A VLSI architecture for lifting-based wavelet packet transform in fingerprint image compression

Tao Zhu

University of Nevada, Las Vegas

Follow this and additional works at: https://digitalscholarship.unlv.edu/rtds

Repository Citation
https://digitalscholarship.unlv.edu/rtds/1574
A VLSI ARCHITECTURE FOR LIFTING-BASED WAVELET PACKET
TRANSFORM IN FINGERPRINT IMAGE COMPRESSION

by

Tao Zhu

Bachelor of Science
Shanghai Jiaotong University, Shanghai, P. R. China
1997

A thesis submitted in fulfillment
of the requirements for the

Master of Science Degree in Engineering
Department of Electrical and Computer Engineering
Howard R Hughes College of Engineering

Graduate College
University of Nevada, Las Vegas
December, 2003

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.
The Thesis prepared by

Tao Zhu

Entitled

"A VLSI Architecture for Lifting-Based Wavelet Transform in FingerPrint Image Compression"

is approved in partial fulfillment of the requirements for the degree of

Master of Science in Electrical Engineering

Examination Committee Chair

Dean of the Graduate College

Examination Committee Member

Graduate College Faculty Representative

PR/1017-53/1-00
ABSTRACT

A VLSI Architecture for Lifting-Based Wavelet Packet Transform
In Fingerprint Image Compression

by

Tao Zhu

Dr. Shahram Latifi, Examination Committee Chair
Professor of Electrical and Computer Engineering
University of Nevada, Las Vegas

FBI uses a technique called Wavelet Scalar Quantization (WSQ), a wavelet packet transform (WPT) based method, to compress its fingerprint images. Though many VLSI architectures have been proposed for wavelet transform in the literature, it is not the case for the WPT. In this thesis, a VLSI architecture capable of computing the WPT is presented for application of WSQ. In the proposed architecture, Lifting Scheme (LS) is used to generate wavelets instead of the traditional convolution filter-bank (FB) specified in original standard. A comparative study between LS and FB shows that quality of images transformed by LS is completely acceptable (with 30dB ~ 40dB PSNR at a target bit rate of 0.75dpp) while fewer operations required. In particular, to compare with FB, the hardware consumption, for our WSQ application, is reduced to half due to the LS. Moreover, this architecture can be easily configured to compute any required WPT application.
TABLE OF CONTENTS

ABSTRACT ........................................................................................................................................iii

TABLE OF CONTENTS ..................................................................................................................iv

LIST OF TABLES ........................................................................................................................vii

LIST OF FIGURES .......................................................................................................................viii

ACKNOWLEDGMENTS ................................................................................................................x

CHAPTER 1  INTRODUCTION .................................................................................................1
  1.1 Statement of Problem and Objectives ..............................................................................2
  1.2 Thesis Overview .................................................................................................................6

CHAPTER 2  WAVELET TRANSFORM AND WAVELET PACKETS .........................7
  2.1 Fourier Transform .............................................................................................................8
  2.2 Wavelet Transform ..........................................................................................................11
  2.3 Wavelet Transform and Filter bank ................................................................................13
    Two-Channel Filter Bank ....................................................................................................13
    Multiresolution Analysis ..................................................................................................15
    Dilation and Wavelet Equations with Filter Banks .........................................................17
  2.4 Wavelet Packets ..............................................................................................................19
  2.5 Summary ..............................................................................................................................21

CHAPTER 3  WAVELET TRANSFORM WITH LIFTING SCHEME ..........................23
  3.1 A General Lifting Scheme ...............................................................................................23
    Split Stage ........................................................................................................................24
    Predict Stage ....................................................................................................................25
    Update Stage ....................................................................................................................26
    The Inverse Transform ...................................................................................................27
  3.2 Boundary Treatment .......................................................................................................28
  3.3 Advantages of Lifting Scheme .........................................................................................29
  3.4 Summary ..............................................................................................................................30

CHAPTER 4  FINGERPRINT AND ITS FBI COMPRESSION SPECIFICATION ........31
  4.1 Types of Fingerprints and Their Interpretation ...............................................................33
  4.2 A Model of the Human Visual System .........................................................................35

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.
CHAPTER 7  CONCLUSIONS AND FUTURE WORK .................................................98
7.1 Conclusions ...................................................................................................................98
7.2 Future Work .................................................................................................................100

BIBLIOGRAPHY .................................................................................................................101

VITA ......................................................................................................................................105
LIST OF TABLES

| Table 4-1 | Types of fingerprint | 33 |
| Table 4-2 | Analysis Wavelet Filter Coefficients for WSQ | 38 |
| Table 4-3 | Huffman Coding Input Symbols | 41 |
| Table 5-1 | Test Fingerprint Images and Their Sizes | 58 |
| Table 5-2 | PSNRs of Reconstructed Fingerprint Images from 64 Subbands | 59 |
| Table 5-3 | PSNRs of Reconstructed Fingerprint Images from Subbands 0 to 59 | 60 |
| Table 5-4 | PSNRs of Compressed Fingerprint Images Using Bior9-7 | 64 |
| Table 5-5 | PSNRs of Compressed Fingerprint Images Using LS9-7I | 65 |
| Table 5-6 | Minutiae Detection and Fingerprint Identification | 67 |
| Table 5-7 | Computational complexities of standard algorithm and Lifting-based algorithm | 71 |
| Table 5-8 | Execution times (second) | 72 |
| Table 6-1 | Constant Coefficients for Forward Transform | 79 |
| Table 6-2 | Constant Coefficients for Inverse Transform | 79 |
| Table 6-3 | Part of the schedule for PM1 and PM2 applied on an even-element (8x8) block | 84 |
| Table 6-4 | Part of the schedule for PM1 and PM2 applied on an odd-element (9x9) block | 85 |
| Table 6-5 | Accessing Order of MEM1 (Example in Section 6.1.5) | 87 |
| Table 6-6 | Preliminary Gate Count Estimates of Basic Components | 96 |
| Table 6-7 | Preliminary Gate Count Estimates and Number of Components Used in the Proposed Architecture and the Convolution Filter-bank Architecture | 96 |
LIST OF FIGURES

Figure 1-1 Block diagram of a fingerprint-collecting device ............................................. 2
Figure 1-2 Subband Organizations after 3 levels transform (a) wavelet transform with pyramid method and (b) wavelet packet transform. .............................. 4
Figure 2-1 A set of Fourier Basic Function ................................................................. 8
Figure 2-2 Varying window sizes for analyzing the frequency content of signals ......... 10
Figure 2-3 Common mother wavelets ........................................................................... 11
Figure 2-4 Dilations (Left) and Translations (Right) of a mother wavelet ................. 12
Figure 2-5 Two-channel filter bank ............................................................................. 14
Figure 2-6 Scaling Function and Wavelet Vector Spaces .............................................. 16
Figure 2-7 Single level wavelet transform .................................................................... 18
Figure 2-8 Multiresolution of filter banks (Forward) .................................................. 19
Figure 2-9 The meaning of channel ABBB in (a) frequency decomposition of wavelet transform domain (wavelet packet), and (b) tree structured representation .............................................................. 21
Figure 3-1 Forward transform based on LS: Split, Predict and Update stages ............... 24
Figure 3-2 Inverse transform based on LS: Update, Predict and Merge stages ............ 27
Figure 3-3 Examples of signal extension: (a) A finite signal, (b) Periodic extension, and (c) Symmetric extension ................................................................. 28
Figure 4-1 A Fingerprint Card .................................................................................... 32
Figure 4-2 Three major types of fingerprint: (a) Arch, (b) Loop, and (c) Whorl ........... 34
Figure 4-3 Gray-scale Bar Charts: (a) gray-scale bar with 32 levels, and (b) gray-scale bar with 256 levels .................................................................................. 36
Figure 4-4 DWT-based encoder and decoder diagram [FBI93] ................................. 37
Figure 4-5 DWT subbands in the WSQ Specification ...................................................... 39
Figure 5-1 A basis Two-dimensional Wavelet Decomposition ....................................... 45
Figure 5-2 The 1D_FILT Procedure ............................................................................ 46
Figure 5-3 The DSAMPLE procedure .......................................................................... 47
Figure 5-4 (a) The 1D_LP procedure and (b) the 1D_HP procedure ......................... 48
Figure 5-5 The 1D_WT procedure ................................................................................ 48
Figure 5-6 Flow chart of (a) 2-D forward wavelet transform and (b) 2-D inverse wavelet transform ......................................................................................... 54
Figure 5-7 Decomposition scheme in WSQ .............................................................. 55
Figure 5-8: Original and Reconstructed Images Using Subbands 0 to 59 (FingerP13, 539x651, Compression Skipped), (a) Original Fingerprint; (b) Bior9-7 (PSNR = +42.65 dB); (c) LS9-7 (PSNR = +42.65 dB); (d) LS9-7I .(PSNR = +41.80 dB) ................................................................. 61
ACKNOWLEDGMENTS

I would like to take this opportunity to express my sincere gratitude to the following people for their invaluable contribution to my research:

My committee chair, Dr. Shahram Latifi, who initially inspired my interests in the image processing field, and who guided me through the entire research process of this thesis. He helped me learn and grow, and for his meticulously correcting my writing and, more importantly, for his continuous encouragement and believing in me.

My committee members, Dr. Emma Regentova, Dr. Henry Selvaraj, and Dr. Wolfgang Bein, who assisted me in improving this thesis.

My wife, Xiaojun. She offers full support to me all the time.

I would like to extend my thanks to all the professors and graduates in the Electrical and Computer Engineering Department, and the Computer Science Department.
CHAPTER 1

INTRODUCTION

Fingerprints have been used as unique identifiers of individuals for a very long time. FBI’s fingerprints collection effort is one of the notable examples. FBI had already maintained about 200 million criminal records with fingerprints in the form of inked impressions on paper cards in 1995 [Bris95]. FBI expected this fingerprint database to be digitized at 500 dots per inch with 8 bits of grayscale resolution. This resulted in some 10 megabytes per card, making the entire database about 2,000 terabytes in size. It would cost a significant storage space. So, FBI adopted an approach to compress the database, which is a discrete wavelet transform-based approach referred to as Wavelet/Scalar Quantization (WSQ). Following this approach, FBI produced archival quality images at compression ratios of up to 15:1 [BrBH93].

Fingerprint data collected by FBI continues to accumulate at a rate of 30000 to 50000 new cards per day. In addition, fingerprints are still collected in the form of inked impressions on paper cards. Those cards are digitized and compressed later. The current situation introduces a crucial problem in transmission, storage, and automatic analysis - the fingerprints will not be analyzed or identified on time. One of the feasible solutions is to collect fingerprints in digital form directly by a portable fingerprint-collecting device with a fingerprint sensor, an image processor, and a storage module (as shown in Figure 1-1). The image processor consists of a wavelet transform module, a quantization
module, and an entropy-encoding module to carry out WSQ. And the Wavelet Transform Module (the shaded block in Figure 1-1) is the vital part of this device.

![Figure 1-1: Block Diagram of a Fingerprint-collecting Device](image)

1.1 Statement of Problem and Objectives

The purpose of this thesis is to design a VLSI architecture for wavelet transform module of the image processor of a portable fingerprint-collecting device. In particular, it attempted to achieve the following objectives:

1. The proposed VLSI architecture is based on Lifting Scheme instead of digital filter bank structure to construct the wavelet used by WSQ in FBI's fingerprint compression specification.

2. The VLSI architecture is able to carry out a two-dimensional Wavelet Packet Transform specified in WSQ for fingerprint application.

There are two generations of wavelets according to the construction methods. The first generation wavelet employs the Fourier transform, in which wavelets are defined as translations and dilations of one function. The transformation based on the first
generation wavelets can be generated from digital filter banks, such as the filter banks used by WSQ, in which a 9-tap lowpass filter and a 7-tap highpass filter are used. It has been implemented by convolution traditionally. Such an implementation demands both a large number of computations and a large storage – features that are not desirable for high-speed or low-power applications.

The second generation wavelets are based on Lifting Scheme (LS), which was introduced in 1995 by W. Sweldens [Swel95]. The main feature of this scheme is to break up the highpass and lowpass filters into a sequence of upper and lower triangular matrices and converts the filter implementation into banded matrix multiplications [DaSw98]. It requires far fewer computations than the previous type.

Wavelet Transform can decompose the image to many subbands to achieve a multiresolution analysis. In the most common wavelet transform, only the results of lowpass filters (lowpass subband) are treated which is based on the fact that the lower frequencies contain more information than the higher frequencies. Thus, it is called a Mallat pyramid method [Mall89]. It results in four subbands at each level transformation. A 3-level wavelet transform using pyramid method is illustrated by Figure 1-2 (a).

The wavelet decomposition method in WSQ differs from the pyramid method, and is called Wavelet Packet Transform (WPT). WPT was introduced by Coifman and Wickerhauser in 1992 [CoWi92]. The main difference is that, in wavelet packets, the basic two-channel filter bank can be iterated over the lowpass subbands, the highpass subband or both of them. In Figure 1-2 (b), a subband organization after a 3-level full WPT is shown. In this example, both the lowpass subbands and highpass subbands are
transformed. Therefore, the first-level transform results in 4 subbands. The second-level transform, applying to the 4 subbands, results in $4 \times 4 = 16$ subbands. And at the third-level transform, each of 16 subbands is decomposed into 4 subbands. At last, it results in $16 \times 4 = 64$ subbands.

![Wavelet Transform with Pyramid Method](image1)

![Wavelet Packet Transform](image2)

**Figure 1-2: Subband Organizations after 3 levels transform**

Although many algorithms and VLSI architectures have been proposed for computing wavelet transform, literature is scarce in regard to WPT, especially two-dimensional WPT. Several papers proposed architectures for WPT for one-dimensional application, such as speech processing and wireless communication. In 1999, Xiaodong Wu et al [WLCh99] proposed a two-buffer memory system-based architecture suitable for signals which are segmented in frames. The size of the two buffers equals the frame
length. In 2000, [TLZa00] proposed a configurable architecture for WPT which was based on [WLCh99] and made some improvements. It is a word-serial architecture where the inputs are supplied to the filters in a serial manner. Corresponding to a J-level WPT, J identical process units and J memory modules are cascaded. This architecture owns a smaller memory module and smaller latency than [WLCh99]. In 2002, [TLZa02] and [JaMa02] proposed two similar word-parallel architectures. These two architectures, both came from [TLZa00], achieve a higher transform speed by feeding the inputs into the filters in a parallel manner. Since the above four architectures were all designed for one dimensional WPT applications, it will be very difficult to extend them to fit to two dimensional applications. The extensions will either require massive memory or cause worse hardware utilization.

We propose a VLSI architecture for two dimensional WPT, which is capable of executing the filters mentioned in WSQ using the LS. In designing this architecture, our goal is the same as in design of any VLSI system, i.e. to achieve high hardware utilization and to reduce chip area. The chip area can be reduced by minimizing the number of processing elements, registers and interconnection wires. And better hardware utilization can be achieved by better scheduling of pipeline and computation. The prototype consists of four processing blocks, which can perform integer-to-integer transforms and generate a pair of wavelet coefficients at a time. The data path is pipelined. The output is written back to memory in-place. Furthermore, this architecture can perform both forward transform and inverse transform by just swapping the lifting coefficients. The proposed architecture is area efficient, and provides parallel executions.
It is suitable for on-chip or single-chip implementation. It can be used not only in fingerprint image processing but also as a WPT engine in other real-time systems.

1.2 Thesis Overview

The thesis is organized as follows:

- Chapter 2 covers a survey on the theoretical background, briefly introducing the Wavelet Transform, Packets and their relations with filter bank structure.
- Chapter 3 explains the Lifting Scheme implementation of the Discrete Wavelet Transform, by highlighting its different phases, properties and advantages.
- Chapter 4 introduces the FBI fingerprint compression standard.
- Chapter 5 describes the performance comparison between filter-bank based DWT algorithm used by FBI WSQ standard and alternative Lifting Scheme based integer-to-integer transform algorithm. This chapter is essential for understanding the hardware design as explained in the chapter 6.
- Chapter 6 proposes the VLSI architecture for wavelet decomposition framework of FBI standard. The organization of the design is a bottom-up fashion and architecture is optimized for WPT.
- Chapter 7 presents a short description of what was achieved in this thesis, discusses the potential of the design and gives some tips for future improvement.
A wavelet transform is similar to a Fourier transform. With a Fourier transform, a function (or a signal) is decomposed into a weighted sum of sinusoids and cosines. With a wavelet transform, a function (or a signal) is decomposed into a weighted sum of wavelet function.

Why should we use wavelets instead of the traditional Fourier methods? There are some important differences between Fourier analysis and wavelets. First, Fourier basis functions are localized in frequencies but not in time. Small frequency changes in Fourier transform will produce changes everywhere in the time domain. Wavelets are local in both frequency/scale (via dilations) and in time (via translations). This localization is an advantage in many cases. In particular, the wavelet transform (WT) is of interest for the analysis of non-stationary signals, because it provides an alternative to classical short-time Fourier transform (STFT). The basic difference is as follows: in contrast to STFT, which uses a single analysis window, the WT uses short windows at high frequencies and long windows at low frequencies.

Second, many classes of functions can be represented by wavelets in a more compact way. For example, functions with discontinuities and functions with sharp spikes usually take substantially fewer wavelet basis functions than sine-cosine basis functions to achieve a comparable approximation. This sparse coding makes wavelets excellent tools
in data compression, like FBI fingerprint compression we focus on in this thesis. The compression ratios are up to 15:1, and the difference between the original image and the decompressed one can be perceived only by an expert [BrBH93].

2.1 Fourier Transform

A survey on Fourier Transform (FT) is first presented prior to going further to the Wavelet Transform because the Fourier Transform is widely used in analyzing and interpreting signals and images.

J. Fourier discovered in the early 18th century that it is possible to compose a periodic signal by superposing a series of sine and cosine functions (Fourier Transform). These sine and cosine functions are known as basic functions (see Figure 2-1) and are mutually orthogonal.

![Figure 2-1: A set of Fourier Basic Function](image)

The transform decomposes the signal into the basic functions, which means that it determines the contribution of each basis function in the structure of the original signal. These individual contributions are called the (Fourier) coefficients. Reconstruction of the
original signal from its Fourier coefficients is accomplished by multiplying each basic function with its corresponding coefficient and adding them up together. The standard forward and inverse Fourier transform can be expressed as:

\[ F(\omega) = \int_{-\infty}^{\infty} f(t) e^{-i\omega t} \, dt \]  
\[ f(t) = \int_{-\infty}^{\infty} F(\omega) e^{i\omega t} \, d\omega \]  

Discrete Fourier Transform (DFT) is an estimation of the Fourier Transform, which uses a finite number of sample points of the original signal to estimate the Fourier Transform of it. The order of computation cost for the DFT is in order of \( n^2 \) where \( n \) is the length of the signal. Fast Fourier Transform (FFT) is an efficient implementation of the Discrete Fourier Transform, which can be applied to the signal if the samples are uniformly spaced. FFT reduces the computation complexity to the order of \( O(n \log n) \) by taking advantage of self similarity properties of the DFT [Daub92].

