Implementation of Image Compression Algorithm using Verilog with Area, Power and Timing Constraints

A THESIS SUBMITTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF

Master of Technology

in

VLSI Design and Embedded System

By

ARUN KUMAR P S

ROLL No: 207EC203

Department of Electronics and Communication Engineering

National Institute Of Technology

Rourkela

2007-2009

Implementation of Image Compression Algorithm using Verilog with Area, Power and Timing Constraints

A THESIS SUBMITTED IN PARTIAL FULFILLMENT
OF THE REQUIREMENTS FOR THE DEGREE OF

Master of Technology

in

VLSI Design and Embedded System

By

ARUN KUMAR P S

ROLL No: 207EC203

Under the Guidance of

Prof. KAMALAKANTA MAHAPATRA

Department of Electronics and Communication Engineering

National Institute Of Technology

Rourkela

2007-2009

National Institute Of Technology Rourkela

CERTIFICATE

This is to certify that the thesis entitled, **“Implementation of Image Compression Algorithm using Verilog with Area, Power and Timing Constraints”** submitted by **ARUN KUMAR P S** in partial fulfillment of the requirements for the award of Master of Technology Degree in **Electronics & Communication Engineering** with specialization in **“VLSI Design and Embedded System”** at the National Institute of Technology, Rourkela (Deemed University) is an authentic work carried out by him under my supervision and guidance.

To the best of my knowledge, the matter embodied in the thesis has not been submitted to any other University / Institute for the award of any Degree or Diploma.

Date:

Prof. K. K. Mahapatra
Dept. of Electronics & Communication Engg.
National Institute of Technology
Rourkela-769008

ACKNOWLEDGEMENTS

This project is by far the most significant accomplishment in my life and it would be impossible without people (especially my family) who supported me and believed in me.

I am thankful to **Dr. K. K. Mahapatra,** Professor in the department of Electronics and Communication Engineering, NIT Rourkela for giving me the opportunity to work under him and lending every support at every stage of this project work. I truly appreciate and value him esteemed guidance and encouragement from the beginning to the end of this thesis. I am indebted to his for having helped me shape the problem and providing insights towards the solution. His trust and support inspired me in the most important moments of making right decisions and I am glad to work with him.

I want to thank all my teachers **Prof. S.K. Patra, Prof. G.Panda, Prof. G.S. Rath, Prof. S. Meher,** and **Prof. D.P.Acharya** for providing a solid background for my studies and research thereafter.

I am also very thankful to all my classmates and seniors of VLSI lab-I especially Sushant Pattnaik, Dr. Jitendra K Das, Swain Ayas Kanta, K Sudeendra Kumar and my friends Deepak Krishnankutty and George Tom V, who always encouraged me in the successful completion of my thesis work.

ARUN KUMAR P S

ROLL No: 207EC203

ABSTRACT

Image compression is the application of Data compression on digital images. A fundamental shift in the image compression approach came after the Discrete Wavelet Transform (DWT) became popular. To overcome the inefficiencies in the JPEG standard and serve emerging areas of mobile and Internet communications, the new JPEG2000 standard has been developed based on the principles of DWT.

An image compression algorithm was comprehended using Matlab code, and modified to perform better when implemented in hardware description language.

Using Verilog HDL, the encoder for the image compression employing DWT was implemented. Detailed analysis for power, timing and area was done for Booth multiplier which forms the major building block in implementing DWT. The encoding technique exploits the zero tree structure present in the bitplanes to compress the transform coefficients.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Data compression is the technique to reduce the redundancies in data representation in order to decrease data storage requirements and hence communication costs. Reducing the storage requirement is equivalent to increasing the capacity of the storage medium and hence communication bandwidth. Thus the development of efficient compression techniques will continue to be a design challenge for future communication systems and advanced multimedia applications.

Data is represented as a combination of information and redundancy. Information is the portion of data that must be preserved permanently in its original form in order to correctly interpret the meaning or purpose of the data. Redundancy is that portion of data that can be removed when it is not needed or can be reinserted to interpret the data when needed. Most often, the redundancy is reinserted in order to generate the original data in its original form. A technique to reduce the redundancy of data is defined as Data compression. The redundancy in data representation is reduced such a way that it can be subsequently reinserted to recover the original data, which is called decompression of the data.

Classification of Compression Algorithms

Data compression can be understood as a method that takes an input data D and generates a shorter representation of the data c(D) with less number of bits compared to that of D. The reverse process is called decompression, which takes the compressed data c(D) and generates or reconstructs the data D' as shown in Figure 1. Sometimes the compression (coding) and decompression (decoding) systems together are called a “CODEC”.

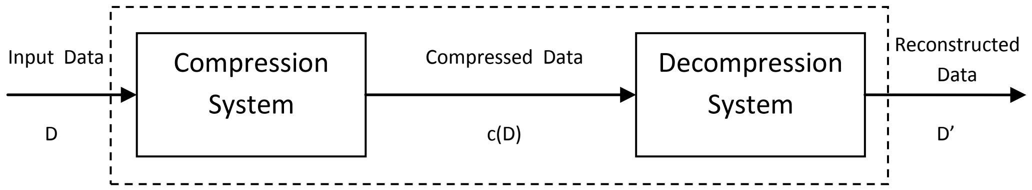


Figure 1 CODEC

The reconstructed data D' could be identical to the original data D or it could be an approximation of the original data D, depending on the reconstruction requirements. If the reconstructed data D’ is an exact replica of the original data D, the algorithm applied to compress D and decompress c(D) is lossless. On the other hand, the algorithms are lossy when D’ is not an exact replica of D. Hence as far as the reversibility of the original data is concerned, the data compression algorithms can be broadly classified in two categories – lossless and lossy. Usually loseless data compression techniques are applied on text data or scientific data.

Sometimes data compression is referred as coding, and the terms noiseless or noisy coding, usually refer to loseless and lossy compression techniques respectively. The term “noise” here is the “error of reconstruction” in the lossy compression techniques because the reconstructed data item is not identical to the original one.

Data compression schemes could be static or dynamic. In static methods, the mapping from a set of messages (data or signal) to the corresponding set of compressed codes is always fixed. In dynamic methods, the mapping from the set of messages to the set of compressed codes changes over time. A dynamic method is called adaptive if the codes adapt to changes in ensemble characteristics over time. For example, if the probabilities of occurrences of the symbols from the source are not fixed over time, an adaptive formulation of the binary codewords of the symbols is suitable, so that the compressed file size can adaptively change for better compression efficiency.

Advantages of Data Compression

i) It reduces the data storage requirements

ii) The audience can experience rich-quality signals for audio-visual data representation

iii) Data security can also be greatly enhanced by encrypting the decoding parameters and transmitting them separately from the compressed database files to restrict access of proprietary information

iv) The rate of input-output operations in a computing device can be greatly increased due to shorter representation of data

v) Data Compression obviously reduces the cost of backup and recovery of data in computer systems by storing the backup of large database files in compressed form

Disadvantages of Data Compression

i) The extra overhead incurred by encoding and decoding process is one of the most serious drawbacks of data compression, which discourages its use in some areas

ii) Data compression generally reduces the reliability of the records

iii) Transmission of very sensitive compressed data through a noisy communication channel is risky because the burst errors introduced by the noisy channel can destroy the transmitted data

iv) Disruption of data properties of a compressed data, will result in compressed data different from the original data

v) In many hardware and systems implementations, the extra complexity added by data compression can increase the system's cost and reduce the system's efficiency, especially in the areas of applications that require very low-power VLSI implementation

A Data Compression Model

A model of a typical data compression system can be described using the block diagram shown in Figure 2. A data compression system mainly consists of three major steps – removal or reduction in data redundancy, reduction in entropy and entropy encoding.

The redundancy in data may appear in different forms. For example, the neighbouring pixels in a typical image are very much spatially correlated to each other. By correlation it means that the pixel values are very similar in the non-edge smooth regions in the image. The composition of the words or sentences in a natural text follows same context model based on the grammar being used. Similarly, the records in a typical numeric database may have some sort of relationship among the atomic entities that comprise each record in the database. There are rhythms and pauses on regular intervals in any natural audio or speech data. These redundancies in data representation can be reduced in order to achieve potential compression.

Removal or reduction in data redundancy is typically achieved by transforming the original data from one form or representation to another. The popular techniques used in the redundancy reduction step are prediction of the data samples using some model, transformation of the original data from spatial domain such as Discrete Cosine Transform (DCT), decomposition of the original data set into different subbands such as Discrete Wavelet Transform (DWT), etc. In principle, this step potentially yields more compact representation of the information in the original data set in terms of fewer coefficients or equivalent. In case of loseless data compression, this step is completely reversible. Transformation of data usually reduces entropy of the original data by moving the redundancies that appear in the known structure of the data sequence.

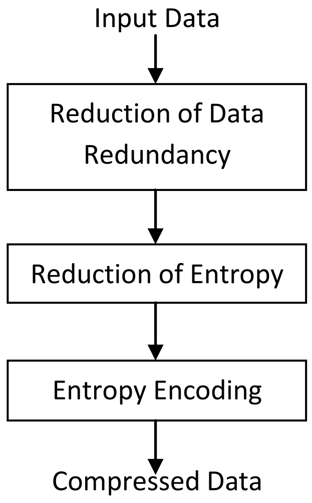


Figure 2 A Data Compression Model

The next major step in a lossy data compression system is to further reduce the entropy of the transformed data significantly in order to allocate fewer bits for transmission or storage. The reduction in entropy is achieved by dropping nonsignificant information in the transformed data based on the application criteria. This is a nonreversible process because it is not possible to exactly recover the lost data or information using the inverse process. This step is applied in lossy data compression schemes and this is usually accomplished by some version of quantization technique. The nature and amount of quantization dictate the quality of the reconstructed data. The quantized coefficients are then losslessly encoded using some entropy scheme to compactly represent the quantized data for storage or transmission. Since the entropy of the quantized data is less compared to the original one, it can be represented by fewer bits compared to the original data set, hence compression is accomplished.

The decomposition system is just an inverse process. The compressed code is first decoded to generate the quantized coefficients. The inverse quantization step is applied on these quantized coefficients to generate the approximation of the transformed coefficients. The quantized transformed coefficients are then inverse transformed in order to create the approximate version of the original data. If the quantization and inverse quantization steps are absent in the codec and the transformation step for redundancy removal is reversible, the decompression system produces the exact replica of the original data and hence the compression system can be called a lossless compression system.

Image Compression

Image compression is the application of Data compression on digital images. The objective of image compression is to reduce redundancy of the image data in order to be able to store or transmit data in an efficient form. Image compression can be lossy or lossless. Lossless compression is sometimes preferred for artificial images such as technical drawings, icons or comics. This is because lossy compression methods, especially when used at low bit rates, introduce compression artifacts. Lossless compression methods may also be preferred for high value content, such as medical imagery or image scans made for archival purposes. Lossy methods are especially suitable for natural images such as photos in applications where minor loss of fidelity is acceptable to achieve a substantial reduction in bit rate. The lossy compression that produces imperceptible differences can be called visually lossless. Run-length encoding and entropy encoding are the methods for lossless image compression. Transform coding, where a Fourier-related transform such as DCT or the wavelet transform are applied, followed by quantization and entropy coding can be cited as a method for lossy image compression.

Compression Artifact

A compression artifact (or artefact) is the result of an aggressive data compression scheme applied to an image, that discards some data that may be too complex to store in the available data-rate, or may have been incorrectly determined by an algorithm to be of little subjective importance, but is in fact objectionable to the viewer. Artifacts are often a result of the latent errors inherent in lossy data compression.

Some of the common artefacts are:

- *Blocking Artifacts* : A distortion that appears in compressed image as abnormally large pixel blocks. Also called "macroblocking," it occurs when the encoder cannot keep up with the allocated bandwidth. Image uses lossy compression, and the higher the compression rate, the more content is removed. At decompression, the output of certain decoded blocks makes surrounding pixels appear averaged together and look like larger blocks.

- *Colour Distortion* : As human eyes are not as sensitive to colour as to brightness, much of the detailed colour (chrominance) information is disposed, while luminance is retained. This process is called "chroma subsampling", and it means that a colour image is split into a brightness image and two colour images. The brightness (luma) image is stored at the original resolution, whereas the two colour (chroma) images are stored at a lower resolution. The compressed images look slightly washed-out, with less brilliant colour.
- *Ringing Artifacts* : In digital image processing, ringing artifacts are artifacts that appear as spurious signals ("rings") near sharp transitions in a signal. Visually, they appear as "rings" near edges. As with other artifacts, their minimization is a criterion in filter design. The main cause of ringing artifacts is due to a signal being bandlimited (specifically, not having high frequencies) or passed through a low-pass filter; this is the frequency domain description. In terms of the time domain, the cause of this type of ringing is the ripples in the sinc function, which is the impulse response (time domain representation) of a perfect low-pass filter. Mathematically, this is called the Gibbs phenomenon.
- *Blurring Artifacts* : Blurring means that the image is smoother than originally.

Literature Review

The discrete wavelet transform (DWT) [1] has gained wide popularity due to its excellent decorrelation property, many modern image and video compression systems embody the DWT as the transform stage [2]. It is widely recognized that the 9/7 filters [3] are among the best filters for DWT-based image compression [4]. In fact, the JPEG2000 image coding standard [5] employs the 9/7 filters as the default wavelet filters for lossy compression and 5/3 filters for lossless compression. The performance of a hardware implementation of the 9/7 filter bank (FB) depends on the accuracy with which filter coefficients are represented. Lossless image compression techniques find applications in fields such as medical imaging, preservation of artwork, remote sensing etc. Day-by-day Discrete Wavelet Transform (DWT) is becoming more and more popular for digital image compression. Biorthogonal (5, 3) and (9, 7) filters have been chosen to be the standard filters used in the JPEG2000 codec standard [5].

Discrete wavelet transform as reported by Zervas et al. [6], there are three basic architectures for the two-dimensional DWT: level-by-level, line-based, and block-based architectures. In implementing the 2-D DWT, a recursive algorithm based on the line based architectures is used. The image to be transformed is stored in a 2-D array. Once all the elements in a row is obtained, the convolution is performed in that particular row. The process of row-wise convolution will divide the given image into two parts with the number of rows in each part equal to half that of the image. This matrix is again subjected to a recursive line-based convolution, but this time column-wise. The result will DWT coefficients corresponding to the image, with the approximation coefficient occupying the top-left quarter of the matrix, horizontal coefficients occupying the bottom-left quarter of the matrix, vertical coefficients occupying the top-right quarter of the matrix and the diagonal coefficients occupying the bottom-right quarter of the matrix.

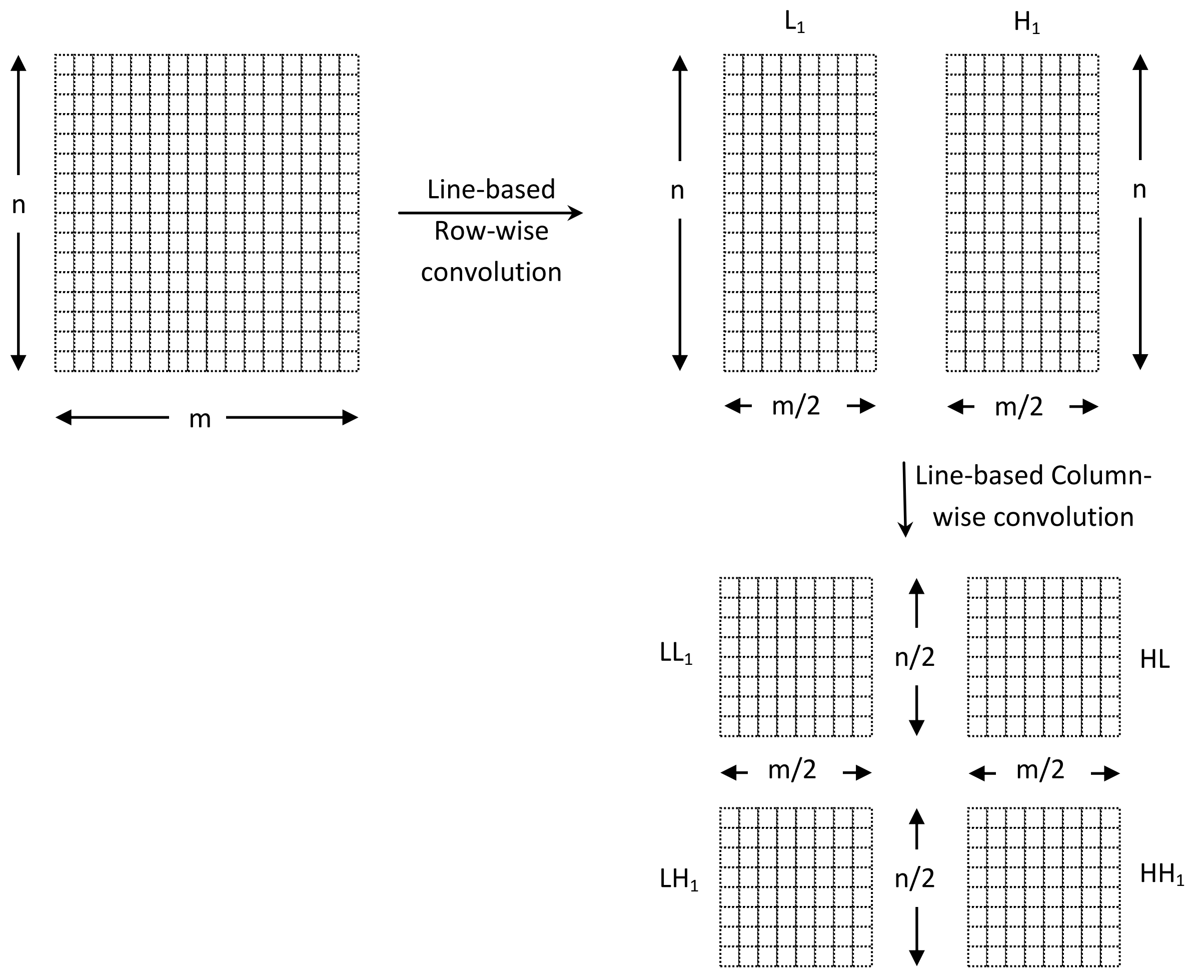

n
m
Line-based Row-wise convolution
L1
H1
n
n
m/2
m/2
Line-based Column-wise convolution
LL1
n/2
HL
m/2
m/2
LH1
n/2
HH1

Figure 3 Line Based Architecture for 2-D DWT

After DWT was introduced, several codec algorithms were proposed to compress the transform coefficients as much as possible. Among them, Embedded Zerotree Wavelet (EZW) [7], Set Partitioning In Hierarchical Trees (SPIHT) [8] and Embedded Bock Coding with Optimized Truncation (EBCOT) [2] are the most famous ones.

The embedded zerotree 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 addition to producing a fully embedded bit stream, EZW consistently produces compression results that are competitive with virtually all known compression algorithms on standard test images. Yet this performance is achieved with a technique that requires absolutely no training, no pre-stored tables or codebooks, and requires no prior knowledge of the image source. The EZW algorithm is based on four key concepts: 1) a discrete wavelet transform or hierarchical subband decomposition, 2) prediction of the absence of significant information across scales by exploiting the self-similarity inherent in images, 3) entropy-coded successive-approximation quantization, and 4) universal lossless data compression which is achieved via adaptive arithmetic coding.

On the contrary, SPIHT, rooted from EZW, enjoys a much simpler coding procedure. If no entropy coding or arithmetic coding methods are incorporated, SPIHT does not require any table with slight loss in compression ratio. With SPIHT, the encoded bit stream can be divided into several successive sections of sub bit streams during the encoding process. With each additional data, the quality of the decoded image is better. This is very important for progressive transmission of digital images in application such as tele-medicine. Moreover, SPIHT can be easily used in fixed rate or variable rate transmission applications. With integer DWT, SPIHT can also be used in lossless image compression. In the image processing applications, first the wavelet coefficients are stored in a 2-D array. Because searching for the descendants of a given coefficient has to be performed very frequently, it is necessary to design a dedicated circuitry to compute the coefficient addresses. This also consumes more number of clock cycles. Second, the three lists used in SPIHT algorithm store the coefficients coordinates. Transactions among the lists are required frequently. Third, linked lists are used as the data structure to store the coefficients. When there is a change to the list, insertion or deletion operations are necessary.

EBCOT algorithm exhibits state-of-the-art compression performance while producing a bit-stream with a rich set of features, including resolution and SNR scalability together with a “random access” property. The algorithm has modest complexity and is suitable for applications involving remote browsing of large compressed images. The algorithm lends itself to explicit optimization with respect to MSE as well as more realistic psychovisual metrics, capable of modelling the spatially varying visual masking phenomenon. While EBCOT has the best compression rate of all and is adopted by JPEG2000, it requires much more complex multi-layer coding procedures, multiple coding tables and arithmetic coding techniques. These make the hardware implementation of EBCOT codec more difficult and expensive.

