# Wavelets analysis (using lifting scheme) and its applications to image compression

**A Dissertation**

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

**MASTER OF ENGINEERING**
**(COMPUTER TECHNOLOGY & APPLICATIONS)**

**BY**

**Rajeev Srivastava**

**Under the guidance of**
**Prof. D. Roy. Choudhary**

**Department of Computer Engineering**
**Delhi College of Engineering, New Delhi-110042**
**(University of Delhi)**

**January-2005**

## CERTIFICATE

This is to certify that the Dissertation entitled **“Wavelets analysis (using lifting scheme) and its applications to image compression** “submitted by **RAJEEV SRIVASTAVA**, in the partial fulfillment for the award of the degree of **“MASTER OF ENGINEERING”** in **“Computer Technology and Applications”** of **Computer Engineering Department** of **Delhi College of Engineering**, **Delhi,** is a bonafide record of the work he has carried under my guidance and supervision**.**

**Prof. D. Roy. Choudhary**
**(Prof. & Head)**
**Department of Computer Engineering**
**Delhi College of Engineering,**
**(University of Delhi)**
**New Delhi-110042**

## Acknowledgement

I would like to express my sincere and profound gratitude to my supervisor **Prof. D. Roy Choudhary** for his invaluable guidance, continuous encouragement, helpful discussions and wholehearted support in every stage of this Thesis. It is my privilege and honor to have worked under his supervision. It is indeed difficult to put his contribution in few words.

I am also thankful to my teachers at DCE viz. Prof A. Dey, Dr. Goldie Gabrani, Dr. S.K. Saxena and Mrs. Rajni Jindal for their support and encouragement.

I would also like to thank my parents, wife and daughter (Saloni) for their encouragement, support and dedication, which has helped me in great sense to complete this course.

Last but not least I would like to thank my colleagues Dr. Sangeeta Sabharwal, Mr. Satish Chand, Ms. Anubha Gupta, for their all kind of support in completing this course.

**(RAJEEV SRIVASTAVA)**
**M.E. (Computer Technology and Applications)**
**College Roll #16/CTA/02**
**Delhi University Roll # 4001**

## ABSTRACT

In this thesis, analysis and synthesis of various Wavelet Transform [1,7,10] techniques with and without lifting scheme [15] has been studied and implemented. Further the implementation and analysis of performance of a "Discrete Wavelet Transform based embedded image codec" in MATLAB has been presented. This codec performs compression and decompression of still gray images in an efficient manner. The image codec consists of two major parts .The first one is **encoder**, which performs compression by applying Wavelet Transform [1,7,10] on image followed by quantization and encoding. Another part is **decoder**, which decompresses the image, and reconstructs it by de-quantization, decoding and Inverse wavelet transform. Main data loss occurs during quantization and encoding phase, hence for efficient quantization and encoding a **SPIHT** (Set Partitioning in Hierarchical Trees) **coder** [3] based on progressive transmission and embedded coding is implemented. The performance of the designed image codec is evaluated for different wavelets filters [6] at different bit rate. The coding gain of SPIHT coder in terms of distortion measures MSE and PSNR over different compression ratios i.e. bit rate is evaluated and an analysis is made. Further the effect on performance of the image codec is also analyzed for different wavelet filters at different levels of decomposition.

**Keywords: Wavelets, DWT, lifting scheme, filters, SPIHT, bit rate, PSNR, MSE, sub-band coding.**

**Table of Contents:**

# Chapter 1

# Introduction

## 1.1 Background:

Digital Signal Processing and in particular Image Processing has been studied for many years. However only the recent advancements in computing technology have made it possible to use Image Processing in day-to-day applications. The increasing use of images raises new challenges and user expects the images to be transmitted in a minimum of time and to take up as little storage as possible. These requirements call for efficient Image Compression algorithms. The user wants this compression and decompression process to be very efficient and accurate. However, the degree of compression and the amount of data-loss are of much concern which indicates the need for a method combining a high compression ratio with a low (even zero) amount of data loss. For this purpose Wavelets can be used. Wavelets are mathematical functions that satisfy certain properties and can be used to transform one function representation in to another. By choosing an appropriate wavelet transform for a particular problem can provide a high compression ratio with no data loss . Wavelet Transform[1,7,10] have Applications to image compression as well as a variety of other applications.

Methods for digital image compression have been the subject of much study over the past decade. Advances in Wavelet Transform [1,7,10]s and quantization methods have produced algorithms capable of surpassing the existing image compression standards like the Joint Photographic Experts Group (JPEG) algorithm. For best performance in image compression, wavelet transforms require filters that combine a number of desirable properties, such as orthogonality and symmetry. Since, practical data sequences or images normally contain a substantial amount of redundancy in form of smoothness of the signal or in other words correlation between the neighboring signal values. A data sequence, which embeds redundancy, can be presented more compactly if the redundancy is removed by means of a suitable transform. An appropriate transform should match the statistical characteristic of the data. Applying the transform on the data results in less correlated transform coefficients, which can be encoded with fewer bits. A popular transform that has been used for years for compression of digital still images and image sequences, is the Discrete Cosine Transform (DCT). The DCT transform uses cosine functions of different frequencies for analysis and decorrelation of data. In the case of still images, after transforming the image from spatial domain to transform domain, the transform domain coefficients are quantized (a lossy step) and subsequently entropy encoded. Another transform that has received a great amount of attention in the last decade, is the Wavelet Transform. Wavelets are 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 supports (the length of the dilated mother wavelet). The larger support extracts information about the frequency behavior of data. Varying the dilation and translation of the mother wavelet, therefore, produces a customizable time/frequency analysis of the input signal. The Wavelet Transform[1,7,10] uses overlapping functions of variable size for analysis. The overlapping nature of the transform alleviates the blocking artifacts, as each input sample contributes to several samples of the output. The variable size of the basis functions, in addition, leads to superior energy compaction and good perceptual quality of the decompressed image. The latter characteristic, besides, means a more graceful degradation of the decoded signal compared with the DCT, as the encoding bit budget decreases. The DCT based JPEG algorithm yields good results for compression ratios till 10:1. As the compression ratio increases, coarse quantization of the coefficients causes blocking effects in the decompressed image. When compression ratio reaches 24:1, the decreased bit budget only allows the DC coefficients, which are the average of the pixels of an 8x8 block, to be encoded. Consequently, the input image is approximated by a series of 8x8 blocks of local averages, which is visually very annoying. For Wavelet Transform followed by Embedded Zero Tree encoding algorithm e.g. Set Partitioning in Hierarchical Trees (SPIHT), in contrast, much higher compressions ratios has been achieved, while still yielding a reconstructed image with an acceptable quality.

## 1.2 Objective:

The purpose of this thesis is to investigate and study the analysis and design of various Discrete Wavelet Transforms with and without the use of lifting techniques and its application in the design and MATLAB implementation of an efficient embedded Image Codec based on SPIHT.The codec, presented in this thesis, performs compression and decompression of still images in an efficient manner. The image codec consists of two major parts –one **encoder** that performs compression by applying Wavelet Transform [1,7,10] on image (with and without lifting scheme) followed by quantization and encoding, and another part is **decoder**, which decompresses the image and reconstruct it by de-quantization, Inverse. Since, main data loss occurs during quantization phase and encoding phase, hence for efficient quantization and encoding a **SPIHT** (Set Partitioning in Hierarchical Trees) **coder** for progressive transmission is studied and implemented. The performance of the designed image codec is evaluated and analyzed for different wavelets families (e.g. Haar, Daubechies, Biortogonal, Symlets) for different levels of decomposition at different compression ratios. Further, the performance of the image codec is also analyzed using lifting techniques for bi-orthogonal CDF(9,7) [9]filter.

## 1.3 Organization of thesis

The thesis is organized as follows:

- Chapter 2 performs a study on theoretical background, briefly introducing the image coding terminologies, data compression, various transform techniques, Sub-band coding, performance metrics etc..
- Chapter 3 continues with the introduction of wavelet transforms, its properties and advantages, Analysis and synthesis of various Discrete Wavelet Transform Techniques
- Chapter 4 explains the concepts of wavelets analysis using lifting scheme

- Chapter 5 explains the SPIHT method for quantization and coding of wavelet coefficients.
- Chapter 6 explains the Design and Implementation of Discrete Wavelet Transform based Embedded Image Codec in MATLAB
- Chapter 7: Results, Performance Evaluation and Conclusion

This chapter explains Performance evaluation and presents the results of the series of simulations performed on different images, different filters, different decomposition levels at different compression ratios (bit rate).

- Future Work
- References
- Appendices

# Chapter 2

# Theoretical Background

This chapter provides theoretical background information, briefly introducing the image coding terminologies, data compression, various transform techniques, performance measurement metrics and Sub-band coding.

## 2.1 Image coding terminologies

### 2.1.1 Data compression (Lossy and Loss-less coding)

Digital image data compression is the art or science of reducing the number of bits needed to represent a given image. The purpose of compression is to facilitate the storage and transmission of data. The reduced representation is typically achieved using a sequence of transformations of data, which are either exactly or approximately invertible. In image coding and data compression there are two main areas, loss-less compression and lossy compression. In loss-less compression redundancy in the data representation is targeted. The key object in loss-less coding is an efficient *exact* representation of the data. Loss-less coding is also referred to as entropy coding.

In lossy coding one accepts that the data is distorted, that is, the data reconstructed from the code is not exactly the data that was coded. It is then possible to compress data even more than in loss less coding. The price to pay is of course that information is lost in the coding process. The information loss in lossy coding comes from quantization of the data. Quantization can be described as the process of sorting the data into different bins and representing each bin with a value. The value selected to represent a bin is called the reconstruction value. Every item in a bin has the same reconstruction value, which leads to information loss (unless the quntization is so fine that every item gets its own bin).

**A) Loss-less coding techniques:**

Loss-less coding guaranties that the decompressed image is absolutely identical to the image before compression. This is an important requirement for some application domains, e.g. medial imaging, where not only high quality is in demand, but unaltered archiving is a legal requirement. Loss-less techniques can also used for the compression of other data types where loss of information is not acceptable, e.g. text documents and program executables. Some compression methods can be made more effective by adding a 1D or 2D delta coding to the process of compression. These deltas make more effectively use of run length encoding, have (statistically) higher maxima in code tables (leading to better results in Huffman and general entropy coding), and build greater equal value areas usable for area coding. Some of these methods can easily be modified to be lossy. Lossy element fits perfectly into 1D/2D run length search. Also, logarithmic quantisation may be inserted to provide better or more effective results.

**a) Run length encoding:** Run length encoding is a very simple method for compression of sequential data. It takes advantage of the fact that, in many data streams, consecutive single tokens are often identical. Run length encoding checks the stream for this fact and inserts a special token each time a chain of more than two equal input tokens are found. This special input advises the decoder to insert the token *n* times into his output stream.

**b) Huffman encoding:** This algorithm, developed by D.A. Huffman, is based on the fact that in an input stream certain tokens occur more often than others. Based on this knowledge, the algorithm builds up a weighted binary tree according to their rate of occurrence. Each element of this tree is assigned a new code word, whereat the length of the code word is determined by its position in the tree. Therefore, the token, which is most frequent and becomes the root of the tree, is assigned the shortest code. Each less common element is assigned a longer code word. The least frequent element is assigned a code word, which may have become twice as long as the input token. The compression ratio achieved by Huffman encoding uncorrelated data becomes something like 1:2. On slightly correlated data, as on images, the compression rate may become much higher, the absolute maximum being defined by the size of a single input token and the size of the shortest possible output token.

**c) LZW entropy coding:** The typical implementation of an entropy coder follows J. Ziv/A. Lempel's approach. Nowadays, there is a wide range of so-called modified Lempel/Ziv coding. These algorithms all have a common way of working. The coder and the decoder both build up an equivalent dictionary of meta -symbols, each of which represents a whole sequence of input tokens. If a sequence is repeated after a symbol was found for it, then only the symbol becomes part of the coded data and the sequence of tokens referenced by the symbol becomes part of the decoded data later. As the dictionary is build up based on the data, it is not necessary to put it into the coded data, as it is with the tables in a Huffman coder. This method becomes very efficient even on virtually random data. The average compression on text and program data is about 1:2, the ratio on image data comes up to 1:8 on the average GIF image. Here again, a high level of input noise degrades the efficiency significantly. Entropy coders are a little tricky to implement, as there are usually a few tables, all growing while the algorithm runs.

**B) Lossy Coding Techniques:**
In most of applications we have no need in the exact restoration of stored image. This fact can help to make the storage more effective, and this way we get to lossy compression methods**.**

***Components of Lossy Image Coding techniques:***

- **Image Modeling (Transformation)** which defines such things as the transformation to be applied to the image
- **Parameter Quantisation** whereby the data generated by the transformation is quantised to reduce the amount of information
- **Encoding,** where a code is generated by associating appropriate code words to the raw data produced by the quantiser. Each of these operations is in some part responsible for the compression.

**Image modeling** is aimed at the exploitation of statistical characteristics of the image (i.e. high correlation, redundancy). Some examples are transform-coding methods, in which the data is represented in a different domain (for example, frequency in the case of the Fourier Transform [FT], the Discrete Cosine Transform [DCT], the Kahrunen-Loewe Transform [KLT], and so on), where a reduced number of coefficients contain most of the original information. In many cases this first phase does not result in any loss of information. Another transform that will be used in this thesis is Discrete Wavelet Transform.

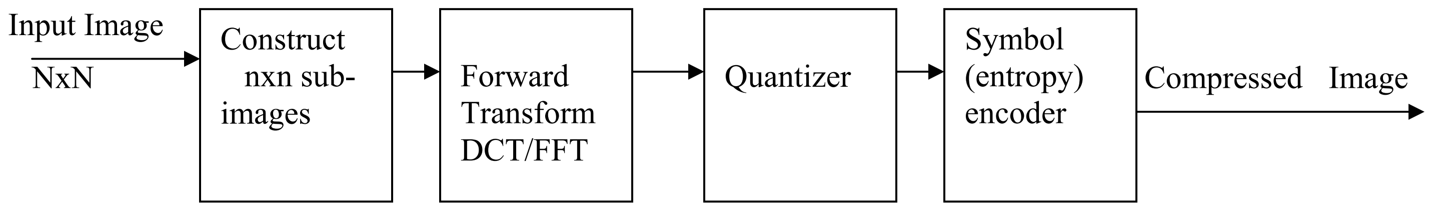
**Quantisation** is used to reduce the amount of data used to represent the information within the new domain. Quantisation is in most cases not a reversible operation: therefore, it belongs to the so called ‘lossy’ methods.

**Encoding** is usually error free. It optimizes the representation of the information (helping, sometimes, to further reduce the bit rate), and may introduce some error detection codes.

### 2.1.2.Transform Coding Models

## i) A General Transform Coding System

**A) Encoder**



**B) Decoder**

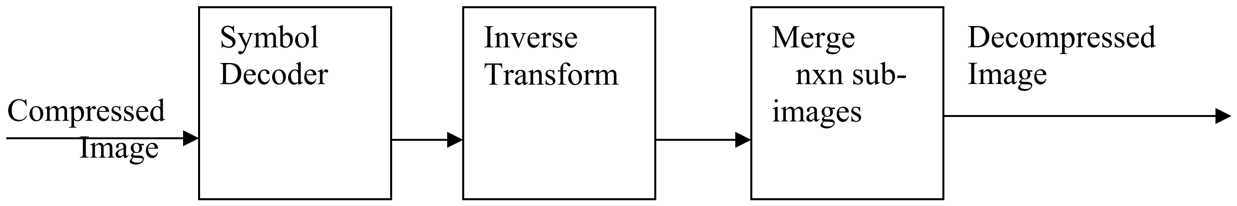


**Figure 2.1**

A general transform coding scheme involves subdividing an *N*x*N* image into smaller *n*x*n* blocks and performing a *unitary transform* on each sub image. A unitary transform is a reversible linear transform whose kernel describes a set of complete, orthonormal discrete basic functions. The goal of the transform is to de-correlate the original signal, and this de-correlation generally results in the signal energy being redistributed among only a small set of transform coefficients. In this way, many coefficients may be discarded after quantisation and prior to encoding. Also, visually loss-less compression can often be achieved by incorporating the HVS (Human Visual System) contrast sensitivity function in the quantisation of the coefficients. Transform coding can be generalized into following stages:

- Image Subdivision

- Image Transformation
- Coefficient Quantisation

Quantisation can be performed in several ways. Most classical approaches use ‘zonal coding’, consisting in the scalar quantisation of the coefficients belonging to a predefined area (with a fixed bit allocation), and ‘threshold coding’, consisting in the choice of the coefficients of each block characterized by an absolute value exceeding a predefined threshold. Another possibility that leads to higher compression factors is to apply a vector quantisation scheme to the transformed coefficients. The same type of encoding is used for each coding method. In most cases a classical Huffman code can be used successfully. The JPEG and MPEG standards are examples of standards based on transform coding

**ii) Block Diagram of a Wavelet Transform based Embedded Image Codec**

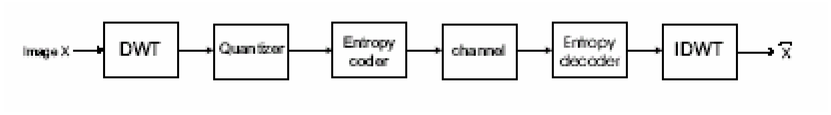


**Figure 2.2(a)**

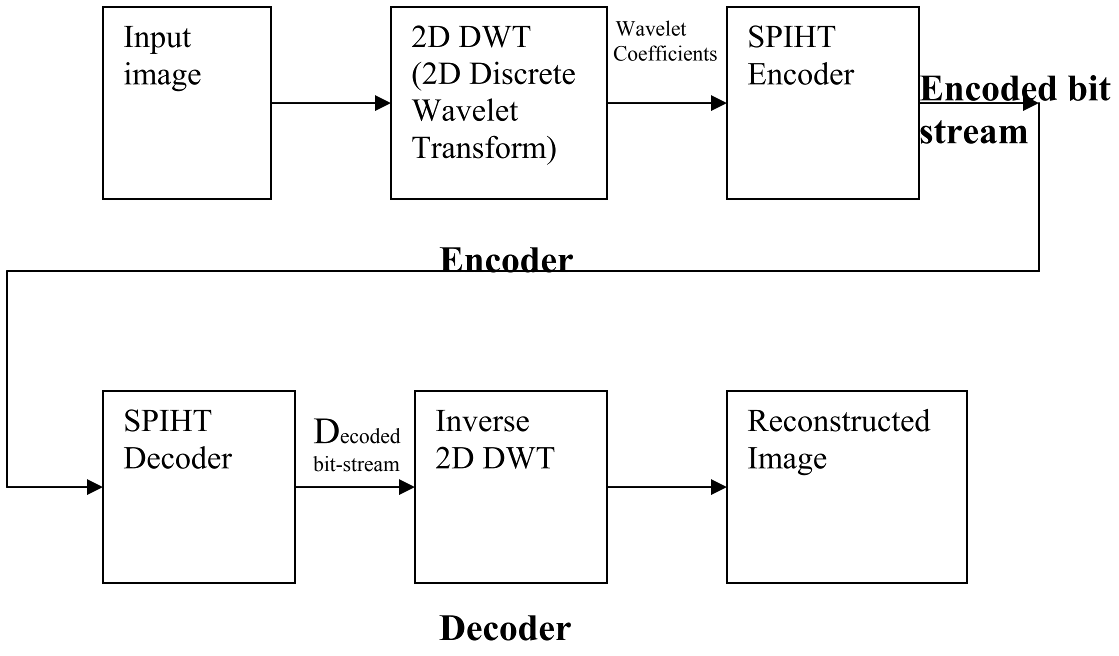


**Figure 2.2 (b)**

The primary goal of any image compression technique is to reduce the number of bits needed to represent the image with little perceptible distortion. Sub-band coding [1,7,12] using wavelets is one of the best performing techniques among different transform based image compression techniques. Figure 2.2 (a) shows the block diagram of a wavelet based image compression system. The three blocks (DWT, quantizer and entropy coder) compress the image data whereas the last two blocks (entropy decoder, inverse discrete wavelet transform (IDWT)) reconstruct the image from the compressed data. The DWT performs an octave frequency sub-band decomposition of the image .In its sub-band representation, an image is more compactly represented since most of its energy is concentrated in relatively few DWT coefficients. The quantizer then performs quantization by representing the transform coefficients with a limited number of bits. Quantization represents lossy compression. Some image information is irretrievably lost. A quantizer in a DWT-based coder exploits the spatial correlation in a wavelet-based, hierarchical scale-space decomposition. The entropy coder follows the quantization stage in a wavelet based image compression system. Entropy coding is lossless; it removes the redundancy from the compressed bit-stream. However, the typical performance improvement of 0.4-0.6 dB achieved by entropy coding is accompanied by higher computational complexity. The channel is the stored or transmitted compressed bit stream. We consider the channel to be noiseless and the received DWT coefficients are free from errors. The synthesis stage reconstructs the image from the compressed data. The entropy decoder and IDWT invert the operations performed by the entropy encoder and DWT, respectively. Figure 2.2(b) shows the block diagram of a wavelet based image compression system used in this thesis. In this thesis entropy coder ( encoder and decoder) is not used but in place of quantizer and entropy coder an efficient and fast SPIHT [3] ( Set Partitioning in Hierarchical Trees) coder based on progressive transmission which uses zero-tree coding approach is used which perform the task of quantization and coding the wavelet coefficients.

### 2.1.3 Various Transform Techniques:

The choice of a particular transform in a given application depends on the amount of reconstruction error that can be tolerated and the computational resources available. Compression is achieved during the quantization of the transformed coefficients not during the transformation step. Image modeling or transformation is aimed at the exploitation of statistical characteristics of the image (i.e. high correlation, redundancy).

Some transform techniques are:

**Fourier Transform (FFT, DFT, WFT)**
**Discrete Cosine Transform (DCT)**
**Walsh-Hadamand Transform (WHT)**
**Wavelet Transform (CWT, DWT, FWT)**

For Fourier Transform and DCT basis images are fixed i.e. they are input independent and sinusoidal (cosines and sines) in nature. Provides frequency view i.e. provide frequency information and temporal information is lost in transformation process.

WHT is non-sinusoidal in nature and easy to implement.(Frequency domain)

Wavelet Transforms provides time-frequency view i.e. provides both frequency as well as temporal (localization) information. Wavelets give time-scale viewpoint and exhibits multiresolution characteristics.Fourier is good for periodic or stationary signals but Wavelet is good for transients i.e. for non-stationary data. Localization property allows wavelets to give efficient representation of transients.

#### A) Fourier Transform

Since the Fourier Transform is widely used in analyzing and interpreting signals and images, I will first have a survey on it prior to going further to the Wavelet Transform. The tool which converts a spatial (real space) description of an image into one in terms of its frequency components is called the **Fourier transform**. Through Fourier Transform, it is possible to compose a signal by superposing a series of sine and cosine functions.

These sine and cosine functions are known as basis functions (Figure 2.2.1) and are mutually orthogonal. The transform decomposes the signal into the basis 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 basis function with its corresponding coefficient and adding them up together, i.e. a linear superposition of the basis functions.

**Fourier Analysis and Orthogonality**

Fourier analysis is one of the most widely used tools in spectral analysis. The basis for this analysis is the Fourier Integral, which computes the amplitude spectral density F(ω) of a time-domain signal f(t).

$$F(\omega) = \int_{-\infty}^{\infty} f(t)\exp(-j\omega t)\,dt$$

F(ω) is actually complex, so one obtains the amplitude spectral density A(f) and phase spectral density φ(f) as a function of frequency. Another way of looking at the Fourier transform is that it answers the question: what continuous distribution of sine waves A (f)cos(jωt+ φ(f)) when added together on a continuous basis best represents the original time signal? We call these distributions the amplitude and phase spectral densities (or spectra). Complex exponentials are popular basis functions because in many engineering and science problems, the relevant signals are sinusoidal in nature. It is noticed that when signals are not sinusoidal in nature, a wide spectrum of the basis function is needed in order to represent the time signal accurately. An important property of any family of basis functions ψ(t) is that it is orthogonal.

The basis functions in the Fourier Transform are **ψ(t) = exp(+jωt)**, so the Fourier Transform could be more generally written as

$$F(\omega) = \int_{-\infty}^{\infty} f(t)\psi^{*}(t)\,dt$$

where * denotes complex conjugate. The test for orthogonality is done as follows

$$\int_{-\infty}^{\infty} \psi_m(t)\psi_n^{*}(t)\,dt = \begin{Bmatrix} k & m=n \\ 0 & m\neq n \end{Bmatrix}$$

For complex exponentials, because they are infinite in duration, one end up with k=∞, when m=n so it is necessary to define the orthogonality test in a different way :

$$\lim_{T\to\infty}\frac{1}{T}\int_{-T/2}^{T/2} \psi_m(t)\psi_n^{*}(t)\,dt = \begin{Bmatrix} k & m=n \\ 0 & m\neq n \end{Bmatrix}$$

For **complex exponential**, this becomes:

$$\lim_{T\to\infty}\frac{1}{T}\int_{-T/2}^{T/2} \exp(j\omega_m t)\exp(-j\omega_n t)\,dt = \lim_{T\to\infty}\frac{\sin(T[\omega_m-\omega_n]/2)}{T[\omega_m-\omega_n]/2} = \begin{Bmatrix} 1 & m=n \\ 0 & m\neq n \end{Bmatrix}$$

When the constant k=1,the function is said to be orthonormal.

Various types of signals can be analyzed with the Fourier Transform. If f(t) is periodic, then the amplitude spectral density clusters at discrete frequencies that are harmonics (integer multiples) of the fundamental frequency. One need to invoke Dirac Delta functions if the Fourier Transform is used– otherwise Fourier series coefficients can be computed and same result can be obtained. If f (t) is deterministic and discrete, the discrete time Fourier Transform (DTFT) may be used to generate a periodic frequency response. If f(t) is assumed to be both periodic and discrete, then the discrete Fourier Transform (DFT), or its fast numeric equivalent the FFT may be applied to compute the spectrum. If f(t) is random, then in general one will have a difficult time of computing the Fourier Integral of the random 'data'. Hence treat the input as data and use an FFT, but the result of doing so is a random spectrum. This single random spectrum can give an idea of the frequency response, but in many instances it can be misleading. A better approach is to take the average of the random spectra. This leads to the formulation of power spectral density, which is an average over the FFT magnitude spectrum squared.
In certain signals, both random and deterministic, we are interested in the spectrum as a function of time in the signal. This suggests finding the spectrum over a limited time bin, moving the bin (sometimes with overlap, sometimes without), re-computing the spectrum, and so on. This method is known as the short-time Fourier Transform (STFT), or the Gabor Transform.

**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 O($n^2$ ),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(nlogn) by taking advantage of self similarity properties of the DFT.

**If the input is a non-periodic signal**, the superposition of the periodic basis functions does not accurately represent the signal.

One way to overcome this problem is to extend the signal at both ends to make it periodic.

Another solution is to use **Windowed Fourier Transform (WFT)**. In this method the signal is multiplied with a window function ( Figure 2.2.2) prior to applying the Fourier transform. The window function localizes the signal in time by putting the emphasis in the middle of the window and attenuating the signal to zero towards both ends.

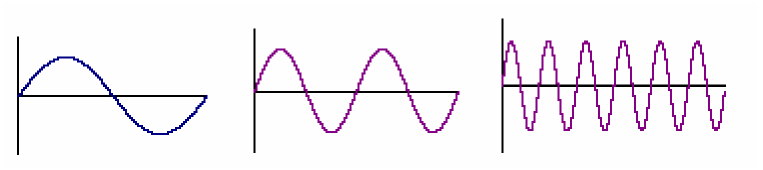
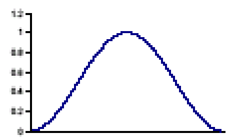
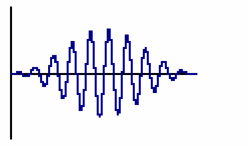


**Figure2.2.1 A Set of Fourier basis functions**





**A window function** **A windowed Signal**
**Figure2.2.2**

**B) Discrete Cosine Transform (DCT)**
The discrete cosine transform (DCT) helps separate the image into parts (or spectral sub-bands) of differing importance (with respect to the image's visual quality). The DCT is similar to the discrete Fourier transform: it transforms a signal or image from the spatial domain to the frequency domain.
With an input image, A, the coefficients for the output "image," B, are:

$$\boxed{B(k_1,k_2)=\sum_{i=0}^{N_1-1}\sum_{J=0}^{N_2-1}4.A(i,j).\cos\left[\frac{\pi.k_1}{2.N_1}.(2.i+1)\right].\cos\left[\frac{\pi.k_2}{2.N_2}.(2.j+1)\right]}$$

The input image is $N_2$ pixels wide by $N_1$ pixels high; A(i,j) is the intensity of the pixel in row i and column j; $B(k_1,k_2)$ is the DCT coefficient in row k1 and column k2 of the DCT matrix. All DCT multiplications are real. This lowers the number of required multiplications, as compared to the discrete Fourier transform. The DCT input is an 8 by 8 array of integers. This array contains each pixel's gray scale level; 8 bit pixels have levels from 0 to 255. The output array of DCT coefficients contains integers; these can range from -1024 to 1023. For most images, much of the signal energy lies at low frequencies; these appear in the upper left corner of the DCT. The lower right values represent higher frequencies, and are often small - small enough to be neglected with little visible distortion. It is computationally easier to implement and more efficient to regard the DCT as a set of **basis functions** which given a known input array size (8 x 8) can be pre-computed and stored. This involves simply computing values for a convolution mask (8 x8 window) that get applied (sum values x pixel the window overlap with image apply window across all rows/columns of image). The values as simply calculated from the DCT formula. The 64 (8 x 8) DCT basis functions are there. Most software implementations use fixed point arithmetic. Some fast implementations approximate coefficients so all multiplies are shifts and adds.

**C) DCT Vs Fourier:**

- DCT is similar to the Fast Fourier Transform (FFT), but can approximate lines well with fewer coefficients .
- DCT (Discrete Cosine Transform) is actually a ***cut-down*** version of the FFT i.e. it is only the **real** part of FFT .
- DCT Computationally simpler than FFT and much effective for Multimedia Compression.

- DCT is associated with very less MSE value in comparison to others.
- DCT has best information packing ability.
- DCT minimizes the block like appearance (blocking articrafts),that results when the boundaries between the sub-images become visible. But DFT gives rise to boundary discontinuities.

**D) Wavelet Transform:**

Wavelet means ‘small wave’. So wavelet analysis is about analyzing signal with short duration finite energy functions. They transform the signal under investigation in to another representation which presents the signal in more useful form. This transformation of the signal is called Wavelet Transform [1,7,10] i.e. Wavelet Transforms are based on small waves, called wavelets, of varying frequency and limited duration. Unlike the Fourier transform, we have a variety of wavelets that are used for signal analysis. Choice of a particular wavelet depends on the type of application in hand. Wavelet Transforms provides time-frequency view i.e. provides both frequency as well as temporal (localization) information and exhibits multiresolution characteristics. Fourier is good for periodic or stationary signals and Wavelet is good for transients. Localization property allows wavelets to give efficient representation of transients. In Wavelet transforms a signal can be converted and manipulated while keeping resolution across the entire signal and still based in time i.e. Wavelets have special ability to examine signals simultaneously in both time and frequency. Wavelets are mathematical functions that satisfy certain criteria, like a zero mean, and are used for analyzing and representing signals or other functions. A set of dilations and Translations of a chosen mother wavelet is used for the spatial/frequency analysis of an input signal. The Wavelet Transform uses overlapping functions of variable size for analysis. The overlapping nature of the transform alleviates the blocking artifacts, as each input sample contributes to several samples of the output. The variable size of the basis functions, in addition, leads to superior energy compaction and good perceptual quality of the decompressed image. Wavelets Transform is based on the concept of sub-band coding [1,7,12].

The current applications of wavelet include statistical signal processing, Image processing, climate analysis, financial time series analysis, heart monitoring, seismic signal de- noising, de-noising of astronomical images, audio and video compression, compression of medical image stacks, finger print analysis, fast solution of partial differential equations, computer graphics and so on.

The detailed discussions about wavelet Analysis and Synthesis is presented in next chapter.

**E) Wavelets Vs Fourier and DCT:**

- Fourier and DCT transforms converts a signal from time Vs amplitude to frequency Vs amplitude i.e. provides only frequency information and temporal information is lost during transformation process. But Wavelet transforms provides both frequency as well as temporal ( localization ) information.
- In Fourier and DCT basis functions are sinusoids (sine and cosine) and cosines respectively but in Wavelet Transform basis functions are various wavelets.

- Since Wavelet Transforms are both computationally efficient and inherently local (i.e. there basis functions are limited in duration),subdivision of original image before applying transformation is not required as required in DCT and others.
- The removal of subdivision step in Wavelet Transform eliminates the blocking articraft but FFT suffers from it. This property also characterizes DCT-based approximation, at higher compression ratios.
- Wavelets provide unconditional basis for large signal class. Wavelet coefficients drops sharply hence good for compression, de-noising, detection and recognition.
- Fourier is good for periodic or stationary signals. Wavelet is good for transients. Localization property allows wavelets to give efficient representation of transients.
- Wavelets have local description and separation of signal characteristics. Fourier puts localization information in the phase in a complicated way. STFT cannot give localization and orthogonality.
- Wavelets can be adjusted or adapted to application.
- Computation of wavelet coefficients is well suited to computer. No derivatives of integrals needed as required in Fourier and DCT and hence turn out to be a digital filter bank.

