FPGA IMPLEMENTATION OF DISCRETE WAVELET TRANSFORM BASED IMAGE COMPRESSION

A DISSERTATION

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

MASTER OF TECHNOLOGY

in

ELECTRONICS AND COMMUNICATION ENGINEERING

(With Specialization in Control and Guidance)

By

U. M. R. CH. SEKHAR BOLLAM

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

JUNE, 2009

CANDIDATE'S DECLARATION

I hereby declare that the work presented in this dissertation entitled **"FPGA Implementation of Discrete Wavelet Transform based Image Compression"** being submitted in partial fulfillment of the award of the degree of **Master of Technology** with specialization in Control & Guidance in the Department of Electronics & Computer Engineering, **Indian Institute of Technology, Roorkee,** under the guidance of **Dr. M. J. Nigam**, Associate Professor, Department of Electronics & Computer Engineering, Indian Institute of Technology Roorkee.

The results embodied in this dissertation have not submitted for the award of any other Degree or Diploma.

Date : 29/06/2009

Place : Roorkee

B. Sekhar

U M R Ch Sekhar Bollam

CERTIFICATE

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

Date: 29.06.09

Place: Roorkee

Dr. M.J.Nigam, Associate Professor,
Department of Electronics & Computer,
Indian Institute of Technology Roorkee

ACKNOWLEDGEMENT

I express my foremost and deepest gratitude to **Dr. M. J. Nigam**, Associate Professor, Department of Electronics & Computer Engineering, Indian Institute of Technology Roorkee for his valuable guidance, support and motivation throughout this work. The valuable hours of discussion and suggestions that I had with him have undoubtedly helped in supplementing my thoughts in the right direction for attaining the desired objective. I consider myself extremely fortunate for having got the opportunity to learn and work under his able supervision over the entire period of my association with him.

My sincere thanks to all faculty members of Control & Guidance for their constant encouragement and suggestions towards the successful completion of this work. My sincere thanks to laboratory staff to access the computers and other resources at will for completion of this work. I am also thankful to my friends who helped me in many different ways at various instances of my work.

Last but not the least, I'm highly indebted to my parents and family members, whose sincere prayers, best wishes, moral support and encouragement have a constant source of assurance, guidance, strength, and inspiration to me.

U M R Ch Sekhar Bollam
M. Tech. (C&G)

ABSTRACT

Images require substantial storage and transmission resources, thus image compression is advantageous to reduce these requirements.

Wavelet Transform has been successfully applied in different fields, ranging from pure mathematics to applied sciences. Pure software implementations of the Discrete Wavelet Transform, however, appear to be the performance bottleneck in real-time systems and suffer from power requirements. Therefore, hardware acceleration of the Discrete Wavelet Transform has become a topic of interest. We can reduce the logic and speed up our operations using Application Specific Integrated Circuits (ASICs). But main problem with ASICs are they require large time to market and initial investments are high. Before developing an ASIC we require to prototype our design. Field programmable Gate Arrays (FPGAs) prove to be a better solution for rapid prototyping. FPGAs are reprogrammable, have large number of logic cells suitable for implementing image Compression applications. We can explore the parallelism and pipelining feature of FPGA

The objective of this dissertation is design, simulation and synthesis of CDF biorthogonal Discrete Wavelet Transform on FPGA and developing a prototype of CDF biorthogonal Discrete Wavelet Transform processor. Initially, CDF Wavelet is studied and the hardware logic is designed. Then this hardware logic is realized and Simulated in Matlab Simulink using Xilinx System generator Blockset and synthesized on Spartan3E xc3s500e-4fg320 FPGA chip. Then using hardware co-simulation feature of Spartan-3E starter kit, the results obtained in software and hardware simulations (i.e. on FPGA kit), are validated. The decomposed image of 512 x 512 at different levels from Matlab Silmulink using System generator shows good decomposition. Decomposed images has been reconstructed by using Matlab code and compared with the original images.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

Chapter 1

INTRODUCTION

1.1. Background

Computer data compression is a powerful, enabling technology that plays a vital role in the information age. Among the various types of data commonly transferred over networks, image and video data comprises the bulk of the bit traffic which is growing day by day. For example, current estimates indicate that image data has taken over 40% of the volume on the Internet. The explosive growth in demand for image and video data, coupled with delivery bottlenecks has kept compression technology at a premium. Although increasing the bandwidth is a possible solution, the relatively high cost makes this option less attractive. Therefore, compression is a necessary and essential method for creating image files with manageable and transmittable sizes.

Joint Photographic Experts Group (JPEG) [1] and Moving Pictures Experts Group (MPEG) are standards for representing images and video. Data compression algorithms are used in those standards to reduce the number of bits required to represent an image or a video sequence. Compression is the process of representing information in a compact form. Data compression treats information in digital form that is, as binary numbers represented by bytes of data with very large data sets. Every compression algorithm has a corresponding decompression algorithm without which it is not employable for usage. Given the compressed file, the original file can be reproduced with the help of decompression algorithm. There have been many types of compression algorithms developed till date. These algorithms fall into two broad categories, which are: lossless algorithms and lossy algorithms. A lossless algorithm reproduces the original exactly. A lossy algorithm, as the name implies, loses some data but has high compression ratio. Data loss may be unacceptable in many applications. For example, text compression must be lossless because a very small difference can result in statements with totally different meanings. There are also many situations where loss may be either unnoticeable or acceptable. In image compression, for example, the exact reconstructed value of each sample of the image

is not necessary. Depending on the quality required of the reconstructed image, varying amounts of loss of information can be accepted.

The newer standard JPEG2000 is based on the Wavelet Transform (WT). Wavelet Transform offers multi-resolution image analysis, which appears to be well matched to the low level characteristic of human vision. The Discrete Cosine Transform (DCT) is essentially unique whereas WT has many possible realizations. Wavelets are the mathematical functions that satisfy a certain requirement (for instance a zero mean), and are used to represent data or other functions. In wavelet transform, dilations and translations of a mother wavelet are used to perform a spatial/frequency analysis on the input data. Recent research on Discrete Wavelet Transform (DWT) has focused on a form of lifting which shows excellent performance compared to the conventional convolution method for implementation. Factoring DWT into lifting steps can reduce the computational complexity by 50% [2] and has advantages, including integer to integer transform [3], symmetric forward and inverse transforms [4]. Line-based architecture for the direct two-dimensional discrete wavelet transform (2D-DWT) is an efficient alternative tradeoff between speed and area [5,6]. For image compression using line based architecture, first all the rows are processed, intermediate results are stored in buffer and then all the columns of intermediate results are processed. The disadvantage of the above method is that, it requires intermediate buffer size equal to size of image, high computation time besides underutilization of hardware. To overcome these disadvantages, parallel architectures are developed in a way to reduce the intermediate buffer size, thus reducing computation time almost by half.

For any image processing system, the important trade-off is time taken to process the image. Naturally, the simple solution is to use a microprocessor or a Digital Signal Processor to implement the algorithms. But this system suffers from a demerit which requires the use of random 'glue logic' to connect large ICs (for example, in generating global control signals and data formatting like serial to parallel conversion, multiplexing, etc.). An alternative is to use dedicated hardware like Application Specific Integrated Circuits (ASICs) built specially for the job, which reduced the effect of 'glue logic'. Typically this has always been a far more expensive

alternative and required more time-to-market. The third solution that presents itself is the use of programmable electronics in the form of Field Programmable Gate Arrays.

Field Programmable Gate Arrays (FPGAs) provide a rapid prototyping platform. FPGAs are devices that can be reconfigured [7] to achieve different functionalities without incurring the non-recurring engineering costs typically associated with custom IC fabrication. In this work, DWT architecture is implemented on a reconfigurable FPGA hardware. The target platform is the Xilinx Spartan-3E FPGA. The design is based on the multi-level decomposition implementation of the Discrete Wavelet Transform. The design utilizes various techniques and specific features of the Xilinx Spartan-3E FPGA to accelerate the computation of the transform. Performance analysis includes the investigation of performance enhancement due to hardware acceleration. It is expected that the proposed design can substantially accelerate the DWT and the inherent scalability can be exploited to reach a higher performance in the future. The implementation can be easily modified to act as a co-processing environment for wavelet compression/decompression or even as a part of the algorithms to be used in future mobile devices for image encoding/decoding using wavelets.

1.2. Motivation and Scope

A majority of today's Internet bandwidth is estimated to be used for images and video [8]. Recent multimedia applications for handheld and portable devices place a limit on the available wireless bandwidth. The bandwidth is limited even with new connection standards. JPEG image compression which is in widespread use today took several years to achieve perfection. Wavelet based techniques such as JPEG2000 [9] for image compression has a lot more to offer than conventional methods in terms of compression ratio. Currently, wavelet implementations are still under development lifecycle and are being perfected. Flexible energy-efficient hardware implementations that can handle multimedia functions such as image processing, coding and decoding are critical, especially in hand-held portable multimedia wireless devices.

As a part of this dissertation, a prototype of Wavelet Transform Processor is developed. This developed prototype may be suitable for ASIC implementation. It can be used in bio-medical image processing applications, and in many more fields.

1.3. Statement of Problem

In modern hardware design, it is a fact that storage resource is more expensive than computation resource. So, the key problem in hardware implementation is to achieve high performance while maintaining low memory requirement.

The objectives of this dissertation work are:

- Study of parallel architectures for image compression using various wavelets on FPGA implementation.
- Developing an improved architecture of 2D-DWT to implement bi-orthogonal Cohen-Daubechies-Feuvea (CDF) wavelet.
- Realizing CDF biorthogonal Wavelet Transform hardware logic using System generator simulator and to carry out the simulations on standard images.
- Implementation on FPGA kit to be done using hardware co-simulation. For hardware co-simulation, we need to model our architecture using Xilinx Blockset in Matlab. Then these algorithms are synthesized and the Spartan-3E starter kit is setup to enable hardware-in-the-loop verification with JTAG co-simulation via the USB configuration port.

1.4. Organization of the Dissertation

Chapter 1 gives a background discussion on image compression, wavelet based image compression and Field programmable gate arrays. The Motivation and Scope of work is discussed. It summarizes the problem statement of this thesis work.

Chapter 2 reviews the complete discrete wavelet transform in detail. The need of wavelet transform in image compression and its features are also discussed. Finally, salient features of multi-resolution analysis of DWT are discussed.

Chapter 3 presents need for compression, principles behind compression, compression model, 2D wavelet analysis, direct form structure, polyphase structure and lifting structure for implementing DWT.

Chapter 4 briefly describes FPGA architecture, design flow followed in FPGA implementation and FPGA configuration.

Chapter 5 explains the algorithm for implementing CDF wavelet. CDF wavelet Simulink model design in system generator and implementation on FPGA are described.

Chapter 6 discusses the Simulink and synthesis results.

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

Chapter 2

WAVELET TRANSFORM

Many evolving multimedia applications require transmission of high quality images over the network. One obvious way to accommodate this demand is to increase the bandwidth available to all users. Of course, this "solution" is not without technological and economical difficulties. Another way is to reduce the volume of the data that must be transmitted. There has been a tremendous amount of progress in the field of image compression during the past 15 years. In order to make further progress in image coding, many research groups have begun to use wavelet transforms.

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

2.1. Need for Wavelet Transforms

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

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

To solve this problem, the Short-Time Fourier Transform (STFT) was introduced. The major drawback of the STFT is that it uses a fixed window width.

The wavelet transform [10], which was developed in the last two decades, provides a better time-frequency representation of the signal than any other existing transforms.

2.2. Short Time Fourier Transform Vs Wavelet Transform

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

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

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

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

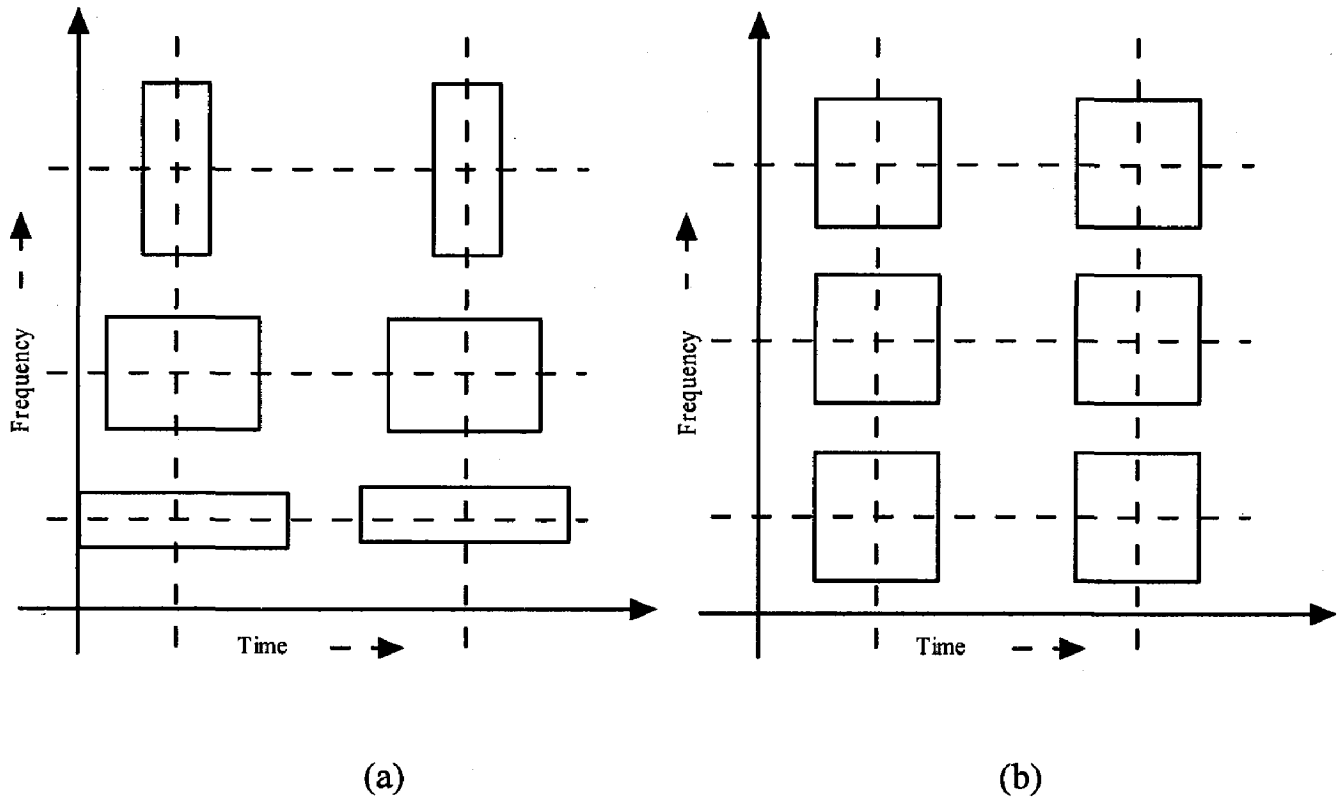


Fig.2.1. Tiling in time-frequency plane by: **(a)** Wavelets and **(b)** STFT [11]

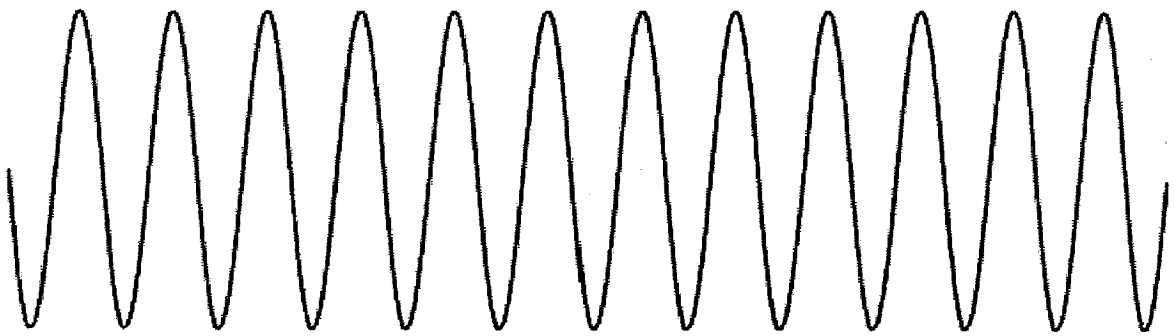
Fig.2.1(a) shows tiling in time-frequency of wavelets and Fig.2.1(b) shows tiling in short-time fourier transform, It is seen that STFT gives a fixed resolution at all times, whereas wavelet transform gives a variable resolution.

