

FPGA BASED "PARALLEL IMPLEMENTATION OF DWT FOR IMAGE COMPRESSION"

A DISSERTATION

*Submitted in partial fulfillment of the
requirements for the award of the degree*

of

MASTER OF TECHNOLOGY

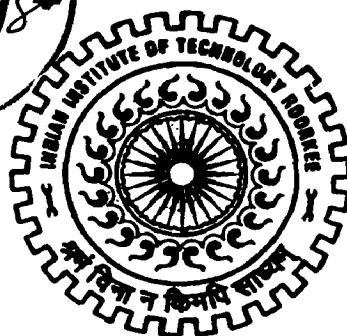
in

ELECTRONICS AND COMMUNICATION ENGINEERING

(With Specialization in Semiconductor Devices and VLSI Technology)

By

K. PRASHANTH KUMAR



DEPARTMENT OF ELECTRONICS AND COMPUTER ENGINEERING
INDIAN INSTITUTE OF TECHNOLOGY ROORKEE
ROORKEE - 247 667 (INDIA)

JUNE, 2008


CANDIDATE'S DECLARATION

I hereby declare that the work being presented in the dissertation report titled “**FPGA Based Parallel Implementation of DWT for Image Compression**” in partial fulfillment of the requirement for the award of the degree of **Master of Technology in Semiconductor Devices and VLSI Technology**, submitted in the Department of Electronics and Computer Engineering, Indian Institute of Technology Roorkee, is an authentic record of my own work carried out under the guidance of Dr. R. C. Joshi and Dr. A. K. Saxena, Professor, Department of Electronics and Computer Engineering, Indian Institute of Technology Roorkee.

I have not submitted the matter embodied in this dissertation report for the award of any other degree.

Dated: 30/06/08

Place: IIT Roorkee.



(K. Prashanth Kumar)

CERTIFICATE

This is to certify that above statements made by the candidate are correct to the best of my knowledge and belief.

Dated: 25/6/08

Place: IIT Roorkee.



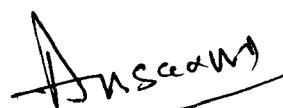
Dr. R. C. Joshi

Professor,

Department of Electronics
and Computer Engineering,

IIT Roorkee,

Roorkee -247667(India).



Dr. A. K. Saxena

Professor,

Department of Electronics
and Computer Engineering,

IIT Roorkee,

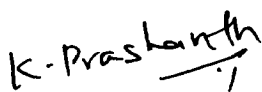
Roorkee -247667(India).

ACKNOWLEDGEMENTS

First of all, I would like to thank my supervisor, **Prof. Dr. R. C. Joshi**, and co-supervisor **Prof. Dr. A. K. Saxena** for getting me interested in the area of DWT based image compression of FPGA, and their consistent and sound critiques of my ideas and work. I am particularly grateful for their enthusiasm, constant support and encouragement. My dissertation could not have been done reasonably without insightful advice from them. Working under their guidance will always remain a cherished experience in my memory. I am also thankful to Indian Institute of Technology Roorkee for giving me this opportunity.

I am also grateful to the staff of special man power development in VLSI and related software, phase-II (SMDB-II), ministry of information technology, and government of India, for their kind cooperation extended by them in the execution of this dissertation. I am also thankful to my friends M. Naveen Kumar and Anil Kumar Sistu for their invaluable support during my dissertation work.

Most importantly, I would like to extend my deepest appreciation to my family for their love, encouragement and moral support. Finally I thank God for being kind to me and driving me through this journey.


(**K. Prashanth Kumar**)

ABSTRACT

An improved architecture for two-dimensional discrete wavelet transform (2D-DWT) to implement bi-orthogonal Cohen-Daubechies-Feuvar (CDF) (2,2) wavelet with line-based method is proposed for FPGA implementation using lifting scheme. The FPGA based hardware implementation profits especially from the high parallelism in the architecture and the moderate number precision required to preserve the qualitative effects of the mathematical models. The proposed architecture is designed to generate 4 sub bands coefficients concurrently per clock cycle that can perform a 1-level decomposition of a $N \times N$ image in exactly $N^2 / 4$ working clock cycles, without any line buffers at the column processor, thus reducing the time for line buffering but with an extra row processor and with 100% hardware utilization.

Proposed architecture is first tested using MATLAB software, then VHDL code is written in Xilinx ISE 9.1 and simulated results are verified using Modelsim software. After final implementation .bit file is generated, which is burned on FPGA successfully.

TABLE OF CONTENTS

CANDIDATE'S DECLARATION & CERTIFICATE	i
ACKNOWLEDGEMENTS	ii
ABSTRACT	iii
TABLE OF CONTENTS.....	iv
LIST OF ABBREVIATIONS.....	vi
LIST OF TABLES.....	vii
LIST OF FIGURES.....	viii
CHAPTER 1 INTRODUCTION.....	1
1.1 Introduction.....	1
1.2 Problem Statement	3
1.3 Organization of Report	3
CHAPTER 2 DISCRETE WAVELET TRANSFORM BASED IMAGE COMPRESSION.....	4
2.1 Why Wavelet Transforms?.....	4
2.2 Wave and Wavelet.....	6
2.3 The Continuous Wavelet Transform.....	7
2.4 The Discrete Wavelet Transform	8
2.4.1 Multi-Resolution Analysis Using Filter Banks.....	9
2.4.2 Conditions for Perfect Reconstruction.....	10
2.5 Classification of Wavelets.....	12
2.6 Wavelet Families	13
2.7 Wavelet Based Image Compression	14
2.7.1 Wavelet Transform as the Source Encoder	15
2.7.2 Image Compression Techniques.....	16
2.8 Features of Image Compression Using Wavelets	16
CHAPTER 3 THE LIFTING SCHEME.....	18
3.1 Direct Form Structure	18
3.2 Polyphase Structure.....	19
3.3 Lifting Scheme.....	20

3.3.1 Factoring Wavelet Filters Into Lifting Scheme	21
3.3.2 ChohenDaubechiesFeauveau(CDF)(2,2) Wavelet Using Lifting Scheme.....	22
3.3.3 Integer-To-Integer Transform	23
3.4 Advantages of Lifting scheme	23
CHAPTER 4 INTRODUCTION TO FPGA IMPLEMENTATION	24
4.1 Architectural Overview	24
4.2 FPGA Design Flow	25
4.3 VHDL.....	27
CHAPTER 5 LIFTING BASED DWT ARCHITECTURES.....	29
5.1 Lifting Based DWT Architecture	29
5.2 Parallel Architecture for Lifting Based DWT.....	30
5.3 Proposed Parallel Architecture For Lifting Based CDF (2,2) Wavelet.....	32
CHAPTER 6 IMPLEMENTATION	34
6.1 Cohen-Daubechies-Feauveau (CDF)(2,2) Wavelet.....	34
6.2 Software Implementation and Results	36
6.3 Hardware Implementation and Results.....	40
CHAPTER 7 CONCLUSION AND SCOPE FOR FUTURE WORK	43
7.1 Conclusion.....	43
7.2 Scope for Future Work	43
REFERENCES.....	44

LIST OF ABBREVIATIONS

1D	One Dimensional
2D	Two Dimensional
CDF	Cohen-Daubechies-Feauveau
CLB	Configurable Logic Block
CWT	Continuous Wavelet Transform
DCT	Discrete Cosine Transform
DFT	Discrete Fourier Transform
DSP	Digital Signal Processing
DWT	Discrete Wavelet Transform
FPGA	Field-Programmable Gate Array
HDL	Hardware Description Language
IC	Integrated Circuit
IEEE	Institute of Electrical and Electronics Engineers
IOB	Input Output Buffer
JPEG	Joint Photographic Experts Group
MPEG	Moving Pictures Experts Group
MRA	Multi resolution Analysis
SNR	Signal to Noise Ratio
STFT	Short Time Fourier Series
USB	Universal Serial Bus
VHDL	Very high speed integrated circuits Hardware Descriptive Language.

LIST OF TABLES

Table 6.1	CDF (2,2) wavelet with lifting scheme (a) Forward transform.....	34
	(b) Inverse transform.....	34
Table 6.2	Device utilization summary.....	41
Table 6.3	Timing report.....	42

LIST OF FIGURES

Fig.2.1	Tiling in time-frequency plane by (a) Wavelets and (b) STFT.....	6
Fig.2.2	Demonstration of (a) Wave and (b) Wavelet.....	7
Fig.2.3	Three-level wavelet decomposition tree.....	9
Fig.2.4	Three-level wavelet reconstruction tree.....	10
Fig.2.5	Two channel filter bank structure showing analysis and synthesis stages...	11
Fig.2.6	Wavelet families (a) Haar (b) Daubechies4 (c) Meyer (d) Morlet (e) Mexican Hat (f) CDF (2,2).....	13
Fig.2.7	Block diagrams of the JPEG2000 (a) Encoder and (b) Decoder.....	14
Fig.3.1	Direct form structure of (a) Analysis filter bank and (b) Synthesis filter.....	18
Fig.3.2	Polyphase structure of (a) Analysis filter bank..... (b) Equivalent representation of analysis filter bank and (c) Synthesis filter bank	20
Fig.3.3	Lifting scheme.....	21
Fig.3.4	Lifting structure for CDF (2,2) wavelet.....	22
Fig.4.1	Spartan 3E family architecture.....	25
Fig.4.2	FPGA design flow for implementation.....	26
Fig.5.1	Lifting based dwt architecture.....	30
Fig.5.2	Block diagram for increasing level of decomposition.....	30
Fig.5.3	Parallel architecture for lifting based DWT.....	31
Fig.5.4	Proposed parallel architecture for lifting based CDF (2, 2) wavelet.....	33
Fig.6.1	Rows and columns of level 1, 2 and 3 decomposition of an image.....	35
Fig.6.2	Original Lena image.....	37
Fig.6.3	Image after level 1 decomposition.....	38
Fig.6.4	Image after level 2 decomposition.....	39
Fig.6.5	Image after level 3 decomposition.....	40
Fig.6.6	Simulation result of top module.....	41
Fig.6.7	Programmed succeeded screen shot.....	42

CHAPTER 1

INTRODUCTION

1.1 Introduction

Among the various types of data commonly transferred over networks, image and video data comprises the bulk of the bit traffic and it is growing day by day. For example, a single small 4" × 4" size color picture, scanned at 300 dots per inch (dpi) with 24 bits/pixel of true color, will produce a file containing more than 4 megabytes of data. This picture requires more than one minute for transmission by a typical transmission line (64k bit/second ISDN). That is why large image files remain a major bottleneck in a distributed environment. Although increasing the bandwidth is a possible solution, the relatively high cost makes this less attractive. Therefore, compression is a necessary and essential method for creating image files with manageable and transmittable sizes.

JPEG (Joint Photographic Experts Group) [1] and MPEG (Moving Pictures Experts Group) are standards for representing images and video. Data compression algorithms are used in those standards to reduce the number of bits required to represent an image or a video sequence. Compression is the process of representing information in a compact form. Data compression treats information in digital form that is, as binary numbers represented by bytes of data with very large data sets. In order to be useful, a compression algorithm has a corresponding decompression algorithm that, given the compressed file, reproduces the original file. Compression algorithms fall into two broad types, lossless algorithms and lossy algorithms. A lossless algorithm reproduces the original exactly. A lossy algorithm, as its name implies, loses some data but has high compression ratio. Data loss may be unacceptable in many applications. For example, text compression must be lossless because a very small difference can result in statements with totally different meanings. There are also many situations where loss may be either unnoticeable or acceptable. In image compression, for example, the exact reconstructed value of each sample of the image is not necessary. Depending on the quality required of the reconstructed image, varying amounts of loss of information can be accepted.

The newer standard JPEG2000 is based on the Wavelet Transform (WT). Wavelets are the mathematical functions that satisfy a certain requirement (for instance a zero mean), and are used to represent data or other functions. In wavelet transform, dilations and

translations of a mother wavelet are used to perform a spatial/frequency analysis on the input data. For spatial analysis, contracted versions of the mother wavelets are used. These contracted versions can be compared with high frequency basis functions in the fourier based transforms. The relatively small support of the contracted wavelets makes them ideal for extracting local information like positioning discontinuities, edges and spikes in the data sequence, which makes them suitable for spatial analysis. Dilated versions of the mother wavelet, on the other hand, have relatively large support (the length of the dilated mother wavelet). The larger support extracts information about the frequency behavior of the data. Varying the dilation and translation of the mother wavelet, therefore, produces a customizable time/frequency analysis of the input signal.

Recent research on DWT has focused on a form of lifting which shows excellent performance compared to the conventional convolution method for implementation. Factoring discrete wavelet transform into lifting steps can reduce the computational complexity by 50% [2] and has advantages, including integer to integer transform [3], symmetric forward and inverse transforms [4]. Line-based architecture for the direct two-dimensional discrete wavelet transform (2D-DWT) is an efficient alternative tradeoff between speed and area [5], [6]. For image compression using line based architecture, first all the rows are processed, intermediate results are stored in buffer and then all the columns of intermediate results are processed. The disadvantage of the above method is it requires intermediate buffer size equal to size of image, high computation time besides underutilization of hardware. To overcome these disadvantages, parallel architectures are developed in a way to reduce the intermediate buffer size, computation time almost by half and with 100% hardware utilization.

Field Programmable Gate Array (FPGA)'s are most suitable for hardware implementation because they support high parallelism in the architecture, since each individual block in the architecture can work independently with separate clock. So, FPGA's are mostly widely used the Digital Signal Processing (DSP) especially when high parallelism is offered by the architecture. In addition reconfigurability [7] of FPGAs allows the implementation of more than one custom application on a single FPGA. Hardware Descriptive Languages (HDL) Verilog, VHDL are used to program FPGA.

1.2 Problem Statement

In modern hardware design, it's a fact that storage resource is more expensive than computation resource. So the key problem in hardware implementation is to achieve high performance while maintaining low memory requirement. To exploit parallelisms fully is a short cut to achieve high performance, and to reuse data is the way to reduce memory requirement.

The objectives of this dissertation work are:

- Study of parallel architectures for image compression using various wavelets on FPGA implementation.
- To investigate for an improved architecture of lifting based CDF (2, 2) for FPGA implementation.
- Testing the improved architecture using MATLAB software.
- Finally, FPGA implementation of tested improved architecture.

1.3 Organization of Report

Chapter 1 gives overview of evolution of lifting based DWT and its advantages over other conventional methods. It summarizes the problem statement of this thesis work.

Chapter 2 reviews the complete discrete wavelet transform in detail and why wavelet transform are used in image compression with its features. Also salient features of multi-resolution analysis of DWT are discussed.

Chapter 3 presents direct form structure, polyphase structure and lifting structure for implementing DWT.

Chapter 4 briefly describes FPGA architecture, design flow followed in FPGA implementation and basic introduction to different programming style's in VHDL.

Chapter 5 presents overview of previous architectures for decomposing the image into different level and also proposed parallel architecture for implementing lifting based CDF (2,2) wavelet.

Chapter 6 explains the algorithm for implementing CDF (2, 2) wavelet. Software and hardware implementation methods of proposed architecture with results are described.

Finally, chapter 7 concludes this thesis with scope for future work mentioned.

