264] ‡ At 50 MHz [17] § At 442.47 MHz * At 274.72 MHz

By employing AI encoding, resulting wavelet decomposed images had SNR and PSNR figures improved by approximately 25–30% when compared to a counterpart fixed-point system with 8-bit word length and 6 fractional bits.

Comparing the paper [1], our proposed Daub-4 and -6 architectures The SNR and PSNR values for the AI-based Daub-6 architecture were approximately 6–10% higher than the figures obtained from the Daub-4 architecture. The better mathematical properties of the Daub-6 wavelets, such as more vanishing moments, explains this difference. Due to its inherent simplicity of coefficients and smaller number of AI numbers, the Daub-4 AI-based architecture had consumed approximately 50% lower power than the Daub-6 systems. Moreover, its maximum frequency of operation is approximately 90% higher under the same conditions.

A *single* FRS is the only source of computational error. Noise injection from intermediate fixed-point errors is non-existent. We proposed several designs for the FRS based on CSD representation and expansion factor scaling. These two methods allowed various configurations of accuracy and tolerable circuit complexities. Applications exist in sub-band coding of high dynamic range image sequences. Standard images were analyzed. FPGA based four-level prototypes for Daubechies 4- and 6-tap wavelet filters are operational at a compilation target frequency of 100 MHz on the Xilinx ML605 board. Place-and-route timing analysis furnished 282.50 MHz and 146.42 MHz for the Daub-4 and -6 architectures, respectively. Daub-4 and -6 single level decomposition architectures were also FPGA prototyped with the Xilinx Virtex-6 device at 442.47 and 274.72 MHz, respectively. CMOS sensor arrays for

imaging are being continuously improved with increasing resolutions. The dynamic range of typical imaging applications are also increasing and more emphasis is being made for picture quality. In the presence of higher resolution, increased dynamic range, and increased frame rate, there is no option but to increase the throughput of the digital filtering architectures.

Finally, it is important to notice that—in principle—the discussed AI based scheme can be applied to any type of DWT as long as the scaling and wavelet coefficients of the corresponding filters could be given and exact representation. For instance, this is the case for the Haar, Daubechies-4/-6, and Bior-5/3 wavelets. On the other hand, wavelets such as gaussian, mexican hat do not have a compatible DWT version.

CHAPTER IV

AI BASED LOW ADDER COUNT ARCHITECTURE FOR THE 2-D DAUBECHIES 4-TAP WAVELET FILTER BANK

4.1 Introduction

This chapter provides a novel representation of the Daubechies 4-tap wavelet filters that aims at minimizing the filter bank arithmetic complexity and, therefore, leading to less demanding hardware requirements. In particular, we focus on the minimization of the number of canonical signed-digit (CSD) terms. Such minimization seeks to obtain a minimum number of adders for the associated hardware realization while maintaining all other features of the earlier design described in [70].

The proposed optimal architecture is consistent with the method advanced in [67] which used integer linear programming for minimizing the number of adders required for the hardware realization of the 1-D Daubechies 6-tap filter banks. The paper unfolds as follows. Section 4.2 describes the mathematical framework of 2-D multi level AI architectures for Daubechies-4 wavelets as detailed in [71]. Section 4.3 furnishes the numerical optimization that leads to a fast algorithm having reduced number of 2-input adder circuits. Section 5.6.4 provides extensive digital design details for practical implementations using reconfigurable logic devices as well as

CMOS technology. The paper is concluded in Section 5.7.

4.2 AI Encoding of Daubechies 4-tap Filter Bank

In this section, we review the previously proposed AI encoding approach for the implementation of Daubechies 4-tap filters.

4.2.1 Daubechies 4-tap filter

The filter bank structure based on the Daubechies 4-tap filter possesses a low-pass filter whose impulse response is furnished by:

$$\mathbf{h} = \frac{1}{4\sqrt{2}} \begin{bmatrix} 1+\sqrt{3} & 3+\sqrt{3} & 3-\sqrt{3} & 1-\sqrt{3} \end{bmatrix}^{\top}, \tag{4.1}$$

where the superscript $^{\top}$ denotes the transpose operation. The presence of irrational quantities, in particular $\sqrt{3}$, poses difficulties when considering fixed-point representation. Conventional implementations often resort to truncation and/or rounding-off operations as a means to approximate $\sqrt{3}$ to a representable quantity in fixed-point arithmetic. However such procedure inevitably introduces computational errors.

In [70, 71], it was shown that (5.1) can be split in two integer coefficient filters. For such, consider the algebraic integer $\zeta = \sqrt{3}$. Therefore, we obtain:

$$\mathbf{h} = \frac{1}{4\sqrt{2}}(\mathbf{h_1} + \zeta \cdot \mathbf{h}_\zeta), \tag{4.2}$$

where $\mathbf{h_1} = \begin{bmatrix} 1 & 3 & 3 & 1 \end{bmatrix}^\top$ and $\mathbf{h}_\zeta = \begin{bmatrix} 1 & 1 & -1 & -1 \end{bmatrix}^\top$. This approach allows integer computation by not explicitly evaluating ζ until the final stage of computation. As a consequence, error propagation is prevented and final results can be appropriately handled.

4.2.2 Mathematical Notation

Let $\mathbf{C}$ be an $N \times N$ matrix with columns $\mathbf{c}_j$, $j = 0, 1, \ldots, N-1$, $\mathbf{C} = \begin{bmatrix} \mathbf{c}_0 & \mathbf{c}_1 & \mathbf{c}_2 & \cdots & \mathbf{c}_{N-1} \end{bmatrix}$ and $\mathbf{v}$ be an N-point column vector. The operation $\oplus\!\!\!\!\!\!\bigcirc$ is defined according to:

$$\begin{aligned} \mathbf{v} \oplus\!\!\!\!\!\!\bigcirc\, \mathbf{C} &\triangleq (2{\downarrow}1) \begin{bmatrix} \mathbf{v} * \mathbf{c}_0 & \mathbf{v} * \mathbf{c}_1 & \mathbf{v} * \mathbf{c}_2 & \cdots & \mathbf{v} * \mathbf{c}_{N-1} \end{bmatrix} \\ &= \begin{bmatrix} \mathbf{v} * \mathbf{c}_0 & \mathbf{v} * \mathbf{c}_2 & \cdots & \mathbf{v} * \mathbf{c}_{N-2} \end{bmatrix}, \end{aligned}$$

where $*$ is the convolution operation. Similarly, operation $\ominus$ is given by: $\mathbf{v} \ominus \mathbf{C} \triangleq \left(\mathbf{v} \oplus\!\!\!\!\!\!\bigcirc\, \mathbf{C}^\top\right)^\top$. In other words, operators $\ominus$ and $\oplus\!\!\!\!\!\!\bigcirc$ are the convolution operation along the rows and columns of a given matrix, respectively, followed by a dyadic down-sampling. Symbols $2{\downarrow}1$ and $1{\downarrow}2$ are used to denote the column-wise and row-wise down-sampling. respectively [47, pp. 6-26].

4.2.3 Daubechies 4-tap filter bank

Let $\mathbf{A}_0$ be an $N \times N$ pixels input image and $\mathbf{A}_1$ be the resulting $N/2 \times N/2$ image approximation after the first level wavelet decomposition. For the one-level

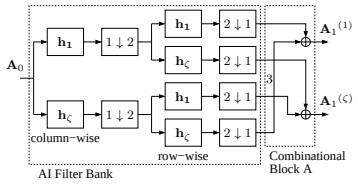


Figure 4.1: Single AI filter bank decomposition.

decomposition, these images are related according to:

$$\beta^2 \cdot \mathbf{A}_1 = \mathbf{h} \ominus \mathbf{h} \oslash \mathbf{A}_0.$$

As shown in [70, 71], the approximation image $\mathbf{A}_1$ consists of two integer enconded parts $\mathbf{A_1}^{(1)}$ and $\mathbf{A_1}^{(\zeta)}$. Superscripts $^{(1)}$ and $^{(\zeta)}$ denote image parts associated to AI basis elements 1 and ζ, respectively. These quantities satisfy the following relation [70, 71]:

$$\beta^2 \cdot \mathbf{A}_1 = \mathbf{A_1}^{(1)} + \zeta \cdot \mathbf{A_1}^{(\zeta)},$$

where $1/\beta = 4\sqrt{2}$. The constituents parts of $\mathbf{A}_1$ are obtained according to the filter bank structure shown in Fig. 4.1.

This process can be iterated to furnish subsequent coarser approximations. Thus the nth approximation $\mathbf{A}_n$ is furnished by $\beta^n \cdot \mathbf{A}_n = \mathbf{h} \ominus \mathbf{h} \oslash \mathbf{A}_{n-1}$ [70, 71].

Moreover, it was established that $\beta^{2n} \cdot \mathbf{A}_n = \mathbf{A_n}^{(1)} + \zeta \cdot \mathbf{A_n}^{(\zeta)}$. Fig. 4.2 shows multi-level AI filter bank for 4-level decomposition with final reconstruction step (FRS). Combinational Block A is detailed in Fig. 4.1, whereas Combination Block B

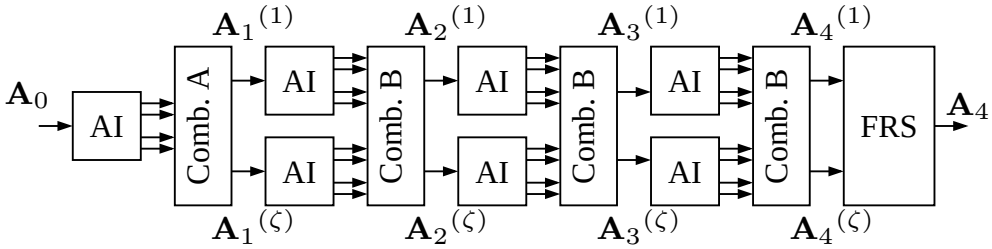


Figure 4.2: Multi-level AI filter bank with final reconstruction step (FRS)

is discussed in Section 4.3.

4.3 Optimized AI Encoding

4.3.1 Number of Additions

The AI encoding discussed in previous section implies integer coefficient filter $\mathbf{h_1}$ and $\mathbf{h}_\zeta$. Whereas the coefficients of $\mathbf{h}_\zeta$ can be trivially implemented without requiring multiplication or bit-shifting operations, the same is not true for the coefficients of $\mathbf{h_1}$. In $\mathbf{h_1}$, the required multiplications by 3 imply extra additions. This is due to the fact that the constant 3 can be implemented by means of one addition and one bit-shift operation ($3 = 2 + 1 = 4 - 1$).

In general terms, depending on the binary representation of the filter coefficients, they may contribute with extra additions. Let us adopt the canonical-signed-digit representation for binary encoding the integer coefficients. Additionally, let $\mathrm{S_{CSD}}(n)$ return the number of additions/subtractions of powers of two required to represent a given integer n. For example, we have: $\mathrm{S_{CSD}}(2) = 0$, $\mathrm{S_{CSD}}(3) = 1$, $\mathrm{S_{CSD}}(11) = 2$ ($11 = 16 - 4 - 1$). Clearly, if all elements of a given filter are pow-

ers of two, then no extra addition is required. In this case, an additive complexity minimum is achieved.

4.3.2 Filter Parametrization and Optimization

Our goal is to rewrite (5.2) is such way that the coefficients of the resulting filters can be represented in CSD form with minimum number of additions. For such let us introduce an integer parameter m in (5.2) as follows:

$$\begin{aligned}\mathbf{h} &= \begin{bmatrix}1 & 3 & 3 & 1\end{bmatrix}^\top - m \cdot \begin{bmatrix}1 & 1 & -1 & -1\end{bmatrix}^\top \\ &\quad + (\zeta + m) \cdot \begin{bmatrix}1 & 1 & -1 & -1\end{bmatrix}^\top \\ &= \begin{bmatrix}1-m & 3-m & 3+m & 1+m\end{bmatrix}^\top \\ &\quad + (\zeta + m) \cdot \begin{bmatrix}1 & 1 & -1 & -1\end{bmatrix}^\top .\end{aligned}$$

Above expression implies two integer filters:

$$\begin{aligned}\mathbf{h_1}' &= \begin{bmatrix}1-m & 3-m & 3+m & 1+m\end{bmatrix}^\top , \\ \mathbf{h}_\zeta{}' &= \mathbf{h}_\zeta = \begin{bmatrix}1 & 1 & -1 & -1\end{bmatrix}^\top .\end{aligned}$$

As a consequence, the required AI element must be replaced for $\zeta' = \zeta + m$. Notice that for $m = 0$ we obtain (5.2).

The elements of $\mathbf{h}_\zeta{}'$ pose no arithmetic complexity. We focus our attention

on $\mathbf{h_1}'$ aiming at finding a suitable value of m such that the resulting coefficients are efficiently represented in CSD representation.