If the frequency content of an input were to vary drastically from interval to interval, the standard FT would sweep over the entire time axis and wash out any local anomalies (e.g. bursts of high frequency) in the signal. Under such conditions, Gabor (1946) resorted to the windowed, short-time Fourier transform (STFT), which moves a fixed-duration window (see figure 2-2) over the time function and extracts the frequency content in the interval.

The STFT positions a window \( g(t) \) at some point \( \tau \) on the time axe, and calculates the Fourier transform of the signal within the extent or spread of the window. The basis function of this transform are generated by modulation and translation of the window function,
\[ F(\omega, \tau) = \int_{-\infty}^{\infty} f(t) \ g(t - \tau) \ e^{-j\omega t} \ dt \]  \hspace{1cm} (2.3)

where $\omega$ and $\tau$ are modulation and translation factors, respectively.

That fixed-duration window method brings a lot of conveniences in signal analysis. But for a signal combining with different frequencies, we cannot know the exact frequency and location of a signal component simultaneously. It means that if a short window is used, high frequency component can be located very well in time; however, short duration windows are insufficient for analyzing low frequency components as shown in figure 2-2. If a long window is used, low frequency components can be analyzed; that is, the signal can be resolved very well in frequency. But now, the high frequency components can no longer be located very well in time. We have to compromise between time resolution and frequency resolution.

![Figure 2-2: Varying window sizes for analyzing the frequency content of signals](image-url)
2.2 Wavelet Transform

Unlike the FT and STFT, the wavelet transform is founded on basis functions formed by dilation and translation of a prototype function. These basis functions are short-duration, high-frequency and long-duration, low-frequency functions. So, the varying of the time interval, or window length, is exactly what the wavelet transform accomplishes.

Wavelets are mathematical functions that satisfy certain criteria, like a zero mean, and are used for analyzing and representing signals or other functions. A single wavelet, from which a set of wavelet functions is generated, is called the mother wavelet. Figure 2-3 shows some of the commonly used mother wavelet.

![Figure 2-3: Common mother wavelets](image)

If given a real mother wavelet, \( \psi(t) \), the set of wavelet function is generated by scaling (dilating or compressing) and translating the mother wavelet as

\[
\psi_{a,t}(t) = \frac{1}{\sqrt{a}} \psi\left(\frac{t - \tau}{a}\right),
\]

(2.4)

where \( a \) is the scaling factor and \( \tau \) is the translation factor. This set of wavelet functions is referred as wavelet basis. In figure 2-4, the left diagram depicts dilations of a prototype (mother) wavelet while the right diagram shows translations of it.
The forward wavelet transform, or analysis part, is simply correlation of wavelet basis with an input function. It calculates the contribution of each dilated and translated version of the mother wavelet in the original data set. These contributions are called the wavelet coefficients. Assuming that the input function is $f(t)$ and wavelet coefficient is denoted $W(a,b)$, we can define the wavelet transform of this input as

$$W(a,b) = \int_{-\infty}^{\infty} f(t) \psi_{a,b}(t)\, dt,$$  \hspace{1cm} (2.5)

where $\psi_{a,b}(t) = \frac{1}{\sqrt{a}} \psi\left(\frac{t-b}{a}\right)$, $a$ and $b$ are the scaling factor and the translation factor, respectively.

Mathematically, the transform can be considered as the inner product of the two functions $f(t)$ and $\psi_{a,b}(t)$. The equation (2.5) also can be represented in the form

$$W(a,b) = \langle f(t), \psi_{a,b}(t) \rangle.$$

(2.6)
Since wavelets are used to transform a function, the inverse transform is needed and defined by

\[ f(t) = \frac{1}{C} \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} \frac{1}{|a|^2} W(a, b) \psi_{a, b}(t) \, da \, db , \]  

(2.7)

where the quantity \( C \) is defined by means of \( \Psi(\omega) \), which is the Fourier transform of \( \psi_{a, b}(t) \).

2.3 Wavelet Transform and Filter bank

In many applications, one never has to deal directly with the scaling functions or wavelets for wavelet transform. Only the scaling function coefficients, wavelet coefficients and the output coefficients after transforms need be considered, and they can be view as digital filters coefficients and digital signals respectively. Therefore, It is possible to develop most of the results of wavelet theory using only filter banks. In this section, we will connect the wavelet transform (DWT) and filter banks.

2.3.1 Two-Channel Filter Bank

A filter is a linear operator defined in terms of its filter coefficients \( h(0), h(1), \ldots \). Since the number of filter coefficients \( h(i) \) is finite, the filter is a finite impulse response or FIR and the operator is represented as

\[ y(n) = \sum_{k} h(k) x(n - k) = h \ast x \]  

(2.8)

where \( x(n) \) and \( y(n) \) are the input and output signals, respectively, and the symbol * indicates a convolution.
Figure 2-5 shows the basic idea of a two-channel filter bank. It shows an *analysis bank* consisting of two filters, a lowpass filter $h_0$ and a highpass filter $h_1$. The outputs of the analysis bank are called *subband coefficients*, which can be represented as

\[
c_o(k) = \sum_n h_o(2k - n)x(n),
\]

\[
d_o(k) = \sum_n h_1(2k - n)x(n).
\]

![Two-channel filter bank](image)

There is a *synthesis bank* also shown in figure 2-5, which consists of two inverse filters $\tilde{h}_0$ and $\tilde{h}_1$. The reconstructed signal can be derived from

\[
\tilde{x}(m) = \sum_k [\tilde{h}_0(2k - m)c_o(k) + \tilde{h}_1(2k - m)d_o(k)].
\]

For perfect reconstruction, i.e., $\tilde{x}(m) = x(m)$, the filter banks must satisfy some conditions. In summary of [BuGG98], these conditions for perfect reconstruction in the two-channel filter bank are

\[
\sum_k [\tilde{h}_0(2k - m)h_0(2k - n) + \tilde{h}_1(2k - m)h_1(2k - n)] = \delta(m - n).
\]
In order for it to hold, the four filters have to be related as

1. In the orthogonal case,
\[ h_i(n) = (-1)^{i+1} h_0(N - 1 - n), \]  \hspace{1cm} (2.13)
\[ \tilde{h}_0(n) = h_0(N - 1 - n), \]  \hspace{1cm} (2.14)
\[ \tilde{h}_i(n) = (-1)^i h_0(n); \]  \hspace{1cm} (2.15)

2. In the biorthogonal case,
\[ \tilde{h}_i(n) = (-1)^i h_0(1 - n), \]  \hspace{1cm} (2.16)
\[ h_i(n) = (-1)^i \tilde{h}_0(1 - n). \]  \hspace{1cm} (2.17)

2.3.2 Multiresolution Analysis

A decomposition of the whole function space into individual subspaces following a multiple scales is known as multiresolution. In multiresolution, there are two families of subspaces, \( V_i \) and \( W_i \), \(-\infty < i < \infty\), which are called as the Scaling Function space Wavelet Vector space in wavelet transform system, respectively. The spaces \( V_i \) are increasing as \( i \) increases. The space \( W_i \) are the differences between the \( V_i \). In summary of [ErHJ96], the family of subspaces \( V_i \) must satisfy the following four requirements.

1. \( V_i \subset V_{i+1} \) and \( \bigcap V_i = \{0\} \) and \( \bigcup V_i = L^2 \)

2. \( f(t) \in V_i \iff f(2t) \in V_{i+1} \)

3. \( f(t) \in V_0 \iff f(t - k) \in V_0 \)

4. \( V_0 \) has an orthonormal basis \( \{\phi(t - k)\} \).

Where \(-\infty < t < \infty\) and \( k = 0, 1, 2, \ldots \) is the shift time. A function \( f(t) \) in the whole space has a projection in each subspace. Those projections contain more and more of the full information in \( f(t) \). The projection in \( V_i \) is called \( f_i(t) \).
Now, we identify the second family of subspaces. \( W_i \) contains the new information

\[
\Delta f_i(t) = f_{i+1}(t) - f_i(t).
\]

This is the detail at level \( i \). From the viewpoint of individual functions,

\[
f_i(t) + \Delta f_i(t) = f_{i+1}(t)
\]

and from the viewpoint of the subspaces they lie in, these are

\[
V_i \oplus W_i = V_{i+1}
\]

(2.19)

where \( \oplus \) denoted the direct sum. As shown in figure 2-6, \( W_i \) is the orthogonal complement of \( V_i \), within the large space \( V_{i+1} \). Each function in \( V_{i+1} \) is the sum of two orthogonal parts, \( f_i \) in \( V_i \) and \( \Delta f_i \) in \( W_i \). When we start from \( V_0 \), and add up details (subspaces), then we have

\[
V_0 \oplus W_0 \oplus W_1 \oplus W_2 \oplus \ldots \oplus W_i = V_{i+1}.
\]

(2.20)

For the functions in these subspaces, this equation is simply

\[
f_0(t) + \Delta f_0(t) + \Delta f_1(t) + \Delta f_2(t) + \ldots + \Delta f_i(t) = f_{i+1}(t).
\]

(2.21)
2.3.3 Dilation and Wavelet Equations with Filter Banks

In a wavelet system, the dilation equation can be defined as [BuGG98]

\[ \phi(t) = \sqrt{2} \sum_k h(k) \phi(2t - k), \] (2.22)

which is a direct consequence of \( V_0 \subset V_1 \). There will be a finite set of coefficients \( h(0), \ldots, h(N) \), when the function \( \phi(t) \) is supported on \([0, N]\).

The scaling functions \( \phi(2^j t - k) \) are orthogonal at each scale separately. They are not orthogonal across scales. For example, the function \( \phi(t) \) in \( V_0 \) is also in \( V_1 \). So \( \phi(t) \) is not orthogonal to the basis function \( \phi(2t-k) \) in \( V_1 \). Orthogonality across scales comes from wavelet subspaces \( W_i \) and their basis function \( W_{ik}(t) \).

The translations of \( \psi(t) \) span \( W_0 \). The translations of \( \psi(2^j t) \) span \( W_i \). Those spaces are orthogonal because \( W_0 \subset V_i \) and \( V_i \perp W_i \) as shown in figure 2-6. The wavelet equation produces the wavelet directly from the scaling function:

\[ \psi(t) = \sqrt{2} \sum g(k) \phi(2t - k). \] (2.23)

The wavelet coefficient, \( g(k) \), are required by orthogonality to be related to the scaling function coefficients (orthogonal wavelet system) by

\[ g(k) = (-1)^k h(N - 1 - k), \] (2.24)

or the dual scaling function coefficients (biorthogonal wavelet system) by

\[ g(k) = (-1)^k \tilde{h}(1 - k). \] (2.25)

According to equation (2.21), any function \( f(t) \) in the whole space could be written

\[ f(t) = \sum_k \tilde{h}_k \phi_k(t) + \sum_j \sum_k \tilde{g}(j, k) \psi_{jk}(t) \] (2.26)

as a series expansion in terms of the scaling function and wavelets.
As discussed above, the construction of wavelets has succeeded by finding the \( V_i \), through the scaling function \( \phi(t) \). Then the wavelet spaces \( W_i \) are the differences between \( V_{i+1} \) and \( V_i \). It means that the wavelet, \( \psi(t) \), can be constructed from a scaling function. Comparing equation (2.8) and (2.22), we find a link between the scaling functions \( \phi(t) \) and the filter bank. Let us design a scaling function coefficients \( h(k) \), which will be considered as the choice of a lowpass filter \( h_0(k) \) in a two-channel filter bank. Then the wavelet coefficients \( g(k) \) related to highpass filter \( h_1(k) \) can be found by equation (2.24) or (2.25). A single level wavelet transform can be implemented by the two-channel filter banks as shown in figure 2-7.

![Figure 2-7: Single level wavelet transform](image)

We also show the filter banks structure for multiresolution analysis in figure 2-8. At each step, the highpass filter produces the new detail \( \Delta f_i \) in \( W_i \) as represented by equation (2.20) and (2.21). It is also called as a *pyramid structured decomposition*. 

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.
2.4 Wavelet Packets

The pyramid structured wavelet transform decomposes a signal into a set of frequency channels that have narrower bandwidths in the lower frequency region. The transform is suitable for signals consisting primarily of smooth components, so that their information is concentrated in the low frequency regions. However, it may not be suitable for quasi-periodic signals such as speech signals whose dominant frequency channels are located in the middle frequency region. To analyze quasi-periodic signals, the concept of wavelet bases has been generalized to include a library of modulated waveform orthonormal bases [Wick94], called wavelet packet bases or simply wavelet packets.

In summary of [ChKu93], the library of wavelet packet basis functions \( \{WP_n\}_{n=0}^\infty \) can be generated by

![Figure 2-8: Multiresolution of Filter Banks (Forward)](image-url)
\[ WP_{2n}(t) = \sqrt{2} \sum_{k} h(k) WP_{n}(2t - k) \]  
(2.27)

\[ WP_{2n+1}(t) = \sqrt{2} \sum_{k} g(k) WP_{n}(2t - k) \]  
(2.28)

where the function \( WP_{0}(t) \) can be identified with the scaling function \( \phi \) and \( WP_{1}(t) \) with the mother board wavelet \( \psi \). Then, the library of wavelet packet bases can be defined to be the collection of orthonormal bases, composed of functions of the form \( WP_{n}(2^{l}t - k) \), where \( l, k \in \mathbb{Z}, n \in \mathbb{N} \). Each element of the library is determined by a subset of the indices: a scaling parameter \( l \), a localization parameter \( k \), and an oscillation parameter \( n \).

The tree-structured wavelet transform as shown in Figure 2-9, provides an algorithmic approach to represent a function in terms of a certain wavelet packet basic.

The 2-D wavelet (or wavelet packet) basis functions can be expressed by the tensor product of 1-D wavelet (or wavelet packet) basis functions along the horizontal and vertical directions. The corresponding 2-D filter coefficients can be expressed as

\[ h_{LL}(k, l) = h(k)h(l), \]

\[ h_{LH}(k, l) = h(k)g(l), \]

\[ h_{HL}(k, l) = g(k)h(l), \]

\[ h_{HH}(k, l) = g(k)g(l) \]  
(2.29)

where the first and second subscripts denote the lowpass and highpass filter characteristics in \( x \) and \( y \) directions, respectively.
In this chapter, we explained the reason for using the wavelet transform instead of the Fourier transform or the short-time Fourier transform. The dilation and translation characteristics of the wavelet transform are advantageous for analyzing a non-stationary signal, such as the fingerprint images. In section 2.3, we gave a summary of filter banks and the connection between wavelets and filters. The connection is that the discrete wavelet transform (DWT) can be implemented by choosing proper filter bank architectures. And it shows us a method to implement wavelet transform in hardware via

Figure 2-9: The Meaning of Channel ABBB in (a) Frequency Decomposition of Wavelet Transform Domain (Wavelet Packet) and (b) Tree Structured Representation
the mature filter bank technique. A summary of a wavelet packet was given in section 2.4. The wavelet packet transform provides a different decomposition structure with a traditional pyramid algorithm. Since it is adopted by FBI in fingerprint image processing, which will be introduced in chapter 4, we will propose a VLSI architecture for it in chapter 6.
CHAPTER 3

WAVELET TRANSFORM WITH LIFTING SCHEME

Wavelets based on dilations and translations of a mother wavelet are referred to as first generation wavelets or classical wavelets. The Lifting Scheme (LS) provides a much more flexible method for constructing wavelets. These wavelets are named as second generation wavelets. Unlike the classical wavelet, wavelets based on LS are not necessarily translations and dilations of one function. Therefore, they can be used conveniently to define wavelet bases for bound intervals, irregular sample grids or even for solving equations or analyzing data on curves or surfaces. In addition, those existing classical wavelets can be implemented with LS by factoring them into Lifting steps [DaSw98].

3.1 A General Lifting Scheme

The basic idea behind the lifting scheme is very simple that the redundancy will be removed by using the correlation in the data. To reach this goal, the data are first split into two subsets, odd subset with the odd-indexed samples and even subset with the even-indexed samples. It is called split stage. And then, we predict the odd subset from the even subset. It is called the predict stage. At last, the prediction error, which is the difference between odd-indexed sample and its prediction, is used to update the even-
indexed data. It is called the update stage. Some details for these three stages are given later in this section. The block diagram of this scheme is illustrated in Figure 3-1.

Suppose that we have a row signal set $X_j$ with $2^l$ values and decompose it into two smaller subsets, $X_{j-1}$ and $Y_{j-1}$, each with $2^{l-1}$ values. Here, the smaller X’s (or Y’s) index, the smaller the size of the subset. For example, the subsets with index $j-1$ are decomposed from the set with index $j$.

![Diagram](image)

Figure 3-1: The forward transform based on LS: Split, Predict and Update stages

3.1.1 Split Stage

In this first stage, the split operation, $S$, splits a row set, $X_j$, into two subsets denoted by $even_{j-1}$ and $odd_{j-1}$. The $even_{j-1}$ contains all the even indexed values $\{x_{j, 2k}\}$ of $X_j$, and the $odd_{j-1}$ contains all the odd indexed values $\{x_{j, 2k+1}\}$ of $X_j$. The range of $k$ is from 0 to $2^{(l-1)} - 1$. Thus, the above operations are denoted by

$$\{even_{j-1}, odd_{j-1}\} = S(X_j) \quad (3.1)$$

This kind of splitting operation is also referred to as the Lazy Wavelet Transform because it does not decorrelate the data. Furthermore, we can define $X_{j-1}$ and $Y_{j-1}$ as

$$X_{j-1} = even_{j-1} = \{x_{j, 2k}\} \quad (3.2)$$
\[ Y_{j-1} = \text{odd}_{j-1} = \{x_{j,2k-1}\} \quad (3.3) \]

### 3.1.2 Predict Stage

Because each value in odd subset is adjacent to the corresponding value in the even subset, the values in two subsets are correlated and either can be used to predict the other. Here, in the second stage -- the predict stage, the even subset \(\text{even}_{j,1}\) is used to predict the odd subset \(\text{odd}_{j,1}\) by a prediction function \(P\), which is independent of the data. The prediction is denoted by

\[ Y_{P,j-1} = P(\text{even}_{j-1}) = P(X_{j-1}). \quad (3.4) \]

Then it seems that \(\text{even}_{j,1}\) can be used to replace the original data set, since we can predict the missing part to reassemble \(X_j\).

But in practice, it is impossible to predict \(\text{odd}_{j,1}\) from \(\text{even}_{j,1}\) exactly. Since \(Y_{P,j-1}\) is very close to \(\text{odd}_{j,1}\) (or \(Y_{j,1}\)), we can replace \(Y_{j,1}\) with the difference between itself and its predicted value \(Y_{P,j-1}\) and denote it by

\[ Y_{j-1} = \text{odd}_{j-1} - Y_{P,j-1} = Y_{j-1} - P(X_{j-1}). \quad (3.5) \]

If the prediction is reasonable, \(Y_{j-1}\), the difference from equation 3.5, will contain much less information than the original \(\text{odd}_{j,1}\) (or \(Y_{j,1}\)).