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

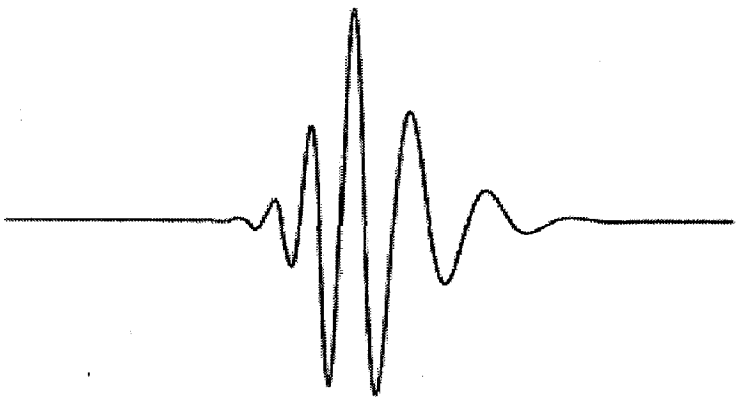
2.3. Wave and Wavelet

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

The wavelet analysis is done similar to the STFT analysis. The signal to be analyzed is multiplied with a wavelet function just as it is multiplied with a window function in STFT, and then the transform is computed for each segment generated. However, unlike STFT, in wavelet transform, the width of the wavelet function changes with each spectral component. The wavelet Transform, at high frequencies, gives good time resolution and poor frequency resolution, while at low frequencies; the Wavelet Transform gives good frequency resolution and poor time resolution.



(a)



(b)

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

2.4. Continuous Wavelet Transform

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

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

The mother wavelet used to generate all the basis functions is designed based on some desired characteristics associated with that function. The translation parameter τ relates to the location of the wavelet function as it is shifted through the signal. Thus, it corresponds to the time information in the Wavelet Transform. The scale parameter *s* is defined as |1/frequency| and corresponds to frequency information. Scaling either dilates (expands) or compresses a signal. Large scales (low frequencies) dilate the signal and provide detailed information hidden in the signal, while small scales (high frequencies) compress the signal and provide global information about the signal. Notice that the Wavelet Transform merely performs the convolution operation of the signal and the basis function. The above analysis becomes very useful as in most practical applications, high frequencies (low scales) do not last for a long duration, but instead, appear as short bursts, while low frequencies (high scales) usually last for entire duration of the signal.

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

2.5. Discrete Wavelet Transform

The Wavelet Series is just a sampled version of CWT and its computation may consume significant amount of time and resources, depending on the resolution required. The Discrete Wavelet Transform (DWT) [10], which is based on sub-band

coding is found to yield a fast computation of Wavelet Transform. It is easy to implement and reduces the computation time and resources required.

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

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

2.5.1. Multi-Resolution Analysis Using Filter Banks

Filters are one of the most widely used signal processing functions. Wavelets can be realized by iteration of filters with rescaling. The resolution of the signal, which is a measure of the amount of detail information in the signal, is determined by the filtering operations, and the scale is determined by upsampling and downsampling (sub sampling) operations [11-13].

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

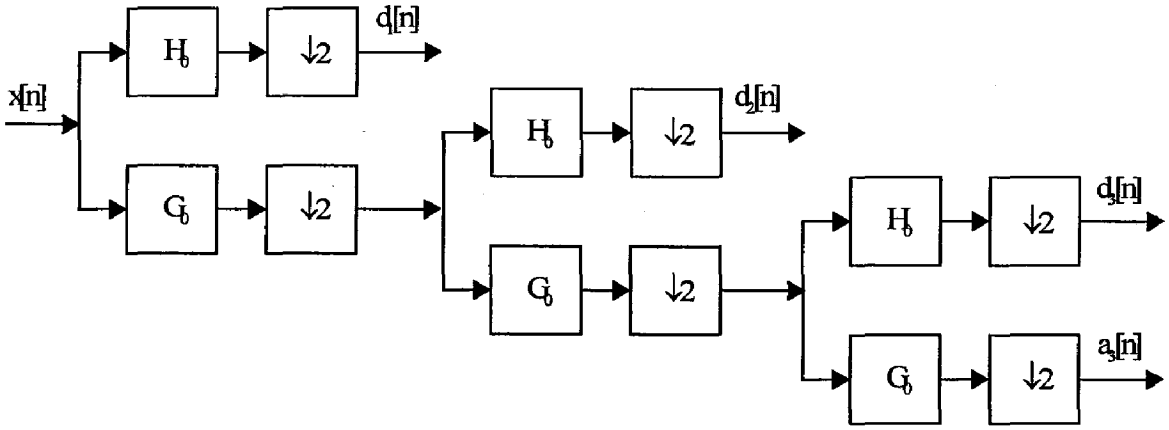


Fig.2.3. Three-level wavelet decomposition tree

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

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

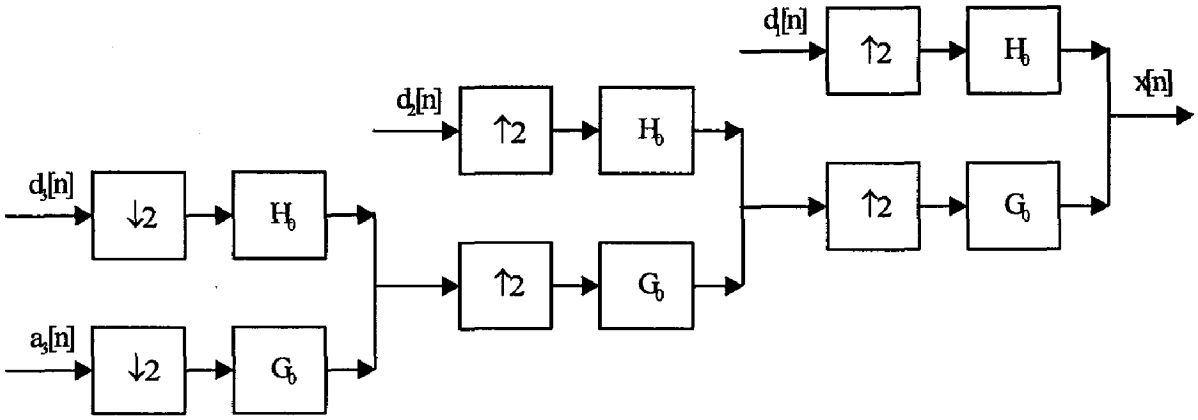


Fig.2.4. Three-level wavelet reconstruction tree

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

2.6. Classification of Wavelets

We can classify wavelets into two classes: (a) orthogonal [8] and (b) biorthogonal [9]. Based on the application, either of them can be used.

2.6.1. Features of Orthogonal Wavelet Filter Banks

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

$$H_0(z) = z^{-N} G_0(-z^{-1}) \tag{2.2}$$

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

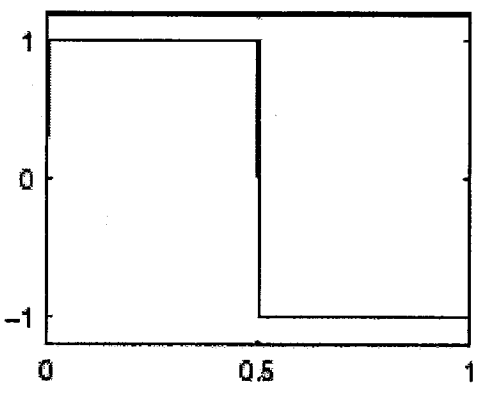
2.6.2. Features of Biorthogonal Wavelet Filter Banks

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

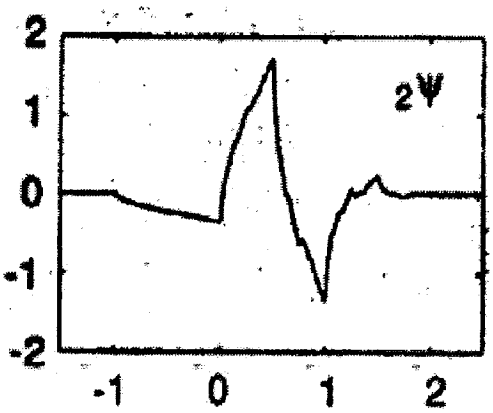
For perfect reconstruction, biorthogonal filter bank has all odd length or all even length filters. The two analysis filters can be symmetric with odd length or one symmetric and the other anti-symmetric with even length. Also, the two sets of analysis and synthesis filters must be dual. The linear phase biorthogonal filters are the most popular filters for data compression applications.

2.7. Wavelet Families

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



(a)



(b)

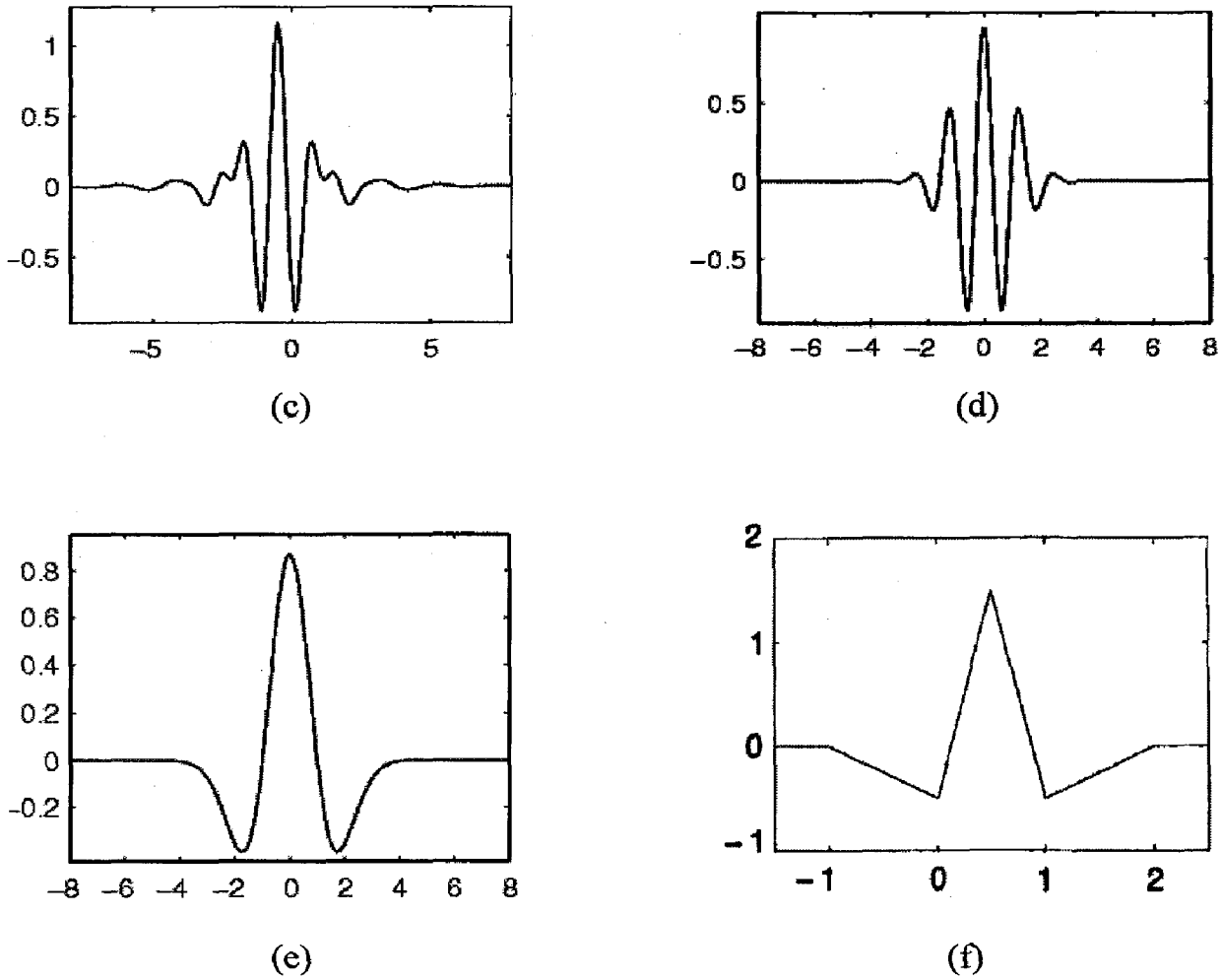


Fig.2.5. Wavelet families **(a)** Haar **(b)** Daubechies4 **(c)** Meyer **(d)** Morlet **(e)** Mexican Hat **(f)** CDF (2,2)

Fig.2.5 illustrates some of the commonly used wavelet functions. Haar wavelet is one of the oldest and simplest wavelet. Therefore, any discussion of wavelets starts with the Haar wavelet. Daubechies wavelets are the most popular wavelets[16]. They represent the foundations of wavelet signal processing and are used in numerous applications. Haar and Daubechies4 wavelets along with Meyer wavelets are capable of perfect reconstruction. The Meyer, Morlet and Mexican Hat wavelets are symmetric in shape. The wavelets are chosen based on their shape and their ability to analyze the signal in a particular application. CDF (2,2) wavelet is also known as the biorthogonal (5,3) wavelet because of the filter length of 5 and 3 for the low and high pass filters, respectively.

Chapter 3

WAVELET BASED IMAGE COMPRESSION

Uncompressed multimedia (graphics, audio and video) data requires considerable storage capacity and transmission bandwidth. Despite rapid progress in mass-storage density, processor speeds, and digital communication system performance, demand for data storage capacity and data-transmission bandwidth continues to outstrip the capabilities of available technologies. The recent growth of data intensive multimedia-based web applications have not only sustained the need for more efficient ways to encode signals and images but have made compression of such signals central to storage and communication technology[16].

In the field of image processing, image compression is the current topic of research. Image compression plays a crucial role in many important and diverse applications, including televideoconferencing, remote sensing, document & medical and facsimile transmission.

3.1. Need for Compression

Image data is by its nature multidimensional and tend to take up a lot of space

- Pictures take up a lot of storage space (either disk or memory).
- A 1000x1000 picture with 24 bits per pixel takes up 3 megabytes.
- The Encyclopedia Britannica scanned at 300 pixels per inch and 1 bit per pixel requires 25,000 pages ×1,000,000 bytes per page = 25 gigabytes.
- Video is even bulkier: 90 minute movie at 640×480 resolution spatially, 24 bit per pixel, 24 frames per second, requires 90×60×24×640×480×3=120 gigabytes.

The examples above clearly illustrate the need for sufficient storage space, large transmission bandwidth, and long transmission time for image, audio, and video data. At the present state of technology, the only solution is to compress multimedia data before its storage and transmission, and decompress it at the receiver for play back. For example, with a compression ratio of 32:1, the space, bandwidth, and

transmission time requirements can be reduced by a factor of 32, with acceptable quality.

- Applications: HDTV, film, remote sensing and satellite image transmission, network communication, image storage, medical image processing, fax.

3.2. Principles behind Compression

A common characteristic of most images is that the neighboring pixels are correlated and therefore contain redundant information. The foremost task then is to find less correlated representation of the image. Two fundamental components of compression are redundancy and irrelevancy reduction. Redundancy reduction aims at removing duplication from the signal source (image/video). Irrelevancy reduction omits parts of the signal that will not be noticed by the signal receiver, namely the Human Visual System (HVS). In general, three types of redundancy can be identified:

- Spatial Redundancy or correlation between neighboring pixel values.
- Spectral Redundancy or correlation between different color planes or spectral bands.
- Temporal Redundancy or correlation between adjacent frames in a sequence of images (in video applications).

Image compression research aims at reducing the number of bits needed to represent an image by removing the spatial and spectral redundancies as much as possible. Since we will focus only on still image compression, we will not worry about temporal redundancy. Different methods for redundancy reduction are

- Spatial redundancy: DCT, DWT, DPCM
- Statistical redundancy: Run-Length coding, Variable-Length coding

3.3. Image compression model

A typical image compression model consists of source encoder which is responsible for reducing or eliminating any coding, interpixel or psychovisual redundancies in the input image. Channel is a transmission path and source decoder

reconstructs the original image whose function is opposite to that of source encoder. The figure.3.1 shows the block diagram of image compression model [17].