Let $h_1'[k]$ and $h_\zeta[k]$, $k = 0, 1, 2, 3$, be the coefficients of $\mathbf{h_1}'$ and $\mathbf{h}_\zeta$, respectively. In view of the above discussion, the optimal value of m, denoted m^*, is the solution of the following minimization problem:

$$\begin{aligned} m^* &= \arg\min_{m\in\mathbb{Z}} \sum_{k=0}^{3} \mathrm{S}_{\mathrm{CSD}}(h_1'[k]) \\ &= \arg\min_{m\in\mathbb{Z}} \Big\{ \mathrm{S}_{\mathrm{CSD}}(1-m) + \mathrm{S}_{\mathrm{CSD}}(3-m) \\ &\qquad + \mathrm{S}_{\mathrm{CSD}}(1+m) + \mathrm{S}_{\mathrm{CSD}}(3+m) \Big\}. \end{aligned} \tag{4.3}$$

Direct optimization tools are not applicable due to the difficulty in analytically manipulate (5.3). Therefore, we resort to numerical search methods as a means to solve (5.3). Such computational search requires that we limit the search space. Let $\{m \in \mathbb{Z} : |m| \leq 1024\}$ be the considered search space. Under these conditions, we could obtain $m^* = \pm 1$, which results in the following filters:

$$\mathbf{h_1}' = \begin{bmatrix} 2 & 4 & 2 & 0 \end{bmatrix}^\top \quad \text{or} \quad \mathbf{h_1}' = \begin{bmatrix} 0 & 2 & 4 & 2 \end{bmatrix}^\top,$$

for $m^* = -1$ and $m^* = 1$, respectively. The implied algebraic integers are $\sqrt{3} \pm 1$, respectively. Notice that $m^* = \pm 1$ not only minimizes (5.3), but it is also a zero of the objective function. This can be directly verified since all coefficients of resulting filters $\mathbf{h_1}'$ are powers of two. Thus the CSD representation of theses coefficients

Table 4.1: Filter Parametrization and Optimization

m	# of CSD additions	$\mathbf{h_1}'$	$\zeta' = \zeta + m$
-5	1	$\begin{bmatrix}6 & 8 & -2 & -4\end{bmatrix}^\top$	$\sqrt{3}-5$
-3	1	$\begin{bmatrix}4 & 6 & 0 & -2\end{bmatrix}^\top$	$\sqrt{3}-3$
-1	0	$\begin{bmatrix}2 & 4 & 2 & 0\end{bmatrix}^\top$	$\sqrt{3}-1$
0	2	$\begin{bmatrix}1 & 3 & 3 & 1\end{bmatrix}^\top$	$\sqrt{3}$
1	0	$\begin{bmatrix}0 & 2 & 4 & 2\end{bmatrix}^\top$	$\sqrt{3}+1$
3	1	$\begin{bmatrix}-2 & 0 & 6 & 4\end{bmatrix}^\top$	$\sqrt{3}+3$
5	1	$\begin{bmatrix}-4 & -2 & 8 & 6\end{bmatrix}^\top$	$\sqrt{3}+5$

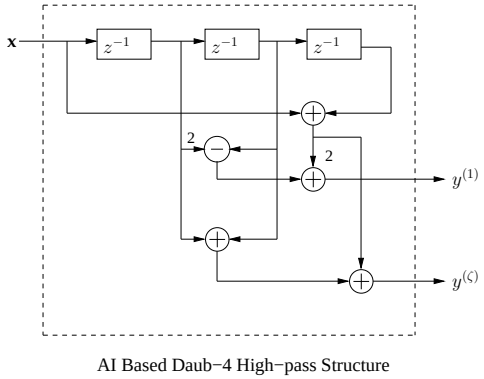


Figure 4.3: Proposed AI based Daubechies 4-tap high-pass filter.

require no extra additions.

Suboptimal solutions occur at $m = \pm 3$ and $m = \pm 5$. In both cases, the resulting filters are improvements over the original filter $\mathbf{h_1}$, requiring only one extra addition. Table 4.1 displays the obtained optimal and suboptimal results. Considering the adopted search space, all values of m—except for the values listed in Table 4.1—resulted in filters whose arithmetical complexity is greater than or equal to the the complexity of the original filter $\mathbf{h_1}$.

4.3.3 High-pass Filter Implementation

The signal analysis provided by the high-pass filter banks can be implemented in a similar fashion as described for the low-pass filter banks. In fact, the impulse response of an analysis high-pass filter is given by:

$$\mathbf{g} = \frac{1}{4\sqrt{2}} \begin{bmatrix} 1+\sqrt{3} & -3-\sqrt{3} & 3-\sqrt{3} & 1+\sqrt{3} \end{bmatrix}^{\top}. \tag{4.4}$$

Therefore, the filter $\mathbf{g}$ can be AI encoded according to the following formalism:

$$\mathbf{g} = \frac{1}{4\sqrt{2}} \begin{bmatrix} 2+\zeta' & -4-\zeta' & 2-\zeta' & 2+\zeta' \end{bmatrix}^{\top}. \tag{4.5}$$

Thus, we maintain the expression below:

$$\mathbf{g} = \frac{1}{4\sqrt{2}} (\mathbf{g_1} + \zeta' \cdot \mathbf{g}_{\zeta'}), \tag{4.6}$$

where $\mathbf{g_1} = \begin{bmatrix} 2 & -4 & 2 & 2 \end{bmatrix}^{\top}$ and $\mathbf{g}_{\zeta'} = \begin{bmatrix} 1 & -1 & -1 & 1 \end{bmatrix}^{\top}$. Notice that (4.6) forms the baseline for the multi-level decomposition required to compute the detail sub-images $\mathbf{Dv}_n$, $\mathbf{Dh}_n$, and $\mathbf{Dd}_n$ as shown in Fig. 2.1. Fig. 4.3 shows Daubechies 4-tap high-pass filter realization.

In fact, our architecture is an efficient framework for computing the standard Daubechies 4-tap filter bank. Because of it exactness, the proposed architecture does not inflict any change in the mathematical properties of the original Daubechies filter

bank. All good analytic features of the Daubechies filter bank are preserved, such as vanishing moments and zero dc leakage. Therefore, since all the multi-encoded AI bases in our work maintain *exact* computation up to the *single* final reconstruction step, the problem of energy leakage cannot arise in this type of implementation.

Additionally, at the FRS the accuracy of AI bases can be independently adjusted according to the required precision. This distinct feature not only allows arbitrarily high levels of precision for each of the coefficients, but also enables the coefficient errors in the final FRS to be completely decoupled from each other. Therefore, the problem of spectral leakage is not present, being irrelevant for this method of AI computation. The details of AI decoding operation of AI bases is discussed in the following section.

4.3.4 Final Reconstruction Step

The proposed multi-level analysis is computed entirely over the AI representation. However, the resulting AI encoded approximations ($\mathbf{A}_n{}^{(1)}$, $\mathbf{A}_n{}^{(\zeta')}$) must be converted to usual fixed-point arithmetic in order to be further processed by standard systems. The decoding operation consists of performing the following calculation:

$$\mathbf{A}_n = \frac{1}{\beta^{2n}}\Big(\mathbf{A}_n{}^{(1)} + \zeta' \cdot \mathbf{A}_n{}^{(\zeta')}\Big).$$

The factor β^{2n} is always a power of two, which can be conveniently implemented with bit-shift operations. This decoding operation is realized at the FRS block

Table 4.2: CSD Representation for ζ

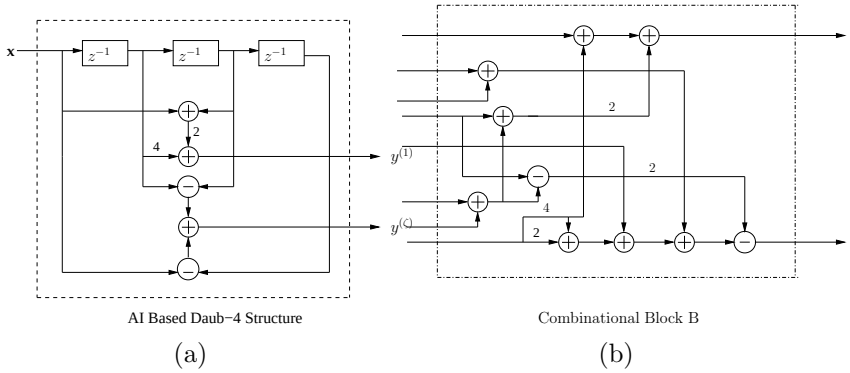
Word-length	CSD Encoding	% Rel. Error
8 bit	$1 - 2^{-2}$	0.0246
10 bit	$1 - 2^{-2} - 2^{-6}$	0.0032
12 bit	$1 - 2^{-2} - 2^{-6}$	0.0032
14 bit	$1 - 2^{-2} - 2^{-6} - 2^{-9}$	0.00040
16 bit	$1 - 2^{-2} - 2^{-6} - 2^{-9}$	0.00040

(Fig. 4.2). Therefore, the only possible source of errors in the proposed architecture is the multiplication by ζ'. The two optimal values of $m^* = \pm 1$ furnish $\zeta' = \sqrt{3} \pm 1$. Thus, since $\sqrt{3} - 1 \approx 0.73205$ has a simpler CSD representation, we select $m^* = -1$ as the final design choice. Table 4.2 displays the CSD encoding of $\zeta' = \sqrt{3} - 1$ for several word-lengths as well as the associate relative errors. Of course, the selected approximation depends on the application and its error tolerance.

4.3.5 Proposed Structures and Savings in Adders

Fig. 4.4 (a) displays the proposed filter structure to implement $\mathbf{h_1}$. This filter requires only 5 adders. In comparison, the previous design detailed in [71] requires seven additions. This represents a decrease of 28% in additive complexity. Each AI block, as shown in Fig. 4.1, contains three instantiations of $\mathbf{h_1}$ which implies a saving of six additions per AI block. Fig. 4.2 shows that an n-level decomposition, $n > 1$, requires $2n - 1$ AI blocks. Thus in terms of AI blocks, we have a saving of $6 \cdot (2n - 1) = 12 \cdot n - 6$ additions.

The associated Combinational Block B is shown in Fig. 4.4. This block



(a) (b)

Figure 4.4: (a) Proposed AI based Daubechies 4-tap filter; (b) Combinational Block B.

requires 10 additions, whereas the previously proposed design requires 8 additions. We need $n-1$ realizations of Combinational Block B (cf. Fig. 4.2) for a $n > 1$ decomposition. Thus, we have an increase of $2 \cdot (n-1)$ additions, when compared to previous design [71]. In both previous and proposed designs, the Combination Block A requires three additions; therefore it accounts for no complexity change.

At the FRS, the CSD representation of the new AI integer choice $\zeta' = \zeta - 1$ requires one less addition when compared to ζ, which was employed in previous design [71]. Thus, one extra addition is saved.

Above discussion is summarized in the following net savings in numbers of two input adder circuits:

$$\begin{aligned} \text{Total savings in adders} &= (12 \cdot n - 6) - (2 \cdot n - 2) + 1 \\ &= 10 \cdot n - 3, \quad n > 1. \end{aligned}$$

Table 4.3: Daub-4 Decomposition Approximations

Level	Base	Expression
1	$\mathbf{A_1}^{(1)}$	$\mathbf{h_1}' \ominus \mathbf{h_1}' \oplus \mathbf{A}_0 + 2 \cdot \mathbf{h}_\zeta \ominus \mathbf{h}_\zeta \oplus \mathbf{A}_0$
	$\mathbf{A_1}^{(\zeta)}$	$\mathbf{h}_\zeta \ominus \mathbf{h_1}' \oplus \mathbf{A}_0 + \mathbf{h_1}' \ominus \mathbf{h}_\zeta \oplus \mathbf{A}_0 - 2 \cdot \mathbf{h}_\zeta \ominus \mathbf{h}_\zeta \oplus \mathbf{A}_0.$
2	$\mathbf{A_2}^{(1)}$	$\mathbf{h_1}' \ominus \mathbf{h_1}' \oplus \mathbf{A_1}^{(1)} - 4 \cdot \mathbf{h}_\zeta \ominus \mathbf{h}_\zeta \oplus \mathbf{A_1}^{(\zeta)} + 2 \cdot \left(\mathbf{h}_\zeta \ominus \mathbf{h}_\zeta \oplus \mathbf{A_1}^{(1)} + (\mathbf{h}_\zeta \ominus \mathbf{h_1}' + \mathbf{h_1}' \ominus \mathbf{h}_\zeta) \oplus \mathbf{A_1}^{(\zeta)} \right)$
	$\mathbf{A_2}^{(\zeta)}$	$\mathbf{h_1}' \ominus \mathbf{h_1}' \oplus \mathbf{A_1}^{(\zeta)} + 6 \cdot \mathbf{h}_\zeta \ominus \mathbf{h}_\zeta \oplus \mathbf{A_1}^{(\zeta)} + \mathbf{h}_\zeta \ominus \mathbf{h_1}' \oplus \mathbf{A_1}^{(1)} + \mathbf{h_1}' \ominus \mathbf{h}_\zeta \oplus \mathbf{A_1}^{(1)}$
		$-2 \cdot \left(\mathbf{h}_\zeta \ominus \mathbf{h}_\zeta \oplus \mathbf{A_1}^{(1)} - (\mathbf{h_1}' \ominus \mathbf{h}_\zeta + \mathbf{h}_\zeta \ominus \mathbf{h_1}') \oplus \mathbf{A_1}^{(\zeta)} \right)$

Table 4.4: Daub-4 Decomposition Approximations (Continued)

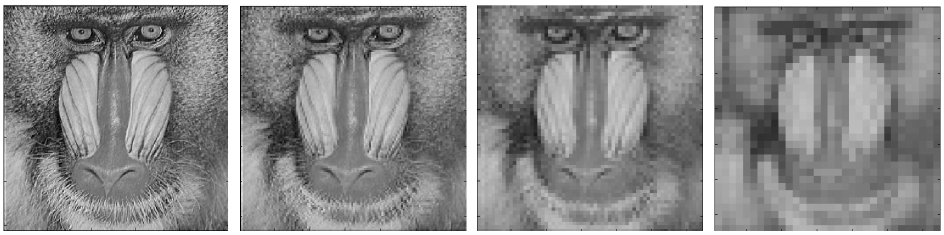
Level	Base	Expression
n	$\mathbf{A}_n{}^{(1)}$	$\mathbf{h_1}' \ominus \mathbf{h_1}' \oplus \mathbf{A}_{n-1}{}^{(1)} - 4 \cdot \mathbf{h}_\zeta \ominus \mathbf{h}_\zeta \oplus \mathbf{A}_{n-1}{}^{(\zeta)} + 2 \cdot \left(\mathbf{h}_\zeta \ominus \mathbf{h}_\zeta \oplus \mathbf{A}_{n-1}{}^{(1)} + (\mathbf{h}_\zeta \ominus \mathbf{h_1}' + \mathbf{h_1}' \ominus \mathbf{h}_\zeta) \oplus \mathbf{A}_{n-1}{}^{(\zeta)} \right)$
	$\mathbf{A}_n{}^{(\zeta)}$	$\mathbf{h_1}' \ominus \mathbf{h_1}' \oplus \mathbf{A}_{n-1}{}^{(\zeta)} + 6 \cdot \mathbf{h}_\zeta \ominus \mathbf{h}_\zeta \oplus \mathbf{A}_{n-1}{}^{(\zeta)} + \mathbf{h}_\zeta \ominus \mathbf{h_1}' \oplus \mathbf{A}_{n-1}{}^{(1)} + \mathbf{h_1}' \ominus \mathbf{h}_\zeta \oplus \mathbf{A}_{n-1}{}^{(1)}$
		$-2 \cdot \left(\mathbf{h}_\zeta \ominus \mathbf{h}_\zeta \oplus \mathbf{A}_{n-1}{}^{(1)} - (\mathbf{h_1} \ominus \mathbf{h}_\zeta + \mathbf{h}_\zeta \ominus \mathbf{h_1}) \oplus \mathbf{A}_{n-1}{}^{(\zeta)} \right)$

