HARDWARE IMPLEMENTATION OF DAUBECHIES WAVELET TRANSFORMS USING FOLDED AIQ MAPPING

A Thesis Submitted to the College of

Graduate Studies and Research

In Partial Fulfillment of the Requirements

For the Degree of Master of Science

In the Department of Electrical and Computer Engineering

University of Saskatchewan

Saskatoon

By

Md Ashraful Islam

PERMISSION TO USE

ABSTRACT

The Discrete Wavelet Transform (DWT) is a popular tool in the field of image and video compression applications. Because of its multi-resolution representation capability, the DWT has been used effectively in applications such as transient signal analysis, computer vision, texture analysis, cell detection, and image compression. Daubechies wavelets are one of the popular transforms in the wavelet family. Daubechies filters provide excellent spatial and spectral locality-properties which make them useful in image compression.

In this thesis, we present an efficient implementation of a shared hardware core to compute two 8-point Daubechies wavelet transforms. The architecture is based on a new two-level folded mapping technique, an improved version of the Algebraic Integer Quantization (AIQ). The scheme is developed on the factorization and decomposition of the transform coefficients that exploits the symmetrical and wrapping structure of the matrices. The proposed architecture is parallel, pipelined, and multiplexed. Compared to existing designs, the proposed scheme reduces significantly the hardware cost, critical path delay and power consumption with a higher throughput rate.

Later, we have briefly presented a new mapping scheme to error-freely compute the Daubechies-8 tap wavelet transform, which is the next transform of Daubechies-6 in the Daubechies wavelet series. The multidimensional technique maps the irrational transformation basis coefficients with integers and results in considerable reduction in hardware and power consumption, and significant improvement in image reconstruction quality.

ACKNOWLEDGEMENTS

First, I would like to express my gratitude to my supervisor, Dr. Khan A.Wahid for introducing me to this research field and for his continuous support, inspiration, and guidance throughout the project. His vast knowledge and expertise in this field added considerably to my graduate experience.

Then, I would like to thank my thesis committee for their guidance and insightful comments.

I would also like to acknowledge the financial support from NSERC, Graduate Studies and Research, University of Saskatchewan and Department of Electrical and Computer Engineering, University of Saskatchewan.

Finally, I would like to thank my family for their love, encouragement, and support.

This thesis is dedicated to my loving parents.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

ASIC	Application Specific Integrated Circuit
AIQ	Algebraic Integer Quantization
CAD	Computer Aided Design
DCT	Discrete Cosine Transform
DWT	Discrete Wavelet Transform
FP	Fixed-Point
FRS	Final Reconstruction Step
FPGA	Field Programmable Gate Array
HVS	Human Visual System
IWT	Integer Wavelet Transform
ISO	International Organization for Standardization
ITU	International Telecommunication Union
JPEG	Joint Photographic Experts Group
VLSI	Very Large Scale Integration

Chapter 1
Introduction

1.1 Introduction

The Discrete Wavelet Transform (DWT) [1, 2] has extensively been used in a wide range of applications, including image and video coding, pattern recognition, etc. After its inclusion in JPEG2000 compression standard [3], significant research has been done to optimize the DWT implementation to reduce the computational complexity, as most applications using it demand real-time processing. The Cohen-Daubechies-Feauveau 9/7 bi-orthogonal [4] and the Daubechies wavelets (DAUB2-DAUB20) [5] are the two highly popular types of basis functions. Most of the research work to reduce the hardware complexity is inclined towards multiplierless implementations by maneuvering the filter banks [6, 7] or using lifting schemes [8, 9, 10, 11]. In all these designs, the use of conventional fixed-point (FP) binary representation to implement the irrational transform coefficients introduces round-off error at the beginning of the process, which then transmits throughout the entire computation process and degrades image reconstruction. In some cases, the Integer Wavelet Transform (IWT) is used, but the need for large bit-width adder tree and complex control circuitry leads to higher implementation cost, as well as poor reconstruction [12, 13, 14].

1.2 Motivation

In order to eliminate the round-off error for DWT, Wahid et al. [15, 16] proposed an Algebraic Integer Quantization (AIQ) technique to compute the Daubechies-4 tap and Daubechies-6 tap filter coefficients, where the irrational transform basis functions are mapped with integers, resulting in very efficient computation. The AIQ representation of the wavelet coefficients provides error-free calculations until the final reconstruction step. This also makes the hardware architecture simple, multiplication-free and inherently parallel. However, in this technique a direct (one-level) AIQ mapping is utilized which involves large number of internal computing elements that result in high cost of hardware resources and silicon area and low throughput.

In our work, we propose a new two-level encoding, called folded mapping, which combines the error-free AIQ scheme and matrix decomposition techniques to compute the DAUB4 and DAUB6 wavelet coefficients. This new scheme enables resource sharing and yields lower hardware cost and power consumption, and higher process throughput. The AIQ-encoded forward basis transformation matrices are first decomposed into multiple sub-matrices; the common structures of the sub-matrices are identified and later shared for implementation. The design has been prototyped onto FPGA and VLSI using 0.18μm CMOS standard cells.

The architecture houses both DAUB4 and DAUB6, where the user selects the desired transform unit based on the application and end user’s requirement. For slowly varying signals, the DAUB4 gives the best results; while for rapidly varying signals, the DAUB6 performs well at high noise levels [4]. The DAUB6 also provides better details of medical images than the DAUB4, but at the expense of consuming more power. It suits nicely in an image compressor for applications like the wireless capsule endoscopy [32]. When the capsule passes through human gastrointestinal tract, having two transforms in one chip will provide the physicians with an added feature of using DAUB6 when images of finer details are desired, and then switching back to DAUB4 for unimportant and/or unconcerned regions (to save power consumption and battery life).

Later, we apply a one-level AIQ scheme to implement the Daubechies-8 wavelet transform. The use of conventional fixed-point binary representation for implementing Daubechies wavelets, introduces round-off or approximation errors at the very beginning of the process due to the lack of exact representation of the irrational numbers that form the coefficient basis. These errors tend to increase as the calculations progress through the architecture, degrading the quality of the image reconstruction. The AIQ mapping helps to reduce the computational and approximation error.

1.3 Thesis objective

This thesis work is directed towards the design of two DWT processors using two-level folded AIQ and one-level AIQ schemes. There are three general objectives behind this research.

1) To investigate the folded AIQ representation of Daubechies-4 & Daubechies-6 tap wavelet coefficients. The motivation for using folded AIQ to represent

Daubechies-4 and Daubechies-6 filter coefficients comes from studying the conjugate structure of the coefficients in closed form representation. The new scheme leads to develop the architecture, and implementation on FPGA and VLSI platform.

2) To investigate the one-level AIQ representation of Daubechies-8 tap wavelet coefficients.

3) To demonstrate the potential performance advantages of these architectures in terms of reduced hardware cost and improved image reconstruction.

All details of these implementations, especially the different design units, will be developed in this thesis work. The results of a final simulation, using a standard image, will be discussed in terms of image reconstruction, hardware complexity and achievable precision. These results will be targeted towards the general area of image compression applications.

1.4 Thesis organization

This thesis is organized into six chapters. In Chapter 2, we briefly discuss the DWT algorithm and present the matrix representation of Daubechies-4, Daubechies-6, and Daubechies-8 wavelet transform. The application of AIQ to implement Daubechies wavelet coefficients is presented in Chapter 3. Chapter 4 presents the proposed folded AIQ algorithm for the Daubechies-4 and Duabechies-6 filter banks. Later, an AIQ based algorithm and architecture of Daubechies-8 is presented. The architectural details, hardware mapping, FPGA and VLSI simulation, synthesis and fabrication results along with performance analysis are summarized in Chapter 5. Chapter 6 concludes the thesis by summarizing the accomplishment of the research work and giving recommendations for future exploration.

Chapter 2
Discrete Wavelet Transform

2.1 Introduction

This chapter presents a brief description to the Discrete Wavelet Transform (DWT). Low bit-rate image compression is essential for the transmission and storage of digital images. A number of different techniques for image coding have been proposed, but due to some very attractive characteristics, the DWT has proven to be very useful. The DWT has extensively been used in a wide range of applications, including numerical analysis, image and video coding, pattern recognition, etc. For many years the Discrete Cosine Transform (DCT) [17] has been the core transform for image compression algorithms. The DCT is used in the JPEG [18] image compression standard. Despite all the advantages of JPEG compression schemes based on the DCT, there are noticeable and annoying "blocking artifacts" particularly at low bit rates due to the inherent characteristics of DCT. To apply DCT the input image needs to be "blocked", which results in correlation across the block boundaries. In case of DWT, the transform is applied to the entire image. As a result, there are no blocking artifacts. The DWT transforms discrete signal from the time domain into time frequency domain. The transformation product is a set of coefficients organized in the way that enables not only spectrum analysis of the signal also spectral behavior of the signal in time. The wavelet transform has emerged as a cutting edge technology, within the field of image compression. Wavelet-based coding provides substantial improvements in picture quality at higher compression ratios [19]. In addition, wavelet coding is better matched to the characteristics of the Human Visual System (HVS). Because of their inherent multiresolution nature [20], wavelet coding schemes are especially suitable for applications where scalability and tolerable degradation are important. After its inclusion in JPEG2000 compression standard, ISO/ITU-T standard for still image coding, etc. significant research has been done to optimize the DWT implementation to reduce the computational complexity, as most applications using it demand real-time processing.

2.2 Definition of Wavelets

Wavelets are functions defined over a finite interval and having an average value of zero [21]. The basic idea of any wavelet transform is to represent an arbitrary function, *x(t),* as a superposition of a set of such wavelets or basis functions. These basis functions or *baby wavelets* are obtained from a single prototype wavelet called the *mother wavelet*, by dilations or contractions (scaling) and translations (shifts). The Wavelet Transform of a finite length signal, *x*(*t*), is given by Eqn. (2.1).

$$\Psi(\tau, s) = \frac{1}{s}\int x(t)\,\Theta\,\Psi^{0}\left(\frac{t-\tau}{s}\right)dt \qquad (2.1)$$

Here, τ = translation and s = scaling.

In spatial domain image processing, the operation is discretized. The Discrete Finite Wavelet Transform can be represented as a matrix, $\Psi(c)$ and the DWT coefficients can be obtained by taking the inner product between the input signals and the wavelet matrix.

Since the basis functions are the translated and dilated versions of each other, the DWT coefficients of one stage can be calculated from the DWT coefficients of the previous stage.

2.2.1 Subband Coding

Subband coding [22] has been used extensively in speech coding and in image coding because of its inherent advantages; namely, variable bit assignment among the subbands and coding error confinement within the subbands. The DWT, due to based on subband coding, reduces the computation time and resources required. In subband coding, an image is decomposed into a set of band limited components, called subband. The decomposition and reconstruction are performed by digital filters. Therefore, in DWT, a time-scale representation of the digital signal is obtained using digital filtering techniques.

Filters are one of the most important functions in signal processing. Wavelets can be implemented by iteration of filters with rescaling. The resolution of the signal (a measure of the amount of detail information in the signal) is determined by the filtering operations, and the scale is determined by subsampling (upsampling and downsampling) operations. The DWT is computed by successive highpass and lowpass filtering of the discrete time-domain signal as shown in Figure 2.1. This is known as the Mallat algorithm or Mallat-tree decomposition [23]. It has the significance of connecting the continuous-time mutiresolution to discrete-time filters. In Figure 2.1, the input signal is denoted by the sequence X[n], where n is an integer. The low pass filter is denoted by LOW while the high pass filter is denoted by HIGH. The first level of decomposition extracts the details (high frequency components, d[n]) of the signal while the second and all the subsequent decompositions extract progressively coarser information (low frequency components, a[n]).

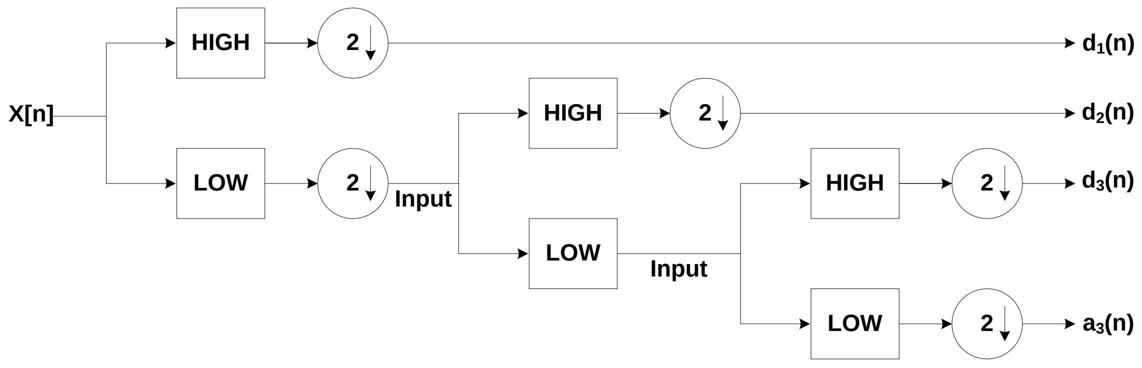


Figure 2.1 Three-level decomposition of wavelet algorithm

This decomposition halves the time resolution since only half the number of samples now characterizes the entire signal. After passing the original signal through a half band lowpass filter, according to Nyquist's rule, half of the samples can be eliminated [24]. Since the signal now has a highest frequency of ω /2 radians instead of ω. The signal can therefore be subsampled by 2, simply by discarding every other sample. The above procedure can be repeated for further decomposition until the desired level is reached. The number of levels depends on the length of the signal. The original signal is then realized by concatenating all the coefficients, a[n] and d[n], starting from the last level of decomposition. Figure 2.2 shows the rebuilding of the original signal from the wavelet coefficients. In Figure 2.2, the output signal is denoted by the sequence Y[n], where n is an integer. The reconstruction is basically the reverse process of decomposition. The detail, d[n] and approximation coefficients, a[n] at every level are up-sampled by two, passed through the high pass and low pass synthesis filters and then added. This process requires the same number of levels as in the decomposition process to obtain the original signal.

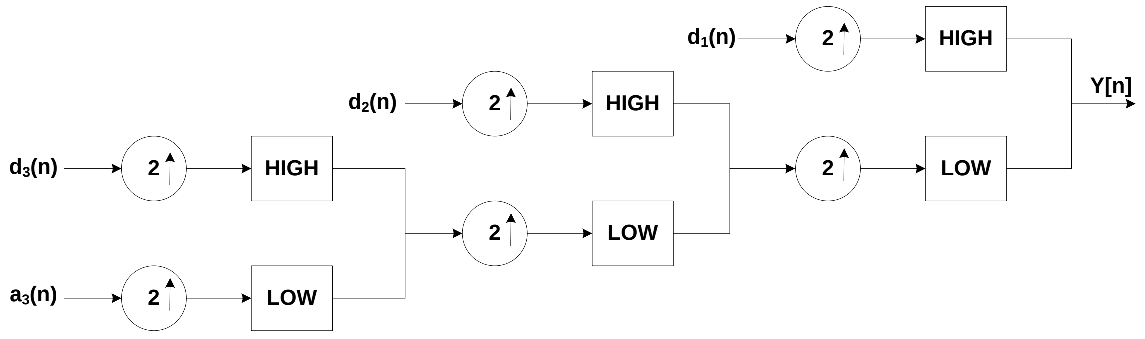


Figure 2.2 Three-level reconstruction of wavelet algorithm

Image processing is vastly benefitted from this particular property of the wavelet transform. DWT can be applied to reduce the image size without losing much of the resolution. For a given image, DWT of each row can be computed, and all values in the DWT that are less than a certain threshold can be discarded. Only those DWT coefficients that are above the threshold for each row can be saved, and used to reconstruct the original image by simply padding each row with as many zeros as the number of discarded coefficients, and then apply the inverse DWT to reconstruct each row of the original image.

Figure 2.3 shows the 3-level decomposition of benchmark “Lena” image using 2-D Daubechies-6 wavelet transform.