### 2.1.4 Performance measurement metrics:

#### *MSE (Mean Squared Error)*

If lossy compression is used it is convenient to be able to quantify the difference, or distortion, between the original signal (image) and the reconstructed image. A popular choice for measuring distortion is the Mean Squared Error, or MSE. In the MSE measurement the total squared difference between the original signal and the reconstructed one is averaged over the entire signal. It is calculated as:

$$\mathbf{MSE=(1/M*N)\Sigma_{I=0:M-1}\Sigma_{J=0:N-1}\,[OI\ (I,\ J)-RI\ (I,\ J)]^{2}}$$

Where MxN the size of image with M rows and N columns.
OI (I, J)=coefficients of original image
RI (I, J)=coefficients of Reconstructed Image

**RMSE (Root Mean Squared Error):**It is another method for distortion measurement.

$$\mathbf{RMSE=SQRT\ (MSE)}$$

**PSNR (Signal to Noise Ratio):** The PSNR relates the MSE to the maximum amplitude of the original signal, this makes the measurement independent of the range of the data. The PSNR is usually measured in decibel as:

$$\mathbf{PSNR=10*log10\ (Q*Q/MSE)\ [dB]}$$

**Or,**

$$\mathbf{PSNR=20*log10\ (Q/RMSE)\ \ [dB]}$$

Where $Q=2^n-1$=Maximum possible peak-to-peak amplitude for an 8-bit Gray Scale Image or 24-bit color RGB Image

For Gray Image: n=8 bits/pixel, Q=255
For RGB Image: n=24 bits/pixel, $Q=2^{24}-1$

**Bit Rate**: No of bits used to represent per pixel in the image
For Gray Image: Actual bpp (bits per pixel)=8 bits
For RGB Image: Actual bpp (bits per pixel =24 bits/pixel
This bit rate may be varied from some minimum value to the mentioned actual bit rate to achieve different compression ratios.

## Compression ratio (CR):

**CR=8/Bit Rate** (for Gray Image)
**CR=24/Bit Rate** (for RGB Color Image)

***SNR (Signal to Noise Ratio):***

SNR=10*log10[(Sum of energies of all pixels of Original image)/(Sum of energies of Noise signal)]

$$\mathbf{SNR=10*log10\ [\{\Sigma_{I=0:M-1}\Sigma_{J=0:N-1}\ [OI\ (I,\ J)]^2\}/\{\Sigma_{I=0:M-1}\Sigma_{J=0:N-1}\ [OI(I,J)-RI(I,J)]^2\}][dB]}$$

**2.1.5 The signal energy, orthogonality and orthonormality**

In this thesis images will be viewed as special cases of two-dimensional signals where pixel values represent signal samples. If there is a referral to "the signal" it should be clear from the context that it is an image.

The **signal energy (E)** is defined as:

$$\mathbf{E=[\Sigma_{I=0:M-1}\ [X\ (I)]^2\ ]} \qquad \text{(1-D Signal)}$$
$$\mathbf{E=[\Sigma_{I=0:M-1}\Sigma_{J=0:N-1}\ [X\ (I,\ J)]^2\ ]}\ \text{(2D Signal)}$$

**Orthogonality** between signals is defined as-

$$\mathbf{X \perp Y= \Sigma\ (X_iY_i\ )=0\ (1D)}$$

**Or in function space ,** $\mathbf{{}_0\int^T X(t).Y(t)\ dt=0}$

**Orthonormality** of two signals X and Y is defined by following three conditions as follows:

$$\mathbf{\Sigma\ (X.Y)=0\ and\ \Sigma\ (X.X)=0,\ \Sigma\ (Y.Y)=0 \quad (1D)}$$

## 2.2 Multiresolution Analysis:

Multiresolution theory incorporates and unifies techniques from a variety of disciplines, including sub-band coding [1,7,12] from signal processing, quadrature mirror filtering from digital speech recognition, and pyramidical image processing. It is concerned with the representations and analysis of signals e.g. images at more than one resolution. The advantage of this approach is that the features that might go undetected at one resolution may be easy to spot at another. A function or signal can be viewed as composed of a smooth background and fluctuations or details on top of it. The distinction between the smooth part and the details is determined by the resolution, that is, by the scale below which the details of a signal cannot be discerned. At a given resolution, a signal is approximated by ignoring all fluctuations below that scale. We can imagine progressively increasing the resolution; at each stage of the increase in resolution finer details are added to the coarser description, providing a successively better approximation to the signal. Eventually when the resolution goes to infinity, we recover the exact signal. The above intuitive description can be made more precise as follows. We label the resolution level by an integer $j$. The scale associated with the level $j=0$ is set to, say, unity and that with the level $j$ is $1/2^j$. Let us consider a function $f(t)$. At resolution level $j$ it is approximated by $f_j(t)$. At the next level of resolution $j+1$, the details at that level denoted by $d_j(t)$ are included and we have the approximation to $f(t)$ at the new resolution level, $f_{j+1}(t) = f_j(t) + d_j(t)$. The original function $f(t)$ is recovered when we let the resolution go to infinity

$$f(t) = f_j(t) + \sum_{k=j}^{\infty} d_k(t).$$

The word multiresolution refers to the simultaneous presence of different resolutions. The above equation represents one way of decomposing the function $f(t)$ into a smooth part plus details. Similarly, we can view the space of functions that are square integrable, $L^2(R)$, as composed of a sequence of subspaces $\{W_k\}$ and $V_j$, such that the approximation of $f(t)$ at resolution $j$, $f_j(t)$, is in $V_j$ and the details $d_k(t)$ are in $W_k$. This brings us to the subject of this section, multiresolution analysis.

**2.2.1 Motivation for multiresolution analysis in Image processing:** When we look at images (mathematically, images are 2-D arrays of intensity values with locally varying statistics that result from different combinations of abrupt features like edges and contrasting homogeneous regions), generally we see connected regions of similar texture and gray level that combine to form the objects. If the objects are small in size or low in contrast, they are normally examined at high resolutions; if they are large in size or high in contrast, a coarse view that all is required. If both small and large objects are present simultaneously, it can be advantageous to study them at several resolutions. In this thesis ,wavelet-based transformations are examined from a multiresolution point of view that uses sub-band coding approach.

**2.2.2 Sub-band coding approach to multiresolution analysis:** This is a very good approach for wavelet based multiresolution analysis. This concept is used in this thesis for the design of an efficient image codec.

### A.1 Objective of sub-band coding

Sub-band coding [1,7,12] is a coding strategy that tries to isolate different characteristics of a signal in a way that collects the signal energy into few components. This is referred to as energy compaction. Energy compaction is desirable because it is easier to efficiently code these components than the signal itself.

### A.2 Coding scheme

The sub-band coding [1,7,12] scheme tries to achieve energy compaction by filtering a signal with filters of different characteristics. By choosing two filters that are orthogonal to each other and decimating the output of these filters a new two-component representation is achieved (Figure A2.1). In this representation, most of the signal energy is located in either **a** or **d**.

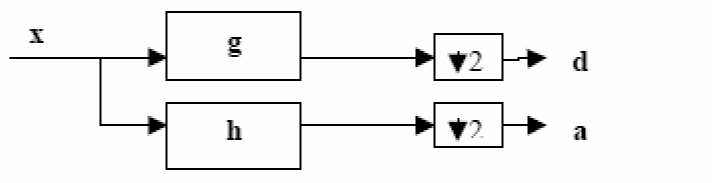


**Figure B2.1:** Splitting of the signal x into two parts.

The filters **h** and **g** are usually low-pass and high-pass filters. The two components **a** and **d** will then be a low-pass and a high-pass version of the signal **x**. Images have a typical low-pass character, hence we would expect **a** to contain most of the energy if **x** is an image. Besides trying to achieve energy compaction the filters **h** and **g** should be chosen so that perfect reconstruction of **x** from **a** and **d** is possible. How to choose **h** and **g** will be described later. In Figure A2.1 a two-component representation of **x** is achieved. It might be desirable to divide the signal into more components. This can be done by using several filters with different characteristics. A more common choice however is to cascade the structure in Figure A2.1.

**Major strategies for cascading the filters**.

- The hierarchical structure
- The flat structure.

In the **hierarchical structure** the output from the low-pass filter is treated as the input to a new filter pair as shown in Figure A2.2. In the **flat structure** both the low-pass and the high-pass outputs are inputs to a filter pair, this structure is shown in Figure A2.3. In both figures the corresponding splitting of the frequency axis is also shown.

The process of dividing the signal into components is referred as decomposition or transform.

X

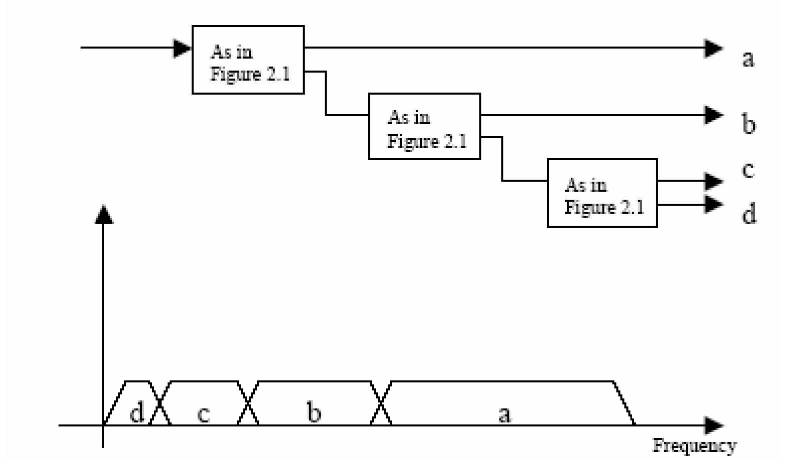


**Figure A2.2:** The Hierarchical Filter Structure

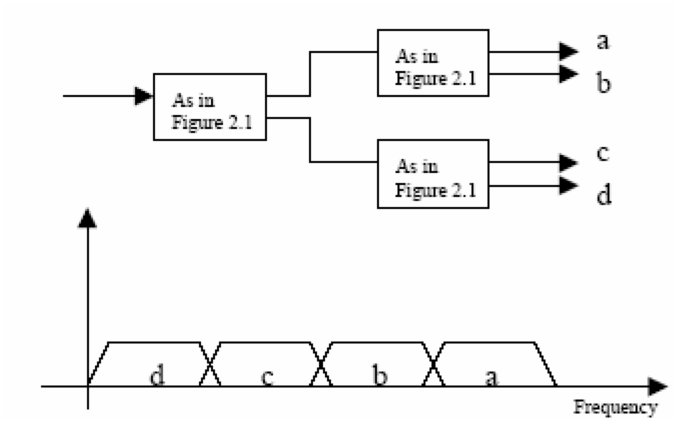


**Figure A2.3:** Flat Filter Structure

### A.3: 2D Transform:

To be able to use sub-band coding [1,7,12] for images the scheme above has to be adapted to two-dimensional signals. The extension of the sub-band coding scheme to higher dimension is straightforward. Apply the filters repeatedly to successive dimensions. For an **N**x**N** image we first compute **N** one-dimensional transforms corresponding to transforming each row of the image as an individual one-dimensional signal. This result in 2 **N**x**M** sub-images, one corresponding to the low-pass filtered rows and one corresponding to the high-pass filtered rows. Each of these sub-images are then filtered along the columns splitting the data into 4 **M**x**M** sub-images (low-pass row low-pass column, low-pass row high-pass column, high-pass row low-pass column, high-pass row high-pass column). This completes one stage of the decomposition of an image. The process is shown in Figure A3.1

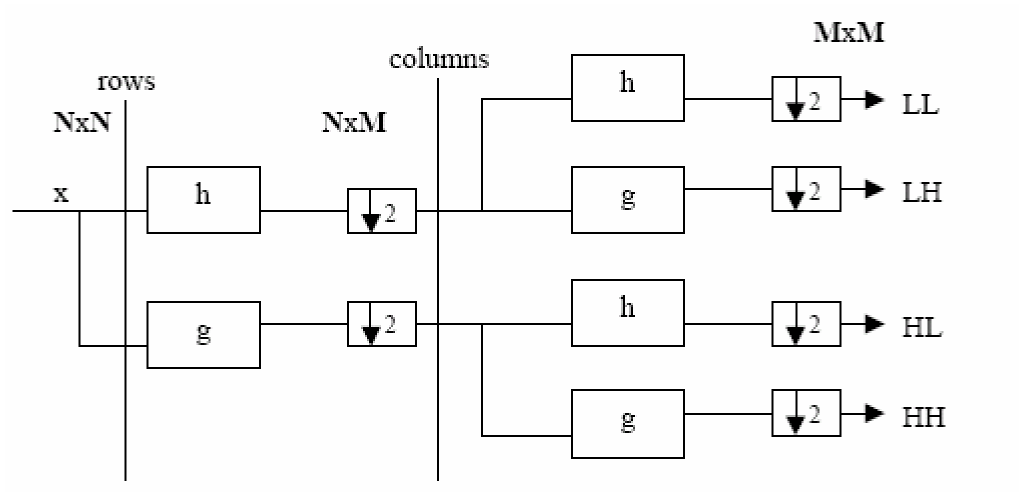


**Figure A3.1:**One stage of a 2D decomposition or Transform

Detailed discussions about Sub-band coding approach for multiresolution analysis is presented in chapter 3.

# Chapter 3:
# Wavelet Analysis and Synthesis

This chapter deals with Wavelet Transform [1,7,8,10] for image coding. Following a brief theoretical introduction about wavelet transforms and their relation to filter banks, an analysis is made of the influence of several wavelet transform parameters on both the theoretical and subjective performances of wavelet transforms in image coding schemes. The application of wavelets in image coding are explained. The field of wavelet theory is very large, here only the general idea of wavelets relevant to image coding from sub-band coding [1,7,12] multiresolution approach will be discussed.

## 3.1 Wavelet Transform

Wavelet analysis is a windowing technique with variable-sized regions. Wavelet analysis allows the use of long time intervals where we want more precise low-frequency information, and shorter regions where we want high-frequency information. Here's what this looks like in contrast with the time-based, frequency-based, and STFT views of a signal.
Wavelets are mathematical functions that satisfy certain criteria, like a zero mean, and are used for analyzing and representing signals or other functions. The wavelets are a family of functions generated from a single function by translation and dilation. A set of dilations and translations $\psi_{\tau,s}(t)$ of a chosen mother wavelet $\psi(t)$ is used for analysis of a signal. The general form of these wavelets is described by

$$\Psi_{\tau,S}(t) = \frac{1}{\sqrt{s}}\Psi\left(\frac{t-\tau}{s}\right) \qquad (1)$$

Where ***s*** is the scaling (dilations) factor and $\boldsymbol{\tau}$ is the translation (location) factor. We can manipulate wavelets in two ways-the first one is translation where we change the central position of the wavelet along the time axis. the second way is scaling where we change the locations or levels. Figure 3.1-left depicts translations of a prototype (mother) wavelet while Figure 3.1-right shows dilations of it.

The **forward wavelet transform (Analysis Part)**, decomposes the input signal ***f (t)*** into the basic functions, i.e. 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**, denoted as $\mathbf{C}_{\boldsymbol{\tau},\mathbf{s}}$ i.e. **wavelet transform** is defined as

$$C_{\tau,s} = \int_{-\infty}^{\infty} f(t)\Psi_{\tau,s}(t)dt \qquad (2)$$

The **inverse wavelet transform (Synthesis Part)** conversely, uses the computed wavelet coefficients and superimposes them in order to calculate the original data set.

$$f(t) = \sum_{\tau,s} C_{\tau,s}\Psi_{\tau,s}(t) \qquad (3)$$

**Discrete Wavelet Transform**

In Discrete Wavelet Transform (DWT) the scale and translate parameters are chosen such that the resulting wavelet set forms an orthogonal set, i.e. the inner product of the individual wavelets $\Psi_{m,n}$ is equal to zero. To this end, dilation factors are chosen to be powers of 2. A common choice for **τ** and **s** is

$$\tau = 2^m, \ s = n.2^m \ \text{ where } n, m \in Z$$

Which reduces equation (1) to:

$$\Psi_{m,n}(t) = 2^{-m/2}\Psi(2^{-m}t - n)$$

**For Discrete Wavelet Transform**, the set of dilation and translation of the mother wavelet is defined as:

$$\Psi_{m,n}(t) = 2^{-m/2}\Psi(2^{-m}t - n) \qquad (4)$$

Here **m** is the scaling factor and **n** is the translation factor. It is obvious that the dilation factor is a power of 2. **Forward (DWT) and inverse transforms (IDWT)** are then calculated using the following equations:

$$\begin{aligned} f(t) &= \sum_{m,n} C_{m,n}.\Psi_{m,n}(t) \\ C_{m,n} &= 2^{-m/2}\int f(t).\Psi_{m,n}(t)dt \end{aligned} \qquad (5)$$

For efficient de-correlation of the data, an analysis wavelet set $\Psi_{\tau,s}$ should be chosen which matches the features of the data well. This together with (bi)orthogonality of the wavelet set will result in a series of sparse coefficients in the transform domain, which obviously will reduce the amount of bits needed to encode it.

The purpose of obtaining this description is that it provides a representation of the signal **f(t)** in terms of both space and frequency localization . In comparison, the Fourier transform is excellent at providing a description of the frequency content of a signal. But if the signal is non-stationary the frequency characteristics vary in space, that is in different regions the signal **f(t)** may exhibit very different frequency characteristics, the Fourier transform does not take this into account. The wavelet transform on the other hand produces a representation that provides information on both the frequency characteristics and where these characteristics are localized in space.

The coefficients **C** $_{\mathbf{m,n}}$ characterizes the projection of **f(t)** onto the base formed by $\Psi_{m,n}$. For different **m,** $\Psi_{m,n}$. represents different frequency characteristics, **n** is the translation of the dilated mother wavelet, therefore **C** $_{\mathbf{m,n}}$ represent the combined space-frequency characteristics of the signal. The **C** $_{\mathbf{m,n}}$ are called wavelet coefficients.

Practical signals are limited both in time (or space in case of images) and frequency. Time limited signals can be represented efficiently using a set of block functions (Dirac delta functions for infinitesimal small blocks). But block signals are not limited in frequency. Band-limited signals can be represented efficiently using a Fourier basis, but sines and cosines are not limited in time. Wavelets are a compromise between these worlds; wavelet functions can be neatly held finite in both time and frequency domains. Therefore they can be used to approximate data with discontinuities or spikes or detect the contours of objects in images . In Wavelet Transform, temporal analysis is performed by applying concentrated (high frequency) versions of the mother wavelet on the input data, while frequency analysis is done using the dilated (low frequency) versions of the mother wavelet. Figure 3.1-left depicts translations of a prototype wavelet while Figure 3.1-right shows dilations of it.

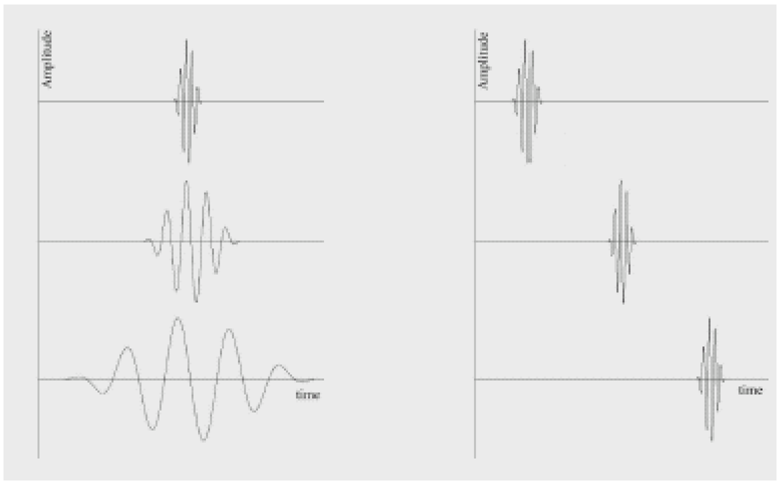


**Figure 3.1:**Translation (left) and dilations(scaling) (right) of a prototype wavelet

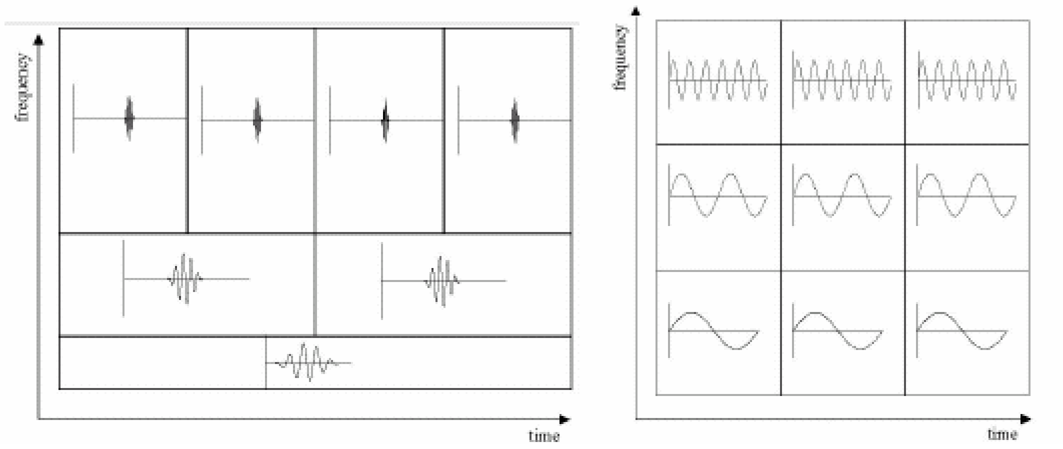


**Figure 3.2(a):**Time –Frequency plane of Discrete Wavelet Transform (left) and Fourier Transform

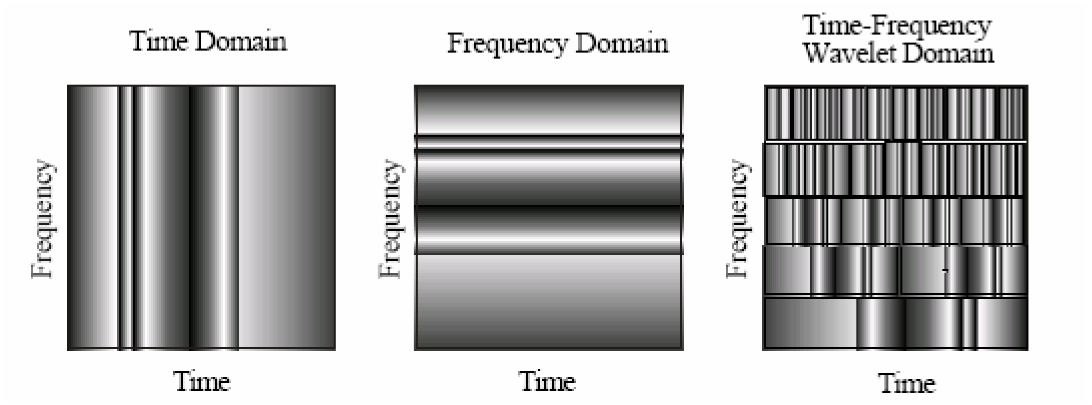


**Figure 3.2 (b)** In the time domain we have full time resolution, but no frequency localization or separation. In the Fourier domain we have full frequency resolution but no time separation. In the wavelet domain we have some time localization and some frequency localization.

## 3.2 Wavelets and its relation to Sub-band coding

The main application of wavelet theory to image coding is in the design of filters for Sub-band coding [1,7,12] . This comes from the possibility to realize the projection in equation (3) as a filter operation where the filters depend on the wavelets. The characteristics of the filters derived from the wavelets will be the same as the characteristics for the wavelets used. The properties of a sub-band coding scheme can then be discussed in terms of wavelet theory.

In **sub-band coding [1,7,12]**, an image is decomposed in to a set of band-limited components, called sub-bands, which can be reassembled to reconstruct the original image without error. Originally developed for speech and image compression, each sub-band is generated by band pass filtering the input. The resulting sub-bands can be down sampled without loss of information, as the bandwidth of the resulting sub-bands is smaller than that of original image. Reconstruction of original signal is accomplished by up-sampling, filtering and summing the individual sub-bands. The principal components of a two-band sub-band coding [1] and decoding system are shown in **Figure 3.3(a).**

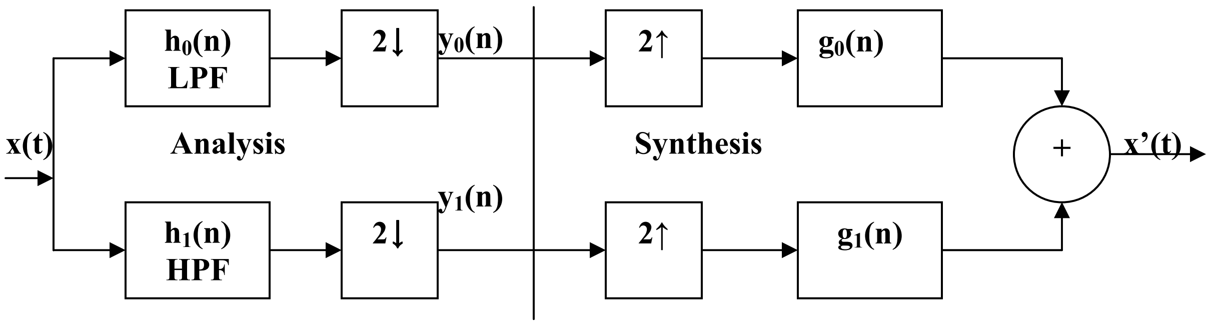


**Figure 3.3(a)A two-band filter bank for 1D sub-band coding and decoding**

(y0(n) is approximation part of x(n) and y1(n) is detail part of x(n))

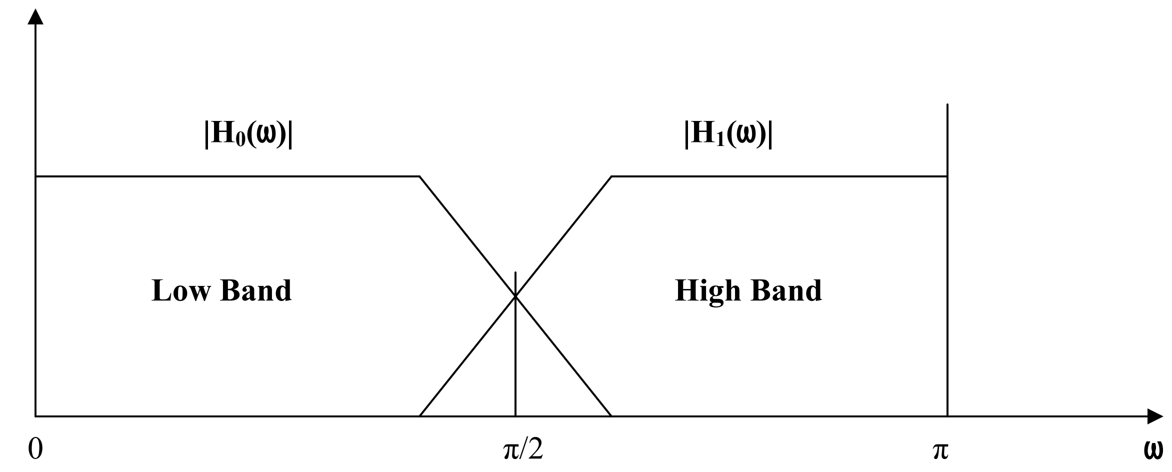


**Figure 3.3(b) Spectrum splitting properties of sub-band coding and decoding**

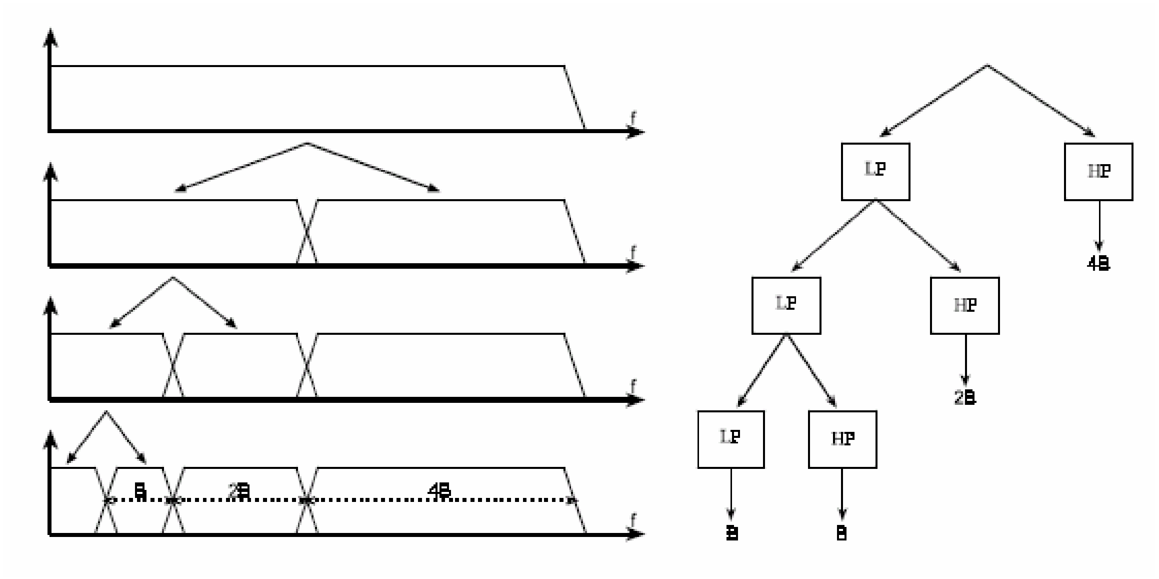


**Figure3.3(c)** Splitting the signal spectrum with an iterated filter bank.

The input to the sub-band coding [1] system figure 3.3(a) is a 1D, band limited discrete time signal x(n) for n=0,1,2,3,…..;The output sequence ,x'(n) ,is formed through the decomposition of x(n) in to $y_0(n)$ and $y_1(n)$ via analysis filters $h_0(n)$ and $h_1(n)$, and subsequent recombination via synthesis filters $g_0(n)$ and $g_1(n)$.Analysis filters $h_0(n)$ and $h_1(n)$ are half-band digital filters whose idealized transfer characteristics $H_0$ and $H_1$ are shown in figure 3.3(b).Filter $H_0$ is a low pass filter (LPF) whose output is approximation of x(n) i.e. it passes only low frequency components and $H_1$ is a high pass filter(HPF) whose output is high frequency or detail part of x(n).All filtering is performed by convolving each filter's input with its impulse response-its response to a unit amplitude impulse function, δ(n).Here our objective is to select $h_0(n)$,$h_1(n)$,$g_0(n)$,$g_1(n)$( or alternately $H_0$,$H_1$,$G_0$ and $G_1$) so that input can be reconstructed perfectly i.e. so that x'(n)=x(n).

The Z-Transform, a generalization of the DFT, is the ideal tool for studying discrete-time, sampled data systems like sub-band coding [1] system of Figure3.3(a).The Z-Transform of sequence x(n) for n=0,1,2,3,…. is

$$X(Z) = \sum_{-\infty}^{\infty} x(n) z^{-n} \qquad (6)$$

Where z is a complex variable .If $z = e^{j\omega}$, equation (6) becomes DFT. Basic advantage of using Z-Transform is that it easily handles the sampling rate changes .

**Down Sampling** by a factor of 2 in the time domain corresponds to the simple Z-domain operation:

$$x_{down}(n) = x(2n) \Leftrightarrow X_{down}(z) = \frac{1}{2}\left[X(z^{1/2}) + X(z^{-1/2})\right] \qquad (7)$$

Double arrow indicates that expressions on the left and right form a Z-Transform pair.

**Up Sampling** by a factor of 2 is defined as:

$$x^{up}(n) = \begin{cases} x(n/2) & \text{for } n=0,2,4,\ldots.. \\ 0 & \text{otherwise} \end{cases} \qquad (8)$$

If sequence x(n) is down sampled and subsequently up sampled to yield x'(n) the equations (7) and (8) gives:

$$X'(z) = \frac{1}{2}\left[X(z) + X(-z)\right] \qquad (9)$$

Where x'(n)= $Z^{-1}\left[X'(z)\right]$ is the resulting down sampled -up sampled sequence.

The term X(-z) in equation (9) is the Z-Transform of an aliased or modulated version of sequence x(n) and its inverse Z-transform is:

$$Z^{-1}\left[X(-z)\right] = (-1)^n x(n) \qquad (10)$$

Hence **Sub-band coding system's output** is:

$$X'(z) = \frac{1}{2}G_0(z)\left[H_0(z)X(z) + H_0(-z)X(-z)\right] + \frac{1}{2}G_1(z)\left[H_1(z)X(z) + H_1(-z)X(-z)\right] \qquad (11)$$

Where, for example, output of filter $h_0(n)$ is defined by transform pair:

h0(n)*x(n)=$\sum_k ho(n-k)x(k) \Leftrightarrow H_0(z)X(z)$.

As with Fourier Transform convolution in time( or spatial) domain is equivalent to the multiplication in Z-domain.

Re-arranging the terms in equation (11), we get:

$$X'(z) = \frac{1}{2}\left[H_0(z)G_0(z) + H_1(z)G_1(z)\right]X(z) + \frac{1}{2}\left[H_0(-z)G_0(z) + H_1(-z)G_1(z)\right]X(-z)$$

(12)

Where the second component ( by virtue of the fact that it contains the –z dependence) represents the aliasing that is introduced by the down-sampling up-sampling process.