CHAPTER 2

DISCRETE WAVELET TRANSFORM BASED IMAGE COMPRESSION

Many evolving multimedia applications require transmission of high quality images over the network. One obvious way to accommodate this demand is to increase the bandwidth available to all users. Of course, this "solution" is not without technological and economical difficulties. Another way is to reduce the volume of the data that must be transmitted. There has been a tremendous amount of progress in the field of image compression especially during the past two decades. In order to make further progress in image coding, many research groups have begun to use wavelet transforms.

In this chapter, we will briefly discuss why wavelet transforms are used for image compression, differences between wave and wavelet, continuous and discrete wavelet transform, multi-resolution analysis of wavelet transform, wavelet based compression and its features.

2.1 Why Wavelet Transforms?

In most Digital Signal Processing (DSP) applications, the frequency content of the signal is very important. The Fourier transform is probably the most popular transform used to obtain the frequency spectrum of a signal. But the Fourier transform is only suitable for stationary signals, i.e., signals whose frequency content does not change with time. The Fourier transform, while it tells how much of each frequency exists in the signal, it does not tell at which time these frequency components occur.

Signals such as image and speech have different characteristics at different time or space, i.e., they are non-stationary. Most of the biological signals too, such as, Electrocardiogram, Electromyography, etc., are non-stationary. To analyze these signals, both frequency and time information are needed simultaneously, i.e., a time-frequency representation of the signal is needed.

To solve this problem, the Short-Time Fourier Transform (STFT) was introduced. The major drawback of the STFT is that it uses a fixed window width. The wavelet transform [8], which was developed in the last two decades, provides a better time-frequency representation of the signal than any other existing transforms.

Short Time Fourier Transform Vs Wavelet Transform

The STFT is a modified version of the fourier transform. The fourier transform separates the waveform into a sum of sinusoids of different frequencies and identifies their respective amplitudes. Thus it gives us a frequency-amplitude representation of the signal. In STFT, the non-stationary signal is divided into small portions, which are assumed to be stationary. This is done using a window function of a chosen width, which is shifted and multiplied with the signal to obtain the small stationary signals. The fourier transform is then applied to each of these portions to obtain the short time fourier transform of the signal.

The problem with STFT goes back to the Heisenberg uncertainty principle which states that, it is impossible for one to obtain which frequencies exist at particular time of instance, but, one can obtain the frequency bands existing in a time interval. This gives rise to the resolution issue where there is a trade-off between the time resolution and frequency resolution. To assume stationarity, the window is supposed to be narrow, which results in a poor frequency resolution, i.e., it is difficult to know the exact frequency components that exist in the signal; only the band of frequencies that exist is obtained. If the width of the window is increased, frequency resolution improves but time resolution becomes poor, i.e., it is difficult to know what frequencies occur at which time intervals. Also, choosing a wide window may violate the condition of stationarity. Consequently, depending on the application, a compromise on the window size has to be made. Once the window function is decided, the frequency and time resolutions are fixed for all frequencies and all times.

The wavelet transform solves the above problem to a certain extent. In contrast to STFT, which uses a single analysis window, the Wavelet Transform uses short windows at high frequencies and long windows at low frequencies. This results in multi-resolution analysis by which the signal is analyzed with different resolutions at different frequencies, i.e., both frequency resolution and time resolution vary in the time-frequency plane without violating the Heisenberg inequality.

In wavelet transform, as frequency increases, the time resolution increases; likewise, as frequency decreases, the frequency resolution increases. Thus, a certain high frequency component can be located more accurately in time than a low frequency component and a low frequency component can be located more accurately in frequency compared to a high frequency component.

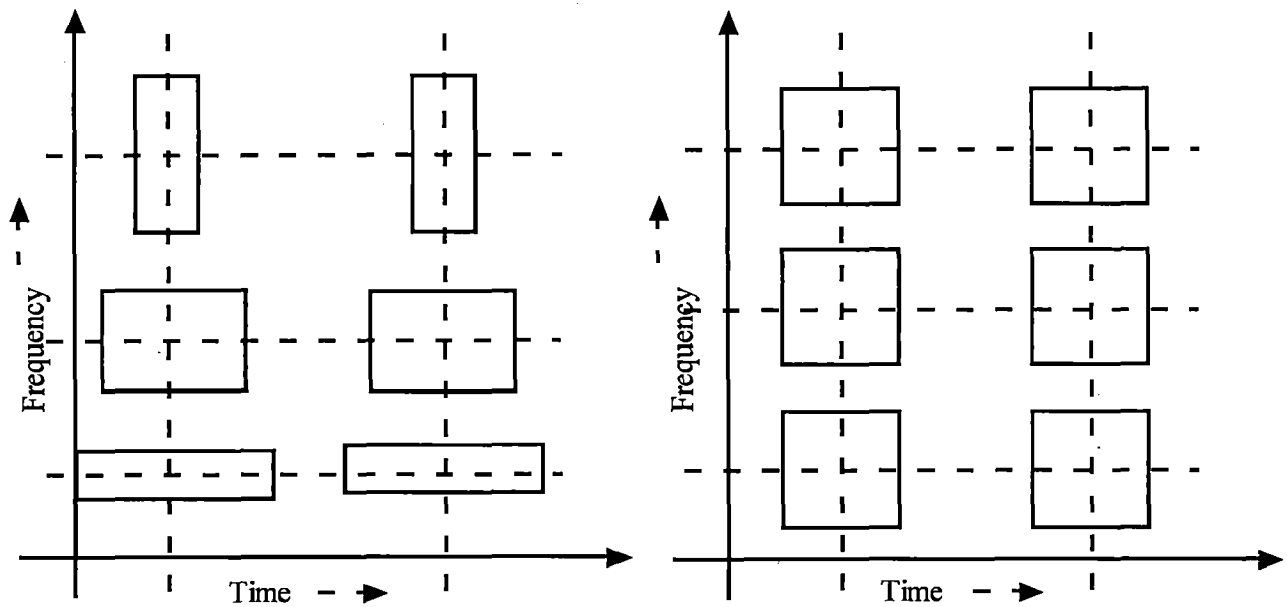


Fig.2.1 Tiling in time-frequency plane by: (a) Wavelets and (b) STFT [9]

Fig.2.1(a) shows tiling in time-frequency of wavelets and Fig.2.1(b) shows tiling in short-time fourier transform, It is seen that STFT gives a fixed resolution at all times, whereas wavelet transform gives a variable resolution.

The wavelet transform was developed independently in applied mathematics and signal processing. It is gradually substituting other transforms in some signal processing applications. For example, previously, the STFT was extensively used in speech signal processing, and Discrete Cosine Transform (DCT) was used for image compression. But now, the wavelet transform is substituting these, due to its better resolution properties and high compression capabilities.

2.2 Wave and Wavelet

A wave as shown in Fig.2.2 (a) is an oscillating function of time or space and is periodic. In contrast, wavelets as shown in Fig.2.2 (b) are localized waves. They have their energy concentrated in time or space and are suited to analysis of transient signals. While fourier transform and STFT use waves to analyze signals, the wavelet transform uses wavelets of finite energy.

The wavelet analysis is done similar to the STFT analysis. The signal to be analyzed is multiplied with a wavelet function just as it is multiplied with a window function in STFT, and then the transform is computed for each segment generated. However, unlike STFT, in wavelet transform, the width of the wavelet function changes with each spectral component.

The wavelet transform, at high frequencies, gives good time resolution and poor frequency resolution, while at low frequencies; the Wavelet Transform gives good frequency resolution and poor-time resolution.

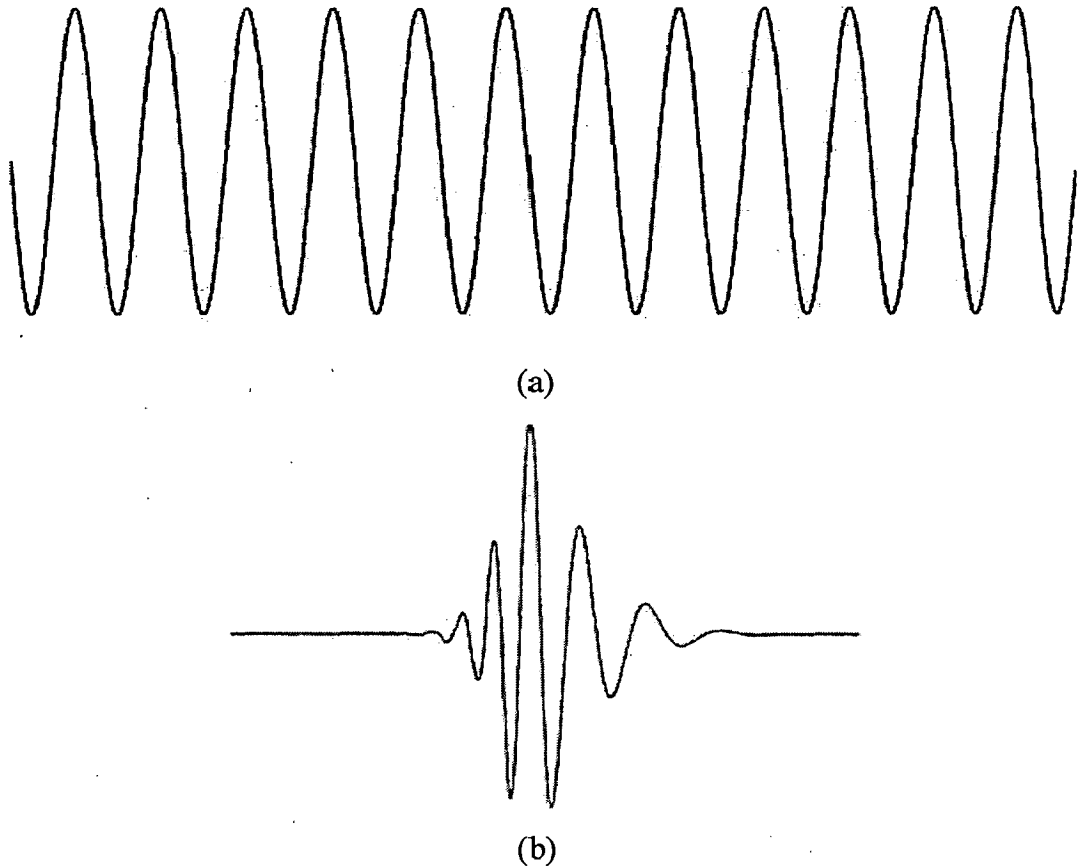


Fig.2.2 Demonstration of (a) Wave and (b) Wavelet

2.3 The Continuous Wavelet Transform

The Continuous Wavelet Transform (CWT) is provided by Eq.2.1, where $x(t)$ is the signal to be analyzed. $\psi(t)$ is mother wavelet or the basis function. All the wavelet functions used in the transformation are derived from the mother wavelet through translation (shifting) and scaling (dilation or compression).

$$X_{WT}(\tau, s) = \frac{1}{\sqrt{|s|}} \int x(t) \cdot \psi\left(\frac{t-\tau}{s}\right) dt \quad (2.1)$$

The mother wavelet used to generate all the basis functions is designed based on some desired characteristics associated with that function. The translation parameter τ relates to the location of the wavelet function as it is shifted through the signal. Thus, it corresponds to the time information in the wavelet transform. The scale parameter s is defined as $|1/\text{frequency}|$ and corresponds to frequency information. Scaling either dilates (expands) or compresses a

signal. Large scales (low frequencies) dilate the signal and provide detailed information hidden in the signal, while small scales (high frequencies) compress the signal and provide global information about the signal. Notice that the wavelet transform merely performs the convolution operation of the signal and the basis function. The above analysis becomes very useful as in most practical applications, high frequencies (low scales) do not last for a long duration, but instead, appear as short bursts, while low frequencies (high scales) usually last for entire duration of the signal.

The wavelet series is obtained by discretizing CWT. This aids in computation of CWT using computers and is obtained by sampling the time-scale plane. The sampling rate can be changed accordingly with scale change without violating the nyquist criterion. Nyquist criterion states that, the minimum sampling rate that allows reconstruction of the original signal is 2ω radians, where ω is the highest frequency in the signal. Therefore, as the scale goes higher (lower frequencies), the sampling rate can be decreased thus reducing the number of computations.

2.4 The Discrete Wavelet Transform

The wavelet series is just a sampled version of CWT and its computation may consume significant amount of time and resources, depending on the resolution required. The Discrete Wavelet Transform (DWT) [8], which is based on sub-band coding is found to yield a fast computation of wavelet transform. It is easy to implement and reduces the computation time and resources required.

The foundations of DWT go back to 1976 when techniques to decompose discrete time signals were devised. Similar work was done in speech signal coding which was named as sub-band coding. In 1983, a technique similar to sub-band coding was developed which was named pyramidal coding. Later many improvements were made to these coding schemes which resulted in efficient multi-resolution analysis schemes.

In CWT, the signals are analyzed using a set of basis functions which relate to each other by simple scaling and translation. In the case of DWT, a time-scale representation of the digital signal is obtained using digital filtering techniques. The signal to be analyzed is passed through filters with different cutoff frequencies at different scales, which is known as Multi Resolution Analysis (MRA).

2.4.1 Multi-Resolution Analysis Using Filter Banks

Filters are one of the most widely used signal processing functions. Wavelets can be realized by iteration of filters with rescaling. The resolution of the signal, which is a measure of the amount of detail information in the signal, is determined by the filtering operations, and the scale is determined by upsampling and downsampling (subsampling) operations [9]-[11].

The DWT is computed by successive lowpass and highpass filtering of the discrete time-domain signal as shown in Fig.2.3. This is called the Mallat algorithm or Mallat-tree decomposition. Its significance is in the manner it connects the continuous-time multiresolution to discrete-time filters. In the figure, the signal is denoted by the sequence $x[n]$, where n is an integer. The low pass filter is denoted by G_0 while the high pass filter is denoted by H_0 . At each level, the high pass filter produces detail information, $d[n]$, while the low pass filter associated with scaling function produces coarse approximations, $a[n]$.

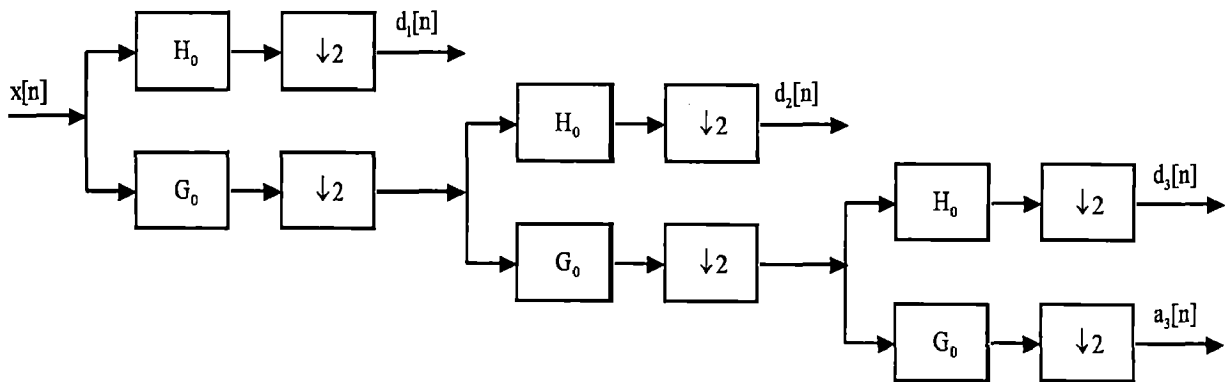


Fig.2.3 Three-level wavelet decomposition tree

At each decomposition level, the half band filters produce signals spanning only half the frequency band. This doubles the frequency resolution as the uncertainty in frequency is reduced by half. In accordance with Nyquist's rule if the original signal has a highest frequency of ω , which requires a sampling frequency of 2ω radians, then it now has a highest frequency of $\omega/2$ radians. It can now be sampled at a frequency of ω radians thus discarding half the samples with no loss of information. This decimation by 2 halves the time resolution as the entire signal is now represented by only half the number of samples. Thus, while the half band low pass filtering removes half of the frequencies and thus halves the resolution, the decimation by 2 doubles the scale.