4.4 FPGA and ASIC Implementation and Results

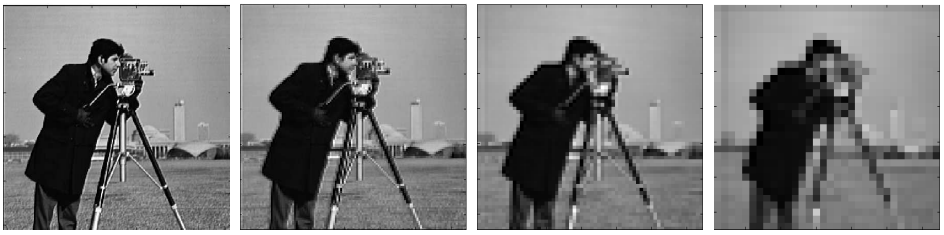
The architectures for Daubechies 4-tap filter bank were physically implemented and hardware co-simulated the Xilinx Virtex xc6vcx240t-1ff1156 FPGA device using the ML605 evaluation board. The proposed designs were also implemented with CMOS 45 nm ASIC technology upto synthesis level at supply voltage $V_{\mathrm{DD}} = 1.1$ V. The designs were tested with three different standard images obtained from [59]. Gray 512×512 images *Lena*, *Mandrill*, and *Cameraman* were submitted, block-by-block, in row-parallel format, to the proposed architecture. Hardware results were verified with MATLAB. Fig. 5.6 displays hardware results of on-chip physical verification on a Virtex-6 vcx240t-1ff1156 for the proposed Daubechies 4-tap filter bank. The implementation results for the Xilinx Virtex-6 vcx240t-1ff1156 FPGA device and the CMOS 45 nm ASIC technology are described in the following subsections.

The SNR and peak PSNR were adopted as image quality measures. Table 5.14 provides these quantities for above mentioned images for both the standard fixed-point implementation and the proposed AI-based design. By employing AI encoding, resulting wavelet decomposed images had SNR and PSNR figures improved by approximately 30–35% when compared to a counterpart fixed-point system with 8-bit word length and 6 fractional bits. In both schemes, the considered word length was 8 bit.

(a) $\mathbf{A}_1$ (b) $\mathbf{A}_2$ (c) $\mathbf{A}_3$ (d) $\mathbf{A}_4$



(e) $\mathbf{A}_1$ (f) $\mathbf{A}_2$ (g) $\mathbf{A}_3$ (h) $\mathbf{A}_4$



(i) $\mathbf{A}_1$ (j) $\mathbf{A}_2$ (k) $\mathbf{A}_3$ (l) $\mathbf{A}_4$

Figure 4.5: Approximation sub-images $\mathbf{A}_1$, $\mathbf{A}_2$, $\mathbf{A}_3$, and $\mathbf{A}_4$ obtained from on-chip physical verification on a Virtex-6 vcx240t-1ff1156.

Table 4.5: SNR and PSNR for Daubechies 4-tap Filter Bank Based on Fixed-point and AI Encoding

Wavelet Filter	Scheme Used	Measured Aspect	Original Image (512 × 512)	Approximate Sub-image			
				$\mathbf{A}_1$ (256 × 256)	$\mathbf{A}_2$ (128 × 128)	$\mathbf{A}_3$ (64 × 64)	$\mathbf{A}_4$ (32 × 32)
Daub-4	Fixed-point	SNR (dB)	Mandrill	41.23	35.38	33.06	31.06
			Lena	41.71	38.12	35.64	33.82
			Cameraman	40.18	38.64	35.98	32.03
		PSNR (dB)	Mandrill	46.03	42.38	40.69	38.81
			Lena	45.05	42.79	41.84	38.75
			Cameraman	46.21	43.80	41.21	39.08

Table 4.6: SNR and PSNR for Daubechies 4-tap Filter Bank Based on Fixed-point and AI Encoding (Continued)

Wavelet Filter	Scheme Used	Measured Aspect	Original Image (512 × 512)	Approximate Sub-image			
				$\mathbf{A}_1$ (256 × 256)	$\mathbf{A}_2$ (128 × 128)	$\mathbf{A}_3$ (64 × 64)	$\mathbf{A}_4$ (32 × 32)
Daub-4	AI Encoding	SNR (dB)	Mandrill	60.26	58.27	54.18	50.83
			Lena	61.16	58.29	55.48	53.86
			Cameraman	61.35	58.07	55.12	52.71
		PSNR (dB)	Mandrill	65.20	62.38	58.17	55.91
			Lena	64.78	62.27	59.64	56.47
			Cameraman	65.16	61.03	59.82	55.37

4.4.1 Resource Consumption and Figures of Merit

Xilinx Virtex-6 Implementation

Table 4.5 lists resource consumption for the number of slice registers, look-up table (LUT) count, and configurable logic blocks (CLB). Critical path delays (CPD) and the maximum operating frequency are also reported. We considered the following word length sizes: 8, 10, 12, 14, and 16 bits.

We adopt the area-time (AT) and area-time^2 (AT^2) as figures of merit. The area-time product is a standard performance metric in digital hardware designs. It refers to chip-area and speed (maximum frequency) of the design. Lower area-time values indicate a higher speed of operation. In an FPGA implementation, the area is provided by the number of slice LUTs employed for logic given by the FPGA design tool called XFLOW and the time is simply the critical path delay. Quantity area-time^2 is useful, when clock speed is the driving factor of design optimization, for high-throughput realizations.

Table 4.6 shows the estimated power consumption obtained from Xilinx power analyzer (XPA) in Xilinx-ISE. Clock net, quiescent, and dynamic powers are reported for the considered input word lengths.

The FPGA implementation employs an oscillator frequency $F_{\text{clock}} = 100$ MHz. Thus the final throughput is F_{clock}/N^2 image frames per second. Thus, for 512×512 input images, a frame rate of ≈ 381 Hz was obtained.

Table 4.7: Hardware Resource Consumption for Xilinx Virtex-6 vcx240t-1ff1156 Implementation

Resource	Word length				
	8 bit	10 bit	12 bit	14 bit	16 bit
Registers	2,194	2,410	2,828	3,252	3,648
LUTs	8,106	9,382	10,258	11,196	12,346
CLB	3,102	3,289	3,761	4,286	4,815
CPD (ns)	3.81	3.94	4.26	4.67	5.14
Area-time ($\times 10^{-5}$)	1.18	1.29	1.60	2.00	2.47
Area-time^2 ($\times 10^{-14}$)	4.50	5.10	6.82	9.34	12.72
Max. Freq. (MHz)	263.15	252.25	239.96	220.05	204.18

Table 4.8: ISE XPower Results

Power (Watt)	Word-length				
	8 bit	10 bit	12 bit	14 bit	16 bit
Clock net	0.048	0.053	0.057	0.062	0.068
Quiescent	2.562	2.563	2.563	2.565	2.565
Dynamic	0.226	0.248	0.27	0.291	0.318
Total	2.788	2.811	2.833	2.856	2.883

CMOS 45 nm ASIC Synthesis

Table 4.7 and 5.12 show resource consumption and estimated power consumption with regard to the ASIC 45 nm implementation generated by Encounter® RTL compiler [72]. We refer to leakage power, dynamic power, and total power as L_p, D_p, and T_p, respectively. The ASIC implementation of the proposed architecture yielded a maximum frequency of 523.56 MHz for 8-bit input data as shown in Table 4.7. The considered supply voltage was $V_{\mathrm{DD}} = 1.1$ V.

4.4.2 Comparison with Existing Methods

Table 4.9: Hardware Resource Consumption for CMOS 45 nm ASIC Synthesis (Supply Voltage $V_{\mathrm{DD}} = 1.1$ V)

Word length	8 bit	10 bit	12 bit	14 bit	16 bit
ASIC Gate Count	360566	402817	443094	494190	528258
Area (mm^2)	1.59	1.78	1.96	2.21	2.33
CPD (time in ns)	1.912	1.931	1.958	1.976	1.997
Area-time	3.04	3.41	3.84	4.38	4.65
Area-time^2	5.74	6.64	7.514	8.41	9.29
F_{max} (MHz)	523.56	516.62	511.51	505.23	500.75

Table 4.10: Power Consumption for CMOS 45 nm ASIC Synthesis (Supply Voltage $V_{\mathrm{DD}} = 1.1$ V)

Word length	8 bit	10 bit	12 bit	14 bit	16 bit
L_p (mW)	13.23	14.64	16.23	17.78	19.22
D_p (mW)	1774.51	1950.26	2172. 80	2299.70	2580.73
T_p (mW)	1787.73	1964.38	2189.01	2317.48	2599.95

The proposed architecture is compared with published AI based DWT architectures [1, 3, 6, 17, 19, 67, 70, 71]. Results are presented in Table 4.9. To compare with other architectures, PSNR values presented in Table 4.9, for proposed architecture, were obtained by employing reconstructions between column and row transforms, whereas the PSNR values in Table 5.14 are entirely 2-D based with *single* final reconstruction step. The proposed architecture provides advantage in hardware complexity (# adders) when compared to remaining considered architectures.

The proposed architecture is entirely multiplier-free with no coupled quantization noise. It also possesses low levels of both uncorrelated and uncoupled quantization noise and outperforms other architectures in terms of the maximum frequency of operation.

4.5 Conclusion

We proposed an optimized multi-rate wavelet filter bank architecture which is AI-based, and multi-encoded. Additionally, the introduced design is capable of furnishing arbitrarily high numerical accuracy using error-free integer arithmetic. In fact, the proposed architecture preserves all features of our earlier architectures such as (i) error-free computation, (ii) defined over integers, and (iii) free of multiplications. It also reduces hardware complexity (number of adders) leading to considerable reduction in cost for the hardware realization of multi-level DWTs of Daubechies 4-tap filter banks. We showed that approximately 10 adders per decomposition levels

Table 4.11: Comparison of Proposed Architectures with Existing AI Based DWT Architectures

	Wahid et al, [1]	Wahid et al, [6]	Wahid et al, [19]	Wahid et al, [3]	Madishetty et al, [71]	Wahid et al, [17]	Wahid et al, [20]	Proposed Daub-4
LR[1]	No	No	No	No	Yes [2]	No	No	Yes [2]
Wavelet	Daub-4/6	Daub-4/6	Daub-6	Daub-4/6	Daub-4/6	Daub-4/6	Daub-4	Daub-4
Architecture	1-D/ 2-D	1-D	1-D	1-D	2-D	1-D	1-D	2-D
IRS[3]	Yes	Yes	Yes	Yes	No	Yes	Yes	No
HC[4]	10/ 18	10/ 25	25	N/A	10/21	9/ 18	9	7
Registers[5]	200/ 494	N/A	N/A	N/A	258/765	115/196	200	236
Logic cells[5]	248/680	N/A	N/A	N/A	426/1040	106/254	N/A	483

[1] Locally Reproduced (LR) [2] Measured [3] Intermediate Reconstruction Step (IRS)
[4] Hardware Complexity (HC) (# adders) [5] For single level decomposition

Table 4.12: Comparison of Proposed Architectures with Existing AI Based DWT Architectures (Continued)

PSNR (dB)[6]	38[†](Daub-4) 39[†](Daub-6)	50.49 (Daub-4) 49.87 (Daub-6)	N/A	22.26	54.64 (Daub-4) 57.12 (Daub-6)	N/A	44.90	64.78
FRS[7]	CSD	CSD	Booth	Booth	CSD/EFM [8]	CSD	CSD	CSD
Throughput	1 ip/ op	1 ip/ op	N/A	N/A	1 ip/op	2 ip/op	1 ip/ op	1 ip/ op
DP[9] (mW)[5]	15.94/ 22.29	N/A	N/A	N/A	38/57	4.51[‡]	N/A	46[§]
Technology	Xilinx VirtexE	N/A	N/A	N/A	Xilinx xc6vcx240t	CMOS 0.18um	Xilinx VirtexE	Xilinx xc6vcx240t
MF[10] (MHz)	148.21/119.57	N/A	N/A	N/A	282.50/146.42	100	148	263.15

[1] Locally Reproduced (LR) [2] Measured [3] Intermediate Reconstruction Step (IRS)
[4] Hardware Complexity (HC) (# adders) [5] For single level decomposition
[6] For 8-bit Lena Image [7] Final Reconstruction Step (FRS) [8] Expansion Factor Method (EFM)
[9] Dynamic Power (DP) [10] Maximum Frequency (MF) [†] Taken from PSNR plots in [1, p. 1264]
[‡] At 50 MHz [17] [§] At 362.18 MHz [*] At 238.74 MHz

are saved when compared to the previous design [71]. In particular, for the 4-level decomposition, *the total savings in number of adders is 37.*

The *single* FRS is the only source of computational error. Noise injection from intermediate fixed-point errors is fully eliminated. Applications exist in subband coding of high dynamic range image sequences. Standard images have been analyzed with `Mandrill`, `Lena`, `Cameraman` examples shown. An FPGA based 4-level prototype is operational at 100 MHz. Place-and-route timing analysis furnished 263.15 MHz for the Daubechies 4-tap architecture. In addition, the proposed architecture was also realized in 45 nm ASIC technology at 523.56 MHz for 8-bit input data.