Hence for **error free construction of the input**, x'(n)=x(n) and X'(z)=X(z) and **conditions** for it are:

$$H_0(-z)G_0(z) + H_1(-z)G_1(z) = 0 \qquad (13)$$

$$H_0(z)G_0(z) + H_1(z)G_1(z) = 2$$

Equation (13) eliminates aliasing by forcing the second term of equation (12) to zero; equation (14) eliminates the amplitude distortion by reducing the first term to X(z).Both can be incorporated in to the single matrix expression:

$$\mathbf{[G_0(z) \quad G_1(z)]H_m(z)=[2 \quad 0]} \qquad (14)$$

Where **Analysis Modulation Matrix** ,$H_m(z)$ is

$$H_M(z) = \begin{bmatrix} H_0(z) & H_0(-z) \\ H_1(z) & H_1(-z) \end{bmatrix} \qquad (15)$$

and **Synthesis Matrix**:

$$\begin{bmatrix} G_0(z) \\ G_1(z) \end{bmatrix} = \frac{2}{\det(H_M(z))}\begin{bmatrix} H_1(-z) \\ -H_0(-z) \end{bmatrix} \qquad (16)$$

equations (13) to (16) reveal several important characteristics of perfect reconstruction filter banks. Matrix equation (16), tells us that G1(z) is a function of H0(-z).The analysis and synthesis filters are cross modulated. For FIR filters the determinate of the modulation matrix is a pure delay i.e. **det(Hm(z)=α$z^{-(2k+1)}$.** Thus the exact form of the cross modulation is a function of α. The term $z^{-(2k+1)}$ can be considered arbitrarily since it is a shift that changes the overall delay of the filter. FIR filters are cross-modulated copies of the analysis filters.

The filter bank needed in sub-band coding [1] can be built in several ways. One way is to build many band-pass filters to split the spectrum into frequency bands. The advantage is that the width of every band can be chosen freely, in such a way that the spectrum of the signal to analyze is covered in the places where it might be interesting. The disadvantage is that we will have to design every filter separately and this can be a time consuming process. Another way is to split the signal spectrum in two (equal) parts, a low-pass and a high-pass part. The high-pass part contains the smallest details we are interested in and we could stop here. We now have two bands. However, the low-pass part still contains some details and therefore we can split it again. And again, until we are satisfied with the number of bands we have created. In this way we have created an ***iterated filter bank***. Usually the number of bands is limited by for instance the amount of data or computation power available. The process of splitting the spectrum is graphically displayed in **figure3.3(c ).** The advantage of this scheme is that we have to design only two filters, the disadvantage is that the signal spectrum coverage is fixed. Looking at **figure3.3( c )** we see that what we are left with after the repeated spectrum splitting is a series of band-pass bands with doubling bandwidth and one low-pass band. (Although in theory the first split gave us a high-pass band and a low-pass band, in reality the high-pass band is a band-pass band due to the limited bandwidth of the signal.) In other words, we can perform the same sub-band analysis by feeding the signal into a bank of band-pass filters of which each filter has a bandwidth twice as wide as his left neighbor (the frequency axis runs to the right here) and a low-pass filter. This is the same as applying a wavelet transform to the signal. The wavelets give us the band-pass bands with doubling bandwidth and the scaling function provides us with the low-pass band. From this we can conclude that a wavelet transform is the same thing as a sub-band coding scheme. The wavelets give us the band-pass bands with doubling bandwidth and the scaling function provides us with the low-pass band. From this we can conclude that a Wavelet Transform [1,7,8,10] is the same thing as a sub-band coding scheme using a constant-Q filter bank. . In general ,this kind of analysis is referred as a multiresolution analysis. Summarizing, if we implement the wavelet transform as an iterated filter bank, we do not have to specify the wavelets explicitly. If one wavelet can be seen as a band-pass filter and a scaling function is a low-pass filter, then a series of dilated wavelets together with a scaling function can be seen as a filter bank.

**A perfect reconstruction filter bank :**decomposes a signal by filtering and sub-sampling. It reconstructs it by inserting zeroes, filtering and summation. A (discrete) two-channel multirate filter bank convolves a signal $a_0$ with a low-pass filter $h_1[n] = h[-n]$ and a high-pass filter $g_1[n] = g[-n]$ and then sub-samples the output:

$$\mathbf{a_1 [n] = a_0 * h_1 [2n]}$$

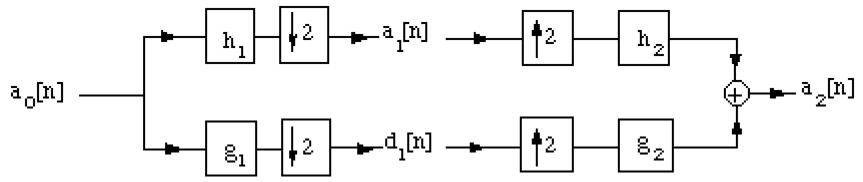
**and**

$$\mathbf{d_1 [n] = a_0 * g_1 [2n]} \; .$$

A reconstructed signal $a_2$ is obtained by filtering the zero expanded signals with a dual low-pass filter $h_2$ and a dual high-pass filter $g_2$. If $z(x)$ denotes the signal obtained from x by inserting a zero between every sample, this can be written as:

$$\mathbf{a_2 [n] = z(a_1) * h_2 [n] + z(d_1) * g_2 [n]} \; .$$

The following figure illustrates the decomposition and reconstruction process.



The filter bank is said to be a ***perfect reconstruction filter bank*** when $\mathbf{a_2 = a_0}$ . If, additionally, $\mathbf{h = h_2}$ and $\mathbf{g = g_2}$, the filters are called ***conjugate mirror filters***.

### 3.2.1 2D four band filter bank for sub-band image coding

Images often display smoothness and it would therefore be desirable that the wavelets used in image coding are smooth, this should accomplish the desired energy compaction (high correlation between the image and the wavelets/filters). The wavelets should also be compactly supported. This is necessary to have perfect reconstruction. If the wavelets have infinite support the filters derived from these wavelets will have infinitely long impulse responses, leading to infinitely long transformed signals. Filters with a finite impulse response are called FIR-filters. Smooth wavelets with compact support, leading to FIR-filters, can be obtained by solving a special dilation equation.The solution to the dilation equation is the scaling function φ(t). From this scaling function smooth wavelets can be generated. There is a tight coupling between the wavelets and the scaling function. By introducing translated and dilated versions of φ(t), in the same manner as with the mother wavelet in equation (1), we get:

$$\phi_{m,n}(t) = 2^{-m/2}\phi\left(2^{-m/2}t - n\right) \tag{17}$$

For a fixed m the $\phi_{m,n}$ constitute a basis for a vector space, $V_m$. $V_m$ and $W_m$, the vector space spanned by $\psi_{m,n}$ for a fixed $m$, are related to each other as:

$$V_{m-1} = V_m \oplus W_m \tag{18}$$

This means that $W_m$ is the orthogonal complement to $V_m$ in $V_{m-1}$. Projecting a signal $x$ onto $V_m$ produces an approximation of $x$ in terms of $\varphi_{m,n}$ and the relation (18) tells that the information lost when going from a finer approximation at resolution $m-1$ to coarser at resolution $m$ is exactly the projection of the resolution $m-1$ approximation onto $W_m$. The $W_m$ projection is referred to as detail information. If the signal is in sampled form (as images are) the $m-1$ resolution approximation can be chosen as the data itself, equation (18) then describe how to divide the data into approximation and detail parts which can be used to reconstruct the signal perfectly. We can achieve a representation of $x$ in more than two components by recursively using equation (18) on $V_m$:

$$V_m = V_{m+1} \oplus W_{m+1} \Rightarrow V_{m-1} = V_{m+1} \oplus W_{m+1} \oplus W_m \tag{19}$$

This means that **x** can be represented as an **m+1** resolution approximation and two detail signals. The process in equation (19) can be continued representing **x** as coarser and coarser approximations and more and more detail signals.The process of decomposing a signal into approximation and detail parts, corresponding to projecting **x** onto $V_m$ and $W_m$ in equation (18), can be realized as a filter bank followed by sub-sampling.( Figure3.3(a)).

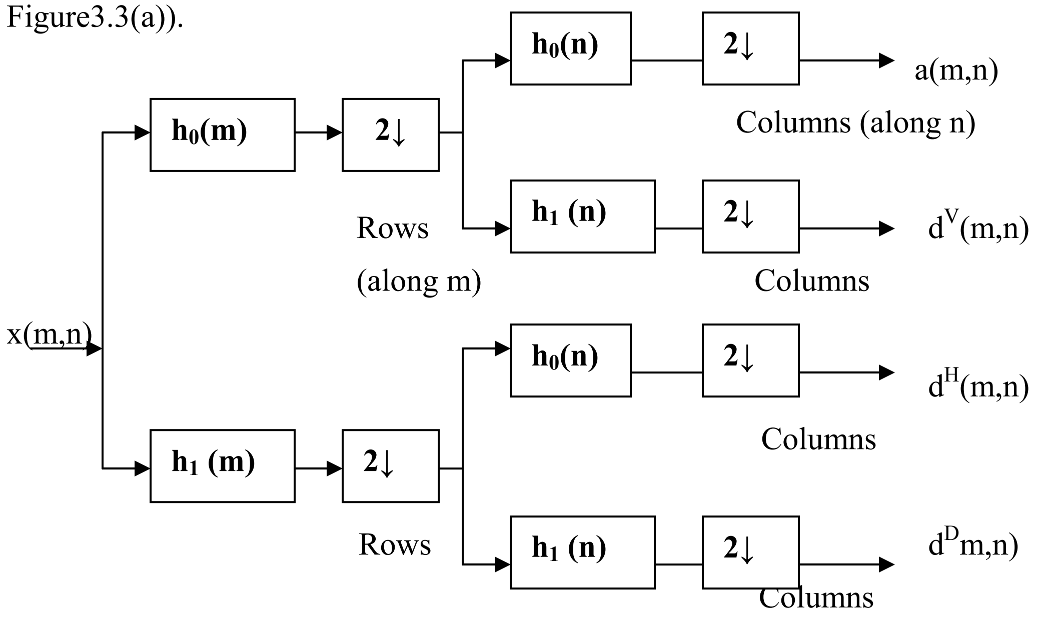


**Figure 3.4** A 2D ,four –band filter bank for sub-band Image Coding (Analysis Part)

Figure 3.4 shows 2D, four –band filter bank for sub-band Image Coding. The separable filters are first applied in one dimension (vertically) and then in the other (horizontally). Down sampling is performed in two stages-once before the second filtering operation to reduce the overall number of computations. The resulting filtered outputs , denoted a(m,n), $d^V$(m,n), $d^H$(m,n), $d^D$(m,n) are called the approximation, vertical detail, horizontal detail and diagonal detail sub-bands of the image. This is called one level decomposition. One or more of these sub-bands can be split in to four smaller sub-bands, which can be split again and so on for further level of decompositions. This process is known as analysis phase. Similarly by applying inverse part( synthesis) the original image can be reconstructed.

### 3.2.2 The Wavelet Packet Decomposition

The DWT was derived as a hierarchical sub-band structure. Another way to decompose a signal into several levels is based on the flat filter structure This method also uses filters derived from wavelets. There will be no mathematical treatment of this structure. Each filter-pair will be viewed as only a low pass high-pass pair with certain properties. The goal of this new method is to achieve a better energy compaction than the DWT by using an adaptive filter structure as opposed to the fixed structure of the DWT. The structure can be any sub-set of the flat filter structure. This decomposition is referred to as the wavelet packet decomposition or the Wavelet Packet Transform (WPT). An example of a one-dimensional WP decomposition is depicted in Figure 3.5.

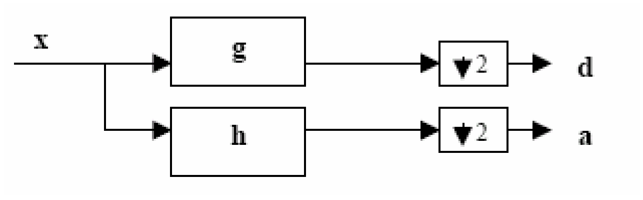


**Figure 3.5** One level of decomposition realized as a filter bank followed by sub-sampling

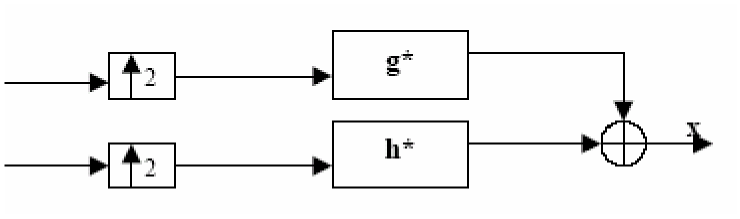


**Figure3.6** One level of reconstruction realized as an up-sampling followed by a filter bank

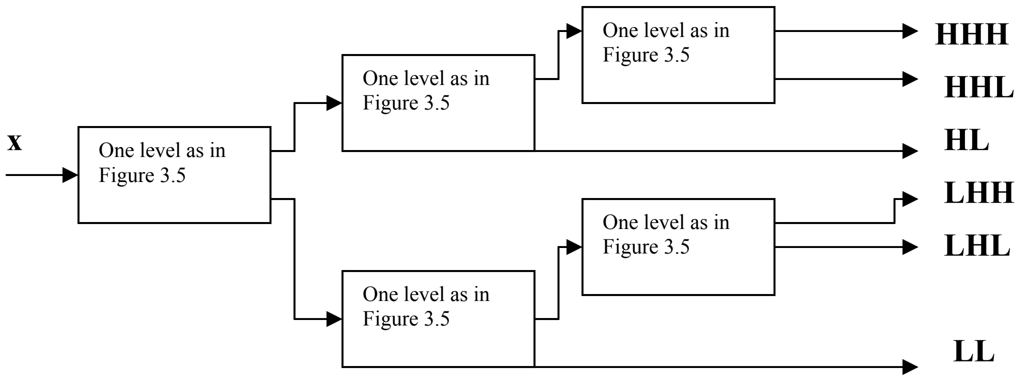

x
One level as in Figure 3.5
One level as in Figure 3.5
One level as in Figure 3.5
One level as in Figure 3.5
One level as in Figure 3.5
HHH
HHL
HL
LHH
LHL
LL

**Figure 3.7** An example of a three level WP-decomposition

In Figure 3.7 the label **XXX** of the outputs represents the different filters that have filtered this output and at which stage. **LHL** for example is a signal that has been filtered low-pass in the first stage high-pass in the second and low-pass in the third stage. As with the DWT the WPT can be inversely transformed. The filter structure is the opposite of that used when transforming and at each stage the filter bank depicted in Figure3.6 is used. Both the DWT and WPT can be extended to multi-dimensional signals.

## 3.3 Representation of Spatial and Frequency Hierarchies

### 3.3.1Representation of Spatial Hierarchies

There exist both spatial and frequency hierarchies between the different sub-bands in

both the DWT and the WPT. The spatial hierarchy for two-dimensional signals (Images) is usually visualized as seen in Figure 3.8.

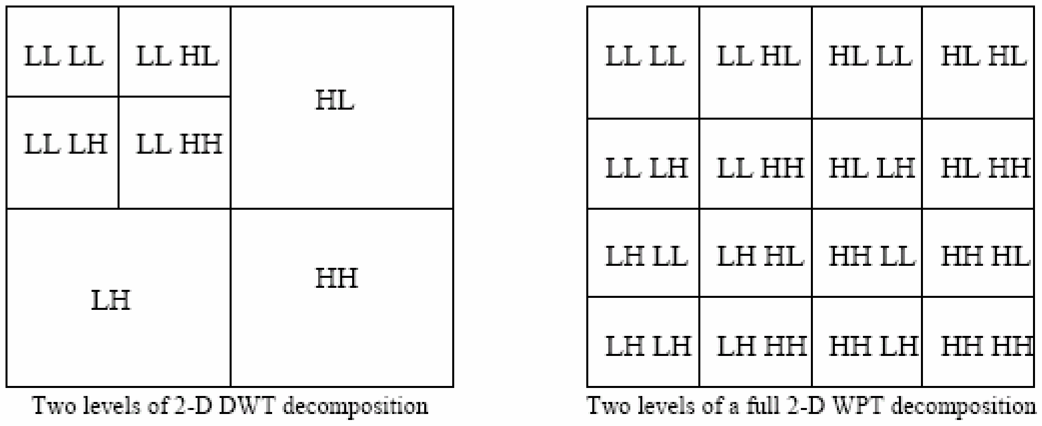


***Figure 3.8 Spatial Hierarchy for 2D Image***

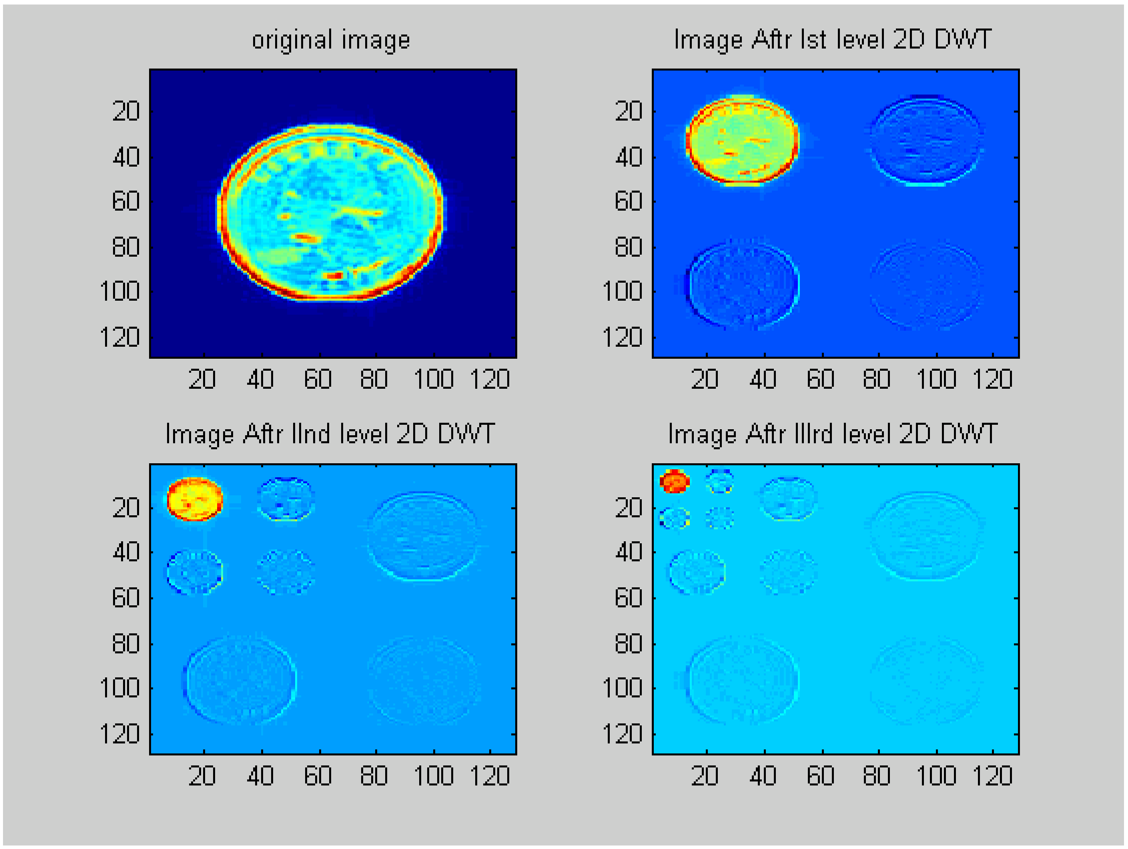


**Figure 3.8(b)** :Example of an 128x128 image at different levels of decompositions by 2D DWT

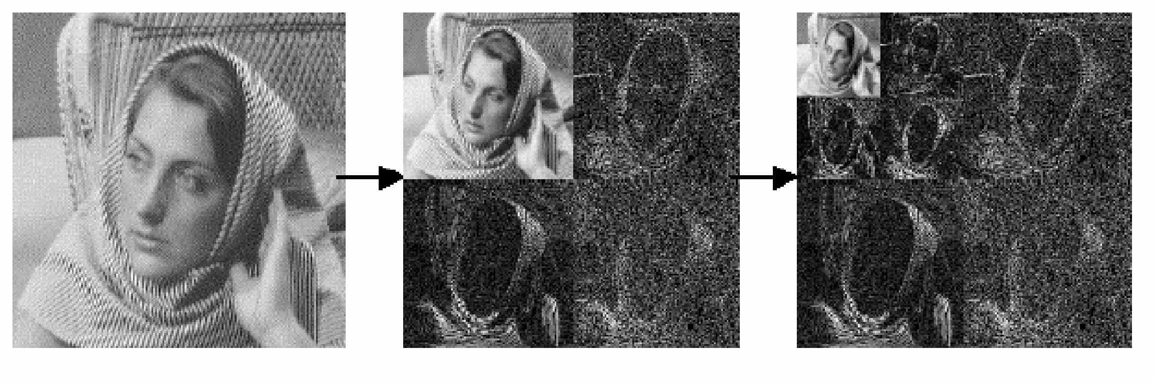


**Figure 3.8 (c)** Example: 2D DWT on Test Image Barbara at levels 1 and 2

In Figure 3.8(a) the letters correspond to the filters applied in x and y direction of the image (2D signal) respectively. Fig 3.8(a) shows the decomposition of image at second level. All rectangles correspond to a sub-image (sub-band) from the decomposition. There exist a spatial correlation between the different sub-bands. If the image has an edge in the x-direction (edge running from top to bottom), corresponding to a high frequency component, then this edge would be evident in the HL sub-image. The four bands in the upper left corner of the DWT decomposition in Figure 3.8(a) are all constructed by applying the high and low-pass filters to the approximation sub-image produced at the first stage. The x-edge would probably also be present in the approximation image. When filtering this approximation image the resulting band LL HL would display this edge. So there exist a correlation between the HL and LL HL bands. The same is of course true for the LH-LL LH and HH-LL HH bands, bands at different levels but with the same orientation. This correlation should be taken advantage of by a coding scheme for the DWT (similar reasoning leads to spatial correlation between sub-bands in the WPT case). In Figure 3.8(b) and Figure3.8(c) different levels (1,2,and3 ) of an octave-band decomposition is shown for a 128x128 gray scale image.

### 3.3.2 Representation of frequency hierarchies

Alongside the spatial hierarchy from Figure 3.8 there exists a frequency hierarchy among sub-bands. This hierarchy can be visualized by a tree structure. The tree structure is a mapping of the structure of the filter-bank used. The frequency hierarchy among the sub-bands from Figure 3.8 is depicted in Figure 3.9.

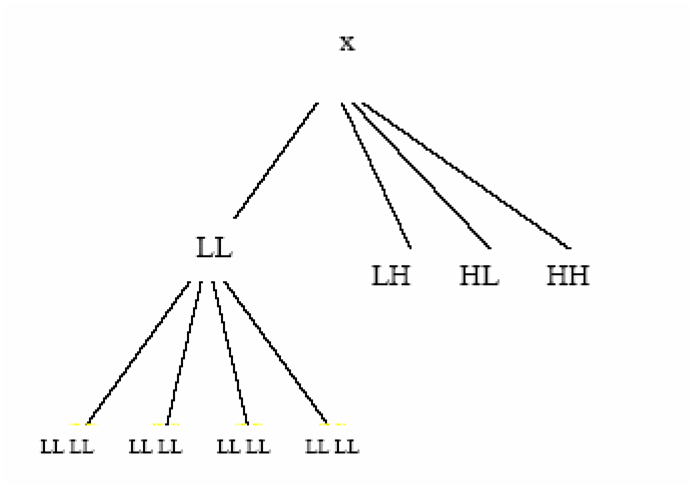


**Figure 3.9(a):** Frequency hierarchy for a two level 2D DWT decomposition

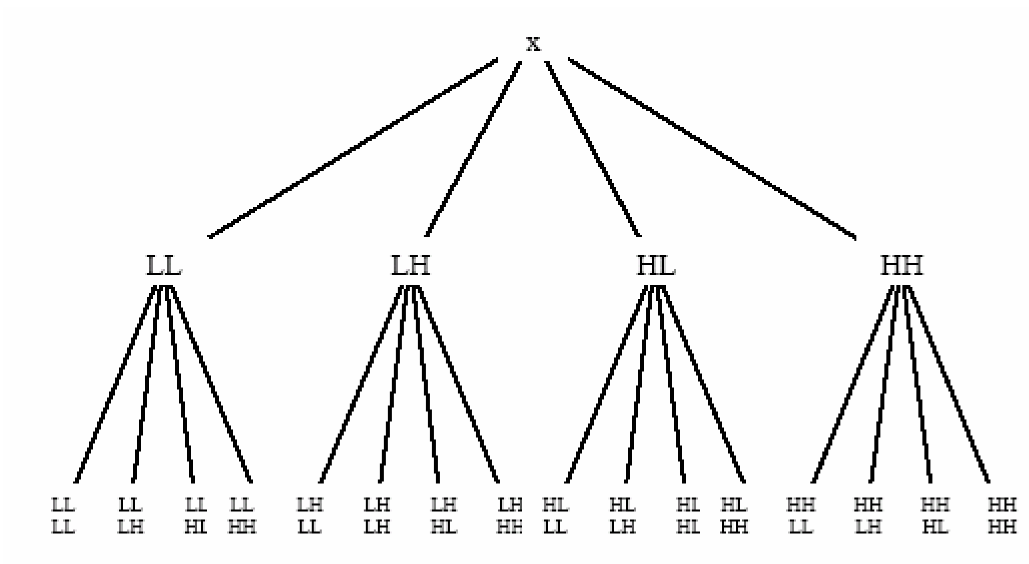


**Figure 3.9(b):** Frequency hierarchy for a two level full 2D WPT decomposition

The mapping from filter structure to frequency tree provides another description of the WPT. A transform corresponding to any sub-tree of the full frequency tree at level **N** is a WPT at level **N**. Every such sub-tree corresponds to a different wavelet basis. An observation concerning the relationship between the DWT and WPT can be made from the frequency tree. The DWT is a special case of the WPT.

## 3.4 Properties of the wavelet filters

Since, it was explained earlier that the use of wavelet theory in image coding is in the area of filter design for a sub-band coding [1] scheme. Here some desirable properties of the filters to be used for image coding will be discussed. Filter characteristics will also in some cases be translated into properties of the wavelet bases used. In the following the terms wavelet decomposition and transform will be used interchangeably.

**Some required properties of the wavelet filters are as follows:**

- Filters derived from smooth wavelets are desirable. This should lead to good correspondence with the low-pass character of images (high energy compaction), the main goal of sub-band coding.
- Second, desirable property of the transform is that it should be orthonormal. If the transform is orthonormal the reconstruction filters h* and g* can be chosen as h*=h and g*=g. An orthonormal transform (corresponding to an orthonormal wavelet basis) preserves the signal energy in the transform domain. This is especially important when a lossy coding scheme is applied to the transformed structure. The orthonormality guarantees that the distortion in the transform domain is the same as in the reconstructed domain. That is if we quantize transform components (sub-bands), and by that introduce distortion the distortion will be the same after inverse transformation of the quantized components.
- Besides smoothness and orthonormality it is desirable that the filters derived from the wavelets should be short for fast computation and have linear phase. The phase linearity makes the cascading of filters in a sub-band coding [1] scheme possible without phase compensation. The linear phase also implies symmetry of the wavelet and also symmetry of the filters derived. This symmetry is most desirable for filters used in image coding.
- By relaxing the orthonormality requirement and using biorthogonal wavelet [9] bases, filters with linear phase and smoothness can be designed. Though we have removed the orthogonality we can still achieve perfect reconstruction by choosing h* and g* differently than in the orthonormal case. By using biorthogonal bases in the filter design the resulting transform is not orhonormal, this takes away the nice feature of being able to control the distortion in the reconstruction by controlling the distortion in the transform domain. The biorthogonal base can however be chosen close to orthonormal in which case distortion in the transform domain almost corresponds to distortion in the reconstruction.

## 3.5 Advantages of using wavelets

Wavelet Transform have several advantages. Here we list a number of these in regard to image compression and processing.

1. One of the main features of Wavelet Transform, which is important for data compression and image processing applications, is its good de-correlating behavior.
2. Wavelets are localized in both the space (time) and scale (frequency) domains. Hence they can easily detect local features in a signal.
3. Wavelets are based on multi-resolution analysis. A wavelet decomposition allows to analyze a signal at different resolution levels (scales), which results in superior objective and subjective performance
4. Wavelets are smooth, which can be characterized by their number of vanishing moments. The higher the number of vanishing moments, the better smooth signals can be approximated with the wavelet basis. A function $f$ defined on the interval $[a, b]$ has $N$ vanishing moments if:

$$\int_a^b f(x)x^i dx = 0 \quad \text{for i= 0,1,....,N-1}$$

5. Fast and stable algorithms are available to calculate the Discrete Wavelet Transform and its inverse. Like FFT, the Discrete Wavelet Transform can be factored into a few sparse matrices using self-similarity properties. This results in an algorithm that requires only order of O( *n)* operations to transform a data series with length *n*. this is called Fast DWT of Mallat and Daubechies.

## 3.6 Various Wavelet families used in Image Coding:

There are different types of wavelet families whose qualities vary according to several criteria. The main criteria are:
1.The support of wavelet function ,Ψ(t), and scaling function ,Φ(t) : the speed of convergence to 0 of this function when the time t or the frequency goes to infinity, which quantifies both time and frequency localizations .
2.The symmetry, which is useful in avoiding de-phasing in image processing.
3. The number of vanishing moments for wavelet function Ψ(t) or scaling function Φ(t), which is useful for compression purposes.
4.The regularity, which is useful for getting nice features, like smoothness of the reconstructed signal or image, and for the estimated function in nonlinear regression analysis.
These are associated with two properties that allow fast algorithm and space-saving coding:
1.The existence of a scaling function Φ .
2.The orthogonality or the bi-orthogonality of the resulting analysis.
They may also be associated with these less important properties:
1.The existence of an explicit expression .
2.The ease of tabulating .
3.The familiarity with use.
The various wavelet families :
1. Wavelets for continuous wavelet transform ( Gaussian, Morlet, Mexican Hat)
2. Haar wavelet
3. Daubechies wavelets
4. Symlets
5. coiflets
6. Biorthogonal spline wavelets
7. Complex Wavelets
In this thesis, the various wavelets used ,for the performance evaluation of Image Coding, which uses Discrete Wavelet Transform are:
Haar, Daubechies and Biorthogonal wavelets [9].

### 3.6.1 The Haar Wavelet Transform :

General characteristics: The oldest and the simplest, compactly supported, biorthogonal and orthogonal wavelet, scaling function phi = 1 on [0 1] and 0 otherwise, wavelet function psi = 1 on [0 0.5], = -1 on [0.5 1] and 0 otherwise, Family Haar, Short name

haar and is the same as db1,Both DWT and CWT possible, Support width 1, Filters length 2, Number of vanishing moments for psi(wavelet function)1,it is regular and symmetric but is not continuous.
The forward transform: Each step in the forward Haar transform calculates a set of wavelet coefficients and a set of averages. If a data set $s_0, s_1, \ldots s_{N-1}$ contains N elements, there will be N/2 averages and N/2 coefficient values. The averages are stored in the lower half of the N element array and the coefficients are stored in the upper half. The averages become the input for the next step in the wavelet calculation, where for iteration i+1, $N_{i+1} = N_i/2$. The recursive iterations continue until a single average and a single coefficient are calculated. This replaces the original data set of N elements with an average, followed by a set of coefficients whose size is an increasing power of two (e.g., $2^0, 2^1, 2^2 \ldots N/2$ ).
The Haar equations to calculate an average ($a_i$) and a wavelet coefficient ($c_i$) from an odd and even element in the data set are shown below:

$$a_i = \frac{s_i + s_{i+1}}{2}$$

$$c_i = \frac{s_i - s_{i+1}}{2}$$

In wavelet terminology the Haar average is calculated by the scaling function. The coefficient is calculated by the wavelet function.

**The Haar inverse transform:** The data input to the forward transform can be perfectly reconstructed using the following equations:

$$s_i = a_i + c_i$$
$$s_{i+1} = a_i - c_i$$

**Haar forward transform via matrix multiply :**
In the linear algebra view of the forward Haar transform, the first average is calculated by the inner product of the signal $[s_0, s_1, \ldots s_{N-1}]$ and the vector, of the same size, [0.5, 0.5, 0, 0, ..., 0]. This is the scaling vector. The first coefficient is calculated by the inner product of the signal and the vector [0.5, -0.5, 0, 0, ..., 0]. This is the wavelet vector. Shifting the scaling and wavelet vectors by two and calculating the inner products calculate the next average and coefficient. In the wavelet literature scaling and wavelet values are sometimes represented by $\mathbf{h_i}$ and $\mathbf{g_i}$ respectively. In the case of the Haar transform the scaling and wavelet values would be:
**scaling function coefficients : $h_0 = 0.5$ , $h_1 = 0.5$**
**wavelet function coefficients: $g_0 = 0.5$, $g_1 = -0.5$**

The scaling and wavelet values for the Haar transform are shown below in matrix form**.**

$$\begin{matrix} h_0 & h_1 & 0 & 0 & \cdots \\ g_0 & g_1 & 0 & 0 & \cdots \\ 0 & 0 & h_0 & h_1 & \cdots \\ 0 & 0 & g_0 & g_1 & \cdots \\ \vdots & \vdots & \vdots & \vdots & \ddots \end{matrix}$$

The first step of the forward Haar transform for an eight-element signal is shown below. Here the forward transform matrix multiplies signal.

$$\begin{bmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \\ c_0 \\ c_1 \\ c_2 \\ c_3 \end{bmatrix} \Leftarrow \begin{bmatrix} a_0 \\ c_0 \\ a_1 \\ c_1 \\ a_2 \\ c_2 \\ a_3 \\ c_3 \end{bmatrix} = \begin{bmatrix} \frac{1}{2} & \frac{1}{2} & 0 & 0 & 0 & 0 & 0 & 0 \\ \frac{1}{2} & -\frac{1}{2} & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & \frac{1}{2} & \frac{1}{2} & 0 & 0 & 0 & 0 \\ 0 & 0 & \frac{1}{2} & -\frac{1}{2} & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & \frac{1}{2} & \frac{1}{2} & 0 & 0 \\ 0 & 0 & 0 & 0 & \frac{1}{2} & -\frac{1}{2} & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & \frac{1}{2} & \frac{1}{2} \\ 0 & 0 & 0 & 0 & 0 & 0 & \frac{1}{2} & -\frac{1}{2} \end{bmatrix} \bullet \begin{bmatrix} s_0 \\ s_1 \\ s_2 \\ s_3 \\ s_4 \\ s_5 \\ s_6 \\ s_7 \end{bmatrix}$$

The arrow represents a split operation that reorders the result so that the average values are in the first half of the vector and the coefficients are in the second half. To complete the forward Haar transform there are two more steps. The next step would multiple the $a_i$ values by a 4x4 transform matrix, generating two new averages and two new coefficients, which would replace the averages in the first step. The last step would multiply these new averages by a 2x2 matrix generating the final average and the final coefficient.