The core coding system in JPEG2000 has been defined in Part 1 of the standard. The whole compression system can be divided into three phases- image preprocessing, compression, and compressed bitstream formation. The concepts behind the preprocessing functionalities, includes tiling of the input image, DC level shifting, and multicomponent transformation, before the actual compression takes place. In lossy compression mode, a dead-zone scalar quantization technique is applied on the wavelet coefficients. The concept of region of interest coding allows one to encode different regions of the input image with different fidelity. The entropy coding and the generation of compressed bitstream in JPEG2000 are divided into two coding steps: *Tier-l* and *Tier-2* coding.

Chapter 2

Discrete Wavelet Transform

Mathematically a “wave” is expressed as a sinusoidal (or oscillating) function of time or space. Fourier analysis expands an arbitrary signal in terms of infinite number of sinusoidal functions of its harmonics. Fourier representation of signals is known to be very effective in analysis of time-invariant (stationary) periodic signals. In contrast to a sinusoidal function, a wavelet is a small wave whose energy is concentrated in time. Properties of wavelets allow both time and frequency analysis of signals simultaneously because of the fact that the energy of wavelets is concentrated in time and still possesses the wave-like (periodic) characteristics. Wavelet representation thus provides a versatile mathematical tool to analyse transient, time-variant (nonstationary) signals that may not be statistically predictable especially at the region of discontinuities – a special feature that is typical of images having discontinuities at the edges.

Wavelet Transforms

Wavelets are functions generated from one single function (basis function) called the prototype or mother wavelet by dilations (scalings) and translations (shifts) in time (frequency) domain. If the mother wavelet is denoted by $\psi(\mathrm{t})$, the other wavelets$\psi_{a,b}(\mathrm{t})$ can be represented as

$$\psi_{a,b}(\mathrm{t}) = \frac{1}{\sqrt{|a|}}\psi\left(\frac{t-b}{a}\right) \qquad (2.1)$$

where *a* and *b* are two arbitrary real numbers. The variables *a* and *b* represent the parameters for dilations and translations respectively in the time axis. From Eq. 2.1, it is obvious that the mother wavelet can be essentially represented as

$$\psi(\mathrm{t}) = \psi_{1,0}(\mathrm{t}) \qquad (2.2)$$

For any arbitrary $a \neq 1$ and $b = 0$, it is possible to derive that

$$\psi_{a,b}(\mathrm{t}) = \frac{1}{\sqrt{|a|}}\psi\left(\frac{t-b}{a}\right) \qquad (2.3)$$

As shown in Eq. 2.3, $\psi_{a,0}(t)$ is nothing but a time-scaled (by *a*) and amplitude-scaled (by $\sqrt{|a|}$) version of the mother wavelet function $\psi(\mathrm{t})$ in Eq. 2.2. The parameter *a* causes contraction of $\psi(\mathrm{t})$ in the time axis when a < 1 and expression or stretching when *a* > 1. That’s why the parameter *a* is called the dilation (scaling) parameter. For *a* < 0, the function $\psi_{a,b}(\mathrm{t})$ results in time reversal with dilation. Mathematically, substituting *t* in Eq. 2.3 by *t-b* to cause a translation or shift in the time axis resulting in the wavelet function $\psi_{a,b}(\mathrm{t})$ as shown in

Eq. 2.1. The function $\psi_{a,b}(t)$ is a shift of $\psi_{a,0}(t)$ in right along the time axis by an amount *b* when *b* > 0 whereas it is a shift in left along the time axis by an amount *b* when *b* < 0. That's why the variable *b* represents the translation in time (shift in frequency) domain.

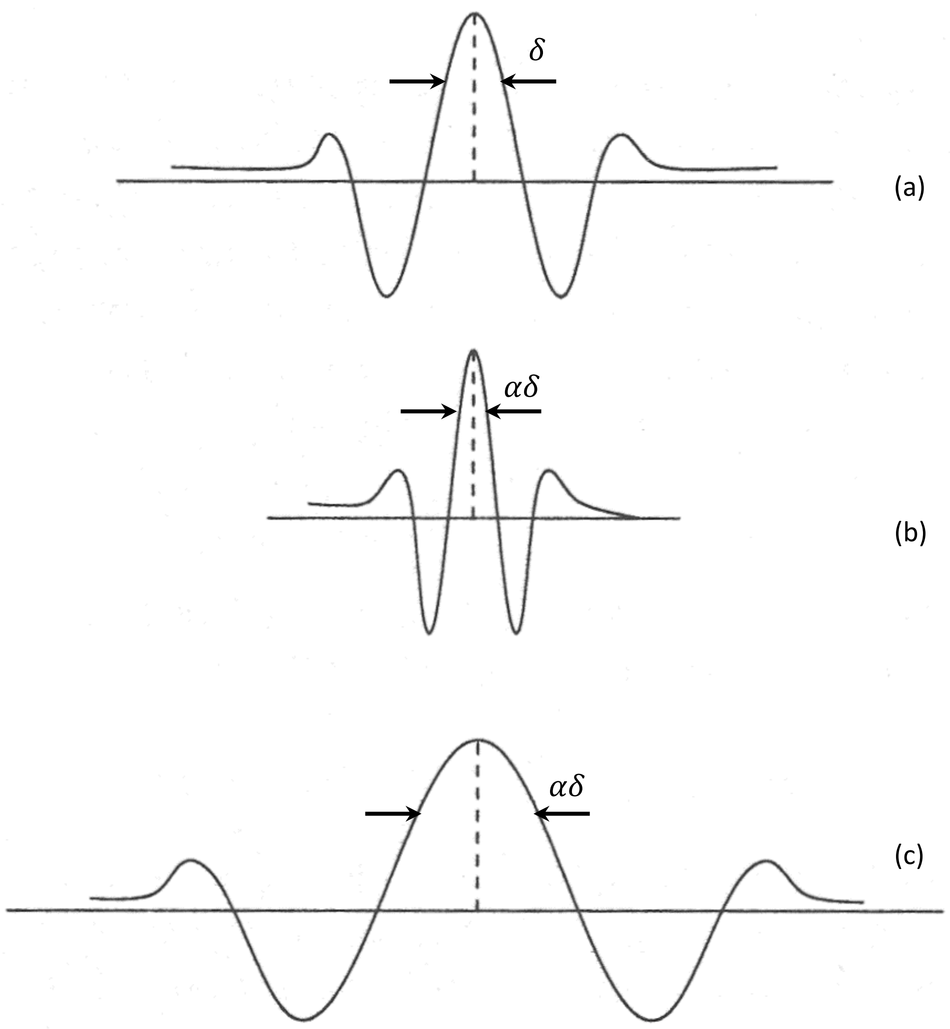


Figure 4 (a) A mother wavelet, (b)$\psi(t/\alpha)$: $0 < \alpha < 1$, (c)$\psi(t/\alpha)$: $\alpha > 1$.

Figure 4 shows an illustration of a mother wavelet and its dilations in the time domain with the dilation parameter $a = \alpha$. For the mother wavelet $\psi(t)$ shown in Figure 4(a), a contraction of the signal in the time axis when $\alpha < 1$ is shown in Figure 4(b) and expansion of the signal in the time axis when $\alpha > 1$ is shown in Figure 4(c). Based on this definition of wavelets, the wavelet transform (WT) of a function (signal) *f*(t) is mathematically represented by

$$W(a,b) = \int_{-\infty}^{+\infty} \psi_{a,b}(\mathrm{t})\, f(\mathrm{t})\mathrm{dt} \qquad (2.4)$$

The inverse transform to reconstruct *f*(t) from W(a, b) is mathematically represented by

$$f(\mathrm{t}) = \frac{1}{C}\int_{a=-\infty}^{+\infty}\int_{b=-\infty}^{+\infty}\frac{1}{|a|^2}\mathrm{W}(a,b)\,\psi_{a,b}(\mathrm{t})\,dadb \qquad (2.5)$$

where

$$C = \int_{-\infty}^{\infty}\frac{|\Psi(\omega)|^2}{|\omega|}d\omega \qquad (2.6)$$

and $\Psi(\omega)$ is the Fourier transform of the mother wavelet $\psi(\mathrm{t})$.

If *a* and *b* are two continuous (nondiscrete) variables and *f*(t) is also a continuous function, $\mathrm{W}(a,b)$ is called the continuous wavelet transform (CWT). Hence the CWT maps a one-dimensional function *f*(t) to a function W(*a*, *b*) of two continuous real variables a (dilation) and b (translation).

Discrete Wavelet Transforms

Since the input signal (e.g., a digital image) is processed by a digital computing machine, it is prudent to define the discrete version of the wavelet transform. To define the wavelet in terms of discrete values of the dilation and translation parameters *a* and *b* instead of being continuous, make *a* and *b* discrete using Eq. 2.6,

$$a = a_0^m, \qquad\qquad b = nb_0a_0^m$$

where *m* and *n* are integers. Substituted *a* and *b* in Eq. 2.1 by Eq. 2.6, the discrete wavelets can be represented by Eq. 2.7.

$$\psi_{m,n}(\mathrm{t}) = a_0^{-\mathrm{m}/2}\psi(a_0^{-m}t - nb_0) \qquad (2.7)$$

There are many choices to select the values of a_0 and b_0. By selecting $a_0 = 2$ and $b_0 = 1$, $a = 2^m$ and $b = n2^m$. This corresponds to sampling (discretization) of *a* and *b* in such a way that the consecutive dicrete values of *a* and *b* as well as the sampling intervals differ by a factor of two. This way of sampling is popularly known as dyadic decomposition. Using these values, it is possible to represent the discrete wavelets as in eq. 2.8, which constitutes a family of orthonormal basis functions.

$$\psi_{m,n}(\mathrm{t}) = 2^{-m/2}\psi(a_0^{-m}t - n) \qquad (2.8)$$

In general, the wavelet coefficients for function *f*(t) are given by

$$C_{m,n}(f) = a_0^{-m/2}\int f(t)\ \psi(a_0^{-m}t - nb_0)dt \qquad (2.9)$$

and hence for dyadic decomposition, the wavelet coefficients can be derived accordingly as

$$C_{m,n}(f) = 2^{-m/2} \int f(t)\ \psi(a_0^{-m} t - nb_0) dt \qquad (2.10)$$

This allows us to reconstruct the signal *f*(t) in form the discrete wavelet coefficients as

$$f(t) = \sum_{m=-\infty}^{\infty} \sum_{n=-\infty}^{\infty} c_{m,n}\ (f) \psi_{m,n}(\mathrm{t}) \qquad (2.11)$$

The transform shown in Eq. 2.9 is called the wavelet series, which is analogous to the Fourier series because the input function *f*(t) is still a continuous function whereas the transform coefficients are discrete. This is often called the discrete time wavelet transform (DTWT). For digital signal or image processing applications executed by a digital computer, the input signal *f*(t) needs to be discrete in nature because of the digital sampling of the original data, which is represented by a finite number bits. When the input function *f*(t) as well as the wavelet parameters a and b are represented in discrete form, the transformation is commonly referred to as the discrete wavelet transform (DWT) of the signal *f*(t).

The discrete wavelet transform (DWT) became a very versatile signal processing tool after Mallat [9] proposed the multiresolution representation of signals based on wavelet decomposition. The method of multiresolution is to represent a function (signal) with a collection of coefficients, each of which provides information about the position as well as the frequency of the signal (function). The advantage of DWT over Fourier transformation is that it performs multiresolution analysis of signals with localization. As a result, the DWT decomposes a digital signal into different subbands so that the lower frequency subbands will have finer frequency resolution and coarser time resolution compared to the higher frequency subbands. The DWT is being increasingly used for image compression due to the fact that the DWT supports features like progressive image transmission (by quality, by resolution), ease of compressed image manipulation, region of interest coding, etc. Because of these characteristics, the DWT is the basis of the new JPEG2000 image compression standard [10].

Concept of Multiresolution Analysis

There are a number of orthogonal wavelet basis-functions of the form $\psi_{m,n}(\mathrm{t}) = 2^{-m/2}\psi(a_0^{-m} t - n)$. The theory of multiresolution analysis presented a systematic approach to generate the wavelets. The idea of multiresolution analysis is to approximate a function *f*(t) at different levels of resolution.

In multiresolution analysis, two functions are considered: the mother wavelet $\psi(\mathrm{t})$ and the scaling function $\phi(\mathrm{t})$. The dilated (scaled) and translated (shifted) version of the scaling function is given by $\phi_{m,n}(\mathrm{t}) = 2^{-m/2}\phi(a_0^{-m}t - n)$. For fixed m, the set of scaling functions $\phi_{m,n}(\mathrm{t})$ are orthonormal. By the linear combinations of the scaling function and its translations, a set of functions can be generated

$$f(\mathrm{t}) = \sum_{\mathrm{n}} \alpha_{\mathrm{n}}\ \phi_{m,n}(\mathrm{t}) \qquad (2.12)$$

The set of all such functions generated by linear combination of the set $\{\phi_{m,n}(\mathrm{t})\}$ is called the span of the set $\{\phi_{m,n}(\mathrm{t})\}$, denoted by Span$\{\phi_{m,n}(\mathrm{t})\}$. Now consider V_m to be a vector space corresponding to Span$\{\phi_{m,n}(\mathrm{t})\}$. Assuming that the resolution increases with decreasing m, these vector spaces describe successive approximation vector spaces, ... $\subset V_2 \subset V_1 \subset V_0 \subset V_{-1} \subset V_{-2} \subset$ ..., each with resolution 2^m (i.e., each space V_{j+1} is contained in the next resolution space V_j). In multiresolution analysis, the set of subspaces satisfies the following properties:

1. $V_{m+1} \subset V_{m1}$, for all m: This property state that each subspaces is contained in the next resolution subspace.
2. $\overline{\bigcup V_m} = \mathcal{L}^2(\mathcal{R})$: This property indicates that the union of subspaces is dense in the space of square integrable functions $\mathcal{L}^2(\mathcal{R})$; $\mathcal{R}$ indicates a set of real numbers (upward completeness property).
3. $\bigcap V_m = 0$ (an empty set): This property is called downward completeness property.
4. $f(t) \in V_0 \leftrightarrow f(2^{-m}t) \in V_m$: Dilating a function from resolution space V_0 by a factor of 2^m results in the lower resolution space V_m (scale or dilation invariance property).
5. $f(t) \in V_0 \leftrightarrow f(t-n) \in V_0$: Combining this with the scale invariance property above, this property states that translating a function in a resolution space does not change the resolution (translation invariance property).
6. There exists a set $\{\phi(t-n) \in V_0$: n is an integer$\}$ that forms an orthogonal basis of V_0.

The basic tenet of multiresolution analysis is that whenever the above properties are satisfied, there exists an orthonormal wavelet basis $\psi_{m,n}(t) = 2^{-m/2}\psi(2^{-m}t - n)$ such that

$$P_{m-1}(f) = P_m(f) + \sum c_{m,n}(f)\psi_{m,n}(\mathrm{t}) \qquad (2.13)$$

where P_j is the orthonormal projection of ψ onto V_j. For each m, consider the wavelet functions $\psi_{m,n}(\mathrm{t})$ span a vector space W_m. It is clear from Eq. 2.13 that the wavelet that generates the space W_m and the scaling function that generates the space V_m are not independent. W_m is exactly the orthogonal complement of V_m in V_{m-1}. Thus, any function in V_{m-1} can be expressed as the sum of a function in V_m and a function in the wavelet space W_m. Symbolically, it is possible to express this as

$$V_{m-1} = V_m \oplus W_m \tag{2.14}$$

Since, m is arbitrary,

$$V_m = V_{m+1} \oplus W_{m+1} \tag{2.15}$$

Thus,

$$V_{m-1} = V_{m+1} \oplus W_{m+1} \oplus W_m \tag{2.16}$$

Continuing in this fashion, it is possible to establish that

$$V_{m-1} = V_k \oplus W_k \oplus W_{k-1} \oplus W_{k-2} \ldots \oplus W_m \tag{2.17}$$

for any $k \geq m$.

Thus, a function belonging to the space V_{m-1} (i.e., the function can be exactly represented by the scaling function at resolution $m - 1$), can be decomposed to a sum of functions starting with lower-resolution approximation followed by a sequence of functions generated by dilations of wavelet that represent the loss of information in terms of details. The successive levels of approximations can be considered as the representation of an image with fewer and fewer pixels. The wavelet coefficients can then be considered as the additional detail information needed to go from a coarser to a finer approximation. Hence, in each level of decomposition the signal can be decomposed into two parts, one is the coarse approximation of the signal in the lower resolution and the other is the detail information that was lost because of approximation. The wavelet coefficients derived by Eq. 2.9 or 2.10, therefore, describe the information (detail) lost when going from an approximation of the signal at resolution 2^{m-1} to the coarser approximation at resolution 2^m.

Implementation by Filters and the Pyramid Algorithm

Multiresolution analysis decomposes signal into two parts – one approximation of the original signal from finer to coarser resolution and the other detail information that was lost sue to the approximation. This can be represented as

$$f(\mathrm{t}) = \sum\nolimits_{\mathrm{n}} a_{m+1,n}\, \phi_{m+1,n} + \sum\nolimits_{\mathrm{n}} c_{m+1,n}\, \psi_{m+1,n} \qquad (2.18)$$

where f(t) denotes the value of input function f(t) at resolution 2^m, $c_{m+1,n}$ is the detail information, and $a_{m+1,n}$ is the coarser approximation of the signal at resolution 2^{m+1}. The functions, $\phi_{m+1,n}$ and $\psi_{m+1,n}$ are the dilation and wavelet basis functions (orthonormal).

In 1989, Mallat [9] proposed the multiresolution approach for wavelet decomposition of signals using a pyramidal filter structure of quadrature mirror filter (QMF) pairs. Wavelets developed by Daubechies [11, 12], in terms of discrete-time perfect reconstruction filter banks, correspond to IFR filters. In multiresolution analysis, it can be proven that decomposition of signals using the discrete wavelet transform can be expressed in terms of FIR filters and the algorithm for computation of the wavelet coefficients for the signal f(t) can be represented as

$$\left.\begin{aligned} c_{m,n}(f) &= \sum\nolimits_k g_{2n-k}\, a_{m-1,k}(f) \\ a_{m,n}(f) &= \sum\nolimits_k h_{2n-k}\, a_{m-1,k}(f) \end{aligned}\right\} \qquad (2.19)$$

where g and h are the high-pass and low-pass filters, $g_i = (-1)^i h_{-i+1}$ and $h_i = 2^{1/2} \int \phi(x - i)\ \phi(2x)dx$. Actually, $a_{m,n}(f)$ are the coefficients characteristics characterizing the projection of the function f(t) in the vector subspace V_m (i.e. approximation of the function in resolution 2^m), whereas $c_{m,n}(f) \in W_m$ are the wavelet coefficients (detail information) at resolution 2^m. If the input signal f(t) is in discrete sampled form, then it is possible to consider these samples as the highest order resolution approximation coefficients $a_{0,n}(f) \in V_0$ and Eq. 2.19 describes the multiresolution subband decomposition algorithm to construct $a_{m,n}(f)$ and $c_{m,n}(f)$ at level m with a low-pass filter h and high-pass filter g from $c_{m-1,n}(f)$, which were generated at level m-1. These filters are called the analysis filters. The recursive algorithm to compute DWT in different levels using Eq 2.19 is popularly called Mallat's Pyramid Algorithm. Since the synthesis filters h and g have been derived from the orthonormal basis functions ϕ and ψ, these filters give exact reconstruction

$$a_{m-1,i}(f) = \sum_n h_{2n-i}\, a_{m,n}(f) + \sum_n g_{2n-i}\, c_{m,n}(f) \qquad (2.20)$$

Most of the orthogonal wavelet basis functions have infinitely supported ψ and accordingly the filters h and g could be with infinitely many taps. However, for practical and computationally efficient implementation of the DWT for image processing applications, it is desirable to have infinite impulse response filters (FIR) with a small number of taps. It is possible to construct such filters by relaxing the orthonormality requirements and using biorthogonal basis functions. It should be noted that the wavelet filters are orthogonal when (h', g') = (h, g), otherwise it is biorthogonal. In such a case the filters (h' and g', called the synthesis filters) for reconstruction of the signal can be different than the analysis filters (h and g) for decomposition of the signals. In order to achieve exact reconstruction, construct the filters such that it satisfies the relationship of the synthesis filter with the analysis filter as shown in Eq. 2.21:

$$\left.\begin{aligned} g'_n &= (-1)^n h_{-n+1} \\ g_n &= (-1)^n h'_{-n+1} \\ \textstyle\sum_n h_n\, h'_{n+2k} &= \delta_{k,0} \end{aligned}\right\} \qquad (2.21)$$

If (*h'*, *g'*) = (*h*, *g*), the wavelet filters are called orthogonal, otherwise they are called biorthogonal. The popular (9, 7) wavelet filter adopted in JPEG2000 is one example of such a biorthogonal filter. The signal is still decomposed using Eq. 2.19, but the reconstruction equation is now done using the synthesis filters *h'* and *g'* as shown in Eq. 2.22:

$$a_{m-1,i}(f) = \sum_n a_{m,n}(f)\, h'_{2n-i} + \sum_n c_{m,n}(f)\, g_{2n-i} \qquad (2.22)$$

Let's summarize the DWT computation here in terms of simple digital FIR filtering. Given the input discrete signal *x(n)* (shown as a(0,n) in Figure 5), it is filtered parallelly by a low-pass (*h*) and a high-pass (*g*) at each transform level. The two output streams are then subsampled by simply dropping the alternate output samples in each stream to produce the low-pass subband y_L. The above arithmetic computation can be expressed as follows:

$$y_L(n) = \sum_{i=0}^{\tau_L-1} h(i)\ x(2n-i)\ ,\quad y_H(n) = \sum_{i=0}^{\tau_H-1} g(i)\ x(2n-i) \qquad (2.23)$$

where τ_L and τ_H are the lengths of the lengths of the low-pass (h) and high-pass (g) filters respectively. Since the low-pass subband a(1, n) is an approximation of the input, applying the above computation again on a(1,n) to produce the subbands a(2,n) and c(2,n) and so on. During the inverse transform to reconstruct the signal, both a(3,n) and c(3,n) are first upsampled by inserting zeros between two samples, and then they are filtered by low-pass (h') and high-pass (g') filters respectively. These two filtered output streams are added together to reconstruct a(2,n). The same continues until the reconstruction of the original signal a(0,n).