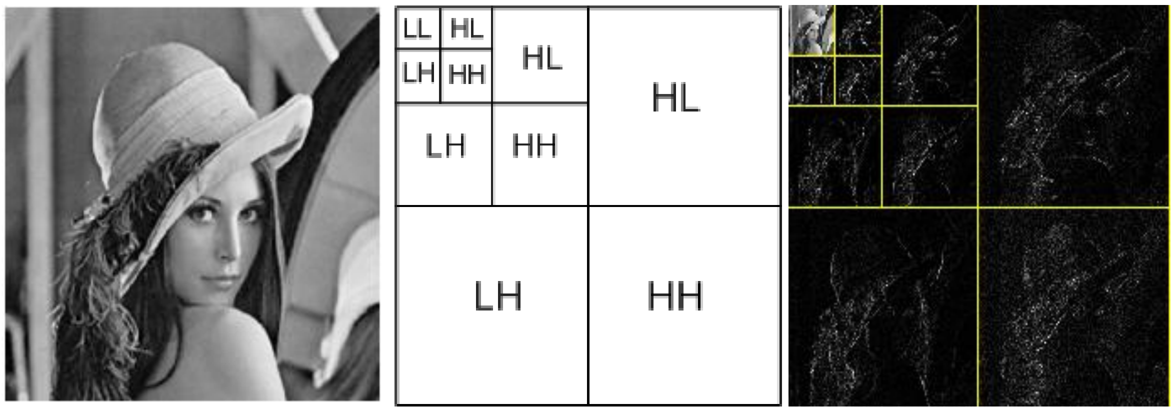


Figure 2.3 (a) Original “Lena” Image; (b) 3-level decomposition of DWT; (c) Decomposed “Lena” image.

2.3 Daubechies Wavelet Transform

Special families of wavelet functions are developed for the DWT. These wavelets are compactly supported, orthogonal or biorthogonal and are characterized by high-pass and low-pass analysis and synthesis filters. The Daubechies wavelets are a family of orthogonal wavelets defining a DWT. The Daubechies family is named after Ingrid Daubechies (a Belgian physicist and mathematician) who invented the compactly supported orthonormal wavelets, making wavelet analysis in discrete time possible. The research work presented in this thesis is restricted to Daubechies wavelets because they immediately lend themselves to AIQ implementation, and are also useful, having class members ranging from highly localized to highly smooth and providing excellent performance in image compression applications. Daubechies wavelet coefficients are based on computing wavelet coefficients, C_n (where n = 0, 1, 2. . . N-1 and N is the number of coefficients) to satisfy the following conditions [25]:

(1) The conservation of area under $x(t) : \sum_n C_n = 2$

(2) The accuracy conditions: $\sum_n (-1)^n n^m C_n = 0\ (where\ m = 0, 1, 2, ..., P-1\ and\ P = \frac{N}{2})$

(3) The perfect reconstruction conditions: $\sum_n C_n^2 = 2\ and\ \sum_n C_n . C_{n+2m} = 0$

Then the low-pass filter is $h(n) = \frac{C_n}{2}$ and the high-pass filter is $g(n) = (-1)^{n+1} h(n - N - 1)$.

Daubechies coefficients range from Daubechies-2 (in short, DAUB2 with 2 coefficients) to Daubechies-20 (DAUB20, 20 coefficients). In this research work, we study the hardware architectures associated with implementing Daubechies-4 (DAUB4), Daubechies-6 (DAUB6) and Daubechies-8 (DAUB8).

2.3.1 DAUB4 Wavelet Transform

The simplest and most localized member in Daubechies family is the DAUB4, which has four coefficients, C_0, C_1, C_2 and C_3 [26] as given in Eqn. (2.2).

$$
\begin{aligned}
C_0 &= (1+\sqrt{3})/4\sqrt{2} \qquad & C_1 &= (3+\sqrt{3})/4\sqrt{2} \\
C_2 &= (3-\sqrt{3})/4\sqrt{2} \qquad & C_3 &= (1-\sqrt{3})/4\sqrt{2}
\end{aligned}
\tag{2.2}
$$

For a 8x8 input data, the DAUB4 forward transform is shown in Eqn. (2.3). In the forward transform matrix, the odd row coefficients, C_0, C_1, C_2 and C_3 implement a low-pass filter and the even row coefficients, $C_3, -C_2, C_1$ and $-C_0$ implement a high-pass filter. Also, the DWT is orthogonal and invertible - the inverse transform, is simply the transpose of the forward transform matrix.

$$\psi_4(C) = \begin{bmatrix} C_0 & C_1 & C_2 & C_3 & 0 & 0 & 0 & 0 \\ C_3 & -C_2 & C_1 & -C_0 & 0 & 0 & 0 & 0 \\ 0 & 0 & C_0 & C_1 & C_2 & C_3 & 0 & 0 \\ 0 & 0 & C_3 & -C_2 & C_1 & -C_0 & 0 & 0 \\ 0 & 0 & 0 & 0 & C_0 & C_1 & C_2 & C_3 \\ 0 & 0 & 0 & 0 & C_3 & -C_2 & C_1 & -C_0 \\ C_2 & C_3 & 0 & 0 & 0 & 0 & C_0 & C_1 \\ C_1 & -C_0 & 0 & 0 & 0 & 0 & C_3 & -C_2 \end{bmatrix} \tag{2.3}$$

2.3.2 DAUB6 Wavelet Transform

The DAUB6, which performs well at high noise levels compare to DAUB4 [15] has six coefficients, C_0, C_1, C_2, C_3, C_4 and C_5 [26] as given in Eqn. (2.4).

$$\begin{aligned} C_0 &= \frac{(1+\sqrt{10}+\sqrt{5+2\sqrt{10}})}{16\sqrt{2}} & C_1 &= \frac{(5+\sqrt{10}+3\sqrt{5+2\sqrt{10}})}{16\sqrt{2}} \\ C_2 &= \frac{(10-2\sqrt{10}+2\sqrt{5+2\sqrt{10}})}{16\sqrt{2}} & C_3 &= \frac{(10-2\sqrt{10}-2\sqrt{5+2\sqrt{10}})}{16\sqrt{2}} \\ C_4 &= \frac{(5+\sqrt{10}-3\sqrt{5+2\sqrt{10}})}{16\sqrt{2}} & C_5 &= \frac{(1+\sqrt{10}-\sqrt{5+2\sqrt{10}})}{16\sqrt{2}} \end{aligned} \tag{2.4}$$

For a 8x8 input data, the Daubechies-6 forward transform matrix is shown in Eqn. (2.5).

$$\psi_6(C) = \begin{bmatrix} C_0 & C_1 & C_2 & C_3 & C_4 & C_5 & 0 & 0 \\ C_5 & -C_4 & C_3 & -C_2 & C_1 & -C_0 & 0 & 0 \\ 0 & 0 & C_0 & C_1 & C_2 & C_3 & C_4 & C_5 \\ 0 & 0 & C_5 & -C_4 & C_3 & -C_2 & C_1 & -C_0 \\ C_4 & C_5 & 0 & 0 & C_0 & C_1 & C_2 & C_3 \\ C_1 & -C_0 & 0 & 0 & C_5 & -C_4 & C_3 & -C_2 \\ C_2 & C_3 & C_4 & C_5 & 0 & 0 & C_0 & C_1 \\ C_3 & -C_2 & C_1 & -C_0 & 0 & 0 & C_5 & -C_4 \end{bmatrix} \qquad (2.5)$$

2.3.3 DAUB8 Wavelet Transform

The DAUB8 has eight coefficients $\mathrm{C}_0, \mathrm{C}_1, \mathrm{C}_2, \mathrm{C}_3, \mathrm{C}_4, \mathrm{C}_5, \mathrm{C}_6$ and C_7 [27] as given in Eqn. (2.6).

$$C_0 = a_0 / 32\sqrt{2} \qquad C_1 = (a_1 + 4a_0) / 32\sqrt{2}$$

$$C_2 = (a_2 + 4a_1 + 6a_0) / 32\sqrt{2} \qquad C_3 = (a_3 + 4a_2 + 6a_1 + 4a_0) / 32\sqrt{2}$$

$$C_4 = (a_0 + 4a_1 + 6a_2 + 4a_1) / 32\sqrt{2} \qquad C_5 = (a_1 + 4a_2 + 6a_3) / 32\sqrt{2} \qquad (2.6)$$

$$C_6 = (a_2 + 4a_3) / 32\sqrt{2} \qquad C_7 = a_3 / 32\sqrt{2}$$

Where $a_0 = \alpha$, $a_1 = 2 - \sqrt{140} + \dfrac{5}{\alpha}$, $a_1 = 2 + \sqrt{140} - \alpha$, $a_3 = -\dfrac{5}{\alpha}$

Where α is the root of the following polynomial

$$\alpha^4 - 2(1+\sqrt{35})\alpha^3 + 40\alpha^2 + 10(1-\sqrt{35})\alpha + 25 = 0$$

The polynomial has a pair of conjugate complex roots and two different real roots α_1 and α_2. The value of α_1 is given by:

$$\alpha_1 = \frac{1}{2} + \frac{1}{2}\sqrt{35} + \frac{1}{6}\sqrt{\frac{3v}{u^{\frac{1}{3}}}} + \frac{1}{6}\sqrt{3s}$$

The value of α_1 is approximately 10.4257, gives the "minimum phase" orthonormal scaling function (this is the normal choice). The value of α_2 is given by:

$$\alpha_2 = \frac{1}{2} + \frac{1}{2}\sqrt{35} + \frac{1}{6}\sqrt{\frac{3v}{u^{\frac{1}{3}}}} - \frac{1}{6}\sqrt{3s}$$

The value of α_2 is approximately 1.4583, gives the "least asymmetric" orthonormal scaling function.

For both of the above expressions,

$$u = 700 + 210\sqrt{15}$$

$$v = 28u^{\frac{1}{3}} - 70.2^{\frac{1}{3}} + (2u)^{\frac{2}{3}} + 6u^{\frac{1}{3}}.\sqrt{35}$$

$$s = \frac{56u^{\frac{1}{3}}\sqrt{v} + 70.2^{\frac{1}{3}}\sqrt{v} - (2u)^{\frac{2}{3}}\sqrt{v} + 12u^{\frac{1}{3}}\sqrt{35v} + 336\sqrt{3u} + 48\sqrt{105v}}{u^{\frac{1}{3}}\sqrt{v}}$$

For a 8x8 input data, the DAUB8 forward transform matrix is shown in Eqn. (2.7).

$$\psi_8(C) = \begin{bmatrix} C_0 & C_1 & C_2 & C_3 & C_4 & C_5 & C_6 & C_7 \\ C_7 & -C_6 & C_5 & -C_4 & C_3 & -C_2 & C_1 & -C_0 \\ C_6 & C_7 & C_0 & C_1 & C_2 & C_3 & C_4 & C_5 \\ C_1 & -C_0 & C_7 & -C_6 & C_5 & -C_4 & C_3 & -C_2 \\ C_4 & C_5 & C_6 & C_7 & C_0 & C_1 & C_2 & C_3 \\ C_3 & -C_2 & C_1 & -C_0 & C_7 & -C_6 & C_5 & -C_4 \\ C_2 & C_3 & C_4 & C_5 & C_6 & C_7 & C_0 & C_1 \\ C_5 & -C_4 & C_3 & -C_2 & C_1 & -C_0 & C_7 & -C_6 \end{bmatrix} \tag{2.7}$$

2.4 Summary

In this chapter, we have presented the fundamentals of wavelets, subband coding (decompositions of signals in subbands) and Daubechies wavelet transforms. In the final part, the properties and mathematical expressions of Daubechies-4, 6 and 8 coefficients are presented which are helpful for the error-free encoding to be presented later in this thesis.

Chapter 3

Algebraic Integer Quantization

3.1 Introduction

This chapter presents a brief description to Algebraic Integer Quantization (AIQ) [28]. The idea of using AIQ in DSP applications is first explored by Cozzens and Finkelstein [29]. The use of any conventional number representation introduces approximation errors at the very beginning of the process due to the lack of exact representation of the irrational numbers that form the coefficient basis. These errors tend to propagate through the wavelet transform computation and degrade the quality of image reconstruction. The AIQ mapping helps to reduce the computational cost and approximation errors.

3.2 Algebraic Integer Quantization (AIQ)

AIQ is defined by real numbers that are roots of monic polynomials with integer coefficients [30]. As an example, let $\omega = e^{\frac{2\pi j}{16}}$ denote a primitive 16th root of unity over the ring of complex numbers. Then ω satisfies the equation $x^8 + 1 = 0$. If ω is adjoined to the rational numbers, then the associated ring of algebraic integers is denoted by $Z[\omega]$.The ring $Z[\omega]$ can be regarded as consisting of polynomials in ω of degree 7 with integer coefficients. The elements of $Z[\omega]$ are added and multiplied as polynomials, except that the rule $\omega^8 = -1$ is used in the product to reduce the degree of powers of ω to below 8. For an integer M, $Z[\omega]_M$ is used to denote the elements of $Z[\omega]$ with coefficients between $-\frac{M}{2}$ and $\frac{M}{2}$.

A real number *x* in $Z[\omega]$ can be written in the form

$$x = a_0 + \sqrt{2+\sqrt{2}}\,a_1 + \sqrt{2}\,a_2 + \sqrt{2-\sqrt{2}}\,a_3 \tag{3.1}$$

The ring of all such elements is denoted by $Z[\sqrt{2+\sqrt{2}}\,]$. If $\theta = \sqrt{2+\sqrt{2}}$, then θ is a root of the polynomial $x^4 - 4x^2 + 2$ and the elements of $Z(\theta)$ have a polynomial form, where the relation $\theta^4 = 4\theta^2 - 2$ is used to reduce power of θ above three. The elements of $Z(\theta)$ are used to process separately the real and imaginary part of $Z[\omega]$. In summary, algebraic integers of an extension of degree *n* can be assumed to be of the form

$$a_0\omega_0 + a_1\omega_1 + \ldots + a_{n-1}\omega_{n-1} \tag{3.2}$$

Where $\{\omega_0, \omega_1, \ldots, \omega_{n-1}\}$ is called the algebraic-integer *basis* and the coefficients a_i are integers.

3.3 Past Work of AIQ

The past work is restricted to DAUB4 and DAUB6 wavelet transforms. The AIQ-based DAUB4 and DAUB6 transform proposed by Wahid et al. [15], [16] is studied in the following sections.

3.3.1 AIQ encoding of DAUB4

From Eqn. 2.2, if $z = \sqrt{3}$, all the coefficients can be expressed (scaled by $4\sqrt{2}$) as a first degree polynomial in *z* with integer coefficients, as follows [16]:

$$C_0 = 1 + z \qquad C_1 = 3 + z \qquad C_2 = 3 - z \qquad and \qquad C_3 = 1 - z \tag{3.3}$$

In this case, the polynomial has the form: $f(z) = a_0 + a_1 z$ where a_0, a_1 are integers. The codes for the DAUB4 coefficients are shown in Table 3.1. By manipulating these polynomial representations of the coefficients, instead of the usual approximate binary representations, any errors can be eliminated in the calculations until the final reconstruction step.

Table 3.1 Exact representation of DAUB4 coefficients

Coefficients	a_0	a_1
C_0	1	1
C_1	3	1
C_2	3	-1
C_3	1	-1

The input data of the corresponding pixels are coded with integers and all the processing required is very simple and, most importantly, inherently parallel. Using the same polynomial expansion both forward and inverse mappings can be performed.

3.3.2 AIQ encoding of DAUB6

The polynomial expansion of DAUB6 coefficients can be generalized into polynomials of two variables. This gives the advantages in terms of choosing the optimal form such that the equivalent representation scheme is as sparse as possible, and, having several different combinations of applicable pairs of parameters, provides considerable flexibility of choice. From an architectural point of view the final reconstruction step can be accomplished by making use of systolic architectures for polynomial evaluations.

If $z_1 = \sqrt{10}$ and $z_2 = \sqrt{5 + 2\sqrt{10}}$ and considering the 2-D polynomial expansion:

$$f(z_1, z_2) = \sum_{i=0}^{K} \sum_{j=0}^{L} a_{ij} z_1^i z_2^j \tag{3.4}$$

For the DWT and the IDWT implementations, $K = 1$ and $L = 1$ can be chosen to guarantee error-free encoding, and the corresponding coefficients, a_{ij}, are encoded in the form $\begin{bmatrix} a_{00} & a_{10} \\ a_{01} & a_{11} \end{bmatrix}$.