We can now iterate the split stage and the predict stage so that \(X_{j,1}\) is split into two subsets \(X_{j,2}\) and \(Y_{j,2}\) and then replace \(Y_{j,2}\) with the difference between \(Y_{j,2}\) and \(P(X_{j,2})\). By iterating the scheme \(m\) times, we can replace the original data set \(X_j\) with the wavelet representation \(\{X_{j,m}, Y_{j,m}, \ldots, Y_{j,1}\}\). Given that the wavelet sets encode the difference with some predicted values based on a correlation model, this gives a more compact representation.
3.1.3 Update Stage

However, the split and predict stages are not sufficient in some case. The reason is that we want some global properties of the original data set to be maintained in the smaller set \( X_{j,m} \). For example, in an image, we would like the smaller images \( X_{j,m} \) to have the same overall brightness, i.e. the same average pixel value, as \( X_j \). If the splitting stage is simply subsampling and we iterate the scheme till \( X_{j,m} \) is only 1 pixel, that pixel will be an arbitrary pixel from the original image. We would rather have the last value to be the average of all the pixel values in the original image instead of that arbitrary pixel value.

To achieve this purpose, the third stage - update stage, is introduced in lifting scheme. The idea is to find a better \( X_{j-1} \), so that a certain scalar quantity \( Q \), e.g. the mean, is preserved, or

\[
Q(X_{j-1}) = Q(X_j). \tag{3.6}
\]

Then, a new operator \( U \) is constructed to ensure the preservation of the quality. Therefore, we use the already computed wavelet set \( Y_{j,i} \) to update \( X_{j,i} \). In other words, we construct an operator \( U \) and update \( X_{j,i} \) as

\[
X_{j-1} = X_{j-1} + U(Y_{j-1}). \tag{3.7}
\]

In summary, the lifting scheme consisting of three stages leads to the following forward wavelet transform algorithm:

for \( j = n \) downto 1
\[
\{X_{j-1}, Y_{j-1}\} := S(X_j);
Y_{j-1} := Y_{j-1} - P(X_{j-1});
X_{j-1} := X_{j-1} + U(Y_{j-1});
\]
end for.
3.1.4 The Inverse Transform

One of the great advantages of the lifting scheme realization of a wavelet transform is that it decomposes the wavelet filters into extremely simple elementary steps, and each of these steps is easily invertible. As a result, the inverse wavelet transform can always be obtained immediately from the forward transform. Figure 3-2 shows the block diagram of the inverse transform. The inverse transform follows the rules like: revert the order of the operations in the forward transform, invert the signs in the lifting steps, and replace the splitting step $S$ by a merging step $M$. This leads to the following algorithm for the inverse wavelet transform:

$$
\text{for } j = 1 \text{ to } n \\
\quad X_{j+1} := X_{j-1} - U(Y_{j-1}); \\
\quad Y_{j+1} := Y_{j-1} + P(X_{j-1}); \\
\quad X_j := M(X_{j-1}, Y_{j-1}); \\
\text{end for.}
$$

The block diagram for the inverse transform is depicted in figure 3-2.

Figure 3-2: The inverse transform based on LS: Update, Predict and Merge stages
3.2 Boundary Treatment

Real world signals are limited in time (space), i.e. they do not extend to infinity. Filter bank algorithms assume, however, that the signal is infinitely long. Thus, we confront a problem when we process the boundary of signals. There are several ways to deal with this problem. One could for instance extend the signal with zeros (*zero padding*). In this case, the number of coefficients of the transformed signal will be obviously more than the original signal. Furthermore, as signals do not generally converge to zero towards the ends, extending the signal with zeros can lead to coefficients with large values, which leads to significant coding inefficiencies. Truncating the number of coefficients to the number of samples of the original signal, or quantization errors of coefficients with large values will significantly distort the reconstructed image.

![Examples of signal extension](image)

(a) A finite signal

(b) Periodic extension

(c) Symmetric extension

Figure 3-3: Examples of signal extension

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.
Another option is to make the signal periodic, i.e. to repeat the signal at its ends (Figure 3-5-b). As the values at the left and right ends of the signal are not necessarily the same, discontinuity will appear at signal ends and as a result, a similar problem can arise as mentioned with zero padding.

For symmetric wavelets, an effective strategy for handling boundaries is to extend the image via reflection. Such an extension preserves continuity at the boundaries and usually leads to much smaller wavelet coefficients than if discontinuities were present at the boundaries (Figure 3-5-c).

3.3 Advantages of Lifting Scheme

Summarized, the lifting scheme has the following immediate advantages, when compared to the classical filter bank algorithm:

- Lifting leads to a speedup when compared to the classic implementation. Classical wavelet transform has a complexity of order $n$, where $n$ is the number of samples. For long filters, Lifting Scheme speeds up the transform with a factor of two [Swel95]. Hence it is also referred to as fast lifting wavelet transform (FLWT).

- All operations within lifting scheme can be done entirely in parallel while the only sequential part is the order of lifting operations.

- Lifting scheme allows a fully in-place calculation of the wavelet transform. An auxiliary memory is not needed since it does not need other samples than the output of the previous lifting step. The value in original signal can be replaced with its transform.
The inverse wavelet transform based on LS can always be obtained immediately from the forward transform. The only thing to do is to reverse the operations and toggle + and −.

3.4 Summary

In this chapter, the Lifting Scheme implementation of the Wavelet Transform is described briefly. We saw that the Lifting scheme uses three simple steps to calculate the wavelet coefficients, namely the Split, Predict and Update phase. We discussed these three steps and explained how the inverse transform can easily be calculated in a similar way. Furthermore, we discussed how boundary issues, rising in finite length signals, can be treated by using adaptive filters in the Lifting scheme. A summary of the nice properties of the Lifting Scheme was given in section 3.3. Conclusively, when we implement wavelet transform in hardware, the LS provides a simpler and more efficient method than the traditional filter bank. In the following chapters, we will use the LS to fulfill the wavelet transform for the fingerprint image processing which once was processed by filter banks.
CHAPTER 4

FINGERPRINT AND ITS FBI COMPRESSION SPECIFICATION

Fingerprints have been used as unique identifiers of individuals for a very long time. Figure 4-1 is a fingerprint card used by FBI. It records one of the biological properties of a person. To file a fingerprint card efficiently, the classification of fingerprints is necessary. The dots, bifurcations, islands (hollow circles and ovals), points where ridges suddenly end, and extremely short ridges are the essential characteristics for fingerprint classification (They are also called fingerprint minutiae.) and evidence for courts [Unit85].

FBI started collecting fingerprints in the form of inked impressions on paper cards since 1924. In 1995, more than 200 million cards were in their database, occupying an acre of filing cabinets in the J. Edgar Hoover building in Washington, D.C.. What’s more, these cards keep accumulating at a rate of 30,000 to 50,000 new cards per day [Bris95]. To save them as an efficiently compressed digital format is necessary.

For fingerprint compression, we want the image reconstructed from the compressed image to keep as many minutiae of the original image as possible. Lossless techniques remove the redundancy, which can be added back, so the decompressed data and the original data are identical. The disadvantage of lossless techniques is the low compression ratio, usually in the range from 1:1 to 3:1. Considered the limitations of the human visual system (HVS) and the quality requirement for a decompressed fingerprint

31
image, the lossy (entropy reduction) techniques can reach the goal of the best reconstructed quality with the highest compression ratio, actually. So FBI adopted a fingerprint image compression scheme called the wavelet/scalar quantization (WSQ). This scheme can achieve a 15:1 (up to) compression ratio at the expense of the quality of decompressed images [BrBH93].

In this chapter, fingerprint, HVS, and FBI specification will be interpreted briefly.

Figure 4-1: A Fingerprint Card
4.1 Types of Fingerprints and Their Interpretation

A fingerprint is the mark that the ridges of skin on fingers and thumbs leave on objects touched. Our fingerprints are perfectly formed seven months before we are born. The fingerprint is unique. In this world of approximately 6.3 billion humans, each with ten fingerprints, there is not one fingerprint exactly like another. Even twins who look exactly alike have different fingerprints.

Fingerprints may be resolved into the following three large general groups of types: loop, arch and whorl [Unit85]. Each group bears the same general characteristics or family resemblance. The types may be further divided into subgroups by means of the smaller differences existing between the types in the same general group. Table 4-1 lists the groups and subgroups. Statistical data show that loops occur in about 65 percent of all fingerprints, whorls in about 30 percent, and arches in the remaining 5 percent [Coll85].

<table>
<thead>
<tr>
<th>ARCH</th>
<th>LOOP</th>
<th>WHORL</th>
</tr>
</thead>
<tbody>
<tr>
<td>Plain arch</td>
<td>Radial loop</td>
<td>Plain whorl</td>
</tr>
<tr>
<td>Tented arch</td>
<td>Ulnar loop</td>
<td>Central pocket loop</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Double loop</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Accidental whorl</td>
</tr>
</tbody>
</table>

Table 4-1: Types of fingerprint

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.
Figure 4-2: Three major types of fingerprint:
(a) Arch, (b) Loop, and (c) Whorl.

We first review a few technical terms used in fingerprint work. Figure 4-2 is a simple sketch to show the characteristics of the three fingerprint types. A pattern area is enclosed by type lines. The pattern area can have cores, deltas, and ridges which are used in classifying fingerprints. Type lines may be defined as the two innermost ridges which start parallel, diverge, and surround or tend to surround the pattern area. The thick lines A and B in figure 4-2-b are the type lines. Delta is the first ridge or part of a ridge nearest the point of divergence of the two type lines. The core is placed upon or within the innermost sufficient recurve. It is important to concern ourselves with the core of the loop type only. For loops and whorls, the pattern area is used for classification and identification. On the other hand, arches or tented arches have no particular boundary to define the pattern area [Unit85].

A core or sufficient recurve, a delta, and a ridge count across a looping ridge are the three essential features of a loop type. In the whorl type, at least two deltas are present with a recurve in front of each delta. The arch type has no pattern area. The accidental whorl is a pattern consisting of a combination of two different types of pattern, with the
exception of the plain arch, which has two or more deltas, or a pattern which possesses some of the requirements for two or more different types, or a pattern which conforms to none of the above definitions [Unit85].

4.2 A Model of the Human Visual System

The physical properties of the optical transmission pathway through the iris to the retina produce a lowpass spectral response which, when combined with the highpass characteristic due to interconnection of the receptors gives an overall band-pass response. Associated with this spatial response is a logarithmic amplitude non-linearity due to adaptation to background luminance necessary for the eye to function over a wide range of average scene intensities. The fact that the eye has preferred regions of spatial frequency response and a non-linear amplitude response mechanism can be utilized to develop better coding algorithms [Clar96].

Figure 4-3 shows gray-scale bars with (a) 32 levels and (b) 256 levels, respectively. Although the intensity of the stripes is constant, we actually perceive a brightness pattern that is strongly scalloped, especially near the boundaries. This phenomenon is called the Mach band effect [Prat92]. Although there are 32 gray-scales in figure 4-3-a, the HVS cannot precisely distinguish them, especially near both ends of the chart. That is caused by the band-pass characteristic of the HVS.
The band-pass spatial frequency response of the eye has led to numerous attempts to improve coding efficiency by preferentially allocating bits to the frequency region (or to the corresponding transform coefficients) to which the eye is most sensitive. The transform coefficients corresponding to the most sensitive part of the HVS spatial response are preferentially weighted with respect to the others and so receive a higher bit allocation (i.e., more accurate quantization) than others [Clar96]. Such data encoded for human perception was referred to as signals in FBI specifications.

4.3 FBI Gray-scale Fingerprint Compression Specification

FBI’s criterion for resolution of their digitized fingerprint images is at 500 dots per inch with 8 bits of grayscale. At this rate, a single fingerprint card turns into about 10MB
of data. Thus, the total size of the digitized collection would be more than 2000 terabytes (a terabyte is $2^{40}$ bytes).

Compression is, therefore, a must. After carefully research, The FBI’s Criminal Justice Information Services Division and researchers at Los Alamos National Laboratory and the National Institute of Standards and Technology have developed the national standards for digitization [ANSt93] and lossy image compression [HBBr93]. The algorithm involves three steps: (1) a 64-subband discrete wavelet transform, (2) an adaptive scalar quantization of wavelet transform coefficients, and (3) a two-pass Huffman coding of the quantization indices. This is the reason for the name wavelet/scalar quantization, or WSQ.

![WSQ Encoder and Decoder Diagram](image)

*Figure 4-4: DWT-based encoder and decoder diagram [FBI93]*

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.
Figure 4-4 shows the main procedures for WSQ encoding and decoding. The same tables specified for an encoder to use to compress a particular image must be provided to a decoder to reconstruct that image.

4.3.1 The 64-subbands Discrete Wavelet Packet Transform

The first step in the fingerprint compression is a symmetric discrete wavelet packet transform using the symmetric filter coefficients lists in Table 4-2. They are symmetric filters with 7 and 9 impulse response taps.

<table>
<thead>
<tr>
<th>Tap</th>
<th>Approximate Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>$h(0)$</td>
<td>0.85269867900940</td>
</tr>
<tr>
<td>$h(\pm 1)$</td>
<td>0.37740285561265</td>
</tr>
<tr>
<td>$h(\pm 2)$</td>
<td>-0.110624404418420</td>
</tr>
<tr>
<td>$h(\pm 3)$</td>
<td>-0.023849465019380</td>
</tr>
<tr>
<td>$h(\pm 4)$</td>
<td>0.037828455506995</td>
</tr>
<tr>
<td>$g(-1)$</td>
<td>0.78848561640566</td>
</tr>
<tr>
<td>$g(-2, 0)$</td>
<td>-0.418092273222210</td>
</tr>
<tr>
<td>$g(-3, 1)$</td>
<td>-0.040689417609558</td>
</tr>
<tr>
<td>$g(-4, 2)$</td>
<td>0.064538882628938</td>
</tr>
</tbody>
</table>

The fingerprint image is decomposed into 64 subbands. Each of these subbands represents information in a particular frequency band. This kind of image decomposition is different to Mallat’s pyramid method. It can be called wavelet packet transform. The subbands organization is show in Figure 4-5. The wavelet transform is first applied to image rows and columns, resulting in $4 \times 4 = 16$ subbands. The SWT is then applied in the same manner to three of the 16 subbands, decomposing each into 16 smaller subbands. The last step is to decompose the top-left subband into four smaller ones.
Figure 4-5: Organization of Subbands in the WSQ Specification

The larger subbands (51-63) contain the fine-detail, high-frequency information of the image. They can later be heavily quantized without loss of any important information (i.e., information needed to classify and identify fingerprints). In fact, subbands 60-63 are completely discarded. Subbands 7-18 are important. They contain the portion of the image frequencies that corresponds to the ridges in a fingerprint. This information is important and should be quantized lightly.

At the decoder, the synthesis filters for inverse wavelet transform are completely determined by the following anti-aliasing relations:

\[ \tilde{h}(n) = (-1)^n g(n - 1) \]  
\[ \tilde{g}(n) = (-1)^{n-1} h(n - 1) \]  

\( (4.1) \)  
\( (4.2) \)
4.3.2 An Adaptive Scalar Quantization

The transform coefficients in the 64 subbands are floating-point numbers. After the subband decomposition is computed, the resulting coefficients are quantized uniformly within subbands. The quantizer zero bin width and step size for each subband are contained in the quantization table. A quantization step size of zero ($Q_k = 0$) indicates that all coefficients within the subband are zero and the subband is not transmitted. The following equations apply to the wavelet coefficients $a_k(m, n)$ in subband $k$.

\[
p_k(m, n) = \begin{cases} 
\left\lfloor \frac{a_k(m, n) - Z_k/2}{Q_k} \right\rfloor + 1, & a_k(m, n) > \frac{Z_k}{2}, \\
0, & -\frac{Z_k}{2} \leq a_k(m, n) \leq \frac{Z_k}{2}, \\
\left\lfloor \frac{a_k(m, n) + Z_k/2}{Q_k} \right\rfloor + 1, & a_k(m, n) < -\frac{Z_k}{2}.
\end{cases}
\] (4.3)

At the decoder, the following equation dequantizes the indices. The value $C$ determines the quantization bin centers.

\[
\hat{a}_k(m, n) = \begin{cases} 
(p_k(m, n) - C) * Q_k + Z_k/2, & p_k(m, n) > 0, \\
0, & p_k(m, n) = 0, \\
(p_k(m, n) + C) * Q_k - Z_k/2, & p_k(m, n) < 0.
\end{cases}
\] (4.4)

$Z_k$ is the width of the center (zero) quantization bin. $Q_k$ is the width of the nonzero quantization bins in the $k^{th}$ subband.

4.3.3 A Two-pass Huffman Coding

After quantization, the subbands are concatenated into three blocks for adaptive Huffman Coding. The first block consists of the low- and mid-frequency subbands 0-18. The second and third blocks contain the highpass detail subbands 19-51 and 52-59, respectively. And, the subbands 60-63 are completely discarded. The quantization bin
indices in each block are first run length coded for zero-runs and mapped to a finite alphabet of symbols in Table 4-3.

After symbolized, the Huffman tables will be constructed. This is a two-pass job. In the first pass, code tables are determined for the blocks. In another pass, the Huffman tables are encoded. In FBI specification, two Huffman tables are calculated for each fingerprint image. The first is based on the symbol frequencies for bands 0 through 18. The second Huffman table is based on symbol frequencies for subbands 19 through 59. Using these tables, the fingerprint image will be saved a compressed format.