With this approach, the time resolution becomes arbitrarily good at high frequencies, while the frequency resolution becomes arbitrarily good at low frequencies. The time-frequency plane is thus resolved as shown in Fig.2.1 (b). The filtering and decimation process is continued until the desired level is reached. The maximum number of levels depends on the length of the signal. The DWT of the original signal is then obtained by concatenating all the coefficients, $a[n]$ and $d[n]$, starting from the last level of decomposition.

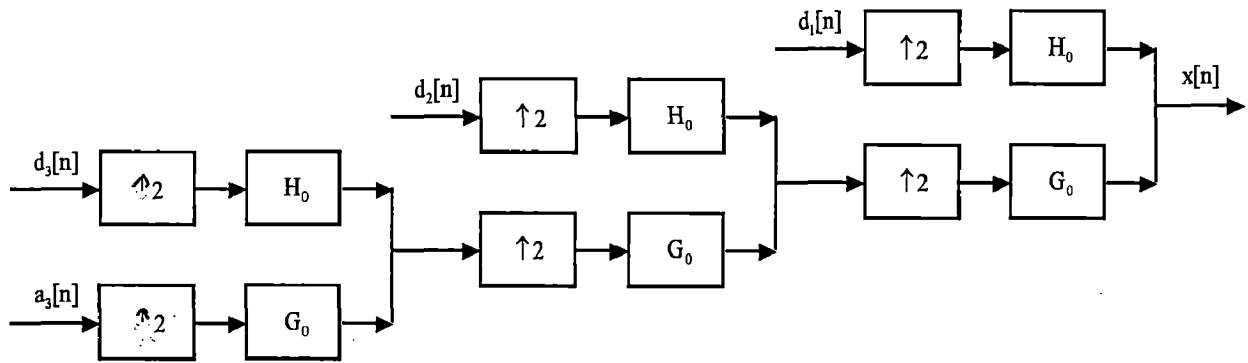


Fig.2.4 Three-level wavelet reconstruction tree

Fig.2.4 shows the reconstruction of the original signal from the wavelet coefficients. Basically, the reconstruction is the reverse process of decomposition. The approximation and detail coefficients at every level are upsampled by two, passed through the low pass and high pass synthesis filters and then added. This process is continued through the same number of levels as in the decomposition process to obtain the original signal. The Mallat algorithm works equally well if the analysis filters, G_0 and H_0 , are exchanged with the synthesis filters, G_1, H_1 .

2.4.2 Conditions for Perfect Reconstruction

In most wavelet transform applications, it is required that the original signal be synthesized from the wavelet coefficients. To get perfect reconstruction consider two-channel filter bank as shown in Fig.2.5, where $G_0(z)$ is the low pass analysis filter and $H_0(z)$ is the high pass analysis filter. As the output signal $\theta_0(n)$ and $\theta_1(n)$ have half the bandwidth of the original input signal, we can use decimation by two and still lose no information. The synthesis filter bank is also shown in Fig.2.5 and consists of two up-samplers and the low-pass synthesis filter $G_1(z)$ and the high pass synthesis filter $H_1(z)$.

From Fig.2.4 we know

$$\theta_0(z) = H_0(z) \cdot X(z)$$

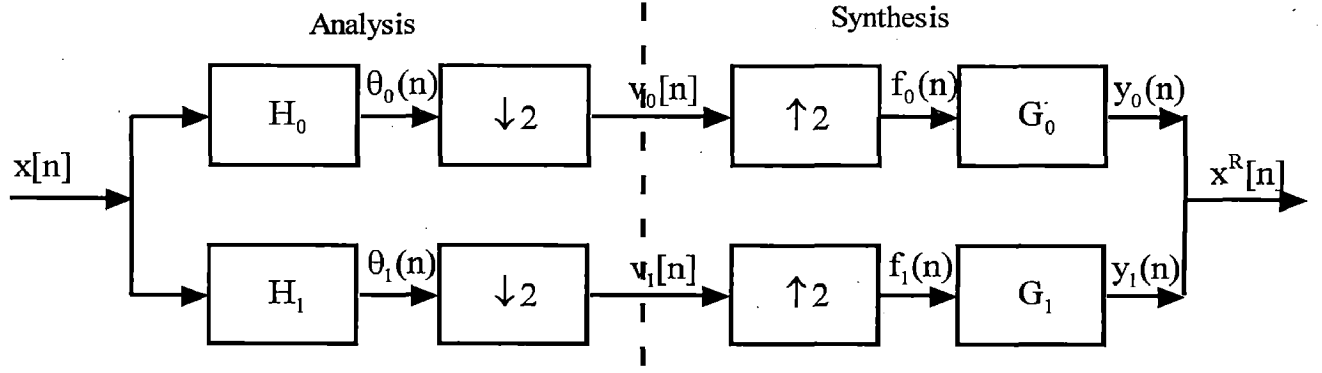


Fig.2.5 Two channel filter – bank structure showing analysis and synthesis stages [9]

$$Y_0(z) = G_0(z) \cdot F_0(z)$$

$$V_0(z) = \frac{1}{2} [\theta_0(z^{1/2}) + \theta_0(-z^{1/2})] \quad (2.2)$$

$$F_0(z) = V_0(z^2)$$

Substituting for $F_0(z)$ and $V_0(z)$, we obtain

$$Y_0(z) = \frac{1}{2} G_0(z) [H_0(z) \cdot X(z) + H_0(-z) \cdot X(-z)] \quad (2.3)$$

and

$$Y_1(z) = \frac{1}{2} G_1(z) [H_1(z) \cdot X(z) + H_1(-z) \cdot X(-z)] \quad (2.4)$$

Finally, we obtain for the output $\hat{X}(z)$,

$$\begin{aligned} X^R(z) &= \frac{1}{2} X(z) [H_0(z) \cdot G_0(z) + H_1(z) \cdot G_1(z)] \\ &+ \frac{1}{2} X(-z) [H_0(-z) \cdot G_0(z) + H_1(-z) \cdot G_1(z)] \end{aligned} \quad (2.5)$$

Eq.2.5 can be simplified to

$$X^R(z) = T(z) \cdot X(z) + S(z) \cdot X(-z) \quad (2.6)$$

where we have defined

$$\begin{aligned} T(z) &= \frac{1}{2} [H_0(z) \cdot G_0(z) + H_1(z) \cdot G_1(z)] \\ S(z) &= \frac{1}{2} [H_0(-z) \cdot G_0(z) + H_1(-z) \cdot G_1(z)] \end{aligned} \quad (2.7)$$

In order to have perfect reconstruction (PR) at the synthesis, we must impose that

$$X^R(z) = cX(z)z^{-n_0} \quad (2.8)$$

where c is a constant and n_0 is a fixed delay.

Thus, the conditions for PR are

$$H_0(-z).G_0(z)+H_1(-z).G_1(z)=0$$

$$H_0(z).G_0(z)+H_1(z).G_1(z)=cz^{-n_0}$$

The first condition implies that the reconstruction is aliasing-free and the second condition implies that the amplitude distortion has amplitude of one. It can be observed that the perfect reconstruction condition does not change if we switch the analysis and synthesis filters. There are a number of filters which satisfy these conditions. But not all of them give accurate wavelet transforms, especially when the filter coefficients are quantized. The accuracy of the wavelet transform can be determined after reconstruction by calculating the Signal to Noise Ratio (SNR) of the signal. Some applications like pattern recognition do not need reconstruction, and in such applications, the above conditions need not apply.

2.5 Classification of Wavelets

We can classify wavelets into two classes: (a) Orthogonal [12] and (b) Biorthogonal [13]. Based on the application, either of them can be used.

2.5.1 Features of Orthogonal Wavelet Filter Banks

The coefficients of orthogonal filters are real numbers. The filters are of the same length and are not symmetric. The low pass filter, G_0 and the high pass filter, H_0 are related to each other by

$$H_0(z) = z^{-N}G_0(-z^{-1}) \quad (2.9)$$

The two filters are alternated flip of each other. Also, for perfect reconstruction, the synthesis filters are identical to the analysis filters except for a time reversal. Orthogonal filters offer a high number of vanishing moments. This property is useful in many signal and image processing applications. They have regular structure which leads to easy implementation and scalable architecture.

2.5.2 Features of Biorthogonal Wavelet Filter Banks

In the case of the biorthogonal wavelet filters, the low pass and the high pass filters do not have the same length. The low pass filter is always symmetric, while the high pass filter could be either symmetric or anti-symmetric. The coefficients of the filters are either real numbers or integers.

For perfect reconstruction, biorthogonal filter bank has all odd length or all even length filters. The two analysis filters can be symmetric with odd length or one symmetric

and the other anti-symmetric with even length. Also, the two sets of analysis and synthesis filters must be dual. The linear phase biorthogonal filters are the most popular filters for data compression applications.

2.6 Wavelet Families

There are a number of basis functions that can be used as the mother wavelet for wavelet transformation. Since the mother wavelet produces all wavelet functions used in the transformation through translation and scaling, it determines the characteristics of the resulting wavelet transform. Therefore, the details of the particular application should be taken into account and the appropriate mother wavelet should be chosen in order to use the wavelet transform effectively.

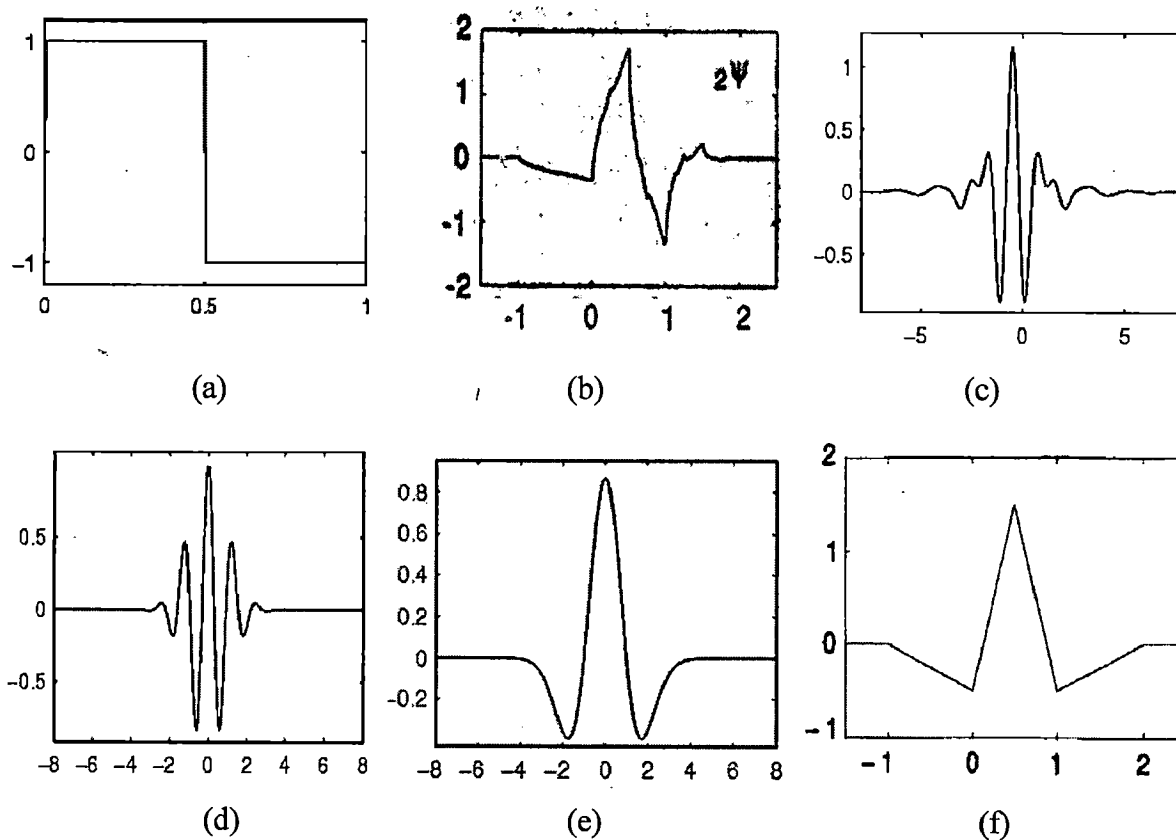


Fig.2.6 Wavelet families (a) Haar (b) Daubechies4 (c) Meyer (d) Morlet (e) Mexican Hat (f) CDF (2, 2)

Fig.2.6 illustrates some of the commonly used wavelet functions. Haar wavelet is one of the oldest and simplest wavelet. Therefore, any discussion of wavelets starts with the Haar wavelet. Daubechies wavelets are the most popular wavelets. They represent the foundations of wavelet signal processing and are used in numerous applications. Haar and Daubechies4 wavelets along with Meyer wavelets are capable of perfect reconstruction. The Meyer, Morlet

and Mexican Hat wavelets are symmetric in shape. The wavelets are chosen based on their shape and their ability to analyze the signal in a particular application. CDF (2, 2) wavelet is also known as the biorthogonal (5, 3) wavelet because of the filter length of 5 and 3 for the low and high pass filters, respectively.

2.7 Wavelet Based Image Compression

In DWT, the most prominent information in the signal appears in high amplitudes and the less prominent information appears in very low amplitudes. Data compression can be achieved by discarding these low amplitudes. The wavelet transforms enables high compression ratios with good quality of reconstruction. At present, the application of wavelets for image compression is one the hottest areas of research. Recently, the wavelet transforms have been chosen for the JPEG 2000 compression standard.

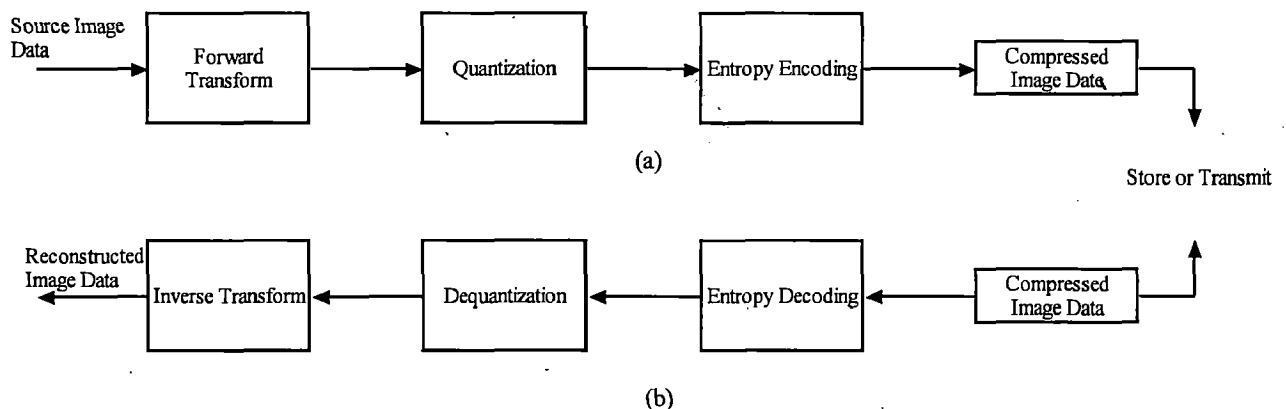


Fig.2.7 Block diagrams of the JPEG2000 (a) Encoder and (b) Decoder

Block diagram of JPEG2000 encoder as shown in Fig.2.7 (a) has mainly three components, the source encoder, the quantizer and the entropy encoder. The input signal (image) has a lot of redundancies that needs to be removed to achieve compression. These redundancies are not obvious in the time domain. Therefore, wavelet transform is applied to the input signal to bring the signal to the spectral domain. The spectral domain output from the transformer is quantized using some quantizing scheme. The signal then undergoes entropy encoding to generate the compressed signal.