CHAPTER V

AI BASED LOW ADDER COUNT ARCHITECTURE FOR THE 1-D/ 2-D DAUBECHIES 6-TAP WAVELET FILTER BANKS

5.1 Introduction

In our recent work [70, 71] we addressed the issue of computational noise injection, introduced due to the fixed-point representation of Daub-4 and -6 filter coefficients using an algebraic integer (AI) based representation. An algebraic integer (AI) encoding of Daubechies filter coefficients. Algebraic integers are roots of monic polynomials [58]. Such representation could effectively control error propagation and precision levels.

In the present chapter, we propose a novel representation of the Daub-6 wavelet filters that results minimizing the filter bank arithmetic complexity of the filter banks and, therefore, leading to low adder count requirements, i.e., minimizing canonical signed-digit (CSD) terms. The proposed optimal architecture is consistent with the method advanced in [67], which employed integer linear programming for minimizing the number of adders required for the hardware realization of the 1-D Daub-6 filter banks.

The chapter unfolds as follows. Section 2.1 reviews the principles of sub-band

coding by means of Daub -6 filter. Section 5.2 provides Daub-6 scaling coefficients and prior art on its AI encoding. Section 4.3 furnishes the numerical optimization that leads to a fast algorithm having reduced number of 2-input adder circuits. The final reconstruction step (FRS) procedure for the proposed analyses are described in Section 3.3. Section 5.6.4 provides extensive digital design details for practical implementations using reconfigurable logic devices as well as CMOS technology. The paper is concluded in Section 5.7.

5.2 Daubechies 6-tap Filter Coefficients and AI basis

5.2.1 Mathematical Background

An algebraic integer is a real or complex number that is a root of a monic polynomial with integer coefficients [51,54,58]. Algebraic integers can be employed to define encoding mappings which can precisely represent particular irrational numbers by means of usual integers. Considering the roots of the monic polynomial $x^2 - 10$ and $x^4 - 10x^2 - 15 = 0$ we can extend the set of integers $\mathbb{Z}$ by including the algebraic integer $\zeta_1 = \sqrt{10}$ and $\zeta_2 = \sqrt{5 + 2\sqrt{10}}$. Doing so, a given quantity y can possibly be represented as $y = c + d \cdot \zeta_1 + e \cdot \zeta_2 + f \cdot \zeta_1\zeta_2$, where a, b, c, d, e, and f are usual integers.

5.2.2 Daub-6 Filter Impulse Response

The 2-D FIR filter bank based on the Daub-6 filter bank is of particular relevance [2, 45]. Let the low-pass filter associate to these filter banks be denoted

as $\mathbf{h}$. This particular filter possesses irrational quantities with an impulse response furnished by [2, 3, 6, 20, 43, 48]:

$$\mathbf{h} = \frac{1}{16\sqrt{2}} \begin{bmatrix} 1+\sqrt{10}+\sqrt{5+2\sqrt{10}} \\ 5+\sqrt{10}+3\sqrt{5+2\sqrt{10}} \\ 10-2\sqrt{10}+2\sqrt{5+2\sqrt{10}} \\ 10-2\sqrt{10}-2\sqrt{5+2\sqrt{10}} \\ 5+\sqrt{10}-3\sqrt{5+2\sqrt{10}} \\ 1+\sqrt{10}-\sqrt{5+2\sqrt{10}} \end{bmatrix}, \tag{5.1}$$

where the superscript $^\top$ denotes transposition. The presence of irrational quantities, in particular $\sqrt{10}$ and $\sqrt{5+2\sqrt{10}}$, poses difficulties when considering fixed-point representation. Conventional implementations always resort to truncation and/or rounding-off operations as a means to approximate $\sqrt{10}$ and $\sqrt{5+2\sqrt{10}}$ to a representable quantity in fixed-point arithmetic. However such procedure inevitably introduces computational errors.

In [71], it was demonstrated that (5.1) can be split in three integer coefficient filters. For such, consider the algebraic integer $\zeta_1 = \sqrt{10}$ and $\zeta_2 = \sqrt{5+2\sqrt{10}}$. Therefore, we obtain:

$$\mathbf{h} = \frac{1}{\beta}(\mathbf{h_1} + \zeta_1 \cdot \mathbf{h}_{\zeta_1} + \zeta_2 \cdot \mathbf{h}_{\zeta_2}), \tag{5.2}$$

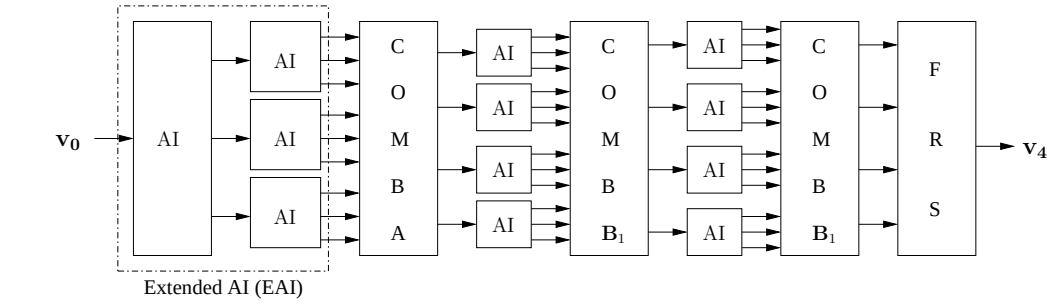
where $\beta = 16\sqrt{2}$, $\mathbf{h_1} = \begin{bmatrix} 1 & 5 & 10 & 10 & 5 & 1 \end{bmatrix}^\top$, $\mathbf{h}_{\zeta_1} = \begin{bmatrix} 1 & 1 & -2 & -2 & 1 & 1 \end{bmatrix}^\top$ and $\mathbf{h}_{\zeta_2} = \begin{bmatrix} 1 & 3 & 2 & 2 & 3 & 1 \end{bmatrix}^\top$. This scheme grants integer computation by not explicitly evaluating ζ_1 and ζ_2 until the final stage of computation. The set $\{1, \zeta_1, \zeta_2, \zeta_1\zeta_2\}$ forms a base for AI encoding. As a consequence, error propagation is deterred and final results can be appropriately obtained.

5.2.3 Multi-level Decomposition

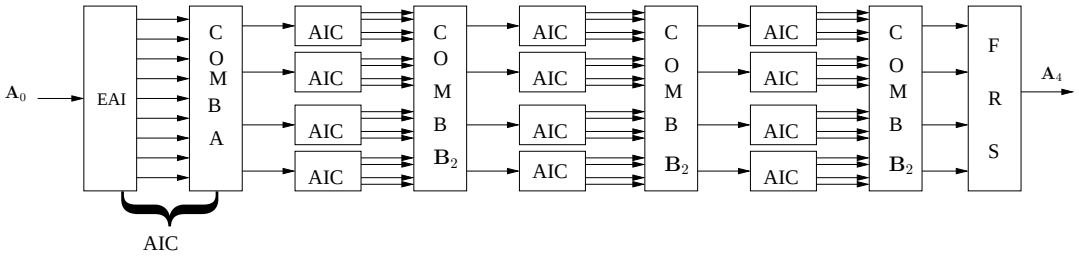
The proposed AI-based wavelet analyses based on Daub-6 filter bank is computed entirely over extended integer fields. Therefore, for a given input data, a single level decomposition of the discussed AI based wavelet analysis results in an output data consisting of four parts, where each part is associated to a basis element. These four parts are required to be combined from a level to the next up to the final reconstruction step. Two types of combinational blocks are required, being their inner structures detailed later in the current work.

In Fig. 5.1(a) and (b), we show the overall scheme for a four-level decomposition for both the 1-D and the 2-D cases, respectively. We refer to the combination steps as Combinational Block A, B_1, and B_2.

The following notation is employed. For the 1-D decomposition (Fig. 5.1(a)), $\mathbf{v}_0$ is an N-point input vector and $\mathbf{v}_4^{(1)}$, $\mathbf{v}_4^{(\zeta_1)}$, $\mathbf{v}_4^{(\zeta_2)}$ and $\mathbf{v}_4^{(\zeta_1\zeta_2)}$ are resulting AI encoded data according to the basis elements $\{1, \zeta_1, \zeta_2, \zeta_1\zeta_2\}$, respectively. These vectors are combined in the final reconstruct step and converted back to standard fixed-point representation. This is required in order to interface the resulting approximation sub-



(a) 1-D



(b) 2-D

Figure 5.1: Multi-level wavelet decomposition with Daubechies 6-tap filter.

images with conventional real time systems. The result is a $N/16$-point vector $\mathbf{v}_4$. Similarly, for the 2-D case (Fig. 5.1(b)), the input image is $\mathbf{A}_0$ and the resulting AI encoded data are ${\mathbf{A}_4}^{(1)}$, ${\mathbf{A}_4}^{(\zeta_1)}$, ${\mathbf{A}_4}^{(\zeta_2)}$, and ${\mathbf{A}_4}^{(\zeta_1\zeta_2)}$, respectively.

5.3 Optimized AI Encoding

5.3.1 Number of Additions

The AI encoding discussed in previous section implies integer coefficient filters $\mathbf{h_1}$, $\mathbf{h}_{\zeta_1}$, and $\mathbf{h}_{\zeta_2}$. In $\mathbf{h_1}$, the required multiplications by 5 and 10 imply extra additions. This is because multiplying a number x by constant 5 or 10 require one addition and one bit-shift operation ($5 \cdot x = x \ll 2 + x$) or one addition and two bit-shift operations ($10 \cdot x = x \ll 3 + x \ll 1$), respectively.

In general terms, depending on the binary representation of the filter coefficients, they may contribute with extra additions. We adopt the canonical-signed-digit representation for binary encoding the integer coefficients to avoid multiplicative complexity by use of adders and bit shifters. Additionally, let $\mathrm{S_{CSD}}(n)$ return the number of additions/subtractions of powers of two required to represent a given integer n. For example, we have: $\mathrm{S_{CSD}}(2) = 0$, $\mathrm{S_{CSD}}(3) = 1$, $\mathrm{S_{CSD}}(11) = 2$ $(11 = 16 - 4 - 1)$. Clearly, if all elements of a given filter are dyadic, then no extra addition is demanded. In this scenario, a minimal additive complexity is attained.

5.3.2 Filter Parametrization and Optimization

Our goal is to rearrange (5.2) in such a way that the coefficients of the resulting filters can be represented in CSD form with minimum number of additions.

For such let us introduce two integer parameter m and n in (5.2) as follows:

$$\beta\mathbf{h} = \begin{bmatrix} 1 \\ 5 \\ 10 \\ 10 \\ 5 \\ 1 \end{bmatrix} - m \begin{bmatrix} 1 \\ 1 \\ -2 \\ -2 \\ 1 \\ 1 \end{bmatrix} + (\zeta_1 + m) \begin{bmatrix} 1 \\ 1 \\ -2 \\ -2 \\ 1 \\ 1 \end{bmatrix} - n \begin{bmatrix} 1 \\ 3 \\ 2 \\ -2 \\ -3 \\ -1 \end{bmatrix} + (\zeta_2 + n) \begin{bmatrix} 1 \\ 3 \\ 2 \\ -2 \\ -3 \\ -1 \end{bmatrix}$$

$$= \begin{bmatrix} 1 - m - n \\ 5 - m - 3n \\ 10 + 2m - 2n \\ 10 + 2m + 2n \\ 5 - m + 3n \\ 1 - m + n \end{bmatrix} + (\zeta_1 + m) \begin{bmatrix} 1 \\ 1 \\ -2 \\ -2 \\ 1 \\ 1 \end{bmatrix} + (\zeta_2 + n) \begin{bmatrix} 1 \\ 3 \\ 2 \\ -2 \\ -3 \\ -1 \end{bmatrix}.$$

Above expression implies three integer filters:

$$\mathbf{h_1}' = \begin{bmatrix} 1 - m - n \\ 5 - m - 3n \\ 10 + 2m - 2n \\ 10 + 2m + 2n \\ 5 - m + 3n \\ 1 - m + n \end{bmatrix}, \mathbf{h_{\zeta_1}}' = \begin{bmatrix} 1 \\ 1 \\ -2 \\ -2 \\ 1 \\ 1 \end{bmatrix}, \mathbf{h_{\zeta_2}}' = \begin{bmatrix} 1 \\ 3 \\ 2 \\ -2 \\ -3 \\ -1 \end{bmatrix}.$$

Notice that $\mathbf{h_{\zeta_1}}' = \mathbf{h_{\zeta_1}}$ and $\mathbf{h_{\zeta_2}}' = \mathbf{h_{\zeta_2}}$. Moreover, the required AI elements must be replaced for $\zeta'_1 = \zeta_1 + m$ and $\zeta'_2 = \zeta_2 + n$. Notice that, for $m = n = 0$, we obtain (5.2).

Since the elements of $\mathbf{h_{\zeta_1}}'$ and $\mathbf{h_{\zeta_2}}'$ pose very low arithmetic complexity, we focus our attention on $\mathbf{h_1}'$ aiming at finding a suitable value of m and n such that the resulting coefficients are efficiently represented with minimum CSD additions.

Let $h'_1[k]$, $k = 0, 1, \ldots, 5$, be the coefficients of $\mathbf{h_1}'$. In view of the above discussion, the optimal value of m and n, denoted m^* and n^*, are the solutions of the following minimization problem:

$$\begin{aligned}(m^*,n^*) &= \arg\min_{m,n\in\mathbb{Z}} \sum_{k=0}^{5} \mathrm{S_{CSD}}(h'_1[k]) \\ &= \arg\min_{m,n\in\mathbb{Z}} \Big\{ \mathrm{S_{CSD}}(1-m-n) + \mathrm{S_{CSD}}(5-m-3n) \\ &\quad + \mathrm{S_{CSD}}(10+2m-2n) + \mathrm{S_{CSD}}(10-2m+2n) \\ &\quad + \mathrm{S_{CSD}}(5-m+3n) + \mathrm{S_{CSD}}(1-m+n) \Big\}.\end{aligned} \tag{5.3}$$

Usual optimization tools are not applicable due to the difficulty in analytically manipulate (5.3). Therefore, we resort to numerical search methods as a means to solve (5.3). Such computational search requires that we limit the search space. Let $\{m, n \in \mathbb{Z} : |m| \leq 1024, |n| \leq 1024\}$ be the considered search space. Under these conditions, we could obtain four distinct solutions: $\{(-3, 0), (-5, -2), (-5, 2), (3, 0)\}$. Each optimum pair leads to different filter structures; and, consequently, to different architectures. We refer to each of these possible implementations as Method 1,

Table 5.1: Filter Parametrization and Optimization

Method	(m^*, n^*)	$\mathbf{h_1}'$	CSD additions
1	$(-3, 0)$	$\begin{bmatrix}4 & 8 & 4 & 4 & 8 & 4\end{bmatrix}^\top$	0
2	$(-5, -2)$	$\begin{bmatrix}8 & 16 & 4 & -4 & 4 & 4\end{bmatrix}^\top$	0
3	$(-5, 2)$	$\begin{bmatrix}4 & 4 & -4 & 4 & 16 & 8\end{bmatrix}^\top$	0
4	$(3, 0)$	$\begin{bmatrix}-2 & 2 & 16 & 16 & 2 & -2\end{bmatrix}^\top$	0

Method 2, Method 3, and Method 4, respectively.