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>zero run of length 1</td>
</tr>
<tr>
<td>2</td>
<td>zero run of length 2</td>
</tr>
<tr>
<td>3</td>
<td>zero run of length 3</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>100</td>
<td>zero run of length 100</td>
</tr>
<tr>
<td>101</td>
<td>escape for positive 8 bit quantizer bin index</td>
</tr>
<tr>
<td>102</td>
<td>escape for negative 8 bit quantizer bin index</td>
</tr>
<tr>
<td>103</td>
<td>escape for positive 16 bit quantizer bin index</td>
</tr>
<tr>
<td>104</td>
<td>escape for negative 16 bit quantizer bin index</td>
</tr>
<tr>
<td>105</td>
<td>escape for 8 bits zero run length</td>
</tr>
<tr>
<td>106</td>
<td>escape for 16 bits zero run length</td>
</tr>
<tr>
<td>107</td>
<td>quantizer bin index value -73</td>
</tr>
<tr>
<td>108</td>
<td>quantizer bin index value -72</td>
</tr>
<tr>
<td>109</td>
<td>quantizer bin index value -71</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>180</td>
<td>Not used. Use symbol 1.</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>253</td>
<td>quantizer bin index value 73</td>
</tr>
<tr>
<td>254</td>
<td>quantizer bin index value 74</td>
</tr>
</tbody>
</table>

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.
4.4 Summary

This chapter presented the WSQ gray-scale fingerprint image compression specification adopted by FBI. It is a lossy compression specification. FBI wants to impose more strict criteria on compressing a fingerprint image than other normal images because the decompressed fingerprint images may be used in court as evidence. However, it does not mean that a lossless compression is required. FBI still chooses this lossy compression technique for two reasons: First, to achieve a higher compression ratio; Second, to keep the acceptable quality by considering the limitations of the human visual system (HVS). In the next chapter, we focus on the wavelet transforms as the basis for fingerprint compression. The filter-banks-based transform scheme is compared with a fast lifting wavelet transform, as a basis for the hardware design.
CHAPTER 5

SOFTWARE IMPLEMENTATION

As discussed in Chapter 3, the lifting scheme provides a new approach to generate wavelets for transform. The integer-to-integer transform, one of the advanced features of lifting scheme, could make the transform more efficient and inverse transform easier. In this chapter, we will further discuss the feasibility of VLSI architecture for WPT using lifting scheme to transform data integer to integer. We will show that although integer-to-integer lifting scheme based transform causes some more loss of minutiae of the fingerprint images than standard convolution filter banks transform does, the reconstructed image quality is still acceptable. PSNR values are only slightly lower. The changes are undetectable by HVS. And, most important, it will not influence matching of the fingerprints.

A software package written in Matlab is developed to simulate three different transform algorithms.

The first one applies the current FBI standard. In this case, WPT is implemented by a 9-tap lowpass filter and a 7-tap highpass filter, which are convolved with the image rows and columns. We call it Bior9-7 in this thesis. A fingerprint image is convolved with this filter pair. The odd samples of the filtered outputs are discarded resulting in
down-sampling the image by a factor of two. To achieve a multiresolution WPT, this filter pair is used to decompose the desired image subbands iteratively. In WSQ, a 5-level decomposition results in 64 image subbands.

Second is the lifting based transform algorithm referred to as LS9-7. This scheme is derived and factorized from Bior9-7. A fingerprint image is decomposed by a set of lifting steps simply. The same 5-level 64 subbands decomposition is achieved. Furthermore, LS9-7 is modified to use integer arithmetic, which can perform an integer-to-integer transform called LS9-7I – the third algorithm.

Quality of reconstructed images by the three algorithms is compared, measured by the traditional distortion measure: the Peak Signal-to-Noise Ratio (PSNR) between the original and reconstructed images.

A fingerprint identification engine called VeriFinger [Neur03] is used to test matching of the reconstructed images with the original one.

In this chapter, we will also show that lifting scheme based algorithms (LS9-7 and LS9-7I) need less arithmetic operations than the standard convolution filter banks algorithm (Bior9-7) to compute the wavelet transform.

5.1 Transformation Algorithms

Image data are usually acquired in a serial manner. In fingerprint processing, a very common way to acquire image is adopted, which is to scan a fingerprint one line at a time. Therefore, wavelet transform for two-dimensional signals, such as fingerprint image, can be separated into two steps. The first step is a transform dealing with rows of the original image. The second step is another transform dealing with columns of the image.
produced by the first step. As shown in Figure 5-1, first, a one-dimensional transform deals with rows and decomposes each row to an approximation subband (A) in left side and a detailed subband (D) in right side. Second, this one-dimensional transform is applied to the image obtained in the first step column by column. Consequently, the whole decomposition results in an approximation image subband (AA) and three detailed image subbands in horizontal (DA), vertical (AD) and diagonal (DD) directions. Based on the above analysis, a one-dimensional algorithm is developed and then extended to a two-dimensional one. Furthermore, a software package is built to perform the WPT.

![Figure 5-1: A basis two-dimensional wavelet decomposition](image)

5.1.1 One-dimensional Wavelet Transform

**Standard Convolution Filter Banks (Bior9-7)**

The first transform algorithm (Bior9-7) is adopted by the current FBI standard. This filter bank pair consists of a 9-tap lowpass filter and a 7-tap highpass filter, which are convolved with signals.
Assuming an input sequence \( X = \{x_0, x_1, ..., x_{N-1}\} \) and a filter with coefficients \( H = \{h_0, h_1, ..., h_{M-1}\} \), the operation of filter can be expressed as a convolution function
\[
Y(k) = \sum_m H(m) X(k - m).
\] (5.1)

Figure 5-2: The 1D_FILT procedure

Based on equation 5.1, a procedure, 1D_FILT, as described in Figure 5-2 is built. There are two nested loops in the procedure. The inner one operates the convolution of filter coefficient with the elements in the input signals. Consequently, the filtering results are downsampled by the DSAMPLE procedure. As illustrated in Figure 5-3, the DSAMPLE procedure takes a signal \( Y(i) \) as an input and produces an output \( Z(i) = Y(2i) \).
The WSQ biorthogonal wavelet system decomposes an input sequence \( X = \{ x_0, x_1, ..., x_{N-1} \} \) and downsamples the filtered results into a lowpass component \( A = \{ a_0, a_2, ..., a_{N-2} \} \) and a highpass component \( D = \{ d_0, d_2, ..., d_{N-2} \} \), corresponding to wavelet and scaling coefficients, which can be represented as:

\[
a_k = \downarrow \left( \sum_n h(2k - n)x_n \right), \tag{5.2}
\]

\[
d_k = \downarrow \left( \sum_n g(2k - n)x_n \right). \tag{5.3}
\]

Therefore, two procedures are required for lowpass- and highpass- filtering functions, respectively. As described in Figure 5-4, a 1D_LP procedure applies the filtering operation (1D_FILT) with a 9-tap lowpass filter coefficients \( H = \{ h_4, ..., h_0, ..., h_4 \} \) and a downsampling operation (DSAMPLE) to the input 1D signal array, and a 1D_HP procedure applies the filtering operation (1D_FILT) with a 7-tap highpass filter coefficients \( G = \{ g_3, ..., g_0, ..., g_3 \} \) and another downsampling operation (DSAMPLE) to the same input 1D signal array as the 1D_LP procedure.
Consequently, the 1D_WT procedure performs the wavelet decomposition of the input one-dimensional array, as illustrated in Figure 5-5. It outputs an array \( Y \), with the same length as the input array \( X \).
In the synthesis stage, the wavelet coefficient component \( a = \{a_0, a_2, ..., a_{N-2}\} \) and a scaling coefficient component \( b = \{b_0, b_2, ..., b_{N-2}\} \) are upsampled, which is interpolated by zero, and past through the corresponding synthesis filter pair, respectively. Then, the synthesis system adds the filtered coefficients up to reconstruct the input signal \( \tilde{x} \). It can be represented as:

\[
\tilde{x}(k) = \left( \sum_n \tilde{h}(2k - n)(\uparrow a) \right) + \left( \sum_n \tilde{g}(2k - n)(\uparrow b) \right),
\]

The procedures can be expressed as:

/* Lowpass Filter*/
Step 1: Upsample the wavelet coefficient vector, \( a \)
Step 2: Get \( \tilde{a} \) by convolving the new vector with lowpass synthesis filter Coefficients

/* Highpass Filter*/
Step 3: Upsample the scaling coefficient vector, \( b \)
Step 4: Get \( \tilde{b} \) by convolving the new vector with highpass synthesis filter coefficients
Step 5: Get \( \tilde{x} \) by adding \( \tilde{a} \) to \( \tilde{b} \)

**LS-based algorithm (LS9-7)**

In hardware implementation, lifting scheme (LS) provides a simple and fast approach to build the wavelet transform system as introduced in Chapter 3. And those existing classical wavelets can be implemented with LS by factorizing them into Lifting steps. We will describe an algorithm for LS-based wavelet transform system, especially for WT in WSQ. The first step is to factorize filters coefficients into LS.
Using the factorization algorithm described in [DaSw98], the polyphase matrix for filter banks can be transformed into a sequence of alternating upper and lower triangular matrices and a diagonal matrix. Thus, the 9-7 filters for WSQ can be expressed as

$$P(z) = \begin{bmatrix} 1 & \alpha (1 + z^{-1}) \\ 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 \\ \beta (1 + z) & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 \\ \delta (1 + z) & 1 \end{bmatrix} \begin{bmatrix} K & 0 \\ 0 & 1/K \end{bmatrix}. \quad (5.2)$$

$\alpha, \beta, \gamma, \delta,$ and $K$ in equations are constants derived from the coefficients of the analysis (low pass) filter defined. Assume the coefficients of the analysis (low pass) filter as $H = \{ h_{-4}, h_3, ..., h_4, h_4 \}$ with $h_n = h_n$. The coefficients of the remainders are computed as:

$$r_0 = h_0 - 2 \times h_4 \times h_1 / h_3$$
$$r_1 = h_2 - h_4 - h_4 \times h_1 / h_3$$
$$s_0 = h_1 - h_3 - h_3 \times r_0 / r_1$$
$$t_0 = r_0 - 2 \times r_1 \quad (5.3)$$

Then constants are defined as:

$$\alpha = h_4 / h_3;$$
$$\beta = h_3 / r_1;$$
$$\gamma = r_1 / s_0;$$
$$\delta = s_0 / t_0;$$
$$K = t_0 = r_0 - 2 \times r_1; \quad (5.4)$$

Substituting the coefficients in equations (5.3) and (5.4) by values shown in TABLE 4-2, we get the constants: $\alpha \approx -1.586134342, \beta \approx -0.052980118, \gamma \approx 0.882911076, \delta \approx 0.443506852,$ and $K \approx 1.14960439.$

The second step applies equation 5.2. The forward discrete wavelet transform with finite filters is decomposed into a finite sequence of simple steps, which are four "lifting"
steps followed by two "scaling" steps. This leads to the following pseudo-code
implementation of this forward wavelet transform using lifting:

for \( n = 1 \) to \((\text{lengthOfVector}/2)\)

\[
y_{2n + 1} = x_{2n + 1} + \alpha \times (x_{2n} + x_{2n + 2}); \quad [S1]
y_{2n} = x_{2n} + \beta \times (y_{2n - 1} + y_{2n + 1}); \quad [S2]
y_{2n + 1} = x_{2n + 1} + \gamma \times (y_{2n} + y_{2n + 2}); \quad [S3]
y_{2n} = x_{2n} + \delta \times (y_{2n - 1} + y_{2n + 1}); \quad [S4]
y_{2n + 1} = K \times y_{2n + 1}; \quad [S5]
y_{2n} = (1/K) \times y_{2n}; \quad [S6]
\]
end for.

And the inverse transform is given by:

for \( n = 1 \) to \((\text{lengthOfVector}/2)\)

\[
x_{2n} = K \times y_{2n}; \quad [S1]
x_{2n + 1} = (1/K) \times y_{2n + 1}; \quad [S2]
x_{2n} = x_{2n} - \delta \times (x_{2n - 1} + x_{2n + 1}); \quad [S3]
x_{2n + 1} = x_{2n + 1} + \gamma \times (x_{2n} + x_{2n + 2}); \quad [S4]
x_{2n} = x_{2n} + \beta \times (x_{2n - 1} + x_{2n + 1}); \quad [S5]
x_{2n + 1} = x_{2n + 1} + \alpha \times (x_{2n} + x_{2n + 2}); \quad [S6]
\]
end for.

**Integer LS-based algorithm (LS9-7I)**

The arithmetic of the previous two schemes is all based on floating-point. It will increase the complexity in hardware design. As one of the advanced features in LS, integer DWT can be used to reduce the complexity. In order to test the feasibility of integer-to-integer transform for fingerprint, we modify the LS-based algorithms presented above.
In each lifting step, a round operation is introduced to the result of the product right before adding to or subtracting from the corresponding pixel value. It means the products are rounded to the next high integer. For instance, numbers $\geq 266.50$ are rounded to 267, and numbers $< 266.50$ are rounded to 266. Based on this rule, the following modified pseudo-code implements an integer wavelet transform using lifting [CDSY98]:

\[
\begin{align*}
\text{for } n = 1 \text{ to } (\text{lengthOfVector}/2) \\
y(2n + 1) &= x(2n + 1) + \lceil \alpha \times [x(2n) + x(2n + 2)] + 1/2 \rceil; \quad [\text{S1}] \\
y(2n) &= x(2n) + \lceil \beta \times [y(2n - 1) + y(2n + 1)] + 1/2 \rceil; \quad [\text{S2}] \\
y(2n + 1) &= y(2n + 1) + \lceil \gamma \times [y(2n) + y(2n + 2)] + 1/2 \rceil; \quad [\text{S3}] \\
y(2n) &= y(2n) + \lceil \delta \times [y(2n - 1) + y(2n + 1)] + 1/2 \rceil; \quad [\text{S4}]
\end{align*}
\]

end for.

And for the inverse transform, we just flip the signs and reverse the operations in the integer forward transform:

\[
\begin{align*}
\text{for } n = 1 \text{ to } (\text{lengthOfVector}/2) \\
x(2n) &= y(2n) - \lceil \delta \times [y(2n - 1) + y(2n + 1)] + 1/2 \rceil; \quad [\text{S1}] \\
x(2n + 1) &= y(2n + 1) - \lceil \gamma \times [x(2n) + x(2n + 2)] + 1/2 \rceil; \quad [\text{S2}] \\
x(2n) &= x(2n) - \lceil \beta \times [x(2n - 1) + x(2n + 1)] + 1/2 \rceil; \quad [\text{S3}] \\
x(2n + 1) &= x(2n + 1) - \lceil \alpha \times [x(2n) + x(2n + 2)] + 1/2 \rceil; \quad [\text{S4}]
\end{align*}
\]

end for.

5.1.2 Two-dimensional Transform

The two-dimensional transform is performed by applying the one-dimensional transform algorithm consecutively on the rows and columns of a two-dimensional signal, such as a fingerprint image. Considering this fingerprint image with dimensions $X$ and $Y$ and denoting $i_X$ and $j_Y$ as the number of columns and rows in $x$ and $y$ direction,

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.
respectively, the forward two-dimensional transform algorithm for one level WT is as follows:

For the forward two-dimensional wavelet transform, we begin with iteration level 0 as current level. The one-dimensional forward transform is first applied to all the rows, and then to all the columns. Subsequently, we move to the next iteration level and repeat the two steps above, and so on, until all the iteration levels are accomplished. In this thesis, five-level decompositions are required.

For the inverse transform, the inverse one-dimensional transform is applied exactly in the opposite order. Flowcharts below illustrate the two-dimensional forward transform and inverse algorithm.

The flowchart in Figure 5-6a shows the 2D_FDWT procedure which decomposes an image or low-level subband into four next-level subbands. As shown in Figure 5-6b, the 2D_IDWT synthesizes an image or low-level subband from four high-level subbands. Notes that 1D_WT procedure and 1D_IWT procedure use three different algorithms corresponding to three testing schemes as discussed in section 5.2.
5.1.3 Decomposition Scheme

In WSQ, the decomposition scheme for the fingerprint images is different to the scheme used by a normal wavelet transform and full wavelet packet transform. It is a five-level decomposition scheme. Different subbands are decomposed level by level.

Figure 5-6: Flow chart of (a) two-dimensional forward wavelet transform

and (b) two-dimensional inverse wavelet transform
Since not all subbands are processed, this transform can be called the partial wavelet packet transform.

![Decomposition scheme in WSQ](image)

Figure 5-7: Decomposition scheme in WSQ

The two-dimensional wavelet transform, which was introduced in section 5.3, can be used to transform the required subbands in WSQ. At the first level, the original fingerprint image is processed by the 2D_FDWT procedure, resulting in $2 \times 2 = 4$ subbands. After the first level of decomposition, all of the resulting four subbands are sent back through the filter bank to further split the subbands into $4 \times 4 = 16$ subbands at the second level. The transform system is then applied in the same manner to three of the 16 subbands, decomposing each into 16 smaller subbands at the third and fourth level. At the fifth level analysis, the top-left subband is decomposed into four smaller ones.
The desired subband structure is obtained, which consists of 64-subbands. Figure 5-7 shows these procedures.

5.2 Experimental Results and Performance Analysis

Two experiments are done. In the first experiment, we want to find out the loss caused by transform only. So, the compression and decoding steps are skipped. We reconstruct the images from wavelet coefficients obtained from transform. In the second experiment, we want to find out the total loss caused by both transform and compression. We apply a compression step to the wavelet coefficients and then reconstruct the images from the compressed data.

5.2.1 Image Quality Measures and Test Image

Distortion Measure

A standard objective measure of distortion of the reconstructed images is Peak Signal-to-Noise Ratio (PSNR). Here, for a 256 gray scale fingerprint image, the PSNR is defined as

$$PSNR = 10 \log_{10} \left( \frac{255^2}{MSE} \right) \text{ [in dB]}.$$  \hspace{1cm} (5.5)

MSE is the mean square error, which is the average of the energy of the difference between the original image and the reconstructed image:

$$MSE = \frac{\sum_{i=0}^{N} \sum_{j=0}^{M} [x(i, j) - \tilde{x}(i, j)]^2}{N \times M}$$ \hspace{1cm} (5.6)

where $N \times M$ is the total number of pixels in the image, $x(i, j)$ is an original pixel, and $\tilde{x}(x, j)$ is a reconstructed pixel. PSNR values range from infinity, for identical images,
to 0, for images with no commonality. Typical PSNR values for images range from 20 to 40. A PSNR value of 30 or greater is considered to be acceptable for subject recognition [GeKi96]. PSNR drops as the compression ratio increases on an image. Though PSNR is not a direct measure of the perceptual visual quality, it is widely used as an indicator of the image quality.

**Fingerprints Matching**

There are many interruptions of ridge across the impression of fingerprints. These interruptions – ridge endings and bifurcations are known as minutiae. The definitive information used to determine that one fingerprint matches another is in the fine detail of these minutiae and their relationships. There are three steps for fingerprint images matching generally.

First step, a fingerprint image is processed through a series of image processing algorithms to obtain a clear unambiguous skeletal image of the original gray tone impression, clarifying smudged areas, removing extraneous artifacts and healing most scars, cuts and breaks.

Second step, minutiae within the skeletal image are identified and encoded, providing critical placement, orientation and linkage information for the matcher.

Third step, the matcher compares data of the input search image against all records in the database to determine if a probable match exists. If yes, it indicates matching and gives file name(s) of the matched image(s). If no, it indicates matching fails.

The fingerprint identification engine that we use to test matching of the reconstructed fingerprint images with the original one is called VeriFinger, offered by Neurotechnologija, Ltd. [Neur03].
Test Fingerprint Images

The set of 17 test fingerprint images, which are the reference test set created from the National Institute of Standards and Technology (NIST) reference implementation, are downloaded from the NIST website [NIST03]. They are in different sizes, shown in Table 5-1.