**The Haar inverse transform:** Like the forward Haar transform, a step in the inverse Haar transform can be described in linear algebra terms. The matrix operation to reverse the first step of the Haar transform for an eight element signal is shown below.

$$\begin{bmatrix} s_0 \\ s_1 \\ s_2 \\ s_3 \\ s_4 \\ s_5 \\ s_6 \\ s_7 \end{bmatrix} = \begin{bmatrix} 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & -1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & -1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & -1 \end{bmatrix} \bullet \begin{bmatrix} a_0 \\ c_0 \\ a_1 \\ c_1 \\ a_2 \\ c_2 \\ a_3 \\ c_3 \end{bmatrix} \Leftarrow \begin{bmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \\ c_0 \\ c_1 \\ c_2 \\ c_3 \end{bmatrix}$$

In this case the arrow represents a merge operation that interleaves the averages and the coefficients.

**Advantages and limitations of the Haar Wavelet Transform:**

**Advantages:** It is conceptually simple, fast and memory efficient, since it can be calculated in place without a temporary array. It is exactly reversible without the edge effects that are a problem with other wavelet transforms.

**Limitations:** The Haar transform also has limitations, which can be a problem for some applications. In generating each set of averages for the next level and each set of coefficients, the Haar transform performs an average and difference on a pair of values. Then the algorithm shifts over by two values and calculates another average and difference on the next pair. The high frequency coefficient spectrum should reflect all high frequency changes. The Haar window is only two elements wide. If a big change takes place from an even value to an odd value, the change will not be reflected in the high frequency coefficients.

### 3.6.2 The Daubechies D4 Wavelet Transform:

General characteristics: Orthogonal, biorthogonal and compactly supported wavelets with external phase and highest number of vanishing moments for a given support width. Associated scaling filters are minimum-phase filters. Family Daubechies, Short name db, Order N (N strictly positive integer), Examples db1 or Haar, db4, db15,both CWT and DWT possible, Support width 2N-1,Filters length 2N, Regularity about 0.2 N for large N, Symmetry far from, Number of vanishing moments for psi(wavelet function) N.

The Daubechies wavelet transforms is named after its inventor, the mathematician Ingrid Daubechies. The function displayed in Figure 3.10 is a so-called **wavelet** function from the Daubechies family of wavelet functions. The Daubechies family of wavelets are only one of a number of wavelet families. The wavelet function (**mother** wavelet) is **orthogonal** to all functions which are obtained by shifting the mother right or left by an integer amount and the mother wavelet is orthogonal to all functions which are obtained by **dilating** (stretching) the mother by a factor of $2^j$ (2 to the jth power) and shifting by multiples of $2^j$ units. The orthogonality property means that the inner product of the mother wavelet with itself is unity, and the inner products between the mother wavelet and the aforementioned shifts and dilates of the mother are zero. The collection of shifted and dilated wavelet functions is called a wavelet **basis**. The grid in **shift-scale** space on which the wavelet basis functions are defined is called the **dyadic** grid. The **orthonormality** of the Daubechies wavelets has a very important mathematical and engineering consequence: any continuous function may be uniquely **projected** onto the wavelet basis functions and expressed as a **linear combination** of the basis functions. The collection of coefficients which weight the wavelet basis functions when representing an arbitrary continuous function are referred to as the **Wavelet Transform** of the given function.

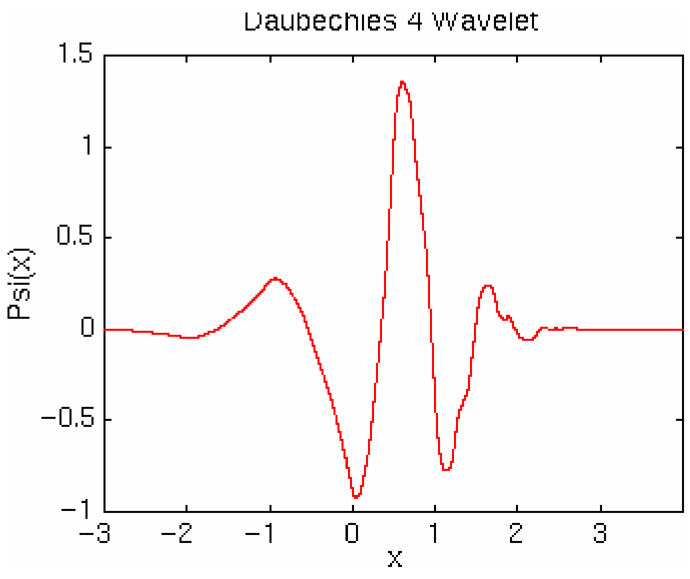


**Figure 3.10:** *The Daubechies D4 Wavelet  Function*

The Daubechies D4 transform has four wavelet and scaling function coefficients. The **scaling function coefficients are :**

$$h_0 = \frac{1 + \sqrt{3}}{4\sqrt{2}} \quad ; h_1 = \frac{3 + \sqrt{3}}{4\sqrt{2}}$$

$$h_2 = \frac{3 - \sqrt{3}}{4\sqrt{2}} \quad ; h_3 = \frac{1 - \sqrt{3}}{4\sqrt{2}}$$

Each step of the wavelet transform applies the scaling function to the  data input. If the original data set has N values, the scaling function will be applied in the wavelet transform step to calculate N/2 smoothed values. In the ordered wavelet transform the smoothed values are stored in the lower half of the N element input vector.

**The wavelet function coefficient values are:**
**$g_0 = h_3$; $g_1 = -h_2$; $g_2 = h_1$; $g_3 = -h_0$**
Each step of the wavelet transform applies the wavelet function to the input data. If the original data set has N values, the wavelet function will be applied to calculate N/2 differences (reflecting change in the data). In the ordered wavelet transform the wavelet values are stored in the upper half of the N element input vector. The scaling and wavelet functions are calculated by taking the inner product of the coefficients and four data values. The equations are shown below:

Daubechies D4 scaling function:

$$a_i = h_0 s_{2i} + h_1 s_{2i+1} + h_2 s_{2i+2} + h_3 s_{2i+3}$$
$$a[i] = h_0 s[2i] + h_1 s[2i+1] + h_2 s[2i+2] + h_3 s[2i+3];$$

Daubechies D4 wavelet function:

$$c_i = g_0 s_{2i} + g_1 s_{2i+1} + g_2 s_{2i+2} + g_3 s_{2i+3}$$
$$c[i] = g_0 s[2i] + g_1 s[2i+1] + g_2 s[2i+2] + g_3 s[2i+3];$$

Each iteration in the wavelet transform step calculates a scaling function value and a wavelet function value. The index *i* is incremented by two with each iteration, and new scaling and wavelet function values are calculated. In the case of the forward transform, with a finite data set (as opposed to the mathematician's imaginary infinite data set), *i* will be incremented until it is equal to N-2. In the last iteration the inner product will be calculated from calculated from s[N-2], s[N-1], s[N] and s[N+1]. Since s[N] and s[N+1] don't exist (they are beyond the end of the array), this presents a problem. This is shown in the transform matrix .

**Daubechies D4 forward transform matrix for an 8 element signal**

$$\begin{bmatrix} h_0 & h_1 & h_2 & h_3 & 0 & 0 & 0 & 0 & 0 & 0 \\ g_0 & g_1 & g_2 & g_3 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & h_0 & h_1 & h_2 & h_3 & 0 & 0 & 0 & 0 \\ 0 & 0 & g_0 & g_1 & g_2 & g_3 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & h_0 & h_1 & h_2 & h_3 & 0 & 0 \\ 0 & 0 & 0 & 0 & g_0 & g_1 & g_2 & g_3 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & h_0 & h_1 & h_2 & h_3 \\ 0 & 0 & 0 & 0 & 0 & 0 & g_0 & g_1 & g_2 & g_3 \end{bmatrix} \bullet \begin{bmatrix} s_0 \\ s_1 \\ s_2 \\ s_3 \\ s_4 \\ s_5 \\ s_6 \\ s_7 \end{bmatrix}$$

Note that this problem does not exist for the Haar wavelet, since it is calculated on only two elements, s[i] and s[i+1]. A similar problem exists in the case of the inverse transform. Here the inverse transform coefficients extend beyond the beginning of the data, where the first two inverse values are calculated from s[-2], s[-1], s[0] and s[1]. This is shown in the inverse transform matrix below.

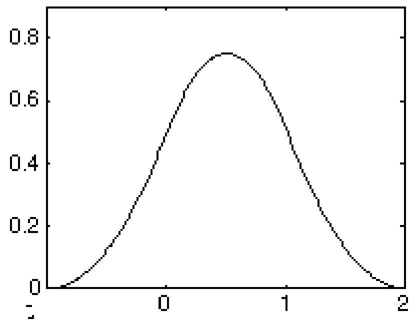
**Daubechies D4 inverse transform matrix for an 8 element transform result :**

$$\begin{bmatrix} h_2 & g_2 & h_0 & g_0 & 0 & 0 & 0 & 0 & 0 & 0 \\ h_3 & g_3 & h_1 & g_1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & h_2 & g_2 & h_0 & g_0 & 0 & 0 & 0 & 0 \\ 0 & 0 & h_3 & g_3 & h_1 & g_1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & h_2 & g_2 & h_0 & g_0 & 0 & 0 \\ 0 & 0 & 0 & 0 & h_3 & g_3 & h_1 & g_1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & h_2 & g_2 & h_0 & g_0 \\ 0 & 0 & 0 & 0 & 0 & 0 & h_3 & g_3 & h_1 & g_1 \end{bmatrix} 0 \bullet \begin{bmatrix} a_i \\ c_i \\ a_{i+1} \\ c_{i+1} \\ a_{i+2} \\ c_{i+2} \\ a_{i+3} \\ c_{i+3} \end{bmatrix}$$

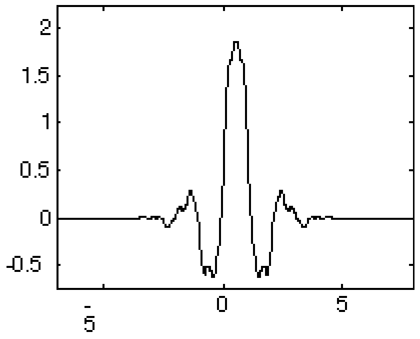
**Methods for handling the edge problem:** 1)Treat the data set as if it is periodic. The beginning of the data sequence repeats following the end of the sequence (in the case of the forward transform) and the end of the data wraps around to the beginning (in the case of the inverse transform).2) Treat the data set as if it is mirrored at the ends. This means that the data is reflected from each end, as if a mirror were held up to each end of the data sequence. 3)Gram-Schmidt orthogonalization. Gram-Schmidt orthoganalization calculates special scaling and wavelet functions that are applied at the start and end of the data set.
Zeros can also be used to fill in for the missing elements, but this can introduce significant error. The Daubechies D4 algorithm treats the data as if it were periodic. In forward transform , for the calculation of the last two values, the start of the data wraps around to the end and elements a[0] and a[1] are used in the inner product.The inverse transform works on N data elements, where the first N/2 elements are smoothed values and the second N/2 elements are wavelet function values. The inner product that is calculated to reconstruct a signal value is calculated from two smoothed values and two wavelet values. Logically, the data from the end is wrapped around from the end to the start.

### 3.6.3 Bi-orthogonal Wavelets:

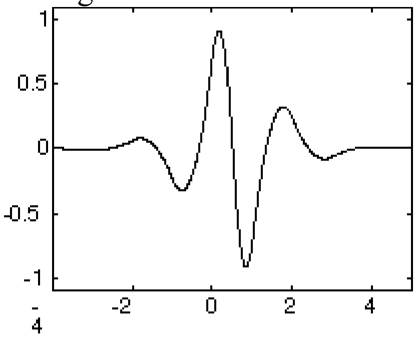
**General characteristics:** Compactly supported biorthogonal spline wavelets for which symmetry and exact reconstruction are possible with FIR filters (in orthogonal case it is impossible except for Haar),Family Biorthogonal, Short name bior, Order Nr, Nd(Nr = 1 , Nd = 1, 3, 5( r for reconstruction , d for decomposition) ), e.g.:[Nr = 2 , Nd = 2, 4, 6, 8],[ Nr = 3 , Nd = 1, 3, 5, 7, 9],[Nr = 4 , Nd = 4],[Nr = 5 , Nd = 5],[Nr = 6 , Nd = 8], Matlab examples :bior3.1, bior5.5,not Orthogonal, both DWT and CWT possible, Support width :2Nr+1 for reconstruction and $2^{Nd}$+1 for decomposition, Filters length max(2Nr,$2^{Nd}$)+2 but essentially bior Nr.Nd. The biorthogonal filters bior 4.4 , 5.5 and 6.8 are such that reconstruction and decomposition functions and filters are close in value. This family of wavelets exhibits the property of linear phase, which is needed for signal and image reconstruction. By using two wavelets, one for decomposition (on the left side) and the other for reconstruction (on the right side) instead of the same single one, interesting properties are derived. A theorem by Cohen, Daubechies and Fauveau gives sufficient conditions for building biorthogonal wavelets [9].



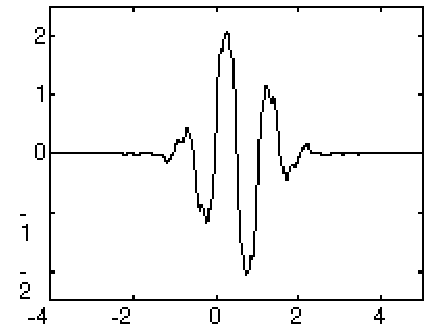
Biorthogonal cubic B-spline scaling function



Dual scaling function



Biorthogonal spline wavelet



Dual Wavelet

**Figure 3.11**: Example of Bi-orthogonal Wavelets

**3.6.4 Symlets Wavelets**

General characteristics:

Near symmetric, compactly supported, Orthogonal [1,9] and biorthogonal wavelets [9] with least asymmetry and highest number of vanishing moments for a given support width. Associated scaling filters are near linear-phase filters, Family symlets, Short name: sym, Order N where N = 2, 3,...,Examples:sym2, sym8, Both DWT and CWT possible, Support width :2N-1,Filters length:2N,Regular,Symmetry near from, Number of vanishing moments for psi=N.

**Conclusions**

In this chapter a concise review of the necessary theoretical background for understanding the Wavelet Transform had been done. The definition of the Wavelet Transform, Discrete Wavelet Transform followed by its analysis and synthesis is explained. Next, Sub-band coding and multi-resolution analysis was introduced, concept of perfect reconstruction filter banks, various families of wavelet used in image coding were introduced and subsequently the properties and advantages of the Wavelet Transform has also been discussed.

# Chapter 4
# Wavelet analysis using lifting scheme

This chapter discusses about the basic lifting schemes [7,15] and its applications.

## 4.1 Background

Wavelets based on dilations and translations of a mother wavelet are referred to as first generation wavelets or classical wavelets. Second generation wavelets, i.e., wavelets which are not necessarily translations and dilations of one function, are much more flexible and can be used to define wavelet bases for bound intervals, irregular sample grids or even for solving equations or analyzing data on curves or surfaces. Second generation wavelets retain the powerful properties of first generation wavelets, like fast transform, localization and good approximation.

Wavelet are building blocks that can quickly de-correlate data. This sentence at least incorporates three of the main features of wavelets. First of all, they are building blocks for general data sets or functions. Mathematically we say that they form a basis or, more general a frame. This means that each element of a general class can be written in a stable way as a linear combination of the wavelets. If we denote the wavelets by $\Psi_i$ and the coefficients by $\gamma_i$, we can write a general function f as

$$\mathbf{f} = \sum_i \gamma_i . \psi_i$$

Secondly, wavelets have the power to decorrelate. This means that the representation of the data in terms of the wavelet coefficients $\gamma_i$ is somehow more compact than the original representation. In information-theoretic jargon, we say that the entropy in the wavelet representation is smaller than in the original representation. In approximation-theoretic jargon, we want to get an accurate approximation of f by only using a small fraction of the wavelet coefficients. The way to get this de-correlation power is to construct wavelets, which already in some way resemble the data we want to represent. More specifically, we would like the wavelets to have the same correlation structure as the data. For example, most signals we encounter in daily life have both correlation in space and frequency. Samples which are spatially close are much more correlated then ones that are far apart, and frequencies often occur in bands. To analyze and represent such signals we need wavelets that are local in space and frequency. Typically this is achieved by building wavelets, which have compact support (localization in space), which are smooth (decay towards high frequencies), and which have vanishing moments (decay towards low frequencies). Finally, we want to quickly find the wavelet representation of the data. More precisely, we want to switch between the original representation of the data and its wavelet representation in a time proportional to the size of the data. The fast de-correlation power of wavelets is the key to applications such as data compression, fast data transmission, noise cancellation, signal recovering, and fast numerical algorithms. The purpose of this topic is to introduce the lifting scheme [7,15], a new tool in the construction of bi-orthogonal wavelets. The main difference with classic constructions is that it does not employ the Fourier transform. Until recently, the Fourier transform has been instrumental in wavelet constructions. The underlying reason is that wavelets are traditionally defined as translates and dilates of one function, and translation and dilation become algebraic operations after Fourier transform. The wavelet construction then relies on certain polynomial factorizations. We refer to wavelets which are translates and dilates of one function as first generation wavelets.

In the case of first generation wavelets, the lifting scheme [7,15] will never come up with wavelets which somehow could not be found by the techniques developed by Cohen, Daubechies, and Feauveau .

## 4.2 Advantages of Lifting Scheme:

**1.**It allows a faster implementation of the wavelet transform. Traditionally, the fast wavelet transform is calculated with a two-band sub-band transform scheme. In each step the signal is split into a high pass and low pass band and then sub-sampled. Recursion occurs on the low pass band. The lifting scheme [7,15] makes optimal use of similarities between the high and low pass filters to speed up the calculation. In some cases the number of operations can be reduced by a factor of two.

**2.** The lifting scheme allows a fully in-place calculation of the wavelet transform. In other words, no auxiliary memory is needed and the original signal (image) can be replaced with its wavelet transform.

**3.**In the classical case, it is not immediately clear that the inverse wavelet transform actually is the inverse of the forward transform. Only with the Fourier transform one can convince oneself of the perfect reconstruction property. With the lifting scheme, the inverse wavelet transform can immediately be found by undoing the operations of the forward transform. In practice, this comes down to simply reversing the order of the operations and changing each + into a - and vice versa.

**4.**Since lifting does not rely on the Fourier transform, it can be used to construct wavelets in settings where translation and dilation, and thus the Fourier transform, cannot be used. Such wavelets are referred as <u>second-generation</u> wavelets.

**5.**Lifting Scheme allows integer-to-integer transform while keeping a perfect reconstruction of the original data set. This is important for implementation and loss-less image coding.

**6.** 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.

## 4.3 Examples of second-generation wavelets:

**Wavelets on bounded domains:** The construction of wavelets on domains in a Euclidean space is needed in applications such as image segmentation and the numerical solution of partial differential equations. A special case is the construction of wavelets on an interval, which is needed to transform finite length signals without introducing artifacts at the boundaries.

**Wavelets on curves and surfaces:** To analyze data that live on curves or surfaces or to solve equations on curves or surfaces, one needs wavelets intrinsically defined on these manifolds, independent of parameterization.

**Weighted wavelets:** Diagonalization of differential operators and weighted approximation require a basis adapted to weighted measures. Wavelets biorthogonal with respect to a weighted inner product are needed.

***Wavelets and irregular sampling:*** Many real life problems require basis functions and transforms adapted to irregularly sampled data. It is obvious that wavelets adapted to these setting cannot be formed by translation and dilation. The Fourier transform can thus no longer be used as a construction tool. The lifting scheme provides an alternative. There are two ways to introduce lifting. The first one is concerned with the basis functions, i.e. the scaling functions, dual scaling functions, wavelets, and dual wavelets, and how lifting affects them.

## 4.4 The basic idea behind lifting:

Lifting scheme [7,15] is a rather new method for constructing wavelets. The main difference with the classical constructions is that it is does not rely on the Fourier transform. In this way, Lifting can be used to construct second-generation wavelets. Lifting scheme can in addition, efficiently implement classical wavelet transforms. Existing classical wavelets can be implemented with Lifting scheme [7,15] by factorization them into Lifting steps.

The basic idea behind the Lifting scheme [7,15] is very simple; one try to use the correlation in the data to remove redundancy. At first the data is split into two sets (***Split phase***): the odd samples and the even samples. If the samples are indexed beginning with 0 (the first sample is the 0th sample), the even set comprises all the samples with an even index and the odd set contains all the samples with an odd index. Because of the assumed smoothness of the data, it is predicted that the odd samples have a value that is closely related to their neighboring even samples. *N* even samples are used to predict the value of a neighboring odd value (***Predict phase***). With a good prediction method, the chance is high that the original odd sample is in the same range as its prediction. The difference between the odd sample and its prediction is calculated and the odd sample is replaced with this difference. As long as the signal is highly correlated, the newly calculated odd samples will be on the average smaller than the original one and can be represented with fewer bits. The odd half of the signal is now transformed. To transform the other half, we will have to apply the predict step on the even half as well. Because the even half is merely a sub-sampled version of the original signal, it has lost some properties that one might want to preserve. In case of images for instance, one would like to keep the intensity (mean of the samples) constant throughout different levels. The third step (***Update phase***) updates the even samples using the newly calculated odd samples such that the desired property is preserved. Now the circle is round and one can move to the next level; these three steps are applied repeatedly on the even samples and each time half of the even samples are transformed, until all samples are transformed. In following sections these three steps are explained in more detail.

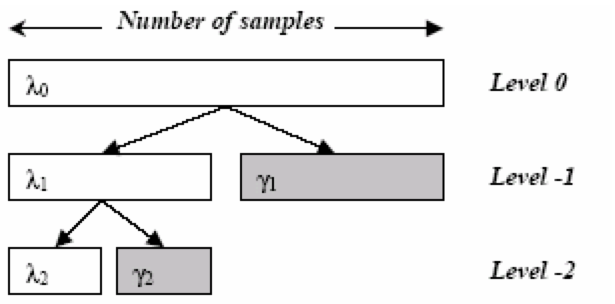


**Figure 4.1: Number of samples in different levels**

**Split phase:**
Assume that the scheme starts at level 0. We denote the data set as $\lambda_{0,k}$ where *k* represents the data element and 0 signifies the iteration level 0. In the first stage, the data set is split into two other sets: the even samples $\lambda_{-1,k}$ and the odd samples $\gamma_{-1,k}$. This is also referred to as the Lazy Wavelet transform because it does not de-correlate the data, but just sub-samples the signal into even and odd samples.

$$\lambda_{-1,k} = \lambda_{0,2k}$$
$$\gamma_{-1,k} = \lambda_{0,2k+1} \qquad (1)$$

The negative indices are used according to the convention that the smaller the data set, the smaller the index.

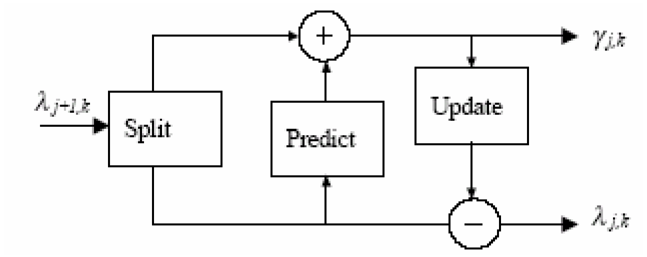


**Figure4.2:** The Lifting Scheme, forward transform: Split, Predict and Update phases

**Predict phase:**
This phase is also known as dual Lifting. In this step , the even set $\lambda_{-1,k}$ is used to predict the odd set $\gamma_{-1,k}$ using some prediction function *P*, which is independent of the data. The prediction can be defines as:

$$\textbf{Prediction=P}\ (\lambda_{-1,k}) \qquad \textbf{(2)}$$

The more correlation present in the original data, the closer the predicted value will be to the original odd set $\gamma_{-1,k}$. Now, the odd set $\gamma_{-1,k}$ is replaced by the difference between itself and its predicted value. Hence,

$$\gamma_{-1,k} = \lambda_{-1,k} - P\ (\lambda_{-1,k}) \qquad \textbf{(3)}$$

There is possibility to use different functions for prediction of odd samples. The easiest choice is to predict that an odd sample is just equal to its neighboring even sample. This prediction method results in to the *Haar* wavelet. This is an easy but not realistic choice, as there is no reason why the odd samples should be the equal to the even ones. Another option is to predict that an odd sample $\gamma_{-1,k}$ is equal to the average of the neighboring even samples at its left end right side $\lambda_{-1,k}$ , $\lambda_{0,k+1}$

$$\gamma_{-1,k}= \lambda_{-1,k} -1/2\ (\lambda_{-1,k} + \lambda_{0,k+1}) \qquad \textbf{(4)}$$

In other words, we assume that the data has a *piecewise linear* behavior over intervals of length 2. If the original signal complies with this model, all wavelet coefficients $(\gamma_{-1,k}, \forall k)$ will be zero. In other words, the wavelet coefficients measure to which extent the original fails to be linear. In terms of frequency content, the wavelet coefficients capture the high frequencies present in the original signal. The prediction does not necessarily have to be linear. One could try to find the failure to be cubic and any other higher order. This introduces the concept of interpolating subdivision. We use some value N to denote the order of the subdivision (interpolation) scheme. For instance, to find a piecewise linear approximation, we use N equal to 2. To find a cubic approximation N should be equal to 4. N is important because it sets the smoothness of the interpolating function used to find the wavelet coefficients (high frequencies). This function is referred to as the dual wavelet. Thus the number of dual vanishing moments defines the degree of the polynomials that can be predicted by the dual wavelet.

**Update phase:**
This phase is also known as primal Lifting. One can convert all the samples, except N coarsest level coefficients $\lambda_{-n,k}$, to their corresponding wavelet coefficients by iterating the predict step on the $\lambda_{j,k}$ outputs of each level for n times,. These last N coarsest coefficients are N samples from the original data and form the smallest version of the original signal which introduces considerable aliasing. Also one would like some global properties of the original data set to be maintained in the smaller versions $\lambda_{j,k}$ .In the case of images it is desired that the smaller images should have the same overall brightness, i.e. the same average pixel value. Hence, one would like the last values to be the average of all the pixel values in the original image. Introducing a third stage ,called Update phase, can solve this problem. In update stage the coefficients $\lambda_{-1,k}$ are lifted with the help of the neighboring wavelet coefficients $\gamma$**s**, so that a certain scalar quantity Q, e.g. the mean, is preserved.

$$Q(\lambda_{-1,k})=Q(\lambda_{0,k}) \qquad (5)$$

By introducing a new operator, *U,* the preservation of this quality can be ensured. Operator *U u*ses a wavelet coefficient of the current level $\gamma_{j,k}$ to update $\tilde{N}$ even samples of the same level $\lambda_{j,k}$. This preserves $\tilde{N}-1$ moments of the lambdas.

$$\lambda_{-1,k} = \lambda_{-1,k} + U(\gamma_{-1,k}) \qquad (6)$$

This is referred to as *primal lifting.* $\tilde{N}$ is also called number of real vanishing moments. The higher $\tilde{N}$ , the less aliasing effect will exist in the resulting transform.

**The lifting scheme inverse transform:**
One of the great advantages of the lifting scheme realization of a wavelet transform is that it decomposes the wavelet filters [6,7,11] 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. The inversion rules are trivial: revert the order of the operations, invert the signs in the lifting steps, and replace the splitting step by a merging step. Here follows a summery the steps to be taken for both forward and inverse transform.

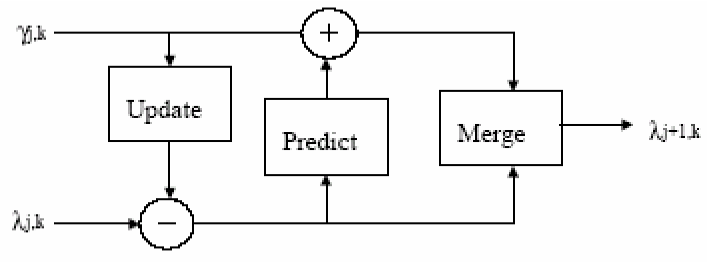


**Figure4.3 :** The lifting Scheme, inverse transform: Update, Predict and Merge stages

## Forward transform:

1- Split phase:

$$\lambda_{j,k} = \lambda_{j+1,2k}$$
$$\gamma_{j,k} = \lambda_{j+1,2k+1}$$

2- Predict phase

$$\gamma_{j,k} = \gamma_{j,k} - P(\lambda_{j,k})$$

3. Update phase:

$$\lambda_{j,k} = \lambda_{j,k} + U(\gamma_{j,k})$$

## Inverse transform:

1.Update phase

$$\lambda_{j,k} = \lambda_{j,k} - U(\gamma_{j,k})$$

2. Predict phase

$$\gamma_{j,k} = \gamma_{j,k} + P(\lambda_{j,k})$$

3. Merge phase:

$$\lambda_{j+1,2k} = \lambda_{j,k}$$
$$\lambda_{j+1,2k+1} = \gamma_{j,k}$$

**4.5 Signal Extensions and 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. There are number of 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. Another option is make the signal periodic, i.e. to repeat the signal at its ends (Figure 4-4).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 symmetrical 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 4-4). An alternative approach is to modify the filter near the boundaries. In this method, not the signal, but the filter is modified around the boundaries to construct boundary filters that preserve filter’s orthogonality . The lifting scheme [11] provides a related method for handling filtering near the boundaries.

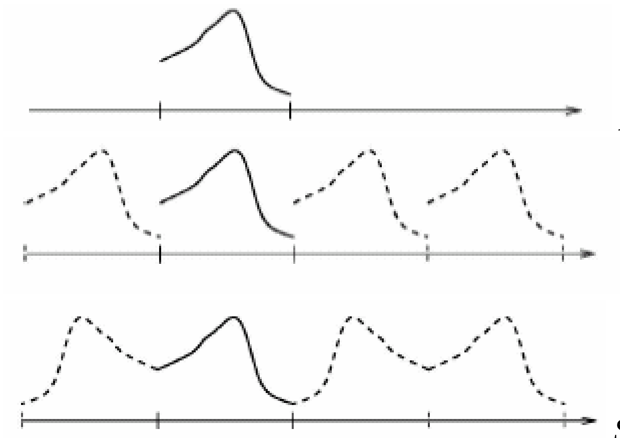


**Figure4.4 :** Examples of signal extension

## 4.6 Examples of Wavelet filters using Lifting Scheme:

The filters are denoted either by their canonical names (e.g. Haar), by (N,N') where N_ (respectively N') is the number of vanishing moments of g' (respectively g Ì ), or by (la-ls) where la is the length of analysis filter h' . and ls is the length of the synthesis filter h. We start with a sequence x={$x_l$ | l ε z} and denote the result of applying the low-pass filter h. (respectively high-pass filter g) and down sampling as a sequence s={ $s_l$ | l ε z} (respectively d).The intermediate values computed during lifting is denoted with sequences $s^{(i)}$ and $d^{(i)}$ .

**4.6.1 Haar wavelets.** In the case of (unnormalized) Haar wavelets we have that

$$h(z) = 1 + z^{-1}$$
$$g(z) = -1/2 + 1/2z^{-1}$$
$$\tilde{h}(z) = 1/2 + 1/2z^{-1}$$
$$\tilde{g}(z) = -1 + 1z^{-1}$$

Using the Euclidean algorithm we can thus write the polyphase matrix as:

$$P(z) = \begin{bmatrix} 1 & -1/2 \\ 1 & 1/2 \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 1 & 1 \end{bmatrix} \begin{bmatrix} 1 & -1/2 \\ 0 & 1 \end{bmatrix}$$

Thus on the analysis side we have:

$$P(z)^{-1} = \tilde{P}(1/z) = \begin{bmatrix} 1 & 1/2 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 \\ -1 & 1 \end{bmatrix}$$

*This corresponds to the following implementation of the forward transform*:

$$s_l^{(0)} = x_{2l}$$
$$d_l^{(0)} = x_{2l+1}$$
$$d_l = d_l^{(0)} - s_l^{(0)}$$
$$s_l = s_l^{(0)} + 1/2\, d_l$$

*While the inverse transform is given by:*

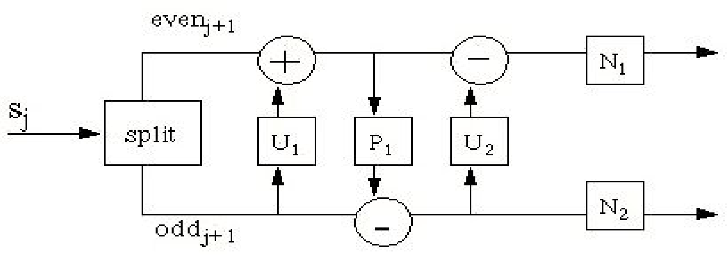
$$s_l^{(0)} = s_l - 1/2\, d_l$$
$$d_l^{(0)} = d_l + s_l^{(0)}$$
$$x_{2l+1} = d_l^{(0)}$$
$$x_{2l} = s_l^{(0)}$$

**4.6.2 Daubechies D4 Transform :**

This wavelet Lifting Scheme was developed by Wim Sweldens and others. Wavelet Lifting Scheme algorithms have several advantages. They are memory efficient and do not require a temporary array as the version of the Daubechies D4 transform above does. As the diagrams below show, the inverse transform is the mirror of the forward transform, when additions exchanged for subtractions.