Therefore all the DAUB6 coefficients can be exactly encoded (scaled by $16\sqrt{2}$) as shown in Eqn. (3.5) and summarized in Table 3.2 [16].

$$C_0 = \begin{bmatrix} 1 & 1 \\ 1 & 0 \end{bmatrix} \qquad C_1 = \begin{bmatrix} 5 & 1 \\ 3 & 0 \end{bmatrix} \qquad C_2 = \begin{bmatrix} 10 & -2 \\ 2 & 0 \end{bmatrix}$$

$$C_3 = \begin{bmatrix} 10 & -2 \\ -2 & 0 \end{bmatrix} \qquad C_4 = \begin{bmatrix} 5 & 1 \\ -3 & 0 \end{bmatrix} \qquad C_5 = \begin{bmatrix} 1 & 1 \\ -1 & 0 \end{bmatrix} \tag{3.5}$$

Table 3.2 Exact representation of DAUB6 coefficients

Coefficients	a_{00}	a_{10}	a_{01}	a_{11}
C_0	1	1	1	0
C_1	5	1	3	0
C_2	10	-2	2	0
C_3	10	-2	-2	0
C_4	5	1	-3	0
C_5	1	1	-1	0

3.3.3 Final Reconstruction Step (FRS)

For the final reconstruction, Horner's rule is used [31]. For the computation of DWT or IDWT, the integer part of the result and the most significant bit of the fractional part need to be recovered, in order to allow correct rounding. Since the final result is in an error-free format, the precision to guarantee sufficient accuracy can be easily estimated. As an example, if the input and output data are to be represented within 8-bits per pixel (bpp), then the representation of z as:

$$z = \sqrt{3} \approx 10.0\bar{1}00\bar{1} = 2 - 2^{-2} - 2^{-8} \text{ (DAUB4)} \tag{3.6}$$

$$z_1 = \sqrt{10} \approx 11.001010 = 3 + 2^{-3} + 2^{-5} \text{ (DAUB6)} \tag{3.7}$$

$$z_2 = \sqrt{5 + 2\sqrt{10}} \approx 100.\bar{1}0\bar{1}00 = 4 - 2^{-1} - 2^{-3} \text{ (DAUB6)} \tag{3.8}$$

is sufficient. The signed-digit encoding errors for 8-bit word-lengths are 0.0913%, 0.19% and 0.0569% for Z, Z_1 and Z_2, respectively [16].

3.3.4 DAUB4 Architecture

The DAUB4 architecture has three parallel channels through which data flows independently and also very simple scheduling is required. No quantization errors are incurred in the main part of the algorithm, only in the final reconstruction step (where z is substituted) and which uses one multiplier. The substitution precision can be chosen in such a way as to get the best reconstruction. Since z is a fixed value in the FRS, general multiplier uses can be avoided. A total of 16 adders are needed, and several registers are also required to hold the intermediate partial results. One of the most important aspects of the DWT architecture is its potential for real-time operations. The proposed pipelined architecture computes *N* coefficients in *N* clock cycles and achieves real-time operation through pipelining. The architecture has a latency of 5Ta (Ta = latency for addition operation). In Figure 3.1, L_0 is the low pass filter output and H_0 is the high pass filter output.

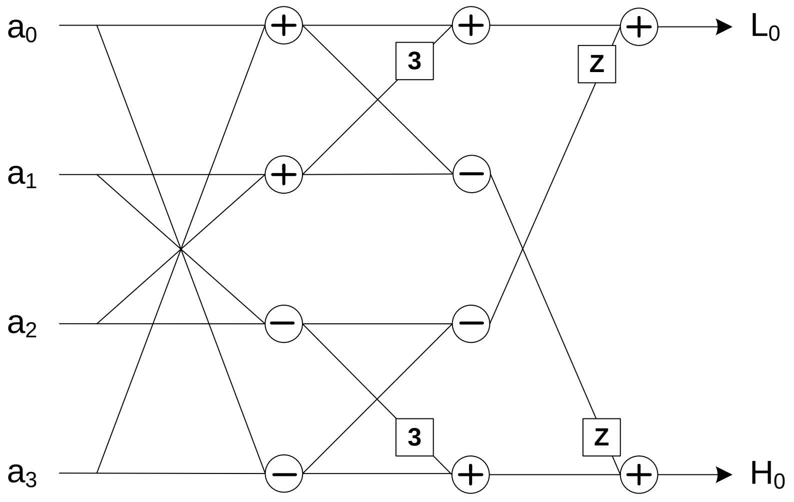


Figure 3.1 AlQ-based DAUB4 filter architecture

3.3.5 DAUB6 Architecture

A total of 42 adders are required to implement the DAUB6 architecture and is shown in Figure 3.2. The architecture has a latency of 6Ta.

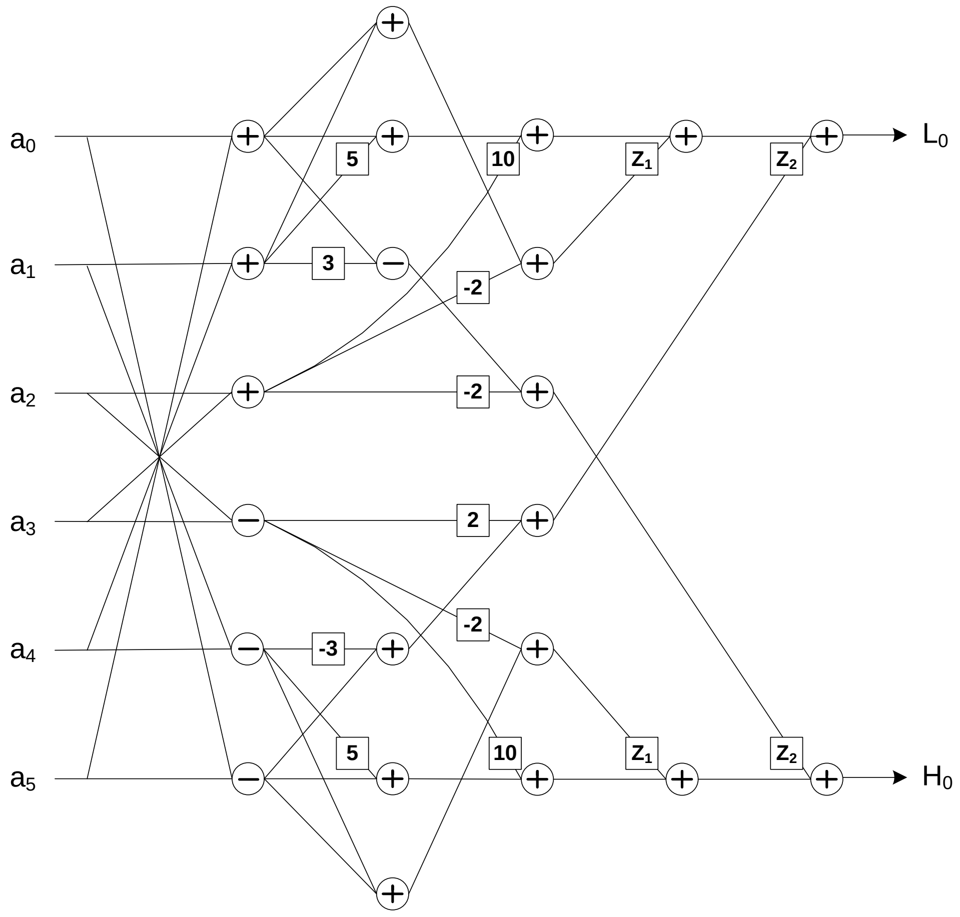


Figure 3.2 AIQ-based DAUB6 filter architecture

3.4 Summary

In this chapter, encoding and architecture of one-level AIQ-based DWT is discussed. The one-level AIQ implementation is simple, multiplication-free and inherently parallel. In the next chapter, we will explore the folded features of the AIQ mapping.

Chapter 4
Folded AIQ Mapping

4.1 Introduction

This chapter presents an introduction to the folded AIQ mapping scheme. Here we propose a new two-level encoding, called folded mapping, which combines the error-free AIQ scheme and matrix decomposition techniques to compute the DAUB4 and DAUB6 wavelet coefficients. This new scheme enables resource sharing and yields lower hardware cost and power consumption, and higher process throughput. The AIQ encoded forward basis transformation matrices are first decomposed into multiple sub-matrices; the common structures of the sub-matrices are identified and later shared for implementation.

In order to eliminate the round-off error for the DWT implementations, AIQ technique is proposed at Wahid et al. [15, 16] to compute the DAUB4 and DAUB6 filter coefficients, where the irrational transform basis functions are mapped with integers, resulting in very efficient computation. However, in those cases a direct (one-level) AIQ mapping is utilized which involves large number of internal computing elements that results in high cost of hardware resources and silicon area and low throughput. The folded scheme adds a two-level AIQ mapping that reduces the number of algebraic integers in the polynomial representation compared to its predecessor. Therefore, compared to existing designs, the proposed scheme reduces significantly the hardware cost, critical path delay and power consumption with a higher throughput rate.

Later, we proposed a one-level AIQ scheme to implement DAUB8 transform which performs better than the fixed-point implementation. The use of conventional fixed-point binary (or any other weighted) representation for implementing Daubechies wavelets, introduces round-off or approximation errors at the very beginning of the process due to the lack of exact representation of the irrational numbers that form the coefficient basis. These errors tend to increase as the calculations progress through the architecture, degrading the quality of the image reconstruction. Here, our aim is to eliminate these errors until we need to convert from the AIQ representation. The AIQ technique eliminates the requirements to approximate the eight transformation matrix elements. Rather, by using one-level AIQ, it is possible to obtain considerable improvement in image reconstruction accuracy with mapping four new coefficients; through reducing error-introducing calculation steps.

4.2 Folded AIQ mapping

Folded AIQ mapping is a new two-level folded mapping technique, an improved version of the AIQ. The scheme is developed on the factorization and decomposition of the transform coefficients that exploits the symmetrical and wrapping structure of the matrices. The technique is very efficient in DSP applications having complex numbers or the coefficients that are in the conjugate form.

4.2.1 Folded AIQ mapping of DAUB4

The four DAUB4 transform coefficients ($C_0,\ldots,C_3$) are expressed below using one-level AIQ mapping (where $Z = \sqrt{3}$ and the scaling factor is $4\sqrt{2}$):

$$C_0 = 1 + Z;\ C_1 = 3 + Z;\ C_2 = 3 - Z;\ C_3 = 1 - Z \tag{4.1}$$

Considering the periodicity of the coefficients in Eqn. (2.3), we decompose the matrix into four sub-matrices as shown below in Eqn. (4.2):

$$\psi_4(C) = \left[\varphi_0(4) \quad \varphi_1(4) \quad \varphi_2(4) \quad \varphi_3(4)\right]^T \tag{4.2}$$

Where, $\varphi_0(4) = \begin{bmatrix} G & \\ H & I_0 \end{bmatrix}$, $\varphi_k(4) = [\varphi_0(4)]_j$, $k = \{0,1,2,3\}$, $j = 2k$ is the column number, $G = [C_0 \quad C_1 \quad C_2 \quad C_3]$ and $H = [C_3 \ -C_2 \quad C_1 \ -C_0]$ are the low-pass (smoothing) and the high pass (non-smoothing) filters, respectively, and I_0 is a 2x4 null matrix.

After plugging Eqn. (4.1) into $\varphi_0(4)$ and decomposing the matrix, we find the new expression as shown below in Eqn. (4.3):

$$\varphi_0(4) = [F_0 \quad I_0] + 3[F_0 \quad I_0] + Z\left(\left[[F_0 \quad I_0] - [F_1 \quad I_0]\right]\right) \tag{4.3}$$

Where, $F_0 = \begin{bmatrix} 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & -1 \end{bmatrix}$, and $F_1 = \begin{bmatrix} 0 & 1 & 1 & 0 \\ 0 & -1 & 1 & 0 \end{bmatrix}$.

This enables folding of the input data. The forward filter can now be implemented using two independent and parallel datapaths. We will use Eqn. (4.3) for the hardware implementation.

4.2.2 Folded AIQ mapping of DAUB6

We follow similar procedure for the DAUB6 wavelet. Like DAUB4, considering the periodicity of the coefficients in Eqn. (2.5), we decompose the matrix into four sub-matrices as shown below in Eqn. (4.4):

$$\psi_6(C) = \left[\varphi_0(6) \quad \varphi_1(6) \quad \varphi_2(6) \quad \varphi_3(6)\right]^T \tag{4.4}$$

Where, $\varphi_0(6) = \begin{bmatrix} G & \\ H & I_0 \end{bmatrix}$, $\varphi_k(6) = [\varphi_0(6)]_j$, $k = \{0,1,2,3\}$, $j = 2k$ is the column number, $G = [C_0 \quad C_1 \quad C_2 \quad C_3 \quad C_4 \quad C_5]$ and $H = [C_5 \quad -C_4 \quad C_3 \quad -C_2 \quad C_1 \quad -C_0]$ are the low-pass (smoothing) and the high-pass (non-smoothing) filters, respectively, and I_0 is a 4x2 null matrix.

The coefficients and their encoded forms (using one-level AIQ) are given below (where, $Z_1 = \sqrt{10}$, $Z_2 = \sqrt{5 + 2\sqrt{10}}$):

$$\begin{bmatrix} C_0 \\ C_1 \\ C_2 \\ C_3 \\ C_4 \\ C_5 \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 \\ 5 & 1 & 3 \\ 10 & -2 & 2 \\ 10 & -2 & -2 \\ 5 & 1 & -3 \\ 1 & 1 & -1 \end{bmatrix} \begin{bmatrix} 1 \\ Z_1 \\ Z_2 \end{bmatrix} \tag{4.5}$$

Now, by applying similar decomposition technique as presented earlier, the folded expressions for DAUB6 can be found as expressed below in Eqn. (4.6):

$$\varphi_0(6) = \left[F_0^{'} \quad I_0'\right] + 5\left[F_1^{'} \quad I_0'\right] + 10\left[F_2^{'} \quad I_0'\right]$$

$$+ Z_1\left(\left[F_0^{'} \quad I_0'\right] + \left[F_1^{'} \quad I_0'\right] - 2\left[F_2^{'} \quad I_0'\right]\right) \tag{4.6}$$

$$- Z_2\left(\left[F_0^{'} \quad I_0'\right] - 3\left[F_1^{'} \quad I_0'\right] + 2\left[F_2^{'} \quad I_0'\right]\right)$$

Where, $F_0^{'} = \begin{bmatrix} 1 & & 1 \\ & I_0 & \\ 1 & & -1 \end{bmatrix}$, $F_1^{'} = \begin{bmatrix} 0 & 1 & & 1 & 0 \\ & & I_0' & & \\ 0 & -1 & & 1 & 0 \end{bmatrix}$, $F_2^{'} = \begin{bmatrix} & 1 & 1 & \\ I_0' & & & I_0' \\ & 1 & -1 & \end{bmatrix}$, and I_0' is a 2x2 null matrix. Like Eqn. (4.3), Eqn. (4.6) allows the folding of the input data, and the forward filter can be implemented using three independent and parallel datapaths. We will use Eqn. (4.6) for the hardware implementation which is described in the following section.

4.2.3 Architecture of DAUB4 and DAUB6

The signal flow graphs of the DAUB4 and DAUB6 filter banks are translated from Eqn. (4.3) and (4.6), and shown in Figure 4.2 and 4.3 respectively, where L_0 is the low-pass and H_0 is the high-pass coefficient. A control pin ($a/\overline{s}$) is used to toggle the two operations, addition/subtraction in alternate clock cycles.

The folded scheme adds a two-level AIQ mapping that reduces the number of AIQ operations in the polynomial representation compared to its predecessor. As a result, the number of adders required to perform DAUB4 is just 9, compared to 16 in the direct mapping [16] (a saving of 44%).