Table 5-1: Test Fingerprint Images and Their Sizes

<table>
<thead>
<tr>
<th>Fingerprint File Name</th>
<th>Width (pixels)</th>
<th>Height (pixels)</th>
</tr>
</thead>
<tbody>
<tr>
<td>FingerP1.raw</td>
<td>589</td>
<td>605</td>
</tr>
<tr>
<td>FingerP2.raw</td>
<td>832</td>
<td>768</td>
</tr>
<tr>
<td>FingerP3.raw</td>
<td>832</td>
<td>768</td>
</tr>
<tr>
<td>FingerP4.raw</td>
<td>815</td>
<td>752</td>
</tr>
<tr>
<td>FingerP5.raw</td>
<td>832</td>
<td>768</td>
</tr>
<tr>
<td>FingerP6.raw</td>
<td>832</td>
<td>768</td>
</tr>
<tr>
<td>FingerP7.raw</td>
<td>545</td>
<td>638</td>
</tr>
<tr>
<td>FingerP8.raw</td>
<td>800</td>
<td>750</td>
</tr>
<tr>
<td>FingerP9.raw</td>
<td>512</td>
<td>678</td>
</tr>
<tr>
<td>FingerP10.raw</td>
<td>375</td>
<td>526</td>
</tr>
<tr>
<td>FingerP11.raw</td>
<td>614</td>
<td>717</td>
</tr>
<tr>
<td>FingerP12.raw</td>
<td>516</td>
<td>716</td>
</tr>
<tr>
<td>FingerP13.raw</td>
<td>539</td>
<td>651</td>
</tr>
<tr>
<td>FingerP14.raw</td>
<td>466</td>
<td>578</td>
</tr>
<tr>
<td>FingerP15.raw</td>
<td>436</td>
<td>670</td>
</tr>
<tr>
<td>FingerP16.raw</td>
<td>666</td>
<td>758</td>
</tr>
<tr>
<td>FingerP17.raw</td>
<td>521</td>
<td>666</td>
</tr>
</tbody>
</table>

5.2.2 Experiments and Results

First Experiment

FBI’s specification only requires a lossy compression. Thus, in the encoding stage, the last four subbands, indexed 60 to 63, are discarded and the reconstructed fingerprint
image is carried out by the first 60 subbands. For the sake of comparison here, we reconstructed the image not only from the first 60 subbands but also from all of the 64 subbands. The PSNR values, for the 17 test fingerprint images using three algorithms, are given in Table 5-2 and Table 5-3, respectively.

Table 5-2: PSNRs of Reconstructed Fingerprint Images from 64 Subbands

<table>
<thead>
<tr>
<th>File Name</th>
<th>Bior9-7</th>
<th>LS9-7</th>
<th>LS9-7I</th>
</tr>
</thead>
<tbody>
<tr>
<td>FingerP1</td>
<td>+133.35 dB</td>
<td>+310.85 dB</td>
<td>infinite</td>
</tr>
<tr>
<td>FingerP2</td>
<td>+133.32 dB</td>
<td>+311.21 dB</td>
<td>infinite</td>
</tr>
<tr>
<td>FingerP3</td>
<td>+139.56 dB</td>
<td>+317.28 dB</td>
<td>infinite</td>
</tr>
<tr>
<td>FingerP4</td>
<td>+135.23 dB</td>
<td>+312.67 dB</td>
<td>infinite</td>
</tr>
<tr>
<td>FingerP5</td>
<td>+133.37 dB</td>
<td>+310.90 dB</td>
<td>infinite</td>
</tr>
<tr>
<td>FingerP6</td>
<td>+132.68 dB</td>
<td>+310.54 dB</td>
<td>infinite</td>
</tr>
<tr>
<td>FingerP7</td>
<td>+133.65 dB</td>
<td>+311.65 dB</td>
<td>infinite</td>
</tr>
<tr>
<td>FingerP8</td>
<td>+135.43 dB</td>
<td>+312.94 dB</td>
<td>infinite</td>
</tr>
<tr>
<td>FingerP9</td>
<td>+135.16 dB</td>
<td>+312.78 dB</td>
<td>infinite</td>
</tr>
<tr>
<td>FingerP10</td>
<td>+134.11 dB</td>
<td>+312.17 dB</td>
<td>infinite</td>
</tr>
<tr>
<td>FingerP11</td>
<td>+135.30 dB</td>
<td>+313.60 dB</td>
<td>infinite</td>
</tr>
<tr>
<td>FingerP12</td>
<td>+134.84 dB</td>
<td>+312.93 dB</td>
<td>infinite</td>
</tr>
<tr>
<td>FingerP13</td>
<td>+136.67 dB</td>
<td>+314.49 dB</td>
<td>infinite</td>
</tr>
<tr>
<td>FingerP14</td>
<td>+135.55 dB</td>
<td>+312.94 dB</td>
<td>infinite</td>
</tr>
<tr>
<td>FingerP15</td>
<td>+134.38 dB</td>
<td>+311.85 dB</td>
<td>infinite</td>
</tr>
<tr>
<td>FingerP16</td>
<td>+134.38 dB</td>
<td>+311.85 dB</td>
<td>infinite</td>
</tr>
<tr>
<td>FingerP17</td>
<td>+135.35 dB</td>
<td>+313.05 dB</td>
<td>infinite</td>
</tr>
</tbody>
</table>

From Table 5-2, in case of images reconstructed from all 64 subbands, we can see that Bior9-7 achieves PSNR values ranging from 132.68 dB to 139.56 dB, and LS9-7 achieves PSNR values ranging from 310.54 dB to 317.28 dB. LS9-7 can reconstruct the image more precisely than the Bior9-7 can. LS9-7I can even reconstruct the test images...
perfectly – identical to the original images. In this case, the two LS-based transforms perform better than Bior9-7.

Table 5-3: PSNRs of Reconstructed Fingerprint Images from Subbands 0 to 59

<table>
<thead>
<tr>
<th>File Name</th>
<th>Bior9-7</th>
<th>LS9-7</th>
<th>LS9-7I</th>
</tr>
</thead>
<tbody>
<tr>
<td>FingerP1</td>
<td>+48.71 dB</td>
<td>+48.71 dB</td>
<td>+46.67 dB</td>
</tr>
<tr>
<td>FingerP2</td>
<td>+53.51 dB</td>
<td>+53.51 dB</td>
<td>+50.25 dB</td>
</tr>
<tr>
<td>FingerP3</td>
<td>+60.14 dB</td>
<td>+60.14 dB</td>
<td>+53.72 dB</td>
</tr>
<tr>
<td>FingerP4</td>
<td>+57.26 dB</td>
<td>+57.26 dB</td>
<td>+52.30 dB</td>
</tr>
<tr>
<td>FingerP5</td>
<td>+59.19 dB</td>
<td>+59.19 dB</td>
<td>+52.10 dB</td>
</tr>
<tr>
<td>FingerP6</td>
<td>+51.70 dB</td>
<td>+51.70 dB</td>
<td>+48.97 dB</td>
</tr>
<tr>
<td>FingerP7</td>
<td>+53.04 dB</td>
<td>+53.04 dB</td>
<td>+49.13 dB</td>
</tr>
<tr>
<td>FingerP8</td>
<td>+45.28 dB</td>
<td>+45.28 dB</td>
<td>+44.20 dB</td>
</tr>
<tr>
<td>FingerP9</td>
<td>+43.53 dB</td>
<td>+43.53 dB</td>
<td>+42.53 dB</td>
</tr>
<tr>
<td>FingerP10</td>
<td>+45.74 dB</td>
<td>+45.74 dB</td>
<td>+44.29 dB</td>
</tr>
<tr>
<td>FingerP11</td>
<td>+51.88 dB</td>
<td>+51.88 dB</td>
<td>+48.51 dB</td>
</tr>
<tr>
<td>FingerP12</td>
<td>+46.88 dB</td>
<td>+46.88 dB</td>
<td>+45.20 dB</td>
</tr>
<tr>
<td>FingerP13</td>
<td>+42.65 dB</td>
<td>+42.65 dB</td>
<td>+41.80 dB</td>
</tr>
<tr>
<td>FingerP14</td>
<td>+43.08 dB</td>
<td>+43.08 dB</td>
<td>+42.13 dB</td>
</tr>
<tr>
<td>FingerP15</td>
<td>+53.99 dB</td>
<td>+53.99 dB</td>
<td>+49.50 dB</td>
</tr>
<tr>
<td>FingerP16</td>
<td>+53.99 dB</td>
<td>+53.99 dB</td>
<td>+49.50 dB</td>
</tr>
<tr>
<td>FingerP17</td>
<td>+46.33 dB</td>
<td>+46.33 dB</td>
<td>+44.62 dB</td>
</tr>
</tbody>
</table>

When reconstructing only from subbands 0 to 59, Bior9-7 and LS9-7 have same PSNR values for all 17 test fingerprint images ranging from 42.65 dB to 60.14 dB, as shown in Table 5-3. And LS9-7I, though a little underperformance, still has the ability to obtain “satisfactory” performance with PSNR values ranging from 41.80 dB to 53.72 dB.

In Figure 5-8 and Figure 5-9, two of the test fingerprint images (FingerP13 with the lowest PSNR - 41.80 dB by LS9-7I, and FingerP3 with the highest PSNR - 53.72 dB by
LS9-7I) and their reconstructed images by three algorithms are shown. In both cases, we can see that the original image and the three reconstructed images cannot be distinguished by human visual system (HVS).

Figure 5-8: Original and Reconstructed Images Using Subbands 0 to 59
(FingerP13, 539×651, Compression Skipped)
Figure 5-9: Original and Reconstructed Images Using Subbands 0 to 59
(FingerP3, 832×768, Compression Skipped)
Second Experiment

In the second experiment, we attach an entropy coder to both Bior9-7 and LS9-7I. The entropy coder is adopted from FBI WSQ standard for gray-scale fingerprint image compression [BrBH93]. So in this experiment, the 17 test fingerprint images are compressed following the WSQ standard – the compression steps (quantization step and Huffman coding) just deal with the first 60 subbands (0 to 59) of 64 subbands obtained in the decomposition step and the target bit rates range from 1.00bpp to 0.10bpp with interval of 0.05bpp. The compressed images are reconstructed and PSNRs to the original image are calculated. PSNR values of Bior9-7 are shown in Table 5-4. PSNR values of LS9-7I are shown in Table 5-5. The difference of PSNR values between Bior9-7 and LS9-7I (PSNR Diff) corresponding to different bit rates are shown in Figure 5-9. While PSNR of Bior9-7 is higher than that of LS9-7I, PSNR Diff is positive, otherwise, negative.
Table 5-4: PSNRs (dB) of Compressed Fingerprint Images Using Bior9-7

<table>
<thead>
<tr>
<th>Image File</th>
<th>1 bpp</th>
<th>0.95 bpp</th>
<th>0.9 bpp</th>
<th>0.85 bpp</th>
<th>0.8 bpp</th>
<th>0.75 bpp</th>
<th>0.7 bpp</th>
<th>0.6 bpp</th>
<th>0.55 bpp</th>
<th>0.5 bpp</th>
<th>0.45 bpp</th>
<th>0.4 bpp</th>
<th>0.35 bpp</th>
<th>0.3 bpp</th>
<th>0.25 bpp</th>
<th>0.2 bpp</th>
<th>0.15 bpp</th>
<th>0.1 bpp</th>
</tr>
</thead>
<tbody>
<tr>
<td>FingerP2</td>
<td>37.91</td>
<td>37.45</td>
<td>37.04</td>
<td>36.54</td>
<td>35.98</td>
<td>35.42</td>
<td>34.86</td>
<td>34.3</td>
<td>33.73</td>
<td>33.19</td>
<td>32.65</td>
<td>32.11</td>
<td>31.63</td>
<td>30.63</td>
<td>29.55</td>
<td>28.34</td>
<td>27.04</td>
<td>25.55</td>
</tr>
<tr>
<td>FingerP3</td>
<td>42.77</td>
<td>42.36</td>
<td>41.94</td>
<td>41.53</td>
<td>41.12</td>
<td>40.69</td>
<td>40.24</td>
<td>39.63</td>
<td>39.01</td>
<td>38.34</td>
<td>37.67</td>
<td>36.95</td>
<td>36.04</td>
<td>34.99</td>
<td>33.88</td>
<td>32.58</td>
<td>31.15</td>
<td>29.61</td>
</tr>
<tr>
<td>FingerP4</td>
<td>41.33</td>
<td>40.93</td>
<td>40.44</td>
<td>39.95</td>
<td>39.43</td>
<td>38.91</td>
<td>38.39</td>
<td>37.85</td>
<td>37.3</td>
<td>36.75</td>
<td>36.18</td>
<td>35.58</td>
<td>34.82</td>
<td>33.81</td>
<td>32.75</td>
<td>31.53</td>
<td>30.16</td>
<td>28.57</td>
</tr>
<tr>
<td>FingerP5</td>
<td>42.79</td>
<td>42.39</td>
<td>41.96</td>
<td>41.5</td>
<td>41.04</td>
<td>40.56</td>
<td>40.07</td>
<td>39.59</td>
<td>39.1</td>
<td>38.63</td>
<td>38.16</td>
<td>37.67</td>
<td>37.15</td>
<td>36.67</td>
<td>36.15</td>
<td>35.65</td>
<td>35.16</td>
<td>34.65</td>
</tr>
<tr>
<td>FingerP6</td>
<td>36.4</td>
<td>35.59</td>
<td>35.09</td>
<td>34.55</td>
<td>34</td>
<td>33.43</td>
<td>32.86</td>
<td>32.27</td>
<td>31.68</td>
<td>31.08</td>
<td>30.51</td>
<td>29.88</td>
<td>29.08</td>
<td>28.11</td>
<td>26.95</td>
<td>25.69</td>
<td>24.31</td>
<td>22.53</td>
</tr>
<tr>
<td>FingerP7</td>
<td>36.81</td>
<td>36.4</td>
<td>35.96</td>
<td>35.52</td>
<td>35.08</td>
<td>34.55</td>
<td>34.03</td>
<td>33.51</td>
<td>32.94</td>
<td>32.47</td>
<td>31.97</td>
<td>31.47</td>
<td>30.95</td>
<td>30.41</td>
<td>29.17</td>
<td>27.71</td>
<td>26.44</td>
<td>24.67</td>
</tr>
<tr>
<td>FingerP8</td>
<td>34.43</td>
<td>34.1</td>
<td>33.74</td>
<td>33.38</td>
<td>33.01</td>
<td>32.59</td>
<td>32.18</td>
<td>31.75</td>
<td>31.33</td>
<td>30.89</td>
<td>30.39</td>
<td>29.9</td>
<td>29.3</td>
<td>28.68</td>
<td>28.03</td>
<td>27.17</td>
<td>26.25</td>
<td>25.2</td>
</tr>
<tr>
<td>FingerP10</td>
<td>33.25</td>
<td>32.92</td>
<td>32.56</td>
<td>32.16</td>
<td>31.77</td>
<td>31.37</td>
<td>30.92</td>
<td>30.46</td>
<td>29.97</td>
<td>29.44</td>
<td>28.94</td>
<td>28.42</td>
<td>27.83</td>
<td>27.2</td>
<td>26.4</td>
<td>25.51</td>
<td>24.59</td>
<td>23.33</td>
</tr>
<tr>
<td>FingerP11</td>
<td>36.25</td>
<td>35.86</td>
<td>35.48</td>
<td>35.07</td>
<td>34.61</td>
<td>34.08</td>
<td>33.56</td>
<td>33.04</td>
<td>32.46</td>
<td>31.81</td>
<td>31.13</td>
<td>30.24</td>
<td>29.22</td>
<td>28.1</td>
<td>26.93</td>
<td>25.66</td>
<td>24.31</td>
<td>22.56</td>
</tr>
<tr>
<td>FingerP12</td>
<td>34.41</td>
<td>34.04</td>
<td>33.62</td>
<td>33.21</td>
<td>32.78</td>
<td>32.32</td>
<td>31.8</td>
<td>31.28</td>
<td>30.75</td>
<td>30.23</td>
<td>29.56</td>
<td>28.87</td>
<td>27.96</td>
<td>26.93</td>
<td>25.87</td>
<td>24.65</td>
<td>23.27</td>
<td>21.65</td>
</tr>
<tr>
<td>FingerP13</td>
<td>31.96</td>
<td>31.66</td>
<td>31.37</td>
<td>31.02</td>
<td>30.65</td>
<td>30.29</td>
<td>29.94</td>
<td>29.54</td>
<td>29.12</td>
<td>28.72</td>
<td>28.27</td>
<td>27.78</td>
<td>27.3</td>
<td>26.73</td>
<td>26.15</td>
<td>25.49</td>
<td>24.67</td>
<td>23.76</td>
</tr>
<tr>
<td>FingerP14</td>
<td>32.76</td>
<td>32.45</td>
<td>32.14</td>
<td>31.84</td>
<td>31.54</td>
<td>31.2</td>
<td>30.82</td>
<td>30.46</td>
<td>30.09</td>
<td>29.69</td>
<td>29.28</td>
<td>28.85</td>
<td>28.35</td>
<td>27.85</td>
<td>27.25</td>
<td>26.67</td>
<td>25.4</td>
<td>24.44</td>
</tr>
<tr>
<td>FingerP15</td>
<td>38.49</td>
<td>38.21</td>
<td>37.93</td>
<td>37.65</td>
<td>37.31</td>
<td>36.97</td>
<td>36.56</td>
<td>36.17</td>
<td>35.77</td>
<td>35.38</td>
<td>34.96</td>
<td>34.49</td>
<td>33.96</td>
<td>33.47</td>
<td>33.28</td>
<td>32.2</td>
<td>31.4</td>
<td>30.49</td>
</tr>
<tr>
<td>FingerP16</td>
<td>39</td>
<td>38.59</td>
<td>38.18</td>
<td>37.75</td>
<td>37.29</td>
<td>36.86</td>
<td>36.26</td>
<td>35.73</td>
<td>35.19</td>
<td>34.66</td>
<td>34.1</td>
<td>33.44</td>
<td>33.77</td>
<td>33.17</td>
<td>32.82</td>
<td>32.2</td>
<td>31.4</td>
<td>30.49</td>
</tr>
<tr>
<td>FingerP17</td>
<td>33.97</td>
<td>33.59</td>
<td>33.21</td>
<td>32.79</td>
<td>32.35</td>
<td>31.92</td>
<td>31.47</td>
<td>31.04</td>
<td>30.61</td>
<td>30.17</td>
<td>29.57</td>
<td>28.9</td>
<td>28.26</td>
<td>27.59</td>
<td>26.75</td>
<td>25.81</td>
<td>24.82</td>
<td>23.64</td>
</tr>
</tbody>
</table>
Table 5-5: PSNRs (dB) of Compressed Fingerprint Images Using LS9-7I