**Forward Transform:** Lifting Scheme wavelet transforms are composed of Update and Predict steps. In this case a normalization step has been added as well. One forward transform step is shown in the diagram below.

Forward transform step of the lifting scheme version of the Daubechies D4:



Daubechies D4 forward wavelet transform

**Figure 4.5**

The split step divides the input data into even elements which are stored in the first half of an N element array section ( $S_0$ to $S_{half-1}$) and odd elements which are stored in the second half of an N element array section ($S_{half}$ to $S_{N-1}$). In the forward transform equations below the expression S[half+n] references an odd element and S[n] references an even element. Although the diagram above shows two normalization steps, in practice they are folded into a single function.

**Forward transform step equations**:

**Update 1:**

for n=0 to half-1

$S[n] = S[n] + \sqrt{3}s[half + n]$

**Predict :**

$$S[half] = s[half] - \frac{\sqrt{3}}{4} S[0] - \frac{\sqrt{3}\;\; -2}{4} S[half-1]$$

for n=1 to half-1

$$S[half+n] = s[half+n] - \frac{\sqrt{3}}{4} S[n] - \frac{\sqrt{3}\;\; -2}{4} S[n-1]$$

**Update 2:**
for n=0 to half-2
S[n]=S[n]-S[half+n+1]
S[half-1]=S[half-1]-S[half]

**Normalize:**
for n=0 to half-1

S[n]= $\frac{\sqrt{3}\;\; -1}{\sqrt{2}} S[n]$

S[n+half]= $\frac{\sqrt{3}\;\; +1}{\sqrt{2}} S[n+half]$

**Inverse transform step of the lifting scheme version of the Daubechies D4:** One of the elegant features of Lifting Scheme versions of the wavelet transform is the fact that the inverse transform is a mirror of the forward transform, in which addition and subtraction operations are interchanged.

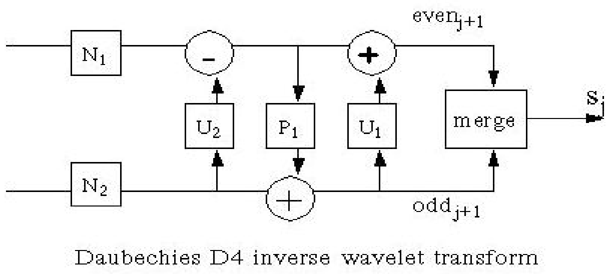


**Figure 4.6**

The merge step interleaves elements from the even and odd halves of the vector (e.g., *even*$_0$, *odd*$_0$, *even*$_1$, *odd*$_1$, ...). As the diagram above shows, the inverse transform equations have addition and subtraction operations interchanged. The inverse normalization step works because, $\frac{\sqrt{3}\;\; -1}{\sqrt{2}} \bullet \frac{\sqrt{3}\;\; +1}{\sqrt{2}} = 1$.

**Inverse transform step equations:**
**Update 1':**
For n=0 to half-1
$S[n] = S[n] - \sqrt{3}s[half + n]$

**Predict':**
$S[half] = s[half] + \frac{\sqrt{3}}{4}S[0] + \frac{\sqrt{3} \quad -2}{4}S[half - 1]$
for n=1 to half-1
$S[half + n] = s[half + n] + \frac{\sqrt{3}}{4}S[n] + \frac{\sqrt{3} \quad -2}{4}S[n-1]$
**Update 2':**
for n=0 to half-2
S[n]=S[n]+S[half+n+1]
S[half-1]=S[half-1]+S[half]

**Normalize':**
for n=0 to half-1
S[n]=$\frac{\sqrt{3} \quad +1}{\sqrt{2}}S[n]$
S[n+half]=$\frac{\sqrt{3} \quad -1}{\sqrt{2}}S[n + half]$

**4.6.3 Cubic B-splines. T**he CDF(4,2) biorthogonal filter . The scaling function here is a cubic B-spline. This example can be obtained again by using the factoring algorithm. The filters are given by:

$$h(z) = 3/4 + 1/2\,(z + z^{-1}) + 1/8\,(z^2 + z^{-2})$$
$$g(z) = 5/4z^{-1} - 5/32\,(1 + z^{-2}) - 3/8\,(z + z^{-3}) - 3/32\,(z^2 + z^{-4})$$

and the factorization reads:

$$P(z) = \begin{bmatrix} 1 & 1/4(1+z^{-1}) \\ 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 \\ (1+z) & 1 \end{bmatrix} \begin{bmatrix} 1 & -3/16(1+z^{-1}) \\ 0 & 1 \end{bmatrix} \begin{bmatrix} 1/2 & 0 \\ 0 & 2 \end{bmatrix}$$

**4.6.4 Biorthogonal CDF 9/7 Wavelet filter:**

Here the lifting scheme [15] and its result for popular CDF(9-7) filter pair has been presented. The analysis filter h' has 9 coefficients, while the synthesis filter h has 7 coefficients. Both high-pass filters g and g' have 4 vanishing moments. We choose the filter with 7 coefficients to be the synthesis filter because it gives rises to a smoother scaling function than the 9 coefficients one (coefficients need to multiplied by $\sqrt{2}$ ). By using factoring algorithm [15] for the following analysis filters:

$\tilde{h}_e(z) = h_4\,(z^2 + z^{-2}) + h_2\,(z + z^{-1}) + h_0$ and

$\tilde{h}_o(z) = h_3\,(z^2 + z^{-1}) + h_1\,(z + 1)$.

The coefficients of the remainders are computed as:

$$r_0 = h_0 - 2\,h_4\,h_1/h_3$$
$$r_1 = h_2 - h_4 - h_4\,h_1/h_3$$
$$s_0 = h_1 - h_3 - h_3\,r_0/r_1$$
$$t_0 = r_0 - 2\,r_1.$$

Then define:

$$\alpha = h_4/h_3 \approx -1.586134342$$
$$\beta = h_3/r_1 \approx -0.05298011854$$
$$\gamma = r_1/s_0 \approx 0.8829110762$$
$$\delta = s_0/t_0 \approx 0.4435068522$$
$$\zeta = t_0 = r_0 - 2\,r_1 \approx 1.149604398$$

Now, polyphase matrix representation of CDF(9,7) wavelet filter is:

$$\tilde{P}(z) = \begin{bmatrix} 1 & \alpha(1+z^{-1}) \\ 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 \\ \beta(1+z) & 1 \end{bmatrix} \begin{bmatrix} 1 & \gamma(1+z^{-1}) \\ 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 \\ \delta(1+z) & 1 \end{bmatrix} \begin{bmatrix} \zeta & 0 \\ 0 & 1/\zeta \end{bmatrix}$$

Note that here too many other factorizations exist; the one we chose is symmetric: every quotient is a multiple of (z+1). This shows how we can take advantage of the non-uniqueness to maintain symmetry.

The factorization leads to the following implementation (Lifting Steps):

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

## *CDF 9/7 lifting scheme filter sequence coefficients*

Sequence=$[[\alpha,\beta,\gamma,\delta;\alpha,\beta,\gamma,\delta]$
= [-1.58613432,-0.05298011854,0.8829110762,0.4435068522;
-1.58613432,-0.05298011854,0.8829110762,0.4435068522];

Scale Factor = $\zeta$=1.149604398.

In this thesis, lifting version of CDF (9,7) is implemented and an analysis is performed with reference to performance. The function inputs are an image X and a number N. The output is the N-stage forward transform of X. For the inverse transform, negate N. While many wavelet packages use periodic boundary handling, this function uses symmetric boundary handling. As images are generally not periodic, this is more appropriate and has superior image compression capabilities. Images sizes below 256x256 transform quickly, while larger images take a few seconds**.**

# Chapter 5:
**SPIHT based Quantization and Coding of Wavelet Coefficients**

This chapter discusses the Quantization and coding of the Wavelet Transformed Image i.e. wavelet coefficients, which is to be sent to decoder for reconstruction. Main compression of Image occurs in this phase. A SPIHT [2,3] based fast coder based on progressive transmission and zero-tree coding method is used and implemented in MATLAB in this Thesis for Image compression. An overview of SPIHT coder, its working, algorithm, flowchart and an example is provided.

## 5.1 Quantization and coding

In the previous chapter the link between wavelet theory and sub-band coding [1] was established. Wavelet theory is used to design filters in a sub-band coding scheme. The sub-band coding scheme should achieve energy compaction of the signal. The motivation for this was that it should be easier to quantize and code the signal if it was split into parts. One of the simplest ways to quantize the sub-bands is to use the method, which uses separate scalar quantizers for each sub-band which tries to minimize the resulting distortion by assigning bits from a budget to the sub-bands based on their variance but this strategy does not take into account the intra band dependencies among the sub-bands. The embedded zero tree wavelet algorithm (EZW) is a simple, yet remarkably effective, image compression algorithm, having the property that the bits in the bit stream are generated in order of importance, yielding a fully embedded code. The embedded code represents a sequence of binary decisions that distinguish an image from the "null" image. Using an embedded coding algorithm, an encoder can terminate the encoding at any point thereby allowing a target rate or target distortion metric to be met exactly. Also, given a bit stream, the decoder can cease decoding at any point in the bit stream and still produce exactly the same image that would have been encoded at the bit rate corresponding to the truncated bit stream. In this thesis a very fast and efficient algorithm, SPIHT [3], is used for the quantization and coding of wavelet coefficients, which is an extension or modified version of EZW [4] algorithm. SPIHT achieves good performance by exploiting the spatial dependencies of the DWT coefficients in different sub-bands i.e. it takes in to account the intra-band dependencies. The set partitioning in hierarchical trees (SPIHT) quantization scheme is used to generate all of the results in this thesis.

### 5.2 SPIHT (Set partitioning in Hierarchical Trees) coder:

The inter-band spatial dependencies are captured in the form of parent-child relationships, which is illustrated in **Figure 5.1**. The arrows in **Figure 5.1** points from the parent node to its four children. With the exception of the coarsest sub-band and the finest sub-bands, each DWT coefficient (parent) at the i $^{th}$ level of decomposition is spatially correlated to four child coefficients at level (i-1) in the form of a 2 x 2 block of adjacent pixels. These four child coefficients are at the same relative location in the sub-band decomposition structure. This relationship is utilized during SPIHT [3] quantization: if a parent coefficient is insignificant with respect to a particular threshold, then all of its children would most likely be insignificant and similarly, significant coefficients in the finer sub-bands most likely correspond to a significant parent in the coarser sub-band. This results in significant savings: only the parent's position information needs to be coded since the children's coordinates can be inferred from the parent's position information.

**Parent=(i, j)**

**Children=[(2i, 2j), (2i+1,2j), (2i, 2j+1), (2i+1,2j+1)]**

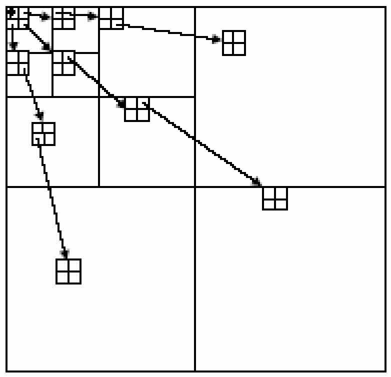


**Figure 5.1** Parent-child relationships for a 3-level Wavelet decomposition.

**SPIHT** captures the current bit-plane information of all the DWT coefficients and organizes them into three ordered lists:

1. List of significant pixels/coefficients (LSP).
2. List of insignificant pixels/coefficients (LIP).
3. List of insignificant sets of pixels/coefficients (LIS).

LSP constitutes the coordinates of all coefficients that are significant. LIS contains the roots of insignificant sets of coefficient. They can be of two different types; the first type known as TYPE D has all the descendants insignificant within a given bit-plane, the second type known as TYPE L excludes the four children of the root node. Finally, LIP contains a list of all the coefficients that do not belong to either LIS or LSP and are insignificant. The operation of SPIHT can be grouped into three sequential steps: initialization, sorting pass (SP) and refinement pass (RP) & threshold update:

**1. Initialization:** The initial threshold is set to 2 $^{\log2(\max(|Cij|j))}$, where max(|Cij|) is the largest DWT coefficient. The algorithm starts at the coarsest band in the sub-band pyramid. All the coefficients in the sub-band are added to the LIP and the coefficients with descendants (tree roots) are added as to LIS as TYPE D. Thus, during initialization, every coefficient is initialized to an insignificant state.

**2. Sorting pass:** At each threshold level, the LIP is coded first, followed by the entries in LIS. A given entry in LIP is tested and moved to LSP if found significant. The sign bit of the significant coefficient is also immediately coded. The LIS entries are coded differently. For TYPE D LIS entries, if any member in the hierarchical tree is found significant, the immediate children are tested and are added to either LIP or LSP. The parent is added to the end of LIS as a TYPE L entry or removed from the LIS if it does not have any grandchildren. For TYPE L entries, if any member in the hierarchical tree is found significant, the immediate children are removed and added as TYPE D entries to the end of LIS. Processing continues till the end of LIS is reached .LSP also records the position of the coefficients that are found significant during the current pass.

**3. Refinement pass and threshold update:** Refinement pass adds precision to the LSP entries obtained before the current sorting pass by outputting the most significant bit corresponding to the existing threshold. On completion of the refinement, the threshold is halved and the cycle is repeated starting from step 2.

## 5.3 SPIHT Algorithm

Let

**O** (i, j): set of coordinates of all offspring of node (i,j); children only

**D** (i, j): set of coordinates of all descendants of node (i,j); children, grandchildren, great-grand, etc.

**H** (i, j): set of all tree roots (nodes in the highest pyramid level); parents

**L** (i, j): **D** (i, j) – **O** (i, j) (all descendents except the offspring); grandchildren, great grand, etc.

**1. Initialization:**

$n = \lfloor \log_2 (\max | coeff(i, j) |) \rfloor$

**LIP**=All elements in H

**LSP**=Empty

**LIS**= D's of Roots

**2. Sorting pass (Significance map Encoding):**

**A) Process LIP**

```
For each coeff (i, j) in LIP
          Output $S_n$ (i, j)
          If $S_n$ (i, j)=1
                    Output sign of coeff (i, j):0/1=-/+
                    Move  (i, j) to LSP
          End if
End loop over LIP
```

**B) Process LIS**

```
For each set (i, j) in LIS
          If (entry is of type D)
                    Send Sn (D (i, j));     Where $S_n(\Gamma) = \begin{cases} 1, & \max_{(i,j)\in\Gamma} \{|C_{i,j}|\} \geq 2^n \\ O, otherwise & \end{cases}$
                    If $S_n$ (D (i, j))=1
                     For each $(k, l) \in O(i, j)$
                              Output $S_n$ (k, l)
                     If $S_n$ (k, l)=1, then add (k, l) to the LSP and output sign of coeff:0/1=-/+
                     If $S_n$ (k, l)=0, then add (k, l) to the end of LIP
                     End for
                     End if
```

```
        Else (entry is of type L)
                Send Sn (L (i, j))
                If $S_n$ (L (i, j))=1
                Add each $(k,l) \in O(i,j)$ to the end of LIS as entry of type D
                Remove (i, j) from LIS
End if on type
End loop over LIS
```

**C) Refinement Pass:**

**Process LSP**

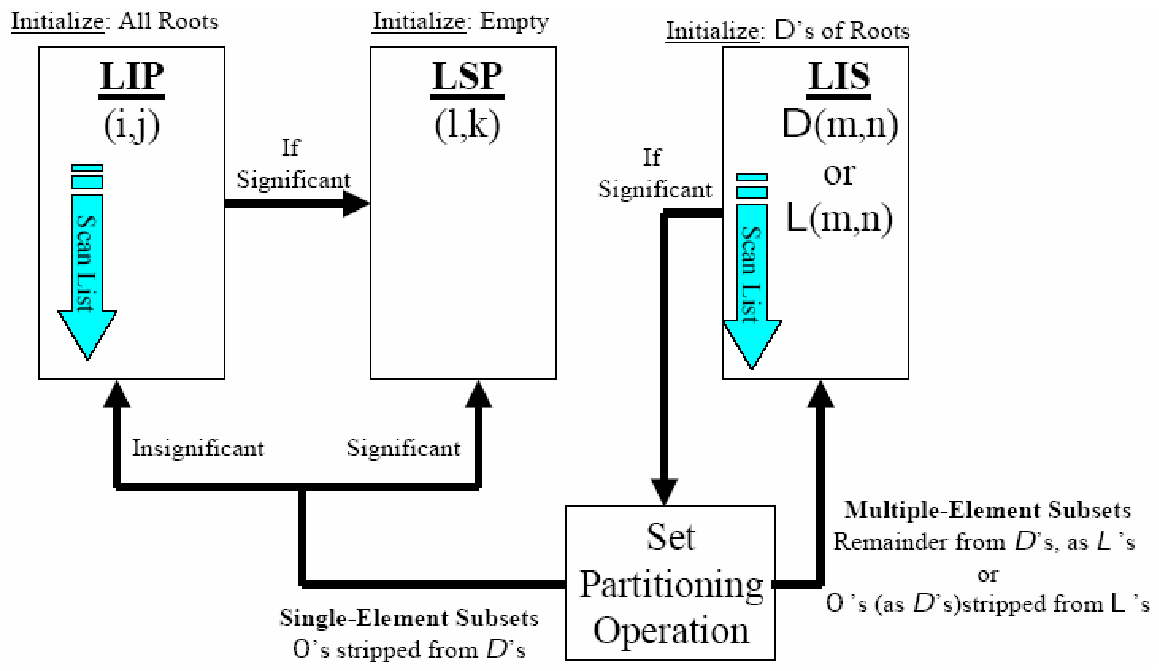
```
For each element (i, j) in LSP (except those just added above)
                Output the nth MSB of coeff
End loop over LSP
```

<u>**Threshold Update:**</u> Decrement n by 1.

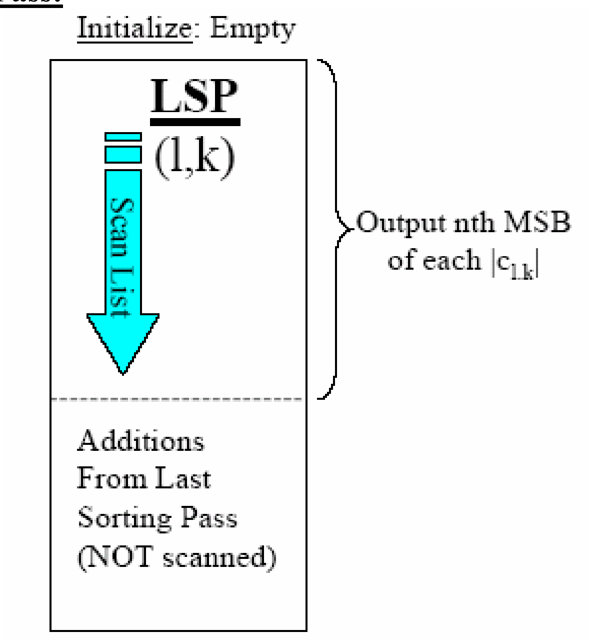
Go to Significance map encoding step

## 5.4 Working of SPIHT

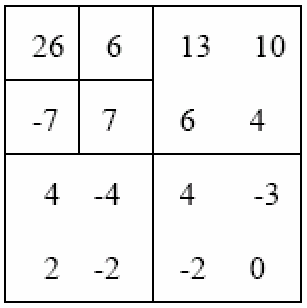
### SPIHT Sorting Pass:



**SPIHT Refinement Pass:**

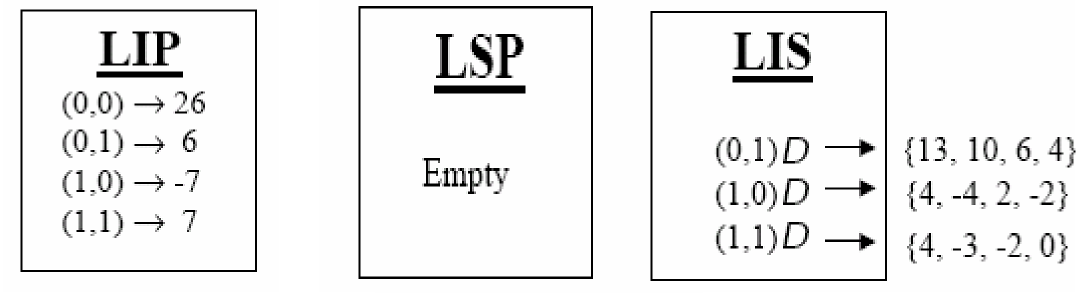

Initialize: Empty
LSP
(l,k)
Scan List
Output nth MSB of each $|c_{l,k}|$
Additions From Last Sorting Pass (NOT scanned)

**5.5 SPIHT Example:**



| | | | |
|---|---|---|---|
| 26 | 6 | 13 | 10 |
| -7 | 7 | 6 | 4 |
| 4 | -4 | 4 | -3 |
| 2 | -2 | -2 | 0 |

**1. Initialization:**

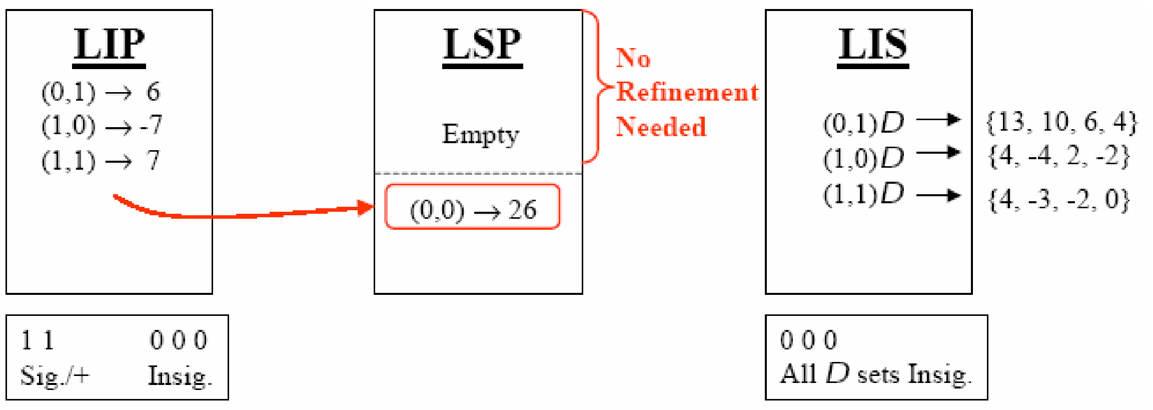

LIP
$(0,0) \to 26$
$(0,1) \to 6$
$(1,0) \to -7$
$(1,1) \to 7$
LSP
Empty
LIS
$(0,1)D \longrightarrow \{13, 10, 6, 4\}$
$(1,0)D \longrightarrow \{4, -4, 2, -2\}$
$(1,1)D \longrightarrow \{4, -3, -2, 0\}$

$$\boxed{n = \lfloor \log_2(26) \rfloor = 4}$$

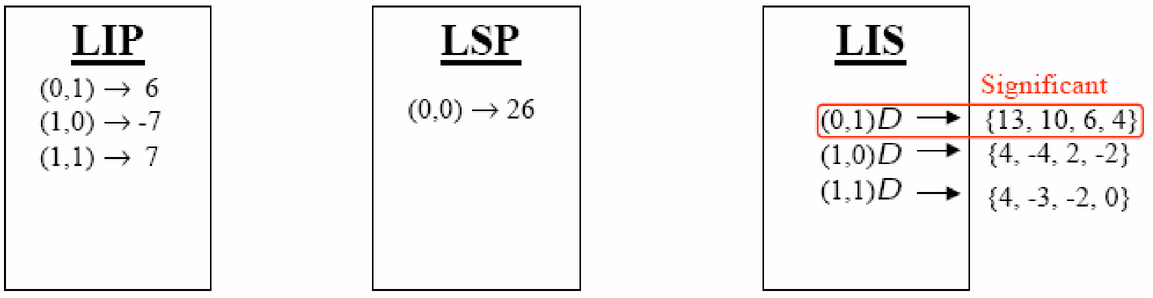
**2. Iteration 1**

**<u>After First sorting Phase:</u>**

**n=4;Threshold=16**



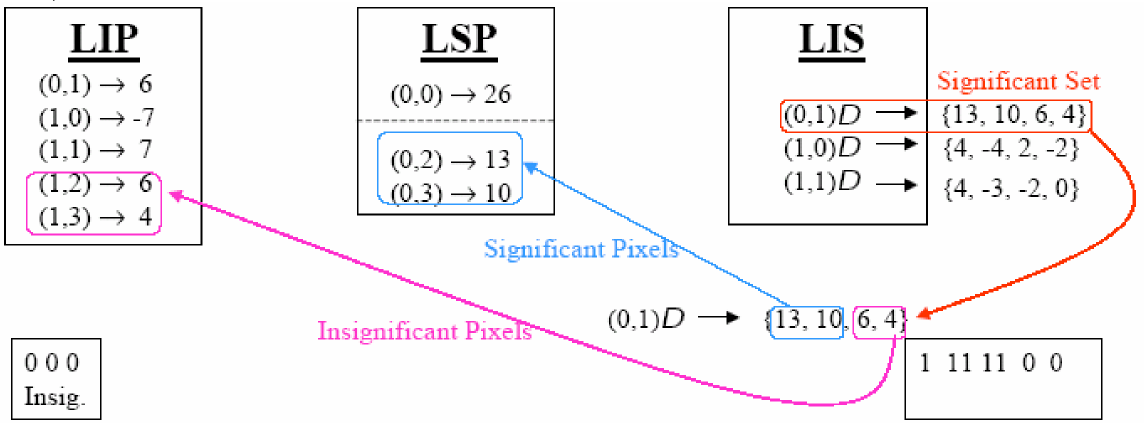
**<u>After First Refinement Phase:</u>**



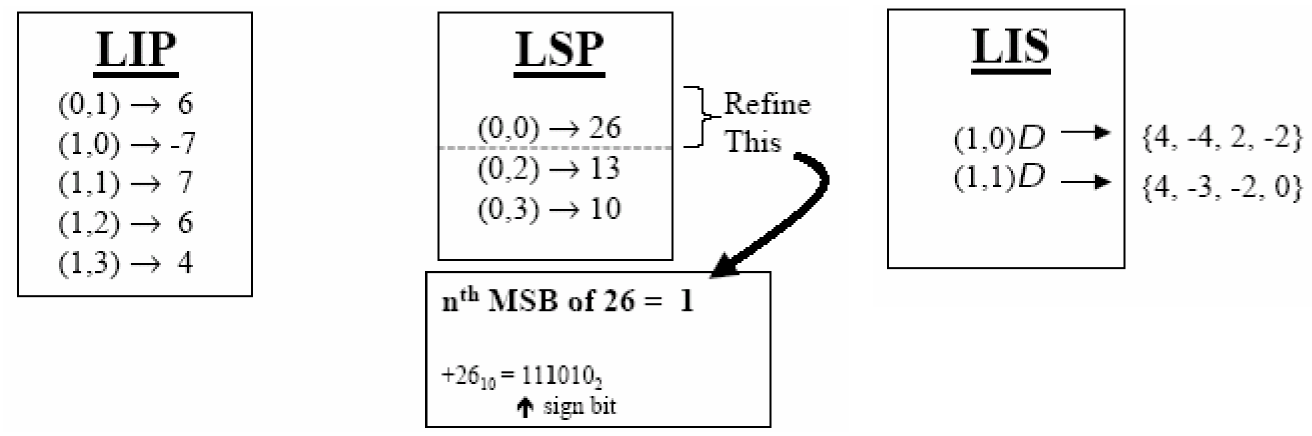
**3. Iteration 2**

**<u>During Second Sorting Pass:</u>**

**n=3; Threshold=8**

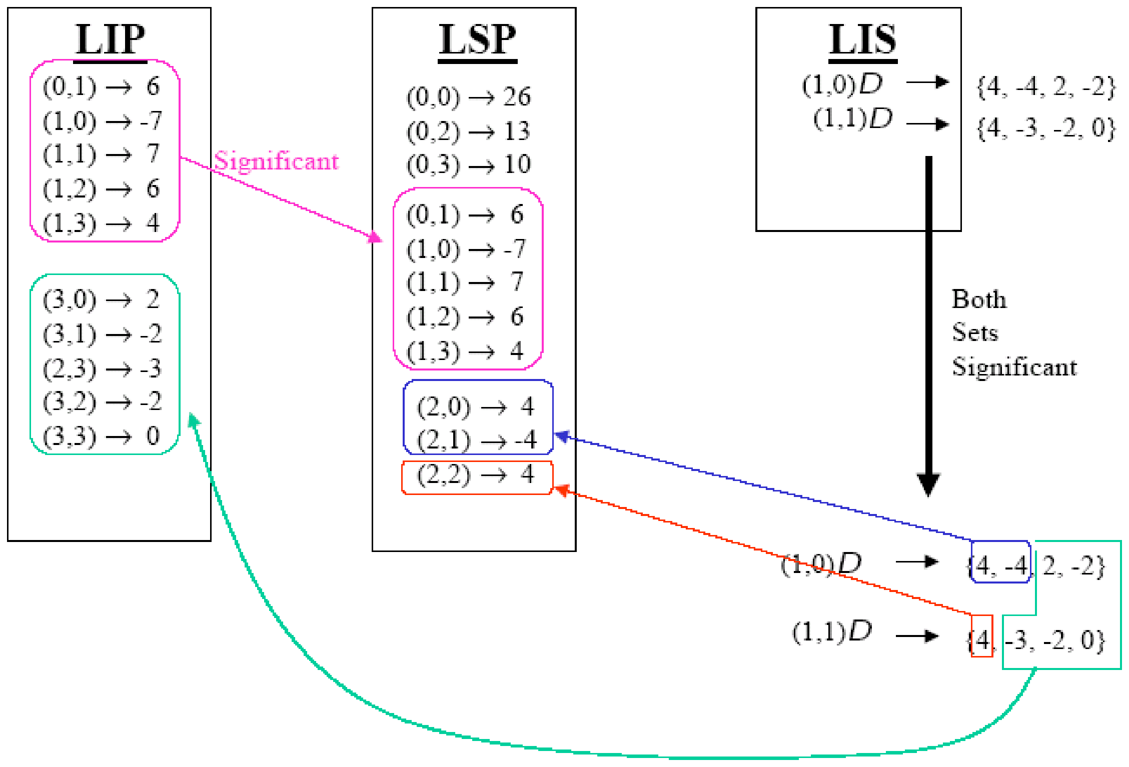


**After second Sorting Pass:**



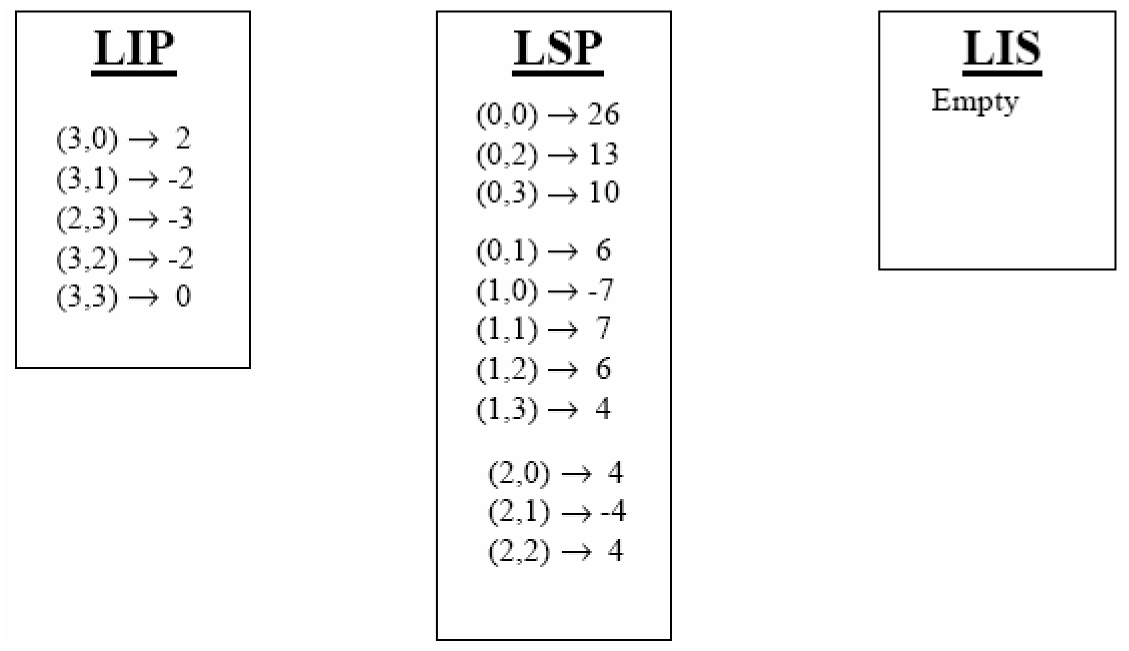
**4.Iteration 3**
**During Third Sorting Pass:**

**n=2;Threshold=4**



**After Third Sorting Pass: (Final Result)**

**Threshold=4**



**LIP**

(3,0) → 2
(3,1) → -2
(2,3) → -3
(3,2) → -2
(3,3) → 0

**LSP**

(0,0) → 26
(0,2) → 13
(0,3) → 10

(0,1) → 6
(1,0) → -7
(1,1) → 7
(1,2) → 6
(1,3) → 4

(2,0) → 4
(2,1) → -4
(2,2) → 4

**LIS**

Empty

# Chapter 6:
**Software Implementation of Wavelet transform based embedded Image Codec**

This chapter provides the implementation details of the Wavelet transform based embedded Image Codec. This software is implemented in **MATLAB6p5**, which performs the efficient and fast compression of gray images.