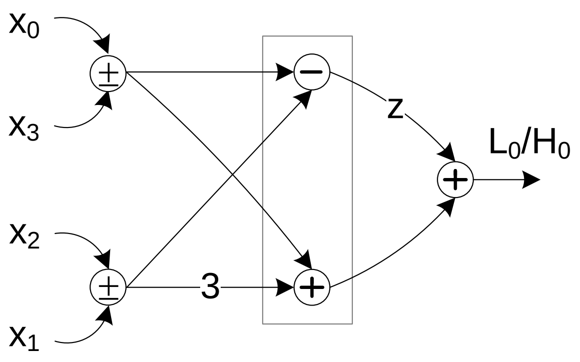


Figure 4.1 Folded AIQ-based DAUB4 filter architecture

In the case of DAUB6, the saving is even higher (57%) as the proposed scheme requires only 18 adders compared to 42 in the one-level design [16].

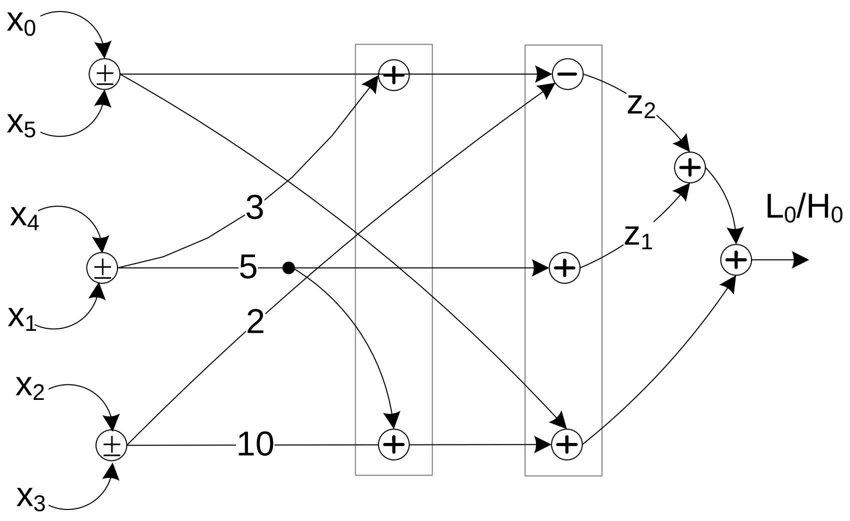


Figure 4.2 Folded AIQ-based DAUB6 filter architecture

Table 4.1 Comparison between 1-level AIQ and folded AIQ

Coefficients	Algorithm	No. of Adders
DAUB4	1-level AIQ	16
	Folded AIQ	9
DAUB6	1-level AIQ	42
	Folded AIQ	18

4.3 AIQ encoding of DAUB8

In this section, we will discuss the fixed-point (FP), and AIQ mapping scheme of DAUB8.

4.3.1 Fixed Point Implementation

Considering the periodicity of the coefficients in Eqn. (2.7), we decompose the matrix into four sub-matrices as shown below in Eqn. (4.7):

$$\psi_8(C) = \left[\varphi_0(8) \quad \varphi_1(8) \quad \varphi_2(8) \quad \varphi_3(8)\right]^T \tag{4.7}$$

Where, $\varphi_0(8) = \begin{bmatrix} G \\ H \end{bmatrix}$, $\varphi_k(8) = [\varphi_0(8)]_j$, $k = \{0,1,2,3\}$, $j = 2k$ is the column number, $G = [C_0 \quad C_1 \quad C_2 \quad C_3 \quad C_4 \quad C_5 \quad C_6 \quad C_7]$ and $H = [C_7 \quad -C_6 \quad C_5 \quad -C_4 \quad C_3 \quad -C_2 \quad C_1 \quad -C_0]$ are the low-pass (smoothing) and the high-pass (non-smoothing) filters, respectively.

Low_filter_output, $L_0 = \sum_{i=0}^{7} x_i C_i$, High_filter_output, $H_0 = \sum_{i=0}^{7} (-1)^i x_i C_{7-i}$

4.3.2 AIQ mapping of DAUB8

The eight DAUB8 transform coefficients $(C_0, \dots \dots \dots, C_7)$ can be encoded using 4 real coefficients as shown in Eqn.(4.8) [27] and summarized in Table 4.2.

$$(C_0, C_1, C_2, C_3, C_4, C_5, C_6, C_7) = \frac{1}{32\sqrt{2}}(1, 4, 6, 4, 1) * (a_0, a_1, a_2, a_3) \tag{4.8}$$

Low and high filter outputs using one-level AIQ mapping are

Low_filter_output, $L_0 = \sum_{i=0}^{3} (x_i + 4x_{i+1} + 6x_{i+2} + 4x_{i+3} + x_{i+4}) * a_i$

High_filter_output, $H_0 = \sum_{i=0}^{3} (x_i - 4x_{i+1} + 6x_{i+2} - 4x_{i+3} + x_{i+4}) * a_{3-i}$

Where, $a_0 = \alpha$, $a_1 = 2 - \sqrt{140} + \frac{5}{\alpha}$, $a_2 = 2 + \sqrt{140} - \alpha$, $a_3 = -\frac{5}{\alpha}$,

Here, $\alpha \cong 10.4257$ is the exact solutions of Daubechies 8 orthonormal scaling coefficients.

In Table 4.2, the polynomial has the form: $f(z) = a_0 z_0 + a_1 z_1 + a_2 z_2 + a_3 z_3$ where a_0, a_1, a_2, a_3 are integers.

Table 4.2 Exact representation of DAUB8 coefficients

Coefficients	a_0	a_1	a_2	a_3
C_0	1	0	0	0
C_1	4	1	0	0
C_2	6	4	1	0
C_3	4	6	4	1
C_4	1	4	6	4
C_5	0	1	4	6
C_6	0	0	1	4
C_7	0	0	0	1

4.3.3 AIQ-based Architecture

The signal flow graphs of FP and AIQ are shown in Figure 4.4 and 4.5. The addition/subtraction operation is achieved simply by the adder/subtractor circuit previously shown in Figure 4.1. A control pin ($a/\overline{s}$) is used to toggle the addition and subtraction operations which are performed at even and odd clock pulses (CPs) respectively.

Fixed point architecture is implemented with 8 finite precision coefficients $(C_0, C_1, C_2, C_3, C_4, C_5, C_6, C_7)$ whereas AIQ is implemented with 4 finite precision coefficients (a_0, a_1, a_2, a_3). All the coefficients or multipliers are constant in nature, and hence can be computed using sequential addition and shift operations only.

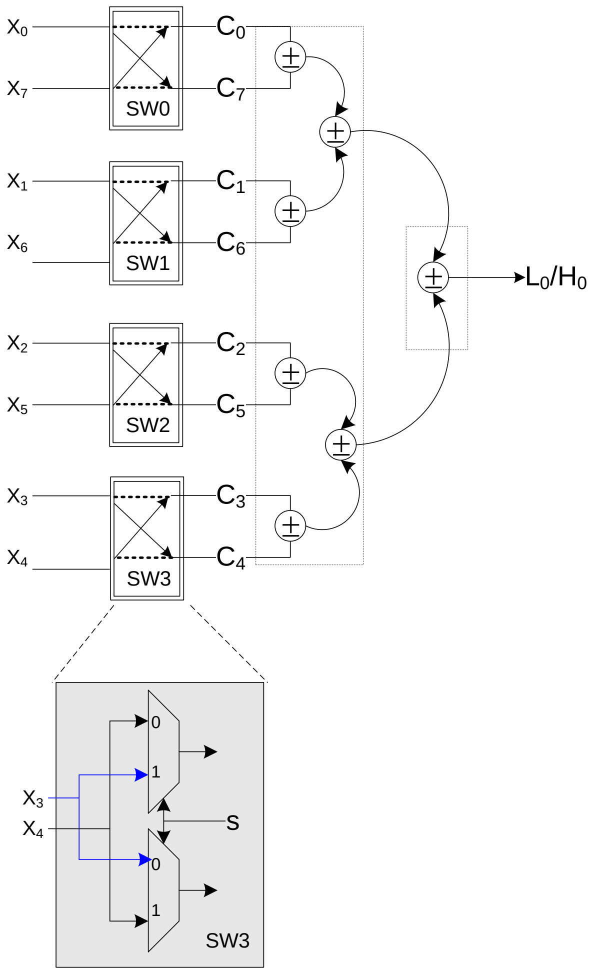


Figure 4.3 Signal flow graph of FP-based DAUB8

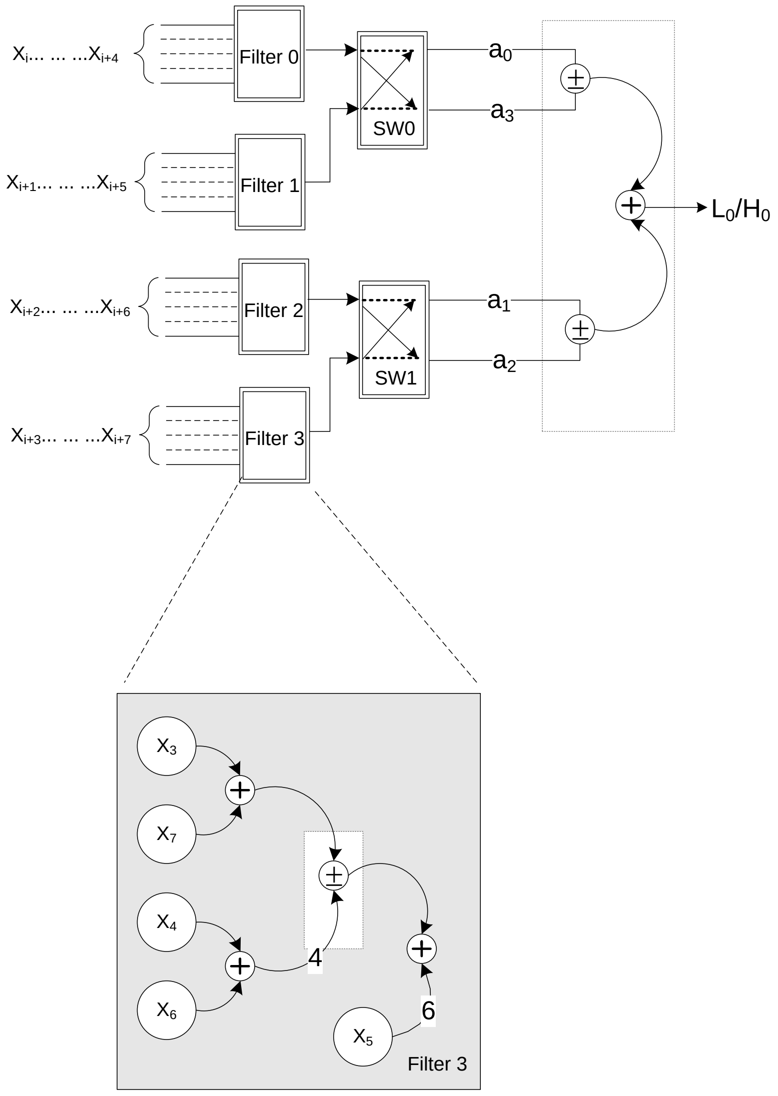


Figure 4.4 Signal flow graph of AIQ-based DAUB8

4.4 Summary

In this chapter, a new folded AIQ scheme to compute DAUB4 and DAUB6 transform coefficients is discussed. The folded AIQ scheme enables simpler implementation and yields lesser hardware compared to previous one-level AIQ scheme. The last section of this chapter describes the one-level AIQ mapping of the DAUB8 transform.

Chapter 5
Folded AIQ-Based Architecture

5.1 Introduction

In this work, we present an efficient implementation of a shared hardware core to compute two 8-point Daubechies wavelet transforms. The architecture is based on a new two-level folded mapping technique, an improved version of the AIQ. The scheme is developed on the factorization and decomposition of the transform coefficients that exploits the symmetrical and wrapping structure of the matrices. The proposed architecture is parallel, pipelined, and multiplexed. The proposed scheme reduces significantly the hardware cost, critical path delay and power consumption with a higher throughput rate. The design has been prototyped onto FPGA and ASIC level using 0.18um CMOS standard cells.

Later, we apply a first level AIQ scheme to compute the DAUB8 wavelet coefficients. The use of conventional fixed-point (FP) binary representation to implement the irrational transform coefficients introduces round-off error at the beginning of the process. These errors tend to increase as the calculations progress through the architecture, degrading the quality of the image reconstruction. The one-level AIQ technique eliminates the requirements to approximate the eight transformation matrix elements as in FP. Using AIQ, it is possible to obtain considerable improvement in image reconstruction accuracy with mapping four new coefficients.

5.2 Folded AIQ-based DAUB4 and Daub6

The proposed dual-core architecture is organized as a linear parallel and double-datapath path pipeline to achieve high throughput, where 2's complement arithmetic has been used to handle the negative numbers. The input is fed to the hardware at a rate of one element per 8-bits per clock cycle through serial in parallel out operation. The transform (DUAB4/DAUB6) low and high pass filter coefficients are outputted serially in double path at 12 bits per coefficient per clock cycle. The parallel outputted data at delay line are folded before feeding them to the filter banks. The transform to be performed is set externally by the user selective *Daub_sel* pin. The processor core consists of four major components: Delay line, Delay line/Shifter, DAUB4 and DAUB6 filter banks, and controller.

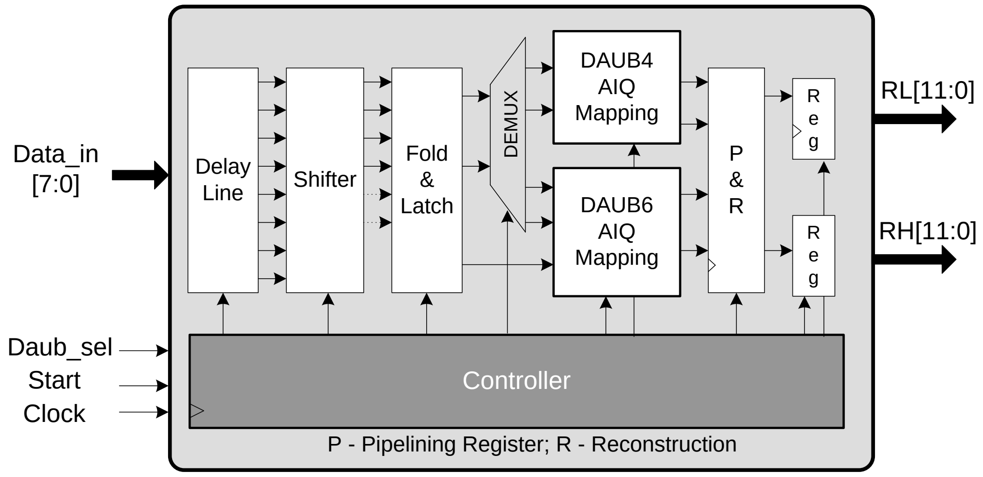


Figure 5.1: Block diagram of the entire system

5.2.1 Delay Line/ Shifter

The input is fed to the hardware at a rate of one sample (8-bit) per clock cycle through serial-in parallel-out operation. This operation is achieved by ping-pong buffer that follows a data shifter which consists of eight processing elements (PE). The architecture is shown in Figure 5.2(a) and 5.2(b). The delay line buffer and the shifter are controlled using *selection* and *enable* pins by the controller. The outputs from the shifter at different clock pulses (CP) are shown in Figure 5.3. Note that, for DAUB4 operation, *x*4 and *x*5 are not passed into the next module.