The resulting filters $\mathbf{h_1}'$ associated to each optimum solution possess zero multiplicative complexity. This can be directly verified since all coefficients of $\mathbf{h_1}'$ are powers of two. Thus the CSD representation of theses coefficients require no extra additions. Table 4.1 summarizes the obtained results and employed terminology.

5.4 Final Reconstruction Step

Decoding operations for Daub-6 1-D and 2-D consist of explicitly performing the following computations, respectively [71]:

$$\beta^n \cdot \mathbf{v}_n = \left(\mathbf{v}_n{}^{(1)} + \zeta_1' \cdot \mathbf{v}_n{}^{(\zeta_1')} + \zeta_2' \cdot \mathbf{v_n}{}^{(\zeta_2')} + \zeta_1'\zeta_2' \cdot \mathbf{v_n}{}^{(\zeta_1'\zeta_2')}\right), \tag{5.4}$$

$$\beta^{2n} \cdot \mathbf{A}_n = \left(\mathbf{A}_n{}^{(1)} + \zeta_1' \cdot \mathbf{A}_n{}^{(\zeta_1')} + \zeta_2' \cdot \mathbf{A_n}{}^{(\zeta_2')} + \zeta_1'\zeta_2' \cdot \mathbf{A_n}{}^{(\zeta_1'\zeta_2')}\right). \tag{5.5}$$

Fortunately, the factor β^{2n} is always a power of two, which can be conveniently realized with bit-shift operations. The above decoding operation is realized at

the FRS blocks depicted in Fig. 5.1(a) and 5.1(b), for the 1-D and 2-D structures, respectively. Therefore, the only possible source of errors in the proposed architectures are the multiplication by AI basis elements.

5.4.1 CSD Approximation

The FRS can be directly implemented by approximating the required irrationals in (5.5) into rationals. A possibility is employing CSD representation. Table 3.4 shows encoding for ζ_1' and ζ_2', and $\zeta_1'\zeta_2'$ for several word lengths as well as the associate relative errors. CSD encoding requires only bit-shifters and adders/subtracters.

5.5 1-D Designs and Results

In this section, we detail and analyse the proposed 1-D Daub-6 designs. The particular choice of elements for the AI Daub-6 filter have a central role in generating optimized architecture. Some of the fundamental blocks employed in the 1-D designs are implicitly present in the 2-D design shown in [71]. Being the blocks in [71] structurally different—although functionally equivalent— we could refer to them for comparison with the new introduced designs.

5.5.1 Proposed Structures and Additive Complexity

Combinational Blocks

Table 5.2: CSD Encoding for ζ_1', ζ_2' and $\zeta_1'\zeta_2'$ for Methods 1 through 4

		Method 1		Method 2		Method 3		Method 4	
AI	**W** †	**CSD**	**ARE** ‡	**CSD**	**ARE** ‡	**CSD**	**ARE** ‡	**CSD**	**ARE** ‡
ζ_1'	8	0.00101	0.0371	$\bar{1}0.010\bar{1}$	0.0137	$\bar{1}0.010\bar{1}$	0.0137	$\bar{1}0\bar{1}.001$	0.0060
	10	0.00101001	0.0131	$\bar{1}0.010\bar{1}0\bar{1}$	0.0052	$\bar{1}0.010\bar{1}0\bar{1}$	0.0052	$10\bar{1}.001011$	0.0016
	12	0.0010100101	0.0071	$\bar{1}0.010\bar{1}0\bar{1}\bar{1}00\bar{1}$	0.00043	$\bar{1}0.010\bar{1}0\bar{1}\bar{1}00\bar{1}$	0.00043	$10\bar{1}.00101011\bar{1}$	0.00034
	14	0.0010100101001	0.0063	$\bar{1}0.010\bar{1}0\bar{1}\bar{1}00\bar{1}0\bar{1}$	0.00030	$\bar{1}0.010\bar{1}0\bar{1}\bar{1}00\bar{1}0\bar{1}$	0.00030	$10\bar{1}.001010101\bar{1}$	0.000027
	16	0.00101001010001	0.00066	$\bar{1}0.010\bar{1}0\bar{1}\bar{1}00\bar{1}0\bar{1}$	0.00030	$\bar{1}0.010\bar{1}0\bar{1}\bar{1}00\bar{1}0\bar{1}$	0.00030	$10\bar{1}.00101010\bar{1}001$	0.000012

† Word Length (W) in bits ‡ Absolute Relative Error (ARE)

Table 5.3: CSD Encoding for ζ_1', ζ_2' and $\zeta_1'\zeta_2'$ for Methods 1 through 4 (Continued)

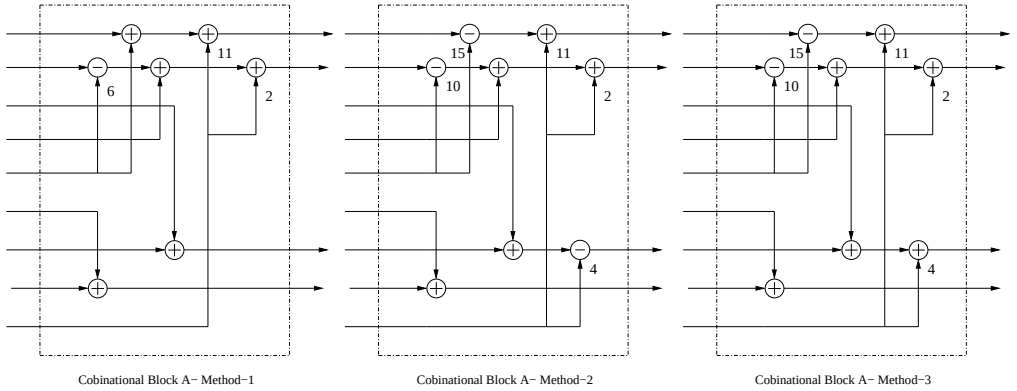
		Method 1		Method 2		Method 3		Method 4	
AI	**W** †	**CSD**	**ARE** ‡	**CSD**	**ARE** ‡	**CSD**	**ARE** ‡	**CSD**	**ARE** ‡
ζ_2'	8	$10.\bar{1}0\bar{1}$	0.0029	$0.1010\bar{1}$	0.0072	$11.10\bar{1}$	0.0072	$10.\bar{1}0\bar{1}$	0.0029
	10	$10.\bar{1}0\bar{1}000\bar{1}$	0.00059	$0.10\bar{1}00\bar{1}\bar{1}$	0.0015	$11.10\bar{1}00\bar{1}\bar{1}$	0.0015	$10.\bar{1}0\bar{1}000\bar{1}$	0.00059
	12	$10.\bar{1}0\bar{1}000\bar{1}00\bar{1}$	0.00030	$0.10\bar{1}00\bar{1}\bar{1}0\bar{1}$	0.000026	$11.10\bar{1}00\bar{1}\bar{1}0\bar{1}$	0.000026	$10.\bar{1}0\bar{1}000\bar{1}00\bar{1}$	0.00030
	14	$10.\bar{1}0\bar{1}000\bar{1}00\bar{1}0\bar{1}$	0.00011	$0.10\bar{1}00\bar{1}\bar{1}0\bar{1}$	0.000026	$11.10\bar{1}00\bar{1}\bar{1}0\bar{1}$	0.000026	$10.\bar{1}0\bar{1}000\bar{1}00\bar{1}0\bar{1}$	0.00011
	16	$10.\bar{1}0\bar{1}000\bar{1}00\bar{1}0\bar{1}$	0.00011	$0.10\bar{1}00\bar{1}\bar{1}0\bar{1}$	0.000026	$11.10\bar{1}00\bar{1}\bar{1}0\bar{1}$	0.000026	$10.\bar{1}0\bar{1}000\bar{1}00\bar{1}0\bar{1}$	0.00011

† Word Length (W) in bits ‡ Absolute Relative Error (ARE)

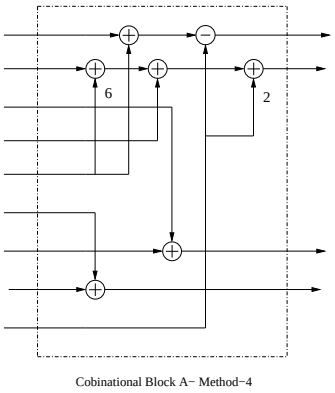
Table 5.4: CSD Encoding for ζ_1', ζ_2' and $\zeta_1'\zeta_2'$ for Methods 1 through 4 (Continued)

		Method 1		Method 2		Method 3		Method 4	
AI	**W** †	**CSD**	**ARE** ‡	**CSD**	**ARE** ‡	**CSD**	**ARE** ‡	**CSD**	**ARE** ‡
	8	0.1000101	0.0129	$\bar{1}0.\bar{1}$	0.0035	$\bar{1}0\bar{1}.001$	0.0015	$1000.0\bar{1}$	0.00061
	10	0.100011001	0.0050	$\bar{1}0.\bar{1}00000\bar{1}$	0.00041	$\bar{1}0\bar{1}.00100\bar{1}$	0.000037	$1000.0\bar{1}$	0.00061
$\zeta_1'\zeta_2'$	12	0.100011001	0.0032	$\bar{1}0.\bar{1}00000\bar{1}00\bar{1}$	0.000026	$\bar{1}0\bar{1}.00100\bar{1}000001$	0.000012	$1000.0\bar{1}000\bar{1}1$	0.00023
	14	0.10001100000101	0.0019	$\bar{1}0.\bar{1}00000\bar{1}00\bar{1}000\bar{1}$	0.0000016	$\bar{1}0\bar{1}.00100\bar{1}00000101$	0.0000023	$1000.0\bar{1}000\bar{1}001$	0.000046
	16	0.10001100000101	0.0019	$\bar{1}0.\bar{1}00000\bar{1}00\bar{1}000\bar{1}$	0.0000016	$\bar{1}0\bar{1}.00100\bar{1}00000101$	0.0000023	$1000.0\bar{1}000\bar{1}0011$	0.0000010

† Word Length (W) in bits ‡ Absolute Relative Error (ARE)



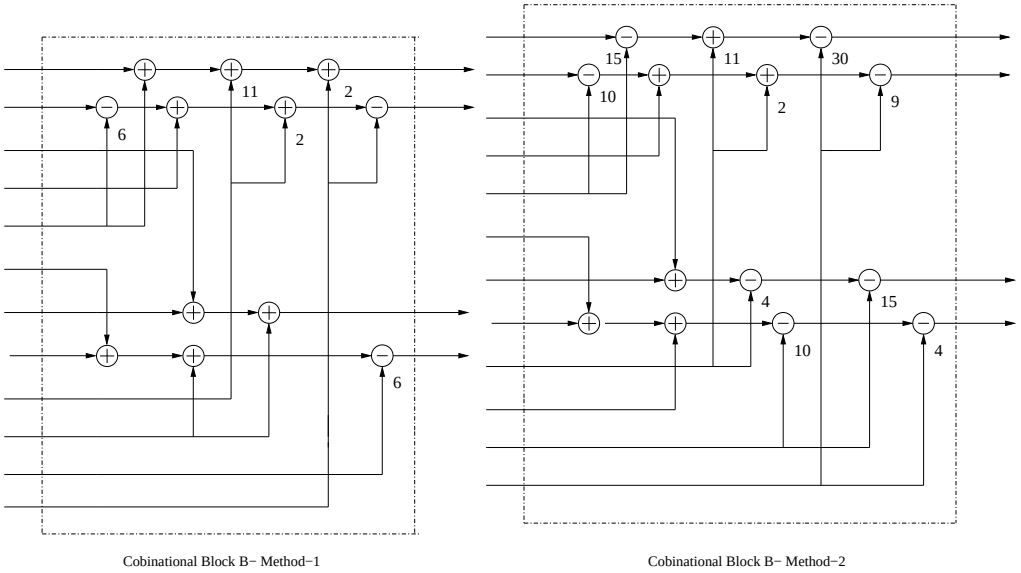
(a) Method 1 (b) Method 2 (c) Method 3



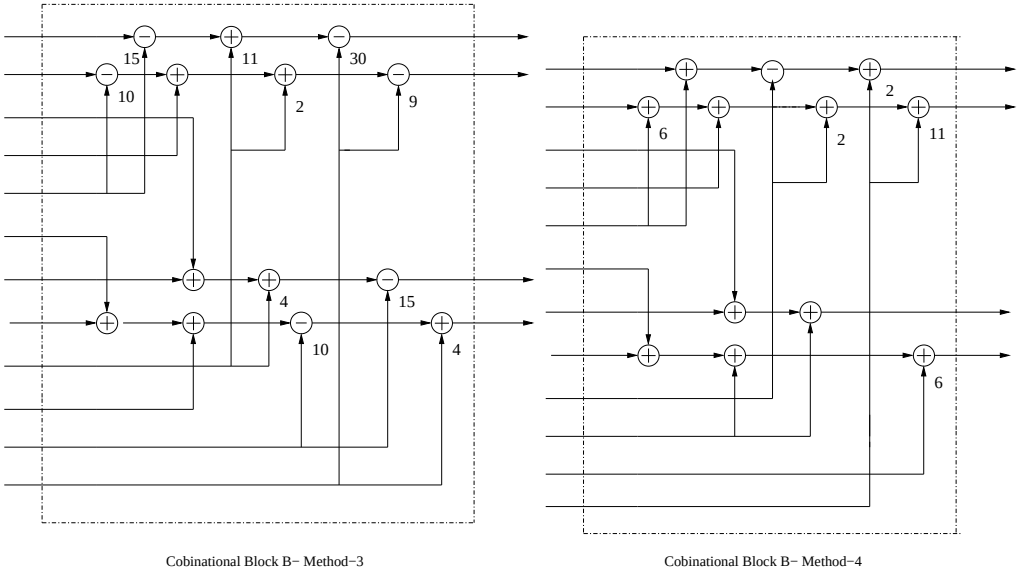
(d) Method 4

Figure 5.2: Combinational Block A for 1-D/ 2-D Daub-6 filter.