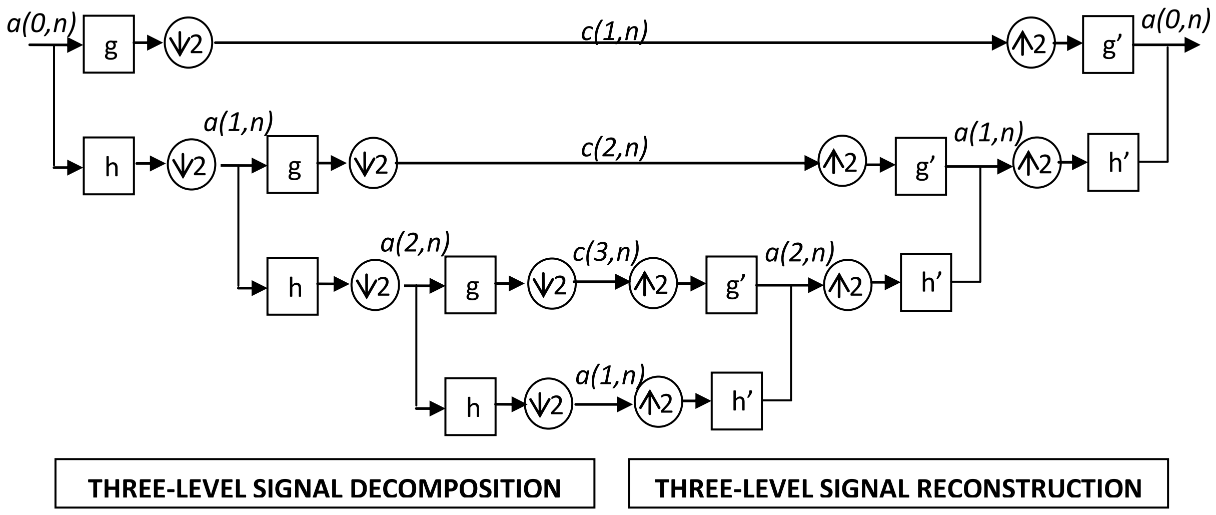
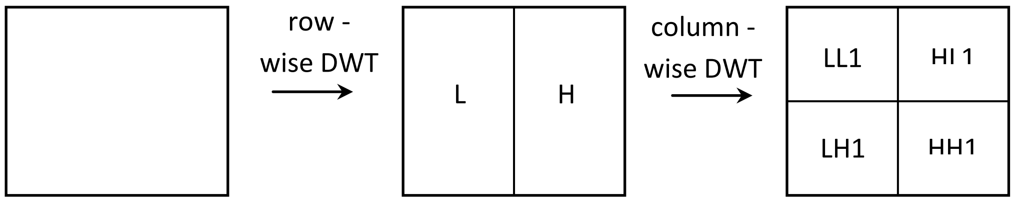


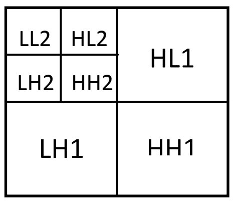
Figure 5 Three-level multiresolution wavelet decomposition and reconstruction and reconstruction of signals using pyramidal filter structure

Extension to Two-Dimensional Signals

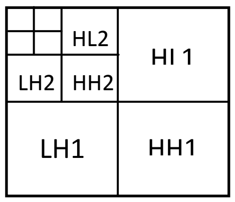
The two-dimensional extension of DWT is essential for transformation of two-dimensional signals, such as a digital image. A two-dimensional digital signal can be represented by a two-dimensional array X[M, N] with M rows and N columns, where M and N are nonnegative integers. The simple approach for two-dimensional implementation of the DWT is to perform the one-dimensional DWT row-wise to produce an intermediate result and then perform the same one-dimensional DWT column-wise on this intermediate result to produce the final result. This is shown in Figure 6(a). This is possible because the two-dimensional scaling functions can be expressed as separable functions which is the product of two-dimensional scaling function such as $\emptyset_2(x, y) = \emptyset_1(x)\emptyset_1(y)$. The same is true for the wavelet function ψ(x, y) as well. Applying the one-dimensional transform in each row, two subbands are produced in each row. When the low-frequency subbands of all the rows (L) are put together, it looks like a thin version (of size M x $\frac{N}{2}$) of the input signal as shown in Figure 6(a). Similarly put together the high-frequency subbands of all the rows to produce the H subband of size M x $\frac{N}{2}$, which contains mainly the high-frequency information around discontinuities (edges in an image) in the input signal. Then applying



(a) First level of decomposition



Second level decomposition



Third level decomposition

Figure 6 Row - Column computation of two-dimensional DWT

a one-dimensional DWT column-wise on these L and H subbands (intermediate result), four subbands LL, LH, HL, and HH of size $\frac{M}{2} \times \frac{N}{2}$ are generated as shown in Figure 6(a). LL is a coarser version of the original input signal. LH, HL, and HH are the high frequency subband containing the detail information. It is also possible to apply one-dimensional DWT column-wise first and then row-wise to achieve the same result. Figure 7 comprehends the idea describe above.

The multiresolution decomposition approach in the two-dimensional signal is demonstrated in Figures 6(b) and (c). After the first level of decomposition, it generates four subbands LL1, HL1, LH1, and HH1 as shown in Figure 6(a). Considering the input signal is an image, the LL1 subband can be considered as a 2:1 subsampled (both horizontally and vertically) version of image. The other three subbands HL1, LH1, and HH1 contain higher frequency detail information. These spatially oriented (horizontal , vertical or diagonal) subbands mostly contain information of local discontinuities in the image and the bulk of the energy in each of these three

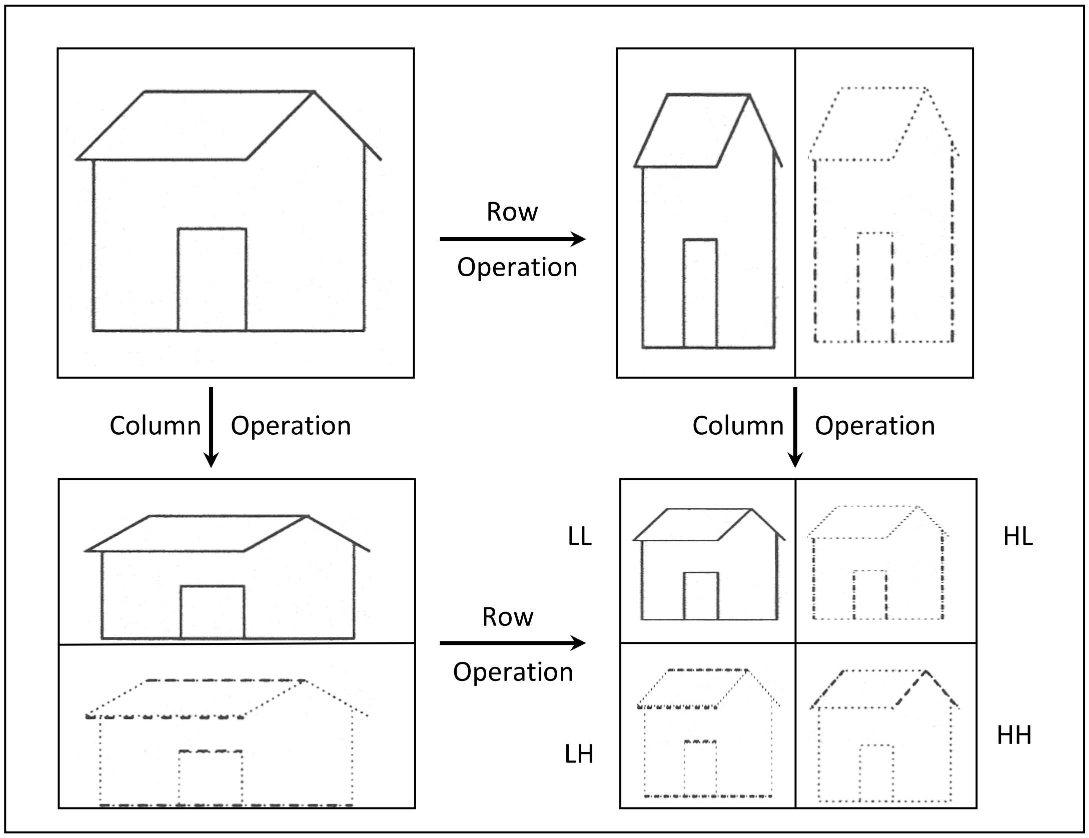


Figure 7 Extension of DWT in two - dimensional signals.

subbands is concentrated in the vicinity of areas corresponding to edge activities in the original image. Since LL1 is acoarser approximation of the input, it has similar spatial and statistical characteristics to the original image. As a result, it can be further decomposed into four subbands LL2, LH2, HL2 and HH2 as shown in Figure 6(b) based on the principle of multiresolution analysis. Accordingly the image is decomposed into 10 subbands LL3, LH3, HL3, HH3, HL2, LH2, HH2, LH1, HL1 and HH1 after three levels of pyramidal multiresolution subband decomposition, as shown in Figure 6(c). The same computation can continue to further decompose LL3 into higher levels.

Chapter 3

Source Coding Algorithm

Source coding can mean both lossless and lossy compression. Depending on the characteristics of the data, each algorithm may give different compression performance. So selection of the particular algorithm will depend upon the characteristics of the data themselves. In a lossy compression mode, the source coding algorithms are usually applied in the entropy encoding step after transformation and quantization.

Run-Length Coding

The neighbouring pixels in a typical image are highly correlated to each other. Often it is observed that the consecutive pixels in a smooth region of an image are identical or the variation among the neighbouring pixels is very small. Appearance of runs of identical values is particularly true for binary images where usually the image consists of runs of 0's or 1's. Even if the consecutive pixels in grayscale or colour images are not exactly identical but slowly varying, it can often be pre-processed and the consecutive processed pixel values become identical. If there is a long run of identical pixels, it is more economical to transmit the length of the run associated with the particular pixel value instead of encoding individual pixel values.

Run-length coding is a simple approach to source coding when there exists a long run of the same data, in a consecutive manner, in a data set. As an example, the data *d* = 5 5 5 5 5 5 5 19 19 19 19 19 19 19 19 19 19 19 19 0 0 0 0 0 0 0 0 23 23 23 23 23 23 contains long runs of 5's, 19's, 0's, 23's etc. Rather than coding each sample in the run individually, the data can be represented compactly by simply indicating the value of the sample and the length of its run when it appears. In this manner the data *d* can be run-length encoded as (5 7) (19 12) (0 8) (23 6). Here the first value represents the pixel, while the second indicates the length of its run.

In some cases, the appearance of runs of symbols may not be very apparent. But the data can possibly be pre-processed in order to aid run-length coding. Consider the data *d* = 26 29 32 35 38 41 44 50 56 62 68 78 88 98 108 118 116 114 112 110 108 106 104 102 100 98 96. A simple pre-process on this data, by taking the sample difference $e(i) = d(i) – d(i-1)$, to produce the processed data *e'* = 26 3 3 3 3 3 3 6 6 6 6 10 10 10 10 10 -2 -2 -2 -2 -2 -2 -2 -2 -2 -2 -2. This pre-processed data can now be easily run-length encoded as (26 1) (3 6) (6 4) (10 5) (-2 11). A variation of this technique is applied in the baseline JPEG standard for still-picture compression. The same technique can be applied to numeric databases as well.

On the other hand, binary (black and white) images, such as facsimile, usually consist of 0's and 1's. As an example, if a segment of a binary image is represented as d = 00000000011111111111000000000000000111000000000000001001111111111, it can be compactly represented as $c(d)$ = (9, 11, 15, 3, 13, 1, 2, 10) by simply listing the lengths of alternate runs of 0's and 1's. While the original binary data d requires 65 bits for storage, its compact representation $c(d)$ requires 32 bits only under the assumption that each length of run is being represented by 4 bits.

Huffman Coding

From Shannon's Source Coding Theory, it is known that a source can be coded with an average code length close to the entropy of the source. In 1952, D.A. Huffman [13] invented a coding technique to produce the shortest possible average code length given the source symbol set and the associated probability of occurrence of the symbols. Codes generated using these coding techniques are popularly known as Huffman codes. Huffman coding technique is based on the following two observations regarding optimum prefix codes.

- The more frequently occurring symbols can be allocated with shorter code words than the less frequently occurring symbols.
- The two least frequently occurring symbols will have codewords of the same length, and they differ only in the least significant bit.

Average length of these codes is close to entropy of the source.

Assume that there are m source symbols $\{s_1, s_2, ..., s_m\}$ with associated probability of occurrence $\{p_1, p_2, ..., p_m\}$. Using these probability values, generate a set of Huffman codes of the source symbols. The Huffman codes can be mapped into a binary tree, popularly known as the Huffman tree. The algorithm to generate Huffman tree and hence the Huffman codes of the source symbols can be shown as below:

i) Produce a set $N = \{N_1, N_2, ..., N_m\}$ of m nodes as leaves of a binary tree. Assign a node Ni with the source symbol s_i, i = 1, 2, ..., m and label the node with the associated probability p_i.
ii) Find the two nodes with the two lowest probability symbols from the current node set, and produce a new node as a parent of these two nodes.

iii) Label the probability of this new parent node as the sum of the probabilities of its two child nodes.

iv) Label the branch of one child node of the new parent node as 1 and the branch of the other child node as 0.

v) Update the node set by replacing the two child nodes with smallest probabilities by the newly generated parent node. If the number of nodes remaining in the node set is greater than 1, go to Step (ii).

vi) Transverse the generated binary tree from the root node to each leaf node N_i, $i = 1, 2, ...,$ m, to produce the codeword of the corresponding symbol si, which is a concatenation of the binary labels (0 or 1) of the branches from the root to the leaf node.

Figure 8 shows the Huffman tree construction for eight symbols with their probability of occurrence of each symbol is indicated in the associated parentheses and corresponding Huffman code table is shown in Table 1.

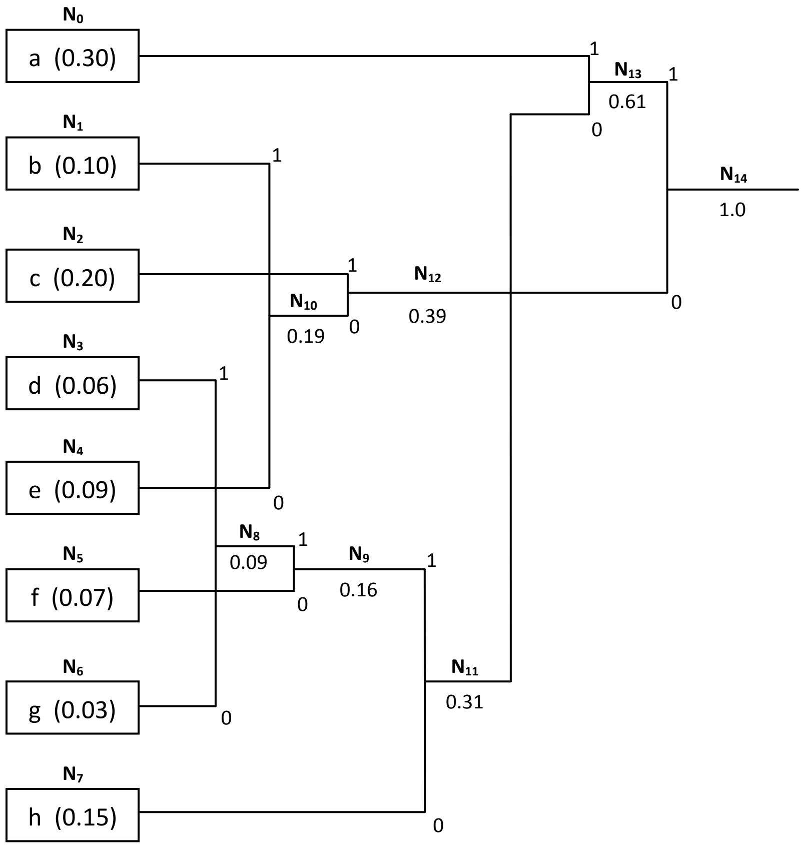


Figure 8 Huffman tree construction

Limitations of Huffman Coding

i) Huffman code is optimal only if exact probability distribution of the source symbols is known.

ii) Each symbol is encoded with integer number of bits.

iii) Huffman coding is not efficient to adapt with the changing source statistics.

iv) The length of the codes of the least probable symbol could be very large to store into a single word or basic storage unit in a computing system.

Table 1 Huffman Code Table

Symbol	Probability	Huffman Code
a	0.30	1 0
b	0.10	0 0 1
c	0.20	0 1
d	0.06	1 1 1 1 1
e	0.09	0 0 0
f	0.07	1 1 1 0
g	0.03	1 1 1 1 0
h	0.15	1 1 0

Arithmetic Coding

Arithmetic coding is a variable-length source encoding technique [14]. In traditional entropy encoding techniques such as Huffman coding, each input symbol in a message is substituted by a specific code specified by an integer number of bits. Arithmetic coding deviates from this paradigm. In arithmetic coding a sequence of input symbols is represented by an interval of real numbers between 0.0 and 1.0. The longer the message, the smaller the interval to represent the message becomes. More probable symbols reduce the interval less than the less probable symbols and hence add fewer bits in the encoded message. As a result, the coding result can reach to Shannon's entropy limit for a sufficiently large sequence of input symbols as long as the statistics are accurate.

Arithmetic coding offers superior efficiency and more flexibility compared to the popular Huffman coding. It is particularly useful when dealing with sources with small alphabets such as binary alphabets and alphabets with highly skewed probabilities. Huffman coding cannot achieve any compression for a source of binary alphabets. As a result arithmetic coding is highly efficient for coding bi-level images. However, arithmetic coding is more complicated and is intrinsically less error resilient compared to the Huffman coding. The arithmetic coding requires significantly higher computation because of the requirement of multiplication to compute the intervals. However several multiplication-free arithmetic coding technique have been developed for binary compression [15].

Encoding Algorithm

The arithmetic coding algorithm is explained here with an example. Consider a four-symbol alphabet $A = \{a, b, c, d\}$ with the fixed symbol probabilities $p(a) = 0.3$, $p(b) = 0.2$, $p(c) = 0.4$, and $p(d) = 0.1$ respectively. The symbol probabilities can be expressed in terms of partition of the half-open range [0.0, 1.0), as shown in Table 2.

Table 2 Probability Model

Index	Symbol	Probability	Cumulative Probability	Range
1	a	0.3	0.3	[0.0, 0.3)
2	b	0.2	0.5	[0.3, 0.5)
3	c	0.4	0.9	[0.5, 0.9)
4	d	0.1	1.0	[0.9, 1.0)

The algorithm for arithmetic coding is presented below. In this algorithm, N is considered as the length of the message (i.e. total number of symbols in the message); $F(i)$ is the cumulative probability of ith source symbol as shown in Table 2.