Forward Transform

Forward Transform is the first major component of image compression system. A variety of linear transforms are available such as Discrete Fourier Transform (DFT), DCT

and DWT. DWT because of its advantages over other transform it is added in newer standard JPEG2000[14], which in focus of our work.

Quantizer

A quantizer reduces the precision of the values generated from the encoder and therefore reduces the number of bits required to save the transform co-coefficients. This process is lossy and quantization can be performed on each individual coefficient. This is known as Scalar Quantization (SQ). If it is performed on a group of coefficients together then it is called Vector Quantization (VQ).

Entropy Encoder

An entropy encoder does further compression on the quantized values. This is done to achieve even better overall compression. The various commonly used entropy encoders are the Huffman encoder, arithmetic encoder, and simple run-length encoder. For improved performance with the compression technique, it's important to have the best of all the three components.

2.7.1 Wavelet Transform as the Source Encoder

The discrete wavelet transform constitutes the function of the source encoder. Digital image is represented as a two-dimensional array of coefficients, each coefficient representing the brightness level in that point. We can differentiate between coefficients as more important ones, and lesser important ones. Most natural images have smooth color variations, with the fine details being represented as sharp edges in between the smooth variations. Technically, the smooth variations in color can be termed as low frequency variations, and the sharp variations as high frequency variations.

The low frequency components (smooth variations) constitute the base of an image, and the high frequency components (the edges which give the details) add upon them to refine the image, thereby giving a detailed image. Hence, the smooth variations are more important than the details.

Separating the smooth variations and details of the image can be performed in many ways. One way is the decomposition of the image using the discrete wavelet transform. Digital image compression is based on the ideas of sub-band decomposition or discrete wavelet transforms. Wavelets which refer to a set of basis functions are defined recursively from a set of scaling coefficients and scaling functions. The DWT is defined using these scaling functions and can be used to analyze digital images with superior performance than classical short-time fourier-based techniques, such as the DCT. The basic difference between

wavelet-based and fourier-based techniques is that short-time fourier-based techniques use a fixed analysis window, while wavelet-based techniques can be considered using a short window at high spatial frequency data and a long window at low spatial frequency data. This makes DWT more accurate in analyzing image signals at different spatial frequency, and thus can represent more precisely both smooth and dynamic regions in image. The compression system includes forward wavelet transform, a quantizer, and a lossless entropy encoder. The corresponding decompressed image is formed by the lossless entropy decoder, a de-quantizer, and an inverse wavelet transform. Wavelet-based image compression has good compression results in both rate and distortion sense.

2.7.2 Image Compression Techniques

Compression methods can be divided in two classes: lossless and lossy compression techniques:

Lossless compression

It guarantees that the original signal can be reconstructed without any errors. This is important for applications like the compression of text. For images, lossless compression is often used as the second step, after the lossy part.

Lossy compression

With lossy compression, we can obtain higher compression rates by not requiring the exact data to be reconstructed. Indeed, because the human visual system is not sensitive to some kinds of errors, the compression potential is much higher when we allow for small reconstruction errors.

Although the integer wavelet transform can be used for lossless compression due to its 100% invertible nature (in contrast to floating point wavelet transform implementations), image compression will usually be lossy due to the high compression rates that are required, except in sensitive applications areas like medical imaging where any data loss is not acceptable.

2.8 Features of Image Compression Using Wavelets

The key features of wavelet-based compression schemes are:

- Wavelets provide good compression ratios. Especially for high compression ratios, wavelets perform much better than competing technologies like JPEG [14], both in terms of signal-to-noise ratio and visual quality. Unlike JPEG, they show no blocking effect but allow for a graceful degradation of the whole image quality, while

preserving the important details of the image. The next version of the JPEG standard (JPEG 2000) will incorporate wavelet based compression techniques.

- The wavelet transform is a fast operation (linear to the amount of data), especially when implemented using the lifting scheme. The wavelet transform is symmetric: both the forward and the inverse transform have the same complexity, allowing fast compression and decompression routines.
- Multi-resolution allows for progressive transmission and zooming, without the need for extra storage. One can e.g. first transmit a thumbnail image, and gradually transmit and decompress more data to increase the resolution and overall image quality.
- Wavelets can not only be used for image compression, but also for various image processing operations. The possibility to combine image processing and compression is a very appealing factor. However, image processing cannot be done on the final encoded data stream: it must be done before the wavelet coefficients are quantized or encoded. But even then we win because the wavelet transform is a common factor in both image processing and image compression.

CHAPTER 3

THE LIFTING SCHEME

There are various architectures for implementing a two channel filter bank. A filter bank basically consists of a low pass filter, a high pass filter, decimators or expanders and delay elements.

In this chapter we consider direct form structure, polyphase structure briefly before we go into depth analysis of lifting structure.

3.1 Direct Form Structure

The direct form analysis filter consists of a set of low pass and high pass filters followed by decimators. The synthesis filter consists of upsamplers followed by the low pass and high pass filters as shown in Fig.3.1:

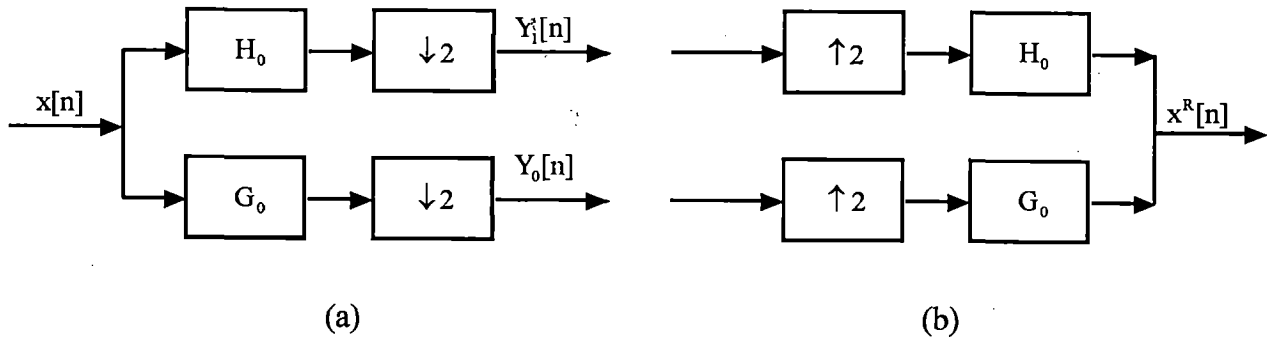


Fig.3.1 Direct form structure of (a) Analysis filter bank and (b) Synthesis filter

In the analysis filter bank, $x[n]$ is the discrete input signal, G_0 is the low pass filter and H_0 is the high pass filter. $\downarrow 2$ represents decimation by 2 and $\uparrow 2$ represents upsampling by 2. In the analysis bank, the input signal is first filtered and then decimated by 2 to get the outputs Y_0 and Y_1 . These operations can be represented by Eq.3.1 and Eq.3.2.

$$Y_0[k] = \sum_n X[n].G_0[2k-n] \quad (3.1)$$

$$Y_1[k] = \sum_n X[n].H_0[2k-n] \quad (3.2)$$

The output of the analysis filter is usually processed (compressed, coded or analyzed) based on the application. This output can be recovered again using the synthesis filter bank. In the

synthesis filter bank, Y_0 and Y_1 are first upsampled by 2 and then filtered to give the original input. For perfect output the filter banks must obey the conditions for perfect reconstruction.

3.2 Polyphase Structure

In the direct form analysis filter bank, it is seen that if the filter output consists of, say, N samples, due to decimation by 2 we are using only $N/2$ samples. Therefore, the computation of the remaining unused $N/2$ samples becomes redundant. It can be observed that the samples remaining after downsampling the low pass filter output are the even phase samples of the input vector X_{even} convoluted with the even phase coefficients of the low pass filter $G_{0\text{even}}$ and the odd phase samples of the input vector X_{odd} convoluted with the odd phase coefficients of the low pass filter $G_{0\text{odd}}$. The polyphase form takes advantage of this fact and the input signal is split into odd and even samples (which automatically decimates the input by 2), similarly, the filter coefficients are also split into even and odd components so that X_{even} convolves with $G_{0\text{even}}$ of the filter and X_{odd} convolves with $G_{0\text{odd}}$ of the filter. The two phases are added together in the end to produce the low pass output. Similar method is applied to the high pass filter where the high pass filter is split into even and odd phases $H_{0\text{even}}$ and $H_{0\text{odd}}$.

The polyphase analysis operation can be represented by the matrix Eq.3.3:

$$\begin{bmatrix} G_{0\text{even}} & G_{0\text{odd}} \\ H_{0\text{even}} & H_{0\text{odd}} \end{bmatrix} \begin{bmatrix} X_{\text{even}} \\ Z^{-1} X_{\text{odd}} \end{bmatrix} = H_p \begin{bmatrix} X_{\text{even}} \\ Z^{-1} X_{\text{odd}} \end{bmatrix} = \begin{bmatrix} Y_0 \\ Y_1 \end{bmatrix} \quad (3.3)$$

The filters with $G_{0\text{even}}$ and $G_{0\text{odd}}$ are half as long as G_0 , since they are obtained by splitting G_0 . Since, the even and odd terms are filtered separately, by the even and odd coefficients of the filters, the filters can operate in parallel improving the efficiency. The Fig.3.2 illustrates polyphase analysis and synthesis filter banks.

In the direct form synthesis filter bank, the input is first upsampled by adding zeros and then filtered. In the polyphase synthesis bank, the filters come first followed by upsamplers which again, reduces the number of computations in the filtering operations by half. Since, the number of computations reduced by half in both the analysis and synthesis filter banks, overall efficiency is increased by 50%. Thus, the polyphase form allows efficient hardware realizations.

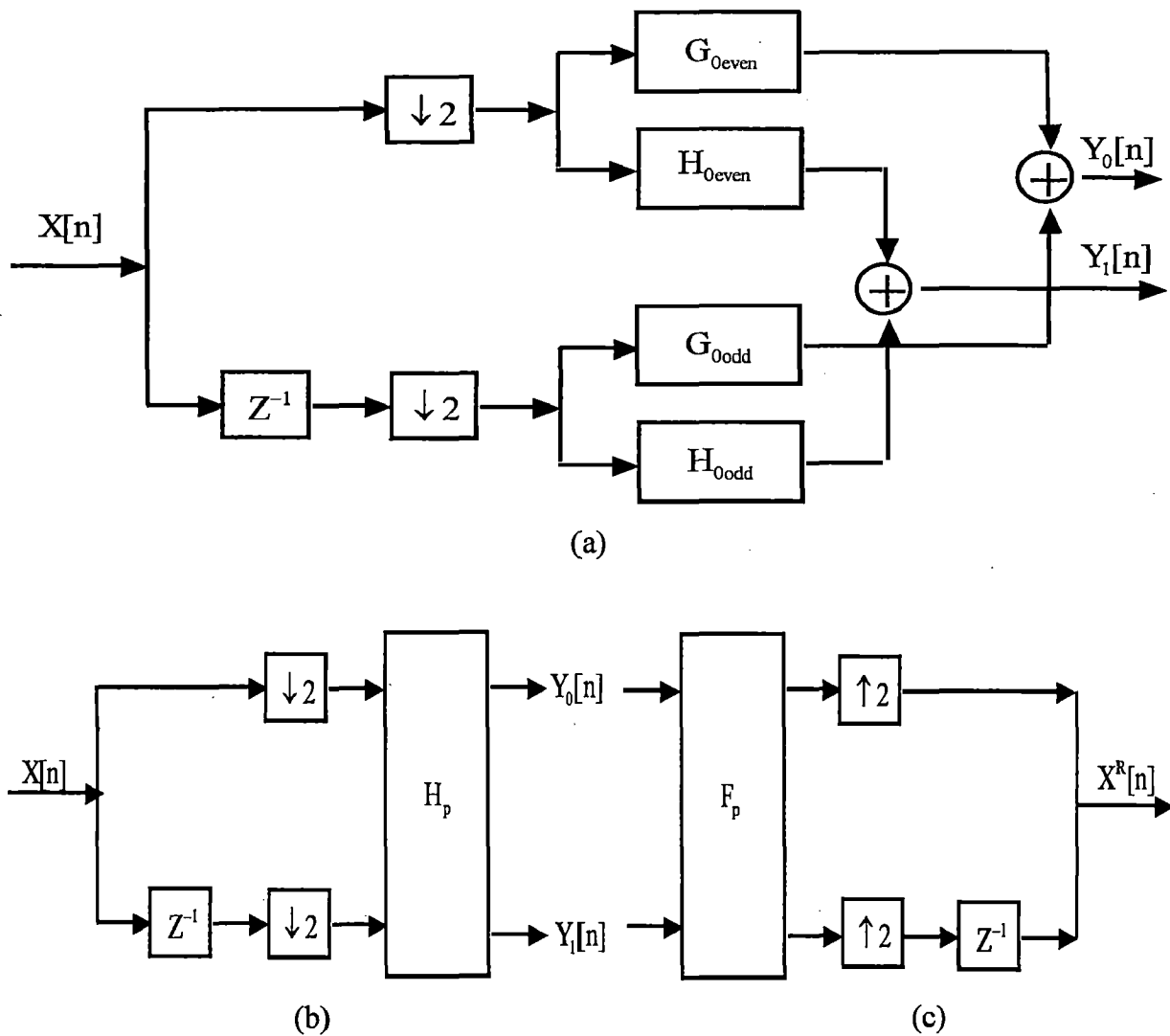


Fig.3.2 Polyphase structure of (a) Analysis filter bank (b) Equivalent representation of analysis filter bank and (c) Synthesis filter bank

3.3 Lifting Scheme

In 1994, Sweldens proposed a more efficient way of constructing the biorthogonal wavelet bases, which he called the lifting scheme [15]. The basic structure of the lifting scheme is shown in Fig.3.3. The input signal $s_{j,k}$ is first split into an update function to even and odd samples. The detail (i.e., high-frequency) coefficients $d_{j-1,k}$ of the signal are then generated by subtracting the output of a prediction function P of the odd samples from the even samples. The smooth coefficients (the low frequency components) are produced by adding the odd samples to the output of an update function U of the details. The computation of either the detail or smooth coefficients is called a lifting step.