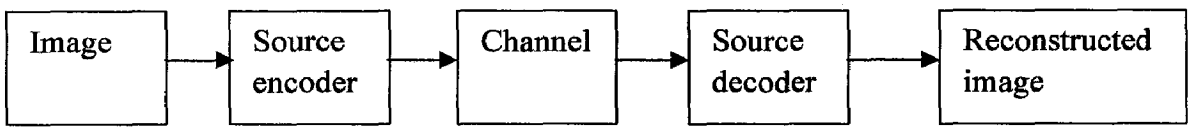
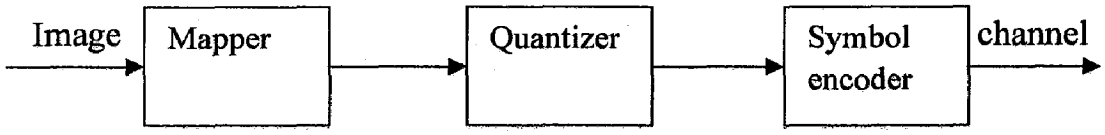
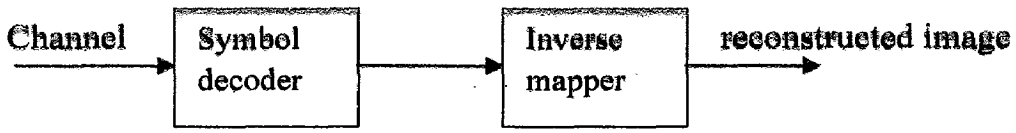


Fig.3.1. Image compression model



(a) Source encoder



(b) Source decoder

Fig.3.2. (a) source encoder **(b)** source decoder

The source encoder consists of three blocks. The first stage of the source encoding process, the mapper transforms the input data into a formate designed to reduce interpixel redundancies in the input image. This operation is generally reversible and may or may not reduce directly the amount of data required to represent the image.

The second stage, or quantizer block in figure.3.2 (a), reduces the accuracy of the mappers output in accordance with some pre established fidelity criterion. The stage reduces the psychovisual redundancies of the input image. This operation is irreversible. Thus it must be omitted when error free compression is desired.

In the third and final stage of the source encoding process, the symbol coder block in figure.3.2. (a) creates a fixed- or variable-length code to represent the quantizer output and maps the output in accordance with the code.

The source decoder shown in figure.3.2 (b) contains only two components symbol decoder and an inverse mapper. These blocks perform, in reverse order, the inverse operations of the source encoder's, symbol encoder and mapper blocks.

3.4. Image Compression Techniques

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

3.4.1. Lossless compression

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

3.4.2. Lossy compression

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

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

3.5. Wavelet Transform as the Source Encoder

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

edges in between the smooth variations. Technically, the smooth variations in color can be termed as low frequency variations, and the sharp variations as high frequency variations.

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

Separating the smooth variations and details of the image can be performed in many ways. One way is the decomposition of the image using the discrete wavelet transform. Digital image compression is based on the ideas of sub-band decomposition or discrete wavelet transforms. Wavelets which refer to a set of basis functions are defined recursively from a set of scaling coefficients and scaling functions. The DWT is defined using these scaling functions and can be used to analyze digital images with superior performance than classical short-time Fourier-based techniques, such as the DCT. The basic difference between wavelet-based and Fourier-based techniques is that short-time Fourier-based techniques use a fixed analysis window, while wavelet-based techniques can be considered using a short window at high spatial frequency data and a long window at low spatial frequency data. This makes DWT more accurate in analyzing image signals at different spatial frequency, and thus can represent more precisely both smooth and dynamic regions in image. The compression system includes forward wavelet transform, a quantizer, and a lossless entropy encoder. The corresponding decompressed image is formed by the lossless entropy decoder, a de-quantizer, and an inverse wavelet transform. Wavelet-based image compression has good compression results in both rate and distortion sense.

3.6. 2D Wavelet Analysis

Images are treated as two dimensional signals, they change horizontally and vertically, thus 2D wavelet analysis must be used for images. 2D wavelet analysis uses the same 'mother wavelets' but requires an extra step at every level of decomposition. The 1D analysis filtered out the high frequency information from the low frequency information at every level of decomposition; so only two subsignals

were produced at each level. In 2D, the images are considered to be matrices with N rows and M columns. At every level of decomposition the horizontal data is filtered, and then the approximation and details produced from this are filtered on columns.

At every level, four sub-images are obtained; the approximation, the vertical detail, the horizontal detail and the diagonal detail. Below the Saturn image has been decomposed to one level. The wavelet analysis has found how the image changes vertically, horizontally and diagonally.

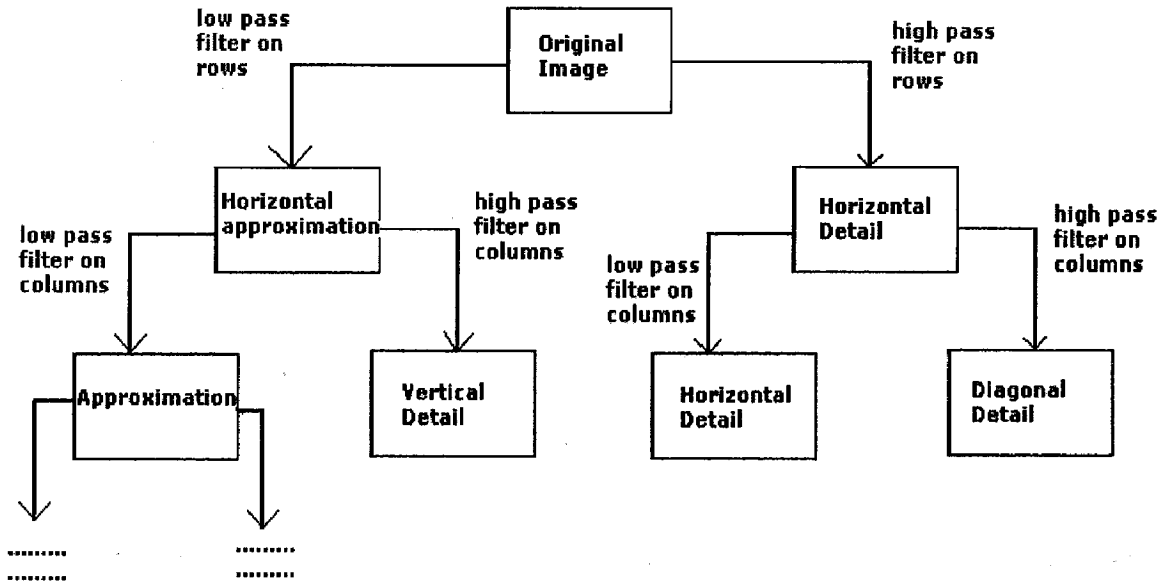


Fig. 3.3. Sub band decomposition of 2-D image [29].

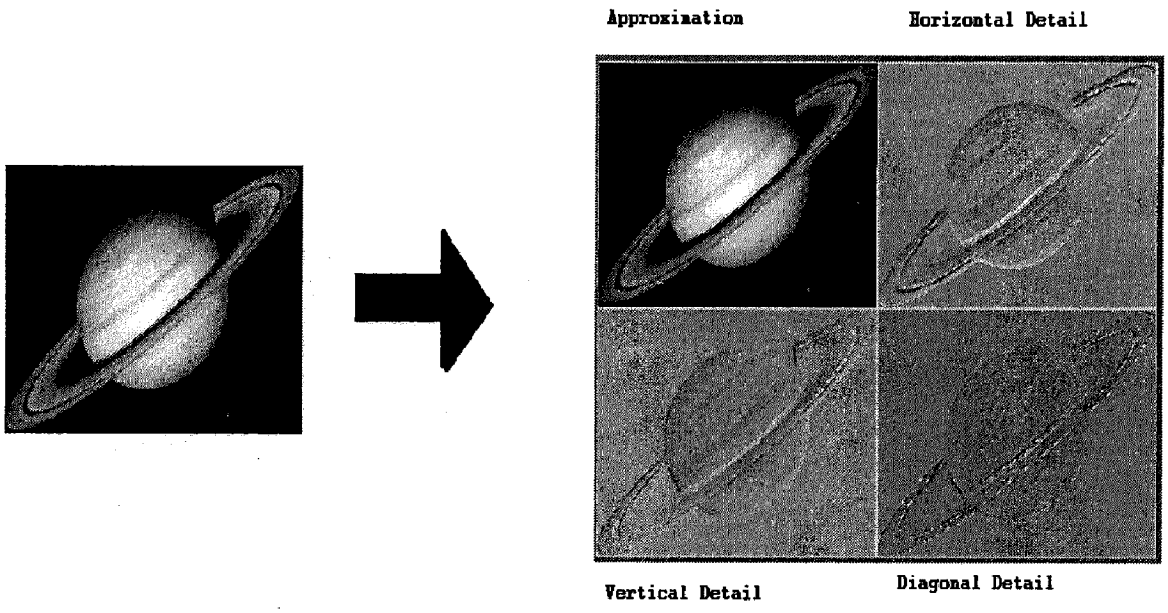


Fig. 3.4. 2-D Decomposition of Saturn Image to level 1[29].

3.7. Features of Image Compression Using Wavelets

The key features of wavelet-based compression schemes are:

- Wavelets provide good compression ratios. Especially for high compression ratios, wavelets perform much better than competing technologies like JPEG [18], both in terms of signal-to-noise ratio and visual quality. Unlike JPEG, they show no blocking effect but allow for a graceful degradation of the whole image quality, while preserving the important details of the image. The next version of the JPEG standard (JPEG 2000) will incorporate wavelet based compression techniques.
- The wavelet transform is a fast operation (linear to the amount of data), especially when implemented using the lifting scheme. The wavelet transform is symmetric: both the forward and the inverse transform have the same complexity, allowing fast compression and decompression routines.
- Multi-resolution allows for progressive transmission and zooming, without the need for extra storage. One can e.g. first transmit a thumbnail image, and gradually transmit and decompress more data to increase the resolution and overall image quality.
- Wavelets can not only be used for image compression, but also for various image processing operations. The possibility to combine image processing and compression is a very appealing factor. However, image processing cannot be done on the final encoded data stream it must be done before the wavelet coefficients are quantized or encoded. But even then we win because the wavelet transform is a common factor in both image processing and image compression.

3.8. Lifting scheme of DWT

There are various architectures for implementing a two channel filter bank. A filter bank basically consists of a low pass filter, a high pass filter, decimators or expanders and delay elements. In this chapter we consider direct form structure, polyphase structure briefly before we go into depth analysis of lifting structure.

3.8.1. Direct Form Structure

The direct form analysis filter consists of a set of low pass and high pass filters followed by decimators. The synthesis filter consists of up samplers followed by the low pass and high pass filters as shown in Fig.3.5.

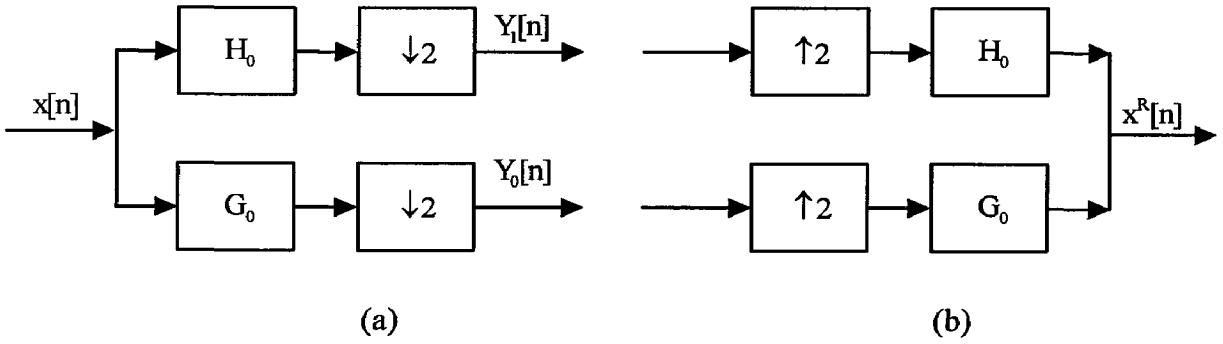


Fig.3.5. Direct form structure of **(a)** Analysis filter bank and **(b)** Synthesis filter`

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

$$Y_0[k] = \sum_n X[n].G_0[2k-n] \tag{3.1}$$

$$Y_1[k] = \sum_n X[n].H_0[2k-n] \tag{3.2}$$

The output of the analysis filter is usually processed (compressed, coded or analyzed) based on the application. This output can be recovered again using the synthesis filter bank. In the synthesis filter bank, Y_0 and Y_1 are first upsampled by 2 and then filtered to give the original input.

3.8.2. Polyphase Structure

In the direct form analysis filter bank, it is seen that if the filter output consists of say, N samples, due to decimation by 2 we are using only $N/2$ samples. Therefore,

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

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

$$\begin{bmatrix} G_{0even} & G_{0odd} \\ H_{0even} & H_{0odd} \end{bmatrix} \begin{bmatrix} X_{even} \\ Z^{-1}.X_{odd} \end{bmatrix} = H_p \begin{bmatrix} X_{even} \\ Z^{-1}.X_{odd} \end{bmatrix} = \begin{bmatrix} Y_0 \\ Y_1 \end{bmatrix} \tag{3.3}$$

The filters with G_{0even} and G_{0odd} are half as long as G_0, since they are obtained by splitting G_0. Since, the even and odd terms are filtered separately, by the even and odd coefficients of the filters, the filters can operate in parallel improving the efficiency. The Fig.3.6. illustrates polyphase analysis and synthesis filter banks.

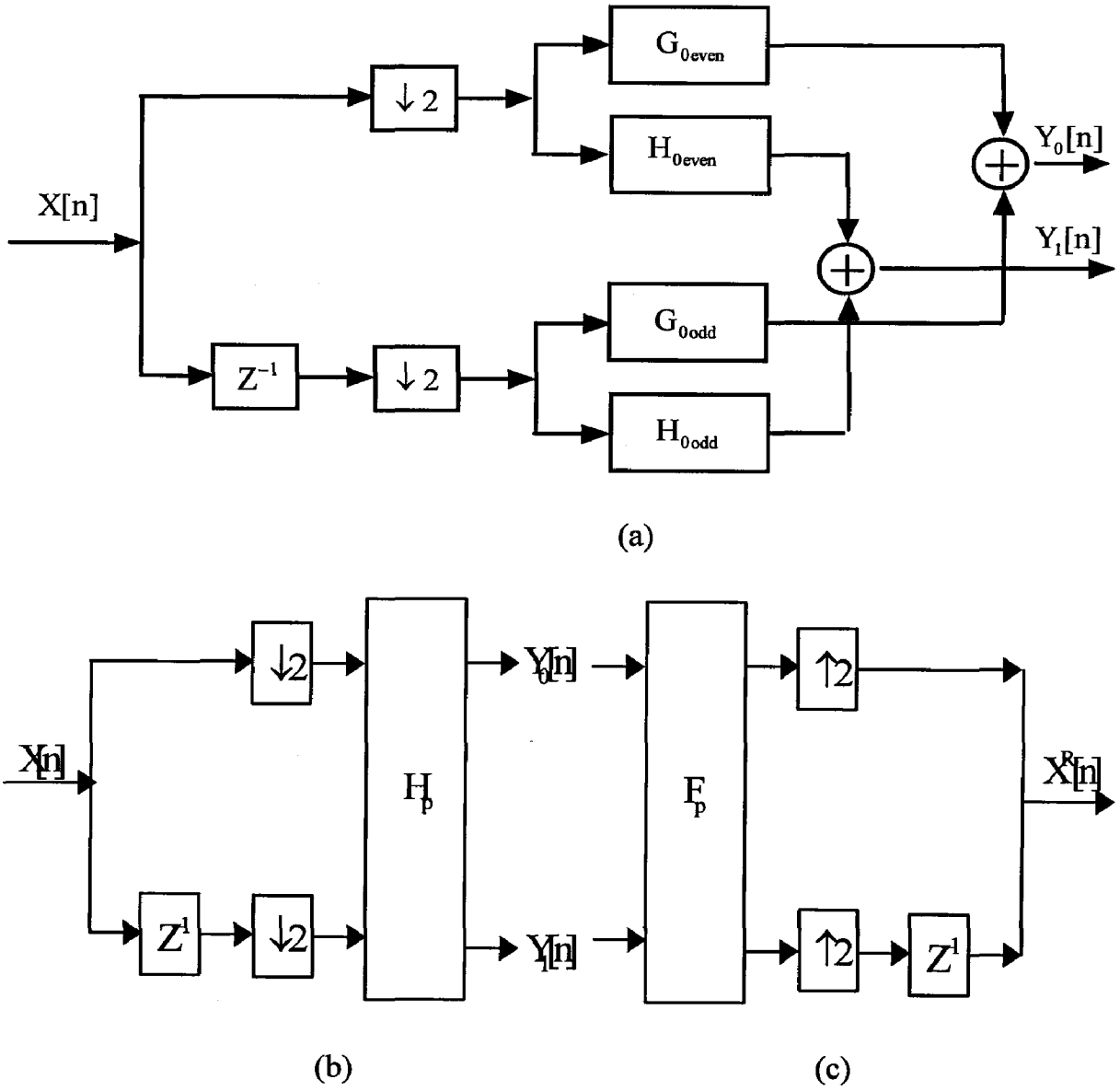


Fig.3.6. Polyphase structure of **(a)** Analysis filter bank **(b)** Equivalent representation of analysis filter bank and **(c)** Synthesis filter bank

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

3.8.3. Lifting Scheme Architecture

In 1994, Sweldens proposed a more efficient way of constructing the biorthogonal wavelet bases, which he called the lifting scheme [19]. The basic structure of the lifting scheme is shown in Fig.3.7. The input signal $s_{j,k}$ is first split into an update function to even and odd samples. The detail (i.e., high-frequency) coefficients $d_{j-1,k}$ of the signal are then generated by subtracting the output of a prediction function P of the odd samples from the even samples. The smooth coefficients (the low frequency components) are produced by adding the odd samples to the output of an update function U of the details. The computation of either the detail or smooth coefficients is called a lifting step.