Figs. 5.2 and 5.3 show the inner details of Combinational Blocks A and B_1 (cf. Fig. 5.1), respectively, when Methods 1, 2, 3, and 4 are considered. Table 5.7 shows the CSD realizations of the constants required for above two blocks. In [71], Combinational Block A was given a design that requires eight adders, which has lower complexity compared to the present design. However, we show that the overall complexity of the proposed schemes is decreased on account of savings in other parts of the architectures.



(a) Method 1 (b) Method 2



(c) Method 3 (d) Method 4

Figure 5.3: Combinational Block B_1 for 1-D Daub-6 filter.

Core Structure

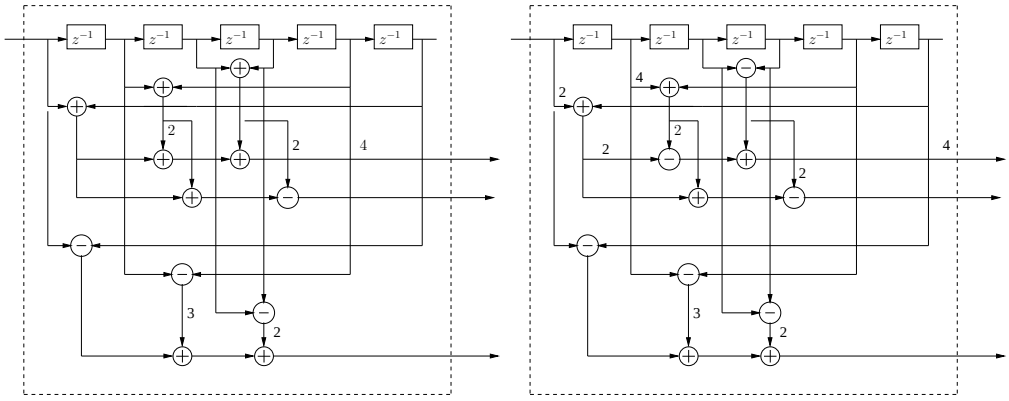
Fig. 5.4 displays the proposed structures for filter **h** for each of the considered methods. These filters require only 13 adders for a single Daub-6 filter realization. In comparison, the previous design detailed in [71] requires fifteen additions. This represents a decrease of approximately 13% in additive complexity. Each AI block contains four instantiations of **h**, which implies a saving of eight additions per AI block.

Final Reconstruction Step

The FRS based on the direct CSD approach was considered according to the data given Table 3.4. The number of required adders at the FRS for each design is shown in Table 5.4. Notice that Method 3 is less efficient requiring 10 adders, whereas all remaining approaches require 8 adders.

5.5.2 Overall Adder Count

For an n-level decomposition, $n > 2$, Fig. 5.1 (a) suggests the following number of required blocks: one Combinational Block A, $4(n-1)$ AI blocks, $n-2$ Combinational Blocks B_1, and one FRS block. Table 5.4 shows the adder count complexity for all 1-D designs. Thus, for instance, the 4-level decomposition shown



(a) Method 1 (b) Method 2

(c) Method 3 (d) Method 4

Figure 5.4: Proposed optimal realization of the Daub-6 filter.

Table 5.5: Number of Adders Required for all Proposed 1-D Designs using Proposed CSD in Table 3.4

Method	AI	Block A	Block B_1	FRS
1	13	10	16	8
2	13	12	22	8
3	13	12	22	10
4	13	10	16	8

in Fig. 5.1 (a), has the following total adder count: 206, 220, 222, and 206, for Methods 1, 2, 3, and 4, respectively.

5.5.3 Resource Consumption and Figures of Merit

Xilinx Virtex-6 Implementation

Proposed 1-D multi-level Daub-6 filters based on Methods 1, 2, 3, and 4 were implemented on the Xilinx Virtex-6 vcx240t-1ff1156 FPGA device. Table 5.5 lists the resource consumption for the number of slice registers and look-up table (LUT) count for the 1-D designs. Critical path delays (CPD) and the maximum operating frequency (MF) are also reported. We considered word length of 8 bits. As figures of merit, we also adopted the area-time (AT) and area-time^2 (AT^2) products. Metric AT provides insight into circuit performance considering chip area, while metric AT^2 is used for circuits where speed requirements are of higher importance. Table 5.6 shows the estimated power consumption obtained from Xilinx-ISE for 1-D designs. Clock net, quiescent and dynamic powers are reported for the considered input word length. The total power reported in Table 5.6 is the sum of quiescent and dynamic powers.

Table 5.6: Hardware resource consumption with Xilinx Virtex-6 vcx240t-1ff1156 for 1-D Daub-6 implementation

Resource	Method 1	Method 2	Method 3	Method 4
Registers ($\times 10^3$)	2.89	2.78	3.10	3.20
LUTs ($\times 10^3$)	5.47	7.18	8.11	8.70
CPD (ns)	2.86	3.13	3.27	3.41
MF ($\times$100 MHz)	3.44	3.29	3.11	3.12
AT ($\times 10^{-5}$)	1.56	2.60	2.61	3.41
AT^2 ($\times 10^{-14}$)	4.46	8.13	8.53	10.09

Table 5.7: Xilinx ISE XPower estimation results for 1-D Daub-6 filter

Power (Watt)	Method 1	Method 2	Method 3	Method 4
Clock net	0.034	0.041	0.048	0.037
Quiescent	1.538	1.562	1.710	1.662
Dynamic	0.204	0.216	0.244	0.235
Total	1.742	1.778	1.954	1.898

Discussion

As shown in Tables 5.5 and 5.6, resource and power consumptions; performance measures AT and AT^2; and maximum frequency reveal that proposed Methods 1 and 2 could outperform the other two methods in all aforementioned performance aspects. Table 5.6 At the CSD based FRS, both Methods 1 and 2 possess the same additive complexity, requiring eight adders as listed in Table 3.4.

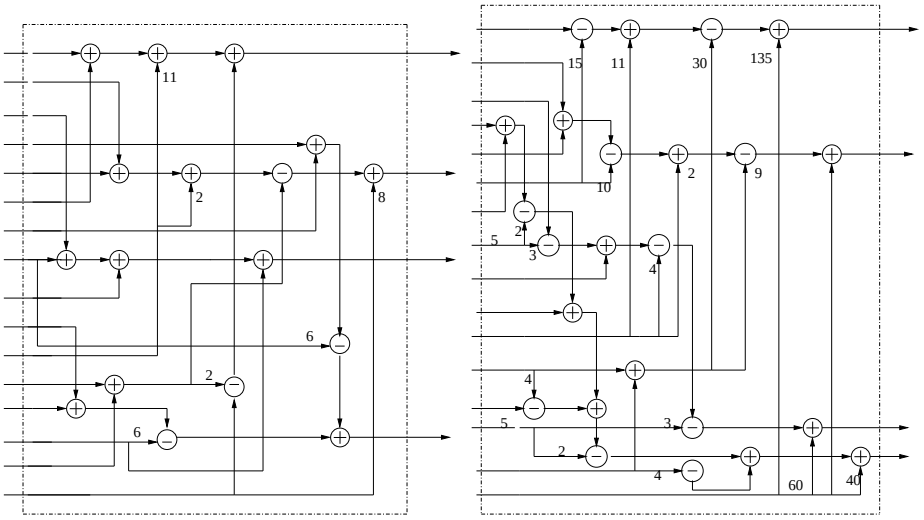
As a result of the above discussion, we could separate Methods 1 and 2 as the best ones; and we regard Methods 3 and 4 as inferior, not considering them as design options hereafter.

5.6 2-D Designs and Results

In this section, we advance 2-D designs for the Daub-6 filters. In terms of architecture complexity, the resource consumption of 2-D designs is expected to be much higher than that of 1-D designs. Therefore the selection of filters for the 2-D design is of essential relevance. As consequence, based on the results of the 1-D designs, we consider only Method 1 and 2 as potentially efficient frameworks for the 2-D designs.

5.6.1 Proposed Structures and Adder Count

Combinational Blocks



(a) Method 1 (b) Method 2

Figure 5.5: Combinational block B_2 for 2-D Daub-6 filter.

The 2-D design requires the Combinational Block A, which was already detailed in the 1-D architecture. Combinational Block B_2 is also necessary for both Methods 1 and 2. Table 5.7 displays the CSD realizations for the numeric coefficients employed in this block.

Fig. 5.1(b) shows that we need $n-1$ realizations of Combinational Block B_2

Table 5.8: CSD realization of the constants required in Combinational blocks A, B_1, and B_2

Coefficient	CSD	Coefficient	CSD
3	2^1+1	15	2^4-1
6	2^2+2^1	30	2^5-2^1
9	2^3+1	40	2^5+2^3
10	2^3+2^1	60	2^6-2^2
11	2^3+2^1+1	135	2^7+2^3

for $n > 1$ levels of decomposition. Combinational Block B_2 requires 21 or 37 when implemented by Method 1 or 2, respectively. The previously proposed design in [71] requires 24 additions. Thus we save three adders or demand 13 extra adders, for Method 1 or 2, respectively, compared to [71].

AIC Block

We refer to the joint extended AI (EAI) and Combinational Block A structures as the AIC Block (cf. Fig. 5.1(b)), which is reused several times in the overall architecture. Fig. 5.1 allows us to conclude that an n-level decomposition, $n > 1$, requires $4 \cdot n - 3$ AIC blocks. Each AIC block contains four AI blocks. Then a total of $16 \cdot n - 12$ AI blocks are needed. As previously discussed, the proposed AI block requires 13 adders; two less adders when compared to [71]. Therefore in terms of AI blocks, we have a saving of $2 \times (16 \cdot n - 12) = 32 \cdot n - 24$ additions.

Moreover, each AIC block require one Combinational Block A, which requires 10 or 12 additions, depending on Method 1 or 2, respectively. Thus, a total of $10 \cdot (4n - 3) = 40 \cdot n - 30$ or $12 \cdot (4 \cdot n - 3) = 48 \cdot n - 36$ adders are required due to instantiations of Combinational Block A , for Method 1 and 2, respectively.

5.6.2 Complexity Assessment

Considering (i) the required number of blocks for an n-level Daubechies-4 architecture and (ii) the adder count for each block as shown in Table 5.8, we could derive the expressions for the overall adder count of the proposed architectures.

Table 5.9: Number of Adders Required and Error Introduced

Method	Block					
	AI	A	AIC	B_2	FRS	TAE‡
Madishetty *et al.* [71]	15	8	68	24	13	0.0733
Method 1	13	10	62	21	8	0.0228
Method 2	13	12	64	37	8	0.0439

‡ Total Absolute Error (TAE) incurred at 8-bit FRS

Methods 1 and 2 require $269 \cdot n - 199$ and $293 \cdot n - 221$ adders, respectively. The total absolute error (TAE) is also listed in Table 5.8. This quantity provides the error incurred in the discussed methods as well as in the earlier scheme described in [71].

For $n = 4$ (Fig. 5.1(b)), we have that the total adder count is 877 and 951, for Method 1 and 2, respectively. The method described in Madishetty *et al.* [71] requires 967 adders. Thus, proposed Method 1 and 2 offer an improvement of 9.3% and 1.6%, respectively.

5.6.3 Overall Savings

Above adder count discussion is summarized in Table 5.8. We obtained the following net savings in adder circuits at n-level decomposition: $27 \cdot n - 16$ and $3 \cdot n + 6$, for Method 1 and 2, respectively. For $n = 4$, the number of adder savings are 92 and 18, respectively.

Above results on complexity assessment and adder savings show that Method 1 is significantly superior in comparison with Method 2. Therefore, we

elect Method 1 as the most adequate approach for the discussed AI-based wavelet scheme. Hereafter, only Method 1 is considered in our analyses and implementations.

5.6.4 FPGA and ASIC Implementation

The architecture for 2-D Daubechies 6-tap filter bank based on Method 1 was physically implemented and hardware co-simulated on Xilinx Virtex xc6vcx240t-1ff1156 FPGA device using the ML605 evaluation board. Additionally, Method 1 was also synthesized with CMOS 45 nm ASIC technology up to synthesis level at supply voltage $V_{\mathrm{DD}} = 1.1$ V. Details of both designs are described in the following subsections.

Xilinx Virtex-6 Implementation

A rapid prototype is realized using a Xilinx Virtex-6 FPGA device, hosted on a ML605 rapid prototyping system. The example designs are targeted to the Xilinx xc6vcx240t-1ff1156 FPGA chip with connectivity to a host PC via a JTAG interface. We considered the following word length sizes: 8, 10, 12, 14, and 16 bits. Table 5.9 lists resource consumption for slice and LUT count for the 2-D designs.