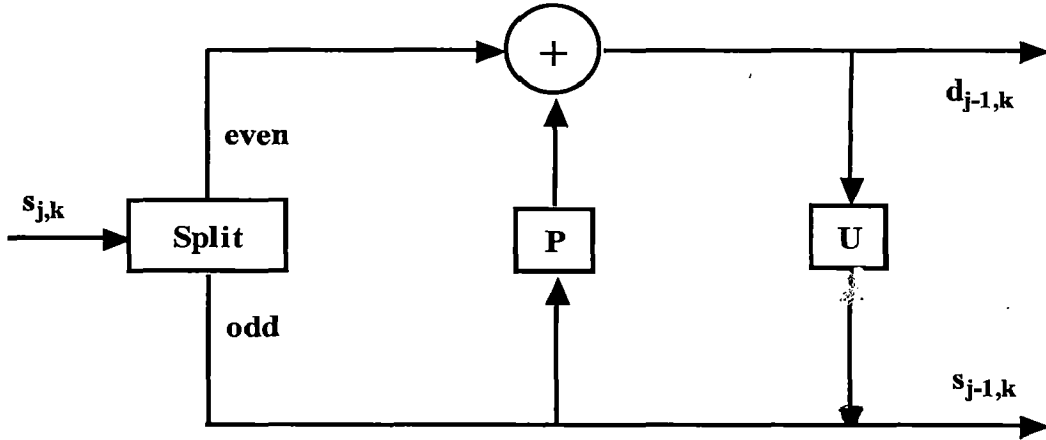


Fig.3.3 Lifting scheme

3.3.1 Factoring Wavelet Filters Into Lifting Scheme

Daubenchies and Sweldens [2] showed that every FIR wavelet or filter bank can be factored into a cascade of lifting steps that is, as a finite product of upper and lower triangular matrices and a diagonal normalization matrix. The high pass filter $g(z)$ and low pass filter $h(z)$ can thus be rewritten as

$$g(z) = \sum_{i=0}^{J-1} g_i z^{-i} \quad (3.4)$$

$$h(z) = \sum_{i=0}^{J-1} h_i z^{-i} \quad (3.5)$$

where J is the filter length. We can split the high pass and low pass filters into even and odd parts:

$$g(z) = g_e(z^2) + z^{-1} g_o(z^2) \quad (3.6)$$

$$h(z) = h_e(z^2) + z^{-1} h_o(z^2) \quad (3.7)$$

The filters can also be expressed as a polyphase matrix as follows:

$$P(z) = \begin{bmatrix} h_e(z) & g_e(z) \\ h_o(z) & g_o(z) \end{bmatrix} \quad (3.8)$$

Using the Eculidean algorithm which recursively finds the greatest common divisors of the even and odd parts of the original filters, the forward transform polyphase matrix $\tilde{P}(z)$ can be factored into lifting steps as follows:

$$\tilde{P}(z) = \prod_{i=1}^m \begin{bmatrix} 1 & 0 \\ -s_i(z^{-1}) & 1 \end{bmatrix} \begin{bmatrix} 1 & -t_i(z^{-1}) \\ 0 & 1 \end{bmatrix} \times \begin{bmatrix} \frac{1}{K} & 0 \\ 0 & K \end{bmatrix}, m \leq K \quad (3.9)$$

where $s_i(z)$ and $t_i(z)$ are Laurent polynomial corresponding to the update and predict steps, respectively, and K is a non zero constant. The inverse DWT is described by the following equation:

$$P(z) = \prod_{i=1}^m \begin{bmatrix} 1 & s_i(z) \\ 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 \\ t_i(z) & 1 \end{bmatrix} \begin{bmatrix} K & 0 \\ 0 & \frac{1}{K} \end{bmatrix} \quad (3.10)$$

3.3.2 Chohen-Daubechies-Feauveau (CDF) (2,2) Wavelet Using Lifting Scheme

The analyzing filter pair for the CDF [13] with 2 vanishing moments for both primal lifting and dual wavelet function is (up to a normalization factor of $\sqrt{2}$)

$$\tilde{h}(z) = -\frac{1}{8}z^{-2} + \frac{1}{4}z^{-1} + \frac{3}{4} + \frac{1}{4}z - \frac{1}{8}z^2 \quad (3.11)$$

$$\tilde{g}(z) = \frac{1}{4}z^{-2} - \frac{1}{2}z^{-1} + \frac{1}{4} \quad (3.12)$$

Following the above procedure from section 3.2 we can factor the analysis polyphase matrix of a CDF(2,2) wavelet

$$\tilde{P}(z) = \begin{bmatrix} 1 & 0 \\ 0 & -\frac{1}{2} \end{bmatrix} \begin{bmatrix} 1 & \frac{1}{4} + \frac{1}{4}z \\ 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 \\ -\frac{1}{2}z^{-1} - \frac{1}{2} & 1 \end{bmatrix} \quad (3.13)$$

The lifting structure for the CDF (2,2) is shown in Fig.3.4.

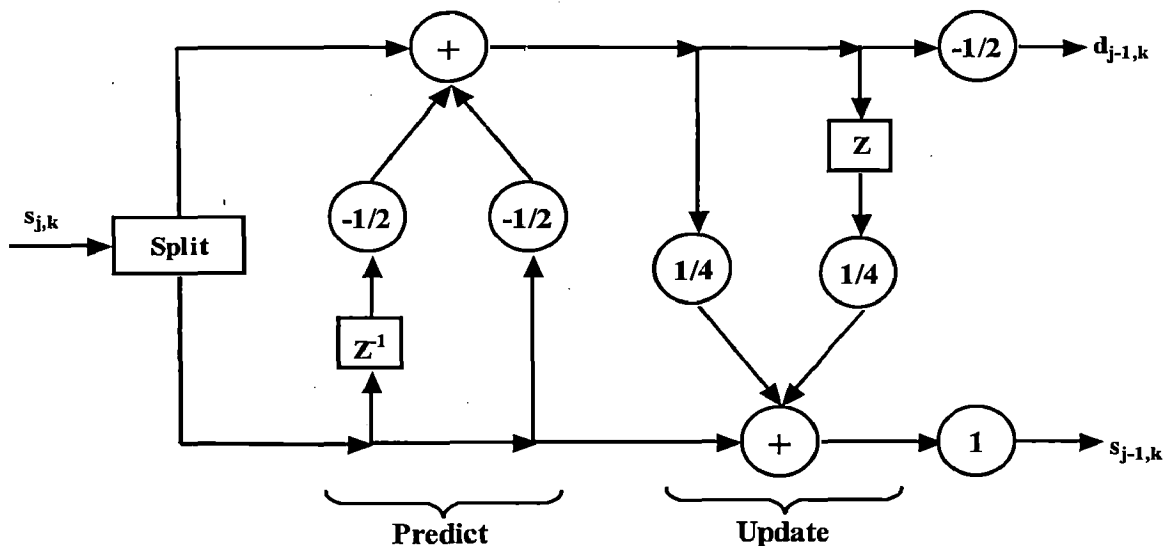


Fig.3.4 Lifting structure for CDF (2,2) wavelet

3.3.3 Integer-To-Integer Transform

Integer arithmetic is generally much faster than floating-point arithmetic. Furthermore, integer numbers are more efficient to encode and take less space to store. As a consequence, we prefer to deal with integer numbers and integer arithmetic rather than floating-point. In many applications, especially in image processing, the input data consists of integer samples. Wavelet coefficients, on the contrary, are floating point values. Thus since the filter coefficients are floating-point numbers, even if the input data is integer applying the lifting and update steps on the data will result in floating-point numbers.

Fortunately the lifting scheme can be easily modified to map integers to integers, which are in addition fully reversible [3], [16] and thus allows a perfect reconstruction of the original image.

3.4 Advantages of Lifting scheme

Lifting scheme has the following advantages, when compared to other 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 another factor of two. Hence it is also referred to as Fast Lifting Wavelet Transform (FLWT).
- All operations within lifting scheme can be done entirely parallel while the only sequential part is the order of lifting operations.
- Lifting can be done in-place. An auxiliary memory is not needed since it does not need other samples than the output of the previous lifting step. At every summation point the old stream is replaced by the new one at every summation point.
- Lifting Scheme allows integer-to-integer transform while keeping a perfect reconstruction of the original data set. This is important for hardware implementation and lossless image coding.
- Lifting allows adaptive wavelet transforms. This means that the analysis of a function can start from the coarsest level, followed by finer levels by refining in the areas of interest.

CHAPTER 4

INTRODUCTION TO FPGA IMPLEMENTATION

For hardware implementation Spartan 3E FPGA kit is used. Spartan 3E FPGA kit is most widely used FPGA kit because of its advantages like low power, low cost etc. In this thesis, code is written in VHDL and implemented on Spartan 3E FPGA kit using Xilinx ISE simulator.

Before going in detail (covered in next chapter), in this chapter we look at the overview of Spartan 3E architecture, FPGA design flow and hardware descriptive language VHDL.

4.1 Architectural Overview

The Spartan-3E family architecture [17] consists of five fundamental programmable functional elements:

- **Configurable Logic Blocks (CLBs)** contain flexible Look-Up Tables (LUTs) that implement logic plus storage elements used as flip-flops or latches. CLBs perform a wide variety of logical functions as well as store data.
- **Input/Output Blocks (IOBs)** control the flow of data between the I/O pins and the internal logic of the device. Each IOB supports bidirectional data flow plus 3-state operation. Support a variety of signal standards, including four high-performance differential standards. Double Data-Rate (DDR) registers are included.
- **Block RAM** provides data storage in the form of 18-Kbit dual-port blocks.
- **Multiplier Blocks** accept two 18-bit binary numbers as inputs and calculate the product.
- **Digital Clock Manager (DCM) Blocks** provide self-calibrating, fully digital solutions for distributing, delaying, multiplying, dividing, and phase-shifting clock signals.

These elements of Spartan 3E family architecture are shown in Fig.4.1. A ring of IOBs surrounds a regular array of CLBs. Each device has two columns of block RAM except for the XA3S100E, which has one column. Each RAM column consists of several 18-Kbit RAM blocks. Each block RAM is associated with a dedicated multiplier. The DCMs are positioned in the center with two at the top and two at the bottom of the device.

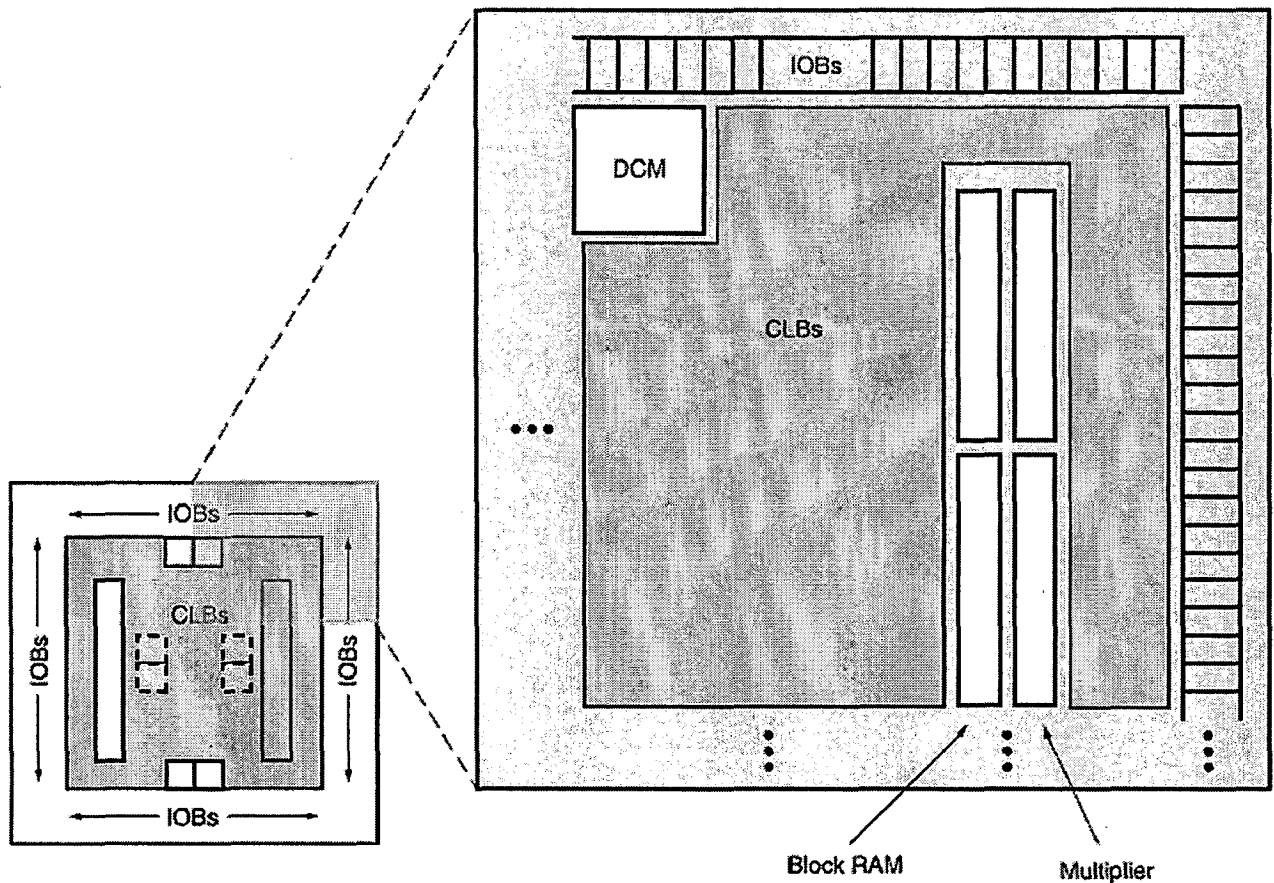


Fig.4.1 Spartan 3E family architecture

The XA Spartan-3E family features a rich network of traces that interconnect all five functional elements, transmitting signals among them. Each functional element has an associated switch matrix that permits multiple connections to the routing.

4.2 FPGA Design Flow

The FPGA design flow [18] comprises the following steps: design entry, design synthesis, design implementation, verification and programming device. Design verification, which includes both functional verification and timing verification, takes places at different points during the design flow as shown in Fig.4.2.

Design Entry

Design entry is the first step in the design flow. During design entry, we create our source files based on your design objectives. You can create your top-level design file using a Hardware Description Language (HDL), such as VHDL, Verilog or using a schematic. We can use multiple formats for the lower-level source files in your design.