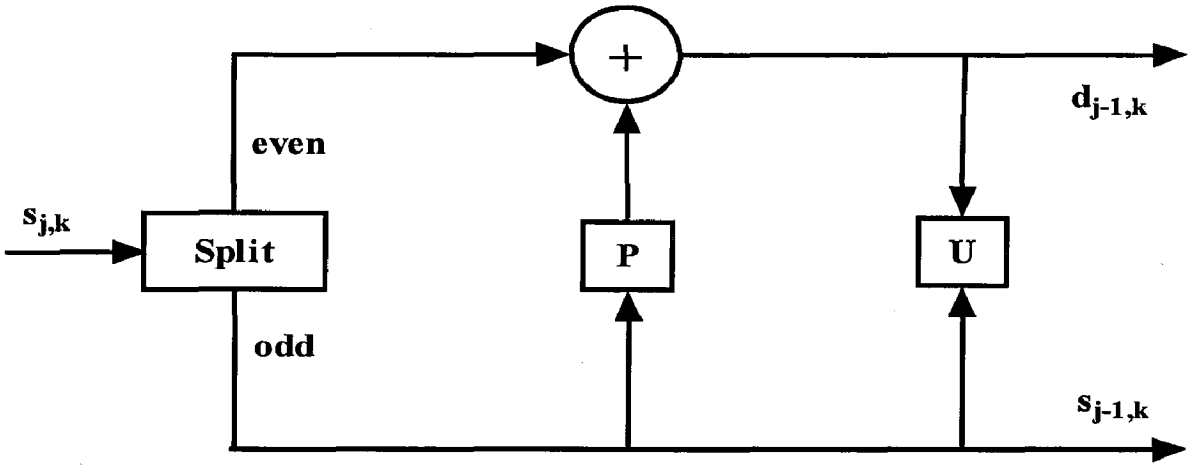


Fig.3.7. Lifting scheme

3.8.3.1. Factoring Wavelet Filters Into Lifting Scheme

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

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

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

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

$$g(z) = g_e(z^2) + z^{-1} g_0(z^2) \tag{3.6}$$

$$h(z) = h_e(z^2) + z^{-1} h_0(z^2) \tag{3.7}$$

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

$$P(z) = \begin{bmatrix} h_e(z) & g_e(z) \\ h_0(z) & g_0(z) \end{bmatrix} \tag{3.8}$$

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

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

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

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

3.8.3.2. Chohen-Daubechies-Feauvea (CDF) (2,2) Wavelet Using Lifting Scheme

The analyzing filter pair for the CDF [15] with 2 vanishing moments for both primal lifting and dual wavelet function is (up to a normalization factor of $\sqrt{2}$)

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

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

Following the above procedure from section 3.8.2 we can factor the analysis polyphase matrix of a CDF(2,2) wavelet

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

The lifting structure for the CDF (2,2) is shown in Fig.3.8.

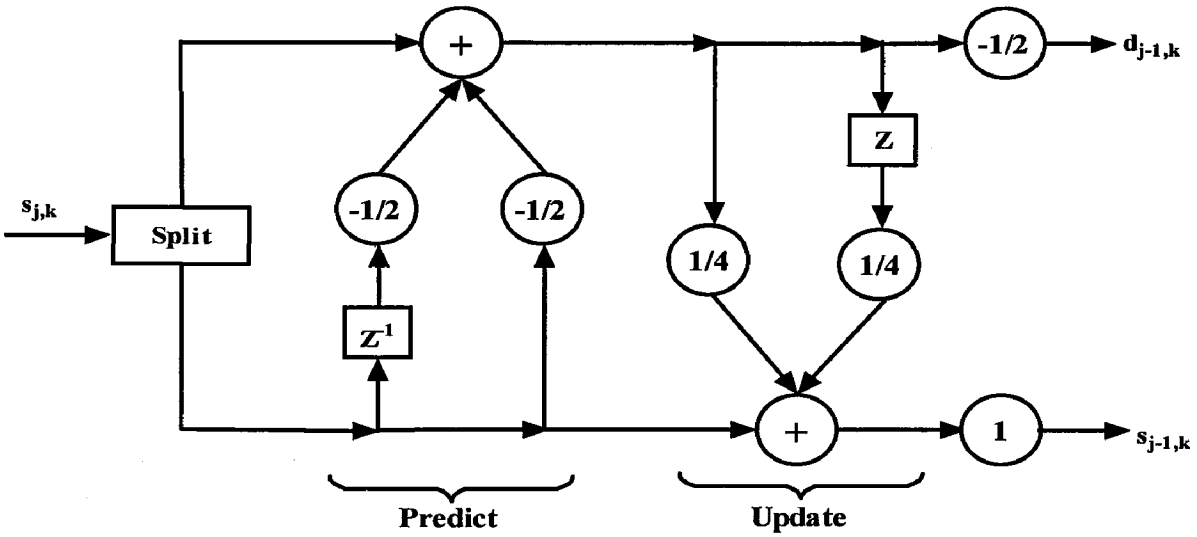


Fig.3.8. Lifting structure for CDF (2,2) wavelet

3.8.3.3. Integer-To-Integer Transform

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

Fortunately the Lifting Scheme can be easily modified to map integers to integers, which are in addition fully reversible [3], [20] and thus allows a perfect reconstruction of the original image.

3.8.4. Advantages of Lifting scheme

Lifting scheme has the following advantages, when compared to other classical filter bank algorithm:

- Lifting leads to a speedup when compared to the classic implementation. Classical wavelet transform has a complexity of order n, where n is the number of samples. For long filters, Lifting Scheme speeds up the transform with another factor of two. Hence it is also referred to as Fast Lifting Wavelet Transform (FLWT).
- All operations within lifting scheme can be done entirely parallel while the only sequential part is the order of lifting operations.
- At every summation point the old stream is replaced by the new one at every summation point.
- Lifting Scheme allows integer-to-integer transform while keeping a perfect reconstruction of the original data set. This is important for hardware implementation and lossless image coding.
- Lifting allows adaptive wavelet transforms. This means that the analysis of a function can start from the coarsest level, followed by finer levels by refining in the areas of interest.

Chapter 4

INTRODUCTION TO FPGA IMPLEMENTATION

For hardware implementation Spartan 3E (xc3s500e-4fg320) FPGA kit is used. Spartan 3E FPGA kit is most widely used FPGA kit because of its advantages like low power, low cost etc. In this thesis, The FPGA implementation was developed using the Xilinx System Generator for DSP_{TM} , a Simulink-based tool for FPGA design. Design flow based on System Generator tool derives a hardware realization directly from the system model via automatic code generation. This methodology is sometimes referred to as 'model-based design' [21] such high level design approaches aim to increased productivity (from higher levels of abstraction) and reliability (from automatic code generation and more robust test methodologies).

In this chapter we look at the overview of Spartan 3E architecture, FPGA design flow using Xilinx system generator tool.

4.1. Architectural Overview

The Spartan-3E family architecture [22] consists of five fundamental programmable functional elements:

CONFIGURABLE LOGIC BLOCKS (CLBS) contain flexible Look-Up Tables (LUTs) that implement logic plus storage elements used as flip-flops or latches. CLBs perform a wide variety of logical functions as well as store data.

INPUT/OUTPUT BLOCKS (IOBS) control the flow of data between the I/O pins and the internal logic of the device. Each IOB supports bidirectional data flow plus 3-state operation. Support a variety of signal standards, including four high-performance differential standards. Double Data-Rate (DDR) registers are included.

BLOCK RAM provides data storage in the form of 18-Kbit dual-port blocks.

MULTIPLIER BLOCKS accept two 18-bit binary numbers as inputs and calculate the product.

DIGITAL CLOCK MANAGER (DCM) BLOCKS provide self-calibrating, fully digital solutions for distributing, delaying, multiplying, dividing, and phase-shifting clock signals.

These elements of Spartan 3E family architecture are shown in Fig.4.1. A ring of IOBs surrounds a regular array of CLBs. Each device has two columns of block RAM

except for the XA3S100E, which has one column. Each RAM column consists of several 18-Kbit RAM blocks. Each block RAM is associated with a dedicated multiplier. The DCMs are positioned in the center with two at the top and two at the bottom of the device.

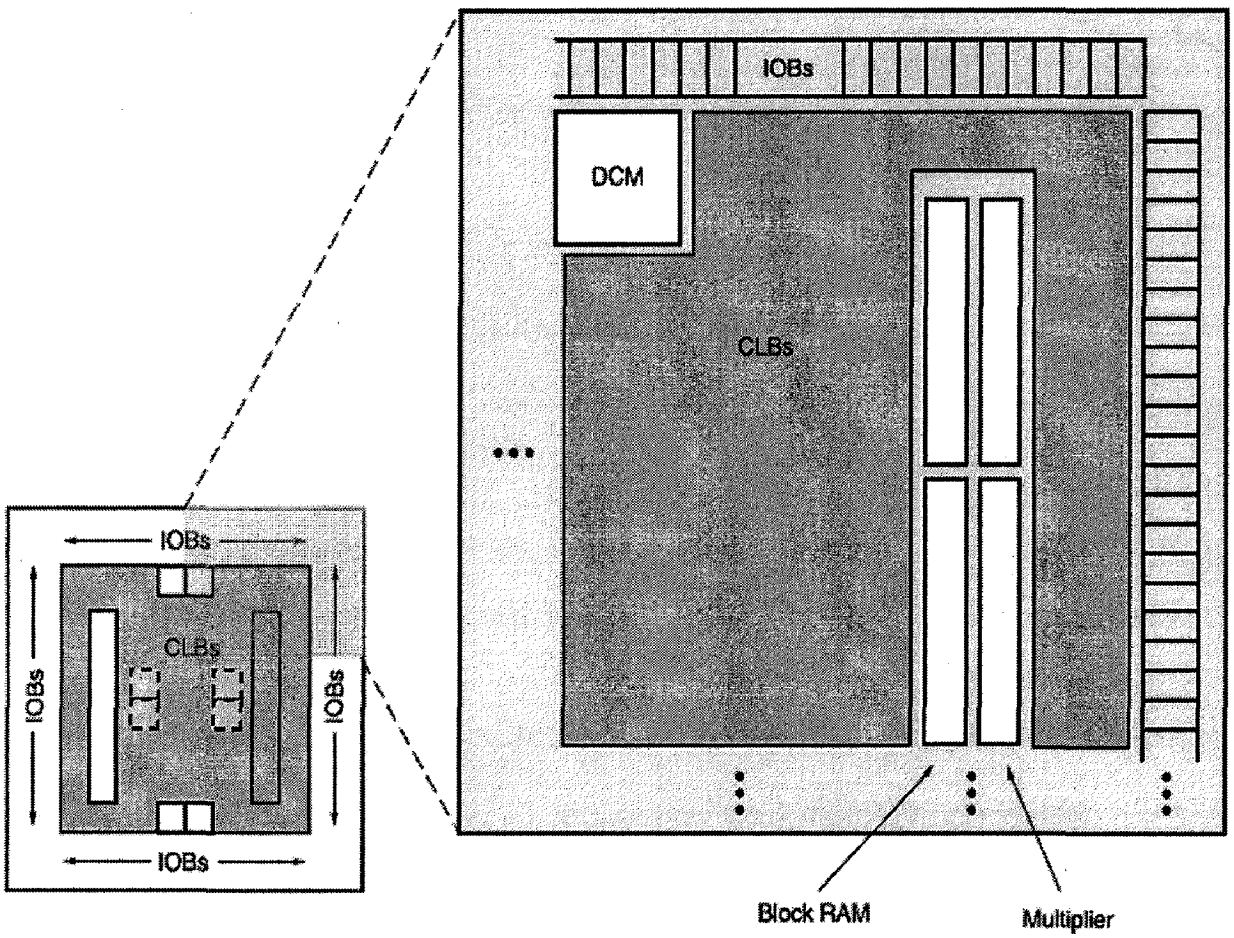


Fig.4.1. Spartan 3E family architecture

4.2. FPGA Design Flow using Xilinx System Generator

In order for an FPGA to carry out a set of instructions, it must be first programmed by loading a design via a bitstream file into the FPGA internal configuration memory. The bitstream file contains all of the configuration information from the physical design defining the internal logic and interconnections of the FPGA, as well as device-specific information from other files associated with the target device.

Simulink/Matlab has been created as a system-level design tool that enabled us to exploit visual data flows. Simulink also offers a path to automatically generate an FPGA bit stream (program the FPGA) on the target development board through third-

party tools such as Xilinx System Generator [23]. Therefore, Xilinx System Generator software has been chosen to exploit a fast route to programming FPGAs.

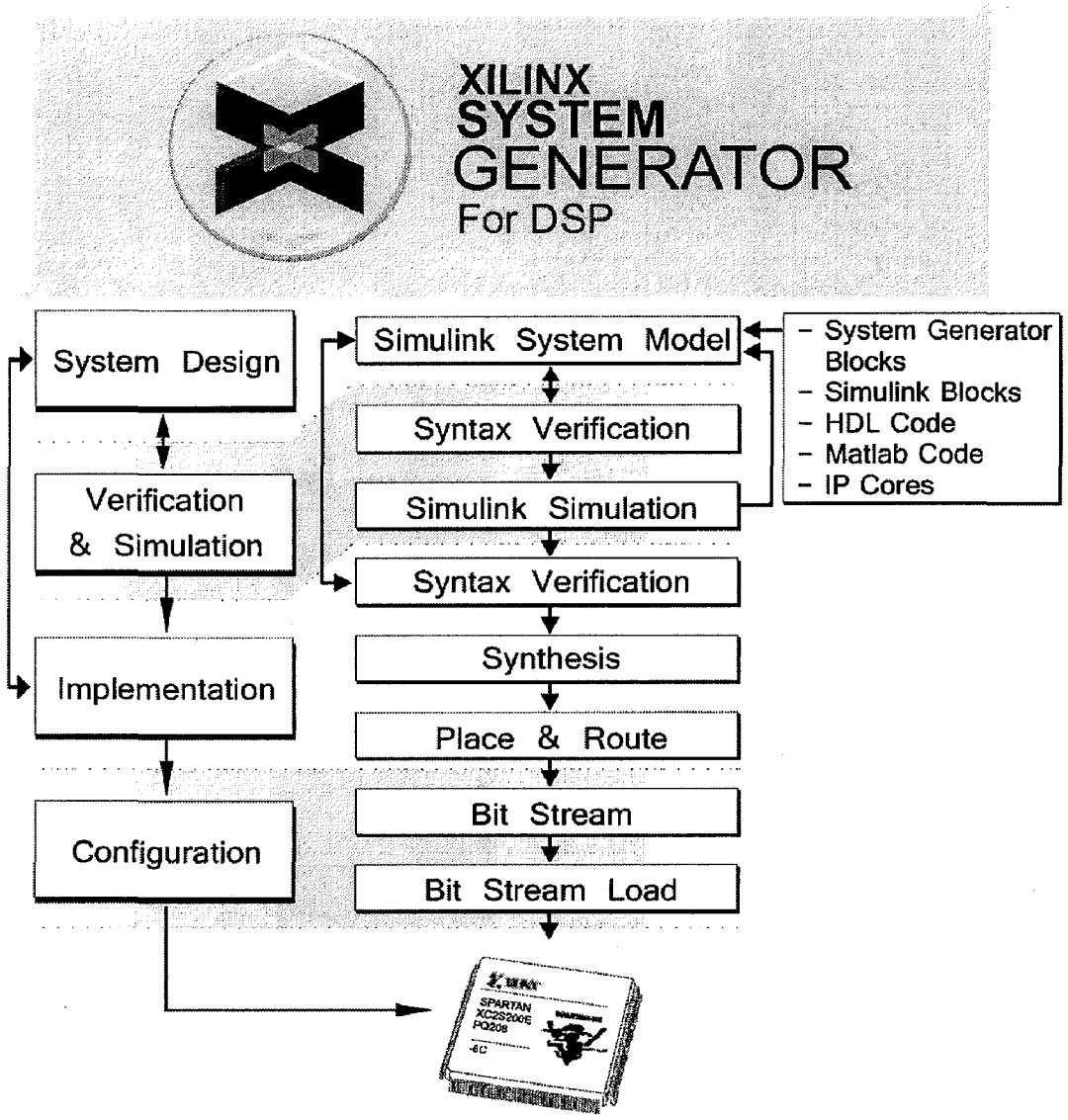


Fig.4.2. FPGA based platform design flow

Figure.4.2. Shows the FPGA design flow methodology used in this work. It consists on a basic four-step methodology for implementing FPGA applications: 1) system design, 2) verification and simulation, 3) implementation and 4) configuration.