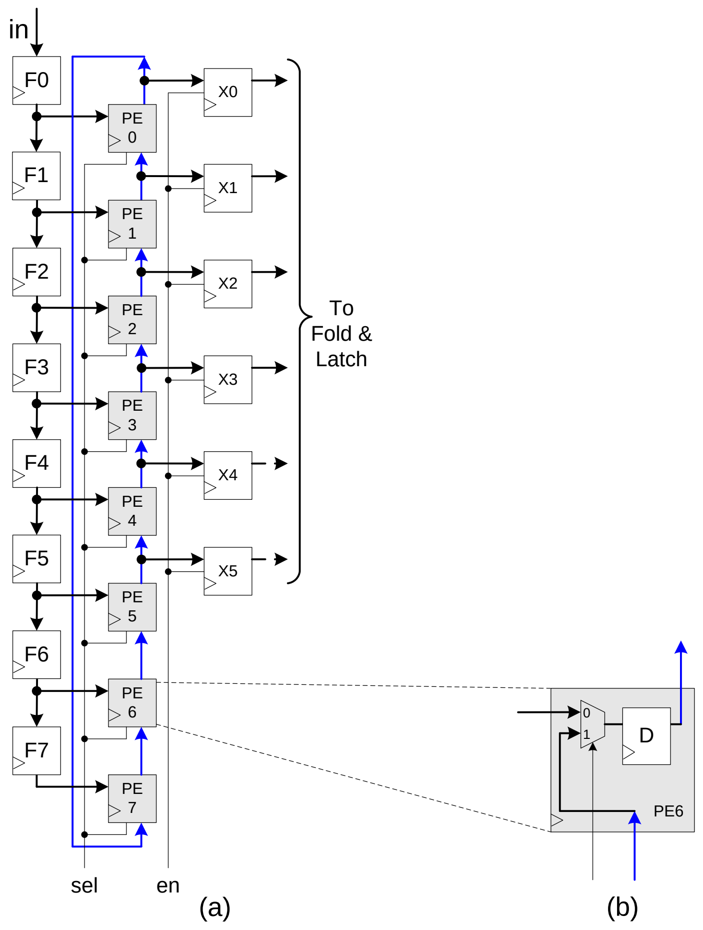


Figure 5.2 (a) Shifter for input scheduling, (b) Processing elements (PE)

N+0	N+2	N+4	N+6
x_0	x_2	x_4	x_6
x_1	x_3	x_5	x_7
x_2	x_4	x_6	x_0
x_3	x_5	x_7	x_1
x_4	x_6	x_0	x_2
x_5	x_7	x_1	x_3

Figure 5.3 Output from shifter at different CP

5.2.2 DAUB4 and DAUB6 filter banks

For both filters, the hardware architecture has been developed using five cascaded stages (as shown in Figure 5.4 and 5.5): Fold & Latch, AIQ mapping, Pipeline Register bank, Reconstruction, and Output.

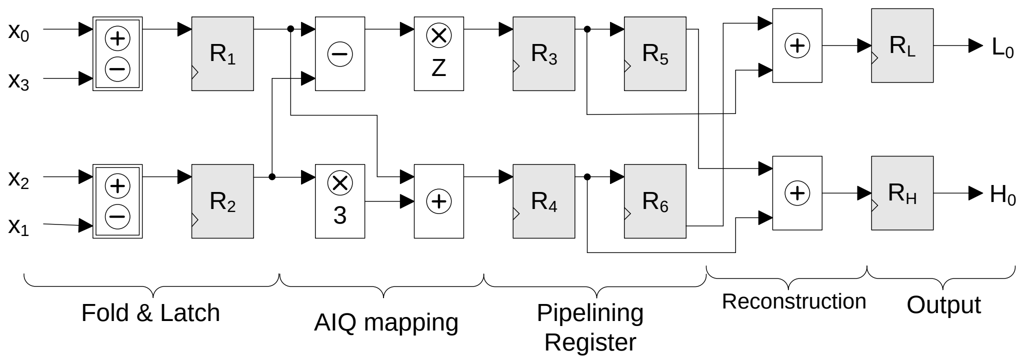


Figure 5.4 DAUB4 filter bank

Using the proposed folded mapping scheme, the number of adders required to perform DAUB4 is just 9, compared to 16 as in the direct mapping [16], which is a savings of 44%. In the case of DAUB6, the savings are even higher (57%) as the proposed two-level scheme requires only 18 adders compared to 42 in the one-level design [16].

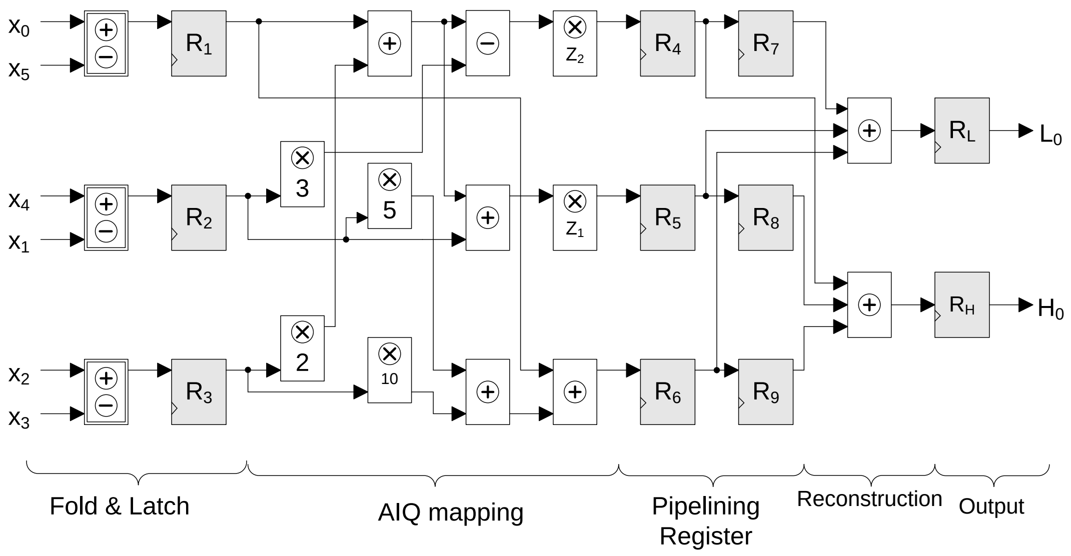


Figure 5.5 DAUB6 filter bank

After the initial fold operation, the intermittent data is latched and then forwarded to the AIQ mapping stage, where the desired shifting and addition operations are performed. The final conversion from algebraic integer to binary also takes place in this stage. Interestingly, all the multipliers are constant in nature, and hence can be computed using sequential addition and shift operations only. Moreover, to reduce the number of addition operations, canonical signed digit (CSD) [33] representation is used.

A precision of 8-bit is used in the multipliers to minimize the hardware and optimize the operation, which is completed in one clock pulse (CP). Internally to the multiplier, all significant bits are retained to guarantee the highest precision of the calculation; however, the multiplier outputs are truncated to discard the fractional part.

5.2.3 Coefficient Multiplier

The forward coefficients of the transforms are constant in nature; hence can be pre-computed and implemented with sequential add/shift operation to make the architecture totally multiplication-free. Also to reduce the number of addition operations, canonical signed digit (CSD) encoding is used in designing the constant multipliers. As an example, the coefficient C1 ($\cos(\pi/16) \approx (0.1111101)_2$) can be implemented with only 2 addition operations as shown in Figure 5.6.

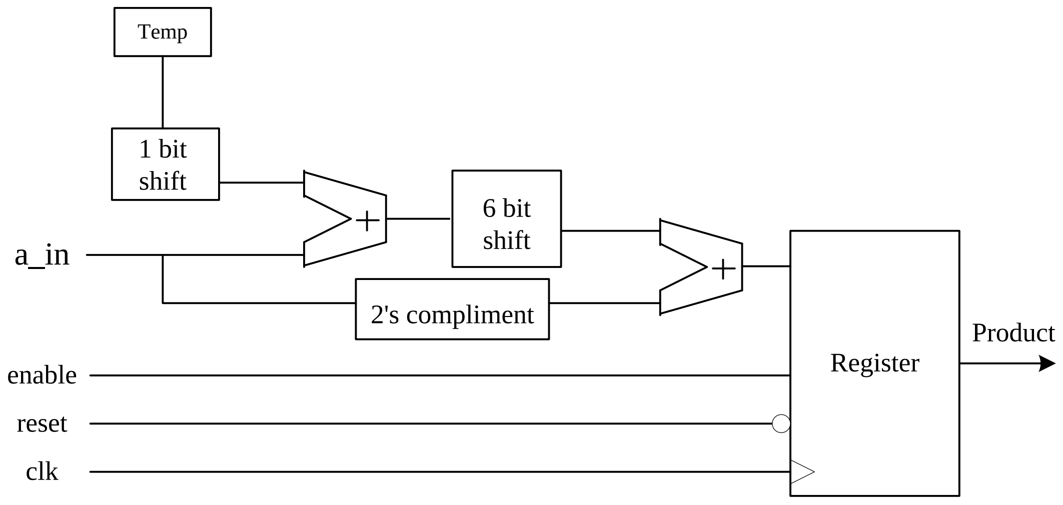


Figure 5.6 Internal circuitry of coefficient multiplier, C1

Note that, the proposed scheme is error-free until this final conversion stage; however, the error introduced at this stage is very small and can be further minimized using higher precision AIQ multipliers. We have performed an error analysis that shows the error incurred for different bit-length of the AIQ multipliers. The error is computed taking a multiplier of 16-bit width as reference. It can be found from the error plot (in Figure 5.7) that the normalized computation error for the proposed DAUB6 multiplier coefficient, z_1 at 8-bit precision is 2.0×10^{-3}. The normalized computation errors for z_2 (DAUB6) and z (DAUB4) at 8-bit precision are 1.8×10^{-3} and 0.5×10^{-3}. The total accuracy of the architecture is reported in performance analysis section.

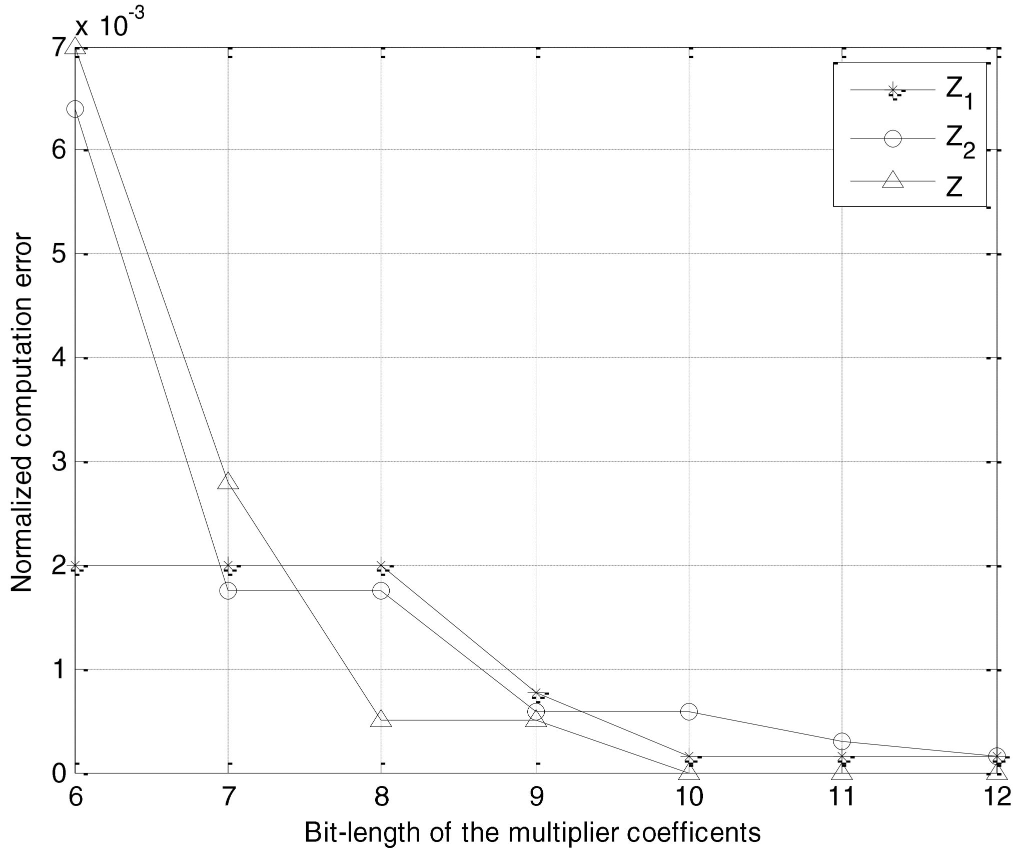


Figure 5.7 Normalized errors incurred for different multiplier precision

After the conversion is completed, the intermittent data are latched into the Pipelining Registers. In order to get the low-pass and high-pass wavelet coefficients simultaneously, a two-level pipelining stage is used. The data are combined in the Reconstruction stage and finally latched into the output registers.

5.2.4 Controller

The controller generates the signals to control various stages, selects datapaths, and indicates the input and output data validity. The internal operations, registers' contents and data validity at different CPs are shown in Figure 5.8 and 5.9. These figures show an operation that starts at the *N*th timing instance. The pipelined implementation allows the mapping to start at the (*N*+1)th CP; at the (*N*+3)th CP, the system outputs the first set of low and high-pass Daubechies coefficients. Thus, we can see that both DAUB4 and DAUB6 take only four clock cycles to complete, and there is no data congestion inside the pipeline, which makes the scheme very suitable for real-time applications. In the next four clock cycles, the input data are reordered accordingly (by the shifter) for further processing.

CP	R_1	R_2	R_3	R_4	R_5	R_6	R_L	R_H	Valid
N+0	X_0+X_3	X_2+X_1	—	—	—	—	—	—	N
N+1	X_0-X_3	X_2-X_1	$[-(X_0+X_3)+(X_2+X_1)].Z$	$(X_0+X_3)+3(X_2+X_1)$	—	—	—	—	N
N+2	X_2+X_5	X_4+X_3	$[-(X_0-X_3)+(X_2-X_1)].Z$	$(X_0-X_3)+3(X_2-X_1)$	$[-(X_0+X_3)+(X_2+X_1)].Z$	$(X_0+X_3)+3(X_2+X_1)$	—	—	N
N+3	X_2-X_5	X_4-X_3	$[-(X_2+X_5)+(X_4+X_3)].Z$	$(X_2+X_5)+3(X_4+X_3)$	$[-(X_0-X_3)+(X_2-X_1)].Z$	$(X_0-X_3)+3(X_2-X_1)$	$L_0=R_3+R_6$	$H_0=R_4+R_5$	Y
N+4	X_4+X_7	X_6+X_5	$[-(X_2-X_5)+(X_4-X_3)].Z$	$(X_2-X_5)+3(X_4-X_3)$	$[-(X_2+X_5)+(X_4+X_3)].Z$	$(X_2+X_5)+3(X_4+X_3)$	$L_0=R_3+R_6$	$H_0=R_4+R_5$	N
N+5	X_4-X_7	X_6-X_5	$[-(X_4+X_7)+(X_6+X_5)].Z$	$(X_4+X_7)+3(X_6+X_5)$	$[-(X_2-X_5)+(X_4-X_3)].Z$	$(X_2-X_5)+3(X_4-X_3)$	$L_1=R_3+R_6$	$H_1=R_4+R_5$	Y

Figure 5.8 Timing diagram of folded AIQ-based DAUB4

CP	R_1	R_2	R_3	R_4	R_5	R_6	R_7	R_8	R_9	R_L	R_H	Valid
N+0	X_0+X_5	X_4+X_1	X_2+X_3	—	—	—	—	—	—	—	—	N
N+1	X_0-X_5	X_4-X_1	X_2-X_3	$[(X_0+X_5)-3(X_4+X_1)+2(X_2+X_3)].Z_2$	$[(X_0+X_5)+(X_4+X_1)-2(X_2+X_3).Z_1$	$(X_0+X_5)+5(X_4+X_1)-10(X_2+X_3)$	—	—	—	—	—	N
N+2	X_2+X_7	X_6+X_3	X_4+X_5	$[(X_0-X_5)-3(X_4-X_1)+2(X_2-X_3)].Z_2$	$[(X_0-X_5)+(X_4-X_1)-2(X_2-X_3)].Z_1$	$(X_0-X_5)+5(X_{4-}X_1)-10(X_{2-}X_3)$	$[(X_0+X_5)-3(X_4+X_1)+2(X_2+X_3)].Z_2$	$[(X_0+X_5)+(X_4+X_1)-2(X_2+X_3).Z_1$	$(X_0+X_5)+5(X_4+X_1)-10(X_2+X_3)$	—	—	N
N+3	X_2-X_7	X_6-X_3	X_4-X_5	$[(X_2+X_7)-3(X_6+X_3)+2(X_4+X_5)].Z_2$	$[(X_2+X_7)+(X_6+X_3)-2(X_4+X_5).Z_1$	$(X_2+X_7)+5(X_6+X_3)-10(X_4+X_5)$	$[(X_0-X_5)-3(X_4-X_1)+2(X_2-X_3)].Z_2$	$[(X_0-X_5)+(X_4-X_1)-2(X_2-X_3)].Z_1$	$(X_0-X_5)+5(X_{4-}X_1)-10(X_{2-}X_3)$	$L_0=R_5+R_6+R_7$	$H_0=R_4+R_8+R_9$	Y

Figure 5.9 Timing diagram of folded AIQ-based DAUB6

The transformed low- and high-pass coefficients are both outputted serially at the same time at a rate of one sample (12-bits/coefficient) per clock cycle.