## 6.1 Implementation Overview
The block diagram of the Image codec implemented in this thesis is as follows:

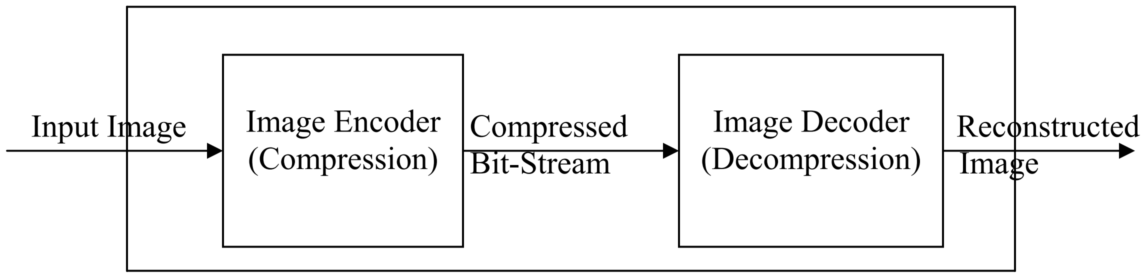


**Figure 5.1** Block diagram of an Image Codec

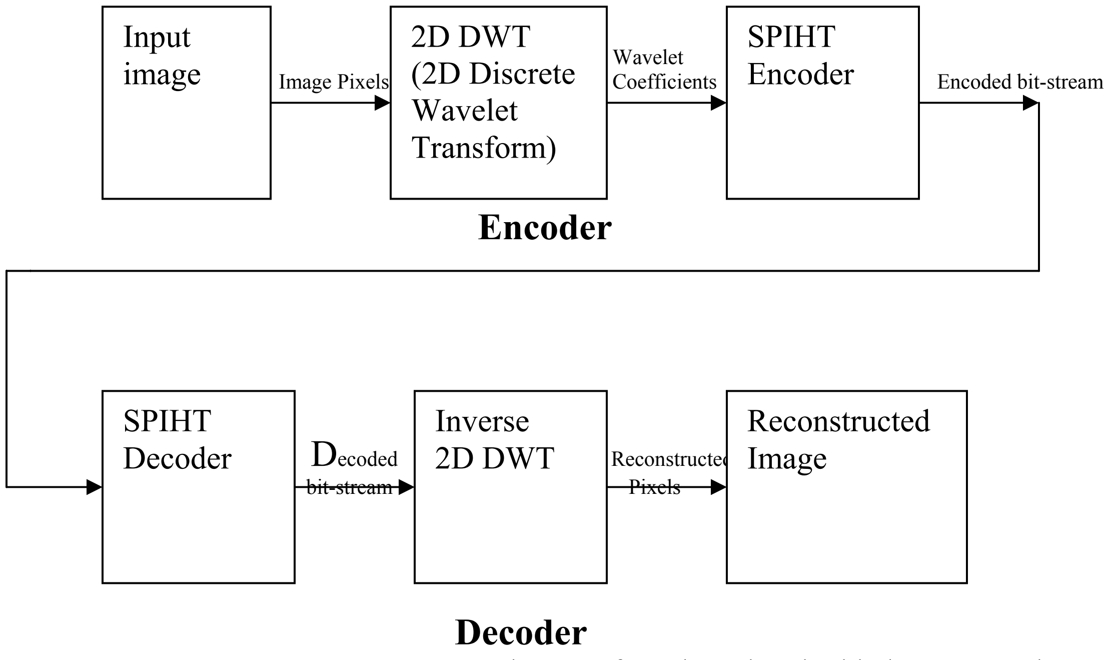


**Figure 6.2** Wavelet transform based embedded Image Codec

The image compression and decompression toolbox developed here consists of following parts:

**A) Image Encoder**: This is the first part of Image Codec, which performs the image compression and consists of following parts:

**A**.**1** Routine for reading raw square (NxN) gray images.

**A**.**2** Routine for performing multilevel 2D DWT (Discrete Wavelet transform) decomposition of image. This routine decomposes the input image at the desired level of decomposition and generates the wavelet coefficients cA, cH, cV, cD for the corresponding image. Max number of decomposition level $n = \lfloor \log_2 N \rfloor$; where NxN is the size of image. 2D DWT of the image at various level of decomposition generates the octave band structure and the strength of the octave-band scheme is that it is simple and it has a rather low complexity while still being effective.

The wavelet coefficients generated depends upon the type of filters used in DWT. In the DWT the various WAVELET FILTERS [6,7,11] used for decomposition are Haar, Daubechies (db2, db3, db4, db5, db6, db7, db8, db9, db10), symlets and the bi-orthogonal 4.4filter (also referred to as the Daubechies 9/7 tap filter). These filters are applied in successive dimensions as described in Chapter 3. Haar, Daubechies and symlets are derived from an orthogonal wavelet basis. The bi-orthogonal 4.4 filter is derived from a bi-orthogonal wavelet basis [9]. The corresponding wavelets show symmetry and this filter is therefore a linear phase filter, which allows for a non-expansive symmetric extension transform to be constructed (Haar wavelet has also this property). This non-expansive transform is crucial since SPIHT is used to code the transformed structure.

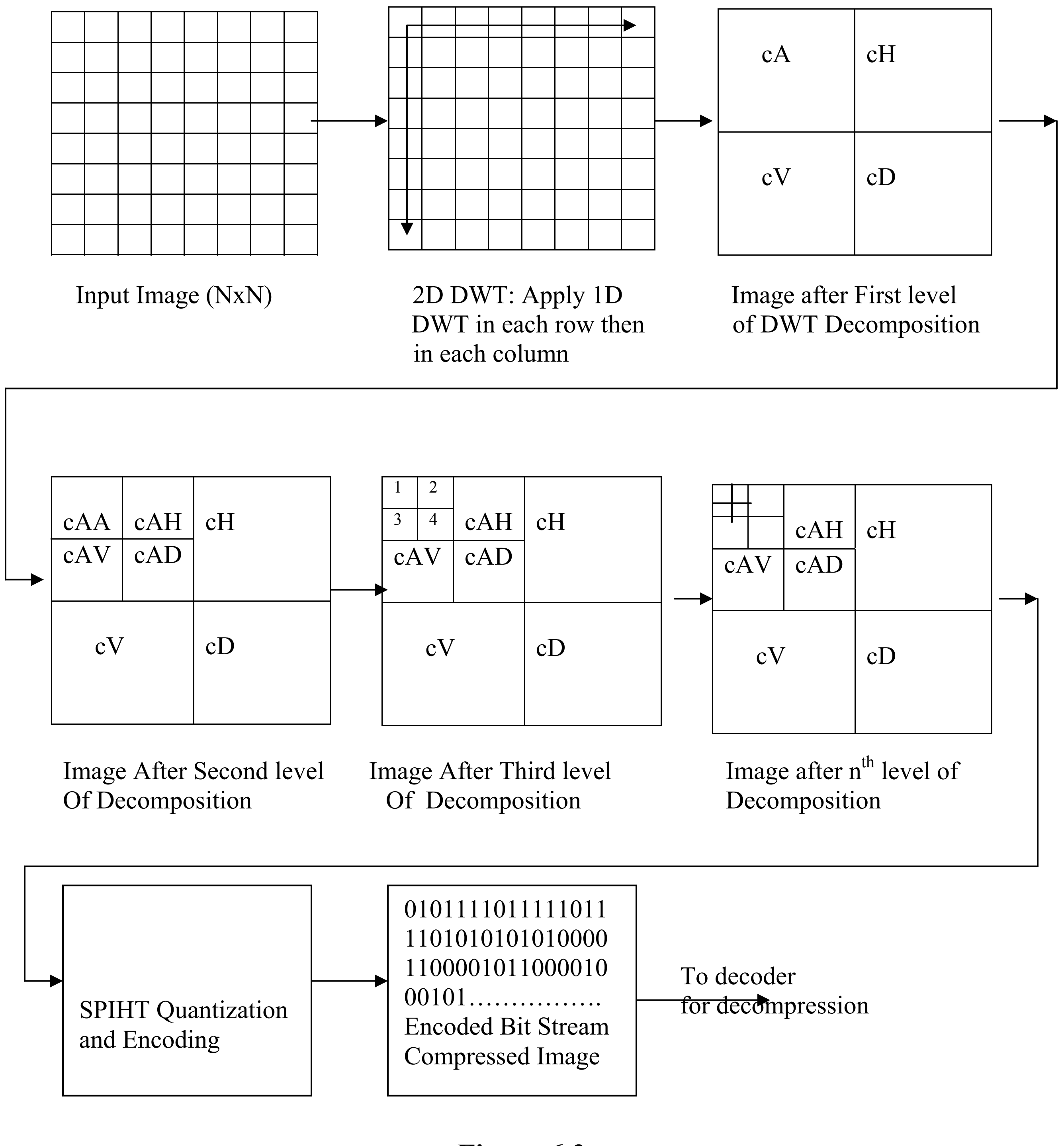
**A.3** Routine for SPIHT Quantization and Encoding which performs qunatization and encoding of wavelet coefficients obtained in previous step **A.2** generates the encoded bit stream which is used by the SPIHT Decoder in Decoder of Image to reconstruct the image. SPIHT [3] algorithm used for quantization and coding of the wavelet coefficients is a high performance coder with the property of producing embedded bit-streams. It is also simple and intuitive and utilizes inter-band dependencies without the use of specially tuned entropy coders based on contexts. SPIHT [3] assumes that the decomposition structure fed to it is an octave-band structure as obtained in step A.2.

**B) Image Decoder:** This is the Second part of Image Codec, which performs the image decompression and consists of following parts:

**B.1** Routine for SPIHT Decoder, which decodes the received encoded bit-stream in to its corresponding wavelet coefficients.

**B.2** Routine for 2D multilevel IDWT (Inverse Discrete Wavelet transform), whose inputs are wavelet coefficients obtained in step **B.1**, which reconstruct the image at the decoder end. For IDWT same filter are used as in DWT.

**6.1.1 Steps involved in image compression (Encoder):**



cA
cH
cV
cD
Input Image (NxN)
2D DWT: Apply 1D DWT in each row then in each column
Image after First level of DWT Decomposition
cAA
cAH
cAV
cAD
cH
cV
cD
1
2
3
4
Image After Second level Of Decomposition
Image After Third level Of Decomposition
Image after nth level of Decomposition
SPIHT Quantization and Encoding
0101111011111011
1101010101010000
1100001011000010
00101…………….
Encoded Bit Stream
Compressed Image
To decoder for decompression

**Figure 6.3**

## 6.2 Implementation Details:

All the function and routines are developed in MATLAB6P5.

### 6.2.1 Wavelet Transform based Embedded Image Encoder and Decoder Functions

**a) Reading the input image:**

**a.1 function Result = funRead(filename, nSize, nRow, nColumn)**

This function reads a RAW format gray scale image from disk.

***Input to function:***

filename: input file
nSize : size of the output image
nRow: row of the output image
nColumn: column of the output image

***Output of function:***

Result: output data in matrix format

**a.2 For reading the standard format (TIF,GIF,JPG,PNG ,BMP etc)input images**

Following inbuilt MATLAB functions can be used:
[im]=imread ('input file name');
Or imshow ('input file name'), [im]=getimage;

**b) 2D Multilevel Wavelet Transform:**

To perform the 2D multilevel wavelet decomposition of the input image following function are designed and used:

**b.1 function [IW , S] = funDWT(I, level, Lo_D, Hi_D);**

This function performs the 2D Wavelet decomposition and uses the function
[c,s] = funwavedec2(x,n,varargin)

***Input to the function:***

I: input image
level: wavelet decomposition level
Lo_D: low-pass decomposition filter
Hi_D: high-pass decomposition filter

***Output of the function:***

IW: decomposed image vector
S: corresponding bookkeeping matrix

**b.2 function [c,s] = funwavedec2(x,n,varargin)**

This is multilevel 2-D wavelet decomposition function used by funDWT( ).

[C,S] = funwavedec2(X,N,'wname') returns the wavelet decomposition of the matrix X at level N, using the wavelet named in string 'wname ' (Pl refer MATLAB function WFILTERS),Outputs are the decomposition vector C and the corresponding bookkeeping matrix S. N must be a strictly positive integer .

*Instead of giving the wavelet name, one can give the filters.*

For [C,S] = funwavedec2 (X, N, Lo_D, Hi_D ),Lo_D is the decomposition low-pass filter and Hi_D is the decomposition high-pass filter.

The output wavelet 2-D decomposition structure [C,S] contains the wavelet decomposition vector C and the corresponding bookkeeping matrix S.

**Vector C is organized as:**
C = [ A(N) | H(N) | V(N) | D(N) | ... H(N-1) | V(N-1) | D(N-1) | ... | H(1) | V(1) | D(1) ].
where A, H, V, D, are row vectors such that: A = approximation coefficients,
H = horizontal. detail coefficients, V = vertical detail coefficients, D = diagonal detail coefficients and each vector is the vector column-wise storage of a matrix.

**Matrix S is such that:**
S(1,:) = size of approximation coefficients(N)
S(i,:) = size of detail coefficients (N-i+2) for i = 2,...,N+1 and
S(N+2,:) = size(X).

**c) SPIHT Encoding:**
**function out = SPIHTEnc (m, max_bits, block_size, level)**

This function performs SPIHT quantization and Encoding; at the specified level of decomposition for the given rate i.e. max no of bits provided for image at a particular compression ratio defined by bit rate; and generates the embedded bit stream.

## Input to the function:

**m:** input image in wavelet domain i.e. IW output of funDWT.
**max_bits:** maximum bits can be used;
max_bits = floor (rate * OrigSize^2) where rate is bit rate in bits per pixel
(i.e. bpp=0.1,0.2,0.3,….1.0 )which specifies the compression ratio.
**block size**: image size (NxN)
**level:** wavelet decomposition level;
level=$n = \lfloor \log_2 N \rfloor$

## Output of the function:

**out:** bit stream

**d) SPIHT Decoding:**
**function m = SPIHTDec (in)**

This function decodes the encoded bit stream obtained from function SPIHTEnc( ) and generates its corresponding wavelet coefficients i.e. reconstructs the output image in wavelet domain.

## Input to the function:

in : bit stream i.e. output of SPIHTEnc( ).

***Output of the function:***

m : reconstructed image in wavelet domain
For the initialization of this function image size, number of bit plane, wavelet decomposition level should be written as bit stream header.

**e) Inverse Wavelet transforms:**
**e.1 function imrec = InvDWT (I_W, S, Lo_R, Hi_R, level)**
This function performs the inverse wavelet decomposition and reconstructs the original image.

***Input to the function:***
IW: decomposed image vector
S: corresponding bookkeeping matrix
Lo_D: low-pass decomposition filter
Hi_D: high-pass decomposition filter
level: wavelet decomposition level

***Output of the function:***
imrec : reconstructed image
***e.2*** *To perform the multilevel 2D inverse DWT the;* **InvDWT( )** *function uses the following functions:*
**function x = funwaverec2(c ,s, varargin )**
This function performs a multilevel 2-D wavelet reconstruction using either a specific wavelet ('wname') or specific reconstruction filters (Lo_R and Hi_R).
For example X = funwaverec2(C, S, 'wname') reconstructs the matrix X based on the multi-level wavelet decomposition structure [C, S ] .
For example X = funwaverec2(C, S, Lo_R, Hi_R), Lo_R is the reconstruction low-pass filter and
Hi_R is the reconstruction high-pass filter.

***e.3*** *The function funwaverec2( ) uses the following function:*
**function a = funappcoef2(c, s ,varargin)**
This function Extract 2-D approximation coefficients and computes the approximation coefficients of a two-dimensional signal.
For example A = funappcoef2(C, S, 'wname', N) computes the approximation coefficients at level N using the wavelet decomposition structure [C,S],'wname' is a string containing the wavelet name, Level N must be an integer such that 0 <= N <= size(S,1)-2 and A = funappcoef2(C, S, 'wname') extracts the approximation coefficients at the last level size(S,1)-2.
Instead of giving the wavelet name, one can give the filters:
For example A = funappcoef2(C, S, Lo_R, Hi_R) or
A = funappcoef2(C, S, Lo_R, Hi_R, N),where Lo_R is the reconstruction low-pass filter and Hi_R is the reconstruction high-pass filter.
**f) Main Function:**
**function [MSE, RMSE, PSNR, SNR, imgSPIHT]=SPIHTMain (file, rate, filtertype )**
This is the main function of the image codec, which performs the compression, and decompression of the input image. It is used for the evaluation of performance metric(PSNR,SNR,RMSE,MSE) of the designed codec at the specified bit rate i.e. bits per pixel.
***Input to the function:***
File: Input NxN image
Rate: bits per pixel e.g. 0.1,0.2,0.3…1.0
Filtertype: Wavelet filter applied for the decomposition e.g. bior4.4, haar, daub9/7, daub5/3 etc
***Output of the function:***

ImgSPIHT: Output image matrix

MSE: Mean Squared Error
RMSE: Root mean squared error
PSNR: Peak Signal to noise ratio
SNR: Signal to noise ratio
This program also plots the original image, image after multilevel decomposition at a specified level ,Output of SPIHT decoder and reconstructed image. Values obtained here will also be used to plot the various performance measurement graphs e.g. bpp Vs MSE, bpp Vs PSNR etc.
Various wavelet filter coefficients used in this thesis for performance evaluation are given in Appendix ‘A’.
Results and performance evaluation of the Wavelet Transform based Embedded Image Codec is discussed in next chapter 7.

# Chapter 7:
# Results, Performance Evaluation and Conclusion

This chapter discusses the results for the Wavelet transform based Embedded Image Codec which uses SPIHT [3] based quantization and coding. Performance of the designed codec is evaluated by performing series of simulations on different images for different wavelet filters [6] with and without the use of lifting scheme [15] at different decomposition levels and at various compression ratios. Various performance measurement metrics are discussed in chapter 2. Low values of MSE and RMSE are desirable. High values of PSNR and SNR are desirable.

## 7.1 Test Settings:

- All images have been chosen so that the dimensions are powers of two. This gives the largest possible number of levels in the decomposition (the largest number of levels is the number of times the original data dimensions can be divided by two with integer result.
  Max no. of decomposition levels=$\lfloor \log_2 N \rfloor$;where NxN is the size of image.

- As a compression measurement Bits Per Pixel (BPP) or Bit Rate is chosen. This compression measurement is dependant of the number of bits used to represent the original signal (image) samples.
  Bit Rate for gray image=BPP(bits per pixel)=8/CR; Where CR is the compression ratio.
  Maximum bits that can be used is defined as:
  Max bits = floor (rate * N*N) ;where rate is bit rate in bits per pixel
  (i.e. BPP=0.1,0.2,0.3,….1.0 )which specifies the compression ratio.

- Various orthogonal and bi-orthogonal [9] Wavelet filters [6] are chosen for decomposing the image and evaluating the performance. For Example Haar, db2, db5, Symlets and Bior4.4. A lifting implementation of CDF(9,7) [9]wavelet filter is also presented. The Low pass and High pass filter values, for analysis (decomposition) and synthesis (reconstruction), of these filters are provided in chapter6.

## 7.2 Results: Without using lifting scheme for wavelet filters

### 7.2.1 Results for 512x512 gray image Lena.jpg for the different wavelet filters

#### A) Wavelet Filter applied 'Haar'

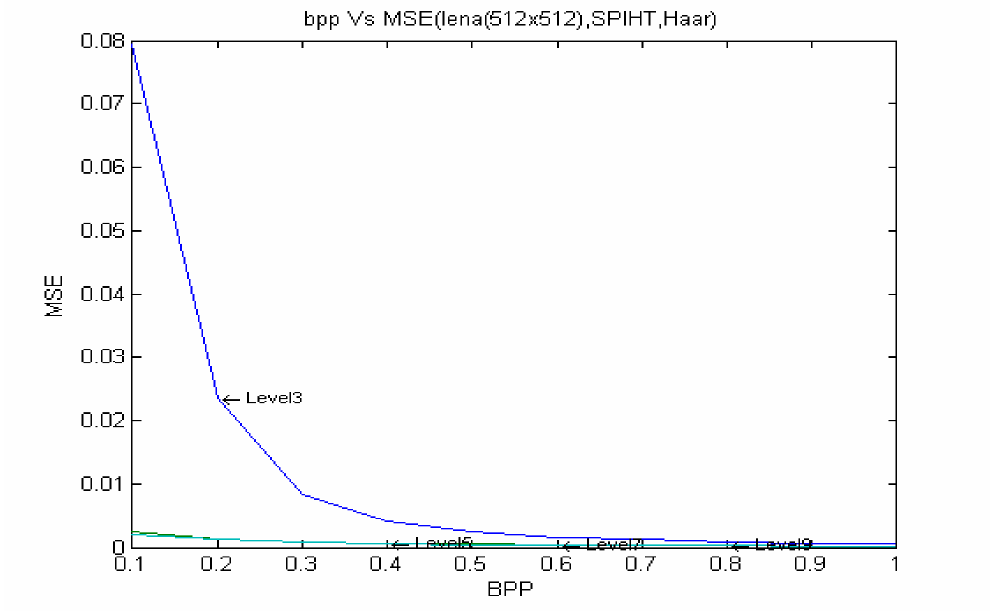


**Figure 7.1** BPP Vs MSE Graph, 512x512,Lena.jpg image at different level of decompositions, filter applied Haar

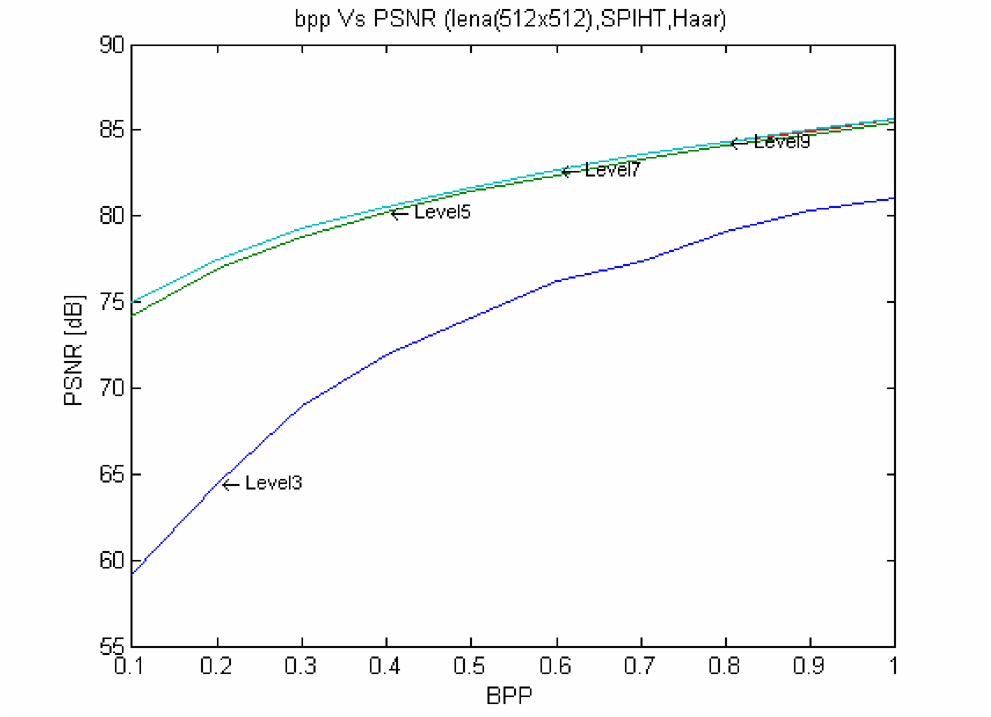


**Figure 7.2** BPP Vs PSNR Graph, 512x512,Lena.jpg image at different level of decompositions, filter applied Haar

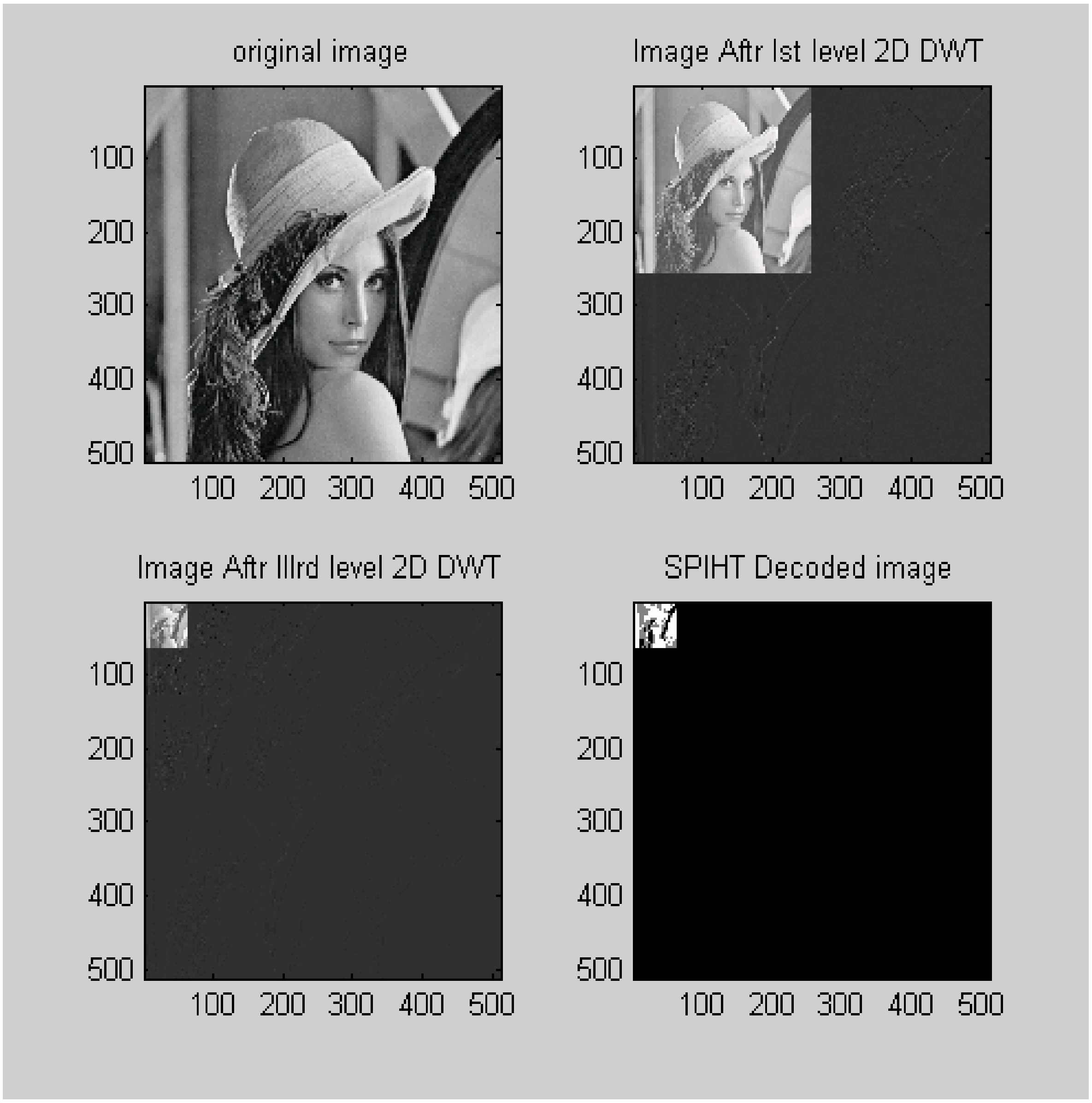


**Figure 7.3:** Output of SPIHT coder for Lena image at third level of decomposition ,filter applied Haar

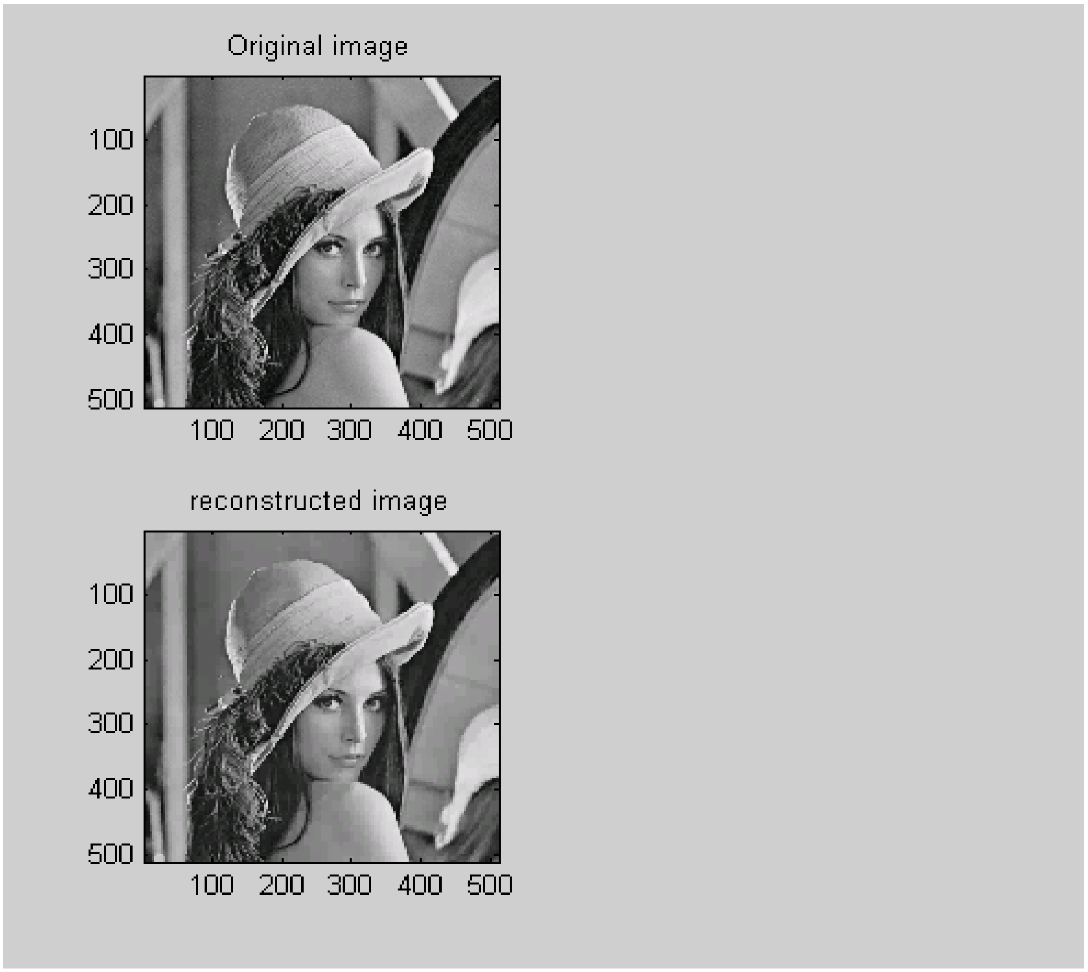


**Figure 7.4:** Input and Output of SPIHT based image codec for Lena image at third level of decomposition, filter applied Haar