1. SYSTEM DESIGN. This first step describes the model design that has to be implemented onto the FPGA device. In our case, we create the designs using System Generator which is a visual programming environment [23].

2. VERIFICATION AND SIMULATION. Functional verification and simulation checks the logical correctness of FPGA design. Once any form of design entry has

represented a design, it is necessary to verify if such a description satisfies the design specifications.

3. IMPLEMENTATION. This step includes the synthesis and place & route processes. Synthesis converts design entry into actual gates/blocks specified in FPGA devices. This is a very important step of the whole design procedure. A lot of efforts have been put into the improvement of the synthesis algorithms. On the other hand, the place & route tool selects the optimal position for elementary design blocks and minimizes length of interconnections on FPGA devices.

4. CONFIGURATION. This final step implies downloading bit stream codes onto the FPGA device. The Xilinx system development provides with the Project Navigator software which includes the iMPACT tool. iMPACT works as the programming interface between a PC (host) and the FPGA.

Spartan-3E Starter Kit shown below in Fig. 4.3 is for hardware implementation

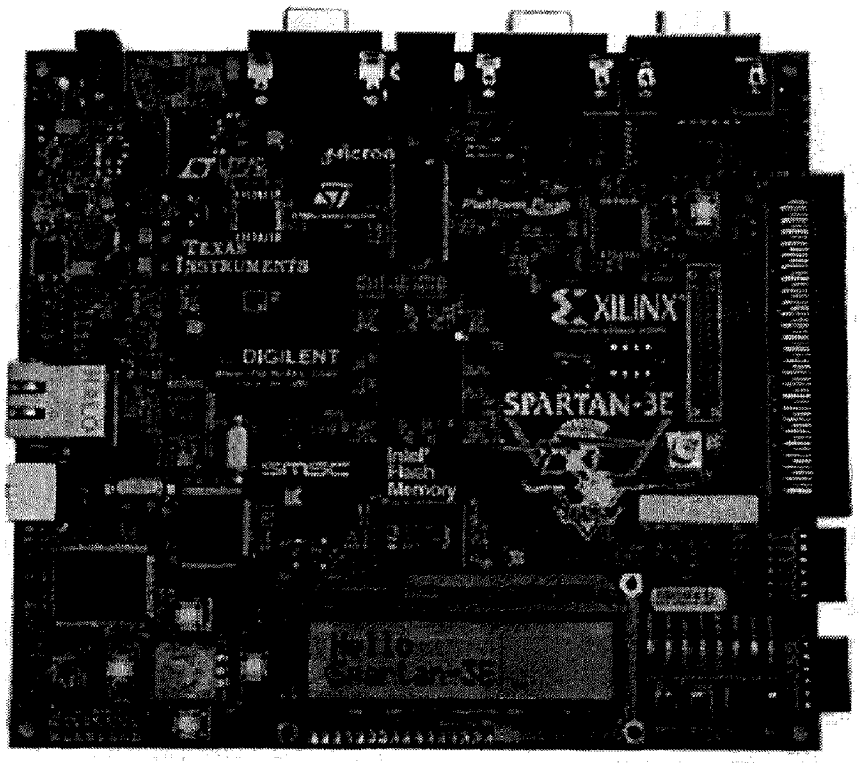


Fig.4.3. Spartan-3E Starter Kit [27]

4.3. FPGA Configuration

This section describes the process of loading the configuration bitstream (.bit file) into the FPGA. To download the .bit file we will be using iMPACT (part of the Xilinx tools Installation) [24]. The following five steps outline this process.

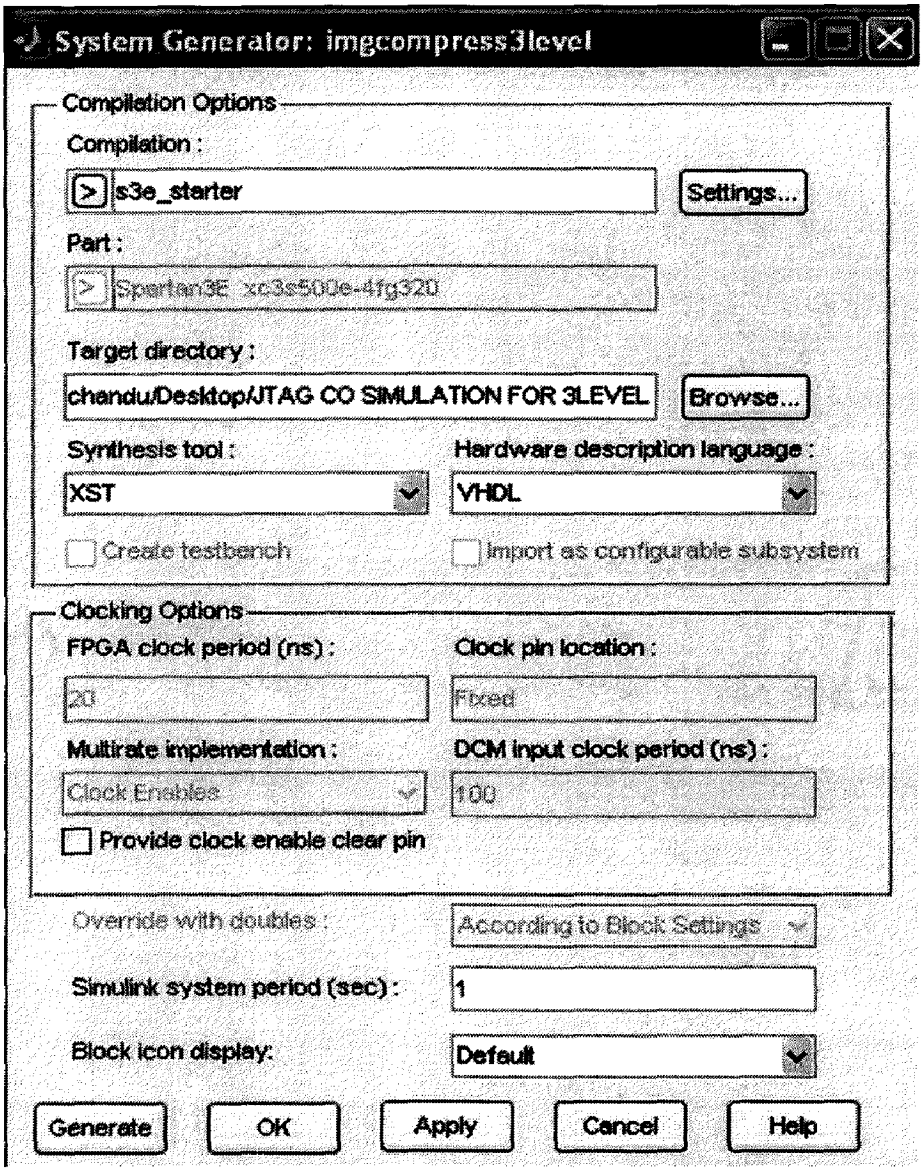


Fig.4.4. System genarator token

1. When iMPACT starts up, we select the default choice Configure Devices in the first dialog. Next, select Boundary-Scan Mode, and the automatically connect to cable and identify Boundary-Scan chain.

2. iMPACT will then connect to the JTAG adapter cable, and search the JTAG chain for devices. If this fails, there is a communication problem with the parallel port on the PC, a problem with the JTAG cable, or a problem with the JTAG scan chain on

the board. It will not be possible to configure the board until these problems are resolved.

3. Once iMPACT has identified all devices on the JTAG scan chain, we have to assign the programming file to each of the devices in the chain, in turn.

4. In the Assign New Configuration File dialog box for each device, we select the configuration file for the type of device and click on Open, or click on Bypass to put that device into bypass mode, causing it not to be configured. When prompted to select a programming file for the FPGA, we select the configuration bit file for the Spartan-3E (choosing the .bit file from the project directory) device.

5. After configuration files have been assigned to all devices, a device is configured by right clicking on its icon in the iMPACT window and choosing Configure from the menu. Appropriate options for the device being programmed must be selected from the Program Options dialog box. iMPACT will configure the device and then indicate whether configuration was successful or not. If HDL code has been created, the Xilinx Integrated Software Environment (ISE) can be used to convert a collection Verilog or VHDL files into a configuration bitstream (.bit file) for the FPGA application.

Chapter 5

HARDWARE IMPLEMENTATION OF CDF LIFTING SCHEME ARCHITECTURE

An improved parallel architecture for implementing 2D-DWT of CDF (2,2) is proposed in this thesis. First the proposed architecture is designed using System generator Simulink, then implemented on FPGA.

In this chapter, algorithm for CDF (2,2) wavelet is briefly explained, then it is designed in System generator Simulink and Finally its hardware implementation on FPGA is done.

5.1. Cohen-Daubechies-Feauveau (CDF)(2,2) Wavelet

The Cohen-Daubechies-Feauveau (CDF) (2,2) Wavelet [29] is widely used for image compression because of its good compression characteristics. The original filters have 5 + 3 = 8 filter coefficients as shown in Eq.3.11 and Eq.3.12, whereas an implementation with the lifting scheme has only 2 + 2 = 4 filter coefficients. The forward and reverse filters are shown in Table 5.1(a) and 5.1(b). Fractional numbers are converted to integers at each stage. Though such an operation adds non-linearity to the transform, the transform is fully invertible as long as the rounding is deterministic. In Table 5.1 x represents the image pixel values and s stands for the summing or the low pass coefficients and d stands for the difference or the high pass coefficients.

Table 5.1. CDF (2,2) wavelet with lifting scheme **(a)** Forward transform **(b)** Inverse transform

(a)

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

(b)

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

In image compression, one row or column of an image is regarded as a signal. Fig.5.1 shows the row and column formation at level 1, level 2 and level 3 for a 512 X 512 pixel image.

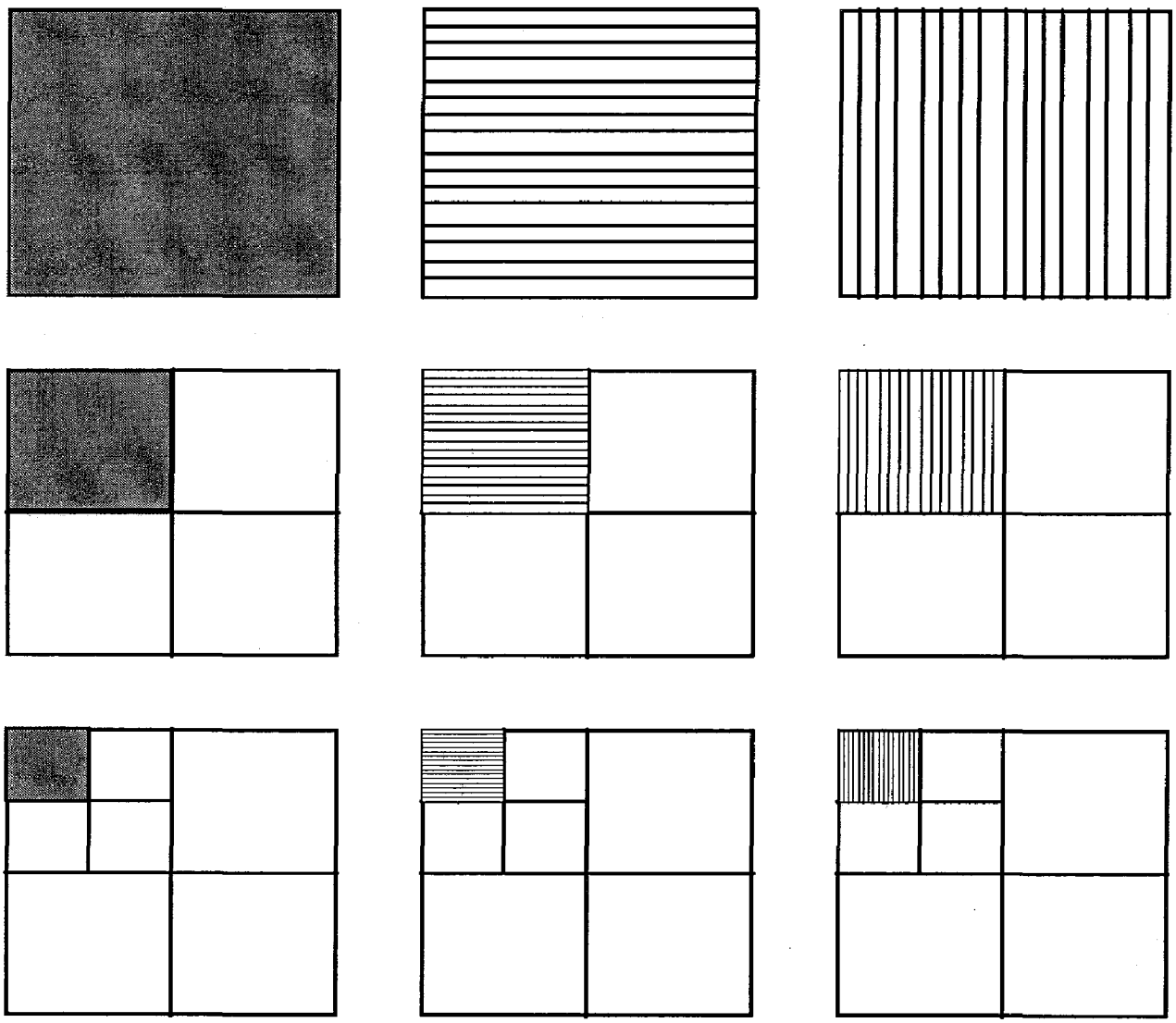


Fig.5.1. Rows and columns of level 1, 2 and 3 decomposition of an image

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

$S_0 D_0 S_1 D_1 S_2 D_3 S_3 D_4 S_4 .....$

In this algorithm, in case of rows it is for row processor whereas in case of column it is for column processor in the parallel architecture of lifting based CDF (2,2)[15] wavelet. The odd pixels should be processed first, then the even pixel due to the data dependency. There are a total of three levels based on the 3-level decomposition wavelet transform algorithm discussed above. In each level, the rows are processed first then the columns. Each level's signal length (amount of each row/column pixels) is half of the previous level.

Algorithm

For every row or column:

Repeat until end of row or column:

If begin or end of row or column

begin

$D_0 = D_0 + D_0 - S_0 - S_0$

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

End

Else

begin

$D_i = D_i + D_i - S_i - S_i$

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

5.2. CDF Wavelet Simulink model in system generator

The input image, which is in matrix format must be converted to a single column matrix, this is called Raster scanning. So, we use Matlab coding to obtain a Raster scan image. The input image is taken from Matlab of 512x512 grayscale image.

A prototype CDF biorthogonal Wavelet processor is modeled and implemented in Matlab Simulink using Xilinx System Generator Blockset. Xilinx system generator has a feature of hardware co-simulation, in which we can validate software and hardware implementation simultaneously, i.e. design developed in software using Xilinx blockset and design implemented on FPGA chip are validated simultaneously. Figure 5.2 shows the schematic diagram of Simulink model of CDF biorthogonal Wavelet transform.