"c a c b a d" is the message to be encoded using the above fixed model of probability estimates. At the beginning of both encoding and decoding processes, the range for the message is the entire half-open interval [0.0, 1.0), which can be partitioned into disjoint subintervals or ranges [0.0, 0.3), [0.3, 0.5), [0.5, 0.9) and [0.9, 1.0) corresponding to the symbols a, b, c, and d respectively, as by the range R(start) [0.0 1.0) the symbol probabilities stipulated by the probability model. As each symbol in the message is processed, the range is narrowed down by the encoder as explained in the algorithm. Since the first symbol of the message is c, the range is first narrowed down to the half-open interval R(c) = [0.5, 0.9). This range is further partitioned into exactly the same proportions as the original one, yielding the four half-open disjoint intervals [0.5, 0.62). [0.62, 0.70), [0.70, 0.86), and [0.86, 0.90) corresponding to a, b, c and d respectively. As a result, the range is narrowed down to R(c a) = [0.5, 0.62) when the second symbol a in the message is processed. This new range [0.5, 0.62) is now partitioned into four disjoint intervals [0.5, 0.536), [0.536, 0.560), [0.560, 0.608), and [0.608, 0.62). After processing the third symbol, c, the range is accordingly narrowed down to R(c a c) = [0.560, 0.608). This is again partitioned into [0.560, 0.5744), [0.5744, 0.5840), [0.5840, 0.6032), and [0.6032, 0.608) in order to process the next symbol in the message. After processing the fourth symbol, b, the range is now narrowed down to R(c a c b) = [0.5744, 0.5840). This is again partitioned into four intervals [0.5744, 0.57728), [0.5728, 0.57920), [0.57920, 0.58304), and [0.58304, 0.584) corresponding to the symbols a, b, c, and d respectively. After processing the fifth symbol, a, the range is now narrowed down to R(c a c b a) = [0.5744, 0.57728). This is further partitioned into the disjoint intervals [0.5744, 0.575264), [0.575264, 0.575840), [0.575840, 0.576992) and [0.576992, 0.57728). The last symbol in the message is d and hence the final range for the message becomes R(c a c b a d) = [0.576992, 0.57728). As a result, the message “c a c b a d” can be encoded by any number in the range [0.576992, 0.57728) because it is not necessary for the decoder to know both ends of the range produced by the encoder. If the midpoint of the interval is used, the encoded value will be 0.577136.

Decoding Algorithm

Both the encoder and the decoder have the same probability model. Initially the decoder starts with the range [0.0, 1.0), which is partitioned into four intervals [0.0, 0.3), [0.3, 0.5), [0.5, 0.9), and [0.9, 1.0) corresponding to the symbols *a, b, c,* and *d* in the alphabet. As soon as the decoder receives an encodes number 0.577, it can immediately decode that the first symbol of the message is *c* because the number 0.577 belongs to the range [0.5, 0.9) and the range is narrowed down to [0.5, 0.9) and partitioned into [0.5, 0.62), [0.62, 0.70), [0.70, 0.86) and [0.86, 0.9) in a similar fashion as the encoder. Since the number 0.577 belong to the range [0.56, 0.62), it can be immediately decode second symbol *a*. The range is now narrowed down to [0.5, 0.62) and partitioned into [0.5, 0.536), [0.536, 0.560), [0.560, 0.608), and [0.608, 0.62). Since the number 0.577 belongs to the range [0.560, 0.608), the decoder can decode the third symbol to be *c*. The range is now narrowed down to [0.560, 0.608) and partitioned into the four subintervals [0.560, 0.5744), [0.5744, 0.5840), [0.5840, 0.6032), and [0.6032, 0.608). Since the number is 0.577 belongs in the range [0.5744, 0.584), the decoder deduces that the next symbol is *b* and narrows the range down to [0.5744, 0.584). The range is now subdivided into [0.5744, 0.57728), [0.5728, 0.57920), [0.57920, 0.58304), and [0.58304, 0.584). Since the number 0.577 belongs within the range [0.5744, 0.57728), the next symbol decoded is *a* and the range is narrowed down to [0.5744, 0.57728) and partitioned into four subintervals [0.5744, 0.575264), [0.575264, 0.575840), [0.575840, 0.576992) and [0.576992, 0.57728). Since 0.577 belongs to the range [0.576992, 0.57728), it is very natural that the decoders decode the next symbol to be *d* and narrows the range down to [0.576992, 0.5770784), [0.5770784, 0.577136), [0.577136, 0.5772512, and [0.5772512, 0.57728) respectively. Hence the decoder could uniquely decode the message "*c a c b a d*" until this step. If the decoder is aware of the length of the message, it can stop decoding here. Otherwise, it can continue decoding the next symbol to be *a* because 0.577 belongs to the range [0.576992, 0.5770784) and so on indefinitely. To resolve the ambiguity, it is possible to ensure that each message ends with a special terminating symbol know to both encoder and decoder. In this example, if *d* is assumed to be the special terminating symbol, the decoder will effectively stop after decoding the message "c a c b a d". Otherwise the length of the original message needs to be known to the decoder in order to stop decoding effectively.

Limitations of Arithmetic Coding

i) The encoded value is not unique because any value within the final range can be considered as the encoded message. It is desirable to have a unique binary code for the encoded message.

ii) The encoding algorithm does not transmit anything until encoding of the entire message has been completed. As a result, the decoding algorithm cannot start until it has received the complete encoded data. The above two limitations can be overcome by using the binary arithmetic coding which will be described in the next section.

iii) The precision required to represent the intervals grows with the length of the message.

iv) Use of the multiplications in the encoding and decoding process, in order to compute the ranges in every step, may be prohibitive for many real time fast applications.

Chapter 4

JPEG2000 Standard

Introduction

JPEG2000 is the new international standard for image compression [16, 17, 18] developed jointly by the International Organization for Standardization (ISO) and the International Electrotechnical Commission (IEC) and also recommended by International Telecommunications Union (ITU). A fundamental shift in the image compression approach came after the Discrete Wavelet Transform (DWT) became popular. Exploiting the interesting features in DWT, many scalable image compression algorithms were proposed in the literature [3, 7, 8, 19, 20, 21, 22]. To overcome the inefficiencies in the JPEG[23] standard and serve emerging applications areas in this age of mobile and Internet communications, the new JPEG2000 standard has been developed based on the principles of DWT and currently more developments in this standard are still in progress in the ISO/IEC standard committee. It incorporated the last advances in the image compression to provide a unified optimized tool to accomplish both lossless and lossy compression and decomposition using the same algorithm and the bit stream syntax. The systems architecture is not only optimized for compression efficiency at even very low bit-rates, it is also optimized for scalability and interoperability in the networks and noisy mobile environments. The JPEG2000 standard will be effective in wide application areas such as internet, digital photography, digital library, image archival, compound documents, image databases, colour reprography (photocopying, printing, scanning, facsimile), graphics, medical imagery, mobile multimedia communication, 3G cellular telephony, client-server networking, e-commerce, etc.

The main drawback of the JPEG2000 standard compared to current JPEG is that the coding algorithm is much more complex and the computational needs are much higher. Moreover, bit-plane-wise computing may restrict good computational performance with a general-purpose computing platform. Analysis shows that JPEG2000 is more than 30 times complex as compared with current JPEG. As a result, there is a tremendous need to develop high-performance architectures and special-purpose custom VLSI chips exploiting the underlying data parallelism to speed up the DWT and entropy encoding phase of JPEG2000 to make it suitable for real-time applications.

Salient Features of JPEG2000

Some of the salient features offered by the JPEG2000 standard that are effective in vast areas of applications are as follows:

- *Superior low bit-rate performance*: It offers superior performance in terms of visual quality and PSNR (peak signal-to-noise ratio) at very low bit-rates (below 0.25 bit/pixel) compared to the baseline JPEG. For equivalent visual quality JPEG2000 achieves more compression compared to JPEG. This feature is very useful for transmission of compressed images through a low-bandwidth transmission channel.

- *Continuous tone and bi-level image compression:* The JPEG2000 standard is capable of compressing and decompressing both the continuous tone (grayscale and colour) and bi-Ievel images. The JBIG2 standard was defined to compress the bi-Ievel images and it uses the same MQ-coder that is used to entropy encode the wavelet coefficients of the grayscale or colour image components.

- *Large dynamic range of the pixels:* The JPEG2000 standard-compliant systems can compress and decompress images with various dynamic ranges for each colour component. Although the desired dynamic range for each component in the requirement document is 1 to 16 bits, the system is allowed to have a maximum of 38 bits precision based on the bitstream syntax. As a matter of fact, JPEG2000 is the only standard that can deal with pixels with more than 16 bits precision. This feature is particularly suitable both for software and hardware implementers to choose the precision requirement for targeted applications.

- *Large images and large numbers of image components:* The JPEG2000 standard allows the maximum size of an image to be (2^{32} - 1) x (2^{32} - 1) and the maximum number of components in an image to be 2^{14}. This feature is particularly suitable for satellite imagery and astronomical image processing involving multispectral images with a large number of components and size.

- *Lossless and lossy compression:* The single unified compression architecture can provide both the lossless and the lossy mode of image compression. Lossy and lossless decompressions are also possible from a single compressed bitstream. The reversible colour transform and the reversible wavelet transform (using integer wavelet filter coefficients) make the lossless compression possible by the same coding architecture. As a result, the same technology is applicable in varying applications areas ranging from medical imagery requiring lossless compression to digital transmission of images through communication networks.

- *Fixed size can be preassigned:* The JPEG2000 standard allows users to select a desired size of the compressed file. This is possible because of the bit-plane coding of the architecture and controlling the bit-rate through the rate control. The compression can continue bit-plane by bit-plane in all the code-blocks until the desired compressed size is achieved and the compression process can terminate. This is a very useful feature for restricted-buffer-size hardware implementation as in reprographic architectures such as printer, photocopier, scanner, etc. This is also a very useful feature to dynamically control the size of the compressed file in a limited-bandwidth communications networking environment.

- *Progressive transmission by pixel accuracy and resolution:* Using the JPEG2000 standard, it is possible to organize the code-stream in a progressive manner in terms of *pixel accuracy* (i.e., visual quality or SNR) of images that allows reconstruction of images with increasing pixel accuracy as more and more compressed bits are received and decoded. This is possible by progressively decoding most significant bit-plane to lower significant bit-planes until all the bit-planes are reconstructed. The code-stream can also be organized as progressive in resolution such that the higher-resolution images are generated as more compressed data are received and decoded. This is possible by decoding and inverse DWT of more and more higher level subbands that were generated by the multiresolution decomposition of the image by DWT. These features are very effective for real-time browsing of images on the Web, downloading or reconstructing the images in a system with limited memory buffer, transmission of images through limited-bandwidth channels, decoding the images depending on the available resolution of the rendering system, etc

- *Region of interest (ROI) coding:* Sometimes it may be desired certain parts of an image that are of greater importance to be encoded with higher fidelity compared to the rest of the image. During decompression the quality of the image also can be adjusted depending on the degree of interest in each region of interest.

- *Random access and compressed domain processing:* By randomly extracting the code-blocks from the compressed bitstream, it is possible to manipulate certain areas (or regions of interest) of the image. Some of the examples of compressed-domain processing could be cropping, flipping, rotation, translation, scaling, feature extraction, etc. One might want to replace one object in the image with another, sometimes even with a synthetically generated image object. It is possible to extract the compressed code-blocks representing the object and replace them with compressed code-blocks of the desired object. This feature is very useful in many applications areas such as editing, studio, animation, graphics, etc.

- *Robustness to bit-errors (error resiliency):* Robustness to bit-errors is highly desirable for transmission of images over noisy communications channels. The JPEG2000 standard facilitates this by coding small size independent code-blocks and including resynchronization markers in the syntax of the compressed bitstream. There are also provisions to detect and correct errors within each code-block. This feature makes JPEG2000 applicable in emerging third-generation mobile telephony applications.

- *Sequential buildup capability:* The JPEG2000-compliant system can be designed to encode an image from top to bottom in a single sequential pass without the need to buffer an entire image, and hence is suitable for low-memory on-chip VLSI implementation. The line-based implementation of DWT and tiling of the images facilitates this feature.

Parts of the JPEG2000 Standard

The standard has 11 parts (because Part 7 has been abandoned) with each part adding new features to the core standard in Part 1. The 11 parts and their features are as follows:

- Part 1-Core Coding System [24] is now published as an International Standard ISO/IEC 15444-1:2000, and this part specifies the basic feature set and code-stream syntax for JPEG2000 .

- Part 2-Extensions [25] to Part 1. This part adds a lot more features to the core coding system.

- Part 3-Motion JPEG2000 [26] specifies a file format (MJ2) that contains an image sequence encoded with the JPEG2000 core coding algorithm for motion video. It is aimed at applications where high-quality frame-based compression is desired.

- Part 4-Conformance Testing [27] is now published as an International Standard (ISO/IEC 15444-4:2002). It specifies compliance-testing procedures for encoding/decoding using Part 1 of JPEG2000.

- Part 5-Reference Software [28]. In this part, two software source packages (using Java and C programming languages) are provided for the purpose of testing and validation for JPEG2000 systems implemented by the developers.

- Part 6-Compound Image File Format [29] specifies another file format (JPM) for the purpose of storing compound images. The ITU-T T.4411S0 16485 multilayer Mixed

Raster Content (MRC) model is used to represent a compound image in Part 6 of JPEG2000.

- *Part* 7-This part has been abandoned.

- Part 8-Secure JPEG2000 (JPSEC). This part deals with security aspects for JPEG2000 applications such as encryption, watermarking, etc.

- Part 9-lnteractivity Tools, APls and Protocols (JPIP). This part defines an interactive network protocol, and it specifies tools for efficient exchange of JPEG2000 images and related metadata.

- Part 10-3-D and Floating Point Data (JP3D). This part is developed with the concern of three-dimensional data such as 3-D medical image reconstruction.

- Part 11-Wireless (JPWL). This part is developed for wireless multimedia applications. The main concerns for JPWL are error protection, detection, and correction for JPEG2000 in an error-prone wireless environment.

- Part 12-ISO Base Media File Format has a common text with ISO/IEC 14496-12 for MPEG-4.

Overview of the JPEG2000 Part 1 Encoding System

Once the encoder system is well understood, it becomes easier to comprehend the decoder system described in the standard document. This section explains the encoder engine for the JPEG2000 Part 1 standard. The whole compression system is simply divided into three phases namely, (1) image preprocessing, (2) compression, and (3) compressed bitstream formation. The functionalities of these three phases is explained in the following sections.

Image Preprocessing

The image preprocessing phase consists of three optional major functions: first tiling, then DC level shifting, followed by the multicomponent transformation.

Tiling

The first preprocessing operation is *tiling.* In this step, the input source image is (optionally) partitioned into a number of rectangular nonoverlapping blocks if the image is very large. Each of these blocks is called a *tile.* All the tiles have exactly the same dimension except the tiles at the image boundary if the dimension of the image is not an integer multiple of the dimension of the tiles. The tile sizes can be arbitrary up to the size of the original image. For an image with multiple components, each tile also consists of these components. For a grayscale image, the tile has a single component. Since the tiles are compressed independently, visible artifacts may be created at the tile boundaries when it is heavily quantized for very-low-bit-rate compression as typical in any block transform coding. Smaller tiles create more boundary artifacts and also degrade the compression efficiency compared to the larger tiles. Obviously, no tiling offers the best visual quality. On the other hand, if the tile size is too large, it requires larger memory buffers for implementation either by software or hardware. For VLSI implementation, it requires large on-chip memory to buffer large tiles mainly for DWT computation. The tile size 256 x 256 or 512 x 512 is found to be a typical choice for VLSI implementation based on the cost, area, and power consideration. With the advances in memory technology with more compaction and reducing cost, the choice of tile size in the near future will be accordingly larger.

DC level Shifting

Originally, the pixels in the image are stored in unsigned integers. For mathematical computation, it is essential to convert the samples into two's complement representation before any transformation or mathematical computation starts in the image. The purpose of DC level shifting (optional) is to ensure that the input image samples have a dynamic range that is approximately centred around the zero. The DC level shifting is performed on image samples that are represented by unsigned integers only. All samples $I_i(x, y)$ in the ith component of the image (or tile) are level shifted by subtracting the same quantity $2^{S_{siz}^{i}-1}$ to produce the DC level shifted sample $I_i^{'}(x, y)$ as follows,

$$I_i^{'}(x, y) \leftarrow I_i(x, y) - 2^{S_{siz}^{i}-1}$$

where $2^{S_{siz}^{i}-1}$ is the precision of image samples signalled in the SIZ (image and tile size) marker segment in compressed bitstream. For images whose samples are represented by signed integers, such as CT (computed tomography) images, the dynamic range is already centred about zero, and no DC level shifting is required.

Multicomponent Transformations

The multicomponent transform is effective in reducing the correlations (if any) amongst the multiple components in a multi component image. This results in reduction in redundancy and increase in compression performance. Actually, the standard does not consider the components as colour planes and in that sense the standard itself is colourblind. However, it defines an optional multicomponent transformation in the first three components only. These first three components can be interpreted as three colour planes (R, G, B) for ease of understanding. That's why they are often called multicomponent colour transformation as well. However, they do not necessarily represent Red-Green-Blue data of a colour image. In general, each component can have different bit-depth (precision of each pixel in a component) and different dimension. However, the condition of application of multi component transform is that the first three components should have identical bit-depth and identical dimension as well.

The JPEG2000 Part 1 standard supports two different transformations: (1) reversible colour transform (RCT), and (2) irreversible colour transform (ICT). The RCT can be applied for both lossless and lossy compression of images. However, ICT is applied only in lossy compression.

Reversible Colour Transformation: For lossless compression of an image, only the reversible colour transform (RCT) is allowed because the pixels can be exactly reconstructed by the inverse RCT. Although it has been defined for lossless image compression, the standard allows it for lossy compression as well. In case of lossy compression, the errors are introduced by the transformation and/or quantization steps only, not by the RCT. The forward RCT and inverse RCT are given by:

Forward RCT:
$$\left.\begin{aligned} Y_r &= \left\lfloor \frac{R+2G+B}{4} \right\rfloor \\ U_r &= B - G \\ V_r &= R - G \end{aligned}\right\} \qquad 4.1$$

Inverse RCT:

$$\left.\begin{aligned} G &= Y_r - \left\lfloor \frac{U_r + V_r}{4} \right\rfloor \\ R &= V_r + G \\ B &= U_r + G \end{aligned}\right\} \qquad 4.1$$

Irreversible Colour Transformation: The irreversible colour transformation (ICT) is applied for lossy compression only because of the error introduced due to forward and inverse transformation by using noninteger coefficients as the weighting parameters in the transformation matrix, as shown in Eqs. 4.3 and 4.4. The ICT is the same as the luminance-chrominance colour transformation used in baseline JPEG. Y is the luminance component of the image representing intensity of the pixels (light) and *Cb* and *Cr* are the two chrominance components representing the colour information in each pixel. In baseline JPEG, the chrominance components can be subsampled to reduce the amount of data to start with. However, in the JPEG2000 standard, this subsampling is not allowed. The forward ICT and inverse ICT are given by:

Forward ICT:

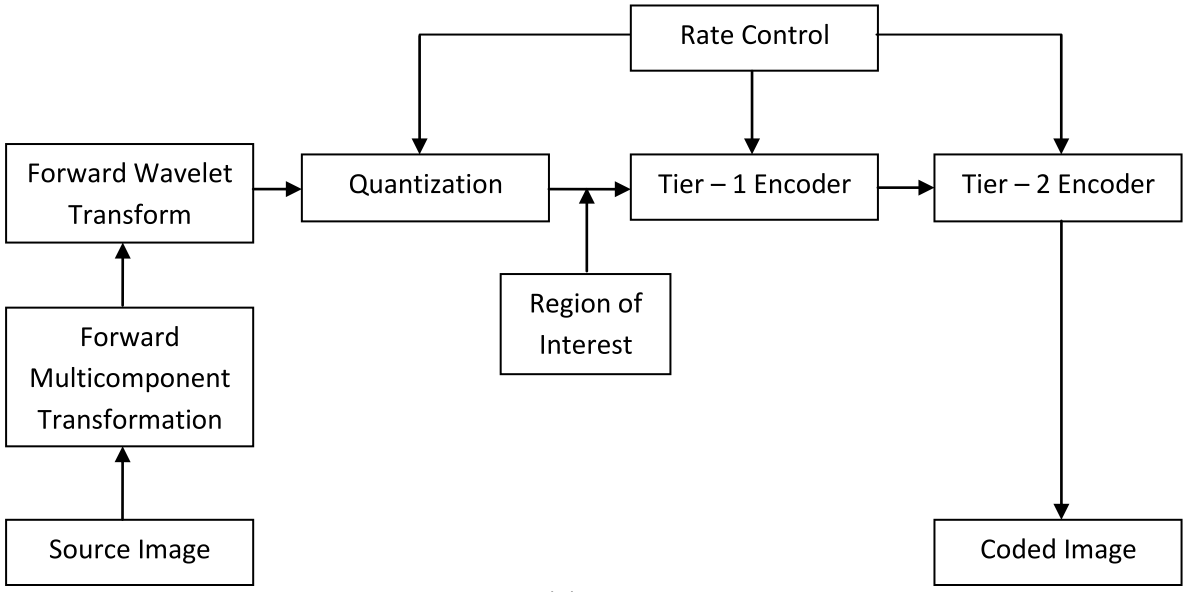
$$\left.\begin{bmatrix} Y \\ Cb \\ Cr \end{bmatrix} = \begin{bmatrix} 0.299000 & 0.587000 & 0.114000 \\ -0.168736 & -0.331264 & 0.500000 \\ 0.500000 & -0.418688 & -0.081312 \end{bmatrix} \begin{bmatrix} R \\ G \\ B \end{bmatrix}\right\} \qquad 4.3$$