<table>
<thead>
<tr>
<th>Image File</th>
<th>1 bpp</th>
<th>0.95 bpp</th>
<th>0.9 bpp</th>
<th>0.85 bpp</th>
<th>0.8 bpp</th>
<th>0.75 bpp</th>
<th>0.7 bpp</th>
<th>0.65 bpp</th>
<th>0.6 bpp</th>
<th>0.55 bpp</th>
<th>0.5 bpp</th>
<th>0.45 bpp</th>
<th>0.4 bpp</th>
<th>0.35 bpp</th>
<th>0.3 bpp</th>
<th>0.25 bpp</th>
<th>0.2 bpp</th>
<th>0.15 bpp</th>
<th>0.1 bpp</th>
</tr>
</thead>
<tbody>
<tr>
<td>FingerP2</td>
<td>36.55</td>
<td>36.26</td>
<td>35.95</td>
<td>35.62</td>
<td>35.28</td>
<td>34.9</td>
<td>34.5</td>
<td>34.02</td>
<td>33.5</td>
<td>32.95</td>
<td>32.38</td>
<td>31.73</td>
<td>31.13</td>
<td>30.94</td>
<td>29.81</td>
<td>28.59</td>
<td>27.15</td>
<td>25.64</td>
<td>23.79</td>
</tr>
<tr>
<td>FingerP3</td>
<td>39.73</td>
<td>39.63</td>
<td>39.33</td>
<td>39.24</td>
<td>38.79</td>
<td>38.66</td>
<td>38.3</td>
<td>38.43</td>
<td>37.87</td>
<td>37.68</td>
<td>37.14</td>
<td>36.96</td>
<td>36.06</td>
<td>35.13</td>
<td>34.01</td>
<td>32.75</td>
<td>31.31</td>
<td>29.64</td>
<td>28.03</td>
</tr>
<tr>
<td>FingerP4</td>
<td>39.26</td>
<td>39.04</td>
<td>38.88</td>
<td>38.67</td>
<td>38.4</td>
<td>38.13</td>
<td>37.86</td>
<td>37.51</td>
<td>37.15</td>
<td>36.71</td>
<td>36.18</td>
<td>35.7</td>
<td>35.04</td>
<td>34.09</td>
<td>32.97</td>
<td>31.72</td>
<td>30.3</td>
<td>28.59</td>
<td>26.98</td>
</tr>
<tr>
<td>FingerP5</td>
<td>38.98</td>
<td>38.97</td>
<td>38.77</td>
<td>38.57</td>
<td>38.41</td>
<td>38.24</td>
<td>37.96</td>
<td>37.73</td>
<td>37.44</td>
<td>37.06</td>
<td>36.7</td>
<td>36.11</td>
<td>35.29</td>
<td>34.33</td>
<td>33.27</td>
<td>31.94</td>
<td>30.36</td>
<td>28.49</td>
<td>26.31</td>
</tr>
<tr>
<td>FingerP6</td>
<td>35.42</td>
<td>35.12</td>
<td>34.8</td>
<td>34.47</td>
<td>34.12</td>
<td>33.71</td>
<td>33.27</td>
<td>32.72</td>
<td>32.17</td>
<td>31.58</td>
<td>30.93</td>
<td>30.28</td>
<td>29.62</td>
<td>28.75</td>
<td>27.51</td>
<td>26.17</td>
<td>24.69</td>
<td>22.99</td>
<td>21.07</td>
</tr>
<tr>
<td>FingerP7</td>
<td>35.62</td>
<td>35.31</td>
<td>35.05</td>
<td>34.73</td>
<td>34.36</td>
<td>33.94</td>
<td>33.51</td>
<td>33.02</td>
<td>32.47</td>
<td>31.86</td>
<td>31.13</td>
<td>30.31</td>
<td>29.45</td>
<td>28.2</td>
<td>26.81</td>
<td>25.42</td>
<td>23.69</td>
<td>21.79</td>
<td>19.76</td>
</tr>
<tr>
<td>FingerP8</td>
<td>34.12</td>
<td>33.85</td>
<td>33.56</td>
<td>33.26</td>
<td>32.93</td>
<td>32.63</td>
<td>32.3</td>
<td>31.94</td>
<td>31.59</td>
<td>31.17</td>
<td>30.77</td>
<td>30.36</td>
<td>29.8</td>
<td>29.19</td>
<td>28.51</td>
<td>27.69</td>
<td>26.79</td>
<td>25.58</td>
<td>24.25</td>
</tr>
<tr>
<td>FingerP9</td>
<td>31.54</td>
<td>31.19</td>
<td>30.81</td>
<td>30.43</td>
<td>30.06</td>
<td>29.7</td>
<td>29.3</td>
<td>28.94</td>
<td>28.54</td>
<td>28.13</td>
<td>27.69</td>
<td>27.25</td>
<td>26.73</td>
<td>26.08</td>
<td>25.42</td>
<td>24.61</td>
<td>23.78</td>
<td>22.81</td>
<td>21.69</td>
</tr>
<tr>
<td>FingerP11</td>
<td>35.38</td>
<td>35.11</td>
<td>34.82</td>
<td>34.42</td>
<td>34.04</td>
<td>33.64</td>
<td>33.26</td>
<td>32.72</td>
<td>32.16</td>
<td>31.59</td>
<td>30.99</td>
<td>30.24</td>
<td>29.41</td>
<td>28.28</td>
<td>26.95</td>
<td>25.57</td>
<td>24.1</td>
<td>22.3</td>
<td>20.35</td>
</tr>
<tr>
<td>FingerP12</td>
<td>33.35</td>
<td>33.07</td>
<td>32.79</td>
<td>32.44</td>
<td>32.09</td>
<td>31.73</td>
<td>31.36</td>
<td>30.98</td>
<td>30.46</td>
<td>29.88</td>
<td>29.29</td>
<td>28.65</td>
<td>27.86</td>
<td>26.95</td>
<td>25.87</td>
<td>24.5</td>
<td>23.08</td>
<td>21.46</td>
<td>19.69</td>
</tr>
<tr>
<td>FingerP13</td>
<td>31.8</td>
<td>31.45</td>
<td>31.09</td>
<td>30.74</td>
<td>30.37</td>
<td>30.02</td>
<td>29.64</td>
<td>29.28</td>
<td>28.92</td>
<td>28.58</td>
<td>28.17</td>
<td>27.74</td>
<td>27.32</td>
<td>26.8</td>
<td>26.1</td>
<td>25.33</td>
<td>24.57</td>
<td>23.6</td>
<td>22.46</td>
</tr>
<tr>
<td>FingerP14</td>
<td>32.45</td>
<td>32.1</td>
<td>31.74</td>
<td>31.34</td>
<td>30.97</td>
<td>30.58</td>
<td>30.18</td>
<td>29.81</td>
<td>29.43</td>
<td>29.04</td>
<td>28.66</td>
<td>28.29</td>
<td>27.88</td>
<td>27.47</td>
<td>26.96</td>
<td>26.29</td>
<td>25.52</td>
<td>24.79</td>
<td>24.02</td>
</tr>
<tr>
<td>FingerP15</td>
<td>36.44</td>
<td>36.23</td>
<td>35.97</td>
<td>35.72</td>
<td>35.41</td>
<td>35.2</td>
<td>34.94</td>
<td>34.65</td>
<td>34.36</td>
<td>34.07</td>
<td>33.78</td>
<td>33.43</td>
<td>32.91</td>
<td>32.41</td>
<td>31.83</td>
<td>31.16</td>
<td>30.46</td>
<td>29.5</td>
<td>28.29</td>
</tr>
<tr>
<td>FingerP16</td>
<td>37.56</td>
<td>37.36</td>
<td>37.13</td>
<td>36.88</td>
<td>36.65</td>
<td>36.35</td>
<td>35.98</td>
<td>35.51</td>
<td>35.05</td>
<td>34.61</td>
<td>34.08</td>
<td>33.53</td>
<td>33</td>
<td>32.44</td>
<td>31.53</td>
<td>30.37</td>
<td>29.13</td>
<td>27.84</td>
<td>25.98</td>
</tr>
</tbody>
</table>
The quality of reconstructed fingerprint images by LS9-7I is acceptable with PSNR Diff values range from -0.651 to 3.811. 24.8% of PSNR Diff values are negative. And 66.8% of PSNR Diff values fall between 0.001 and 1.5. Four of the 17 test images (FingerP3, FingerP4, FingerP5, FingerP15) have PSNR Diff values higher than 1.5. However, the LS9-7I PSNR values corresponding to those relatively high PSNR Diff values range from 34.65 to 39.73, which are acceptable. In addition, we can see that PSNR Diff values are reduced while bpp decreases for all 17 test fingerprint images, from Figure 5-10.

To further evaluate quality of the reconstructed fingerprint images by LS9-7I. We use Verifinger to calculate minutiae and test fingerprint images matching. The fingerprint database contains the 17 original fingerprint images. The test fingerprint images are reconstructed from the compressed images which bit rate is 0.75 bpp, since

Figure 5-10 PSNR difference between Bior9-7 and LS9-7I
FBI’s default bit rate for compression in standard is 0.75bpp [BrBH93]. Results are shown in Table 5-6.

Table 5-6: Minutiae Detection and Fingerprint Identification

<table>
<thead>
<tr>
<th>Original Image</th>
<th>Reconstructed Image</th>
<th>Bior9-7</th>
<th>LS9-7I</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Minutiae</td>
<td>Minutiae</td>
<td>Difference</td>
</tr>
<tr>
<td>FingerP1</td>
<td>164</td>
<td>163</td>
<td>-1</td>
</tr>
<tr>
<td>FingerP2</td>
<td>188</td>
<td>192</td>
<td>4</td>
</tr>
<tr>
<td>FingerP3</td>
<td>141</td>
<td>142</td>
<td>1</td>
</tr>
<tr>
<td>FingerP4</td>
<td>150</td>
<td>161</td>
<td>11</td>
</tr>
<tr>
<td>FingerP5</td>
<td>122</td>
<td>120</td>
<td>-2</td>
</tr>
<tr>
<td>FingerP6</td>
<td>131</td>
<td>140</td>
<td>9</td>
</tr>
<tr>
<td>FingerP7</td>
<td>91</td>
<td>92</td>
<td>1</td>
</tr>
<tr>
<td>FingerP8</td>
<td>104</td>
<td>112</td>
<td>8</td>
</tr>
<tr>
<td>FingerP9</td>
<td>115</td>
<td>112</td>
<td>-3</td>
</tr>
<tr>
<td>FingerP10</td>
<td>58</td>
<td>61</td>
<td>3</td>
</tr>
<tr>
<td>FingerP11</td>
<td>139</td>
<td>138</td>
<td>-1</td>
</tr>
<tr>
<td>FingerP12</td>
<td>99</td>
<td>99</td>
<td>0</td>
</tr>
<tr>
<td>FingerP13</td>
<td>127</td>
<td>140</td>
<td>13</td>
</tr>
<tr>
<td>FingerP14</td>
<td>134</td>
<td>134</td>
<td>0</td>
</tr>
<tr>
<td>FingerP15</td>
<td>114</td>
<td>107</td>
<td>-7</td>
</tr>
<tr>
<td>FingerP16</td>
<td>118</td>
<td>118</td>
<td>0</td>
</tr>
<tr>
<td>FingerP17</td>
<td>121</td>
<td>115</td>
<td>-6</td>
</tr>
</tbody>
</table>

In the Difference column, zero means same number of minutiae obtained, negative means some minutiae are lost, and positive means some minutiae are added. In the Identification column, OK means that the reconstructed image matches the original image. From Table 5-6, we can see that both Bior9-7 and LS9-7I cause minutiae lost in some fingerprint images, and minutiae adding in some others. Same number of minutiae is obtained in 2 out of 17 fingerprint images by Bior9-7, while 6 out of 17 by LS9-7I. There is no specific pattern for both algorithms. All of the reconstructed images, either by Bior9-7 or by LS9-7I, are matched to the original images successfully.
A test fingerprint image (FingerP5), with the largest PSNR Diff value corresponding to .75 bpp, its reconstructed images by Bior9-7 and LS9-7I, and their skeletal images with minutiae are shown in Figure 5-11. The original image has 122 minutiae, the reconstructed image by Bior9-7 has 120 minutiae, and the reconstructed image by LS9-7I has 122 minutiae. We can see that the original image (a) and the two reconstructed images (b, c) cannot be distinguished by HVS, though its PSNR Diff value, 2.32 dB, is relatively high.
Figure 5-11: Comparison of Compressed Fingerprint Image and Their Skeletal Images with Minutiae – FingerP5
5.3 Computational Complexity

In this section, we compare the computational complexities of the wavelet transform computed using standard algorithm (Bior9-7) with that computed using lifting-based algorithm (LS9-7I). Bior9-7 is used as a comparison base. The unit we use is the cost, measured in number of multiplications and additions, of computing one sample pair \((a^k_j, d^k_j)\).

In general, the cost of applying a \(n\)-tap filter \(h\) is \(n\) multiplications and \((n - 1)\) additions. The cost of the standard algorithm thus is \((2n - 1)\). If the filter coefficients are symmetric and the length of filter is odd, the cost is \((3n - 1)/2\) with \(((n - 1)/2 + 1)\) multiplications and \((n - 1)\) additions. Therefore, for \((9, 7)\) filters in WSQ, the total cost of the standard algorithm is \((13 + 10) = 23\), in which the cost of applying 9-tap filter is 13 with 5 multiplications and 8 additions, and the cost of applying 7-tap filter is 10 with 4 multiplications and 6 additions.

Let us consider the lifting-based algorithm now. The cost of two scale steps (S5, S6) is 1 multiplication each. The cost of four lifting steps (S1, S2, S3 and S4) is 1 multiplication and 2 additions each. Thus, the total cost of the lifting algorithm is \((4 \times (1 + 2) + 2 \times 1) = 14\). Roughly, the cost of the lifting algorithm for \((9, 7)\) filter is 60 percent of the cost of the standard algorithm. Table 5-1 gives the cost \(S\) of standard algorithm, the cost \(L\) of the lifting-based algorithm, and the relative speedup \((S/L - 1)\) for the \((9, 7)\) filters using in WSQ.
Table 5-7: Computational Costs and Speedup of Bior9-7 and LS9-7

<table>
<thead>
<tr>
<th>Filter</th>
<th>Bior9-7</th>
<th></th>
<th>LS9-7</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Multiplications</td>
<td>Additions</td>
<td></td>
<td>Multiplications</td>
</tr>
<tr>
<td>(9,7)</td>
<td>9</td>
<td>14</td>
<td></td>
<td>6</td>
</tr>
<tr>
<td>Total Cost:</td>
<td>23</td>
<td></td>
<td></td>
<td>14</td>
</tr>
<tr>
<td>Speedup:</td>
<td>0%</td>
<td></td>
<td></td>
<td>64%</td>
</tr>
</tbody>
</table>

The reduction of the cost is significant and then requires less execution time. In Table 5-2, the CPU time for applying three algorithms in 19 images is tabulated. Although the CPU time cannot fully reflect the real execution time of these algorithms, they can still show us that the lifting-based algorithm need less time than the standard algorithm to compute the wavelet transform no matter what the sizes of images are.
Table 5-8: CPU Time for Three Algorithms (second)

<table>
<thead>
<tr>
<th>Fingerprint File Name</th>
<th>Width (pixel)</th>
<th>Height (pixel)</th>
<th>Bior9-7 (WSQ)</th>
<th>LS9-7</th>
<th>LS9-71</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td>Forward Transform</td>
<td>Inverse Transform</td>
<td>Forward Transform</td>
</tr>
<tr>
<td>cmp00001</td>
<td>589</td>
<td>605</td>
<td>7.593</td>
<td>9.924</td>
<td>1.031</td>
</tr>
<tr>
<td>cmp00002</td>
<td>832</td>
<td>768</td>
<td>14</td>
<td>18.298</td>
<td>2.047</td>
</tr>
<tr>
<td>cmp00003</td>
<td>832</td>
<td>768</td>
<td>13.922</td>
<td>18.196</td>
<td>2.046</td>
</tr>
<tr>
<td>cmp00004</td>
<td>815</td>
<td>752</td>
<td>13.281</td>
<td>17.358</td>
<td>2.125</td>
</tr>
<tr>
<td>cmp00005</td>
<td>832</td>
<td>768</td>
<td>14.36</td>
<td>18.768</td>
<td>2.047</td>
</tr>
<tr>
<td>cmp00007</td>
<td>545</td>
<td>638</td>
<td>7.625</td>
<td>9.966</td>
<td>1.016</td>
</tr>
<tr>
<td>cmp00008</td>
<td>800</td>
<td>750</td>
<td>13.391</td>
<td>17.502</td>
<td>1.751</td>
</tr>
<tr>
<td>cmp00009</td>
<td>512</td>
<td>678</td>
<td>7.781</td>
<td>10.170</td>
<td>0.953</td>
</tr>
<tr>
<td>cmp00010</td>
<td>375</td>
<td>526</td>
<td>4.125</td>
<td>5.391</td>
<td>0.515</td>
</tr>
<tr>
<td>cmp00011</td>
<td>614</td>
<td>717</td>
<td>9.641</td>
<td>12.601</td>
<td>1.281</td>
</tr>
<tr>
<td>cmp00012</td>
<td>516</td>
<td>716</td>
<td>8.031</td>
<td>10.496</td>
<td>1.047</td>
</tr>
<tr>
<td>cmp00013</td>
<td>539</td>
<td>651</td>
<td>7.531</td>
<td>9.843</td>
<td>0.968</td>
</tr>
<tr>
<td>cmp00014</td>
<td>466</td>
<td>578</td>
<td>5.75</td>
<td>7.515</td>
<td>0.734</td>
</tr>
<tr>
<td>cmp00015</td>
<td>436</td>
<td>670</td>
<td>10.797</td>
<td>14.111</td>
<td>1.453</td>
</tr>
<tr>
<td>cmp00016</td>
<td>666</td>
<td>758</td>
<td>11.172</td>
<td>14.602</td>
<td>1.407</td>
</tr>
<tr>
<td>cmp00017</td>
<td>521</td>
<td>666</td>
<td>7.469</td>
<td>9.762</td>
<td>0.984</td>
</tr>
<tr>
<td>cmp00018</td>
<td>750</td>
<td>750</td>
<td>12.35</td>
<td>16.337</td>
<td>1.656</td>
</tr>
<tr>
<td>cmp00019</td>
<td>750</td>
<td>750</td>
<td>12.406</td>
<td>16.214</td>
<td>1.656</td>
</tr>
</tbody>
</table>
5.4 Summary

In this chapter, we first described the algorithm for traditional wavelet transform. After that, we factorized this wavelet transform into a scheme based on lifting. Furthermore, we modified the lifting-based algorithm to perform an integer-to-integer transform. Consequently, a comparative study of these three algorithms was performed considering Peak Signal-Noise-Ratio of the original and reconstructed fingerprint images and cost of computation. The experimental results indicate the following:

- The quality of image reconstructed by lifting-based wavelet packet transform is not significantly worse than that by the standard wavelet packet transform. The lifting-based algorithm can be used to substitute the standard algorithm in WSQ.
- The computational cost of the lifting algorithm for (9, 7) filter is 60 percent of the cost of the standard algorithm. In hardware implementation, the reduction of the cost for computation makes lifting-based schemes a great choice for both high throughput and low-power applications.