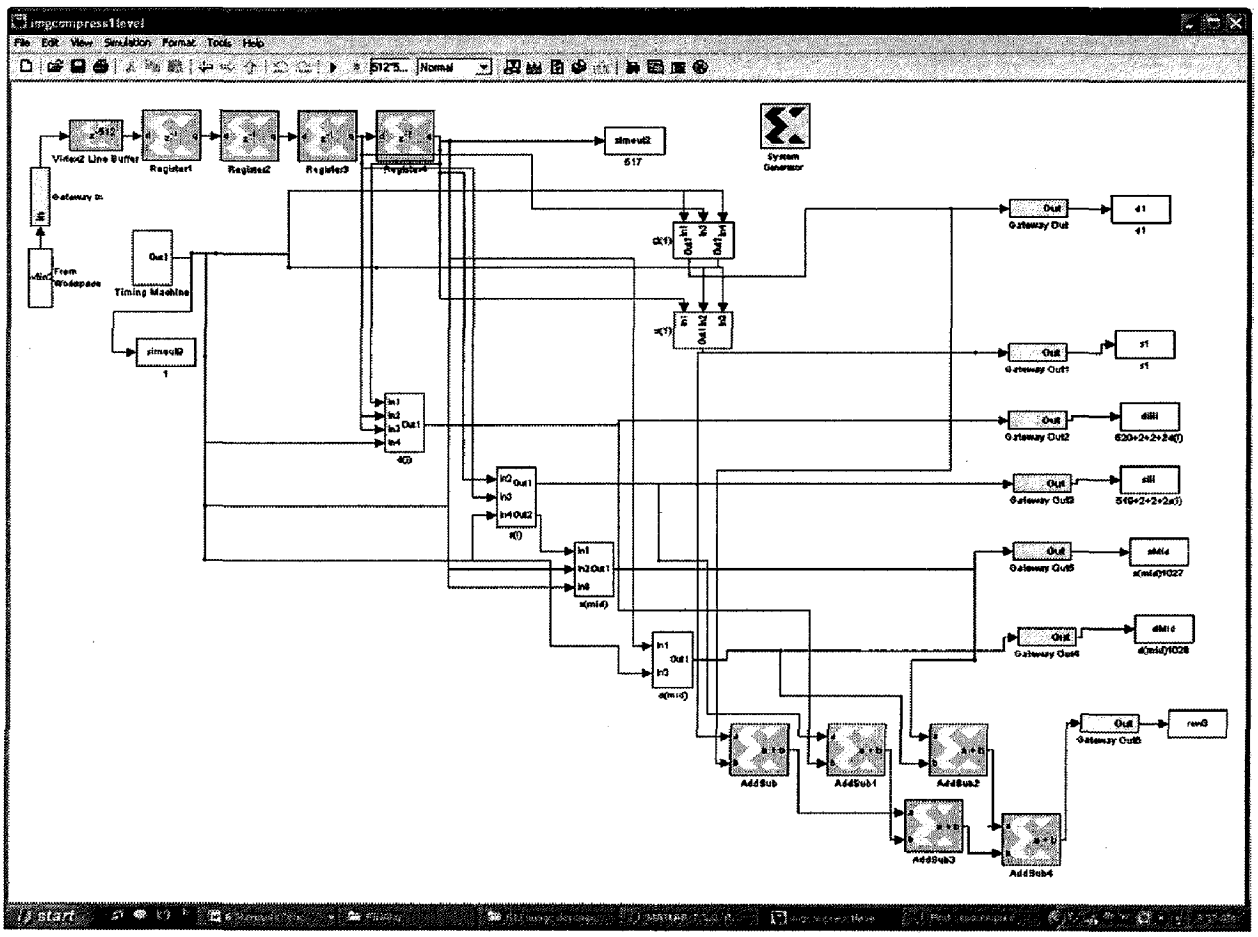


Fig.5.2. Simulink model of CDF Wavelet transform.

5.3. Implementation on FPGA

Having reached the desired performance throughout the verification and simulation processes, source design can be implemented. Basically, this is accomplished by translating the block model in Figure 5.2 into an HDL code or a .bit configuration file. This step is the combination of several process chained automatically. First, double-click is pressed on the System Generator token in the top level of the model to open the hardware generation GUI that lets you specify FPGA family and device, netlist type, Simulink clock rate, and whether a testbench is needed. Clicking Generate produces a cycle and bit accurate HDL netlist that can be synthesized and placed-and-routed using Xilinx ISE Foundation FPGA implementation software. Figure 5.3 illustrates the configuration options environment from the System Generator block. This block generates all necessary files for the FPGA application.

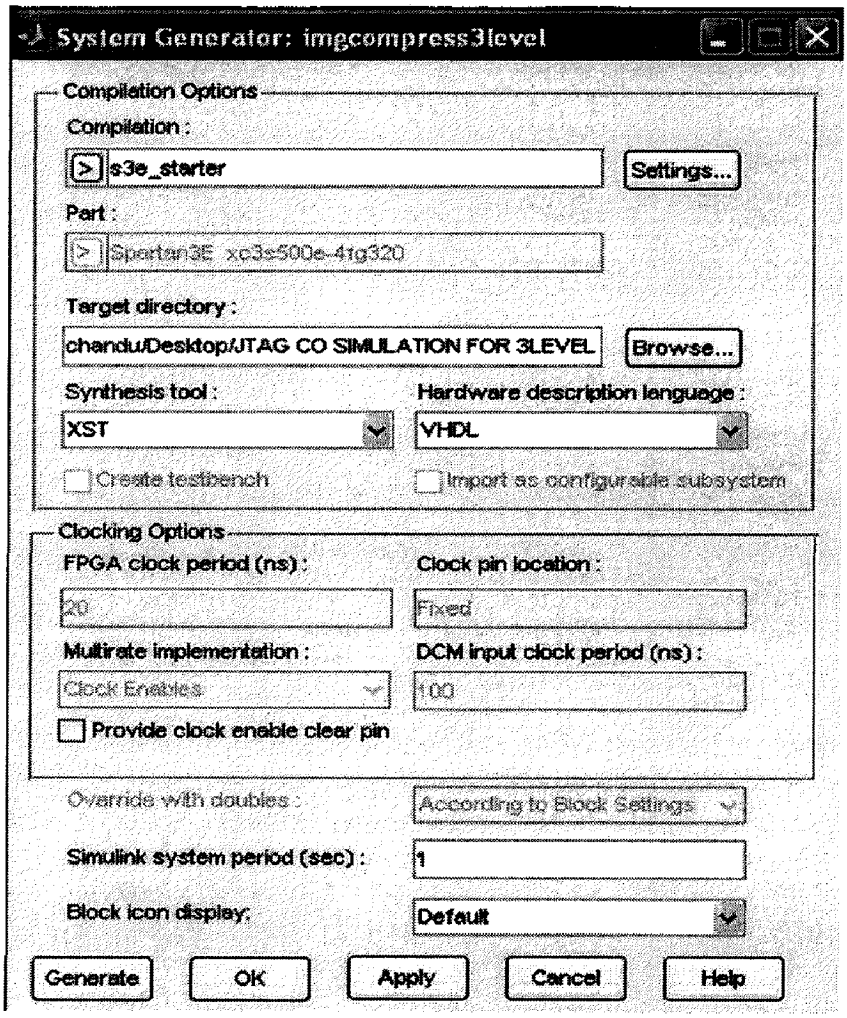


Fig.5.3. System generator token

After HDL conversion, it automatically generates the Hardware co-simulation block which is the FPGA equivalent Simulink model and having same number of inputs and outputs of previous simulink model. Figure 5.4 shows the Hardware co simulation block and also known as JTAG co-simulation block, then connect the inputs to JTAG co-simulation block and double click on this block, then invoke the cable option and set cable as speed as 12 MHz. Fig. 5.4 and Fig.5.5.shows both hardware and software model of CDF wavelet.

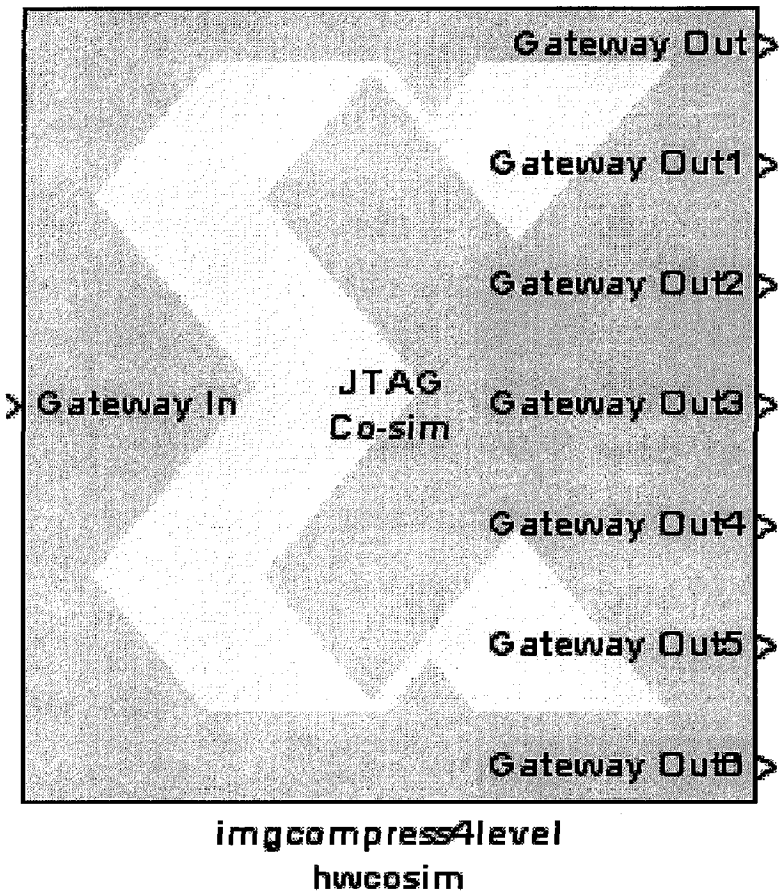


Fig.5.4. JTAG Co-simulation block

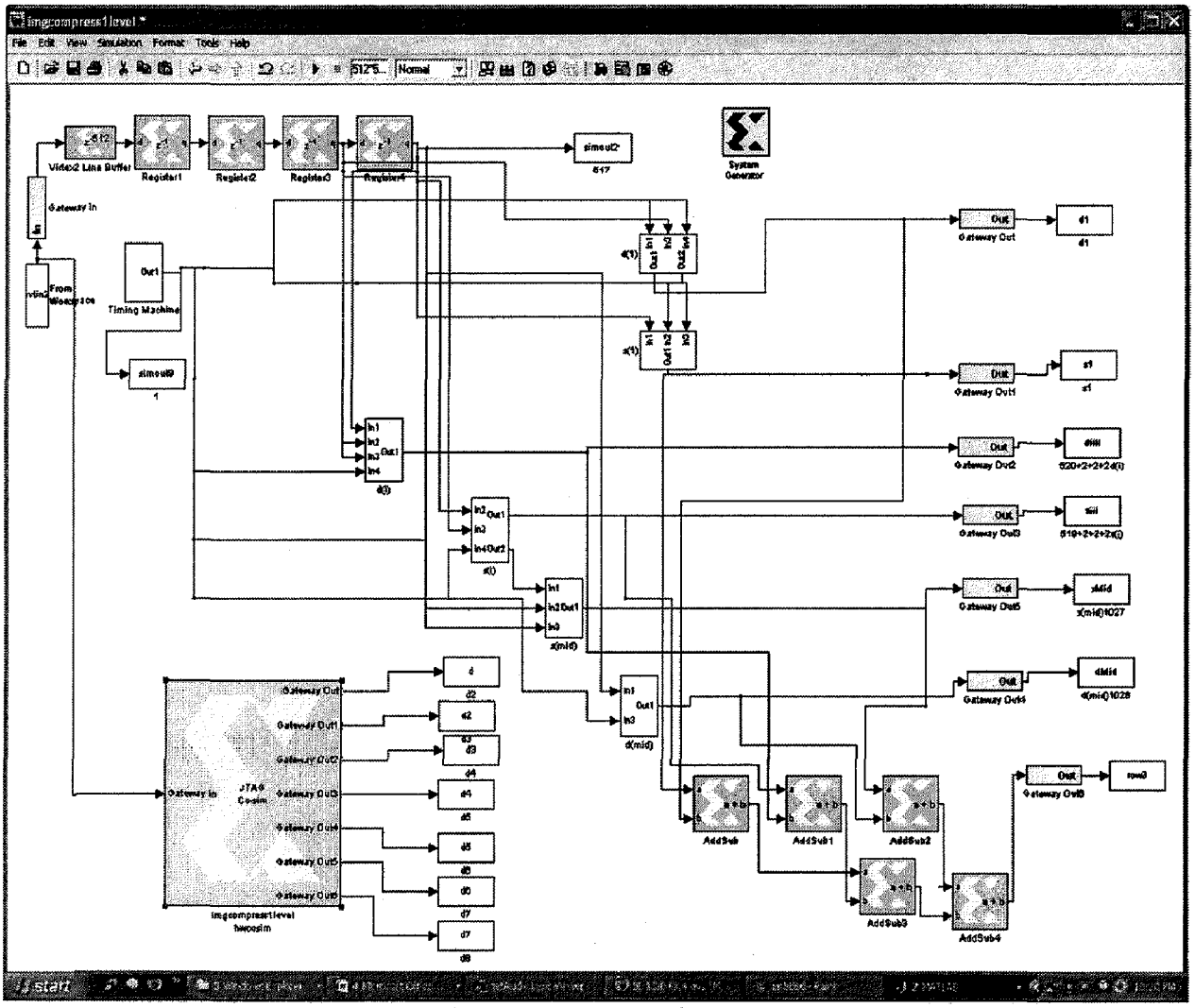


Fig.5.5. Simulink model with Hardware co-simulation block

Chapter 6

RESULTS AND DISCUSSION

In this chapter, the results of CDF biorthogonal Wavelet transform decomposition for different input images are presented. Then simulation and synthesis results of Prototype CDF biorthogonal Wavelet Processor modeled in Matlab Simulink using Xilinx System Generator Blockset are discussed. Hardware co-simulation is carried out successfully and simulation of both hardware and software are validated successfully. Device utilization summary, which indicates amount of logic occupied on the FPGA chip by our design, is discussed.

The input image, which is in matrix format must be converted to a single column matrix, this is called Raster scanning. So, we use Matlab coding to obtain a Raster scan image. These codes are given in Appendix-B. The input image is taken from Matlab of 512x512 grayscale image.

6.1. DWT in X and Y directions

Each pixel in the input frame is represented by 8 bits, accounting for 1 pixel per memory word. These 8 bits goes into the higher 8 bits (MSB) of the 16 bit memory word. Thus each memory read process brings in one consecutive pixels of a row. Thus two clock cycles generates one value each of **f** and **g** coefficients.

0 1	.....	254 255	256 257	.....	510 511

Pixel data

f0 f1	.....	f254 f255	g0 g1	.....	g254 g255

Coefficeint data

Fig.6.1. Coefficient Ordering in X (row) direction

These have to be written back in a Mallot ordering scheme i.e. to write back the coefficients in a way to put all the low frequency coefficients **f** ahead of the high frequency coefficients **g**. This is shown in the coefficient data in Fig. 6.1. It allows progressive image transmission/reconstruction. The bulk of the 'average' information

i.e. low frequency information is ahead followed by the minor 'difference' information i.e. high frequency information. Although an in-place write back scheme is available, it is not implemented because after the end of one row operation the coefficients would have to be ordered again to follow Mallot ordering. Once the filter has been applied along all the rows in a stage, the same filter is applied along the columns. The column coefficients are also written in the same aforementioned Mallot ordering.

The current hardware implementation processes the 512 x 512 IITR.TIF input image frame with 3 level of wavelet transform. Fig 6.2. shows the input image of IITR.TIF In this, for 1st level decomposition 512 pixels of each row are used to compute 256 high pass coefficient **g** and 256 low pass coefficients **f**, as shown in figure 6.1. The coefficients are written back in a rearranged manner such that the low frequency ones are ahead of the high frequency ones then the resulting image is shown in fig.6.3. Once all the 512 rows are processed, the filters are applied in the Y direction. The resulting decomposition after 1st level is shown in fig.6.4.