Inverse ICT:

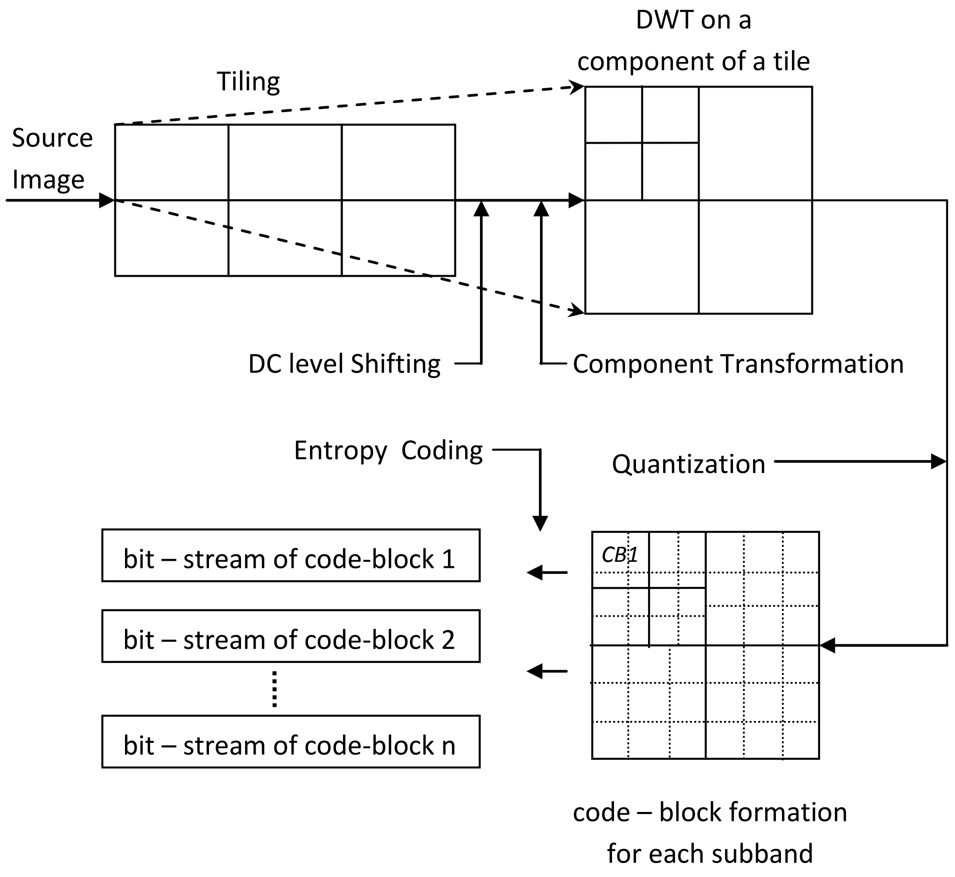
$$\begin{bmatrix} R \\ G \\ B \end{bmatrix} = \begin{bmatrix} 1.0 & 0.0 & 1.402000 \\ 1.0 & -0.344136 & -0.714136 \\ 1.0 & 1.772000 & 0.0 \end{bmatrix} \times \begin{bmatrix} Y \\ Cb \\ Cr \end{bmatrix} \quad \Big\} \quad 4.4$$

Compression

After the optional preprocessing phase, the compression phase actually generates the compressed code. The computational block diagram of the functionalities of the compression system is shown in Figure 9(a). The data flow of the compression system is shown in Figure 9(b). As shown in Figure 9(b), each preprocessed component is independently compressed and transmitted as shown in Figure 9(a). The compression phase is mainly divided into three



(a)


DWT on a component of a tile
Tiling
Source Image
DC level Shifting
Component Transformation
Entropy Coding
Quantization
bit – stream of code-block 1
bit – stream of code-block 2
bit – stream of code-block n
CB1
code – block formation for each subband

(b)

Figure 9 (a) Block diagram of the JPEG2000; (b) dataflow

sequential steps: (1) Discrete Wavelet Transform (DWT), (2) Quantization, and (3) Entropy Encoding. After preprocessing, each component is independently analyzed by a suitable discrete wavelet transform (DWT). The DWT essentially decomposes each component into a number of sub bands in different resolution levels. Each subband is then independently quantized by a quantization parameter, in case of lossy compression. The quantized subbands are then divided into a number of smaller code-blocks of equal size, except for the code- blocks at the boundary of each subband. Typical size of the code-blocks is usually 32 x 32 or 64 x 64 for better memory handling and is very suitable for VLSI implementation with on-chip memory in the encoder architecture. The standard allows the limit of code-block sizes with some restrictions. Each code-block is then entropy encoded independently to produce compressed bitstreams as shown in the dataflow diagram in Figure 9(b). The three major functions in the compression phase is discussed in the following sections.

Discrete Wavelet Transformation

The key difference between current JPEG and JPEG2000 starts with the adoption of discrete wavelet transform (DWT) instead of the 8 x 8 block based discrete cosine transform (DCT). The DWT essentially analyzes a tile (image) component to decompose it into a number of subbands at different levels of resolution. The two-dimensional DWT is performed by applying the one-dimensional DWT row-wise and then column-wise in each component. In the first level of decomposition, four subbands LL1, HL1, LH1, and HHI are created. The low-pass subband (LL1) represents a 2:1 subsampled in both vertical and horizonal directions, a low-resolution version of the original component. This is an approximation of the original image in subsampled form. The other subbands (HL1, LH1, and HH1) represent a downsampled residual version (error because of coarser approximation) of the original image needed for the perfect reconstruction of the original image. The LLI subband can again be analyzed to produce four subbands LL2, HL2, LH2, and HH2, and the higher level of decomposition can continue in a similar fashion. Typically, it won’t give much compression benefit after five levels of decomposition in natural images. However, theoretically it can go even further. The maximum number of levels of decomposition allowed in Part 1 is 32. In Part 1 of the JPEG2000 standard, only power of 2 dyadic decomposition in multiple levels of resolution is allowed. The standard supports both the convolution and the lifting-based approach for DWT. Next section presents the two default wavelet filter pairs supported by Part 1 of the JPEG2000 standard.

Discrete Wavelet Transformation for Lossy Compression: For lossy compression, the default wavelet filter used in the JPEG2000 standard is the Daubechies (9, 7) *biorthogonal spline* filter. By (9, 7) we indicate that the analysis filter is formed by a 9-tap low-pass FIR filter and a 7-tap high-pass FIR filter. Both filters are symmetric. The analysis filter coefficients (for forward transformation) are as follows:

- 9-tap low-pass filter : $[h_{-4},\ h_{-3},,\ h_{-2},\ h_{-1},\ h_0,\ h_1,\ h_2,\ h_3,\ h_4]$

$h_4 = h_{-4}$ = +0.026748757410810

$h_3 = h_{-3}$ = -0.016864118442875

$h_2 = h_{-2}$ = -0.078223266528988

$h_1 = h_{-1}$ = +0.266864118442872

h_0 = +0.602949018236358

- 7-tap high-pass filter: $[\ g_{-3},\ g_{-2},\ g_{-1},\ g_0,\ g_1,\ g_2,\ g_3]$

$g_3 = g_{-3}$ = +0.0912717631142495

$g_2 = g_{-2}$ = -0.057543526228500

$g_1 = g_{-1}$ = -0.591271763114247

g_0 = +1.115087052456994

For the synthesis filter pair used for inverse transformation, the low-pass FIR filter has seven filter coefficients and the high-pass FIR filter has nine coefficients. The corresponding synthesis filter coefficients are as follows:

- 7-tap low-pass filter: $[\ h'_{-3},\ h'_{-2},\ h'_{-1},\ h'_0,\ h'_1,\ h'_2,\ h'_3]$

$h'_3 = h'_{-3}$ = -0.0912717631142495

$h'_2 = h'_{-2}$ = -0.057543526228500

$h'_1 = h'_{-1}$ = +0.591271763114247

h'_0 = + 1.115087052456994

- 9-tap high-pass filter: $[\ g'_{-4},\ g'_{-3},\ g'_{-2},\ g'_{-1},\ g'_0,\ g'_1,\ g'_2,\ g'_3,\ g'_4]$

$g'_4 = g'_{-4}$ = +0.026748757410810

$g'_3 = g'_{-3}$ = +0.016864118442875

$g'_2 = g'_{-2}$ = -0.078223266528988

$g'_1 = g'_{-1}$ = -0.266864118442872

g_0' = +0.602949018236358

For lifting implementation, the (9, 7) wavelet filter pair can be factorized into a sequence of primal and dual lifting. The most efficient factorization of the polyphase matrix for the (9, 7) filter is as follows [30]:

$$\bar{P}(z) = \begin{bmatrix} 1 & a(1+z^{-1}) \\ 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 \\ b(1+z) & 1 \end{bmatrix} \begin{bmatrix} 1 & c(1+z^{-1}) \\ 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 \\ d(1+z) & 1 \end{bmatrix} \begin{bmatrix} K & 0 \\ 0 & 1/K \end{bmatrix}$$

where a = -1.586134342, b = -0.05298011854, c = 0.8829110762, d = -0.4435068522, K = 1.149604398.

Reversible Wavelet Transform for Lossless Compression: For loss less compression, the default wavelet filter used in the JPEG2000 standard is the Le Gall (5, 3) spline filter [31]. Although this is the default filter for loss less transformation, it can be applied in lossy compression as well. However, experimentally it has been observed that the (9, 7) filter produces better visual quality and compression efficiency in lossy mode than the (5, 3) filter. The analysis filter coefficients for the (5, 3) filter is as follows:

- 5-tap low-pass filter : [h_{-2}, h_{-1}, h_0, h_1, h_2]

$h_2 = h_{-2}$ = -1/8

$h_1 = h_{-1}$ = +1/4

h_0 = +3/4

- 3-tap high-pass filter: [g_{-1}, g_0, g_1]

$g_1 = g_{-1}$ = -1/2

g_0 = +1

The corresponding synthesis filter coefficients are as follows:

- 3-tap low-pass filter: [h_{-1}', h_0', h_1'

$h_1' = h_{-1}'$ = +1/2

h_0' = + 1

- 5-tap high-pass filter: [g'_{-2}, g'_{-1}, g'_0, g'_1, g'_2]

$$g'_2 = g'_{-2} = -1/8$$
$$g'_1 = g'_{-1} = -1/4$$
$$g'_0 = +3/4$$

Therefore, the polyphase for the (5,3) filter can be written as follows,

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

Boundary Handling Like a convolution, filtering is applied to the input samples by multiplying the filter coefficients with the input samples and accumulating the results. Since these filters are not causal, they cause discontinuities at the tile boundaries and create visible artifacts at the image boundaries as well. This introduces the dilemma of what to do at the boundaries. In order to reduce discontinuities in tile boundaries or reduce artifacts at image boundaries, the input samples should be first extended *periodically* at both sides of the input boundaries before applying the one dimensional filtering both during row-wise and column-wise computation. By symmetrical/mirror extension of the data around the boundaries, one is able to deal with the noncausal nature of the filters and avoid edge effects. The number of additional samples needed to extend the boundaries of the input data is dependent on filter length.

Quantization

After the DWT, all the subbands are quantized in lossy compression mode in order to reduce the precision of the subbands to aid in achieving compression. Quantization of DWT sub bands is one of the main sources of information loss in the encoder. Coarser quantization results in more compression and hence in reducing the reconstruction fidelity of the image because of greater loss of information. Quantization is not performed in case of lossless encoding. In Part 1 of the standard, the quantization is performed by uniform scalar quantization with dead-zone about the origin. In dead-zone scalar quantizer with step-size Δ_b, the width of the dead-zone (i.e., the central quantization bin around the origin) is $2\Delta_b$ as shown in Figure 10. The standard supports separate quantization step sizes for each subband. The quantization step size (Δ_b) for a subband (b) is calculated based on the dynamic range of the subband values. The formula of uniform scalar quantization with a dead-zone is

$$q_b(i,j) = sign(y_b(i,j)) \left\lfloor \frac{|y_b(i,j)|}{\Delta_b} \right\rfloor \qquad (4.5)$$

where $y_b(i,j)$ is a DWT coefficient in subband band Δ_b is the quantization step size for the subband b. All the resulting quantized DWT coefficients $q_b(i,j)$ are signed integers.

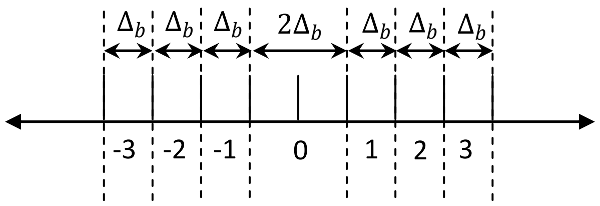
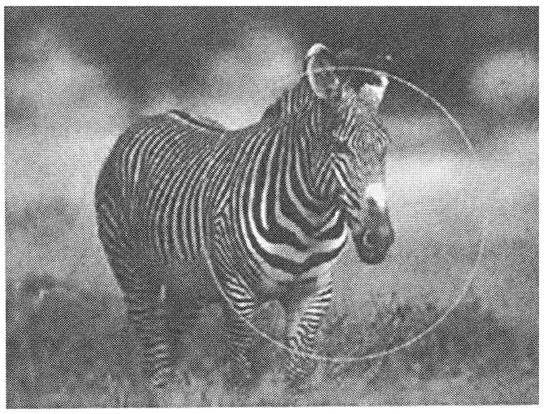


Figure 10 Dead-zone quantization about the origin

All the computations up to the quantization step are carried out in two's complement form. After the quantization, the quantized DWT coefficients are converted into sign-magnitude represented prior to entropy coding because of the inherent characteristics of the entropy encoding process.

Region of Interest Coding

The *region of interest* (ROI) coding is a unique feature of the JPEG2000 standard. It allows different regions of an image to be coded with different fidelity criteria. These regions can have arbitrary shapes and be disjoint to each other. In Figure 11, we show an example of ROI coding. We compressed the ROI portion of the Zebra image losslessly and introduced losses in the non-ROI (background) part of the image. The reconstructed image after decompression is shown in Figure 11(a). We indicate the ROI by a circle around the head of the Zebra in Figure 11(a). In Figure 11(b), we pictorially show the difference between the original image and the reconstructed image after ROI coding and decoding. The values of difference of the original and the reconstructed pixels in the ROI region (i.e., inside the circle) are all zeros (black) and they are nonzero (white) in the non-ROI parts of the image. This shows the capability of the


(a)

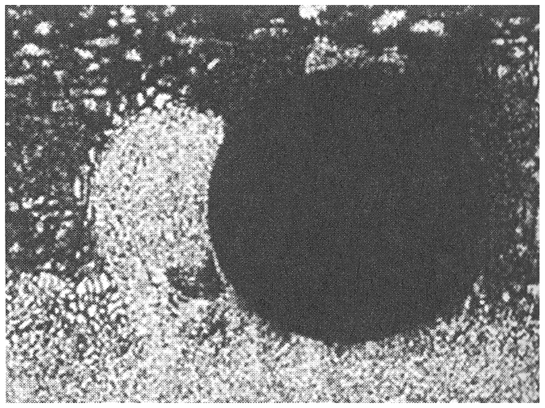
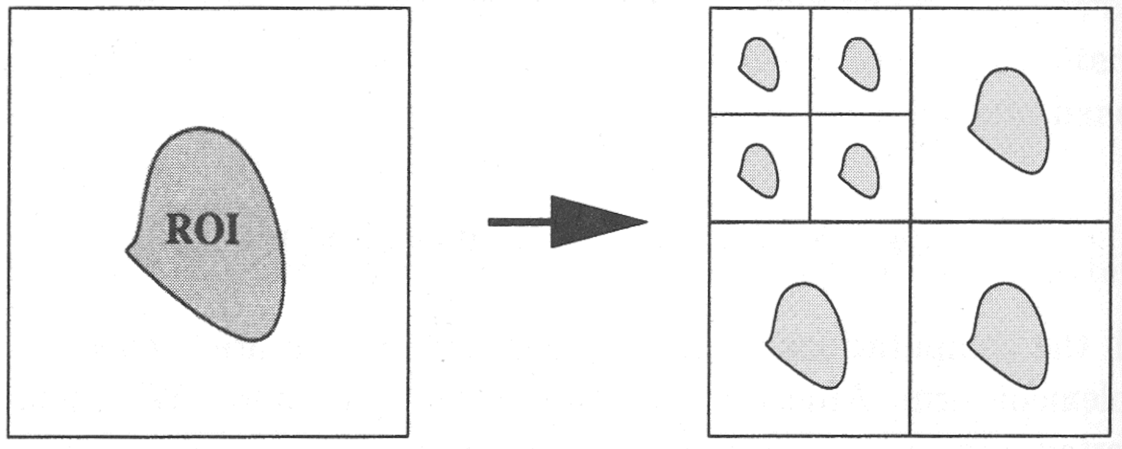

(b)

Figure 11 (a) Reconstructed image with circular shape ROI. (b) Difference between original image and reconstructed image



Spacial domain

(a)

Wavelet domain

(2-level DWT)

(b)

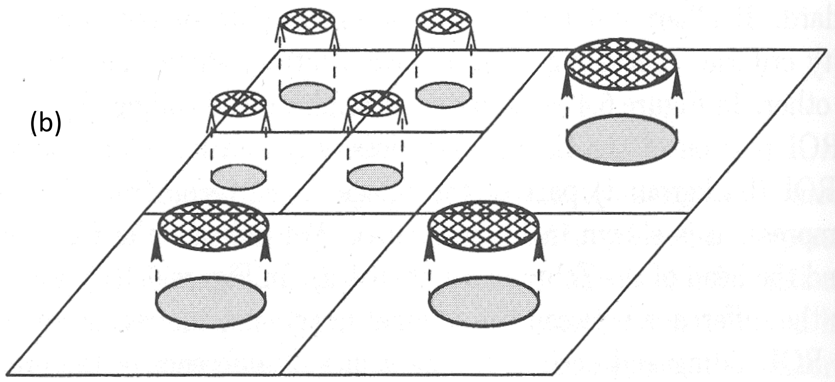


Figure 12 (a) ROI mask (b) Scaling of ROI coefficients

JPEG2000 standard in how we can compress different regions of an image with different degrees of fidelity.

The ROI method defined in the JPEG2000 Part 1 standard is called the MAXSHIFT method [32]. The MAXSHIFT method is an extension of the scaling-based ROI coding method [33]. During ROI coding, a binary mask is generated in the wavelet domain for distinction of the ROI from the background as shown in Figure 12(a). In the scaling-based ROI coding, the bits associated with the wavelet coefficients corresponding to an ROI (as indicated by the ROI mask) are scaled (shifted) to higher bit-planes than the bits associated with the non-ROI portion of the image. This is shown by a block diagram in Figure 12(b).

During the encoding process, the most significant ROI bit-planes are encoded and transmitted progressively before encoding the bit-planes associated with the non-ROI background region. As a result, during the decoding process, the most significant bit-planes of ROI can be decoded before the background region progressively in order to produce high fidelity in the ROI portions of the image compared to its background. In this method, the encoding can stop at any point and still the ROI portion of the reconstructed image will have higher quality than the non-ROI portion. In scaling-based ROI, the scaling parameter and the shape information needs to be transmitted along with the compressed bitstream. This is used in the Part 2 extension of the standard.

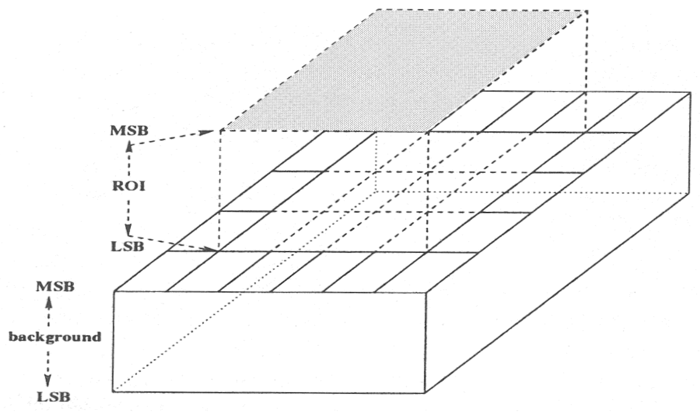


Figure 13 MAXSHIFT

In JPEG2000 Part 1, the MAXSHIFT technique is applied instead of the more general scaling-based technique. The MAXSHIFT allows arbitraryshaped regions to be encoded without requiring to transmit the shape information along with the compressed bitstream. As a result, there is no need for shape coding or decoding in the MAXSHIFT technique. The basic principle of the MAXSHIFT method is to find the minimum value *(V_{min})* in the ROI and the maximum value in the background (both in wavelet transformed domain) and then scale (shift) the wavelet coefficients in ROI in such a manner that the smallest coefficient in the ROI is always greater than the largest coefficient in the background. Then the bit-planes are encoded in the order of the most significant bit (MSB) plane first to the least significant bit (LSB) plane last. Figure 13 shows an example where the LSB plane of ROI is shifted above the MSB plane of the background region. During the decompression process, the wavelet coefficients that are larger than *Vmin* are identified as the ROI coefficients without requiring any shape information or the binary mask that was used during the encoding process. The ROI coefficients are now shifted down relative to *Vmin* in order to represent it with original bits of precision.