Based on above study, the conclusion is that the lifting-based integer wavelet transform is an excellent alternative to the standard wavelet transform on fingerprint processing, other than those wavelet transforms based on filter bank structure. We will propose a VLSI architecture for WPT using lifting scheme to transform data integer to integer in the following chapter.
CHAPTER 6

HARDWARE IMPLEMENTATION

In this chapter, a novel FPGA architecture for Wavelet Packet Transform (WPT) is proposed which is optimized for special decomposition procedure based on WSQ specification. We follow a bottom-up approach to describe the design -- starting from the lowest level and building up the structure of the design gradually.

6.1 Proposed Architecture

The proposed architecture calculates the forward transform and inverse transform (IWPT) in row-column fashion on a block of data of size $M \times N$. To perform the WPT, the architecture reads in the block data, carries out the transform, and outputs the LL, LH, HL, and HH data at each level of decomposition. To perform the IWPT, first all the sub-bands from the lowest level are read in. At the end, the LL, HL, LH and HH values of the next higher level are obtained. Then, the transform values of all the four subbands (LL, LH, HL, and HH) are read in, and IWPT is carried out in the new data set.

The architecture, as shown in Figure 6-1, consists of two processing modules (PM1 and PM2), and two memory modules (MEM1, MEM2). Each processing module consists of two processing blocks (PB) along with a register file (REG). We have PB1, PB2, and REG1 for PM1, and have PB3, PB4, and REG2 for PM2. As mentioned in Chapter 3, the forward transform and inverse transform of each subband are symmetrical.
if the lifting scheme is used. Hence, in the rest of this thesis, we discuss all the details in terms of forward transform while inverse transform is considered an extension to forward transform.

![Block Diagram of the proposed architecture](image)

**Figure 6-1: Block Diagram of the proposed architecture**

### 6.1.1 Data Flow Design

In the transform for each fingerprint image subband, there are two passes. Each pass calculates the transform along one dimension. In the first pass, PB1 and PB2 in PM1 read in the data from external memory (Ext.MEM), executed the first two lifting steps (as introduced in section 5.1.1), and write results into MEM1. PB3 and PB4 in PM2 execute the next two lifting steps and write results (highpass and lowpass terms along the row) into MEM2, in which the lowpass terms are stored in the first bank of MEM2 and the highpass term are stored in second bank. Afterward, the data in MEM2 are written back to Ext.MEM. This finishes the transforms along rows. In the second pass, the transform
is calculated along columns. Data flow of the second pass is exactly same as the first pass. At the end of the second pass, all four subbands (LL, HL, LH, and HH) are written to Ext.MEM, whereas the data in the next subband for transformation are fetched into the system line by line. The data flow is shown in Figure 6-2.

![Figure 6-2: Data flow of transform](image)

6.1.2 Transform Computation Style and Order

All four processor calculate either the row or column transform at any given instant. As illustrated in Figure 6-3, PB1 updates the odd elements $x_{01}, x_{03}, \ldots, \text{etc.}$ along the rows to $y_{01}, y_{03}, \ldots, \text{etc.}$, whereas PB2 updates the even elements $x_{00}, x_{02}, \ldots, \text{etc.}$ along the rows to $y_{00}, y_{02}, \ldots, \text{etc.}$ PB3 and PB4 start computation as soon as the required elements are updated by PB1 and PB2. PB3 calculates the highpass elements (odd) $z_{01}, z_{03}, \ldots, \text{etc.}$ and PB4 calculates the lowpass elements (even) $z_{00}, z_{02}, \ldots, \text{etc.}$ This is further illustrated in scheduling shown in Table 6-1. In general, PB3 calculate highpass values along rows in first pass and along columns in the second pass. Similarly PB4 calculated lowpass values. A sequential order of calculation is used.
6.1.3 Processing Block and Module Design

All four processing blocks (PBs) in the proposed architecture consist of two adders, one multiplier and one multiplexer (MUX). Figure 6-4 illustrates the internal structure of a PB. And a processing module (PM) consists of two PBs. For each PM, two coefficients for two subbands (ID) are generated in alternate cycles since fewer resources are being used on purpose. In order to carry out the scaling step, a shifter (not shown in Figure 6-4) is connected to the output of PB3 and PB4.
Note that in this lifting-based architecture, PBs are not only designed for forward transform but also for inverse transform because DWT and IDWT of each subband are symmetrical. A multiplexer built in processing block achieves the coefficient-choosing mechanism for both transforms. Thus, there are two working modes for system, mode 0 and mode 1. Selecting mode 0 will make the PBs working on forward operation, that the coefficients of forward transform are input to multipliers. Selecting mode 1 will make system working on inverse operation, and the coefficients of forward transform are input.

6.1.4 Integer-to-integer Transform Design

As discussed in Chapter 5, the lifting scheme makes the transform possible from integer to integer without loss of the image quality. And in the hardware implementation, the integer arithmetic will be executed more efficiently and the hardware complexity will be reduced dramatically. So, an integer-to-integer transform is adopted here.
Lifting Coefficients

The absolute value of LS coefficients for 9-7 filters considered range from 0.052980118 to 1.586134342. In order to convert the filter coefficients to integers, the coefficients are multiplied by 256. The range of the filter coefficients for transform is now 1 to 512 as shown in Table 6-1 and Table 6-2, which implies that the coefficients require 10 bits to be represented in 2’s complement form. In order to implement into hardware, we convert these scaled lifting coefficients into 2’s complement form. It can be looked as those real number coefficients being shifted left by 8 bits. The scaled coefficients and their 2’s complement form for forward transform and inverse transform are tabulated in Table 6-1 and Table 6-2, respectively. For purpose of simplifying, we directly assign the binary value to the two constant elements for lifting coefficients in PB (shown in Figure 6-4).

Table 6-1: Constant Coefficients for Forward Transform

<table>
<thead>
<tr>
<th></th>
<th>Origin Value</th>
<th>Integer Value (x256)</th>
<th>Binary Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>α</td>
<td>-1.586134342</td>
<td>-406</td>
<td>1001101010</td>
</tr>
<tr>
<td>β</td>
<td>-0.052980118</td>
<td>-13</td>
<td>1111110011</td>
</tr>
<tr>
<td>γ</td>
<td>0.882911076</td>
<td>226</td>
<td>0011100010</td>
</tr>
<tr>
<td>δ</td>
<td>0.443506852</td>
<td>113</td>
<td>0001110001</td>
</tr>
<tr>
<td>K</td>
<td>1.14960439</td>
<td>294</td>
<td>0100100110</td>
</tr>
<tr>
<td>1/K</td>
<td>0.8698645</td>
<td>223</td>
<td>0011011110</td>
</tr>
</tbody>
</table>

Table 6-2: Constant Coefficients for Inverse Transform

<table>
<thead>
<tr>
<th></th>
<th>Origin Value</th>
<th>Integer Value (x256)</th>
<th>Binary Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>-α</td>
<td>1.586134342</td>
<td>406</td>
<td>0110010110</td>
</tr>
<tr>
<td>-β</td>
<td>0.052980118</td>
<td>13</td>
<td>0000001101</td>
</tr>
<tr>
<td>-γ</td>
<td>-0.882911076</td>
<td>-226</td>
<td>1100011110</td>
</tr>
<tr>
<td>-δ</td>
<td>-0.443506852</td>
<td>-113</td>
<td>1110000111</td>
</tr>
<tr>
<td>K</td>
<td>1.14960439</td>
<td>294</td>
<td>0100100110</td>
</tr>
<tr>
<td>1/K</td>
<td>0.8698645</td>
<td>223</td>
<td>0011011110</td>
</tr>
</tbody>
</table>
At the end of the multiplication, the product is shifted right by 8 to get the required result. This is implemented in hardware by rounding the eight least significant bits. The products are rounded to the next highest integer. And rounding operation is applied to the individual product terms instead of applying rounding to the results of the filter operation in order to avoid building a bigger accumulator.

Signal Values

The signal values have to be shifted left 2 bits as well in order to increase the precision. This shift operation occurs at the input signal. After accumulating all factors, the PB will shift the result right by 2 bits. An example is given below to illustrate this integer-to-integer transform scheme.

Consider the general structure in a lifting-based scheme

\[ y = x_1 + a \times (x_0 + x_2) \]  

(6-1)

where \( a \) is the lifting coefficient, \( x_0 \) are the signal samples, \( y \) is the transform value and with \( a = 0.2345, x_0 = 156, x_1 = 10, x_2 = 56 \). The floating-point implementation result is \( y = 59.714 \). Let's assume that the lifting coefficient is shifted left by 8 bits (and round to nearest integer) and the signal samples are shifted left by 2 bits. Then, \( a = 60, x_0 = 624, x_1 = 40, x_2 = 224 \). The product is \( 60 \times 624 + 224 = 50880 \). Shifting the product right by 8 bits and rounding will yield 199. Therefore, \( y = 40 + 199 = 239 \). This should be interpreted as round \( [239/4 + \text{decimal equivalent of two LSBs of 239}] = \text{round} [59 + 0.75] = 60 \).
Since the integer transform scheme is determined, in this architecture, the data path width is fixed at 16 bits. The adders are designed for 16-bit data. The multiplier multiplies a 16-bit number (signal value) by a 10-bit number (filter coefficient) and then rounds the product with eight LSBs and two MSBs (16 bits are required to represent the outputs and therefore, the MSBs would be sign extension bits) to form a 16-bit output. Figure 6-5 specifies the assignments for data path.

6.1.5 Scheduling

We have generated a detailed schedule for this lifting-based architecture. This schedule is resource constrained list-based schedule, where the resources consist of an adder, and a multiplier. In our design, the latency of the adder is one time unit and the latency of the multiplier is two time units. This is justified since the multiplier is a
parallel multiplier that is generated by Core Generator and is typically two times slower than an adder. A snapshot of the schedule for the (9, 7) filter applied on an even-length row and an odd-length row are provided in Table 6-3 and Table 6-4.

The schedule in tables should be read as follows. In the fourth cycle, Adder1 of PB1 in processing module 1 adds the elements \(x_{0,4}, x_{0,6}\) and stores the sum in register R1. The multiplier reads this sum in the next cycle (fifth cycle), carries out the required product (it takes two cycles), and stores the data in register R2 (seventh cycle). The second adder (adder2) reads the value in R2 and pluses the elements \(x_{0,5}\) to generate \(y_{0,5}\) in the next cycle (eighth cycle). The output of the second adder is stored in a suitable memory location in MEM1 module and is also supplied to PB2 using REG1. Thus, to process a step in this scheme, the PB1 takes five cycles. Adder1 in PB2 starts computation in the eighth cycle. After five cycles (twelfth cycle), the result \(y_{0,4}\) computed from elements \(y_{0,3}, y_{0,5}\) and \(x_{0,4}\) is absorbed by Processing Module 2 and is stored in MEM1 too. Adder1 in PB3 starts in the twelfth cycle to execute the third step in LS-based DWT.

Since two identical processing modules (PM1 and PM2) are used in this architecture, the PB3 and PB4 in PM2 follow the similar schedules as PB1 and PB2 in PM1 do. There are only two differences between these two modules. The first is that the PBs in PM2 get the elements from MEM1 instead of Ext.MEM. The second is that the results in PM2 are written into MEM2 in a deinterleaved fashion instead of MEM1.

For a fingerprint image of size \(N \times M\), the length of two dimensions is not required to be same. And they can be either even-length or odd-length. Thus, the schedule of odd-length differs from the schedule of even length described above. An idle cycle is added
right after PB1 and PB3 complete the computation of a row or column. This is illustrated in Table 6-4 as marked a symbol '·'.
Table 6-3: Part of the schedule for PM1 and PM2 applied on an even-element (8x8) block

<table>
<thead>
<tr>
<th>Time</th>
<th>Processing Module 1</th>
<th>Processing Module 2</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>PB1</td>
<td>PB2</td>
</tr>
<tr>
<td></td>
<td>Adder 1</td>
<td>Multiplier</td>
</tr>
<tr>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>$x_0 + x_0$</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>$x_0 + x_0$</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>$x_0 + x_0$</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>$x_0 + x_0$</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>$x_0 + x_0$</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>$x_0 + x_0$</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>$x_0 + x_0$</td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>$x_0 + x_0$</td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>$x_0 + x_0$</td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>$x_0 + x_0$</td>
<td></td>
</tr>
<tr>
<td>12</td>
<td>$x_0 + x_0$</td>
<td></td>
</tr>
<tr>
<td>13</td>
<td>$x_0 + x_0$</td>
<td></td>
</tr>
<tr>
<td>14</td>
<td>$x_0 + x_0$</td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>$x_0 + x_0$</td>
<td></td>
</tr>
<tr>
<td>16</td>
<td>$x_0 + x_0$</td>
<td></td>
</tr>
<tr>
<td>17</td>
<td>$x_0 + x_0$</td>
<td></td>
</tr>
<tr>
<td>18</td>
<td>$x_0 + x_0$</td>
<td></td>
</tr>
<tr>
<td>19</td>
<td>$x_0 + x_0$</td>
<td></td>
</tr>
<tr>
<td>20</td>
<td>$x_0 + x_0$</td>
<td></td>
</tr>
<tr>
<td>21</td>
<td>$x_0 + x_0$</td>
<td></td>
</tr>
<tr>
<td>22</td>
<td>$x_0 + x_0$</td>
<td></td>
</tr>
<tr>
<td>23</td>
<td>$x_0 + x_0$</td>
<td></td>
</tr>
<tr>
<td>24</td>
<td>$x_0 + x_0$</td>
<td></td>
</tr>
<tr>
<td>25</td>
<td>$x_0 + x_0$</td>
<td></td>
</tr>
</tbody>
</table>
Table 6-4: Part of the schedule for PM1 and PM2 applied on an odd-element (9x9) block

<table>
<thead>
<tr>
<th>Time</th>
<th>Processing Module 1</th>
<th></th>
<th>Processing Module 2</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Adder 1</td>
<td>Multiplier</td>
<td>Adder 2</td>
<td>Adder 1</td>
</tr>
<tr>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>$y_{0,0} + y_{0,2}$</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>$y_{0,2} + y_{0,4}$</td>
<td>$a \cdot R_1$</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>$y_{0,4} + y_{0,6}$</td>
<td>$a \cdot R_1$</td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>$y_{0,6} + y_{0,8}$</td>
<td>$a \cdot R_1$</td>
<td>$R_2 + y_{0,8}$</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>$- a \cdot R_1$</td>
<td>$R_2 + y_{0,6}$</td>
<td>$y_{0,1} + y_{0,1}$</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>$x_{1,1} + x_{1,2}$</td>
<td>$- R_2 + x_{0,0}$</td>
<td>$y_{0,1} + y_{0,3}$</td>
<td>$\beta \cdot R_1$</td>
</tr>
<tr>
<td>8</td>
<td>$x_{1,3} + x_{1,4}$</td>
<td>$a \cdot R_1$</td>
<td>$R_2 + x_{0,7}$</td>
<td>$y_{0,3} + y_{0,5}$</td>
</tr>
<tr>
<td>9</td>
<td>$x_{1,5} + x_{1,6}$</td>
<td>$a \cdot R_1$</td>
<td>$- y_{0,5} + y_{0,7}$</td>
<td>$\beta \cdot R_1$</td>
</tr>
<tr>
<td>10</td>
<td>$x_{1,7} + x_{1,8}$</td>
<td>$a \cdot R_1$</td>
<td>$R_2 + x_{1,1}$</td>
<td>$y_{0,7} + y_{0,7}$</td>
</tr>
<tr>
<td>11</td>
<td>$- a \cdot R_1$</td>
<td>$R_2 + x_{1,3}$</td>
<td>$y_{1,1} + y_{1,1}$</td>
<td>$\beta \cdot R_1$</td>
</tr>
<tr>
<td>12</td>
<td>$- a \cdot a \cdot R_1$</td>
<td>$R_2 + x_{1,5}$</td>
<td>$y_{1,3} + y_{1,3}$</td>
<td>$\beta \cdot R_1$</td>
</tr>
<tr>
<td>13</td>
<td>$- a \cdot a \cdot R_1$</td>
<td>$R_2 + x_{1,7}$</td>
<td>$y_{1,5} + y_{1,5}$</td>
<td>$\beta \cdot R_1$</td>
</tr>
<tr>
<td>14</td>
<td>$- a \cdot a \cdot R_1$</td>
<td>$y_{1,7} + y_{1,7}$</td>
<td>$\beta \cdot R_1$</td>
<td>$R_2 + x_{1,2}$</td>
</tr>
<tr>
<td>15</td>
<td>$- a \cdot a \cdot R_1$</td>
<td>$y_{1,9} + y_{1,9}$</td>
<td>$\beta \cdot R_1$</td>
<td>$R_2 + x_{1,4}$</td>
</tr>
<tr>
<td>16</td>
<td>$- a \cdot a \cdot R_1$</td>
<td>$y_{1,7} + y_{1,7}$</td>
<td>$\beta \cdot R_1$</td>
<td>$R_2 + x_{1,6}$</td>
</tr>
<tr>
<td>17</td>
<td>$- a \cdot a \cdot R_1$</td>
<td>$R_2 + x_{1,8}$</td>
<td></td>
<td></td>
</tr>
<tr>
<td>18</td>
<td>$- a \cdot a \cdot R_1$</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>19</td>
<td>$- a \cdot a \cdot R_1$</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>20</td>
<td>$- a \cdot a \cdot R_1$</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>21</td>
<td>$- a \cdot a \cdot R_1$</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>22</td>
<td>$- a \cdot a \cdot R_1$</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>23</td>
<td>$- a \cdot a \cdot R_1$</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>24</td>
<td>$- a \cdot a \cdot R_1$</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>25</td>
<td>$- a \cdot a \cdot R_1$</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>26</td>
<td>$- a \cdot a \cdot R_1$</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>27</td>
<td>$- a \cdot a \cdot R_1$</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
6.1.6 Memory Organizations and Size

The proposed architecture consists of two memory modules: MEM1 and MEM2. Both modules consist of two banks. All banks have one read and one write port. Further, system is set up to access these internal memory modules two times per cycle.