5.3 Hardware Synthesis

The architecture is coded in Verilog 2001 [34], [35] and prototyped first onto Xilinx VirtexE [36] FPGA to assess the performance. The architecture is later implemented using CMOS 0.18μm standard cell library and fabricated from Canadian Micro Corporation (CMC) using CMOS 0.18μm TSMC technology.

5.3.1 FPGA Synthesis

Field Programmable Gate Array (FPGA) is used to synthesize the architecture in this thesis. FPGAs are programmable logic devices made up of routing channels and arrays of logic cells. FPGAs can be used to implement any logical function that an ASIC could perform with an added advantage that they are reprogrammable. Therefore, new features and modifications can be easily added and they can be used as a tool for comparing different architectures before implementing the final design in ASIC. Currently, Xilinx

Corporation and Altera Corporation are the leading vendors of programmable devices. The architecture of FPGA is vendor specific.

The typical design cycle for FPGAs using Computer Aided Design (CAD) tools is shown in Figure in 5.10. The design is first entered using text entry or graphic entry. Functionality of the design is extracted in next stage. Then the design is targeted on a selected FPGA device and its timing is extracted. In the final stage the actual hardware device is programmed. Appropriate verification is done at every stage to check the working of the design. For design entry, text is popular as it allows more control over the design compared to graphic design entry.

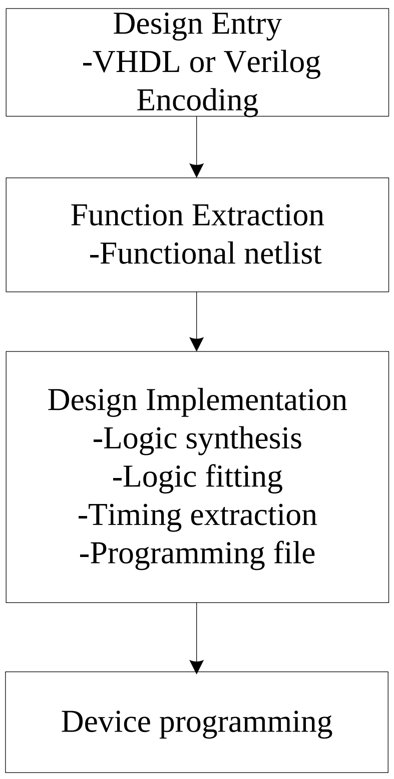


Figure 5.10 CAD design cycle

Table 5.1 presents the comparison of the synthesized results with the previous DAUB-4 and 6 fixed-point (FP) and the one-level AIQ designs [16]. It can be seen from the table that the proposed folded scheme costs lesser hardware resources and has lower critical path delay. The design is synthesized in Xilinx VirtexE (xcv300epq240-8) FPGA; here D4 = DAUB4; D6 = DAUB6; Tm = latency for multiply operation; Ta = latency for addition operation

Table 5.1 FPGA implementation and hardware comparison

Scheme		**Fixed-point [16]**		**One-level AIQ [16]**		**Proposed two-level folded AIQ**			**Overall savings (%)**	
		D4	D6	D4	D6	D4	D6	Dual (D4+D6)	vs. FP	vs. 1-AIQ
FPGA	Datapath	4	6	3	6	2	3	3	70	66
	Adders	32	44	16	42	9	18	27	65	53
	Logic cells	536	728	248	680	106	254	311	75	66
	Registers	422	520	200	494	115	196	360	62	48
	Critical path	Tm+ 2Ta	Tm+ 3Ta	5Ta	6Ta	3Ta	5Ta	5Ta	--	--

5.3.2 VLSI implementation and chip fabrication

The architecture is implemented using CMOS (Complementary Metal–Oxide–Semiconductor) 0.18μm standard cell library and CMOSP18 design kit. The VLSI design (Design run code: 0903CF and Full design ID: ICFSKASH) is later fabricated from CMC using CMOS 0.18μm TSMC technology. VLSI implementation and Chip fabrication preparation is done according to CMC's "Digital IC design flow" [37], which is shown in Figure 5.12.

The micrograph of the fabricated chip is shown in Figure 5.11 and the core features are summarized in Table 5.3. The power consumptions for both DAUB4 and DAUB6 filters are reported in Table 5.2. The maximum power consumption occurs for the DAUB6 (while running at 50MHz with 1.6V supply voltage), and hence, is reported as the chip power. When compared to the previous designs [16] at ASIC level, the proposed two-level design consumes much less silicon area and power (as seen from Table 5.2).

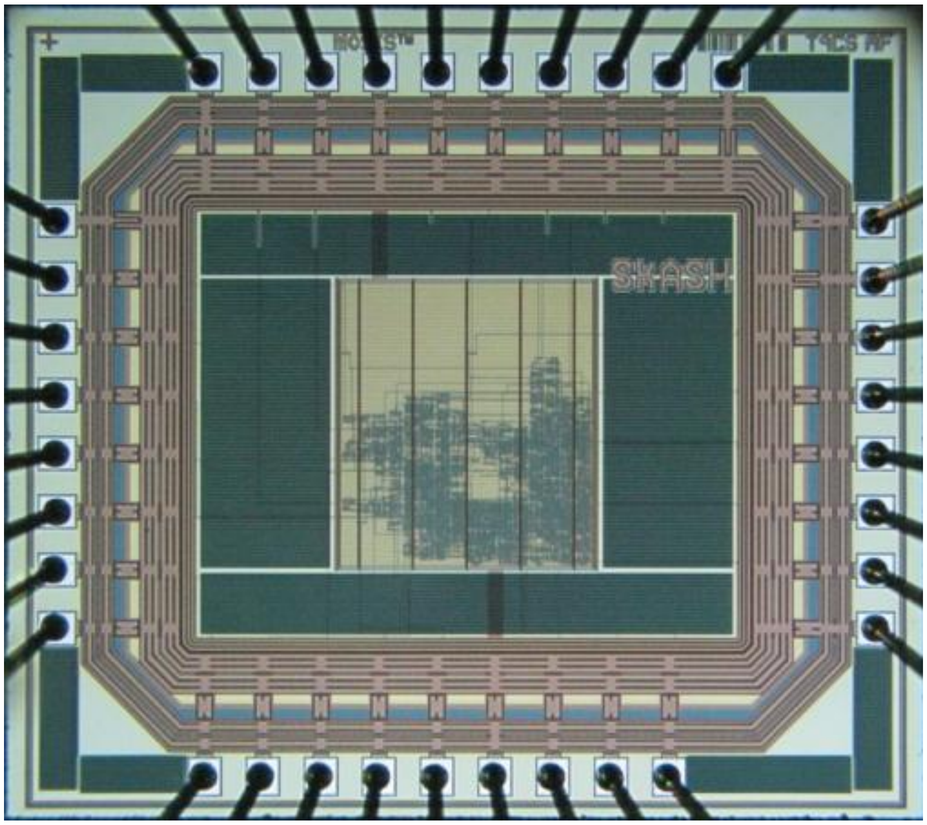


Figure 5.11 Final layout of the folded AIQ-based wavelet transform

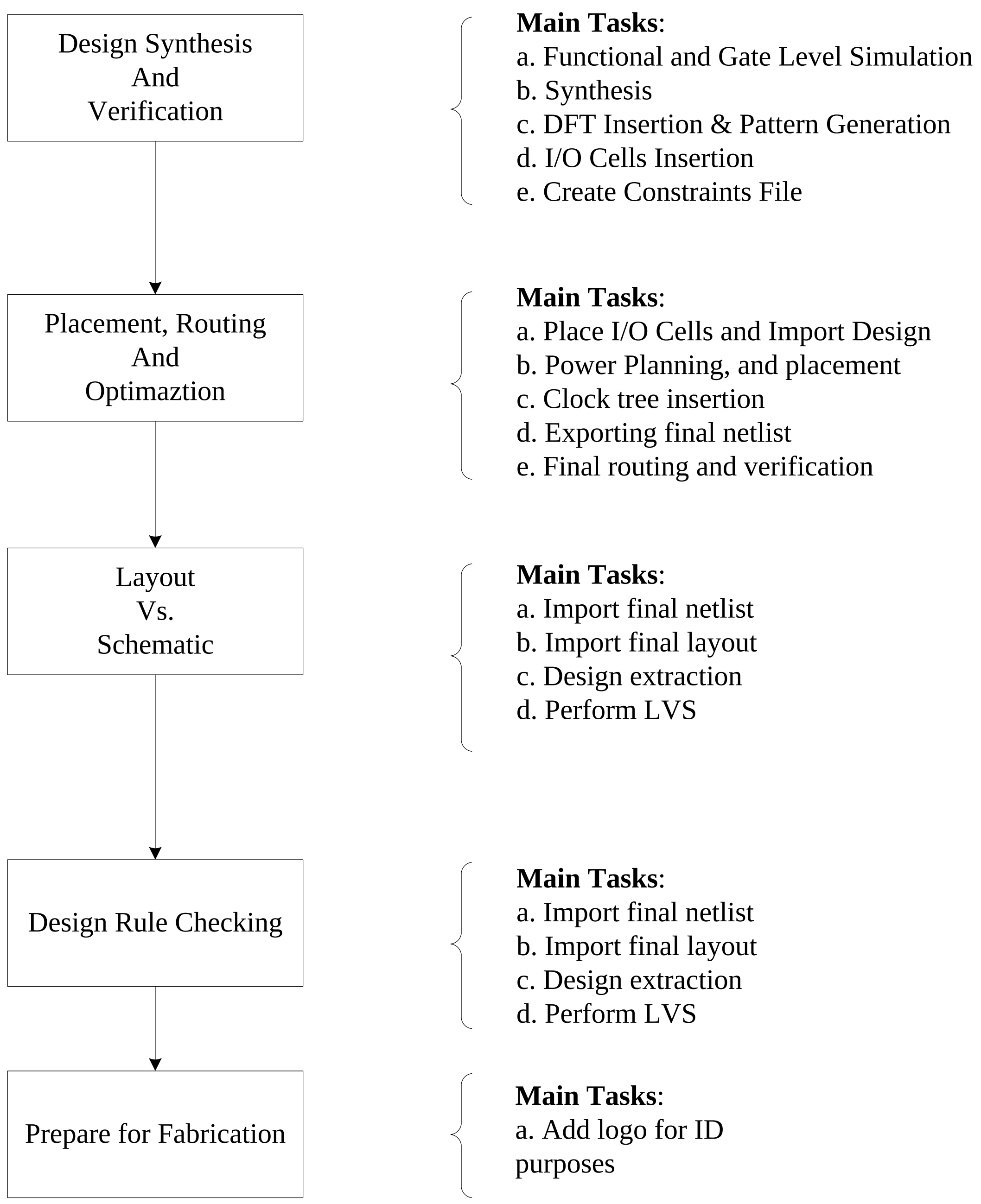


Figure 5.12 VLSI Design Flow [37]

Table 5.2 VLSI implementation and hardware comparison

Scheme		Fixed-point [16]		One-level AIQ [16]		Proposed two-level Folded AIQ			Overall savings (%)	
		D4	D6	D4	D6	D4	D6	Dual (D4+D6)	vs. FP	vs. 1-AIQ
VLSI	Gate count	7,096	9,032	3,934	10,050	3,424	5,048	7,533	53	46
	Power (mW)	16.01	17.41	15.94	22.29	2.46	4.51	4.51	--	--

Table 5.3 Chip specification

Inputs / Outputs	8 bits / 12 bits
Technology	0.18 μm CMOS
Number of gates / cells	7,533 / 1,705
Core / Chip size	0.60 mm x 0.57 mm (core) 1.3 mm x 1.6 mm (chip)
Power consumption	4.51 mW @ 50 MHz
Latency	8 CC
Throughput (per cycle)	2-inputs/output
Maximum frequency	100 MHz

In Table 5.4, we compare our results with other architectures in terms of hardware cost (i.e. number of multipliers, adders, and registers), critical path delay, and throughput rate. From the popularity and applications perspective, we have limited our study to schemes of Duabechies-4 and -6, and 5/3 and 9/7 IWT. As found from Table 5.4 that due to the two-level integer mapping and data folding, the proposed folded AIQ scheme outperforms other designs as it requires no multipliers and lesser registers; the critical path delay is also the least with a high process throughput.

Table 5.4 Hardware comparison of filter

Scheme	Mult	Add	Reg.	Throughput (input/output)
[6] – D4/D6	8/12	6/10	--	--
[12]	4	12	--	--
[13]	0	19	9	1 / 1
[8]	2	4	10	1 / 1
[9]	4	8	22	2 / 1
[10]	4	8	4	2 / 1
[38]	2	4	20	1 / 1
[11]	4	8	28	2 / 1
Prop. D4	0	9	8	2 / 1
Prop. D6	0	18	11	2 / 1

Table 5.5 shows the comparison among different designs at the ASIC level. The technology used is different in some cases; nevertheless, it gives us a rough estimate and a relative position of our design compared to other schemes. The RISC IWT 5/3 and 9/7 bi-orthogonal filters in [12] require multiplication of kernels of large size which makes the control circuitry complicated, but integer nature results in faster operational

frequency. Compared to [12], our proposed design is based on a small kernel size (8x8) with much simpler control signaling yielding the least silicon area and power consumption; however, the final conversion limits the frequency of operation. The presented scheme performs better compared to other reported designs. Note that, the processor core houses two Daubechies wavelet transforms. The dual core chip has been tested with the lab setup as shown in Figure 5.13. In this lab setup, the ring power supply has been set at 3.3 V, and the core power supply has been set at 1.62 V. A fixed test input vector (8'b00011111 at each clock cycle) has been used to measure the core power consumption with a 50 MHz clock frequency. An ammeter has been used to record the current consumption by the core. In this testing, DAUB6 filter consumes the maximum power, 4.51 mW at 50 MHz, which is reported as the core power. DAUB4 consumes 2.46 mW at 50 MHz with the given test input vector.

Table 5.5: Performance comparison in VLSI

Scheme	**Tech. (um)**	**Chip Area (mm^2)**	**Power (mW)**	**Max. Freq. (MHz)**
[7]	0.13	6.5	4.68	100
[14]	0.13	--	12.88	--
[12]	0.18	4.84	197.6	459
[9]	0.18	2.16	102.6	100
Proposed – Dual	0.18	2.08	4.51[1]	100

[1]Measured at 50MHz

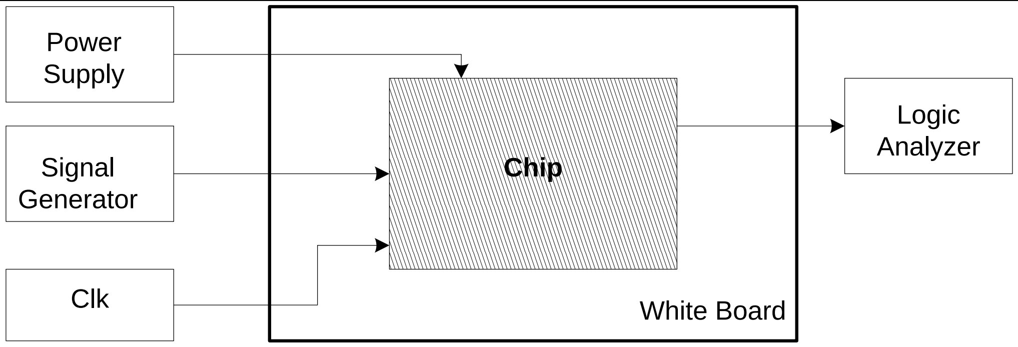


Figure 5.13 Chip testing setup

5.3.3 Performance Analysis

The output of the proposed design has been verified using a benchmark image (Lena), and the results for the DAUB4 and DAUB6 are shown in Table 5.6. In this verification process, Modelsim simulation has been used to process the forward transform and the image has been restored performing inverse transform in Matlab. The reconstructed Lena images are shown in Figure 5.14.