In JPEG2000, due to the sign-magnitude representation of the quantized wavelet coefficients required in the bit-plane coding, there is an implementation precision for number of bit-planes. Scaling the ROI coefficients up may cause an overflow problem when it goes beyond this implementation precision. Therefore, instead of shifting ROI up to higher bit-planes, the coefficients of background are downscaled by a specified value s, which is stored in the RGN (ReGioN) marker segment in the bitstream header. The decoder can deduce the shape information based on this shift value *s* and magnitude of the coefficients. By choosing an appropriate value of *s*, we can decide how many bit-planes to truncate in the background in order to achieve overall bit-rate without sacrificing the visual quality of the ROI.

Rate Control

Although the key encoding modules of JPEG2000 such as wavelet transformation, quantization, and entropy coding (bit-plane coding and binary arithmetic coding) are clearly specified, some implementation issues are left up to the prerogative of the individual developers. Rate control is one such open issue in JPEG2000 standard. Rate control is a process by which the bitrates (sometimes called coding rates) are allocated in each code-block in each subband in order to achieve the overall target encoding bit-rate for the whole image while minimizing the distortion (errors) introduced in the reconstructed image due to quantization and truncation of codes to achieve the desired code rate. It can also be treated in another way. Given the allowed distortion in the MSE (mean square energy) sense, the rate control can dictate the optimum encoding rate while achieving the maximum given MSE.

The JPEG2000 encoder generates a number of independent bitstreams by encoding the code-blocks. Accordingly a rate-distortion optimization algorithm generates the truncation points for these bitstreams in an optimal way in order to minimize the distortion according to a target bit rate. After the image is completely compressed, the rate-distortion optimization algorithm is applied once at the end using all the rate and rate-distortion slope information of each coding unit. This is the so-called postcompression rate-distortion (PCRD) algorithm.

There is another simple way to control bit-rate by choosing the quantization step size. The bigger the step size, the lower the rate will be. However, this method can apply only to lossy compression mode, and every time the step sizes change, the Tier-l encoding needs to be recomputed. Since the Tier-l coding is a very computationally intensive module in JPEG2000 standard, this approach of bit-rate control may not be suitable for some applications that are computationally constrained.

However, the bit-rate control is purely an encoder issue, and remains an open issue for the JPEG2000 standard. It is up to the prerogative of the developers how they want to accomplish the rate-distortion optimization in a computationally efficient way without incurring too much computation and/or hardware cost.

From the hardware implementation perspective, the rate-distortion algorithm requires a microcontroller to compute the breakpoints using a rate distortion optimization technique and supply these breakpoints to the entropy encoding engine for formation of the compress bitstream.

Entropy Encoding

Physically the data are compressed by the entropy encoding of the quantized wavelet coefficients in each code-block in each subband. The entropy coding and generation of compressed bitstream in JPEG2000 is divided into two coding steps: *Tier-l* and *Tier-2* coding.

Tier-1 Coding : In Tier-l coding, the code-blocks are encoded independently. If the precision of the elements in the code-block is *p*, then the code-block is decomposed into *p* bit-planes and they are encoded from the most significant bit-plane to the least significant bit-plane sequentially. Each bit-plane is first encoded by a *fractional bit-plane coding* (BPC) mechanism to generate intermediate data in the form of a *context* and a binary *decision* value for each bit position. In JPEG2000 the *embedded block coding with optimized truncation* (EBCOT) algorithm [2] has been adopted for the BPC. EBCOT encodes each bit-plane in three coding passes, with a part of a bit-plane being coded in each coding pass without any overlapping with the other two coding passes. That is the reason why the BPC is also called *fractional* bit-plane coding. The three coding passes in the order in which they are performed on each bit-plane are *significant propagation pass, magnitude refinement pass,* and *cleanup pass.*

The binary decision values generated by the EBCOT are encoded using a variation of *binary arithmetic coding* (BAC) to generate compressed codes for each code-block. The variation of the binary arithmetic coder is a *context adaptive BAC* called the MQ-coder, which is the same coder used in the JBIG2 standard to compress bi-Ievel images. The *context* information generated by EBCOT is used to select the estimated probability value from a lookup table and this probability value is used by the MQ-coder to adjust the intervals and generate the compressed codes. JPEG2000 standard uses a predefined lookup table with 47 entries for only 19 possible different contexts for each bit type depending on the coding passes. This facilitates rapid probability adaptation in the MQ-coder and produces compact bitstreams.

TIER-2 Coding and Bitstream Formation

After the compressed bits for each code-block are generated by Tier-1 coding, the Tier-2 coding engine efficiently represents the layer and block summary information for each code-block. A *layer* consists of consecutive bit-plane coding passes from each code-block in a tile, including all the subbands of all the components in the tile. The block summary information consists of length of compressed code words of the codeblock, the most significant magnitude bit-plane at which any sample in the code-block is nonzero, as well as the truncation point between the bitstream layers, among others. The decoder receives this information in an encoded manner in the form of two tag trees. This encoding helps to represent this information in a very compact form without incurring too much overhead in the final compressed file. The encoding process is popularly known as *Tag Tree coding.*

Chapter 5

Simulation of Image Compression Algorithm using Matlab

Digital image is composed of a finite number of elements, each has a particular location and value. Image compression is a complex process and involves numerous steps of calculation to attain a reduction in the amount of data required to represent a digital image. Data redundancy is a fundamental issue in image compression. A lossy image compression technique which provides a higher level of data reduction but result in a less than perfect reproduction of original image is implemented here using matlab code. An image compression method which exploits the ability of discrete wavelet transform to decorrelate adjacent pixels of the image and uses Huffman coding to encode the transformed coefficients based on the estimated probability of occurrence for each possible value of the source symbol is presented with their performance analysis.

Algorithm

The first step of the encoding process is to DC level shift the samples of the S_{siz}-bit unsigned image to be coded by subtracting 2^{Ssiz-1}. If the image has more than one component - like the red, green and blue planes of a colour image – each component is individually shifted. If there are exactly three components, they may be optionally decorrelated using a reversible or nonreversible linear combination of the components. Here the code is developed for handling a grayscale image, i.e. only one component specifying the gray level of the image at that point, having values ranging from 0 to 255 for each pixel. A bitmap format 512 x 512 lena image with 8-bit pixel representation is used as the test image. The image is a grayscale image has values ranging from 0 to 255 for each pixel.

After the image has been level shifted, its components are optionally divided into tiles. Tiles are rectangular arrays of pixels that contain the same relative portion of all components. Thus, the tiling process creates tile components that can be extracted and reconstructed independently, providing a simple mechanism for accessing and/or manipulating a limited region of a coded image. The tile size holds inverse relation with the distortion in the reconstructed image. So the larger the tile size the better will be the reconstructed image in terms of distortions. Here the tile size is taken as equal to the image size. The discrete wavelet transform is applied on the image to obtain the decorrelated transform coefficients. Cohen-Daubechies-Feauveau wavelet, a biorthogonal 9/7 coefficient scaling-wavelet vector is employed for a lossy compression. The discrete wavelet transform is implemented as two parts, first a row-wise convolution followed by a column-wise convolution to generate each set of coefficients namely approximation, horizontal, vertical and diagonal. The image is convoluted row-wise using low pass filter coefficients to get a matrix of half the number of columns, while the number of rows remaining same. This matrix is then convoluted column-wise with the low pass and high pass filter coefficients to get the approximation and horizontal coefficients respectively. The image when convoluted row-wise with high pass filter coefficients, then column-wise with low pass and high pass will generate the vertical and diagonal coefficients respectively.

It is necessary to pad elements around the image to avoid the distortion of the reconstructed image at the edges. This distortion occurs due to the failure to handle the periodicity issue properly during convolution. Thus unless proper padding is implemented, the result is erroneous, commonly referred to as *wraparound error*. Different kinds of padding include: padding with zeros, extending the elements at the edge etc. These padding needs additional number elements to be included in the result to reproduce desired result. So to avoid the wraparound error while keeping the size of the coefficient matrix same as that of the image, padding is done with the elements of the opposite edge of the image, i.e. the number of elements in the filter minus one elements, required to pad on one edge of the image is taken from the opposite edge. This style of padding avoids duplication of any element within the image or an array of zeros and provides the information back when same style of padding is repeated at the time of deconvolution.

When each of the tile components has been processed, the total number of transform coefficients is equal to the number of samples in the original image – but the important visual information is concentrated in a few coefficients. To reproduce the number of bits needed to represent the transform, coefficient $a_b(u, v)$ of subband *b* is quantized to value $q_b(u,v)$ using

$$q_b(u,v) = sign[a_b(u,v)].floor\left[\frac{|a_b(u,v)|}{\Delta_b}\right] \qquad (5.1)$$

where quantization step size Δ_b is

$$\Delta_b = 2^{R_b-\varepsilon_b}\left(1+\frac{\mu_b}{2^{11}}\right) \qquad (5.2)$$

R_b is the nominal dynamic range of subband b, and ε_b and μ_b are the number of bits allotted to the exponent and mantissa of the subband's coefficients. The nominal dynamic range of subband

b is the sum of the number of bits used to represent the original image and the analysis gain bits for subband b. For error-free compression, $\mu_b = 0$, $R_b = \varepsilon_b$, and $\Delta_b = 1$. For irreversible compression, no particular quantization step size is specified in the standard.

Now the coefficients are entropy coded using Huffman coding. The final steps of the encoding process are coefficient packetizing, where the output consists of an array of 1's and 0's, packed as groups of sixteen. Packets are the fundamental unit of the encoded code stream. The decoders simply invert the operations described previously.

Simulation Results

The grayscale lena image of 512x512 pixel, with 8-bit representation for each pixel is used as the test input. The test image was compressed to different scales, from one to three and the compression ratio as well as the root mean square error of the reconstructed image were calculated for minimal error case and quantised case. The result is tabulated below:

Table 3 RMS error and Compression ratio for different scale

SCALE	Minimal Error Case		Qunatized Case	
	RMS Error	Compression Ratio	RMS Error	Compression Ratio
1	0.7916	1.4744	0.6744	1.4497
2	0.9759	1.7640	1.2858	1.9555
3	1.1563	1.8676	2.2667	2.2000

Figure 14 Test input - Lena image

The actual test image and the recovered image after compression, with the difference from actual image for scales 1 & 2 for minimal error case and quantized case are shown in Figures 14 – 18.

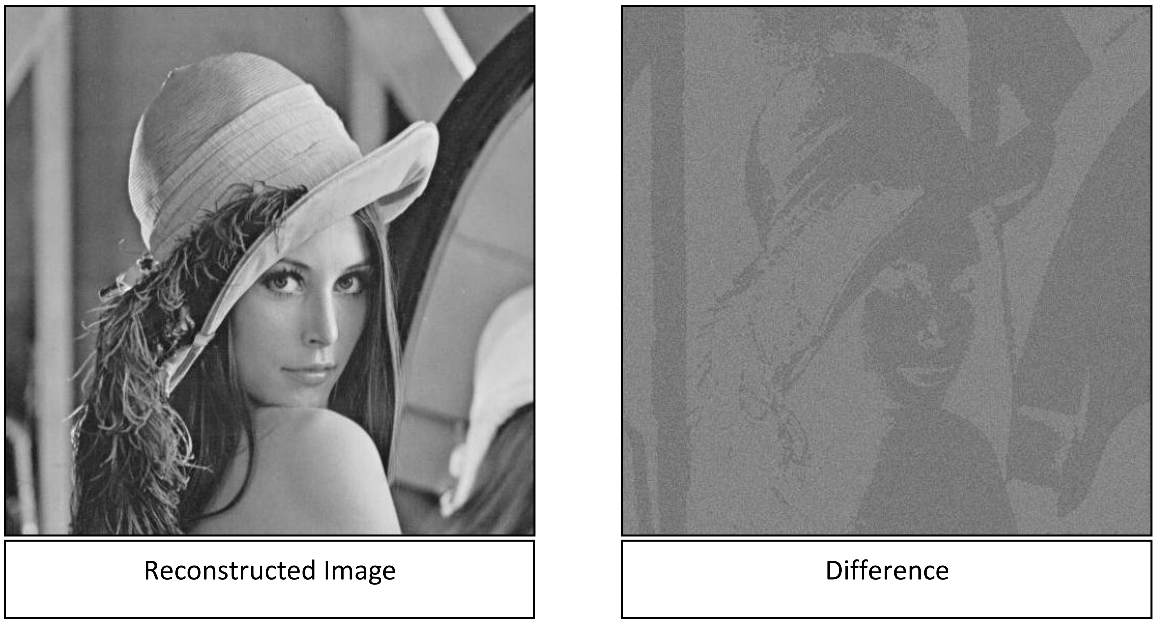


Figure 15 Scale 1 and Minimal error case

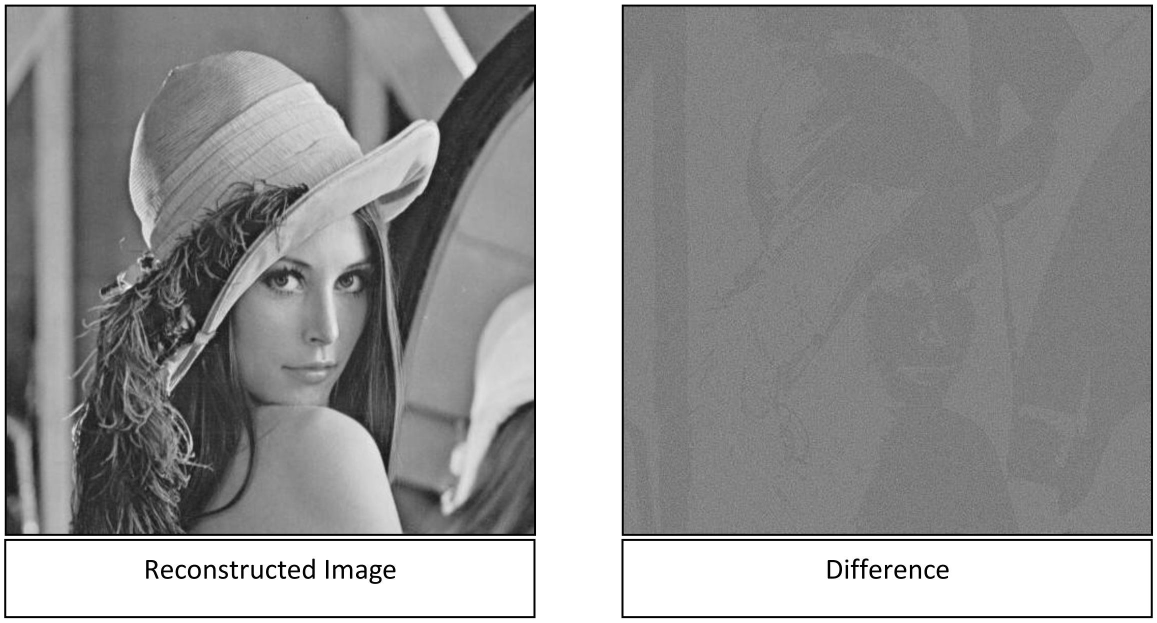


Figure 16 Scale 1 and Quantized case

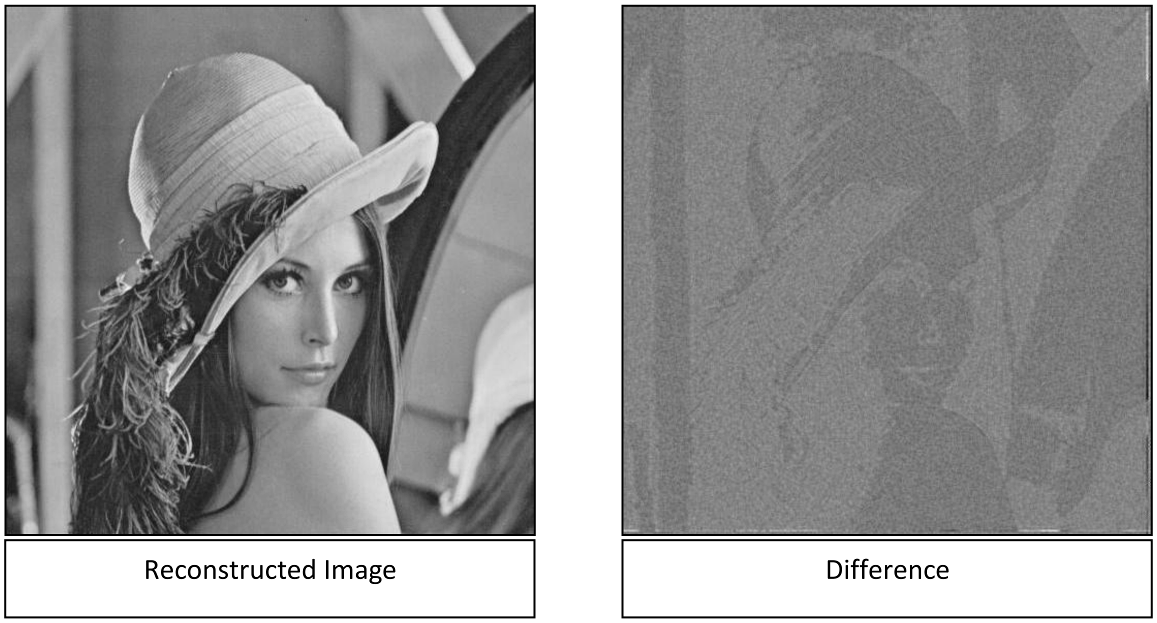


Figure 17 Scale 2 and Minimal error case

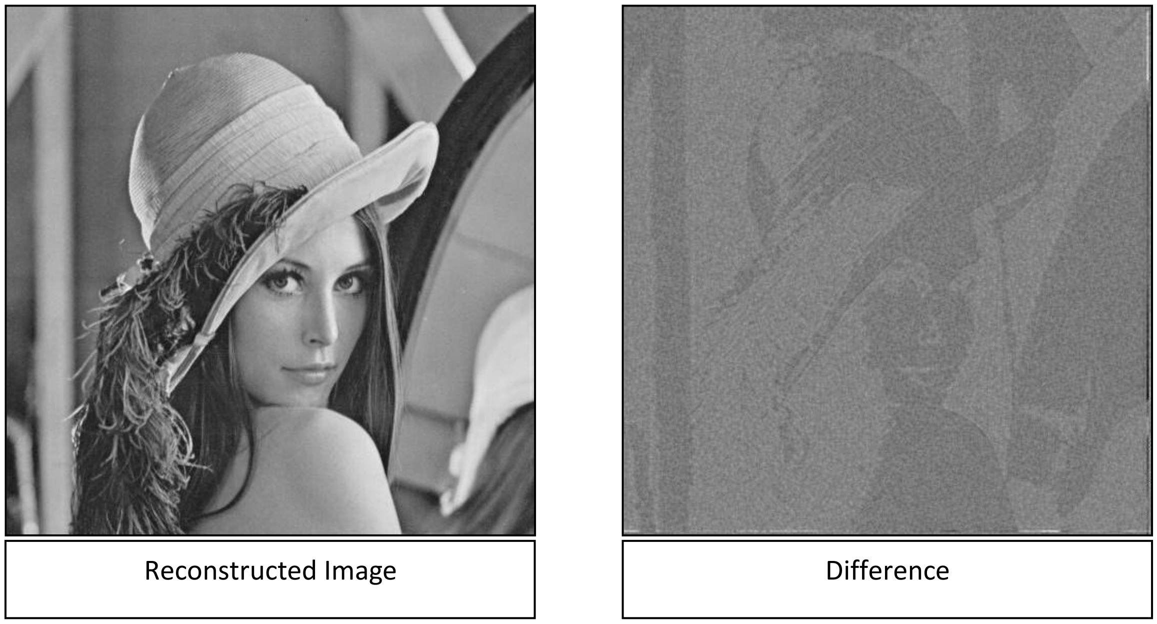


Figure 18 Scale 2 and Quantized case

Chapter 6

Power Analysis of Booth Multiplier

Introduction

With rapid developments in field of signal processing, control systems and other computer applications, arithmetic circuits mainly multipliers are becoming very important. And also handheld electronic gadgets are trend of today. To design high performance handheld gadgets, circuit designs must be optimized for area, power and timing. All the three tenets of VLSI design: area, power and timing must given attention during design. Here the focus is on designing booth multipliers with different adder architectures and their ASIC implementation. Booth multipliers are designed using CLA, RCA and CSA adders. Area, power and timing analysis of different designs is discussed. Also a detailed power estimation techniques based switching activity is applied on different multiplier designs, which gives deeper insight into power dissipation in digital design. The three main layers of abstraction include RTL, gate and transistor level. The various power values are calculated using Synopsys tools in this research [34] [35].