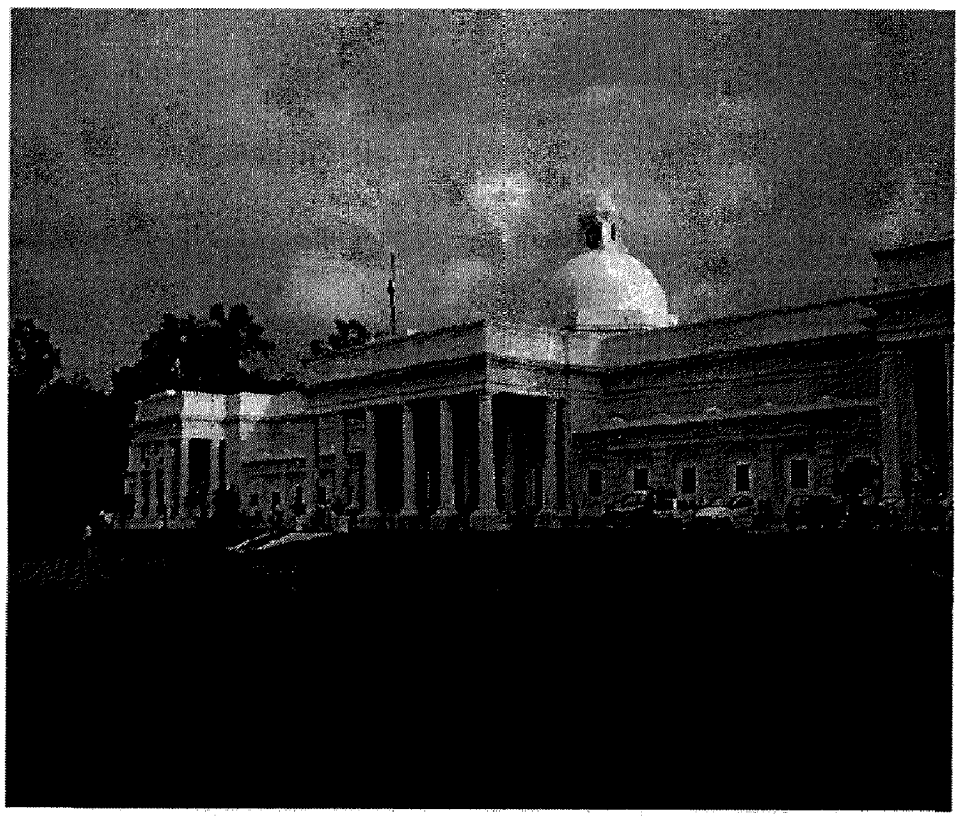


Fig.6.2. Original image IITR.TIF of size 512x512

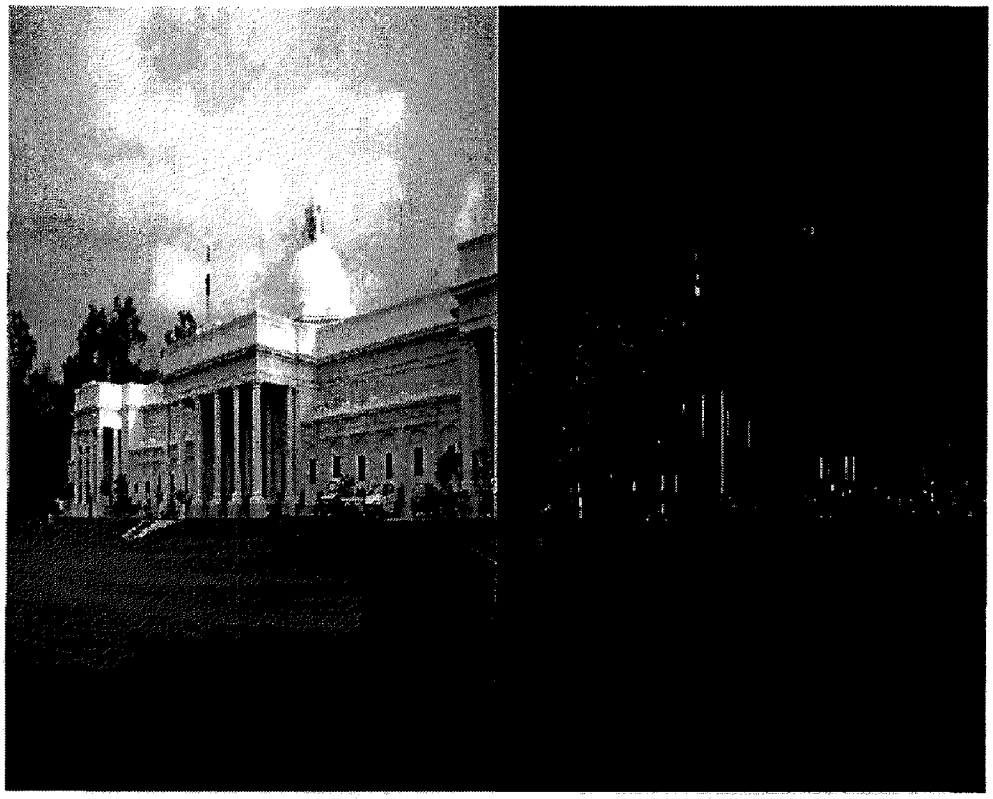


Fig.6.3. After level 1 in X direction

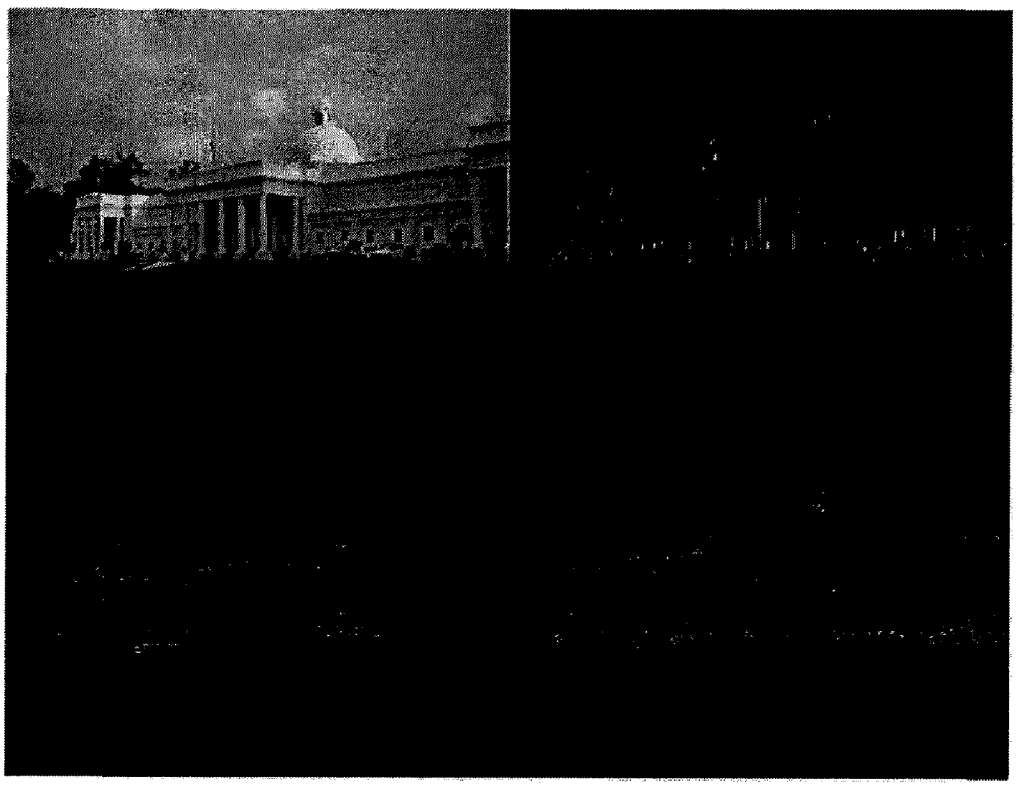


Fig.6.4. After level1 in Y direction i.e after first level of decomposition

6.2. Implementation using Xilinx System Generator

In this section the results of CDF biorthogonal Wavelet implementation using the Prototype CDF Wavelet processor are discussed. A prototype CDF Wavelet processor is modeled in Matlab Simulink using Xilinx System Generator Blockset. Xilinx system generator has a feature of hardware co-simulation, in which we can validate software and hardware implementation simultaneously, i.e. design developed in software using Xilinx blockset and design implemented on FPGA chip are validated simultaneously.

The prototype CDF Wavelet processor is validated for software and hardware implementation and results for different images inputs are discussed. The Figures indicate (6.2) input image, (6.4) 1st level decomposed image (6.5) 2nd level decomposed image (6.6) 3rd level decomposed image using system generator simulation and For simplicity and ease of implementation the input images are resized to 512x512.

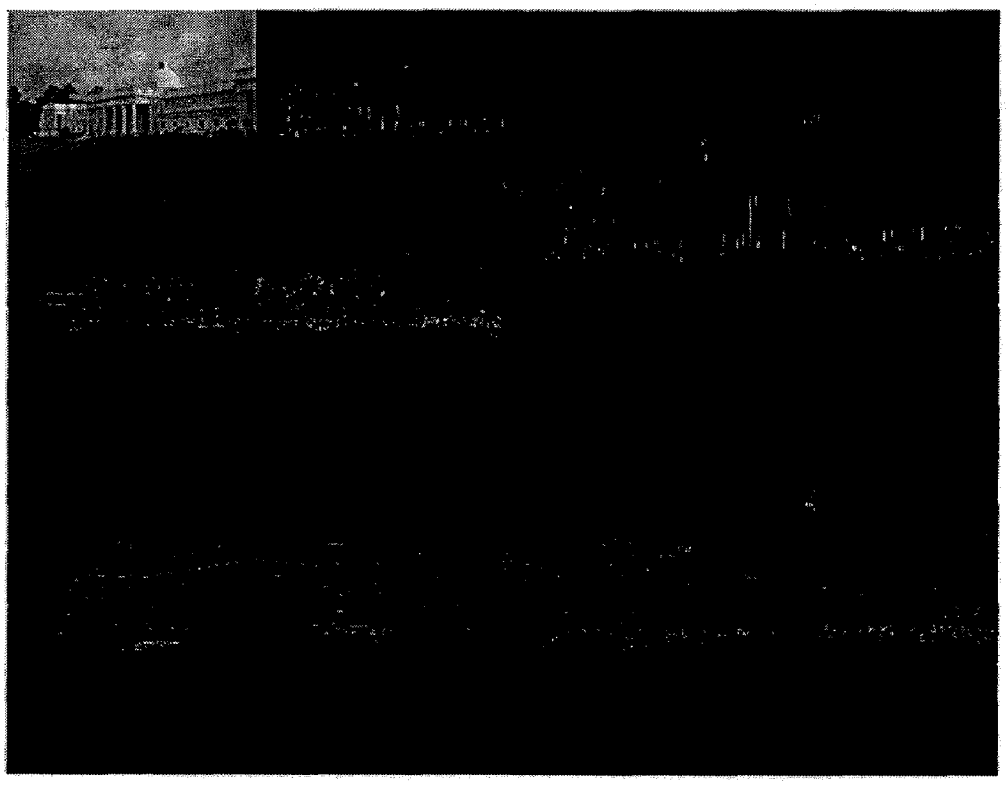


Fig.6.5. 2nd level decomposition of IITR.TIF Image

Fig.6.6. 3rd level decomposition of IITR.TIF Image

CDF Wavelet Processor Simulink model for the generation of JTAG hardware co-simulation block is shown in Fig.6.8. And the generated JTAG hardware co-simulation block is shown in Fig. 6.7.

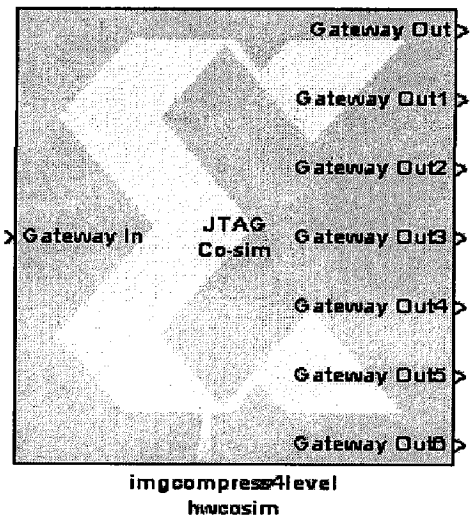


Fig.6.7. JTAG Co-simulation block

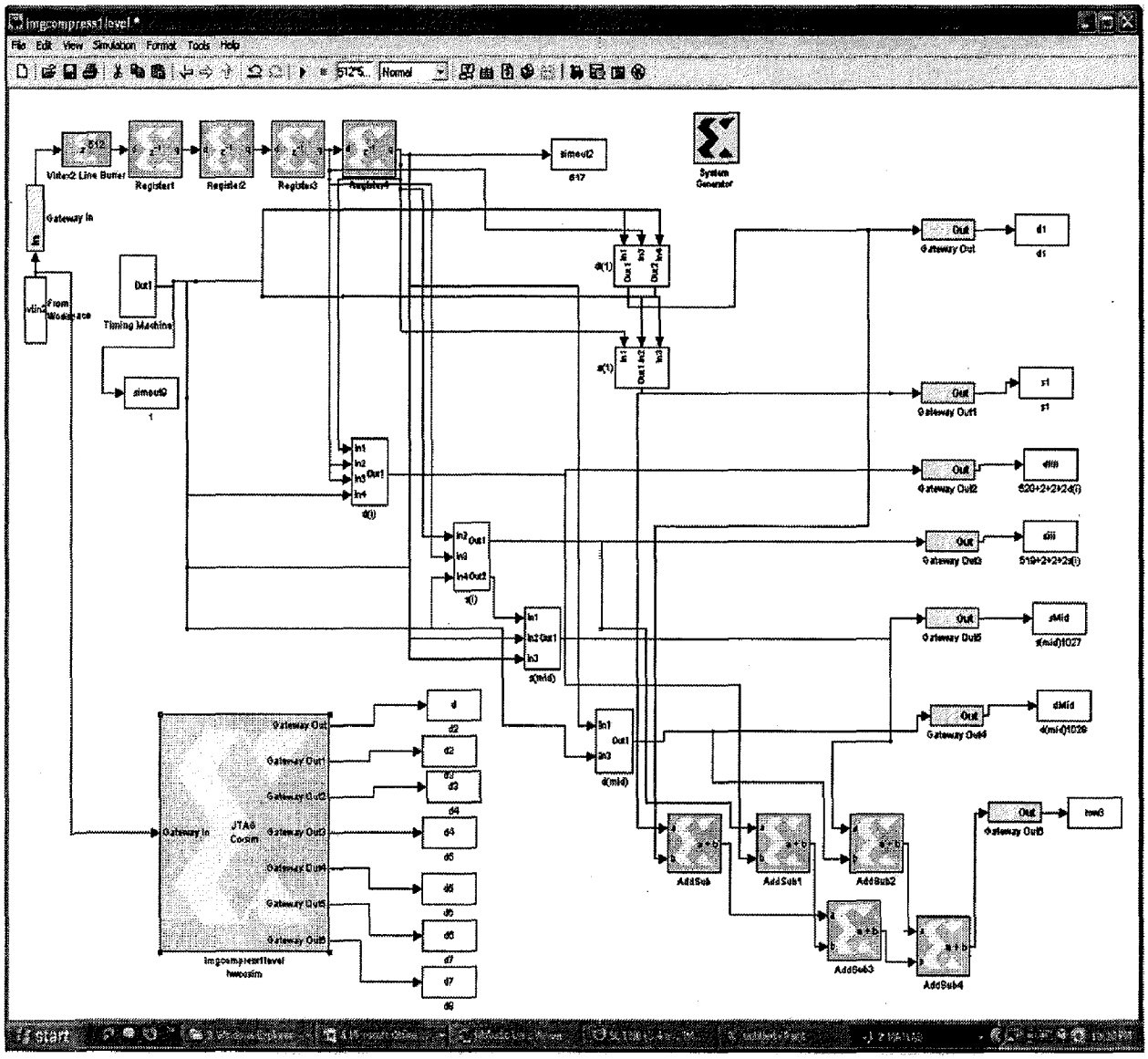


Fig.6.8. Simulink model of CDF wavelet processor with Hardware JTAG co-simulation block

3-level decomposition of different input.tif images are shown below.