## B) Wavelet Filter applied Daubechies 'db5'

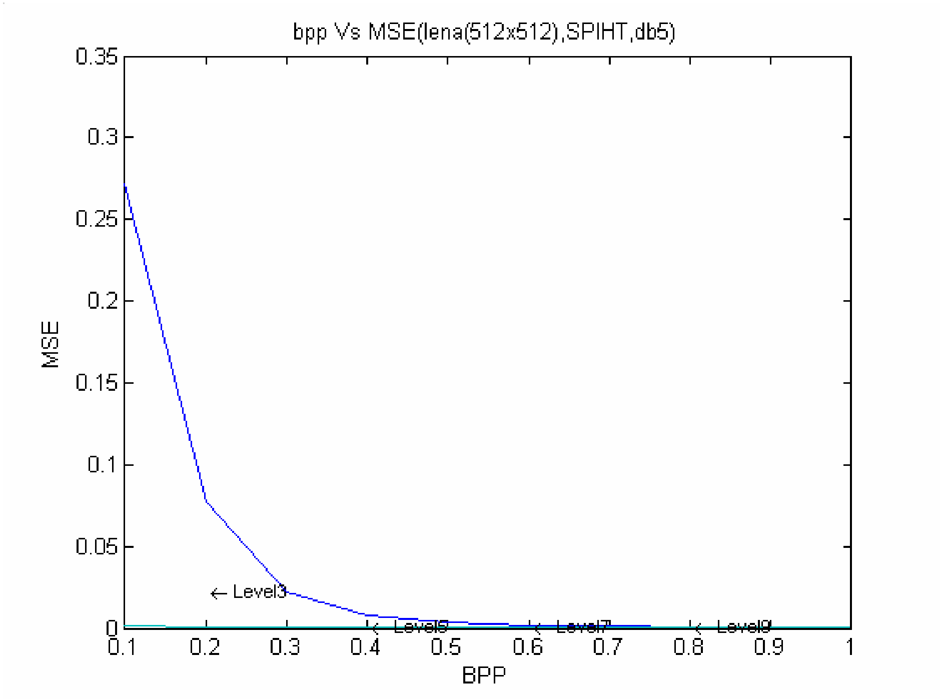


**Figure 7.5 (a)** BPP Vs MSE graph of Lena.jpg for Daubechies wavelet filter db5 at different level of decompositions.

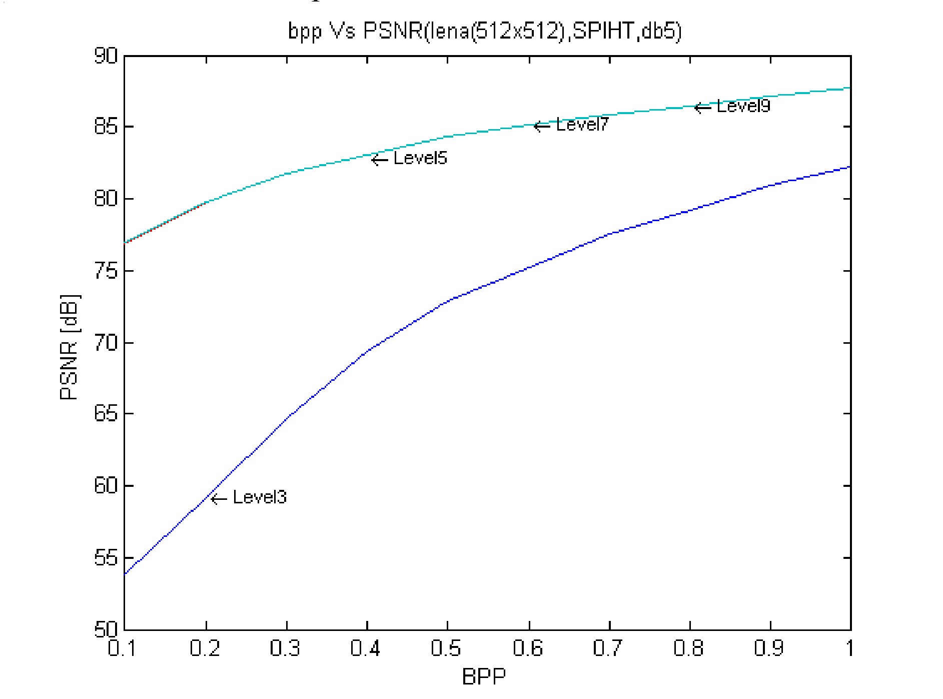


**Figure 7.5 (b)** BPP Vs PSNR graph of Lena.jpg for wavelet filter db5 at different level of decompositions.

**B) Wavelet Filter applied 'Biorthogonal (CDF 9/7)', bior4.4**

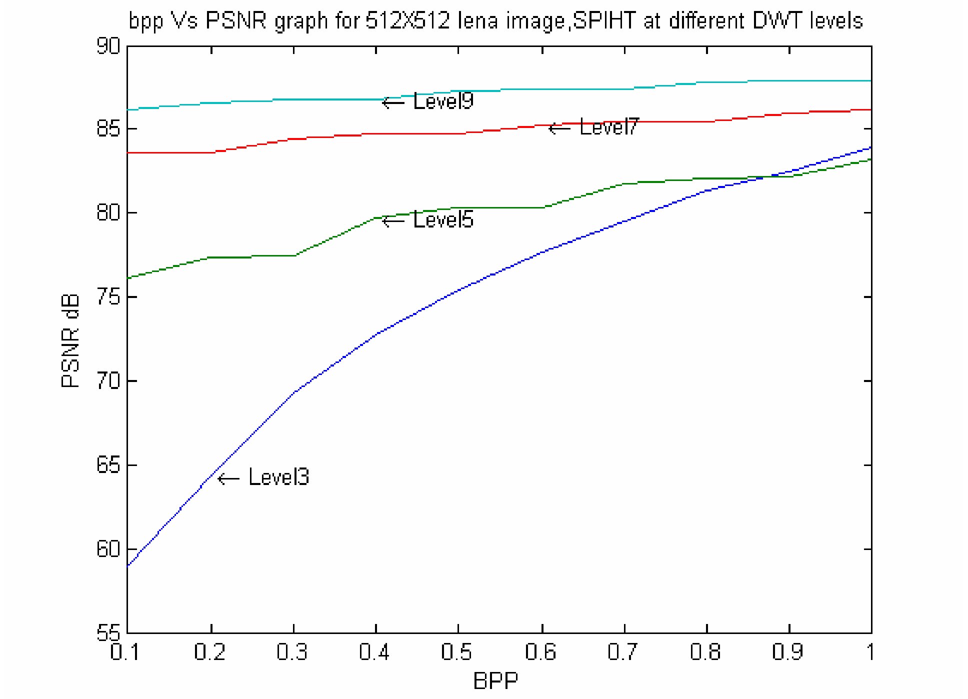


**Figure7.6**: BPP Vs MSE and BPP Vs PSNR graph of Lena.jpg for wavelet filter BIOR4.4 at max level of decomposition, L=9

**C) Performance comparison of different wavelet filters for the image (512x512) Lena.jpg at max level of decomposition, 9**

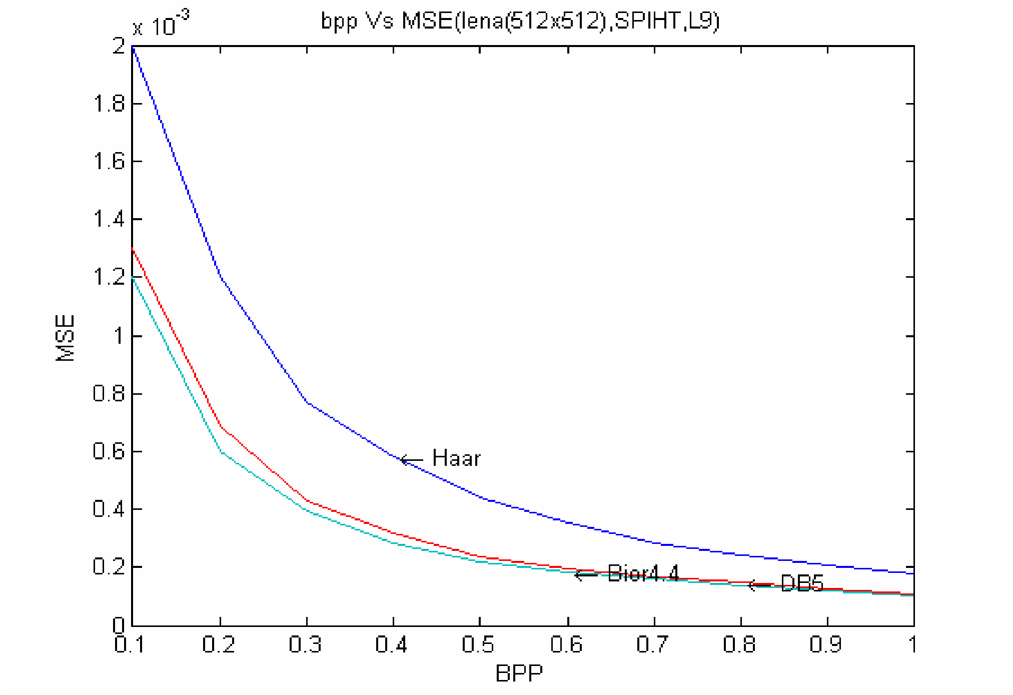


**Figure7.7:** BPP Vs MSE graph for Lena.jpg for different filters at max level of decomposition,9.

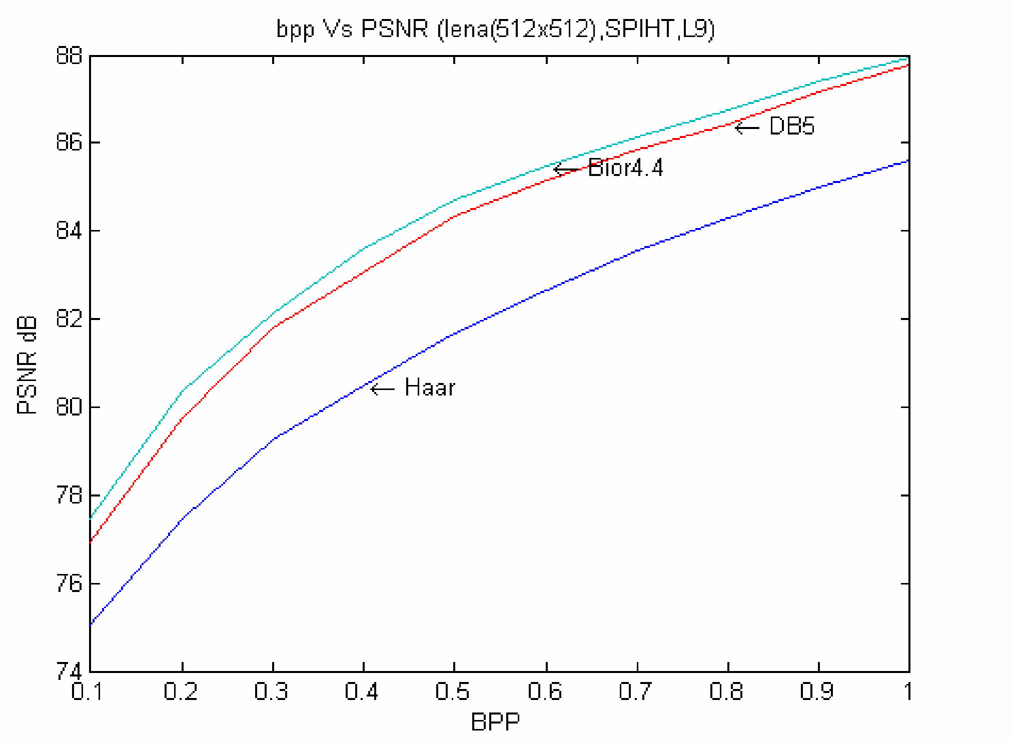


**Figure7.8:** BPP Vs PSNR graph for Lena.jpg for different filters at max level of decomposition,9.

### 7.2.2 Results for 512x512 gray raw image 'Lena.raw' for the different wavelet filters

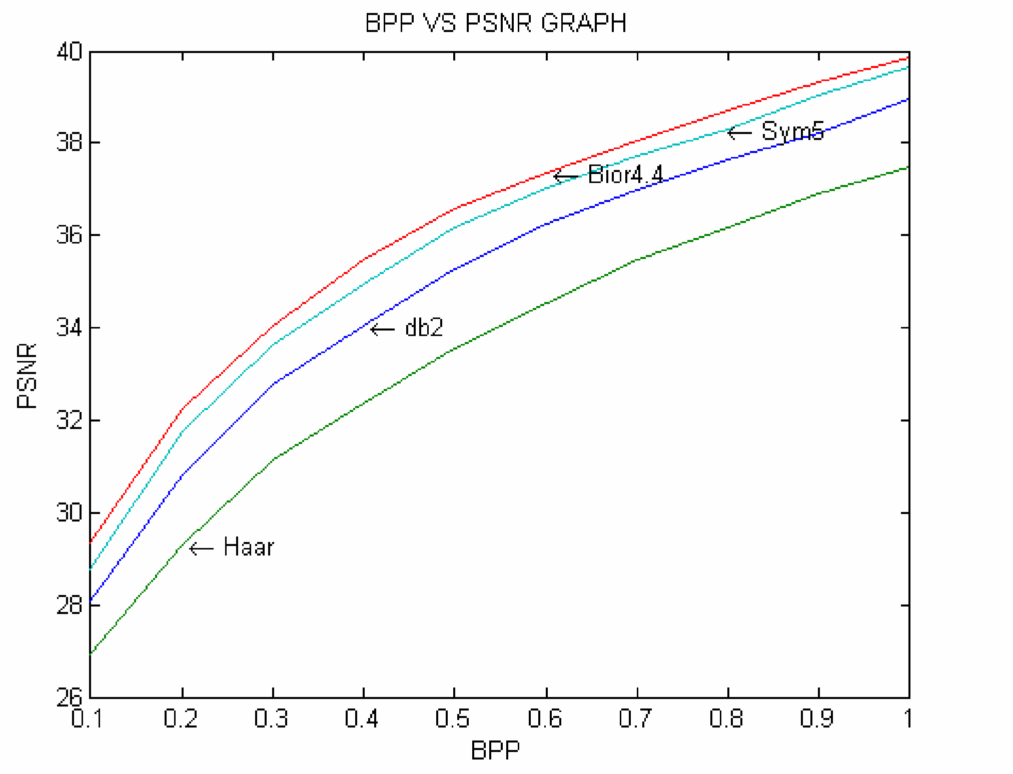


**Figure 7.9:** BPP Vs PSNR graph for Lena.raw (512x512) for different wavelet filters

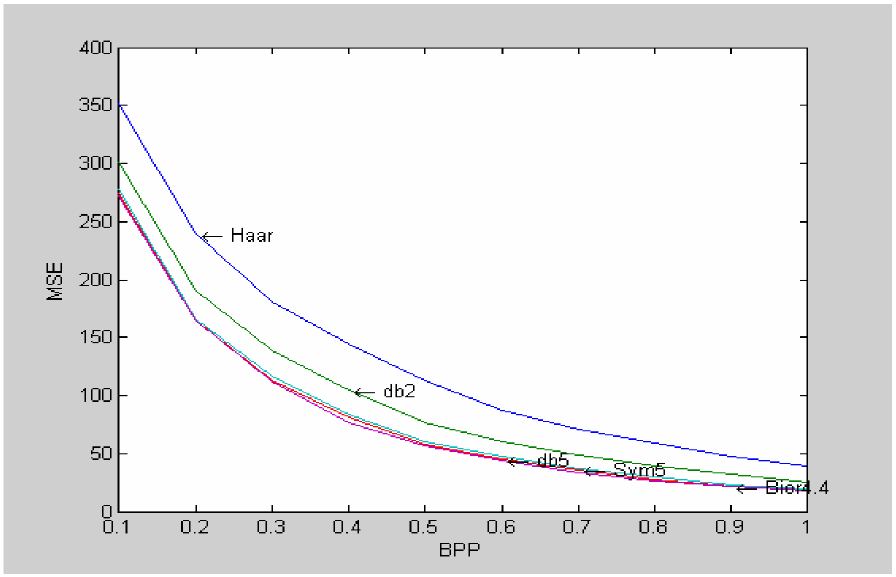


**Figure 7.10:** BPP Vs MSE graph for image Barbara.raw for different wavelet filters at max level of decomposition, 9.

### 7.2.3 Results for 512x512 gray raw image 'Brabara.raw' for the different wavelet filters

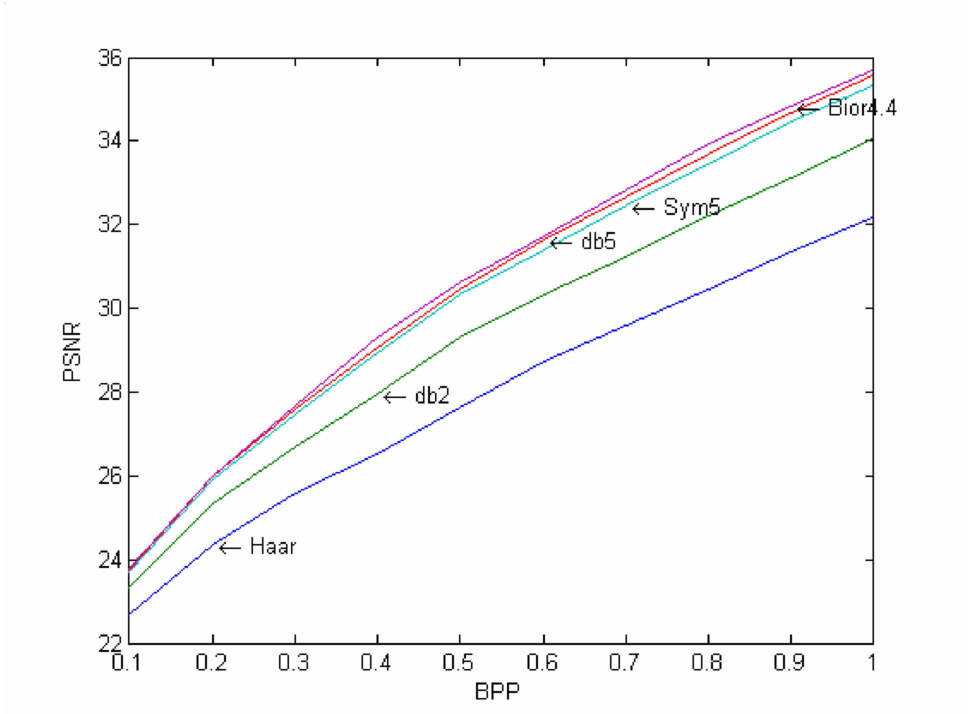


**Figure 7.11** BPP Vs PSNR graph for image Barbara.raw for different wavelet filters at max level of decomposition, 9.

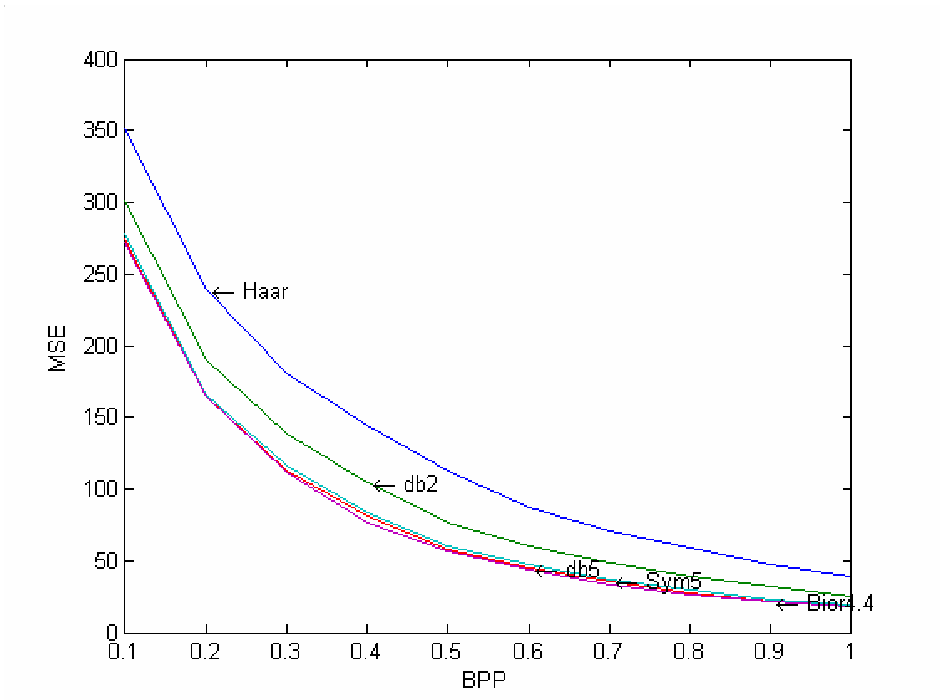


**Figure 7.12:** BPP Vs MSE graph for gray raw image Barbara.raw for different wavelet filters at max level of decomposition, 9.

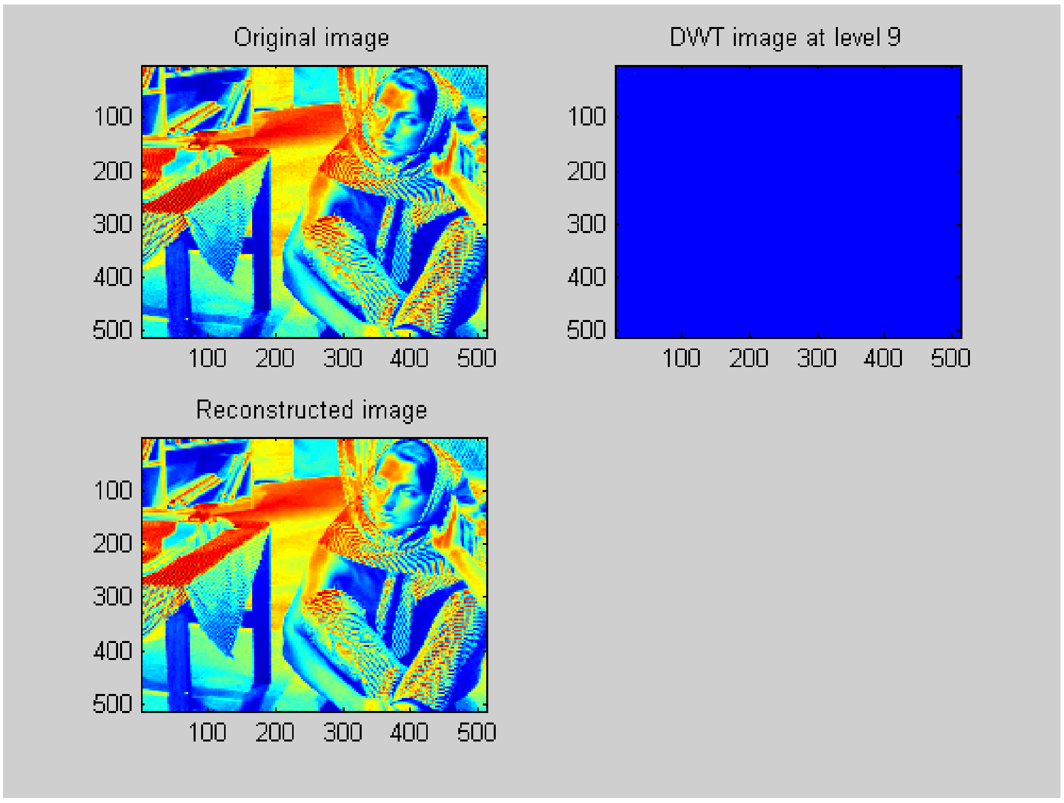


**Figure7.13:** Output of SPIHT based image codec at max level of decomposition,9, filter applied BIOR4.4

**7.3 Results: Using lifting scheme for wavelet filters**

This section discusses the results and performance analysis of the designed image codec where lifting scheme [15] is used to derive the wavelet filters [9,15] for wavelet transform of the image. A, lifting version of bi-orthogonal filter CDF (9,7) (i.e.bior4.4) [9] is implemented and an analysis is performed. The function inputs are an image X and a number N. The output is the N-stage forward transform of X. While many wavelet packages use periodic boundary handling, this function uses symmetric boundary handling. As images are generally not periodic, this is more appropriate and has superior image compression capabilities.

**7.3.1 Comparison of Results for 512x512 image lena.png, using lifting scheme**

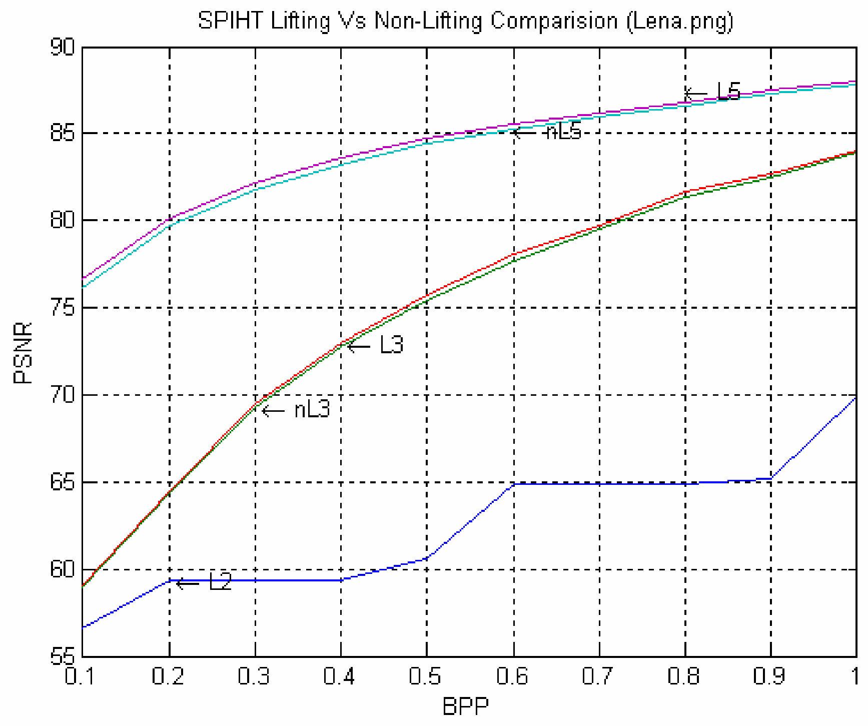


**Figure:7.13( a)** BPP Vs PSNR graph (Comparison of lifting and non-lifting scheme version)

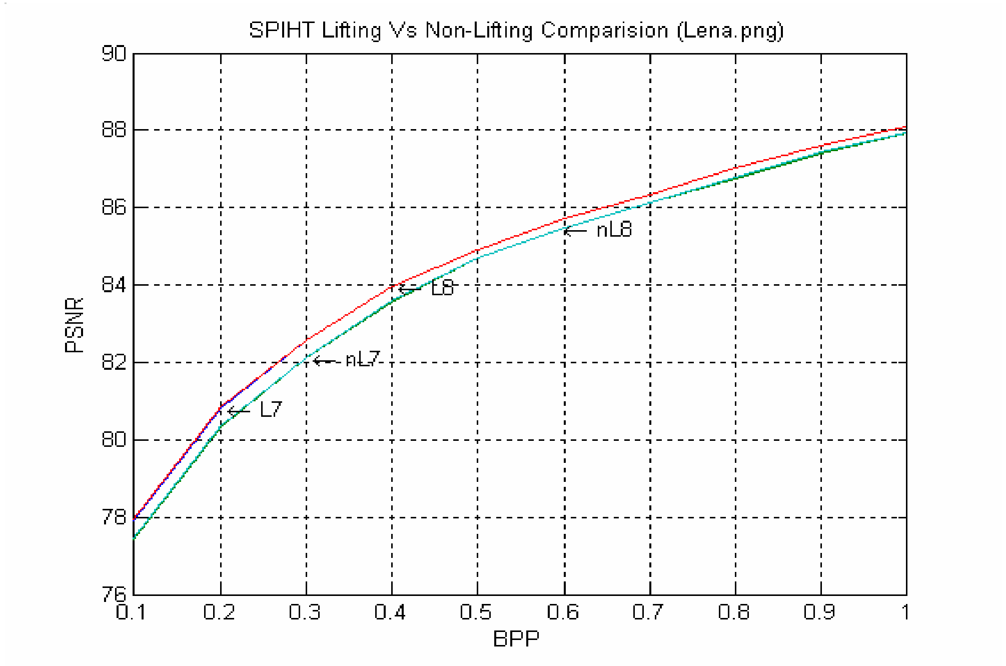


**Figure:7.13( b)** BPP Vs PSNR graph (Comparison of lifting and non-lifting scheme at various level of decompositions for 512x512 image, L –Level of decomposition for lifting version and nL level of decomposition for non lifting version

## 7.3.2 Comparison of Results for 128x128 image 6.tif, using lifting scheme

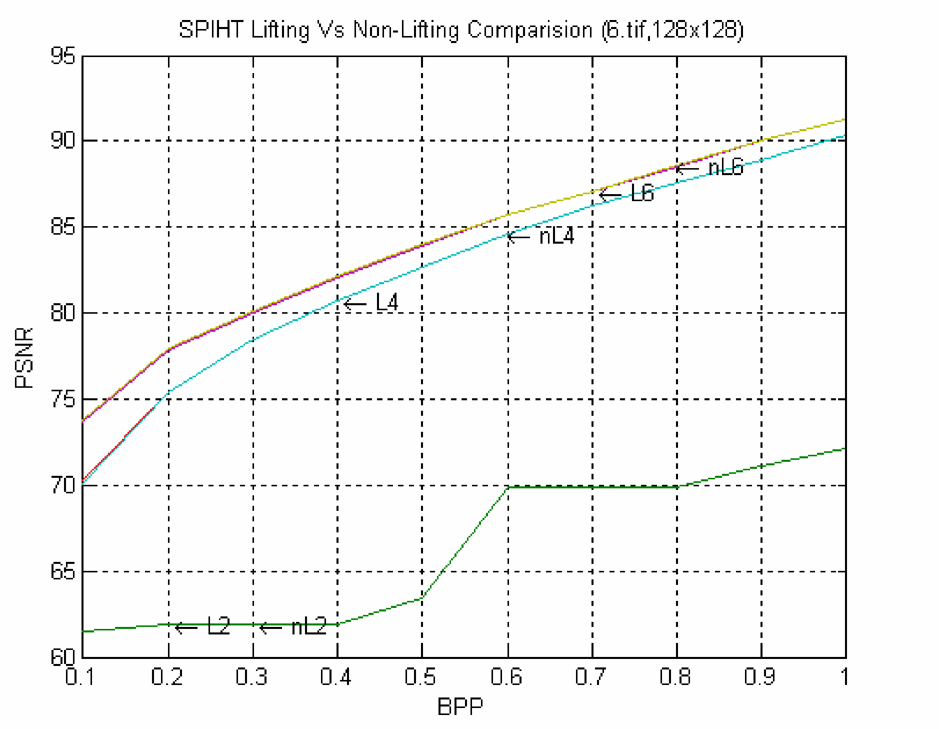


**Figure:7.14 (a)** BPP Vs PSNR graph (Comparison of lifting and non-lifting scheme at various level of decompositions for 128x128 image)

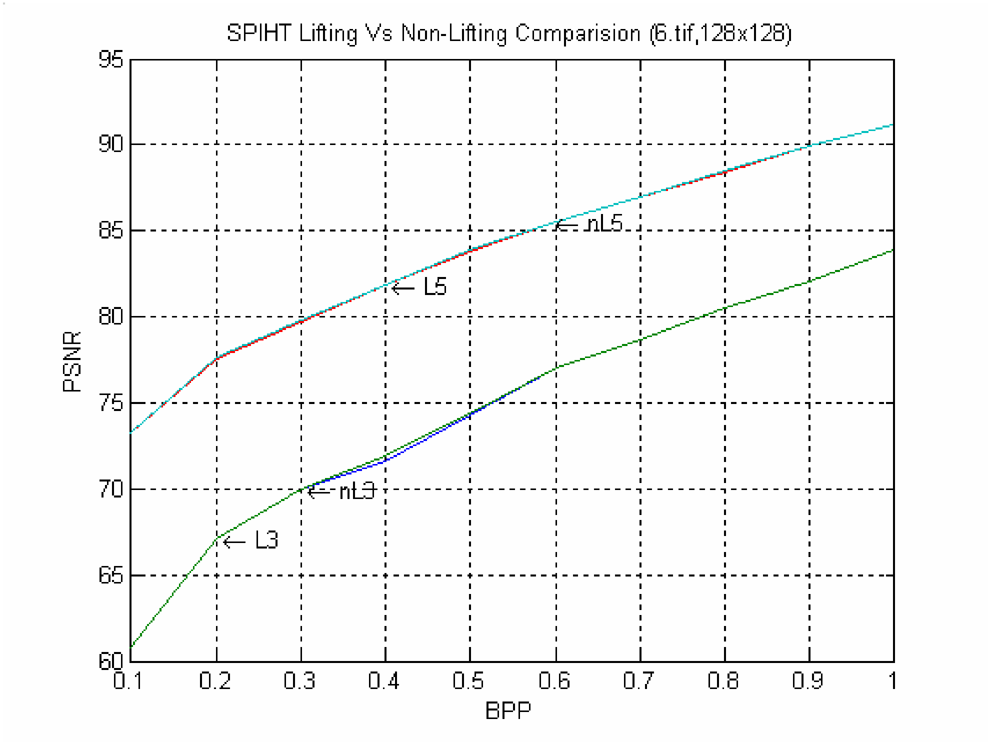


**Figure:7.14 (b)** BPP Vs PSNR graph (Comparison of lifting and non-lifting scheme at various level of decompositions for 128x128 image)

## 7.4 Performance analysis:

Simulation Results for the performance evaluation of "DWT based embedded image codec" for different wavelet filters at different level of decompositions using lifting and non-lifting techniques are included in **Appendix-B**.

### 7.4.1 <u>For Non-lifting version implementation of wavelet filters</u>

- **Effect of total number of wavelet decompositions:**

Figure 7.1 is a graph between BPP (bits per pixel) and MSE, whereas Figure 7.2 is a graph between BPP and PSNR at different level of decompositions of image using Haar Wavelet. Figure 7.5 is a graph between BPP (bits per pixel) Vs MSE, and BPP Vs PSNR at different level of decompositions of image using Daubechies Wavelet (db5) for the image lena.jpg .

In all these graphs, at level 3 of decomposition the value of PSNR increases with the increase of BPP value. Change in the value of PSNR at levels 5,7 and 9 are nearly same. There is significant difference of PSNR values at level 3 and levels5, 7,9.Hence it can be concluded that a good performance can be achieved at level 5 without decomposing the image at further levels thereby computational complexity can be reduced and system resources can be utilized efficiently.

- **Effect of bit rate (compression ratio) on MSE and PSNR values for theSPIHT based Coder:** As it can be seen from the various graphs that PSNR increases with bit rate but a good compression of 80% can be achieved at a bit rate of 0.1 with acceptable PSNR value. For the lena.jpg image the PSNR values for different wavelet filters [6] at 80% compression ratio are: 75.0566 dB for Haar wavelet, 76.9106 dB for db5, 77.4527 dB for bior4.4 wavelet filter which are far better. For the lena.raw image the PSNR values for different wavelet filters [6] at 80% compression ratio are : 26.929 dB for Haar wavelet, 28.077 dB for db2, 28.769 dB for symlet5, 29.32 dB for bi-orthogonal wavelet bior4.4.For the barabra.raw image the PSNR values for different wavelet filters [6] at 80% compression ratio are: 22.672 dB for Haar wavelet, 23.344 dB for db2 wavelet, 23.748 dB for db5, 23.686 dB for sym5 and 23.786 dB for bior4.4.The PSNR values obtained for Lena and Barbara raw gray images are far better as found in literatures that uses another quantization and coding method. Hence, with the help of SPIHT based coder better result can be obtained.