In CMOS technologies, the chip components draw power only during a logic transition. This is considered as an attractive feature, it makes the power dissipation highly dependent on 'switching activity' inside these circuits. This complicates the power estimation problem because the power dissipation will become input pattern dependent. Even though, it is practically impossible to estimate the power by simulating the circuit for all possible inputs. There are several techniques like using probabilities to describe the set of all the possible logic signals, some statistical techniques are also applied [34][36]. Probability and statistical method based techniques use simplified delay models, so that they do not provide the same accuracy as simulation. Also statistical techniques are slow and not feasible for big circuits [34]. For simple combinational circuits like adders and multipliers, switching based power calculation is feasible.

Our main goal is to analyze area; timing and power of Booth multiplier with different adder architectures in TSMC 65nm technology. For each of the architecture, power is calculated at various levels of abstraction using Power Compiler and Primepower. One of the contributions of our research is calculating the switching activity based power at RTL level and comparing them with the Primepower, which gives post layout power. Timing analysis is done using Primetime.

The remainder of this chapter is structured as follows. Section 2 gives the brief description of both multiplier design with carry look ahead adder (CLA), ripple carry adder (RCA), and carry save adder (CSA). Section 3 describes the power analysis methodology adopted for calculating the various levels of design flow. Finally in section 4 and 5, results and conclusion are drawn.

Booth Multiplier

Booth Algorithm

Multipliers form the basic and central module in any computing system. There are several algorithms for performing this complex process depending on the type of numbers taking into consideration as multiplicand and multiplier. Booth's algorithm [37], a technique explained by Andrew D. Booth half a century back whereby two binary numbers of either sign may be multiplied together by a uniform process which is independent of any fore knowledge of the signs of these numbers. The basic idea remains the same to the long multiplication, to find partial product and sum, except for some modifications to suit the signed base-two representation.

Booth's algorithm takes on account two bits starting from the least significant digit in the multiplier at a time to decide on the partial products, which are variants of the multiplicand as shown in Table 4. Except for the last partial product for a particular multiplier, the partial products are shifted and added to the existing sum of partial products.

Table 4 Partial products in Booth's algorithm

$X_{n+1}X_n$	**Partial Product**
00	zero
01	same as the multiplicand
10	negative of the multiplicand
00	zero

Booth presented an algorithm based on radix-2 recording for the multiplication of two signed fixed point numbers, by a method of multiplication which uses uniform shift by single place following an addition or subtraction and permits predicting the number of cycles that will be required from the size of the multiplier. The last cycle of operation needs special handling. By using higher radices the number of cycles required to complete the multiplication operation can be reduced [38]. This reduction in the number of cycles will result in superior performance in terms of speed with a trade-off between the speed and complexity.

Addition of partial products forms an important stage in the implementation of a Booth's multiplier. By using different types of adders like carry look-ahead adder and carry save adder instead of ripple carry adder, improvements can be done in delay and power consumption of the multiplier. In this chapter, power consumption of signed 16-bit Booth Multiplier at various levels is analysed for above adders and combination of them.

Booth Algorithm for Radix-4 Fixed Point Multiplication

Radix-4 Booth's algorithm analyze more number bit at a time and consequently gives the product with less number of cycles compared to the standard Booth's algorithm. For high radices Booth's algorithm correction cycles are employed. For the radix-4, a divide-by-2 correction cycle is needed when the length of the multiplier is even, and a correction cycle is not needed when the length of the multiplier is odd [39].

Table 5 Partial products for Radix-4 Booth's algorithm

Bit Grouping	**Partial Product**
000 or 111	(P+0)/4
001 or 010	(P+B)/4
011	(P+2B)/4
100	(P-2B)/4
101 or 110	(P-B)/4

The rules for the radix-4 Booth recording are as follows:

1. Append a zero to the right of the LSB of the multiplier number, A.
2. Analyse the groups of three adjacent bits of A, starting with the LSB and the appended zero.
    a. If the triad is 000 or 111, then shift the partial product two bits to the right (i.e. divide the partial product by 4).

b. If the triad is 001 or 010, then add the multiplicand, B, to the partial product and shift the new partial product by two bits to the right.

c. If the triad is 101 or 110, then subtract the multiplicand, B, from the partial product and shift the new partial product by two bits to the right.

d. If the triad is 011, then add twice the multiplicand, 2B, to the partial product and shift the new partial product by two bits to the right.

e. If the triad is 100, then subtract twice the multiplicand, 2B, from the partial product and shift the new partial product by two bits to the right.

3. The step 2 is continued in a manner such that the LSB of the current triad is MSB of the previous.
4. The last triad is analysed, correction right shifts [39] are performed on the product produced by the relation '(L – 1) mod2', where L is the length of the multiplier.

Adders Description

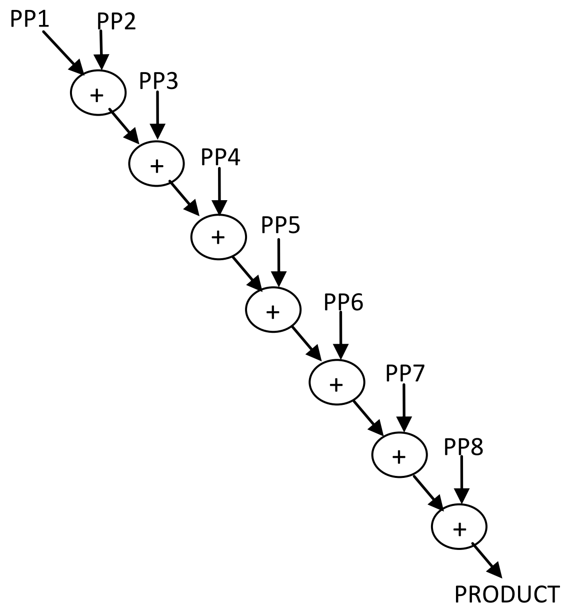


Figure 19 Arrangement of CLA in Booth Multiplier

The radix-4 booth recording of a 16-bit number results in 8 partial products, which are added to get the product. The primary option for addition of these partial products is to use ripple carry adder (RCA). When it is desired to improve the performance of the multiplier in terms of the delay or area, the use of alternatives for RCA is a good technique. Adders like carry look ahead adder (CLA) [40] and carry save adder (CSA) are the alternatives considered for RCA. Adders of two different sizes, result's size and partial product's size, are used and compared the performance of the multiplier with reference of the size of the adder size.

The RCA is a simple adder in which the carry has to propagate or ripple through the fulladder modules. The worst case delay for a 31-bit RCA is 30 carry bit calculations and that for a 17-bit RCA is 16 carry calculations. The CLA adder computes several carries at the same time, thereby reducing the computational time. In the extreme case, all the carries could be computed at the same time. A two-level CLA adder is used to calculate the sum of two 31-nbit number. The 31-bit number (n = 31) is divided into groups of 4-bits (m = 4). The above eight groups are again grouped into two to form two levels (p = 2). The CLA module consists of four parts: (i) the computation of xor (p_i), carry generate (g_i), carry propagate (a_i) for each bit; (ii) the computation of carry lookahead generator in which carry at each bit position of a group is calculated simultaneously; (iii) the computation of carry propagate (A_i), carry generate (G_i) and carry-out at each level; (iv) computation of sum (s_i).

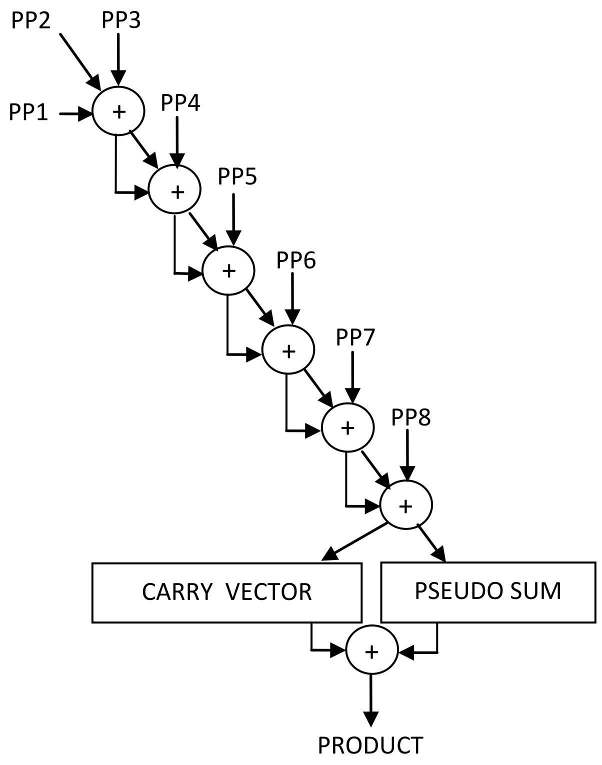


Figure 20 Arrangement of CLA in Booth Multiplier

The worst case delay is of a two-level CLA is given by the expression:

$$T_{2\text{-}CLA} = t_{a,g} + t_{A,G} + (n/pm)\, t_{clg} + t_{clg} + t_s$$

where $t_{a,g}$ is delay of computation of a_i, g_i; $t_{A,G}$ of A_i, G_i; t_{clg} of carry lookahead generator, t_s of s_i and (n/pm) t_{clg} of rippling carry between groups.

CLA adder of sizes 31-bit and 17-bit were realised, with above architecture. For 17-bit adder, the delay due to the term '(n/pm) t_{clg}' is less, resulting in lesser worst case delay. The arrangement of CLA in the booth multiplier for adding eight partial products is shown in the Figure 19.

The CSA adder performs addition of three binary vectors using an array of fulladders, but without propagating the carries. The output is represented by two binary vectors called carry vector and the pseudo-sum vector. The CSA produces a reduction from three binary vectors to two binary vectors. The carry-save representation is converted to conventional using a carry-propagate adder with the two operands being carry vector and the pseudo-sum vector. The time of carry-save addition corresponds to the delay of one full-adder, independent of the number of bits. Carry propagation occurs only in the last stage, hence that the carry-save adder becomes more efficient in terms of delay with the number carry-save additions. The worst case delay is given by the expression:

$$T_{CSA} = n.T_{FA} + T_{CPA}$$

Where T_{FA} is the full adder delay, 'n' is the number of CSA stages and T_{CPA} is the worst case delay of CPA, which is equal to the propagation time of carry from lsb to msb. The arrangement of CSA in the booth multiplier for adding eight partial products is shown in the Figure 20.

Power Estimation Model

Switching activity based power analysis is widely adopted method. Typically the methods for creating the switching activity inside a circuit fall into either simulation based methodology or estimation based methodology. Based on the analysis flow that is used, switching activity can be modeled in different levels of detail.

Power analysis is done at three stages: RTL level, gate level, post layout analysis. RTL power estimator is used to calculate power at first (P1).In RTL level, at first SAIF (Switching Activity Interchange Format) file is generated using power compiler. SAIF is an open ASCII format and captures the switching statistics for each node in the design in terms of static and dynamic attributes that be state and path dependent. Both static and dynamic attributes can be state-dependent, capturing the switching statistics separately for the different states of a cell. State dependent static attributes are useful for computing state dependent leakage power and for computing dynamic power. The dynamic attributes can also be path dependent, capturing separate switching statistics based on which input path caused the transition on the pin [34][35].

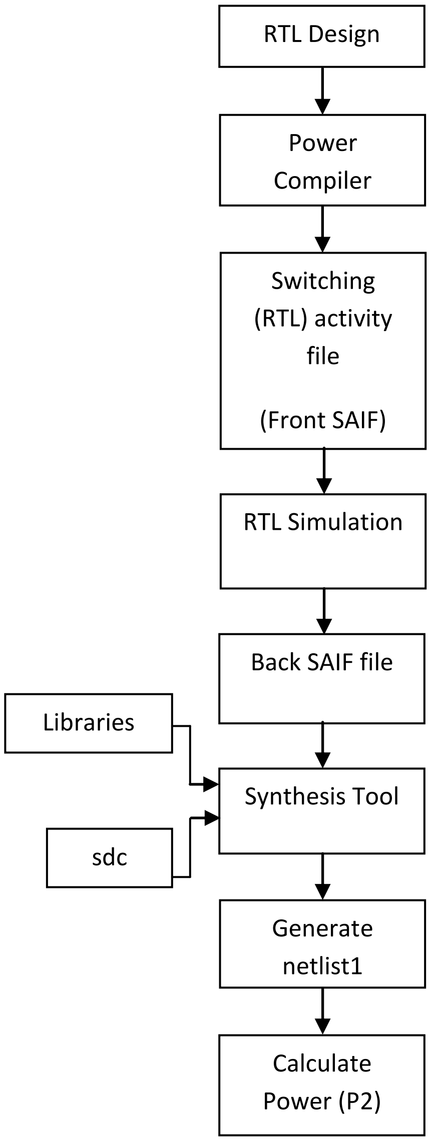


Figure 22 RTL Power

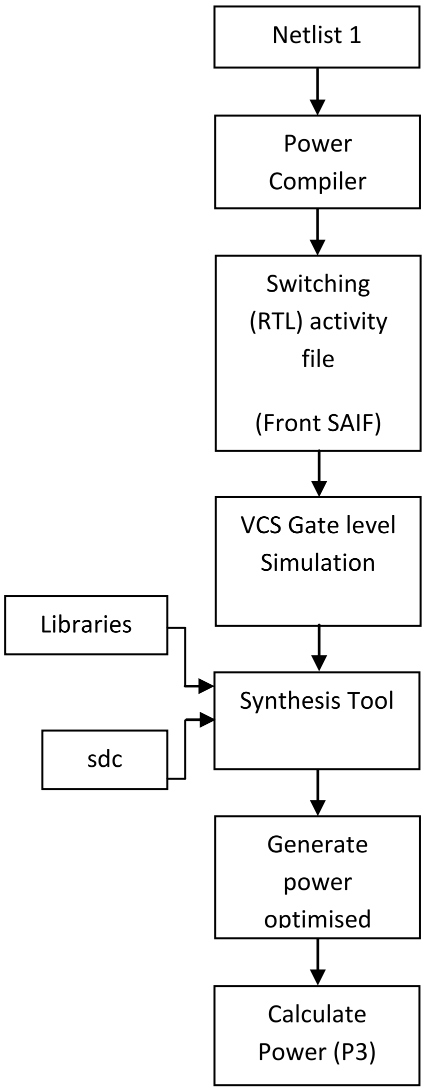


Figure 21 Netlist based Power

RTL simulation based power analysis flow within power compiler is shown in Figure 21. Switching activity is captured via RTL simulation at the synthesis invariant points in the design.

These include the hierarchy boundaries and sequential elements. Capacitance and power models for wires and gates are taken from the library. The RTL design is synthesized to gate level along with all the constraints. Activity information that is captured at the synthesis invariant points is propagated to all the input pins of all cells in the gate-level design. This information is passed to the power computation engine, which reports the power of entire design. Front SAIF file generated by the Power Compiler is used in RTL simulation to generate back annotated SAIF file from simulator. The back annotated SAIF is fed into synthesis tool (Design Compiler) and synthesizing the design. After synthesis, calculate power (P2). The detailed flow is shown in Figure 21.

The gate-level simulation based power analysis flow is similar, except that no internal activity propagation is required because activity is captured at the input pins of all the cells in the gate-level netlist via gate-level simulation. Because this activity is captures in full detail, it is possible to use the state and path dependent information in the library models and in SAIF to perform a more accurate power analysis (P3). The detailed flow is similar to RTL simulation flow and shown in Figure 22 [41][42].

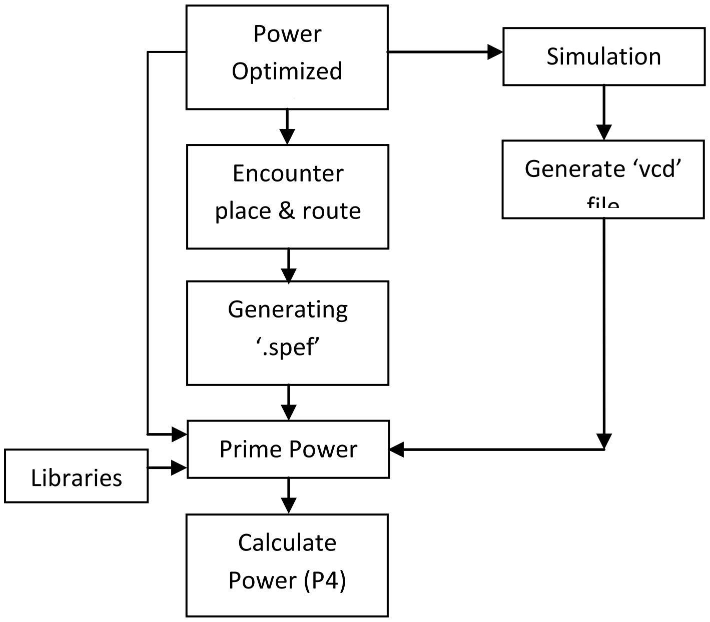


Figure 23 Post Layout Power

The complete time-based power profile view of the chip is calculated using the value change dump (VCD) or VCD formats, which are generated based on gate-level based switching activity. After post placement and routing netlist, the wire capacitances other parasitic are back annotated from layout. Primepower provides a detailed analysis of the power dissipation in a design (P4) and relies on the complete VCD switching activity format and back annotated parasitic file. It works on a gate-level netlist with gate-level simulation data and is targeted to full-chip capacity. Along with the average power numbers, it also gives the time-based waveforms of power consumption in different ports of design and allows designers to locate hot spots in design. The detailed flow is shown in Figure 23 [34] [41] [42].

Results

For the synthesis and power calculation, 65nm library from TSMC is used. The Table 6 shows the power numbers for booth multiplier with different adders. The variation of different power values from power estimator (P1), RTL switching activity (P2), gate level switching activity (P3) and post layout power calculated from Primepower (P4) can be observed.

The results of Primepower i.e. P4 can be considered as more accurate power than others. RTL and gate level switching activity based power (P2 & P4) gives information on switching power.

Table 6 Power Calculated

Adder Architecture	P1(mW)	P2(μW)	P3(μW)	P4(μW)
RCA 31-bit	7.0893	158.0197	94.3455	185.147
CLA 31-bit	9.8731	216.4607	154.7195	195.983
CSA 31-bit RCA 31-bit	6.4396	147.3754	85.5816	144.568
CSA 31-bit CLA 31-bit	6.8364	156.5996	95.0299	147.621
RCA 17-bit	5.4041	128.0270	63.5296	117.97
CLA 17-bit	6.7736	150.4392	88.5719	118.795
CSA 17-bit RCA 31-bit	5.4912	133.0934	72.8919	111.736
CSA 17-bit CLA 31-bit	5.6856	138.8414	77.2312	107.553

Timing analysis and area for the eight different architectures are tabulated in the Table 7. T1 is pre-layout timing analysis and T2 is post-layout timing analysis. The results clearly show the reduction in delay and increase in area for architectures using CLA over RCA. The adder size also affects the timing and area of the multiplier. Delay is more for multiplier architecture using 17-bit adders due to the increase in complexity and the area reduces with the adder size.

Table 6 shows the RTL, gate-level and post layout power of booth multiplier using different adder architectures. RCA 17-bit adder is giving lowest power. The best architecture depends on the application, for a low power application the best option is the multiplier using 17-bit adders. If the speed is considered then the combination of 31-bit CSA and CLA will be the better option. If area is considered as constraint, the multiplier with the combination of 17-bit CSA and 31-bit RCA excel other architectures.

Table 7 Timing Analysis and Area Calculated

Adder Architecture	T1(µs)	T2(µs)	AREA (µm^2)
RCA 31-bit	6.03	1.15	6249.600098
CLA 31-bit	5.41	1.17	7600.680176
CSA 31-bit RCA 31-bit	5.66	1.11	5680.080078
CSA 31-bit CLA 31-bit	3.92	1.09	5873.040039
RCA 17-bit	7.15	1.02	4931.640137
CLA 17-bit	5.77	1.04	5533.560059
CSA 17-bit RCA 31-bit	5.14	1.06	4883.760254
CSA 17-bit CLA 31-bit	3.95	1.02	4920.840332

In VLSI design, there is always tradeoff between area, power and speed. By careful observation of the results, it can be concluded that CSA-17 bit RCA 31-bit multiplier is best, when it comes to area. RCA 17-bit and CSA 17-bit, CLA 31-bit are best multipliers with respect to speed. For power, CSA 17-bit CLA31-bit gives the best post-layout power. But RCA 17-bit gives best gate–level switching power. Booth multiplier with cs-17 bit RCA 31-bit is used for performing multiplication in Discrete Wavelet Transform implementation due to its least area and better performance for timing and power analysis.

Chapter 7

Verilog Implementation of Image Compression Algorithm

Verilog HDL is a hardware description language that can be used to model a digital system at many levels of abstraction ranging from the algorithmic-level to gate-level to the switch-level. Image compression being a complex procedure, algorithmic-level coding in verilog HDL is suitable. The verilog code for the image compression encoder is implemented using two parts namely, discrete wavelet transform and the encoding of the transform coefficients. An Image Compression Encoder with a two level discrete wavelet transform with encoding is realised here.