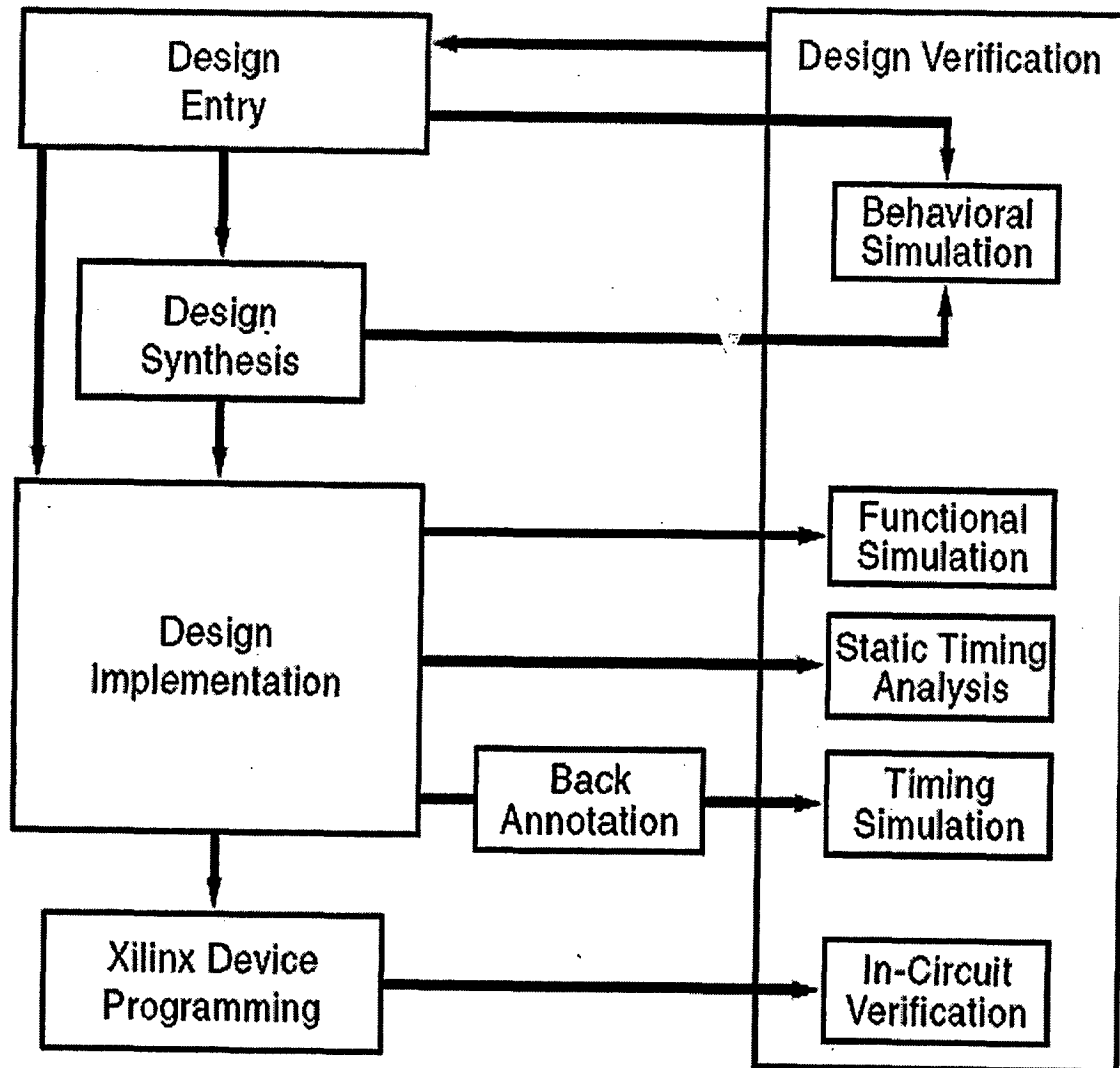


Fig.4.2 FPGA design flow for implementation

Design Synthesis

After design entry and optional simulation, you run synthesis. During this step, VHDL, Verilog, or mixed language designs become netlist files that are accepted as input to the implementation step.

Implementation

After synthesis, we run design implementation, which converts the logical design into a physical file format that can be downloaded to the selected target device. From Project Navigator, we can run the implementation process in one step, or we can run each of the implementation processes separately.

Verification

We can also verify the functionality of our design at several points in the design flow. We can use simulator software to verify the functionality and timing of our design or a

portion of our design. The simulator interprets VHDL or Verilog code into circuit functionality and displays logical results of the described HDL to determine correct circuit operation. Simulation allows you to create and verify complex functions in a relatively small amount of time. We can also run in-circuit verification after programming your device.

Device Configuration

After generating a programming file, we configure your device. During configuration, we generate configuration files and download the programming files from a host computer to a FPGA kit.

4.3 VHDL

VHDL stands for VHSIC (Very High Speed Integrated Circuits) Hardware Description Language. The other widely used hardware description language is Verilog [19]. Both are powerful languages that allow you to describe and simulate complex digital systems.

Although these languages look similar as conventional programming languages, there are some important differences. A hardware description language is inherently parallel, i.e. commands, which correspond to logic gates, are executed (computed) parallel, as soon as a new input arrives. These languages can be made to behave as sequential for design of sequential systems. A HDL program mimics the behavior of a physical, usually digital system. It also allows incorporation of timing specifications (gate delays) as well as to describe a system as an interconnection of different components.

Code in VHDL [20], [21] can be written in behavioral style, data flow or structural style.

Behavioral Modeling

In this modeling style, the behavior of the entity is expressed using sequentially executed, procedural code, which is very similar in syntax and semantics to that of a high-level programming language like C. A process statement is the primary mechanism used to model the behavior of an entity.

Data Modeling

A dataflow model specifies the functionality of the entity without explicitly specifying its structure. This functionality shows the flow of information through the entity, which is expressed primarily using concurrent signal assignment statements and block statements.

Structural Modeling

In this style of modeling an entity is modeled as a set of components connected by signals, that is, as a netlist. The behavior of the entity is not explicitly apparent from its model. The component instantiation statement is the primary mechanism used for describing such a model of an entity.

CHAPTER 5

LIFTING BASED DWT ARCHITECTURES

With the inclusion of lifting based DWT in newer standard JPEG 2000, lifting based DWT architectures has been hot field and several efficient architectures [22-25] have been proposed since then. Mainly work has been done in reducing the memory, parallel implementation and 100% utilization of hardware.

In this chapter we study lifting based DWT, parallel architecture for lifting based DWT and proposed parallel architecture for lifting based CDF (2, 2) wavelet. Proposed parallel architecture is removes the line buffers at the column processor and produces output in exactly $N^2 / 4$ cycles with 100% hardware utilization but with an extra row processor.

5.1 Lifting Based DWT Architecture

In lifting based DWT architecture two processors are used one for row processing and other for column processing. Lifting based DWT architecture is shown in Fig.5.1, row processor takes the image data line by line, process the data and store the processed data. When required number of line buffers are filled column processor takes the data from line buffer memory and produces LL, LH or HL, HH data correspondingly. After all the rows and columns are processed we get the level 1 decomposed image. To increase the level of decomposition we have iterate the processed data as shown in Fig.5.2. In Fig.5.2 transform module is complete lifting based DWT architecture and it requires another external memory for $\frac{N}{2} \times \frac{N}{2}$ which decreases iteratively by the same factor. The architecture of row processors can be designed by directly mapping the lifting factorization of CDF (2, 2) wavelet, which can also be extended to that of column processor.

The main disadvantages of lifting based DWT architecture are number of clock cycles it takes to process data is almost $N^2 / 2$ and internal line buffer memory.

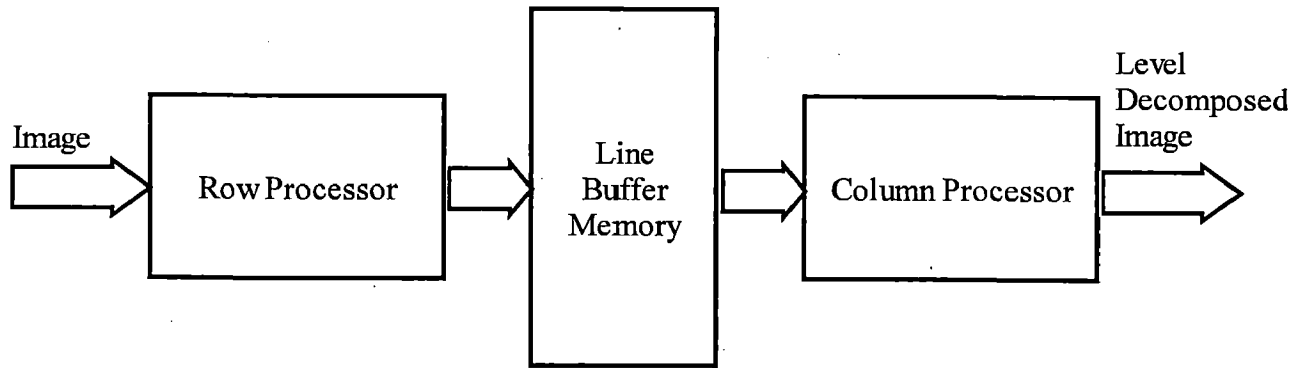


Fig.5.1 Lifting based DWT architecture

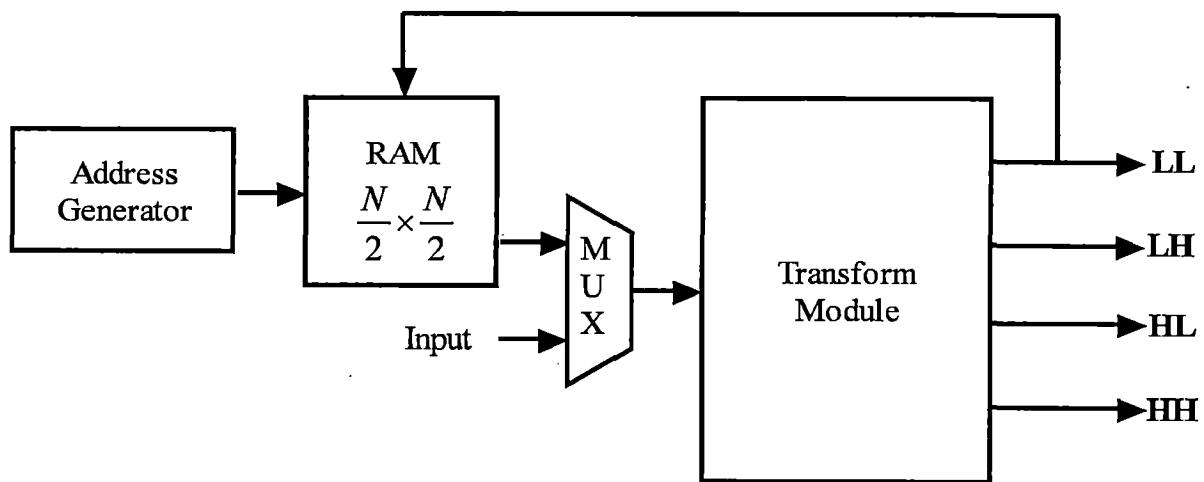


Fig.5.2 Block diagram for increasing level of decomposition [6]

5.2 Parallel Architecture for Lifting Based DWT

The line-based parallel architecture [25] for 1-level lifting based DWT is shown in Fig.5.3, which is composed of an input buffer unit (IBU) and a wavelet transform module (WTM). The wavelet transform module (WTM) is a four-input four-output architecture that includes two row-wise 1D DWT modules (R_WT1 and R_WT2) for performing horizontal filtering, two column-wise 1D DWT modules (C_WT1 and C_WT2) for performing vertical filtering, and a scale normalization unit (SNU, which work in parallel and pipelined. SNU integrates the scale normalization operations required respectively in row transform and column transform to reduce efficiently the number of multipliers required in the architecture of 2-D DWT because the scale normalization factor for low-pass filtering is reciprocal to that of high-pass.

Four input samples are required simultaneously to input to WTM in each internal working clock cycle with four sub bands coefficients generated synchronously. Two input

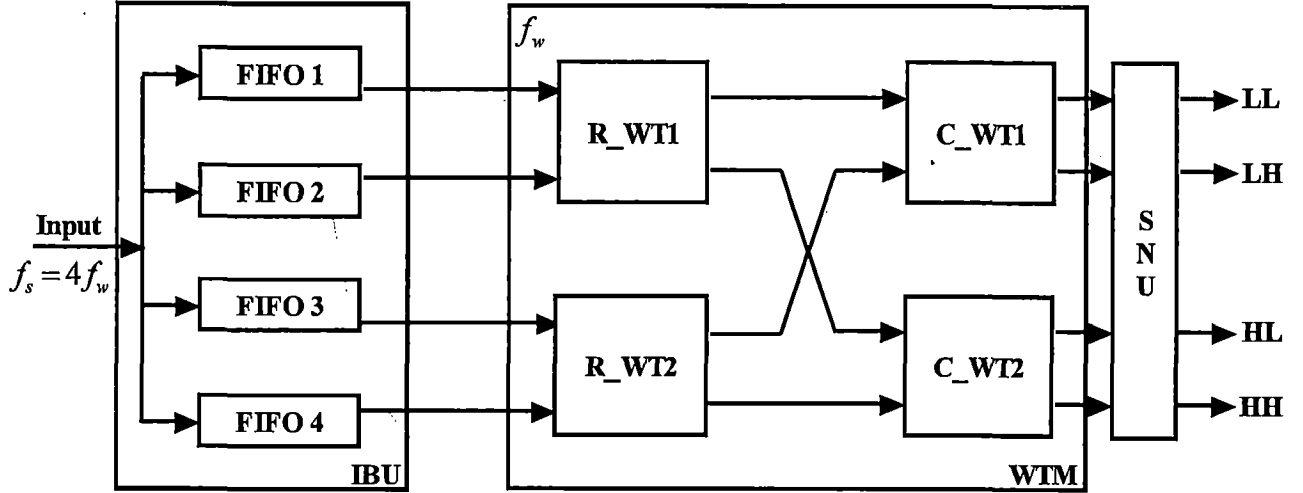


Fig.5.3 Parallel architecture for lifting based DWT [25]

samples are from the even-numbered row, and the other two are from the odd-numbered row. Two lines of signals are required to input simultaneously to the WTM. Since the data samples are assumed to input in a line-by-line way, an input buffer unit (IBU) is required. The IBU can be implemented by four first-in-first-out (FIFO) (named FIFO 1, FIFO 2, FIFO 3, FIFO 4 with sizes of respectively about $N/2, N/2, N/2, N/2$, where N represents the width of image), which are used to store the samples separately being from even-row-even-column, even-row-odd-column, odd-row-even-column and odd-row-odd-column. In order to provide 4 samples at an internal working clock cycle, 4 times faster clock rate, i.e. $f_s = 4f_w$ (f_s denotes input data sampling frequency, and f_w denotes internal working frequency, respectively), is required to acquire input data samples. Let $x_{ee}(m, n)[x_{oe}(m, n)]$ represent the samples of even-numbered row and even-numbered column [odd-numbered row and even-numbered column], and $x_{eo}(m, n)[x_{oo}(m, n)]$ represent the samples of even-numbered row and odd-numbered column [odd-numbered row and odd-numbered column], respectively. And let $L_e(m, n)$ and $[L_o(m, n)$ and $H_o(m, n)]$ to denote the low-frequency coefficients and high-frequency coefficients of the even-numbered rows [odd-numbered rows], respectively. In each internal clock cycle, four inputs, $x_{ee}(m, n)$ and $x_{oe}(m, n)$, as well as $x_{eo}(m, n)$ and $x_{oo}(m, n)$, are respectively inputted to R_WT1 and R_WT2 in parallel. R_WT1 generates one low-frequency coefficient $L_e(m, n)$ and one high-frequency coefficient $H_e(m, n)$ in each clock cycle, while R_WT2 produces one low-frequency coefficient $L_o(m, n)$ and one high-frequency coefficient $H_o(m, n)$ in each clock cycle. The outputs of R_WT1 and R_WT2 are

then pipelined to C_WT1 and C_WT2, i.e., $L_e(m,n)$ and $L_o(m,n)$ are inputted to C_WT1, and decomposed into sub bands low-low frequency (LL) and low-high frequency (LH) components by scale normalization operations, meanwhile $H_e(m,n)$ and $H_o(m,n)$ are inputted in parallel to C_WT2, and decomposed into sub bands high-low frequency (HL) and high-high frequency (HH) components by scale normalization operations as well. The architecture of row-wise wavelet transform module can be designed by directly mapping the lifting factorization of chosen wavelet filter, which can also be extended to that of column-wise wavelet transform module.