![Diagram of Memory Modules]

(a) MEM1 Module  
(b) MEM2 Module

Figure 6-6: Memory modules structure

**MEM1 module:**

MEM1 module consists of two banks, MEM$_{10}$ and MEM$_{11}$, as shown in Figure 6-6. The odd-indexed terms and even-indexed terms of a row are stored in these banks after processed by Processing Module 1. Here, MEM$_{10}$, written by PB1, contains the odd-index terms of a row. And MEM$_{11}$, written by PB2, contains the even-index terms. During the computation period in Processing Module 2, MEM$_{10}$ will supply the odd-indexed samples for PB3, one of three data required by PB3. And MEM$_{11}$ will supply the even-indexed samples for PB4. Thereby, one read access is needed by each of these two memory banks to complete this supplying operation.
The size of MEM1 depends on the lifetime of a row of data. In this architecture, PB1 and PB2 calculate the first and second lifting steps in which the row elements are written into the MEM1 as the order shown in the Table 6-5. In the later computation, one of three inputs required by PB3 and PB4 will be fed by MEM1. For instance, \(y_{0,1}\) is generated by PB1 in sixth cycle and is absorbed by PB3 in thirteenth cycle. So, the memory cell for \(y_{0,1}\) can be only refreshed after 8 cycles. But during these 8 cycles, eight odd-indexed coefficients are resulted by PB1. Thus, at least eight coefficients need to be saved in memory before they can be used by system. Considering the case of odd-length signals, in this architecture, we build each of the memory banks (MEM1_0 and MEM1_1) to hold 10 coefficients at most.
MEM2 Module

The MEM2 module consists of two banks (MEM20 and MEM21), as shown in figure 6-6. They contain the final transformed coefficients of a row or column. PB3 writes the highpass coefficients in MEM20, and PB4 writes the lowpass coefficients in MEM21. All coefficients will replace the original data saved to the Ext.MEM.

\[
\begin{array}{cccccccc}
  x_{0,0} & x_{0,1} & x_{0,2} & x_{0,3} & x_{0,4} & x_{0,5} & \cdots & x_{0,N-1} \\
\end{array}
\]

Original Data

\[
\begin{array}{cccccccc}
  z_{0,0} & z_{0,2} & z_{0,4} & z_{0,6} & \cdots & \cdots & z_{0,2} & z_{0,1} & z_{0,3} & z_{0,5} & \cdots & z_{0,2i-1} \\
\end{array}
\]

Coefficients

Figure 6-7: Coefficient ordering in memory

In our schedule, the original data in a row or column segments in a subband are fetched from storage space in a sequential order by PM1, whereas the result (transform coefficients) is written back in a deinterleaved fashion (the lowpass coefficients followed by the highpass coefficients in a certain row or column as illustrated in Figure 6-7). It means that, although the lowpass coefficients and highpass coefficients are generated in alternated cycles as scheduled in section 6.1.5, the storage spaces for these coefficients are not available in Ext.MEM. Therefore, these transform coefficients cannot be written back to Ext.MEM right after they were generated. To solve this conflict, MEM2 is build in the system to store these coefficients till the original data in a certain segment are all get by system. And its size is determined and equals to the larger size between row and column. Therefore, the memory banks are of size \( \left\lfloor \max(N, M) / 2 \right\rfloor \) each. Here, for FBI's
application here, up to 850x850 pixels per fingerprint, the quantity to contain 500 pixel values each bank is sufficient.

6.1.7 Control

Control signals are needed primarily to maintain the steady flow of data to and from the processors. Our design consists of two PM controllers and one WPT controller. The PM controllers locate in each of the processing modules. And the WPT controller is a top-level controller of the whole architecture. All controllers communicate with each other by handshaking signals. WPT controller generates the addresses needed to get the correct input subband. And the PM controllers command the processing modules to do the actual transform works. Each controller consists of three components: Counter, Memory Read and Write Signals Generation Unit (MSGU), and Address Generation Unit (AGU).

In the WPT controller, Counter records the number of the wavelet transform level and the number of the subbands being processed. MSGU generates the read and write logic to direct the system to access subbands in the Ext.MEM. According to output of counter and logic signals, AGU generates the addresses of the start point and end point of subbands. These signals are passed to processing module and drive the PM to complete the wavelet transform for each subband. WPT controller is programmable to suit for different wavelet transform schemes by changing the read and write logic. Here, a five-level transform with 64 subbands is designed. And it is easy to expand system to execute a fully WPT.

PM controllers start working after getting commands from WPT controller. In this controller, Counters keep track of the number of rows and the number of elements in each
row that have been processed, which are primarily used to generate the memory read and write signals. All the counters are capable of counting up to a maximum of $N$. MSGU, which is driven by counter output, generates the signals required for memory reads and writes (i.e., row, element values). One of the inputs to the second adder (in all the processing blocks) has to be read from memory, and the memory write signals are generated based on this signal. And the exactly element address is generated by AGU with an “in place” addressing scheme.

6.1.8 Timing

The latency of this system depends on latency of the processing blocks and the start time of PB3. We assume the latency of the adder is $\tau_a$, and the delay of the multiplier is $\tau_m$. For each processing block, the latency is

$$\text{latency}_{PB} = 2\tau_a + \tau_m.$$ (6-2)

And in our architecture, PB3 can start the transform right after two required elements of the first adder are ready. As shown in Table 6-1, it takes one cycle. Thereby, to get the first output of transformed coefficient, the latency is

$$\text{latency} = 4 \times \text{latency}_{PB} + 1$$

$$= 8\tau_a + 4\tau_m + 1.$$ (6-3)

Therefore, the total time required for one level of decomposition of an $N \times M$ subband is calculated as following

$$\text{time}_{Total} = \text{time}_{column} + \text{time}_{row}$$

$$= (\text{latency} + \lceil N/2 \rceil \times M ) + (\text{latency} + \lceil M/2 \rceil \times N )$$

$$= 2 \times \text{latency} + (\lceil N/2 \rceil \times M + \lceil M/2 \rceil \times N )$$ (6-4)

Substituted equation 6-3 in equation 6-4, the time required for this architecture is
\[
\text{time}_{\text{total}} = 2 \times (8 \tau_a + 4 \tau_m + 1) + (\lceil N/2 \rceil \times M + \lceil M/2 \rceil \times N)
\]  
(6-5)

6.2 Implementations and Analysis

6.2.1 Design Flow

The proposed design is implemented in an FPGA development platform, based on the Xilinx Spartan-II FPGA family.

The design is modeled described in VHDL hardware description language, and it is simulated and analyzed using Active-HDL. The behavior of the architecture is...
synthesized using Synplify Pro synthesis package. Then, the implementation tool, Xilinx
ISE, is used to map the logic to components, place the logic cells, route the signals, and
generate a bitstream file to do the final configuration in the FPGA device.

The technology libraries and symbol libraries that were used in Active-HDL,
Synplify and Xilinx ISE for synthesis and simulation were all based on the standard
library for Xilinx FPGA. This guarantees that synthesized design has the nearest
simulation results as the final implementation.

Transferring the design from Synplify Pro to Xilinx ISE is accomplished by
generating a VHDL netlist as a logical design file format, which creates symbol and
schematic views for all levels of the design hierarchy. In our design, an EDIF file (one
format of netlist) is created for the following usages.

By implementing to a specific Xilinx architecture, the logical design file format, like
the EDIF just achieved, is converted into a physical file format. The physical information
for FPGAs is contained in the Native Circuit Description (NCD) file.

Many verification simulation steps are involved in the design flow. The first one is
named Register Transfer Level (RTL) simulation which uses VHDL simulation tools to
verify the functional correctness of the architecture’s behavior. The second is a post-
synthesis functional simulation which is applied to confirm the functionality of the
extracted gate-level circuit. The third, post-implementation simulation ensures that the
gate-level schematic, which has been imported and edited in Xilinx ISE, is equivalent to
the proposed architecture. These are followed by the DRC (design rule checking), which
evaluate the NCD file for problems that could prevent the design from functioning
properly and is done with Xilinx BitGen. Once DRC is done, it is assumed that the
design is functional correct.

At last, the final bitstream is generated from the NCD file. It can be used to program
the Xilinx device with Xilinx iMPACT software package.

6.2.2 Design Units and Their Implementations

Design Units

Here is a short description of the VHDL design units:

Adder/Subtractor: This is a component created by Core Generator. It operates on
signed data. The data inputs are provided on the A and B input buses,
which is a 16-bit bus each. It is used in the design unit — Processing
Block.

Multiplier: This component was generated with the Core Generator, too. This
multiplier functions a 16-bit data by 10-bit data multiplying operation.
The output bus width is 20-bit. It is used in the Processing Block for
multiplying the lifting coefficients with the corresponding data which is a
sum of two pixels.

Processing Block: This unit consists of three components which are one
multiplier, one 2-to-1 multiplex, and two adders. It executes one of four
lifting steps. Processing block is used as a component in the architecture
of entity — Processing Modules. The output of multiplier in this unit is
rounded from 26 bits to 16 bits as illustrated in section 6.1.4.
Processing Module: Each processing module consists of two processing block. They are controlled by a state machine to read/write data. This entity-architecture pair implements the schedule as specified in section 6.1.5.

DWT_97_Core: This entity-architecture pair includes two components which are PM1 and PM2. The function of this unit is to execute a one-dimension transform for elements based on line. Under command of a state machine, it also reads data from MEM modules and writes the results back.

MEM1 Module, MEM2 Module: These components were generated with Core Generator. They function as storage elements for the results of two Processing Modules in this system as described in section 6.1.6.

WPT Controller: This entity-architecture pair implements the control functions for Wavelet Packet Transform. It generates signals and data required by transformation, such as the size of subband, and the location of data in storage space.

Transform (top-level entity): This entity-architecture pair instantiates all necessary components and describes the interconnections to compose the organization described in section 6.1. In this design unit, a number of state machines implement this 2-D wavelet packet transform algorithms.

Implementation Results

The logic resources of the FPGA are divided in different categories as Look-Up-Tables (LUT), Input/Output Blocks (IOB), Flip-flops, Multiplex, RAM Blocksetc. A specific group of a number of these resources is called Configurable Logic Block (CLB)
and a group of CLBs forms a Slice. The following shows the resource usage for this design.

Target Device: Xilinx Spartan-II FPGA x2s200

Target Package: fg256

Target Speed: -5

Mapper Version: spartan2 -- Revision: 1.4

Synthesis Tools: Synplicity Pro 7, Xilinx ISE (alliance version)

Timing (post synthesis):

<table>
<thead>
<tr>
<th>Description</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Minimum period</td>
<td>27.130ns</td>
</tr>
<tr>
<td>Maximum Frequency</td>
<td>36.860MHz</td>
</tr>
<tr>
<td>Minimum input arrival time before clock</td>
<td>11.723ns</td>
</tr>
<tr>
<td>Maximum output required time after clock</td>
<td>18.721ns</td>
</tr>
</tbody>
</table>

Resource usage:

<table>
<thead>
<tr>
<th>Description</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of Slice Flip Flops:</td>
<td>758 out of 4,704 17%</td>
</tr>
<tr>
<td>Number of occupied Slices:</td>
<td>967 out of 2,352 41%</td>
</tr>
<tr>
<td>Number of Slices containing related logic:</td>
<td>967 out of 967 100%</td>
</tr>
<tr>
<td>Total Number 4 input LUTs:</td>
<td>1607 out of 4,704 35%</td>
</tr>
<tr>
<td>Number used as logic:</td>
<td>1533</td>
</tr>
<tr>
<td>Number used as a route-thru:</td>
<td>74</td>
</tr>
<tr>
<td>Number of bonded IOBs:</td>
<td>70 out of 176 40%</td>
</tr>
<tr>
<td>IOB Flip Flops:</td>
<td>8</td>
</tr>
<tr>
<td>Number of Block RAMS:</td>
<td>3 out of 14 22%</td>
</tr>
<tr>
<td>Number of GCLKs:</td>
<td>2 out of 4 50%</td>
</tr>
</tbody>
</table>
Number of GCLKIOBs: 2 out of 4 50%

Notes that since RAM blocks have been integrated in the SPARTAN family FPGAs, the two memory modules in this architecture are directly implemented via those RAM blocks. Therefore, MEM Module 1 occupies one RAM Block and MEM Module2 occupies two.

The evaluation of the implemented architecture considers mainly on the number of the preliminary gates (2-input NAND gate equivalents). The number of gates required in a design is related to the area of solid device. The more gates a design has, the more area a device need. In this design, the preliminary gate count of the major units, Adder, Multiplier and Control Unit, are provided in Table 6-6.

<table>
<thead>
<tr>
<th>Component</th>
<th>Gate count</th>
</tr>
</thead>
<tbody>
<tr>
<td>Adder (16 bits + 16 bits)</td>
<td>317</td>
</tr>
<tr>
<td>Multiplier (16 bits x 10 bits)</td>
<td>2,600</td>
</tr>
<tr>
<td>Control Unit</td>
<td>8,007</td>
</tr>
</tbody>
</table>

Table 6-7: Preliminary Gate Count Estimates and Number of Components Used in the Proposed Architecture and the Convolution Filter-bank Architecture

<table>
<thead>
<tr>
<th></th>
<th>Adder</th>
<th>Multiplier</th>
<th>Control Unit</th>
<th>Gate Count</th>
</tr>
</thead>
<tbody>
<tr>
<td>Proposed Architecture without MEM Modules</td>
<td>8</td>
<td>4</td>
<td>1</td>
<td>20966</td>
</tr>
<tr>
<td>Convolution Filter-bank Architecture without MEM Modules</td>
<td>9</td>
<td>14</td>
<td>1</td>
<td>47283</td>
</tr>
</tbody>
</table>

In Table 6-7, the number of basic components used in the proposed architecture is listed and the prototype of the architecture occupies 20,966 gates without the memory.
modules. For the sake of comparison, the preliminary gate counts for the convolution filter-bank architecture are also tabulated in Table 6-7. Because of the hierarchy in the design, the same basic components like adder unit and multiplier unit, and the same top-level controller are used. Only the DWT_97_Core is implemented with convolution-based algorithm instead of LS based one. Comparing first row and second row in table 6-7, we can notice that the number of Adder unit decreases from 14 in convolution based scheme down to 8 in the proposed architecture. In the meantime, the number of multiplier unit decreases from 9 down to 4. The reductions cause the total gates required by proposed architecture based on integer LS scheme decreasing dramatically, which is from 47283 down to 20966. The ratio of reduction on gates reaches to 56 percentages.

### 6.3 Summary of Chapter 6

In this chapter, a hardware architecture for WPT algorithm in fingerprint image processing was proposed and explored in details. First, the framework was explained as a combination of several functional modules – processing module, memory module, etc. Subsequently, the data flow among the modules was discussed and an integer transform was designed. Furthermore, the schedules of computation were analyzed. Based on these schedules, the processing modules and memory modules were designed and arranged well. At last, this architecture was implemented into a hardware platform, which is a Xilinx Spartan-II family FPGA. The result of implementation shows us that the usage of hardware by lifting architecture is only half of the usage of convolution filter structure. The goal of reducing hardware requirement was achieved.
CHAPTER 7

CONCLUSIONS AND FUTURE WORK

7.1 Conclusions

In this thesis, a VLSI architecture implementing a wavelet packet transform algorithm for fingerprint image compression was proposed. The wavelet packet transform algorithm is based on lifting scheme (LS9-7I) instead of the traditional convolution method (Bior9-7) used by FBI. And it can transform the image pixel values in the integer format. This architecture was implemented on reprogrammable hardware – FPGA. We compared the quality of the reconstructed images processed by LS9-7I and Bior9-7 by means of image quality measure – Peak Signal-to-Noise Ratio (PSNR) and the application of identification – fingerprint matching. We also compared the hardware usages after synthesis of both architectures. Comparison results show that our proposed architecture can reduce the size of processor to half and the loss of image quality is insignificant and acceptable.

The proposed hardware can fulfill the wavelet packet transform on fingerprint image efficiently. The quality of the reconstructed image is acceptable. To compare it to one by Bior9-7, the difference cannot be distinguished by human visual system, though the fingerprint images reconstructed by LS9-7I have slightly lower peak signal-to-noise ratio values than those by Bior9-7. Furthermore, the reconstructed image still can be matched with the original image successfully, by calculating the minutiae.
Lifting scheme leads to a speedup when compared to the traditional (convolution filter-banks) implementation. At the meantime, the hardware usage of lifting-based architecture is significantly less than that of the traditional architecture.

In the processor hardware, integer arithmetic is generally much faster than floating-point arithmetic. Integer numbers are more efficient to transform and take less space to store. Hence, the feature of the integer-to-integer transformation of lifting scheme makes the processor computationally efficient as well as permits lossless representation of the image pixels.

Computation intensive problems often require a hardware intensive solution. Unlike a general purpose microprocessor with fixed number of Arithmetic and Logic Unit (ALU), a hardware implementation on FPGAs can achieve greater parallelism, and hence higher throughput.

7.2 Future Work

The lessons learned from this experience will help us enhance similar implementations in the future. Few of the improvements that we now foresee are listed below:

The controller unit is reconfigurable so that, if necessary, this architecture is not only able to process WPT for fingerprint images, but also provides a method to process any type of two-dimensional WPT.

The design has the potential of further performance improvement. The hardware unit is designed to be hierarchical and scalable. This means that, when enough hardware
resources are available, the multiple units can be cascaded to achieve the requirement of higher speed processing.

A corresponding coder-codec can be built on the FPGA to process the obtained wavelet coefficients. Furthermore, it is worth to implement FBI fingerprint image processing on a single processor completely.

Data movement from processor to out-chip image storage and back to processor takes a significant amount of the processing time. Data movement could have been minimized. A suggestion with respect to embedded memory architecture is to increase the size of embedded memory modules, so that the subband that will be processed in next iteration can be stored and fetched by processor more quickly.

When FPGAs are utilized more than 40%, the timing performance drops sharply. This is because the FPGA runs out of routing resources. Consequently many long and circuitous routes are resulted. Hence the overall system clock drops. One possible solution to this problem is to compromise between FPGA utilization and timing performance to achieve more optimized results. Another possible solution is to improve the timing performance by applying the FPGA decomposition method to optimize the route scheme in the synthesis step.
BIBLIOGRAPHY


{IAFIS-IC-0110v2}


[NIST03] ftp://sequoyah.nist.gov/pub/cmp_imgs/cmp_imgs/, Reference Test Fingerprint Images Set, National Institute of Standards and Technology (NIST) in USA


VITA

Graduate College
University of Nevada, Las Vegas

Zhu, Tao

Local Address:
4250 S Arville St, Apt 204
Las Vegas, Nevada 89103

Home Address:
118 Tianhe Road, Room 706
Guangzhou, Guangdong, P.R.China 510620

Degree:
Bachelor of Science, Telecommunication Engineering, 1997
Shanghai Jiaotong University

Thesis Title: A VLSI Architecture for Lifting-Based Wavelet Packet Transform In Fingerprint Image Compression

Thesis Examination Committee:
Chair, Dr. Shahram Latifi, Ph. D.
Committee Member: Dr. Emma Regentova, Ph. D.
Committee Member: Dr. Henry Selvaraj, Ph. D.
Graduate Faculty Representative: Dr. Wolfgang Bein, Ph. D.