Discrete Wavelet Transform

The discrete wavelet transform is performed by row-wise convolution followed by column-wise convolution. The image to be transformed is serially inputted, and saved as a two dimensional array of bit-vector. Once a row is completely obtained, the module corresponding to the row-wise convolution is activated using a signal. This signal will stay high till the row-wise convolution is completed. Once the row-wise convolution is finished, the control signal corresponding to the activation of column-wise convolution is made high. After the completion of the column-wise convolution, four set of coefficients are obtained, approximation coefficients at the top left corner, vertical coefficients at the top right corner, horizontal coefficients at the bottom left corner and finally the diagonal coefficients at the bottom right corner.

For the second level discrete wavelet transform, the approximation coefficients from above transform is again transformed with discrete wavelet in the fashion specified above. But the activation of row-wise convolution for second level discrete wavelet transform can be done when the column-wise convolution of first level reaches midway, for the reason that the approximation coefficients are generated at this point.

During convolution, both row-wise and column-wise padding is required to avoid the wrap around error. The padding as specified earlier is done with, number of filter coefficients minus one elements, from the opposite edge of the image. The row-wise convolution is followed by a down sampling which will avoid alternate elements reducing the number of columns to half that of input, while the columns-wise convolution is followed by a down sampling reducing the number of rows to half of the input. Here the calculations corresponding to the deleted elements are unnecessary and is eliminated by customising the code. The convolution procedure involves a large number of multiplication and addition. A fixed point 16-bit signed Booth multiplier used

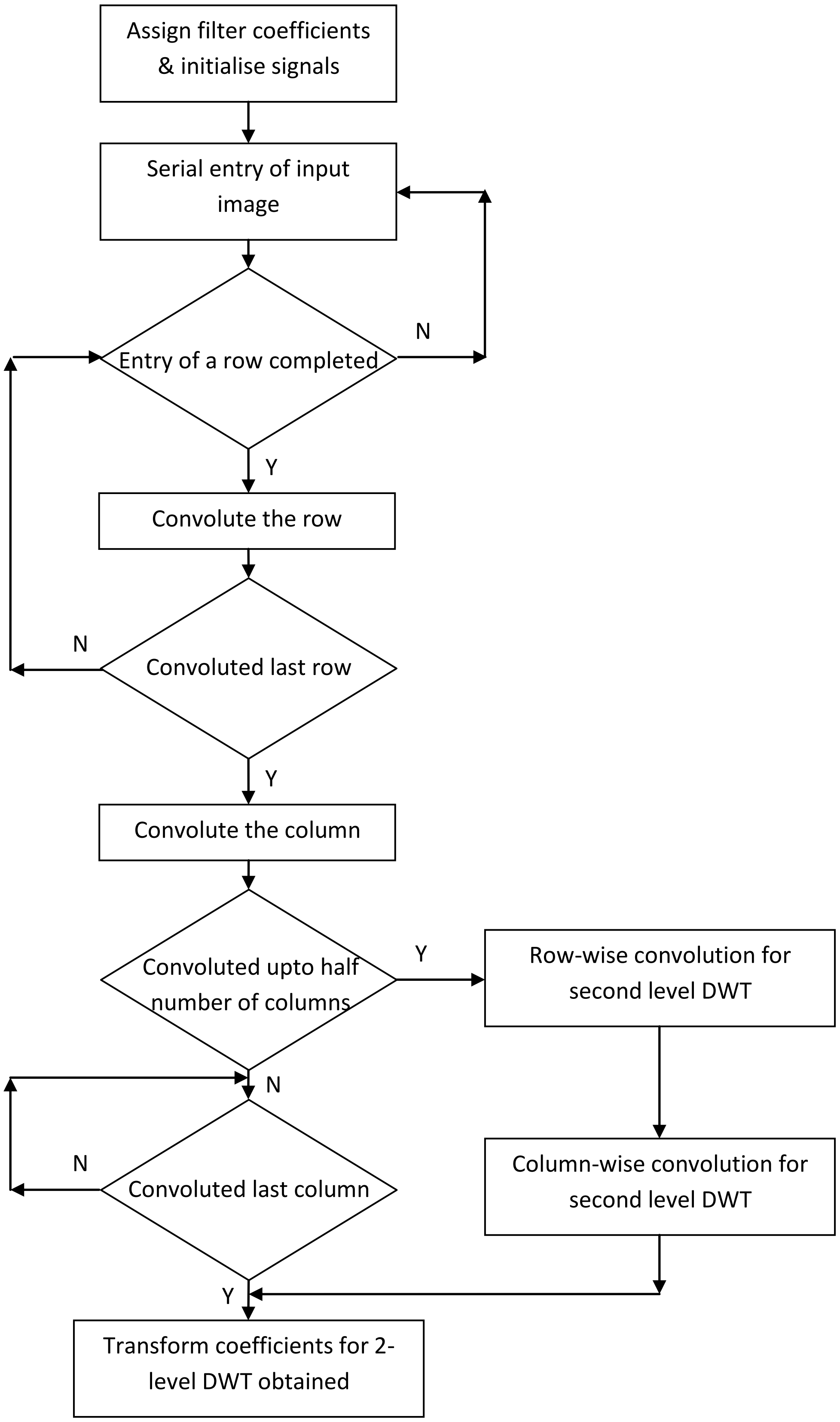

Assign filter coefficients & initialise signals
Serial entry of input image
Entry of a row completed
N
Y
Convolute the row
Convoluted last row
N
Y
Convolute the column
Convoluted upto half number of columns
Y
Row-wise convolution for second level DWT
N
Convoluted last column
N
Column-wise convolution for second level DWT
Y
Transform coefficients for 2-level DWT obtained

Figure 24 Block Diagram 2-D DWT Algorithm

to find the product. The algorithm followed for the Discrete Wavelet Transform is shown diagrammatically in Figure 24.

Encoding

The encoding technique implemented here is similar to embedded zero wavelet coefficient (EZW). After obtaining the wavelet transform coefficients, the encoding process is started. The encoding process takes in to consideration a tree like structure between the coefficients in consecutive levels of transforms as shown in the Figure 25.

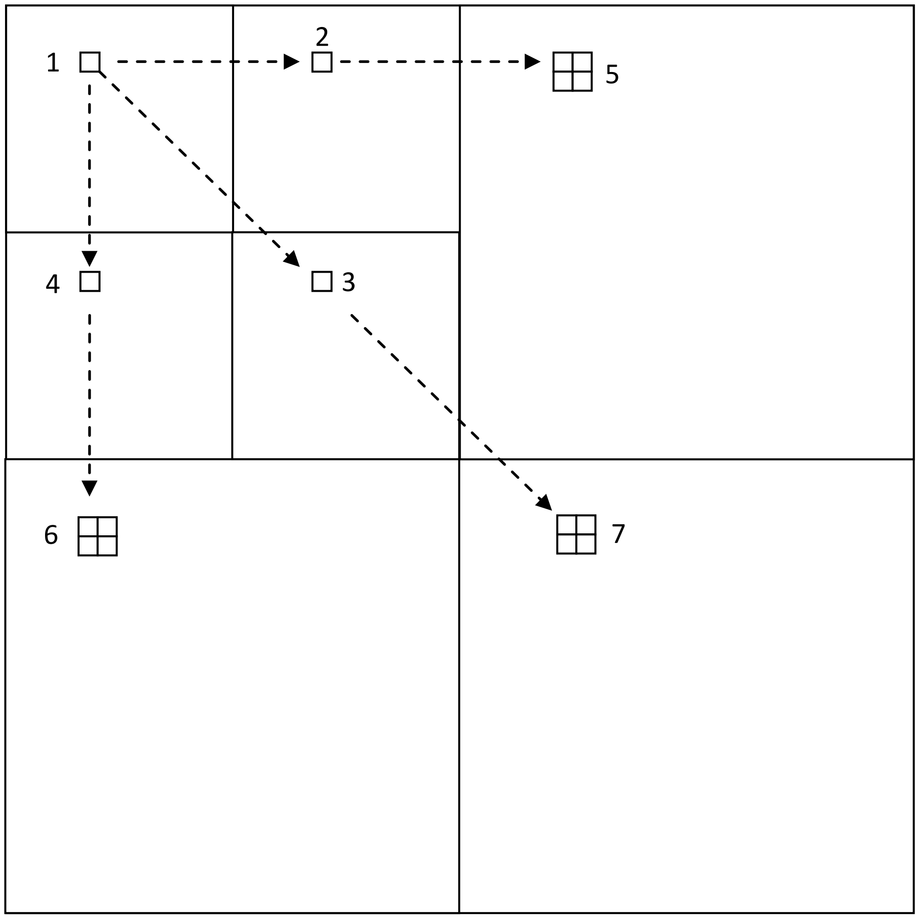


Figure 25 Tree structure used Encoding

The encoder will generate two streams of bits which will be saved in two registers namely, Flag Register and Data Register. The algorithm follows bitplane coding, starting with the bitplane corresponding to the MSB and moves down to LSB. In each bitplane the algorithm will search for a zero tree start from point 1 or points 2, 3, 4. If a zero tree is encountered starting from point 1, a bit ‘1’ is saved in the flag register corresponding to the point 1 position. In such a case a group of 16 bits is represented by a single bit. Another case is a zero tree starting from points 2, 3 or 4, i.e. the point 1 is having a non zero bit while all or some of the point 2, 3, 4 forms a zero tree. In this case, corresponding to the points 1 and non zero-tree points of 2, 3 & 4, flag registers will have ‘0’ bit. A ‘0’ bit encountered in the flag register, implies data corresponding to the positions of that tree is stored in the data register. A zero tree at points 2, 3, 4 will substitute five bits by a single bit.

Output Waveform and Compression Ratio

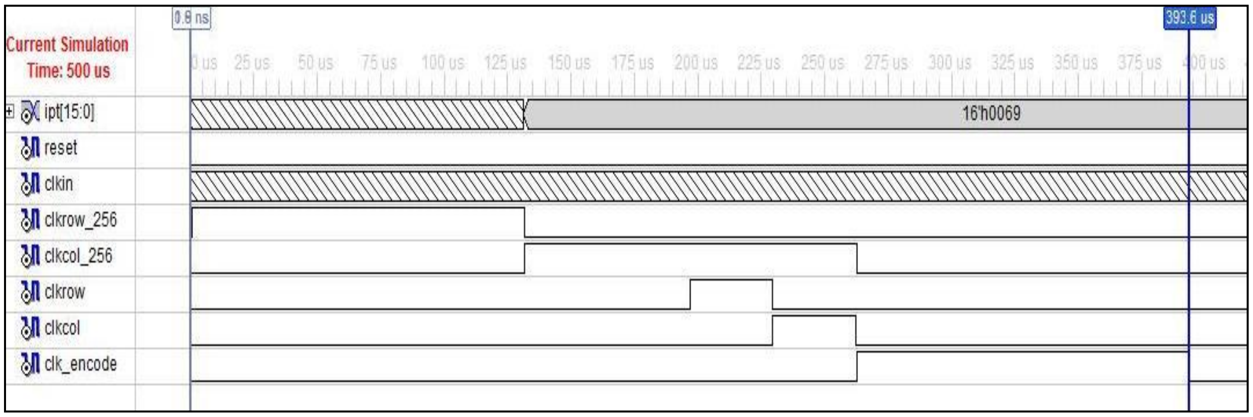


Figure 26 Output waveform for 2-level DWT in Verilog HDL

The output waveform shown in Figure 26 gives a clear picture of the order of operations performed with time. There are six different clocks. ‘clkin’ is the input clock which synchronizes the serial input of the image. ‘clkrow_256’ maintains a high level during row-wise convolution. Once the row-wise convolution is over, column-wise convolution begins as indicated by the high level of ‘clkcol_256’. When the column-wise convolution reaches half way, the second level of DWT’s row-wise convolution starts as is indicated by the signal ‘clkrow’. The row-wise convolution of second level DWT is followed by column-wise convolution of second level DWT. The column-wise convolution of DWT for both the levels finishes at the same time. Having calculated all the transforms coefficients, the encoding process is initialised by making the ‘clk_encode’ high. The compression ratio attained by the algorithm is 0.9895.

Conclusions

An image compression algorithm was simulated using Matlab to comprehend the process of image compression. Modifications on the padding style showed reduction in the error, because it offers a better reproduction of image at its edges. It also supports faithful reproduction of the image, keeping the size of the transform coefficient matrix equal to the image size.

For the VLSI implementation of an image compression encoder, Verilog HDL was chosen. Understanding the importance of the multiplier in implementing a Discrete Wavelet Transform(DWT), fixed-point 16-bit signed Booth Multipliers were implemented in Verilog HDL for different architectures and a thorough analysis in terms of timing, power and area were carried out. Finally the Booth Multiplier architecture using 17-bit Carry Save Adder for adding partial products and a 31-bit Ripple Carry Adder for adding pseudo sum and carry vector was selected to perform the multiplication operation in DWT. Selection of this architecture was based on the least area and its better performance in timing and power analysis. The discrete wavelet transform was computed using a customised code to reduce the redundancy and to avoid the needless computation. The transform coefficients obtained after DWT is encoded by exploiting the presence of zero trees to obtain the compressed form of the image. The compressed image was stored using two bit streams namely, flag register and data register which complements each other to represent the image in a compressed form.

Reference

[1] Olivier Rioul and Martin Vetterli, "Wavelets and Signal Processing", IEEE Trans. on Signal Processing, Vol. 8, Issue 4, pp. 14 - 38 October 1991.

[2] D. S. Taubman, "High performance scalable image compression with EBCOT", IEEE Transaction Image Processing, Vol. 9, No. 7, pp. 1158–1170, July 2000

[3] M. Antonini, M. Barlaud, P. Mathieu, and I. Daubechies, "Image Coding Using Wavelet Transform," IEEE Trans. on Image Processing, Vol. 1, No.2, pp. 205-220, April 1992.

[4] "Wavelet filter evaluation for image compression". al., J. Liao et. August 1995, IEEE Trans. Image Process., Vol. 4, pp. 1053–1060.

[5] "JPEG 2000 Final Committee Draft". Boliek, M. 2000.

[6] Evaluation of Design Alternatives for the 2-D-Discrete Wavelet Transform. Nikos D. Zervas, Giorgos P. Anagnostopoulos, Vassilis Spiliotopoulos, Yiannis Andreopoulos, and Costas E. Goutis. DECEMBER 2001, IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS FOR VIDEO TECHNOLOGY, Vols. 11, NO. 12, pp. 1246-1262.

[7] J. M. Shapiro, "Embedded Image Coding Using Zerotrees of Wavelet Coefficients," IEEE Trans. on Signal Processing, Vol. 41, No. 12, pp. 3445-3462, December 1993.

[8] A. Said and W. A. Pearlman, "A New Fast and Efficient Image Codec Based on Set Partitioning in Hierarchical Trees," IEEE Trans. on Circuits and Systems for Video Technology, Vol. 6, No.3, pp. 243-250, June 1996.

[9] S. Mallat, "A Theory for Multiresolution Signal Decomposition: The Wavelet Representation," IEEE Trans. on Pattern Analysis and Machine Intelligence, Vol. 11, No.7, pp. 674-693, July 1989.

[10] JPEG2000 Final Committee Draft (FCD), "JPEG2000 Committee Drafts," http:j jwww.jpeg.orgjCDsI5444.htm.

[11] I. Daubechies, "The Wavelet Transform, Time-Frequency Localization and Signal Analysis," IEEE Trans. on Inform. Theory, Vol. 36, No. 5, pp. 961-1005, September 1990.

[12] I. Daubechies, Ten Lectures on Wavelets. SIAM, CBMS series) Philadelphia, 1992.

[13] D. A. Huffman, "A Method for the Construction of Minimum-Redundancy Codes," proceedings of the IRE, Vol. 40, No.9, pp. 1098-1101, 1952.

[14] H. Witten, R. M. Neal, and J. G. Cleary, "Arithmetic Coding for Data Compression," Communications of the ACM, Vol. 30, No.6, June 1987

[15] B. Fu and K. K. Parhi, "Generalized Multiplication-Free Arithmetic Codes," IEEE Transactions on Communications, Vol. 45, No.5, May 1997.
[16] A. N. Skodras, C. A. Christopoulos, and T. Ebrahimi, "JPEG2000: The Upcoming Still Image Compression Standard," Proceedings of the 11th Portuguese Conference on Pattern Recognition, Porto, Portugal, pp. 359366, May 11-12, 2000.
[17] A. Skodras, C. Christopoulos, and T. Ebrahimi, "The JPEG2000 Still Image Compression Standard," IEEE Signal Processing Magazine, pp. 36-58, September 2001.
[18] D. S. Taubman and M. W. Marcellin. JPEG2000: Image Compression Fundamentals, Standards and Practice. Kluwer Academic Publishers, MA,2002.
[19] R. A. DeVore, B. Jawerth, and B. J. Lucier, "Image Compression Through Wavelet Transform Coding," IEEE Trans. on Information Theory, Vol. 38, No.2, pp. 719-746, March 1992.
[20] A. S. Lewis and G. Knowles, "Image Compression Using the 2-D Wavelet Transform," IEEE. Trans. on Image Processing, Vol. 1, No.2, pp. 244250, April 1992.
[21] A. Said and W. A. Peralman, "An Image Multiresolution Representation for Lossless and Lossy Compression," IEEE Trans. on Image Processing, Vol. 5, pp. 1303-1310, September 1996.
[22] P. Chen, T. Acharya, and H. Jafarkhani, "A Pipelined VLSI Architecture for Adaptive Image Compression," International Journal of Robotics and Automation, Vol. 14, No.3, pp. 115-123, 1999.
[23] W. B. Pennebaker and J. L. Mitchell, JPEG Still Image Compression Standard. Chapman & Hall, New York, 1993.
[24] ISO/IEC 15444-1, "Information Technology-JPEG2000 Image Coding System-Part 1: Core Coding System," 2000.
[25] ISO/IEC 15444-2, Final Committee Draft, "Information TechnologyJPEG2000 Image Coding System-Part 2: Extensions," 2000.
[26] ISO/IEC 15444-3, "Information Technology-JPEG2000 Image Coding System-Part 3: Motion JPEG2000," 2002.
[27] ISO/IEC 15444-4, "Information Technology-JPEG2000 Image Coding System-Part 4: Conformance Testing," 2002.
[28] ISO/IEC 15444-5, "Information Technology-JPEG2000 Image Coding System-Part 5: Reference Software," 2003.
[29] ISO/IEC 15444-6, Final Committee Draft, "Information TechnologyJPEG2000 Image Coding System-Part 6: Compound Image File Format," 2001.

[30] I. Daubechies and W. Sweldens, "Factoring Wavelet Transforms into Lifting Schemes," Journal of Fourier Analysis and Applications, Vol. 4, pp. 247-269, 1998.
[31] D. L. Gall and A. Tabatabai, "Subband Coding of Digital Images Using Symmetric Short Kernel Filters and Arithmetic Coding Techniques," Proceedings of IEEE International Conference Acoustics, Speech, and Signal Processing, Vol. 2, pp. 761-764, New York, April 1988.
[32] D. Nister and C. Christopoulos, "Lossless Region of Interest with Embedded Wavelet Image Coding," Signal Processing, Vol. 78, No.1, pp. 1-17, 1999.
[33] E. Atsumi and N. Farvardin, " Loss/Lossless Region-of-Interest Image Coding Based on Set Partitioning in Hierarchical Tree," IEEE International Conference Image Processing, pp. 87-91, Chicago, October 1998.
[34] Forrid N Najam (1994), "A Survey of Power Estimation in VLSI Circuits", IEEE Transactions on VLSI Systems vol.2, No.4, Dec-1994.
[35] Christian Piguet, Taylor, Francis (2006), "Low Power CMOS Circuits, Technology, Logic Design and CAD Tools" by - 2006.
[36] Maddu Karunaratne, Chamara Ranasinghe (2008), "A Dynamic Switching Activity Generation Technique for Power Analysis of Electronic Circuits", 48th Midwest Symposium on Circuits and Systems.
[37] A. D. Booth (1951), "A signed binary multiplication technique," Quart. J. Mech. Appl. Math., vol. 4, pp. 236-240, 1951. (Reprinted in [8, pp. 100-104.]).
[38] O. L. MacSorley (1961), "High speed arithmetic in binary computers," Proc .IRE, Jan. 1961.
[39] Philip E. Madrid, Brian Millar, and Earl E. Swartzlander Jr. (1993), "Modified Booth Algorithm for High Radix Fixed-Point Multiplication", IEEE Transactions on Very Large Scale Integration (VLSI) Systems, vol. 1, No. 2, June-1993.
[40] Milos D. ERCEGOVAC, Tomas LANG (2004), "Digital Arithmetic".
[41] Power Compiler Manuals, www.synopsys.com.
[42] Prime Power manuals, www.synopsys.com.