5.3 Proposed Parallel Architecture For Lifting Based CDF (2,2) Wavelet

The proposed architecture as shown in Fig.5.4 is composed of three row wise 1-D DWT modules (R_WT1, R_WT2 and R_WT3) and two column wise 1-D DWT modules (C_WT1 and C_WT2). In normal CDF (2,2) implementation, first all the rows are processed then all the columns are processed. So, for parallel processing column processor not only requires present two processed row elements but also next processed row; in order to supply future row processed elements for column processor, extra row processor is used which not only avoids extra line buffering at column processor but also decomposes the image in exactly $N^2/4$ cycles. Therefore, nine input samples are required simultaneously in each internal working clock cycle with four sub bands coefficients generated synchronously. R_WT1 module requires even-row-even-column, even-row-odd column, even-row-next even column and R_WT2 module requires odd-row-even-column, odd-row-odd column, odd-row-next even column and R_WT3 requires next even-row-even-column, next even-row-odd column, next even-row-next even column. R_WT1, R_WT2 and R_WT3 row processors operates in parallel and produces low frequency and high frequency components simultaneously, then column processors C_WT1 and C_WT2 takes low frequency components and high frequency components from row processors respectively and generates LL,LH,HL and HH components in the same clock cycle. The architecture of row processors can be designed by directly mapping the lifting factorization of CDF (2,2) wavelet, which can also be extended to that of column processor.

To increase the level of decomposition of the image the architecture can be modified to Fig. 5.2. In Fig.5.2 transform module is replaced with proposed architecture and remaining modules are exactly same as before. As the level of decomposition increases the compression ratio increases.

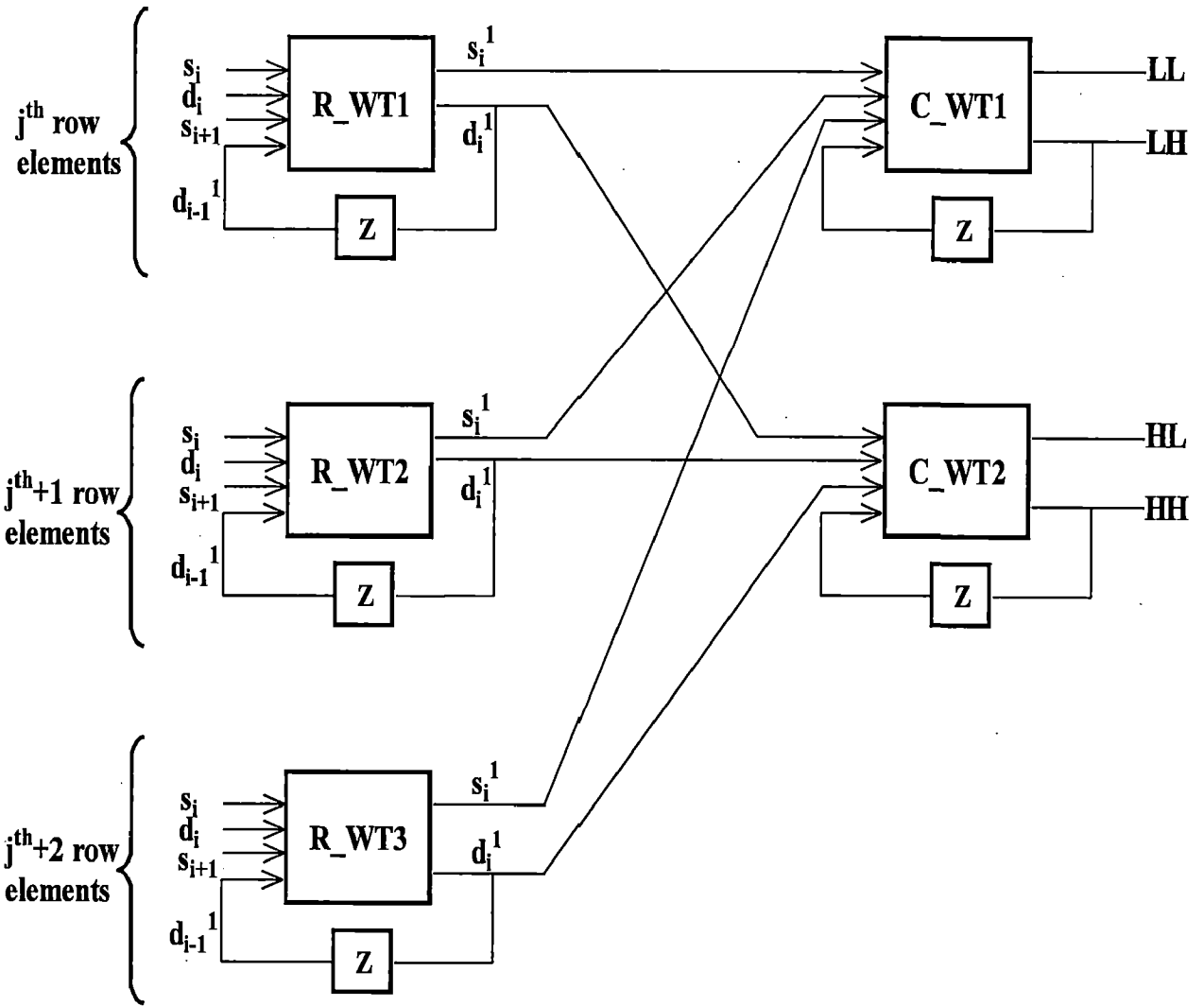


Fig.5.4 Proposed parallel architecture for lifting based CDF (2,2) wavelet

CHAPTER 6

IMPLEMENTATION

An improved parallel architecture for implementing 2D-DWT of CDF (2, 2) is proposed in this thesis. First the proposed architecture is tested using MATLAB software, then the hardware implementation of tested architecture is implemented on FPGA using VHDL language.

In this chapter, algorithm for CDF (2, 2) wavelet is briefly explained, then its software implementation and results. Finally its hardware implementation on FPGA and simulation results obtained using Modelsim are explained.

6.1 Cohen-Daubechies-Feauveau (CDF) (2, 2) Wavelet

The Cohen-Daubechies-Feauveau (CDF) (2, 2) wavelet [13] is widely used for image compression because of its good compression characteristics. The original filters have $5 + 3 = 8$ filter coefficients as shown in Eq.3.11 and Eq.3.12, whereas an implementation with the lifting scheme has only $2 + 2 = 4$ filter coefficients. The forward and reverse filters are shown in Table 6.1(a) and 6.1(b). Fractional numbers are converted to integers at each stage. Though such an operation adds non-linearity to the transform, the transform is fully invertible as long as the rounding is deterministic. In Table 6.1 x represents the image pixel

Table 6.1 CDF (2, 2) wavelet with lifting scheme (a) Forward transform (b) Inverse transform
(a)

Splitting	$s_i \leftarrow x_{2i}$ $d_i \leftarrow x_{2i+1}$
Dual Lifting	$d_i \leftarrow d_i - \frac{1}{2}(s_i + s_{i+1})$
Primal Lifting	$s_i \leftarrow s_i + \frac{1}{4}(d_{i-1} + d_i)$

(b)

Inverse Primal Lifting	$s_i \leftarrow s_i - \frac{1}{4}(d_{i-1} + d_i)$
Inverse Dual Lifting	$d_i \leftarrow d_i + \frac{1}{4}(s_i + s_{i+1})$
Merging	$x_{2i} \leftarrow s_i$ $x_{2i+1} \leftarrow d_i$

values and s stands for the summing or the low pass coefficients and d stands for the difference or the high pass coefficients. In image compression, one row or column of an image is regarded as a signal. Fig.6.1 shows the row and column formation at level 1, level 2 and level 3 for a 512 X 512 pixel image.

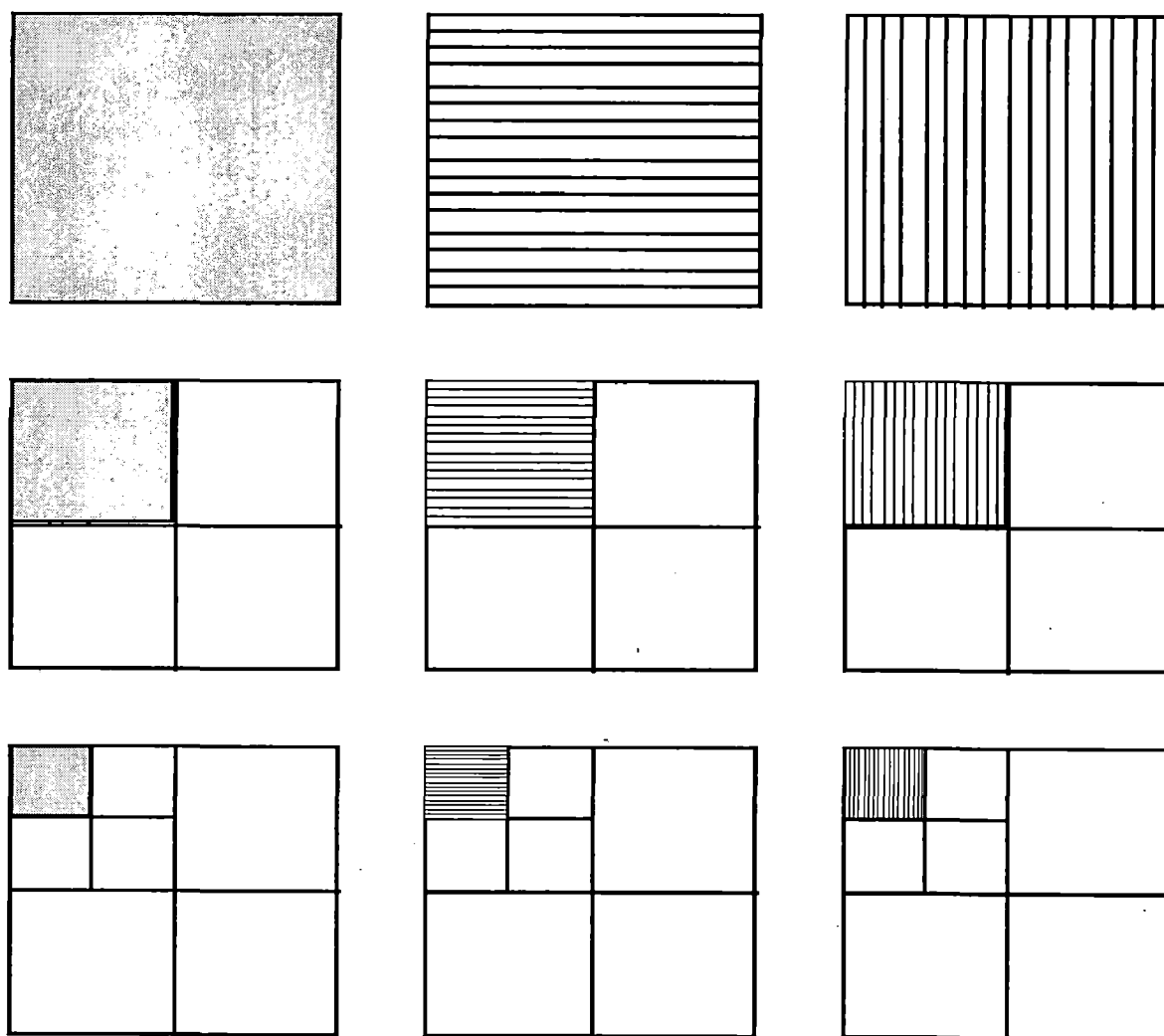


Fig.6.1 Rows and columns of level 1, 2 and 3 decomposition of an image

Every row or column is arranged and assigned in the following manner:

$$S_0 D_0 S_1 D_1 S_2 D_2 S_3 D_3 S_4 D_4 S_4 \dots$$

In this algorithm, in case of rows it is for row processor whereas in case of column it is for column processor in the parallel architecture of lifting based CDF (2,2) wavelet. The odd pixels should be processed first, then the even pixel due to the data dependency. There are a total of three levels based on the 3-level decomposition wavelet transform algorithm discussed

above. In each level, the rows are processed first then the columns. Each level's signal length (amount of each row/column pixels) is half of the previous level.

Algorithm

For every row or column:

Repeat until end of row or column:

If begin or end of row or column

begin

$D_0 = D_0 + D_0 - S_0 - S_0$

$S_0 = S_0 + (2 * D_0 / 8)$

End

Else

begin

$D_i = D_i + D_i - S_i - S_{i+1}$

$S_i = S_i + ((D_{i-1} + D_i) / 8)$

End

End

6.2 Software Implementation and Results

MATLAB is powerful mathematical modeling software which is used for the software implementation. MATLAB is also used for the development of software modules to convert image files to memory files, and vice versa. The MATLAB programming environment provided all the necessary functions and tools needed to achieve this.

So I have used MATLAB software for first testing my proposed parallel architecture of DWT for image compression before its actual hardware implementation. Code for row processor and column processor functions are written using the above algorithm. Main program reads the image and sends the corresponding values to the row processors and column processors, then the computed results generated by row and column processors are stored in the same place, after complete level decomposition the results are aligned according to LL, LH, HL and HL components.

The test image used is the 512 X 512 pixel image 'Lena.jpg' as shown in Fig.6.2. Level 1, level 2 and level 3 decomposed images are shown in Fig.6.3, 6.4 and 6.5

ively. From these level decomposed images exact image can be reproduced using
e transform without any loss of information.



Fig.6.2 Original Lena image



Fig.6.3 Image after level 1 decomposition



Fig.6.4 Image after level 2 decomposition

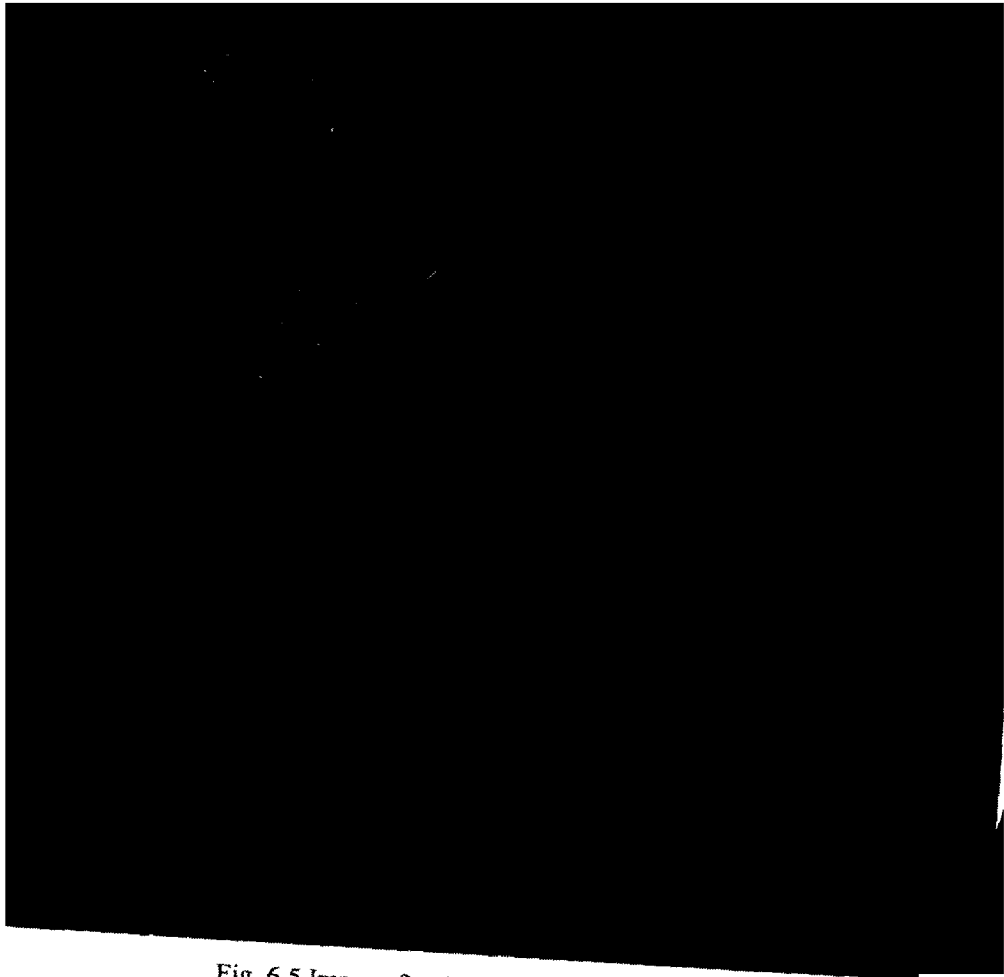


Fig. 6.5 Image after level 3 decomposition

Hardware Implementation and Results

Before any hardware implementation and testing is performed, the correct functionality using the Modelsim functional simulation tool. The necessary environment for complete functional simulation is set up on the target hardware, i.e., Xilinx Spartan 3E FPGA. The Modelsim tool features high performance and platform adaptability. It also allows for modification and verification of the code simultaneously.