- **Effect of various wavelet filters on the performance:** Figure 7.9,7.11 provides the comparative performance of various wavelet filters [6] Haar, db2, symlets5, bior4.4.The performance order achieved is Haar<db2<db5<symlets5<bior4.4 i.e. again bi-orthogonal wavelet filters CDF(9,7) [9] are more better but symlet family of wavelets [6] are not more behind. As, it can be seen from the various graphs and values listed that Bi-orthogonal filter bior4.4 is far better than the other wavelet filters though good results can also be achieved with the help of Daubechies family of wavelets, Symlet and Haar wavelets. These good performances are achieved at the cost of extra computations. Among all of the wavelet filters Haar is very simple from implementation point of view but it does not provide good results. Hence for those applications where performance is critical and data loss is not acceptable Bior4.4 i.e. CDF(9,7) [9]should be used.

### 7.4.2 Analysis for Lifting version implementation of bi-orthogonal wavelet filter CDF (9,7)

- **Effect of lifting on performance (PSNR) of the codec:**The section 7.3 presents the results and performance analysis of the designed image codec where lifting scheme [15] is used to derive the wavelet filters for wavelet transform of the image. A, lifting version of bi-orthogonal filter CDF (9,7) (i.e.bior4.4) is implemented and an analysis is performed. From graphs of figure 7.13 and 7.14, it can be concluded that the performance of the coder can be further increased by using lifting scheme for the implementation of the wavelet transforms for the images of larger size(512 x512 ,1024x1024 ). Though there is a very little improvement on performance for the images of little sizes e.g. 128x128.Hence lifting scheme improves the performance of the codec for images of larger size.
- **Effect of lifting on speed and computational complexity:** It has been observed through series of simulations on different size of images that images size below 256x256 transform quickly, while larger images take a few seconds. The lifting scheme [15] makes optimal use of similarities between the high and low pass filters to speed up the calculation. In some cases the number of operations can be reduced by a factor of two.
- The lifting scheme [15] allows a fully in-place calculation of the wavelet transform. In other words, no auxiliary memory is needed and the original signal (image) can be replaced with its wavelet transform.

## Conclusion:

Hence at last it can be concluded that the performance of the SPIHT based embedded image codec for the compression and decompression of images are very good and efficient for biorthogonal family of filters that uses lifting scheme specially for bior4.4 i.e. CDF(9,7) which allows perfect reconstruction of image.

SPIHT quantization and encoding technique achieves good performance by exploiting the spatial dependencies of the DWT coefficients in different sub-bands i.e. it takes in to account the intra-band dependencies. Further for the better performance and reducing the computational complexity of the coder the desired number of decomposition levels i.e. wavelet transform levels is found to be approximately abs(N/2)+2 for an image of size NxN (i.e.for 512x512 image the desired level of decomposition is 6).Lifting scheme implementation of the biorthogonal wavelet filter CDF(9,7) improves the performance of the codec for images of larger size as well as speed up the calculation. In some cases the number of operations can be reduced by a factor of two. Further the lifting scheme allows a fully in-place calculation of the wavelet transform. In other words, no auxiliary memory is needed and the original signal (image) can be replaced with its wavelet transform. This codec can be used for those applications where performance is critical and data loss is not acceptable to greater extent.

## Future Work:

The Discrete Wavelet Transform based Embedded Image codec, which uses SPIHT based quantization and coding, developed in this thesis can be used for those applications where performance is critical and data loss is not acceptable to greater extent. This coder works for square 2D Gray images.

The performance of this SPIHT based codec can be more increased by including the arithmetic coding phase after SPIHT [2,3] encoding.

Since this codec performs compression and decompression of only gray 2D images, it can be further extended to perform compression and decompression of color RGB images. This can be done by separately transforming the each R,G and B channels and then applying SPIHT quantization and coding to wavelet coefficients of each R,G and B channels.

Concept of SPIHT quantization and coding can also be extended and applied for the compression and decompression of 3D image stacks e.g. medical images.

The Wavelet based applications can also be developed by using re-configurable computing (Hardware) approach and can be implemented using FPGAs for very complex problems to achieve best results.

At last, I would like to say that Wavelet Theory and applications is a very vast field and it is a rich area of active research in the field of Digital Signal Processing, Statistical signal processing, Image processing and multimedia technology.

**References:**

**1.** Rafael C. Gonzalez, Richard E. Woods (2002),”*Digital Image Processing”*, Pearson Education

**2.** Pankaj N. Topiwala (1998),”*Wavelet Image and Video Compression”*, Kluwer Academic Publishers

**3.** Said A., Pearlman W. A.(1996), *“A new, fast, and efficient image codec based on set portioning in hierarchical trees”* in *IEEE Transactions on circuits and systems for video technology*, Vol. 6, No. 3, pp. 243-250.

**4.** Shapiro J. M. (1993), *“Embedded image coding using zero-trees of wavelet coefficients”* in IEEE Transactions on signal processing, Vol. 41, No. 12, pp. 3445-3462.

**5.** N.F. Chamberlain (2002),”Introduction to Wavelets”, South Dakota School of Mines and Technology

**6.** MATLAB 6p5,Wavelet tool box

**7.** K.P. Soman, K.I. Ramachandran (2004),”*Insight in to Wavelets from Theory to Practice”*, PHI, New Delhi

**8.** I. Daubechies. Ten Lectures on Wavelets. SIAM, Philadelphia PA, first edition, 1992.

**9.** Cohen,A.,I. Daubechies and J. Feauveau,(1992), *”Biorthogonal bases for compactly supported Wavelets”*,Comm. Pure Applied Mathematics,45,485-560.

**10.**An Introduction to Wavelets: http://www.amara.com/IEEEwave/IEEEwavelet.html

**11.** Strang, G. and T.Q. Nguyen,(1998), *”Wavelet and Filter Banks”,* Revised Edition, Wellesley Cambridge Press, Wellesley, MA

**12.** Prof. Mark Fowler,”Wavelet Transform Theory”, Department of Electrical Engineering State University of New York at Binghamton

**13.** Geoffrey M. Davis, Aria Nosratinia, "*Wavelet-based Image Coding: An Overview”,* Applied and Computational Control, Signals, and Circuits, vol. 1, pp. 205--269, 1998

**14.** Zixiang Xiong, Kannan Ramachandran, Michael T. Orchard and Ya-Qin Zhang (1999), *”A comparative study of DCT and Wavelet based image coding”,* IEEE Transactions on Circuit and Systems for Video Technology, Vol 9, No5, August 1999.

**15.**Daubechies, Wim Swelden (1997),”Factoring wavelet transforms in to lifting steps”, Princeton University, NJ and Lucent technologies, NJ.

## Appendix-A

**Various wavelet filters (without lifting scheme) used in this thesis for 2D DWT:**

Where the four output filters for a specific wavelet filter are:

LO_D, the decomposition low-pass filter
HI_D, the decomposition high-pass filter
LO_R, the reconstruction low-pass filter
HI_R, the reconstruction high-pass filter

| Wavelet Filter | LO_D | HI_D | LO_R | HI_R |
|---|---|---|---|---|
| Haar | [0.7071,0.7071] | [-.7071,0.7071] | [0.7071,0.7071] | [0.7071,-.7071] |
| db1 | [0.7071,0.7071] | [0.7071,0.7071] | [0.7071,0.7071] | [0.7071,-.7071] |
| db2 | [-.1294,0.2241,0.8365, 0.4830] | [-.4830,0.8365, -0.2241,-.1294] | [0.4830,0.8365, 0.2241,-0.1294] | [-0.1294,-0.2241, 0.8365,-0.4830] |
| db3 | [0.0352,-0.0854,-.1350, 0.4599, 0.8069,0.3327] | [-0.3327,0.8069, -0.4599,-0.1350, 0.0854,0.0352] | [0.3327,0.8069, 0.4599,-0.1350, -0.0854,0.0352] | [0.0352,0.0854, -0.1350,-0.4599, 0.8069 , -0.3327] |
| db4 | [-0.010597,0.032883, 0.030841,-0.18703, -0.027984,0.63088, 0.71485,0.23038] | [-0.23038,0.71485, -0.63088,-0.027984, 0.18703,0.030841, -0.032883,-0.010597] | [0.23038,0.71485,0.63088,-0.027984,-0.18703,0.030841,0.032883,-0.010597] | [-0.010597,-0.032883,0.030841,0.18703,-0.027984,-0.63088,0.71485,-0.23038] |
| Bior4.4 | [0,0.037828,-0.023849,-0.11062, 0.3774,0.8527,0.3774 ,-0.11062, -0.023849,0.037828] | [0,-0.064539,0.040689, 0.41809,-0.78849 ,0.41809,0.040689, -0.064539,0] | [0,-0.064539, -0.040689, 0.41809,0.78849, 0.41809, -0.040689, -0.064539,0,0] | 0,-0.037828,-0.023849,0.11062 , 0.3774,-0.8527,0.3774, 0.11062,-0.023849,-0.037828 |

Wavelet filters [6] listed here in table are only examples. Other orthogonal filters used for analysis and synthesis include: db5, db6, db7, db8, db9, db10, db45 and symlets.

**Appendix –B**
**Simulation Results for the performance evaluation of "DWT based embedded image codec" for different wavelet filters at different level of decompositions using lifting and non-lifting techniques**

**B.1. Results for different wavelet filters using Non-lifting technique for 'lena.png', 512x512 image**
**B.1.1 Wavelet Filter Applied =Haar**
**No of decomposition levels, L=3**

| BPP | MSE | PSNR[dB] |
|---|---|---|
| 0.1 | 0.0794 | 59.1342 |
| 0.2 | 0.0234 | 64.4300 |
| 0.3 | 0.0083 | 68.9572 |
| 0.4 | 0.0041 | 71.9723 |
| 0.5 | 0.0025 | 74.1361 |
| 0.6 | 0.0016 | 76.2047 |
| 0.7 | 0.0012 | 77.4087 |
| 0.8 | 8.0858e-004 | 79.0536 |
| 0.9 | 6.0736e-004 | 80.2963 |
| 1.0 | 5.0684e-004 | 81.0821 |

**L5:**

| BPP | MSE | PSNR[dB] |
|---|---|---|
| 0.1 | 0.0025 | 74.1655 |
| 0.2 | 0.0013 | 76.9896 |
| 0.3 | 8.6220e-004 | 78.7747 |
| 0.4 | 6.1246e-004 | 80.2601 |
| 0.5 | 4.7017e-004 | 81.4082 |
| 0.6 | 3.7683e-004 | 82.3694 |
| 0.7 | 3.0144e-004 | 83.3388 |
| 0.8 | 2.5159e-004 | 84.1238 |
| 0.9 | 2.1631e-004 | 84.7801 |
| 1.0 | 1.8449e-004 | 85.4712 |

**No 0f decomposition levels ,L=7**

| BPP | MSE | PSNR[dB] |
|---|---|---|
| 0.1 | 0.0021 | 75.0128 |
| 0.2 | 0.0012 | 77.4273 |
| 0.3 | 7.7178e-004 | 79.2559 |
| 0.4 | 5.8185e-004 | 80.4827 |
| 0.5 | 4.4276e-004 | 81.6691 |
| 0.6 | 3.5395e-004 | 82.6414 |
| 0.7 | 2.8669e-004 | 83.5567 |
| 0.8 | 2.4276e-004 | 84.2791 |
| 0.9 | 2.0669e-004 | 84.9777 |
| 1.0 | 1.7868e-004 | 85.6101 |

**L9(max level of decomposition):**

| BPP | MSE | PSNR[dB] |
|---|---|---|
| 0.1 | 0.0020 | 75.0566 |
| 0.2 | 0.0012 | 77.4465 |
| 0.3 | 7.6863e-004 | 79.2736 |
| 0.4 | 5.7974e-004 | 80.4985 |
| 0.5 | 4.4171e-004 | 81.6794 |
| 0.6 | 3.5280e-004 | 82.6555 |
| 0.7 | 2.8588e-004 | 83.5690 |
| 0.8 | 2.4239e-004 | 84.2857 |
| 0.9 | 2.0617e-004 | 84.9884 |
| 1.0 | 1.7844e-004 | 85.6159 |

**B .1.2 Results for 512x512 Lena.jpg for wavelet filter db5:**

**No 0f decomposition levels, L =3**

| BPP | MSE | PSNR[dB] |
|---|---|---|
| 0.1 | 0.2717 | 53.7906 |
| 0.2 | 0.0777 | 59.2270 |
| 0.3 | 0.0220 | 64.6968 |
| 0.4 | 0.0075 | 69.3878 |
| 0.5 | 0.0033 | 72.8852 |
| 0.6 | 0.0020 | 75.1821 |
| 0.7 | 0.0012 | 77.5155 |
| 0.8 | 7.9048e-004 | 79.1519 |
| 0.9 | 5.2092e-004 | 80.9631 |
| 1.0 | 3.9010e-004 | 82.2190 |

**L=5**

| BPP | MSE | PSNR[dB] |
|---|---|---|
| 0.1 | 0.0017 | 75.8492 |
| 0.2 | 7.7701e-004 | 79.2266 |
| 0.3 | 4.6198e-004 | 81.4845 |
| 0.4 | 3.4326e-004 | 82.7746 |
| 0.5 | 2.5483e-004 | 84.0683 |
| 0.6 | 2.0707e-004 | 84.9696 |
| 0.7 | 1.7477e-004 | 85.7062 |
| 0.8 | 1.5212e-004 | 86.3090 |
| 0.9 | 1.2903e-004 | 87.0240 |
| 1.0 | 1.1246e-004 | 87.6208 |

**No 0f decomposition levels , L =7**

| BPP | MSE | PSNR[dB] |
|---|---|---|
| 0.1 | 0.0013 | 76.8419 |
| 0.2 | 6.9083e-004 | 79.7371 |
| 0.3 | 4.3056e-004 | 81.7905 |
| 0.4 | 3.2179e-004 | 83.0551 |
| 0.5 | 2.4020e-004 | 84.3251 |
| 0.6 | 1.9797e-004 | 85.1648 |
| 0.7 | 1.6876e-004 | 85.8581 |
| 0.8 | 1.4799e-004 | 86.4284 |
| 0.9 | 1.2464e-004 | 87.1741 |
| 1.0 | 1.0903e-004 | 87.7553 |

**L=9**

| BPP | MSE | PSNR[dB] |
|---|---|---|
| 0.1 | 0.0013 | 76.9106 |
| 0.2 | 6.8652e-004 | 79.7643 |
| 0.3 | 4.2864e-004 | 81.8099 |
| 0.4 | 3.2100e-004 | 83.0657 |
| 0.5 | 2.3945e-004 | 84.3387 |
| 0.6 | 1.9756e-004 | 85.1739 |
| 0.7 | 1.6840e-004 | 85.8675 |
| 0.8 | 1.4775e-004 | 86.4356 |
| 0.9 | 1.2445e-004 | 87.1810 |
| 1.0 | 1.0886e-004 | 87.7619 |

**B.1.3 Results for 512x512 Lena.jpg for wavelet filter BIOR4.4 at max level of decomposition, L=9:**

| BPP | MSE | PSNR[dB] |
|---|---|---|
| 0.1 | 0.0012 | 77.4527 |
| 0.2 | 5.9789e-004 | 80.3646 |
| 0.3 | 3.9766e-004 | 82.1357 |
| 0.4 | 2.8513e-004 | 83.5803 |
| 0.5 | 2.2042e-004 | 84.6983 |
| 0.6 | 1.8459e-004 | 85.4687 |
| 0.7 | 1.5802e-004 | 86.1437 |
| 0.8 | 1.3676e-004 | 86.7712 |
| 0.9 | 1.1770e-004 | 87.4231 |
| 1.0 | 1.0444e-004 | 87.9423 |

**B.1.4 Results for 512x512 Lena.raw for different wavelet filters at max level of decomposition, L=9:**

**A) Wavelet Filter applied=Haar**

| BPP | CR=8/BPP | MSE | PSNR | SNR | RMSE |
|---|---|---|---|---|---|
| 0.1 | 80 | 131.88 | 26.929 | 2.7932e-006 | 11.484 |
| 0.2 | 40 | 76.331 | 29.304 | 4.8261e-006 | 8.7367 |
| 0.3 | 26.667 | 49.915 | 31.149 | 7.3801e-006 | 7.065 |
| 0.4 | 20 | 37.642 | 32.374 | 9.7863e-006 | 6.1353 |
| 0.5 | 16 | 28.66 | 33.558 | 1.2854e-005 | 5.3535 |
| 0.6 | 13.33 | 22.829 | 34.546 | 1.6136e-005 | 4.778 |
| 0.7 | 11.42 | 18.502 | 35.459 | 1.9911e-005 | 4.3014 |
| 0.8 | 10 | 15.692 | 36.174 | 2.3475e-005 | 3.9613 |
| 0.9 | 8.89 | 13.31 | 36.889 | 2.7676e-005 | 3.6483 |
| 1.0 | 8 | 11.578 | 37.494 | 3.1817e-005 | 3.4027 |

**B) Wavelet Filter applied=db2**

| BPP | CR=8/BPP | MSE | PSNR | SNR | RMSE |
|---|---|---|---|---|---|
| 0.1 | 80 | 101.24 | 28.077 | 3.6387e-006 | 10.062 |
| 0.2 | 40 | 54.087 | 30.8 | 6.8108e-006 | 7.3544 |
| 0.3 | 26.667 | 34.417 | 32.763 | 1.0703e-005 | 5.8666 |
| 0.4 | 20 | 25.703 | 34.031 | 1.4332e-005 | 5.0698 |
| 0.5 | 16 | 19.334 | 35.268 | 1.9053e-005 | 4.3971 |
| 0.6 | 13.33 | 15.407 | 36.254 | 2.391e-005 | 3.9252 |
| 0.7 | 11.42 | 13.044 | 36.977 | 2.8241e-005 | 3.6116 |
| 0.8 | 10 | 11.13 | 37.666 | 3.3097e-005 | 3.3362 |
| 0.9 | 8.89 | 9.8128 | 38.213 | 3.754e-005 | 3.1325 |
| 1.0 | 8 | 8.2723 | 38.955 | 4.4531e-005 | 2.8762 |

**C) Wavelet Filter applied=Symlet5**

| BPP | CR=8/BPP | MSE | PSNR | SNR | RMSE |
|---|---|---|---|---|---|
| 0.1 | 80 | 86.333 | 28.769 | 4.2669e-006 | 9.2916 |
| 0.2 | 40 | 43.623 | 31.734 | 8.4445e-006 | 6.6048 |
| 0.3 | 26.667 | 28.106 | 33.643 | 1.3107e-005 | 5.3015 |
| 0.4 | 20 | 20.838 | 34.942 | 1.7678e-005 | 4.5649 |
| 0.5 | 16 | 15.698 | 36.172 | 2.3467e-005 | 3.962 |
| 0.6 | 13.33 | 12.916 | 37.02 | 2.8522e-005 | 3.5938 |
| 0.7 | 11.42 | 10.994 | 37.719 | 3.3506e-005 | 3.3158 |
| 0.8 | 10 | 9.6366 | 38.292 | 3.8227e-005 | 3.1043 |
| 0.9 | 8.89 | 8.0956 | 39.048 | 4.5503e-005 | 2.8453 |
| 1.0 | 8 | 7.0729 | 39.635 | 5.2083e-005 | 2.6595 |

**D) Wavelet Filter applied=Bior4.4**

| BPP | CR=8/BPP | MSE | PSNR | SNR | RMSE |
|---|---|---|---|---|---|
| 0.1 | 80 | 76.043 | 29.32 | 4.8443e-006 | 8.7 |
| 0.2 | 40 | 38.721 | 32.251 | 9.5137e-006 | 6.2226 |
| 0.3 | 26.667 | 25.691 | 34.033 | 1.4339e-005 | 5.0686 |
| 0.4 | 20 | 18.387 | 35.486 | 2.0035e-005 | 4.288 |
| 0.5 | 16 | 14.246 | 36.594 | 2.5858e-005 | 3.7744 |
| 0.6 | 13.33 | 11.899 | 37.376 | 3.096e-005 | 3.4494 |
| 0.7 | 11.42 | 10.19 | 38.049 | 3.6151e-005 | 3.1922 |
| 0.8 | 10 | 8.76 | 38.706 | 4.2052e-005 | 2.9597 |
| 0.9 | 8.89 | 7.5633 | 39.344 | 4.8706e-005 | 2.7501 |
| 1.0 | 8 | 6.7242 | 39.854 | 5.4783e-005 | 2.593 |

**B.1.5 Results for 512x512 Barbara.raw for different wavelet filters at max level of decomposition, L=9**

**A) MSE and PSNR**

| Bpp rate | MSE(Barbara.raw) | | | | PSNR | | | |
|---|---|---|---|---|---|---|---|---|
| | Haar | DB2 | DB5 | SYM5 | Haar | DB2 | DB5 | SYM5 |
| 0.1 | 351.49 | 301.08 | 274.35 | 278.29 | 22.672 | 23.344 | 23.748 | 23.686 |
| 0.2 | 239.38 | 190.1 | 163.83 | 166.02 | 24.34 | 25.341 | 25.987 | 25.929 |
| 0.3 | 180.47 | 139.14 | 113.21 | 116.12 | 25.567 | 26.696 | 27.592 | 27.482 |
| 0.4 | 145 | 104.51 | 80.773 | 83.225 | 26.517 | 27.939 | 29.058 | 28.928 |
| 0.5 | 112.35 | 76.556 | 58.439 | 60.505 | 27.625 | 29.291 | 30.464 | 30.313 |
| 0.6 | 87.068 | 60.512 | 44.734 | 46.961 | 28.732 | 30.312 | 31.624 | 31.413 |
| 0.7 | 71.226 | 48.905 | 35.18 | 36.905 | 29.604 | 31.237 | 32.668 | 32.46 |
| 0.8 | 58.769 | 39.228 | 27.855 | 29.367 | 30.439 | 32.195 | 33.682 | 33.452 |
| 0.9 | 47.758 | 31.738 | 22.136 | 23.293 | 31.34 | 33.115 | 34.68 | 34.459 |
| 1.0 | 39.448 | 25.477 | 17.953 | 19.161 | 32.171 | 34.069 | 35.589 | 35.307 |

**RMSE and SNR:**

| Bpp rate | SNR(Barbara.raw) | | | | RMSE | | | |
|---|---|---|---|---|---|---|---|---|
| | Haar | DB2 | DB5 | SYM5 | Haar | DB2 | DB5 | SYM5 |
| 0.1 | 1.0465e-006 | 1.2217e-006 | 1.3408e-006 | 1.3218e-006 | 18.748 | 17.352 | 16.563 | 16.682 |
| 0.2 | 1.5366e-006 | 1.935e-006 | 2.2452e-006 | 2.2157e-006 | 15.472 | 13.788 | 12.8 | 12.885 |
| 0.3 | 2.0382e-006 | 2.6438e-006 | 3.2493e-006 | 3.1677e-006 | 13.434 | 11.796 | 10.64 | 10.776 |
| 0.4 | 2.5369e-006 | 3.5198e-006 | 4.5541e-006 | 4.4199e-006 | 12.042 | 10.223 | 8.9874 | 9.1228 |
| 0.5 | 3.2742e-006 | 4.805e-006 | 6.2946e-006 | 6.0796e-006 | 10.599 | 8.7496 | 7.6445 | 7.7785 |
| 0.6 | 4.2248e-006 | 6.0789e-006 | 8.223e-006 | 7.8331e-006 | 9.331 | 7.7789 | 6.6883 | 6.8528 |
| 0.7 | 5.1645e-006 | 7.5216e-006 | 1.0456e-005 | 9.9674e-006 | 8.4396 | 6.9932 | 5.9313 | 6.075 |
| 0.8 | 6.2592e-006 | 9.3771e-006 | 1.3206e-005 | 1.2526e-005 | 7.6661 | 6.2632 | 5.2777 | 5.4191 |
| 0.9 | 7.7023e-006 | 1.159e-005 | 1.6618e-005 | 1.5792e-005 | 6.9107 | 5.6336 | 4.7049 | 4.8263 |
| 1.0 | 9.3249e-006 | 1.4438e-005 | 2.0489e-005 | 1.9198e-005 | 6.2808 | 5.0475 | 4.2371 | 4.3773 |

B) **Results for 512x512 Barbara. raw for BIOR 4.4 filter at max level of decomposition, L=9**

| BPP | CR=8/BPP | MSE | PSNR | SNR | RMSE |
|---|---|---|---|---|---|
| 0.1 | 80 | 271.97 | 23.786 | 1.3525e-006 | 16.491 |
| 0.2 | 40 | 163.83 | 25.987 | 2.2453e-006 | 12.8 |
| 0.3 | 26.667 | 111.28 | 27.667 | 3.3057e-006 | 10.549 |
| 0.4 | 20 | 76.211 | 29.311 | 4.8267e-006 | 8.7299 |
| 0.5 | 16 | 56.244 | 30.63 | 6.5402e-006 | 7.4996 |
| 0.6 | 13.33 | 43.662 | 31.73 | 8.4248e-006 | 6.6077 |
| 0.7 | 11.42 | 33.847 | 32.836 | 1.0868e-005 | 5.8178 |
| 0.8 | 10 | 26.347 | 33.923 | 1.3962e-005 | 5.133 |
| 0.9 | 8.89 | 21.451 | 34.816 | 1.7148e-005 | 4.6315 |
| 1.0 | 8 | 17.599 | 35.676 | 2.0902e-005 | 4.1951 |

**B.2 RESULTS: Using LIFTING Technique for wavelet filter CDF(9,7),for 'Lena.png' (512x512) image at various levels of decomposition**

**L2**:Second level of decomposition

| bpp | mse | psnr | snr |
|---|---|---|---|
| 0.1 | 0.1413 | 56.627 | 0.0013093 |
| 0.2 | 0.075222 | 59.367 | 0.002461 |
| 0.3 | 0.075222 | 59.367 | 0.002461 |
| 0.4 | 0.075222 | 59.367 | 0.002461 |
| 0.5 | 0.056593 | 60.603 | 0.0032711 |
| 0.6 | 0.021334 | 64.84 | 0.0086772 |
| 0.7 | 0.021334 | 64.84 | 0.0086772 |
| 0.8 | 0.021334 | 64.84 | 0.0086772 |
| 0.9 | 0.019656 | 65.196 | 0.0094184 |
| 1 | 0.0066773 | 69.885 | 0.027724 |

**L3:**

| bpp | mse | psnr | snr |
|---|---|---|---|
| 0.1 | 0.080175 | 59.09 | 0.002309 |
| 0.2 | 0.023301 | 64.457 | 0.0079449 |
| 0.3 | 0.0074095 | 69.433 | 0.024985 |
| 0.4 | 0.0032706 | 72.984 | 0.056602 |
| 0.5 | 0.0017408 | 75.723 | 0.10635 |
| 0.6 | 0.0010178 | 78.054 | 0.18189 |
| 0.7 | 0.0006951 | 79.71 | 0.26632 |
| 0.8 | 0.0004457 | 81.64 | 0.41531 |
| 0.9 | 0.0003509 | 82.678 | 0.52745 |
| 1 | 0.0002554 | 84.058 | 0.72478 |

**L5:**

| bpp | mse | psnr | snr |
|---|---|---|---|
| 0.1 | 0.0014056 | 76.652 | 0.1317 |
| 0.2 | 0.0006361 | 80.095 | 0.291 |
| 0.3 | 0.0003972 | 82.14 | 0.466 |
| 0.4 | 0.0002863 | 83.562 | 0.64646 |
| 0.5 | 0.0002191 | 84.723 | 0.8447 |
| 0.6 | 0.0001829 | 85.507 | 1.0118 |
| 0.7 | 0.0001567 | 86.178 | 1.1808 |
| 0.8 | 0.0001353 | 86.816 | 1.3676 |
| 0.9 | 0.0001166 | 87.462 | 1.5872 |
| 1 | 0.0001037 | 87.972 | 1.7848 |

**L7:**

| bpp | mse | psnr | snr |
|---|---|---|---|
| 0.1 | 0.0010565 | 77.892 | 0.17522 |
| 0.2 | 0.00053861 | 80.818 | 0.3437 |
| 0.3 | 0.00036137 | 82.551 | 0.51228 |
| 0.4 | 0.00026216 | 83.945 | 0.70615 |
| 0.5 | 0.00020983 | 84.912 | 0.88225 |
| 0.6 | 0.00017396 | 85.726 | 1.0642 |
| 0.7 | 0.00015171 | 86.321 | 1.2202 |
| 0.8 | 0.00012871 | 87.035 | 1.4293 |
| 0.9 | 0.00011276 | 87.609 | 1.6418 |
| 1 | 0.00010092 | 88.091 | 1.8343 |

**L8:**

| bpp | mse | psnr | snr |
|---|---|---|---|
| 0.1 | 0.0010429 | 77.948 | 0.17751 |
| 0.2 | 0.00053626 | 80.837 | 0.34521 |
| 0.3 | 0.00036039 | 82.563 | 0.51368 |
| 0.4 | 0.00026151 | 83.956 | 0.70789 |
| 0.5 | 0.00020945 | 84.92 | 0.88386 |
| 0.6 | 0.00017377 | 85.731 | 1.0653 |
| 0.7 | 0.00015159 | 86.324 | 1.2212 |
| 0.8 | 0.00012854 | 87.04 | 1.4402 |
| 0.9 | 0.00011261 | 87.615 | 1.6439 |
| 1 | 0.00010085 | 88.094 | 1.8357 |

## B.3 Results for non –lifting version of wavelet filter CDF(9,7) i.e. bior4.4, for the image 'Lena.png' (512x512)

**L3:**

| bpp | mse | psnr | snr |
|---|---|---|---|
| 0.1 | 0.082111 | 58.987 | 0.0022546 |
| 0.2 | 0.02363 | 64.396 | 0.0078342 |
| 0.3 | 0.0077318 | 69.248 | 0.023943 |
| 0.4 | 0.0034619 | 72.738 | 0.053474 |
| 0.5 | 0.0018622 | 75.431 | 0.09941 |
| 0.6 | 0.0011051 | 77.697 | 0.16752 |
| 0.7 | 0.00073514 | 79.467 | 0.25182 |
| 0.8 | 0.00047869 | 81.33 | 0.38673 |
| 0.9 | 0.0003674 | 82.479 | 0.50388 |
| 1.0 | 0.00026499 | 83.899 | 0.69862 |

**L5:**

| bpp | mse | psnr | snr |
|---|---|---|---|
| 0.1 | 0.0015776 | 76.151 | 0.11735 |
| 0.2 | 0.00069944 | 79.683 | 0.26468 |
| 0.3 | 0.00043358 | 81.76 | 0.42697 |
| 0.4 | 0.00031166 | 83.194 | 0.59399 |
| 0.5 | 0.00023303 | 84.457 | 0.79442 |
| 0.6 | 0.00019359 | 85.262 | 0.95625 |
| 0.7 | 0.00016333 | 86 | 1.1335 |
| 0.8 | 0.00014353 | 86.561 | 1.2898 |
| 0.9 | 0.00012177 | 87.275 | 1.5203 |
| 1.0 | 0.00010716 | 87.83 | 1.7275 |

**L6:**

| bpp | mse | psnr | snr |
|---|---|---|---|
| 0.1 | 0.0012385 | 77.202 | 0.14947 |
| 0.2 | 0.00062381 | 80.18 | 0.29676 |
| 0.3 | 0.00040525 | 82.054 | 0.45682 |
| 0.4 | 0.00029159 | 83.483 | 0.63488 |
| 0.5 | 0.00022257 | 84.656 | 0.83176 |
| 0.6 | 0.00018648 | 85.424 | 0.99272 |
| 0.7 | 0.00015904 | 86.116 | 1.164 |
| 0.8 | 0.00013818 | 86.726 | 1.3397 |
| 0.9 | 0.00011859 | 87.39 | 1.561 |
| 1.0 | 0.00010494 | 87.921 | 1.7641 |

**L7:**

| bpp | mse | psnr | snr |
|---|---|---|---|
| 0.1 | 0.0011808 | 77.409 | 0.15678 |
| 0.2 | 0.00060392 | 80.321 | 0.30654 |
| 0.3 | 0.00039929 | 82.118 | 0.46363 |
| 0.4 | 0.00028632 | 83.562 | 0.64656 |
| 0.5 | 0.00022094 | 84.688 | 0.83789 |
| 0.6 | 0.00018499 | 85.459 | 1.0007 |
| 0.7 | 0.00015822 | 86.138 | 1.17 |
| 0.8 | 0.00013709 | 86.761 | 1.3504 |
| 0.9 | 0.00011788 | 87.416 | 1.5705 |
| 1.0 | 0.00010456 | 87.937 | 1.7706 |

**L8:**

| bpp | mse | psnr | snr |
|---|---|---|---|
| 0.1 | 0.0011721 | 77.441 | 0.15794 |
| 0.2 | 0.00059915 | 80.355 | 0.30898 |
| 0.3 | 0.00039799 | 82.132 | 0.46514 |
| 0.4 | 0.00028549 | 83.575 | 0.64843 |
| 0.5 | 0.0002205 | 84.697 | 0.83957 |
| 0.6 | 0.0001847 | 85.466 | 1.0023 |
| 0.7 | 0.00015805 | 86.143 | 1.1713 |
| 0.8 | 0.00013681 | 86.769 | 1.3531 |
| 0.9 | 0.00011773 | 87.422 | 1.5724 |
| 1 | 0.00010446 | 87.941 | 1.7722 |