PSNR: A useful measure of the accuracy of the DWT coefficients is the Peak-Signal-to-Noise-Ratio (PSNR) [39]. Here, the signal is represented by the floating-point DWT coefficients and the noise is the difference between the floating-point and finite-precision approximations. In the remainder of this thesis, PSNR is used as a measure of image reconstruction. The PSNR is defined by equation 5.1.

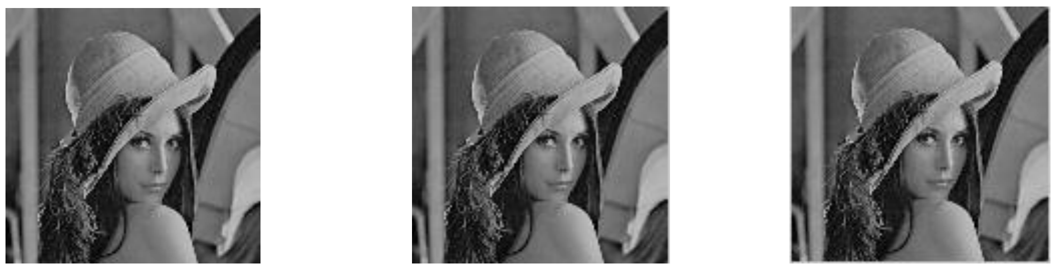
$$PSNR = 10\log\left[\frac{255\times 255}{\frac{1}{WH}\sum_{m-1}^{W}\sum_{n-1}^{H}(c_{m,n}-\tilde{c}_{m,n})}\right] \tag{5.1}$$

Where, W is the width of the frame, H is the height of the frame, $c_{m,n}$ and $\tilde{c}_{m,n}$ are the pixel values of the original and the reconstructed image respectively.

To compute PSNR, we performed forward wavelet transform (using both transforms) on the 128x128 standard “Lena” image, and then computed the inverse transform on the transformed coefficients to recover the original image data.

Table 5.6 PSNR of DAUB4 and DAUB6

Transform	Benchmark Image	PSNR(dB)
DAUB4	Lena	82
DAUB6		85



(a) (b) (c)

Figure 5.14 (a) Original “Lena” image, (b) DAUB4 implementation, and (c) DAUB6 implementation

5.4 DAUB8 Architecture

In this work, we apply both the fixed-point (FP) and one-level AIQ scheme to compute the DAUB8 wavelet coefficients. The AIQ scheme enables a better reconstruction of image compared to fixed-point architecture. Both designs are prototyped onto FPGA with 8, 10, 12-bit precision of the coefficients and compared with their reconstruction errors.

The proposed architecture is organized as parallel double-datapath pipeline to achieve high throughput. 2’s complement arithmetic has been used to handle the negative numbers. The processor core consists of four major components: Delay line / shifter, DAUB8 FP /AIQ filter banks, and controller.

5.4.1 Delay Line / Shifter

The input is fed to the hardware at a rate of one sample (8-bit) per clock cycle through serial-in-parallel-out operation. This operation is achieved by ping-pong buffer that follows a data shifter which consists of eight processing elements (PE). The architecture is shown in Figure 5.15(a) and 5.15(b). The delay line buffer and the shifter are controlled using *selection* and *enable* pins by the controller. The outputs from the shifter at different clock pulses (CP) are shown in Figure 5.16.

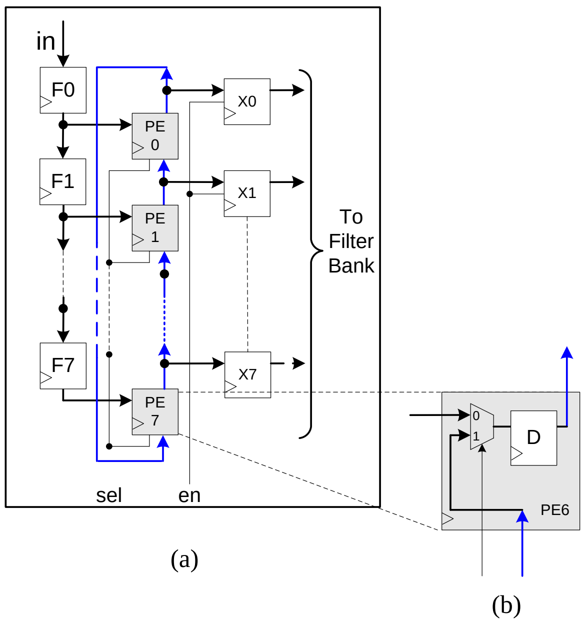


Figure 5.15 Hardware diagrams: (a) Shifter for input scheduling; (b) Processing element (PE);

N+0	N+2	N+4	N+6
x_0	x_2	x_4	x_6
x_1	x_3	x_5	x_7
x_2	x_4	x_6	x_0
x_3	x_5	x_7	x_1
x_4	x_6	x_0	x_2
x_5	x_7	x_1	x_3
x_6	x_0	x_2	x_4
x_7	x_1	x_3	x_5

Figure 5.16 Output from shifter at different CP

5.4.2 FP and AIQ-based DAUB8 Filter Banks

Fixed point architecture is implemented with 8 finite precision coefficients whereas AIQ is implemented with 4 finite precision coefficients. All the coefficients or multipliers are constant in nature, and hence can be computed using sequential addition and shift operations only. Moreover, to reduce the number of addition operations, CSD representation is used in both architectures.

8-bit, 10-bit and 12-bit precision are used in the multipliers. In each of the three cases the multiplication is completed in one clock pulse (CP). Internally to the multiplier, all significant bits are retained to guarantee the highest precision of the calculation; however, the multiplier outputs are truncated to discard the fractional part. The error introduced at this stage is compared in between the two architectures and minimized using higher precision. The architecture of DAUB8 FP filter bank and AIQ filter bank are shown in Figure 5.17 and Figure 5.18.

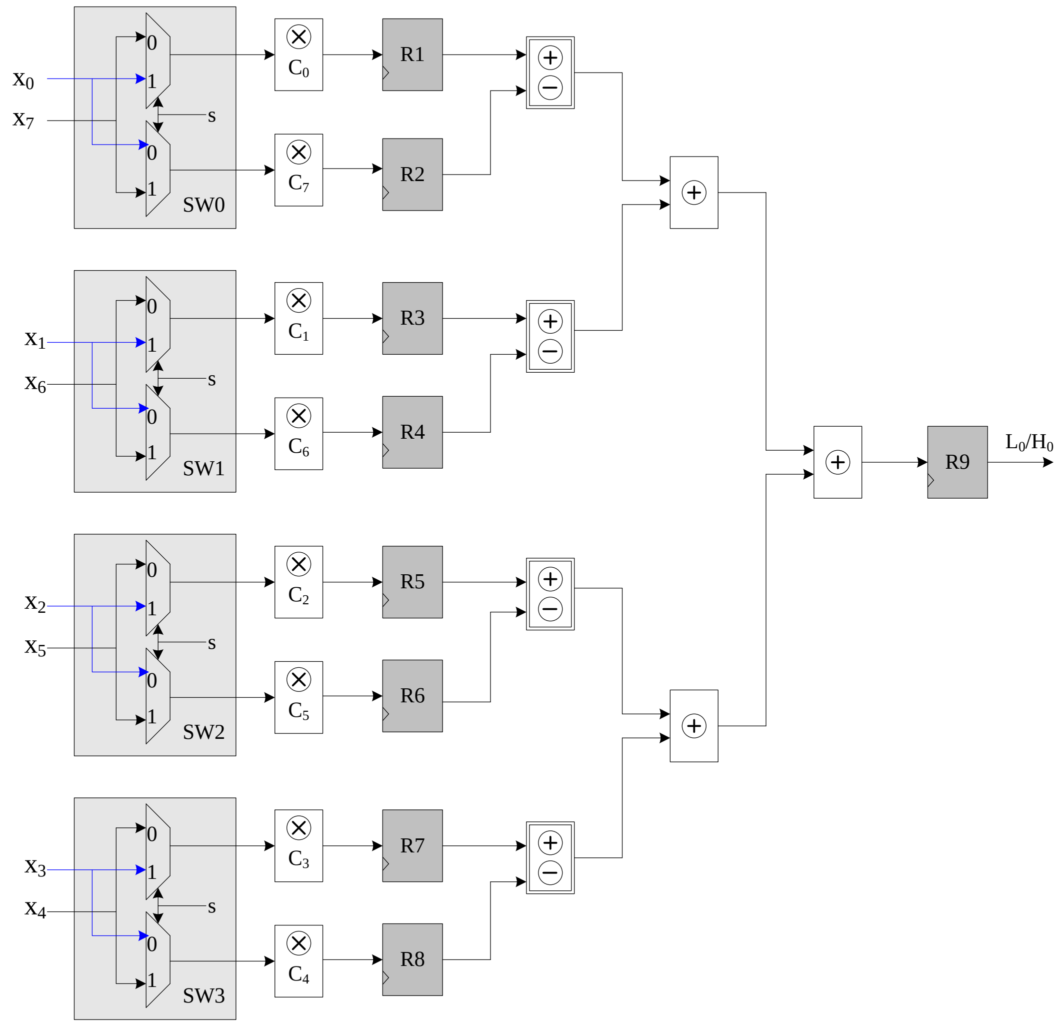
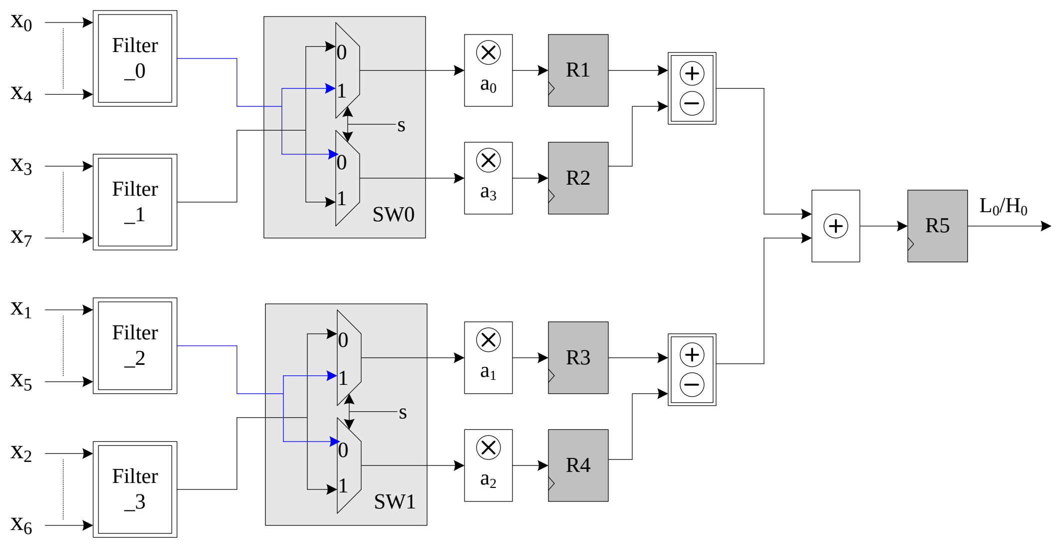
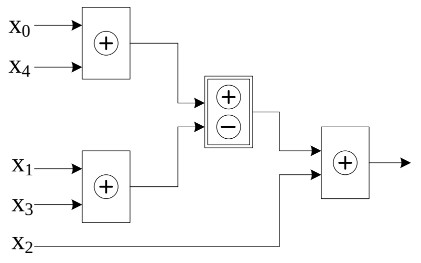


Figure 5.17 DAUB8 FP filter bank



(a)



(b)

Figure 5.18 (a) DAUB8 AIQ filter bank, (b) Filter_0 architecture

5.4.3 Controller

The controller generates the signals to control various stages, selects datapaths, and indicates the input and output data validity. The pipelined implementation allows the mapping to start at the (N+1)th CP; at the (N+3)th CP, the system outputs the first set of low and high-pass Daubechies coefficients. Thus, we can see that both FP and AIQ take only four clock pulses to complete, and there is no data congestion inside the pipeline. In the next four clock pulses, the input data are reordered accordingly (by the shifter) for further processing. The transformed low- and high-pass coefficients are both output serially in alternate CP at a rate of one sample per clock pulse.

5.4.4 Comparisons

Our results are based on the reconstruction of standard 8-bit "Lena" image using fixed point and AIQ scheme. We compare a complete n-bit (n=8, 10, 12) FP calculation for the entire wavelet transform (i.e. no vector quantization) with an AIQ computation. In comparison to FP architecture where it has eight coefficients to map, AIQ uses four coefficients which effectively reduce down the error-introducing steps. The hardware comparison results (in terms of Logic cells, Registers & frequency) are summarised in Table 5.6 where we provide a comparison of the arithmetic hardware complexity for the AIQ implementation against the fixed-point binary (FP) implementation.

Two of the error metrics used to compare the various image compression techniques are the Peak Signal to Noise Ratio (PSNR) and Root Mean Square Error (RMSE) [39]. The RMSE is the cumulative squared root error between the reconstructed and the original image. The mathematical formula for RMSE is:

$$RMSE = \left[\sqrt{\frac{1}{WH} \sum_{m-1}^{W} \sum_{n-1}^{H} (c_{m,n} - \tilde{c}_{m,n})^2} \right] \tag{5.2}$$

Where, W is the width of the frame, H is the height of the frame, $c_{m,n}$ and $\tilde{c}_{m,n}$ are the pixel values of the original and the reconstructed image respectively.

An interesting comparison between the architecture is to select similar or near to similar performance and then compare the hardware consumption to attain that quality. An example of this from Table 5.7 & 5.8 is, to attain a PSNR of 89.5db required number of logic cells in AIQ-based transform is 518; whereas required number of logic cells is 546 to achieve 86 db PSNR in FP architecture.

In Figure 5.20 the image reconstruction of “Lena” image with 8-bit AIQ and 8-bit FP scheme is presented.

Moreover, from Figure 5.19 it is clearly understood the AIQ encoding scheme has better accuracy in image reconstruction compared to fixed point with less hardware consumption; which allows multiplication-free, parallel, and real time hardware implementation. DAUB8 AIQ architecture is very suitable for application like finger print detection [40] and ECG signal denoising [41] where higher precision of image or signal reconstruction is required.

In Table 5.9 we compare our design with other architectures. Compared to [40] and [41] our FP and AIQ architectures consume less hardware. Please note that [40] and [41] reported only low FIR filter implementation results. Studying the mathematical algorithm of DAUB8 from chapter 3 it is obvious High FIR filter will require almost the same number of logic cells for the corresponding architecture. So the hardware consumption in [40, 41] are extended for HIGH FIR filter in Table 5.9 for a rational comparison.