Test vectors consisting of sequential samples of data from a standard *Lena* image [59] were systematically routed, block-by block, to the FPGA using the hardware co-simulation. The FPGA realization used the JTAG connection to obtain input data, which is filtered using Daub-6 wavelet filters on chip, and results are

Table 5.10: Hardware Resource Consumption with Xilinx Virtex-6 vcx240t-1ff1156 for 2-D Daub-6 Implementation

Resource	Word length (bits)				
	8	10	12	14	16
Slices ($\times 10^3$)	14.16	16.22	18.39	20.03	22.74
LUTs ($\times 10^3$)	38.53	43.81	49.16	54.22	57.16
CPD (ns)	6.22	6.63	7.15	7.59	8.17
AT ($\times 10^{-5}$)	10.67	13.10	16.18	19.12	23.41
AT^2 ($\times 10^{-13}$)	6.64	8.68	11.56	14.51	19.68
MF($\times$100 MHz)	1.68	1.59	1.52	1.44	1.38

Table 5.11: Xilinx ISE XPower Estimation Results for 2-D Daub-6 Filter

Power (Watt)	Word-length (bits)				
	8	10	12	14	16
Clock net	0.091	0.104	0.112	0.128	0.142
Quiescent	4.271	4.271	4.272	4.272	4.273
Dynamic	0.304	0.319	0.337	0.354	0.371
Total	4.575	4.59	4.609	4.626	4.644

routed back to the PC. Fig. 5.6 shows measured results following on-chip physical implementation as obtain from the FPGA implementation.

Table 5.10 shows the estimated power consumption obtained from Xilinx-ISE for Method-1 2-D design. Clock net, quiescent, and dynamic powers are reported for the considered input word lengths. The total power reported in Table 5.10 is the sum of quiescent and dynamic powers.

Table 5.12: Hardware Resource Consumption for CMOS 45 nm ASIC Synthesis (Supply Voltage $V_{\mathrm{DD}} = 1.1$ V)

Resource	Word length (bits)				
	8	10	12	14	16
AGC ($\times 10^5$)	8.60	9.33	10.23	10.98	11.74
Area (mm^2)	6.04	6.62	7.34	8.01	8.82
CPD (ns)	3.43	3.78	4.11	4.58	4.92
Area-time	2.07	2.50	3.02	3.67	4.34
Area-time^2	7.10	9.45	12.41	16.81	21.35
F_{max} (MHz)	306.44	283.26	261.17	239.84	212.52

CMOS 45 nm ASIC Synthesis

Using the free PDK from [72], the digital design for 8-bit images based on AI-based Method 1 was mapped to the 45 nm CMOS standard cell technology up to synthesis level using Cadence Encounter. Power consumptions and timing was estimated and reported. Table 5.11 shows the resource consumption for the ASIC 45 nm synthesis of the 2-D architecture, generated by Encounter® RTL compiler. We included the ASIC gate count (AGC), chip area, as well as the already discussed figures of merits. The ASIC synthesis yielded a maximum frequency of 306 MHz when 8-bit input data is considered. Table 5.12 displays the estimated power consumption. We refer to leakage power, dynamic power, and total power as L_p, D_p, and T_p, respectively. Table 5.13 provides a quantitative and comprehensive comparative study of published AI based DWT architectures.

5.6.5 Image Quality Assessment

Table 5.13: Power Consumption for CMOS 45 nm ASIC Synthesis (Supply Voltage $V_{\mathrm{DD}} = 1.1$ V)

	Word length (bits)				
Power	8	10	12	14	16
L_p (mW)	32.86	34.11	36.50	38.42	40.74
D_p (W)	4.12	4.63	5.18	5.62	6.02
T_p (W)	4.15	4.66	5.21	5.66	6.06

For comparison purposes, we devised a version of the proposed system that operates over fixed-point arithmetic instead of AI based arithmetic. For such, we employed 8-bit words with 6 fractional bits. In this case, the required filter banks were implemented by quantizing the exact filter coefficients into the fixed-point representation.

As image quality measures, we adopted the signal to noise ratio (SNR) and peak SNR (PSNR). Table 5.14 provides the obtained measures for both the fixed-point implementation as well as the proposed AI based design. We noticed an approximate 5% increase in SNR/PSNR figures in favour of the proposed design when compared to [71].

5.7 Conclusion

We proposed an optimized multi-rate Daub-6 filter bank architecture which is AI-based, and multi-encoded. Additionally, the introduced design is capable of furnishing arbitrarily high numerical accuracy using error-free integer arithmetic.

Table 5.14: Comparison of Proposed Architectures with Existing AI Based DWT Architectures in Literature

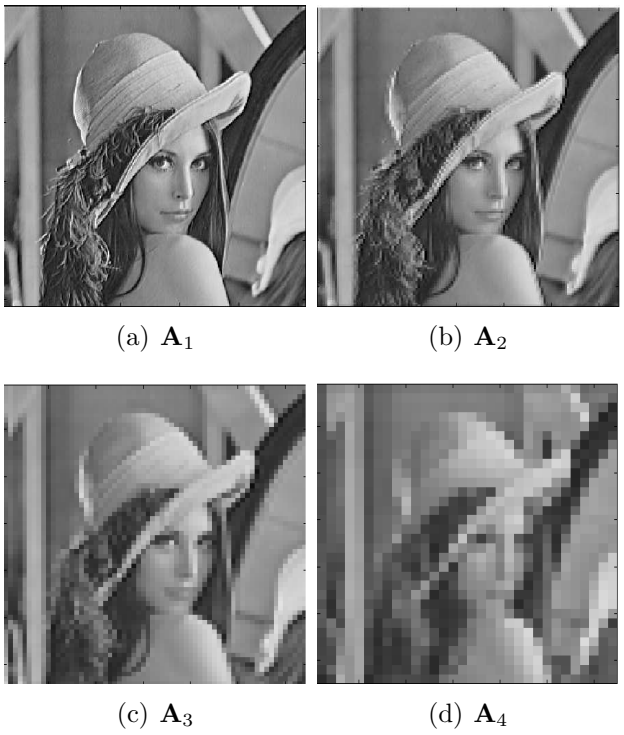
	Wahid *et al.* [1]	Wahid *et al.* [6]	Wahid *et al.* [19]	Wahid *et al.* [3]	Wahid *et al.* [17]	Gustafsson *et al.* [67]	Wahid *et al. [20]*	Madishetty *et al.* [71]	Proposed Method-1
LR[1]	No	No	No	No	No	No	No	Yes [2]	Yes [2]
Wavelet	Daub-4/6	Daub-4/6	Daub-6	Daub-4/6	Daub-4/6	Daub-6	Daub-6	Daub-4/6	Daub-6
Architecture	1-D / 2-D	1-D	1-D	1-D	1-D	1-D	1-D	2-D	1-D / 2-D
IRS[3]	Yes	Yes	Yes	Yes	Yes	N/A	Yes	No	No
HC[4]	10/ 18	10/ 25	25	N/A	9/ 18	18	18	10/21	17 */ 18 **
Registers[5]	200/ 494	N/A	N/A	N/A	115/196	N/A	494	258/765	177 $^{\diamond}$/ 593$^{\diamond\diamond}$
Logic cells[5]	248/680	N/A	N/A	N/A	106/254	N/A	340	426/ 1040	248 $^{\diamond}$/ 867 $^{\diamond\diamond}$

[1] Locally Reproduced (LR) [2] Measured [3] Intermediate Reconstruction Step (IRS)
[4] Hardware Complexity (HC) (# adders) for an AI Daub-6 filter [5] For single level decomposition
† Taken from PSNR plots in [1, p. 1264] ‡ At 50 MHz [17] †† At 374.25 MHz ‡‡ At 346.58 MHz
$^{\diamond}$ For 1-D Design $^{\diamond\diamond}$ For 2-D Design

Table 5.15: Comparison of Proposed Architectures with Existing AI Based DWT Architectures in Literature (Continued)

PSNR (dB)[6]	38 †(Daub-4) 39 †(Daub-6)	50.49 (Daub-4) 49.87 (Daub-6)	N/A	22.26	N/A	N/A	43.20	66.82 (Daub-4) 68.12 (Daub-6)	71.54
FRS[7]	CSD	CSD	Booth	Booth	CSD	CSD	CSD	CSD/EFM [8]	CSD
Throughput	1 ip/ op	1 ip/ op	N/A	N/A	2 ip/op	1 ip/ op	1 ip/ op	1 ip/ op	1 ip/ op
DP[9](mW)[5]	15.94/ 22.29	N/A	N/A	N/A	4.51‡	N/A	N/A	38§/ 57*	61 ††
Technology	Xilinx VirtexE	N/A	N/A	N/A	CMOS 0.18um	N/A	Xilinx Virtex E	Xilinx Virtex-6 vcx240t	Xilinx Virtex-6/ CMOS 45 nm
MF[10] (MHz)	148.21 ◇ 119.57 ◇◇	N/A	N/A	N/A	100	N/A	120 (Daub-6)	282.50◇◇ (Daub-4) 146.42◇◇ (Daub-6)	168.83 [A] 306.15 [B]

[6] For 8-bit Lena Image [7] Final Reconstruction Step (FRS) [8] Expansion Factor Method (EFM)
[9] Dynamic Power (DP) [10] Maximum Frequency (MF) † Taken from PSNR plots in [1, p. 1264] ‡ At 50 MHz [17]
§ At 442.47 MHz [A] Using a Xilinx Virtex-6 xc6vcx240t-1ff1156 FPGA device
[B] Using CMOS 45 nm implementation generated by Encounter® RTL compiler * At 274.72 MHz †† At 374.25 MHz
‡‡ At 346.58 MHz ◇ For 1-D Design ◇◇ For 2-D Design



(a) $\mathbf{A}_1$ (b) $\mathbf{A}_2$

(c) $\mathbf{A}_3$ (d) $\mathbf{A}_4$

Figure 5.6: (a)–(d) Approximation sub-images $\mathbf{A}_1$, $\mathbf{A}_2$, $\mathbf{A}_3$, and $\mathbf{A}_4$ obtained from on-chip physical verification on a Virtex-6 vcx240t-1ff1156 considering Method 1.

In fact, the proposed architecture preserves all features of our earlier architectures such as (i) error-free computation, (ii) defined over integers, and (iii) free of multiplications. It also reduces hardware complexity (number of adders) leading to considerable reduction in cost for the hardware realization of multi-level DWTs of Daubechies 6-tap filter banks. The *single* FRS is the only source of computational error. Noise injection from intermediate fixed-point errors is fully eliminated. An FPGA based 4-level prototype is operational at 100 MHz. Place-and-route timing analysis furnished 344 MHz for the Daubechies 6-tap architecture.

Table 5.16: SNR and PSNR for Lena Image Approximation for 2-D Daub-6

Scheme	Measured	Approximation			
Used	Aspect	$\mathbf{A}_1$ (256×256)	$\mathbf{A}_2$ (128×128)	$\mathbf{A}_3$ (64×64)	$\mathbf{A}_4$ (32×32)
	SNR (dB)	41.23	35.38	33.06	31.06
Fixed-point	PSNR (dB)	45.05	42.79	41.84	38.75
	SNR (dB)	68.27	65.41	62.48	60.37
Method 1	PSNR (dB)	71.54	67.12	65.05	63.86

CHAPTER VI

CONCLUSIONS & FUTURE WORK

We proposed an optimized multi-rate 1-D/2-D Daub-4 and Daub-6 filter bank architectures which are *purely* AI-based, and multi-encoded. Additionally, the introduced designs are capable of furnishing arbitrarily high numerical accuracy using error-free integer arithmetic. In fact, the proposed architectures preserve all features of our earlier architectures such as (i) error-free computation, (ii) defined over integers, and (iii) free of multiplications. It *also* reduces hardware complexity (number of adders) leading to considerable reduction in cost for the hardware realization of multi-level 2-D DWTs of Daubechies 4-tap/ 6-tap filter banks. The *single* FRS is the only source of computational error.

By employing AI encoding, resulting wavelet decomposed images had SNR and PSNR figures improved by approximately 30–35% when compared to a counterpart fixed-point system with 8-bit word length and 6 fractional bits.

Standard images have been analyzed with `Mandrill`, `Lena`, `Cameraman`, `Woman`, `CT head/brain`, `Reflection` examples shown. Noise injection from intermediate fixed-point errors is fully eliminated. An FPGA based 4-level prototypes are operational at 100 MHz. Place-and-route timing analysis furnished 344 MHz/146 MHz for the multi-level Daubechies 4-tap/6-tap architectures. In addition, the proposed

architectures were also synthesized in 45 nm ASIC technology at 523.56 MHz/ 306 MHz for 8-bit input data.

CMOS sensor arrays for imaging are being continuously improved with increasing resolutions. The dynamic range of typical imaging applications are also increasing and more emphasis is being made for picture quality. In the presence of higher resolution, increased dynamic range, and increased frame rate, there is no option but to increase the throughput of the digital filtering architectures.

Finally, it is important to notice that—in principle—the discussed AI based scheme can be applied to any type of DWT as long as the scaling and wavelet coefficients of the corresponding filters could be given and exact representation. For instance, this is the case for the Haar, Daubechies-4/-6, and Bior-5/3 wavelets. On the other hand, wavelets such as Gaussian and Mexican hat do not have a compatible DWT version.

This work has feasible scope for extending to Daubechies 8-tap and 10-tap wavelet filters provided we can encode the scaling filter coefficients with optimal number of AI bases. Wei Chang et al., [73] can provide more insight and can be used as primary source of reference for this research effort.

BIBLIOGRAPHY

[1] K. Wahid, V. Dimitrov, and G. Jullien. VLSI architectures of Daubechies wavelets for algebraic integers. *Journal of Circuits, Systems, and Computers*, 13(6):1251–1270, 2004.

[2] K. A. Wahid, V. S. Dimitrov, G. A. Jullien, and W. Badawy. An algebraic integer based encoding scheme for implementing Daubechies discrete wavelet transforms. In *Asilomar Conf. Signals, Syst. Comp.*, volume 1, pages 967–971, 2002.

[3] K. A. Wahid, V. S. Dimitrov, G. A. Jullien, and W. Badawy. An analysis of Daubechies discrete wavelet transform based on algebraic integer encoding scheme. In *Proc. Third Int. Workshop Digital and Computational Video DCV 2002*, pages 27–34, 2002.

[4] Guiwei Xing, Jin Li, Shipeng Li, and Ya-Qin Zhang. Arbitrarily shaped video-object coding by wavelet. 11(10):1135–1139, 2001.