Simulation result of top module in Modelsim is shown in Fig.6. From the simulation result we can see that incoming image row data are stored in FIFO's. And also we can note that row processors work are positive edge triggered. Results obtained from column processors are negative edge triggered. Results obtained from row processor during in the same clock cycle. Row processor works in parallel and produces LL, LH, HL, HH components.

clock cycle. Thus complete image can be decomposed in exactly $N^2/4$ cycles without any need of line buffers at the column processors.

After the correct functional verification by Modelsim, the VHDL description is synthesized using XST tool. A number of iterations are made to match the VHDL description to the timing and synthesis constraints. Subsequently, the implementation tool of the Xilinx ISE 8.1 is used to implement the design. After implementation information regarding device utilization summary and timing report is generated as shown in Table 6.1 and Table 6.2 respectively, before burning on FPGA Table 6.1 gives an overview of FPGA after implementation. Table 6.2 gives information regarding maximum frequency of operation, setup time and hold time.

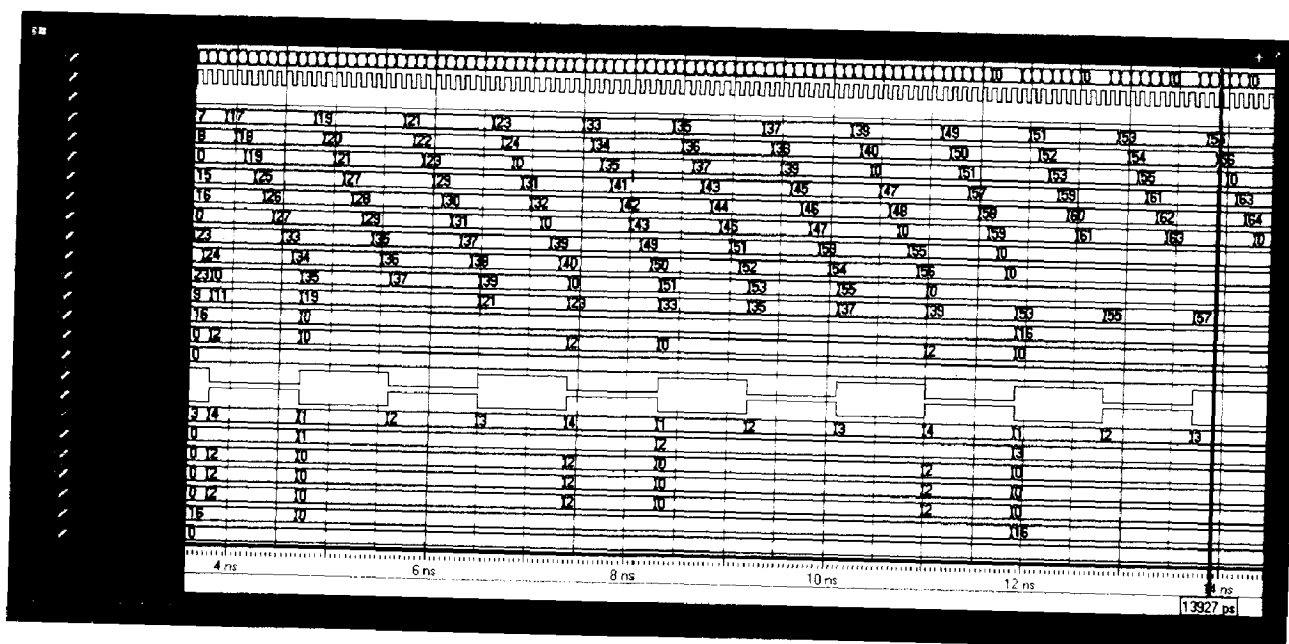


Fig.6.6 Simulation result of top module

Table 6.2 Device utilization summary

Number Of Slices	351 out of 4656	7%
Number Of Slice Flip Flops	103 out of 9312	1%
Number Of 4 input LUTs	632 out of 9312	6%
Number of bonded IOBs	128 out of 232	55%
Number of GCLKs	1 out of 24	4%

Table 6.3 Timing report

Minimum period	6.846ns
Maximum Frequency	146.071MHz
Minimum input arrival time before clock (Setup Time)	5.423ns
Maximum output required time after clock (Hold Time)	46.517ns

After implementation Bit file is generated then the hardware implementation is carried out by programming the FPGA through the Universal Serial Bus (USB) port of a computer. Once the Bit file is burned on FPGA, program succeeded notification will be displayed as shown in Fig.6.7, which indicates our code is successfully translated on to FPGA.

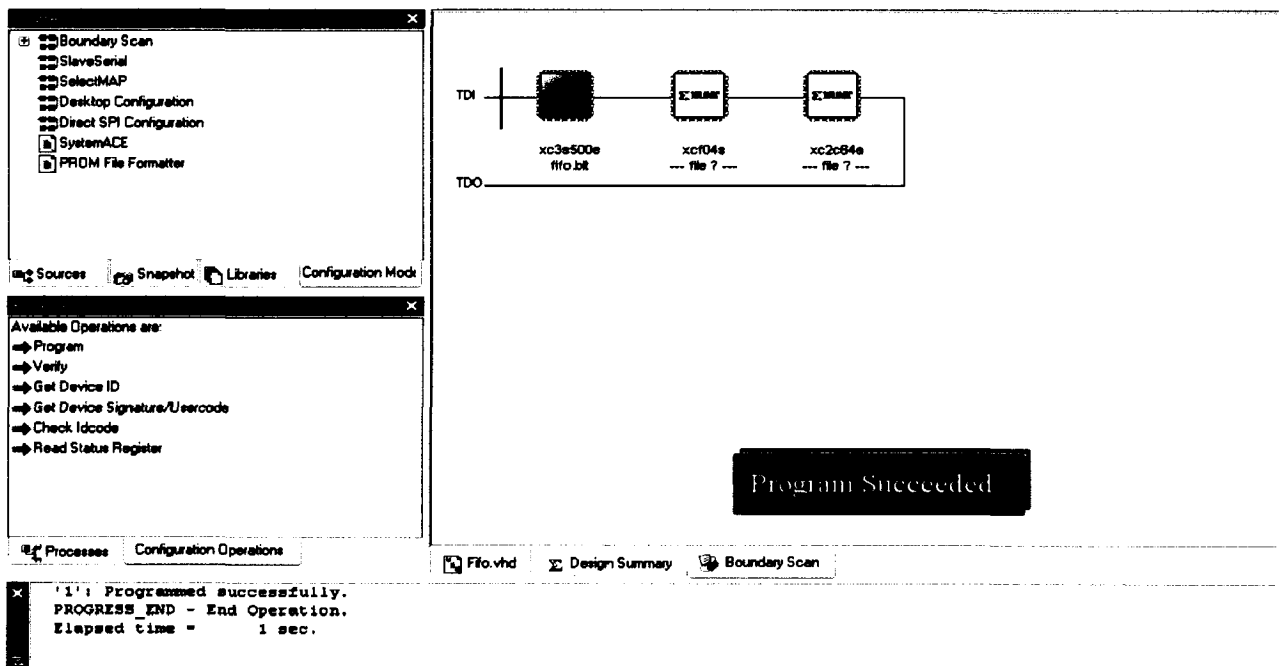


Fig.6.7 Programmed succeeded screen shot

CHAPTER 7

CONCLUSION AND SCOPE FOR FUTURE WORK

7.1 Conclusion

In this thesis, an improved parallel architecture for implementation of lifting based CDF(2,2) wavelet is proposed. Proposed architecture is first tested using MATLAB software, then VHDL code is written in Xilinx ISE 8.1 and simulated results are verified using Modelsim software. Subsequently, the implementation tool of the Xilinx ISE is used to implement the design followed by the generation of the bit file. The hardware implementation is carried out by programming the FPGA through the USB port of a computer.

The following conclusions can be made from the results obtained using the proposed parallel architecture for CDF (2,2) wavelet:

- ✓ Compared to previous parallel architecture [25], proposed architecture can perform level 1 decomposition of an $N \times N$ image in exactly $N^2 / 4$ working clock cycles.
- ✓ It does not require any line buffers at the column processor as in case of previous parallel architectures.
- ✓ Works with 100% hardware utilization.

7.2 Scope for Future Work

Obviously there is a scope for future work for my proposed architecture. The possible improvements in the future are listed as below:

- Similarly, an inverse transform can be implemented for extracting the original image from compressed image.
- Work can be extended to color image implementation either by multiplexing the bits or by multiplying the hardware three times, one for each color.
- Architecture for other wavelets can be developed similarly with reduced clock cycles and line buffers.

REFERENCES

- [1] C. Christopoulos, A. Skodras and T. Ebrahimi, "The JPEG2000 still image coding system: an overview," *IEEE Trans. on Consumer Electronics*, vol.46, no. 4, pp.1103-1127, 2000.
- [2] I. Daubechies, W. Sweldens, "Factoring wavelet transforms into lifting schemes," *J. Fourier Anal. Appl.*, vol.4, pp.247-269, 1998.
- [3] A .R. Calderbank, I. Daubechies, W. Sweldens, and B.L. Yeo, "Wavelet transform that map integers to integers," *Applied and Computational Harmonic Analysis (ACHA)*, vol.5, no.3, pp.332-369, 1998.
- [4] K. Andra, C. Chakrabarti and T. Acharya, "VLSI architecture for lifting-based forward and inverse wavelet transform", *IEEE Trans. on Signal Processing*, vol.50, no.4, pp.966-977, 2002.
- [5] C. Chrysafis, and A. Ortega, "Line-based, reduced memory, wavelet image compression," *IEEE Trans. on Image Processing*, vol.9, no.3, pp.378-389, 2000.
- [6] P. Wu, and L. Chen, "An efficient architecture for two-dimensional discrete wavelet transform," *IEEE Trans. on Circuits and Systems for Tech.*, vol.11, no.4, pp.536-545, 2001.
- [7] D. Bhatia, "Reconfigurable computing," 10th International Conference on VLSI Design, Hyderabad, India, pp. 356-359, 1997.
- [8] R. Polikar, "Wavelet tutorial", http://users.rowan.edu/~polikar/WAVELETS/WT_tutorial.html (Last accessed on 28th June 2008).
- [9] A. Abbate, C. M. DeCusatis and P. K. Das, "Wavelets and subbands: Fundamentals and applications", Brikhäuser, 2002.
- [10] S. G. Mallat, "A Theory for multiresolution signal decomposition: The wavelet representation," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol.11, no.7, pp.674-693, 1989.
- [11] S. G. Mallat, "Multifrequency channel decompositions of images and wavelet models," *IEEE Trans. on Acoustics, Speech and Signal Processing*, vol.37, no. 12, pp.2091-2110, 1989.
- [12] I. Daubechies, "Orthonormal bases of compactly supported wavelets," *Communications on Pure and Applied Mathematics*, John Wiley and Sons, pp.909-996, 1988.
- [13] I. Daubechies, "Biorthogonal bases of compactly supported wavelets," *Communications on Pure and Applied Mathematics*, John Wiley and Sons, pp.485-560, 1992.

- [14] T. Archarya, P. S. Tsai, "JPEG 2000 standard for image compression: Concepts, algorithms and VLSI architectures," Wiley Interscience, 2005.
- [15] W. Sweldens, "The lifting scheme: A new philosophy in biorthogonal wavelet constructions," in Proc. SPIE, vol.2569, pp.68-79, 1995.
- [16] M. D. Adams, F. Kossentine, "Reversible integer-to-integer wavelet transform of image compression: Performance evaluation and analysis," IEEE Trans. on Image Processing, vol.9, pp.1010-1024, 2000.
- [17] "XE Spartan-3E automotive FPGA family data sheet", Product Specification, Xilinx 2007, www.xilinx.com/support/documentation/data_sheets/ds635.pdf (Last accessed on 28th June 2008).
- [18] "FPGA design flow overview", http://toolbox.xilinx.com/docsan/xilinx7/help/iseguide/html/ise_fpga_design_flow_overview.htm (Last accessed on 28th June 2008).
- [19] S. Palnitkar, "Verilog HDL: A guide to digital design and synthesis", Prentice hall, Second Edition, 2003.
- [20] D. Perry, "VHDL programming by example," McGraw-hill, Fourth edition, 2002.
- [21] J. Bhasker, "VHDL primer", Pearson education, Third edition, 2003.
- [22] M. Wisdom and P. Lee, "An efficient implementation of a 2D DWT on FPGA", Int. Conf. on Field Programmable Logic and Applications (FPLA), pp. 222-227, 2007.
- [23] A. Savakis and R. Carbone, "Discrete wavelet transform core for image processing applications," Proceedings of the SPIE, vol. 5671, pp. 142-151, 2005.
- [24] M. Ferretti and D. Rizzo, "A parallel architecture for the 2-D discrete wavelet transform with integer lifting scheme," Journal of VLSI Signal Processing, vol. 28,no.4, pp. 165-185, 2001.
- [25] C. Xiong, J. Tian and J. Liu, "Efficient parallel architecture for lifting-based row-dimensional discrete wavelet transform," IEEE Int. Workshop VLSI Design and Video Tech., Suzhou, China, pp.75-78, 2005.

LIST OF PUBLICATIONS

1. K. Prashanth Kumar, R. C. Joshi and A. K. Saxena, "Improved parallel architecture for implementation of lifting based CDF (2,2) wavelet", IEEE Region 10 Colloquium and Third Int. Conf. on Industrial and Information Systems, IIT Kharagpur, Kharagpur, India, 2008 (Communicated).
2. K. Prashanth Kumar, R. C. Joshi and A. K. Saxena, "FPGA based efficient parallel architecture of lifting based CDF (2,2) for image compression", International Journal on Intelligent Electronic Systems, Sathyabama University, Chennai, India, 2008 (Communicated).