Table 5.7 FPGA implementation & hardware costs between FP and AIQ

Scheme	Fixed-point			Proposed AIQ		
	8-bit	10-bit	12-bit	8-bit	10-bit	12-bit
Logic Cells	490	546	631	518	572	637
Registers	255	255	262	186	186	187
Frequency	95	83	74	71	64	54

Table 5.8 Image quality at different bit precision between FP and AIQ

Quality	Fixed-point			Proposed AIQ		
	8-bit	10-bit	12-bit	8-bit	10-bit	12-bit
RMSE	0.0261	0.0130	0.0067	0.0085	0.0063	0.0061
PSNR	80	86	91.5	89.5	92	92.5

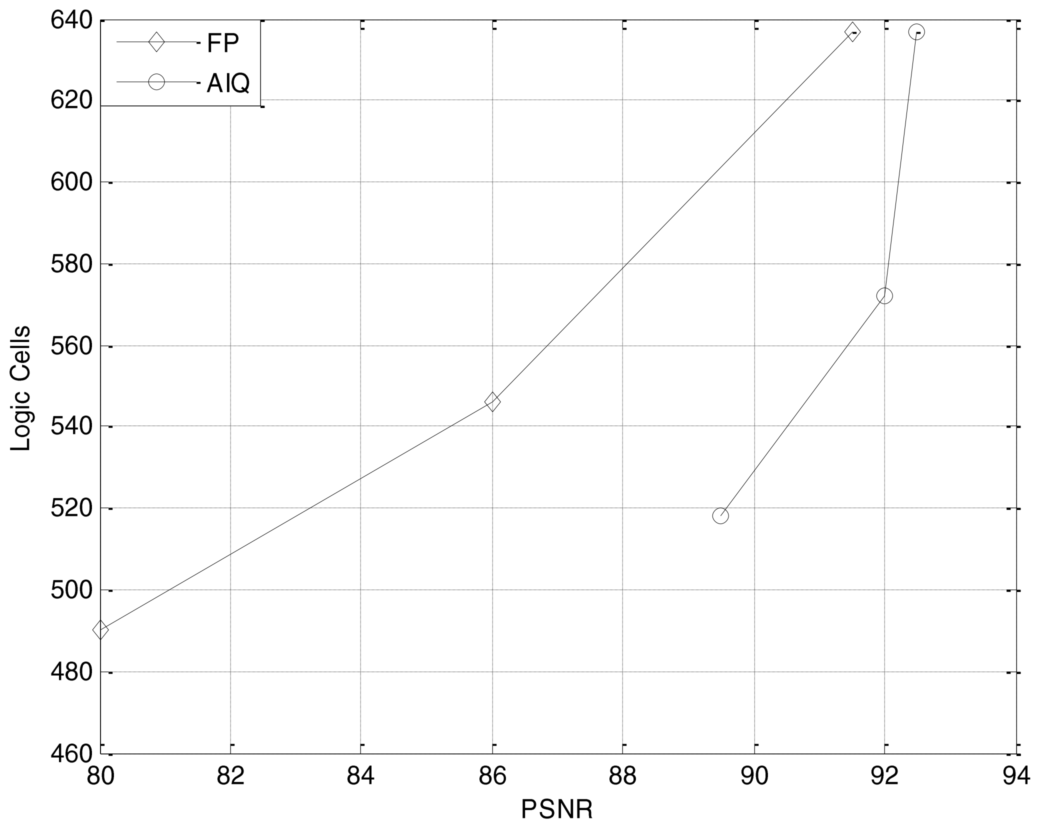


Figure 5.19: Harwdare cost of FP and AIQ at corresponding PSNR

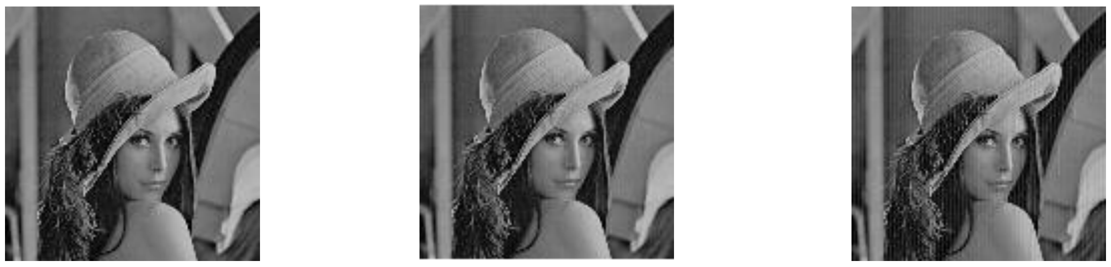


Figure 5.20 (a) original "Lena" image; (b) 8-bit AIQ (PSNR = 89.5 dB); and (c) 8-bit FP (PSNR = 80 dB) implementation

Table 5.9 Comparison of hardware between different architectures

Scheme	Architecture	Logic Cells	Speed (MHz)	Input Word Length	Coefficient Bit Length
[40]	Conventional	1,120	54.3	9	9
	Distributed Arithmetic	748	72.7	9	9
[41]	Hard Router	900	161	9	8
	Benkrid architecture with adder tree	632	159	9	8
Prop.	FP	490	95	8	8
	AIQ	518	71	8	8

5.6 Summary

In this chapter, the implementation of the proposed folded DAUB4 and DAUB6 architecture is discussed. The architecture is implemented using the mathematical expressions and signal flow graphs developed in Chapter 4. The hardware synthesized results are then presented and compared with one-level AIQ which shows significant reduction in hardware due to efficient folded technique. The fabricated chip performances are compared with other reported designs, which shows the presented scheme performs strongly.

Later, a fixed-point DAUB8 DWT and its proposed AIQ based architectures are implemented. The image reconstruction accuracy and required hardware consumption is compared in between fixed-point and AIQ scheme. At fixed PSNR, AIQ scheme consumes less hardware compared to fixed-point. DAUB8 transform can reconstruct the image with finer details. So the AIQ-based DAUB8 transform is very suitable in such applications (e.g., finger print, ECG signal denoising) where higher accuracy is required.

Chapter 6
Conclusion and Future Work

6.1 Summary of Accomplishments

In this thesis, we have presented an area and power efficient architecture to compute two 8-point wavelet transforms: four- and six-tap Daubechies orthonormal wavelet filters. The architecture is developed using a two-level folded mapping technique that is based on the factorization and decomposition of the transform matrices. The use of multi-dimensional AIQ encoding reduces computation error. The architecture is fabricated using 0.18μm CMOS process. Performance comparisons indicate that, the proposed scheme provides an efficient alternative with much lesser computational complexity, silicon area, critical path delay and power consumption, and higher throughput.

DAUB4 and DAUB6 architectures are implemented as a shared hardware core in FPGA and 0.18μm CMOS technology When fabricated in 0.18 μm CMOS technology, the chip area of the dual-DWT processor is 2.08 sq. mm, the maximum frequency is 100 MHz, the gate count is 7,533, and the power consumption is 4.51 mW. Compared to existing designs, the proposed scheme reduces significantly the hardware cost, critical path delay and power consumption with a higher throughput rate.

Later, both FP and AIQ architecture of DAUB8 is implemented in FPGA and compared their performances. At required PSNR, AIQ performs better with less hardware consumption. The architecture is also compared with other designs to assess the hardware cost. Compared to existing designs, the AIQ architecture consumes less hardware.

In Table 6.1 the three architectures are arranged in order of high, medium, low with indexes - (i) Image reconstruction capability, and (ii) Hardware cost. Here the idea is to compare the three architectures among themselves and choosing the right transform for appropriate applications.

Table 6.1 Degree of performance and hardware cost

Degree	Image Reconstruction	Hardware Cost
High	DAUB8	DAUB8
Medium	DAUB6	DAUB6
Low	DAUB4	DAUB4

In case of moderate image reconstruction, DAUB4 would be an excellent choice to save area and power compare to any other architectures studied in this work. At high noise level we propose DAUB6 as an efficient architecture with good image reconstruction capability. In application where we need detail pixels information (e.g., finger print, ECG signal denoising) we propose DAUB8 which is able to reconstruct the image with finer details.

6.2 Recommendations for future work

Future work needs to be directed towards the detailed design and VLSI fabrication of the two-level AIQ-based DAUB8 wavelet transform.

Also, the efficient scheduling of the input datapath and timing operations can be used to implement the one-level AIQ-based DAUB8 architecture. That will reduce the number of sub filter banks used from four to one and will significantly reduce the hardware cost.

Finally, very recently the Finite Ridgelet Transform (FRIT) has been introduced [44] as a sparse expansion of functions on both continuous and discrete spaces that are smooth away from discontinuities along lines. To compute the FRIT, a 1-D DWT is used in the intermediate stage as a secondary transform. So, it would be very useful to investigate the potential benefits of using an AIQ implementation of a 1-D Daubechies wavelet transform in the FRIT application.

REFERENCES

1. S. Mallat, *A wavelet tour of signal processing*. New York: Academic, 1998.

2. I. Daubechies, “The Wavelet Transform Time-Frequency Localization and Signal Analysis,” *IEEE Trans. Information Theory*, vol. 36, pp. 961-1005, Sept. 1990.

3. *JPEG2000 Image Coding System*, ISO/IEC/JTC1/SC29/WG1 N390R, March 1997.

4. J. Walker, *A primer on wavelets and their scientific applications*. CRC Press LLC, 1999.

5. I. Daubechies, *Ten Lectures on Wavelets*. Philadelphia: SIAM, 1992.

6. Q. Dai, X. Chen, and C. Lin, “A novel VLSI architecture for multidimensional discrete wavelet transform,” *IEEE Trans. Circuits and Systems for Video Technology*, vol. 14, pp. 1105-1110, Aug. 2004.

7. A. Acharyya, K. Maharatna, B. Al-Hashimi, S. Gunn, “Memory reduction methodology for distributed arithmetic based DWT/IDWT exploiting data symmetry,” *IEEE Trans. Circuits and Systems II*, vol. 56, pp. 285-289, April 2009.

8. G. Shi, W. Liu, L. Zhang, F. Li, “An efficient folded architecture for lifting-based discrete wavelet transform,” *IEEE Trans. Circuits and Systems II*, vol. 56, pp. 290-294, April 2009.

9. Y. Lai, L. Chen, Y. Shih, “A high-performance and memory-efficient VLSI architecture with parallel scanning method for 2-D lifting-based discrete wavelet transform,” *IEEE Trans. Consumer Electronics*, vol. 55, pp. 400 – 407, May 2009.

10. C. Huang, P. Tseng, and L. Chen, “Flipping structure: an efficient VLSI architecture for lifting based discrete wavelet transform,” *IEEE Trans. Signal Processing*, vol. 52, pp. 1080–1089, April 2004.

11. Y. Seo, and D. Kim, “VLSI architecture of line-based lifting wavelet transform for motion JPEG2000,” *IEEE J. of Solid-State Circuits*, vol. 42, pp. 431-440, Feb. 2007.

12. S. Lee and S. Lim, “VLSI design of a wavelet processing core,” *IEEE Trans. Circuits and Systems for Video Technology*, vol. 16, pp. 1350-1360, Nov. 2006.

13. M. Martina and G. Masera, “Multiplierless, folded 9/7-5/3 wavelet VLSI architecture,” *IEEE Trans. Circuits and Systems II*, 54, pp. 770-774, Sept. 2007.

14. M. Martina and G. Masera, “Low-complexity, efficient 9/7 wavelet filters VLSI implementation,” *IEEE Trans. Circuits and Systems II*, vol. 53, pp. 1289–1293, Nov. 2006.

15. K. Wahid, V. Dimitrov, G. Jullien and W. Badawy, “Error-Free computation of Daubechies wavelets for image compression applications,” *Electronic Letters*, vol. 39, no. 5, pp. 428-429, May 2003.

16. K. Wahid, V. Dimitrov, and G. Jullien, “VLSI architectures of Daubechies wavelet transforms using algebraic integers,” *J. Circuits, Systems, and Computers*, vol. 13, no.6, pp. 1251-1270, June 2004.

17. N. Ahmed, T. Natarajan and K. Rao, “Discrete Cosine Transform,” *IEEE Trans. Computers*, vol. 23, pp. 90-93, Jan. 1974.

18. *JPEG Committee Draft CD10918*, ISO/IEC JTC1/SC29/WG10, Oct. 1991.

19. V.Srinivasa Rao, Dr P.Rajesh Kumar, G.V.H.Prasad, M.Prema Kumar, and S.Ravichand, “Discrete Cosine Transform Vs Discrete Wavelet Transform: An Objective Comparison of Image Compression Techniques for JPEG Encoder,” *Int. J. Advanced Engineering & Applications*, Jan. 2010

20. S. Mallat, “Multifrequency Channel Decompositions of Images Wavelet Models,” *IEEE Trans. Acoustics, Speech and Signal Processing*, vol. 37, pp. 2091-2110, Dec. 1989.

21. A. Primer, *Introduction to Wavelets and Wavelet Transforms*. Prentice Hall, 1998.

22. Rafael C. Gonzalez, and Richard E. Woods, *Digital Image Processing*. New Jersey: Pearson Prentice Hall, 2008.

23. S. Mallat, “A Theory for Multiresolution Signal Decomposition: The Wavelet Representation,” *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol.11, pp. 674-693, July 1989.

24. Robi Polikar (January 12, 2001), The Wavelet Tutorial (2nd ed.) [Online]. Available: http://users.rowan.edu/~polikar/WAVELETS/WTtutorial.html). Access: Aug. 2010.

25. I. Daubechies, "Orthonormal bases of compactly supported wavelets," *Commun. Pure Appl. Math*. Vol. 41, pp. 909-996, 1988.

26. W. H. Press, S. A. Teukolsky, W. T. Vetterling and B. P. Flannery, *Numerical Recipes in C*, 2nd ed. Cambridge University Press, 1999.

27. W. Shann, and C. Yen, “Exact Solutions for Daubechies Orthonormal Scaling Coefficients,” Dept. Mathematics, National Central University, Tech. Rep. TR-9704, Sept. 13, 1997.

28. Richard Dedekind, *Theory of Algebraic Integers, Translated and introduced by John Stillwell*. Cambridge University Press, Sept. 1996.

29. J. H. Cozzens and L. A. Finkelstein, “Computing the Discrete Fourier Transform using Residue Number Systems in a Ring of Algebraic Integers,” *IEEE Trans. Information Theory*, vol. 31, pp. 580-588, Sept. 1985.

30. K. A. Wahid, V. S. Dimitrov, and G. A. Jullien, “Error-Free Arithmetic for Discrete Wavelet Transforms using Algebraic Integers,” in *Proc. of the IEEE Symp. Computer Arithmetic*, Spain 2003, pp. 238-244.

31. D. Knuth, *The Art of Computer Programming, Seminumerical Algorithms*, 3rd ed., vol. 2. Addison-Wesley,1981.

32. K. Wahid, S-B. Ko, and D. Teng, “Efficient hardware implementation of an image compressor for wireless capsule endoscopy applications,” in *Proc. of the IEEE Int. Joint Conf. Neural Network*, Hong Kong, 2008, pp. 2761-2765.

33. R. Hewlitt and E. Swartzlander. “Canonical signed digit representation for FIR digital filters,” in *Proc. of the IEEE workshop on Signal Processing Systems*, Lafayette, LA , USA, 2000, pp. 416-426.

34. *1364-1995 IEEE Standard for Verilog Hardware Description Language*, IEEE, 1995.

35. Wikipedia: Verilog, Wikimedia Foundation Inc. Available: http://en.wikipedia.org/wiki/Verilog. Access: July 23, 2010.

36. *Synthesis and Simulation Design Guide*, Xilinx Development System, Xilinx Inc., 2008.

37. JL Derome, “A tutorial on RMC’s Digital Design Flow (based on CMOSP18 Artisan),” Version (5.0D) for Cadence 2006a, January 28, 2008.

38. B. Wu and C. Lin, “A high-performance and memory-efficient pipeline architecture for the 5/3 and 9/7 discrete wavelet transform of JPEG2000 codec,” *IEEE Trans. Circuits and Systems for Video Technology*, vol. 15, pp. 1615-1628, Dec. 2005.

39. S. Winkler, *Digital Video Quality: Vision Models and Metrics*. UK: Wiley, 2005.

40. M. Tico, E. Immonen, P. Ramo, P. Kuosmanen, and J. Saarinen, “Fingerprint Recognition Using Wavelet Features,” in *Proc. of IEEE Int. Symp. Circuits and Systems*, vol. 2, Sydney, NSW 2001, pp. 21- 24.

41. B. Singh, and A. Tiwari, “Optimal selection of wavelet basis function applied to ECG signal denoising,” *Digital Signal Processing, Elsevier*, vol. 16, issue. 3, pp. 275–287, May 2006.

42. A. M. Al-Haj, “Fast Discrete Wavelet Transformation Using FPGAs and Distributed Arithmetic,” *Int. J. Applied Science and Engineering*, vol. 1, no. 2, pp. 160-171, July 2003.

43. Benkrid, K. Benkrid and D. Crookes, “A novel FIR filter architecture for efficient signal boundary handling on Xilinx VIRTEX FPGAs,” in *Proc. 11th IEEE symp. Field-Programmable Custom Computing Machines*, pp. 273-275, Sept. 2003.

44. M. Do, and M. Vetterli, “The Finite Ridgelet Transform for Image Representation,” *IEEE Trans. Image Processing*, vol. 12, pp. 16-28, Jan. 2003.