[5] Yan Wu, R. J. Veillette, D. H. Mugler, and T. T. Hartley. Stability analysis of wavelet-based controller design. In *Proc. American Control Conf the 2001*, volume 6, pages 4826–4827, 2001.

[6] K. A. Wahid, V. S. Dimitrov, and G. A. Jullien. Error-free arithmetic for discrete wavelet transforms using algebraic integers. In *Proc. 16th IEEE Symp. Computer Arithmetic*, pages 238–244, 2003.

[7] S.-C. B. Lo, Huai Li, and M. T. Freedman. Optimization of wavelet decomposition for image compression and feature preservation. 22(9):1141–1151, 2003.

[8] M. Martone. Multiresolution sequence detection in rapidly fading channels based on focused wavelet decompositions. 49(8):1388–1401, 2001.

[9] P. P. Vaidyanathan. *Multirate Systems and Filter Banks*. PTR Prentice Hall, Englewoodcliffs, New Jersey 07632, 1992.

[10] B.K. Mohanty and P.K. Meher. Merged-cascaded systolic array for vlsi implementation of discrete wavelet transform. In *Circuits and Systems, 2006. APCCAS 2006. IEEE Asia Pacific Conference on*, pages 462–465, 2006.

[11] P.-Q. Vaidyanathan, P. P. , Hoang. Lattice structures for optimal design and robust implementation of two-channel perfect-reconstruction QMF banks. *IEEE Transactions on Circuits and Systems*, 36(1):81–94, 1988.

[12] D. B. H. Tay. Balanced spatial and frequency localised 2-D nonseparable wavelet filters. In *Proc. IEEE Int. Symp. Circuits and Systems ISCAS 2001*, volume 2, pages 489–492, 2001.

[13] S. Mallat. *A Wavelet Tour of Signal Processing*. Academic Press, Burlington, MA, 2008.

[14] D. B. H. Tay and N. G. Kingsbury. Design of nonseparable 3-D filter banks/wavelet bases using transformations of variables. *IEEE Proceedings on Visual Image Signal processing*, 143:51–61, 1996.

[15] Abdelhamid Meraghni Mountassar Maamoun , Mehdi Neggazi and Daoud Berkani. VLSI design of 2-D disceret wavelet transform for area efficient and high-speed image computing. *World Academy of Science, Engineering and Technology*, 45:538–543, 2008.

[16] Selvaraju Murugesan and D. B. H Tay. New techniques for rationalizing orthogonal and biorthogonal wavelet filter coefficients. *IEEE Transactions on Circuits and Systems*, 59(3):628–637, March 2011.

[17] Md. Ashraful Islam and K. A. Wahid. Area- and power-efficient design of Daubechies wavelet transforms using folded AIQ mapping. 57(9):716–720, September 2010.

[18] F. Marino, D. Guevorkian, and J. T. Astola. Highly efficient high-speed/low-power architectures for the 1-D discrete wavelet transform. 47(12):1492–1502, 2000.

[19] K. A. Wahid, V. S. Dimitrov, G. A. Jullien, and W. Badawy. Error-free computation of Daubechies wavelets for image compression applications. *Electronics Letters*, 39(5):428–429, 2003.

[20] Khan A. Wahid. *Low Complexity Implementation of Daubechies Wavelets for Medical Imaging Applications*. InTech, 2011.

[21] T. Acharya and Po-Yueh Chen. VLSI implementation of a DWT architecture. In *Proc. IEEE Int. Symp. Circuits and Systems ISCAS '98*, volume 2, pages 272–275, 1998.

[22] S. Gnavi, B. Penna, M. Grangetto, E. Magli, and G. Olmo. DSP performance comparison between lifting and filter banks for image coding. In *Proc. IEEE Int. Acoustics, Speech, and Signal Processing (ICASSP) Conf.*, volume 3, 2002.

[23] I. Urriza, J. I. Artigas, J. I. Garcia, L. A. Barragan, and D. Navarro. VLSI architecture for lossless compression of medical images using the discrete wavelet transform. In *Proc. Design, Automation and Test in Europe*, pages 196–201, 1998.

[24] B. K. Mohanty, A. Mahajan, and P. K . Meher. Area and power efficient architecture for high-throughput implementation of lifting 2-D DWT. *IEEE Trans. Circuits. Syst. II, Exp. Briefs.*, 59(7):434–438, July 2012.

[25] B. K. Mohanty and P. K. Meher. Memory-efficient high-speed convolution-based generic structure for multilevel 2-D DWT. *IEEE Transactions on Circuits and Systems for Video Technology*, 23:353–363, 2013.

[26] J. P. Andrew, P. O. Ogunbona, and F. J. Paoloni. Comparison of "wavelet" filters and subband analysis structures for still image compression. In *Proc. IEEE Int Acoustics, Speech, and Signal Processing ICASSP*, 1994.

[27] H. Kaida and M. Okuda. Image compression suitable for high dynamic range image rendering. In *Proc. Int. Conf. Advanced Information Networking and Applications - Workshops*, pages 1029–1033, 2008.

[28] Subhasis Saha. Image compression— from DCT to wavelets— A review, 2000.

[29] R. Baghaie and V. Dimitrov. Computing Haar transform using algebraic integers. In *Proc. Conf Signals, Systems and Computers Record of the Thirty-Fourth Asilomar Conf*, volume 1, pages 438–442, 2000.

[30] D. Knuth. *The Art of Computer Programming*, volume 2. Addison Wesley, 3rd edition, 1981.

[31] Khan A. Wahid, Md. Ashraful Islam, and Seok-Bum Ko. Lossless implementation of Daubechies 8-tap wavelet transform. In *Proc. IEEE Int. Symp. Circ. Syst.*, pages 2157–2160, Rio de Janeiro, Brazil, May 2011.

[32] J. H. Park, K. O. Kim, and Y. K. Yang. Image fusion using multiresolution analysis. In *Proceedings of the IEEE International Geoscience and Remote Sensing Symposium*, volume 2, pages 864–866, Sydney, Australia, 2001.

[33] G. Plonka. A global method for invertible integer DCT and integer wavelet algorithms. *Applied and Computational Harmonic Analysis*, 16(2):79–110, March 2004.

[34] G. Dimitroulakos, M. D. Galanis, A. Milidonis, and C. E. Goutis. A high-throughput and memory efficient 2D discrete wavelet transform hardware architecture for JPEG2000 standard. In *Proc. IEEE Int. Symp. Circuits and Systems ISCAS 2005*, pages 472–475, 2005.

[35] Bing-Fei Wu and Chung-Fu Lin. A high-performance and memory-efficient pipeline architecture for the 5/3 and 9/7 discrete wavelet transform of JPEG2000 codec. 15(12):1615–1628, 2005.

[36] K. A. Wahid, V. S. Dimitrov, and G. A. Jullien. On the error-free realization of a scaled DCT algorithm and its VLSI implementation. 54(8):700–704, 2007.

[37] D. B. H. Tay. Integer wavelet transform for medical image compression. In *Proc. Intelligent Information Systems Conf. The Seventh Australian and New Zealand 2001*, pages 357–360, 2001.

[38] A. Skodras, C. Christopoulos, and T. Ebrahimi. The JPEG 2000 still image compression standard. 18:36–58, 2001.

[39] Q. Dai, X. Chen, and C. Lin. A novel VLSI architecture for multidimensional discrete wavelet transform. 14(8):1105–1110, August 2004.

[40] C. Huang, P. Tseng, and L. Chen. Flipping structure: an efficient VLSI architecture for lifting based discrete wavelet transform. *IEEE Trans. Signal. Process.*, 52(4):1080–1089, April 2004.

[41] G. Shi, W. Liu, and F. Li. An efficient folded architecture for lifting-based discrete wavelet transform. *IEEE Trans. Circuits. Syst. II , Exp. Briefs*, 56(4):290–294, April 2009.

[42] M. Martina and G. Masera. Multiplierless, folded 9/7-5/3 wavelet VLSI architecture. *IEEE Trans. Circuits. Syst. II , Exp. Briefs*, 54(9):770–774, September 2007.

[43] J. Walker. *A Primer on Wavelets and their Scientific Applications*. Chapman & Hall/ CRC Press, Boca Raton, FL, 1999.

[44] K. Wahid, Seok-Bum Ko, and D. Teng. Efficient hardware implementation of an image compressor for wireless capsule endoscopy applications. In *Proc. (IEEE World Congress Computational Intelligence). IEEE Int. Joint Conf. Neural Networks IJCNN 2008*, pages 2761–2765, 2008.

[45] Martin Vetterli and Jelena Kovačević. *Wavelets and Subband coding*. Prentice Hall PTR, Englewood Cliffs, NJ, 1995.

[46] S. G. Mallat. A theory for multiresolution signal decomposition: the wavelet representation. 11(7):674–693, 1989.

[47] M. Misiti, Y. Misiti, G. Oppenheim, and J.-M. Poggi. *Wavelet Toolbox User's Guide*. Mathworks, Inc, 2011.

[48] W. H. Press, S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery. *Numerical recipes in C*. Cambridge University Press, 2 edition, 1999.

[49] J. Cozzens and L. Finkelstein. Computing the discrete Fourier transform using residue number systems in a ring of algebraic integers. 31(5):580–588, 1985.

[50] Richard Dedekind. *Theory of Algebraic integers*. September 1996.

[51] G. H. Hardy and E. M. Wright. *An Introduction to the Theory of Numbers*. Oxford University Press, London, 4 edition, 1975.

[52] K. Wahid, V. Dimitrov, and G. Jullien. Error-free computation of 8×8 2D DCT and IDCT using two-dimensional algebraic integer quantization. In *Proc. 17th IEEE Symp. Comp. Arith.*, pages 214–221, 2005.

[53] R. J. Cintra. An integer approximation method for discrete sinusoidal transforms. *Journal of Circuits, Systems, and Signal Processing*, 30(6):1481–1501, 2011.

[54] R. Baghaie and V. Dimitrov. Systolic implementation of real-valued discrete transforms via algebraic integer quantization. *Computers and Mathematics with Applications*, 41:1403–1416, 2001.

[55] V. S. Dimitrov, G. A Jullien, and W. C. Miller. A new DCT algorithm based on encoding algebraic integers. In *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing*, volume 3, pages 1377–1380, Seattle, WA, 1998.

[56] A. Madanayake, R. J. Cintra, D. Onen, V. S. Dimitrov, N. T. Rajapaksha, L. T. Bruton, and A. Edirisuriya. A row-parallel 8×8 2-D DCT architecture using algebraic integer based exact computation. *IEEE Trans. Circuits. Syst. Video Technol.,*, 22(6):915–929, June 2011.

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

[58] Richard A. Games, Sean D. O'Neil, and Joseph J. Rushanan. Algebraic integer quantization and conversion. Technical report, Rome Air Development Center, Griffiss Air Force Base, NY 13441-5700, July 1988.

[59] ImageProcessingPlace.com. Image databases. `http://www.imageprocessingplace.com/root_files_V3/image_databases.htm`, March 2012.

[60] Tze-Yun Sung, Hsi-Chin Hsin, Yaw-Shih Shieh, and Chun-Wang Yu. Low-power multiplierless 2-D DWT and IDWT architectures using 4-tap daubechies filters. In *Proc. Seventh Int. Conf. Parallel and Distributed Computing, Applications and Technologies PDCAT '06*, pages 185–190, 2006.

[61] F. Marino. Two fast architectures for the direct 2-D discrete wavelet transform. 49(6):1248–1259, 2001.

[62] Chao-Tsung Huang, Po-Chih Tseng, and Liang-Gee Chen. Generic RAM-based architectures for two-dimensional discrete wavelet transform with line-based method. 15(7):910–920, 2005.

[63] Hongyu Liao, M. Kr. Mandal, and B. F. Cockburn. Efficient architectures for 1-D and 2-D lifting-based wavelet transforms. 52(5):1315–1326, 2004.

[64] K. Andra, C. Chakrabarti, and T. Acharya. A VLSI architecture for lifting-based forward and inverse wavelet transform. 50(4):966–977, 2002.

[65] Chunhui Zhang, Yun Long, Seong Yong Oum, and F. Kurdahi. 'software-pipelined' 2-D discrete wavelet transform with VLSI hierarchical implementation. In *Proc. IEEE Int Robotics, Intelligent Systems and Signal Processing Conf*, volume 1, pages 148–153, 2003.

[66] Keshab. K. Parhi. *VLSI Digital Signal Processing Systems Design & Implementation*. John Wiley & Sons, Inc., 2007.

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

[68] Chao-Tsung Huang, Po-Chih Tseng, and Liang-Gee Chen. Efficient VLSI architectures of lifting-based discrete wavelet transform by systematic design method. In *Proc. IEEE Int. Symp. Circuits and Systems ISCAS 2002*, volume 5, 2002.

[69] Bing-Fei Wu and Chung-Fu Lin. A rescheduling and fast pipeline VLSI architecture for lifting-based discrete wavelet transform. In *Proc. Int. Symp. Circuits and Systems ISCAS '03*, volume 2, 2003.

[70] Shiva Madishetty, Arjuna Madanayake, Renato J. Cintra, Dale Mugler, and Vassil Dimitrov. Error-free VLSI architecture for Daubechies 4-tap filter using algebraic integers. In *ISCAS*, pages 1484–1487, May 2012.

[71] Shiva Madishetty, Arjuna Madanayake, Renato J. Cintra, Vassil Dimitrov, and Dale Mugler. VLSI architecture for Daubechies 4-tap and 6-tap wavelet filters using algebraic integers. *IEEE Trans. Circuits. Syst. I (Accepted, In Press)*, 2012.

[72] J. E. Stine, I. Castellanos, M. Wood, J. Henson, F. Love, W. R. Davis, P. D. Franzon, M. Bucher, S. Basavarajaiah, J. Oh, and R. Jenkal. FreePDK: An open-source variation-aware design kit. In *IEEE International Conference on Microelectronic Systems Education, 2007*, pages 173–174, June 2007.

[73] Wei Chang Shann and Chien Chang Yen. Exact solutions for daubechies orthonormal scaling coefficients. Technical Report 9704, Department of Mathematics, National Central University, September 1997.