Fig.6.9. Original image leena.tif of size 512x512

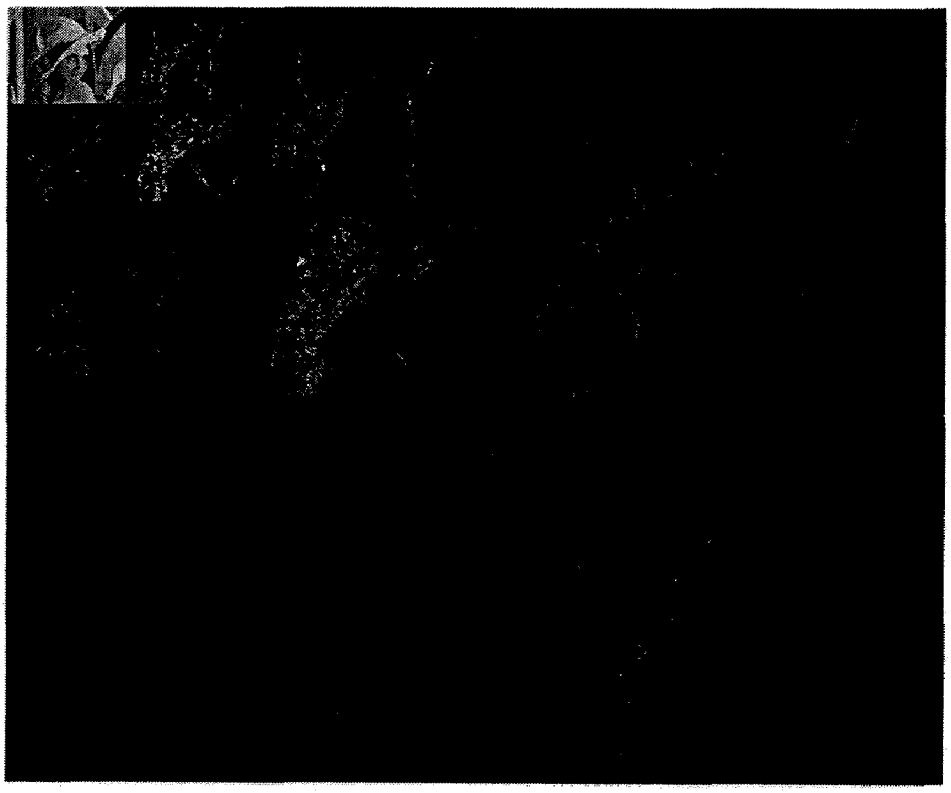


Fig.6.10. 3-level decomposed image of image leena.tif

Fig.6.11. Original image cameraman.tif of size 512x512

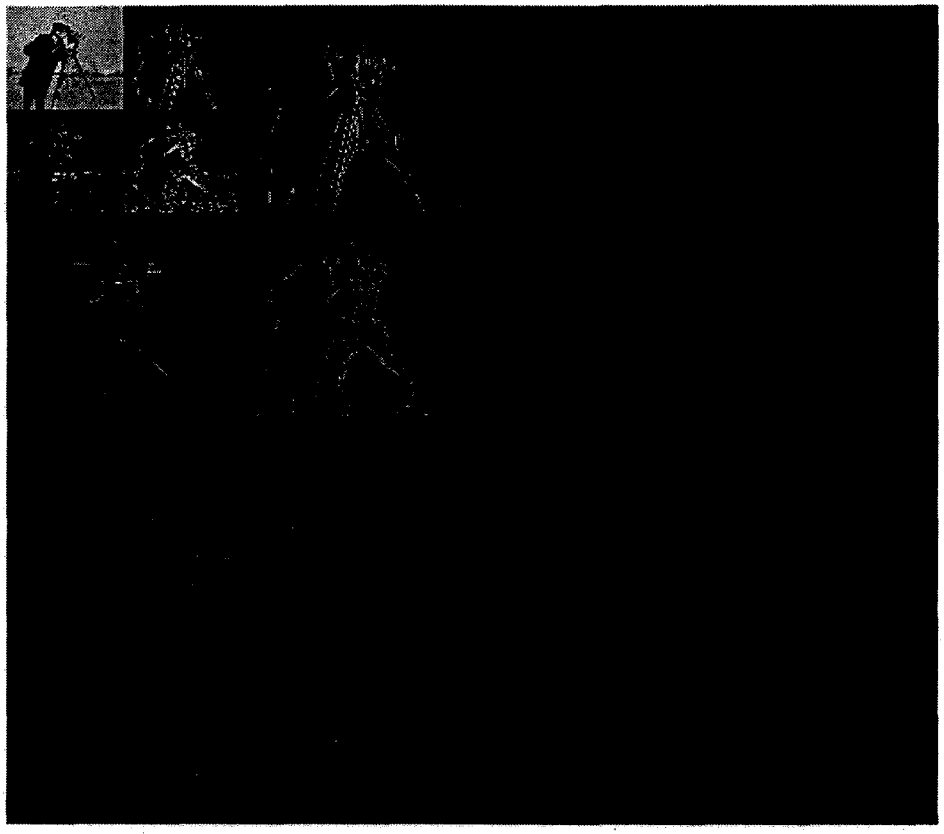


Fig.6.12. 3-level decomposed image of image cameraman.tif

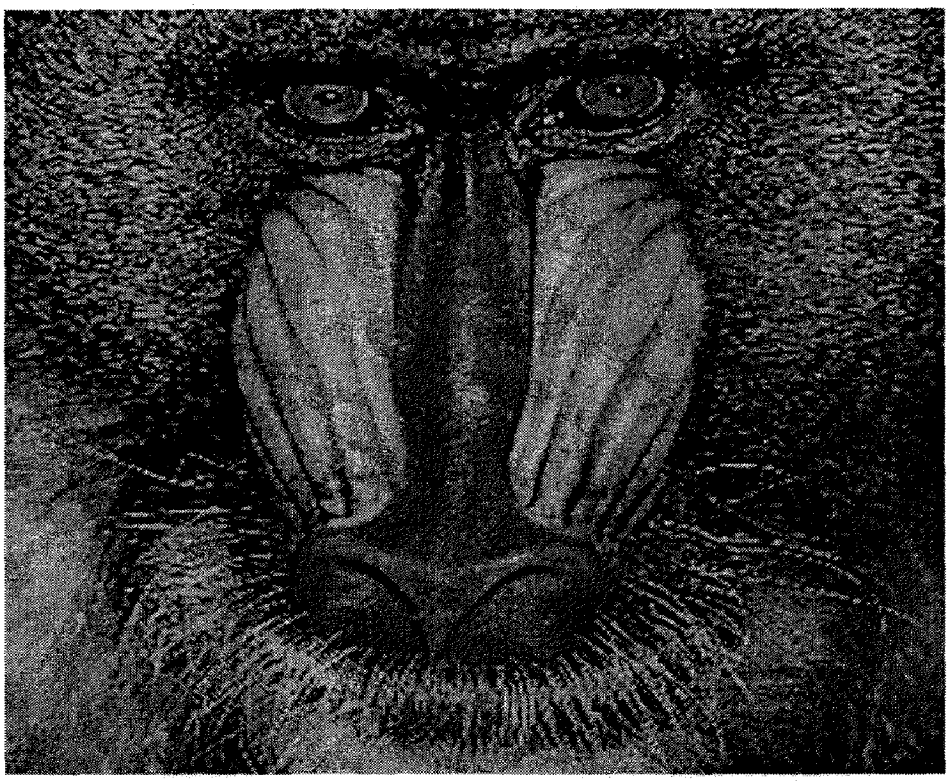


Fig.6.13. Original image mandrilla.tif of size 512x512

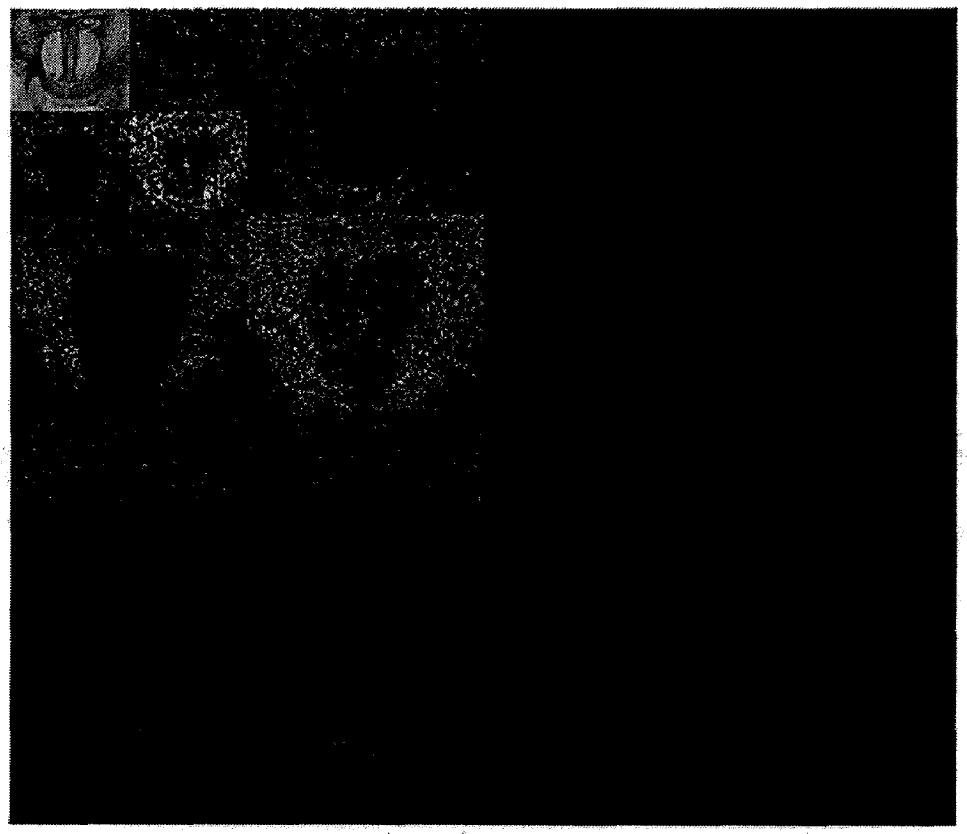


Fig.6.14. 3-level decomposed image of image mandrilla.tif

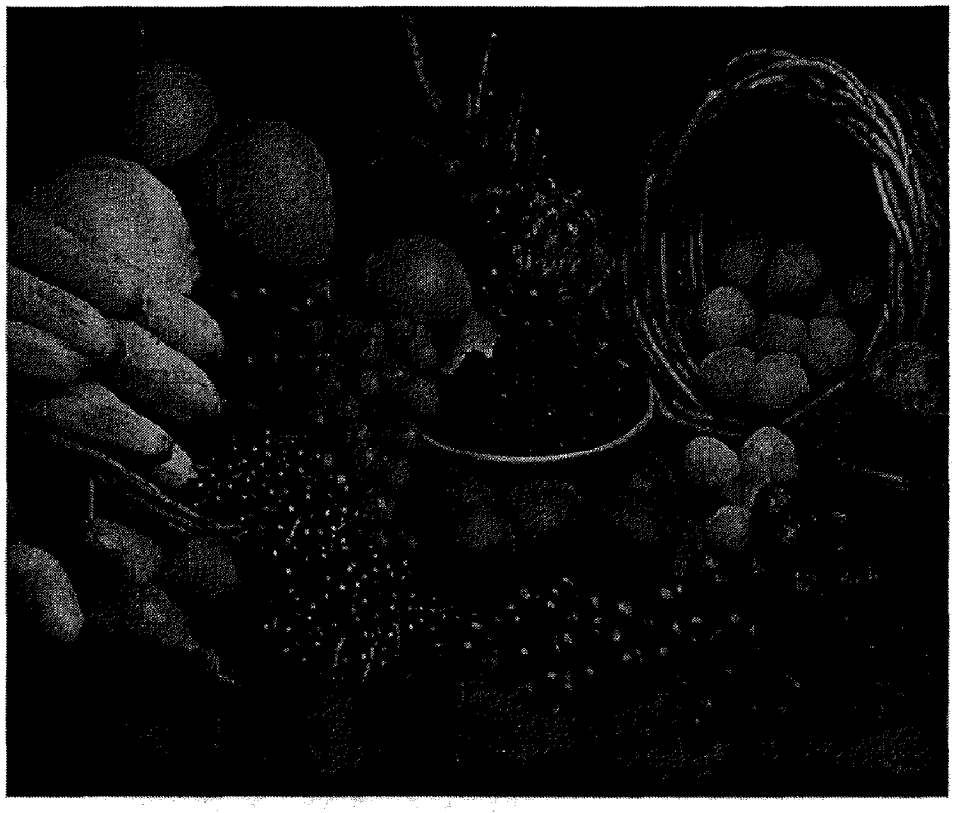


Fig.6.15. Original image fruits.tif of size 512x512

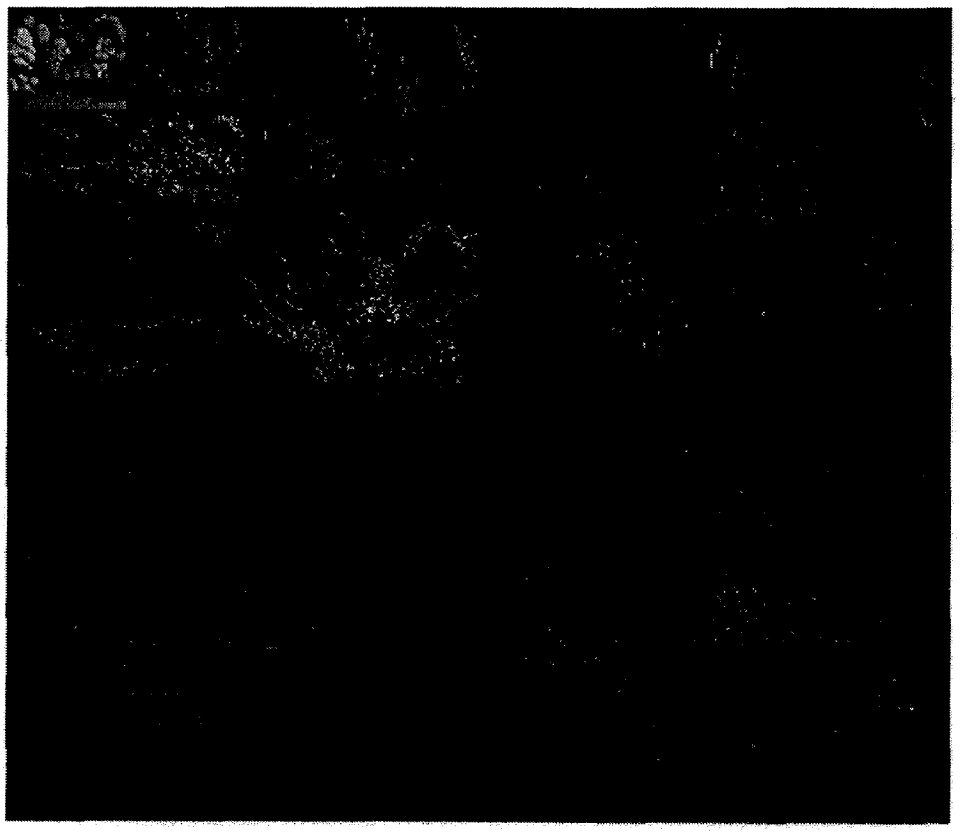


Fig.6.16. 3-level decomposed image of image fruits.tif

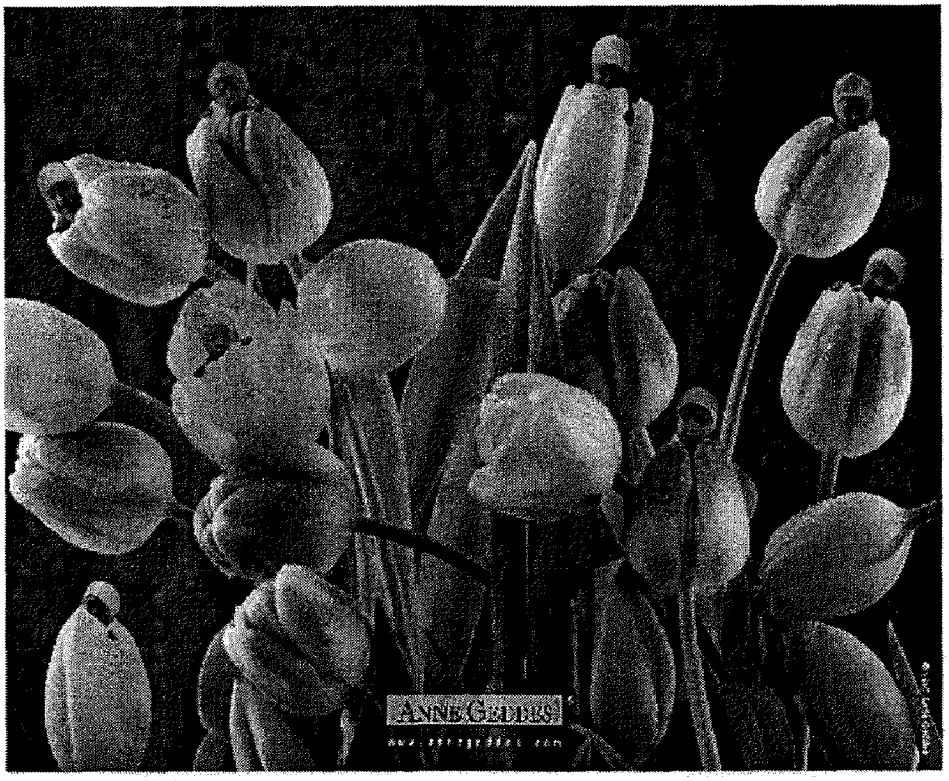


Fig.6.17. Original image flowers.tif of size 512x512

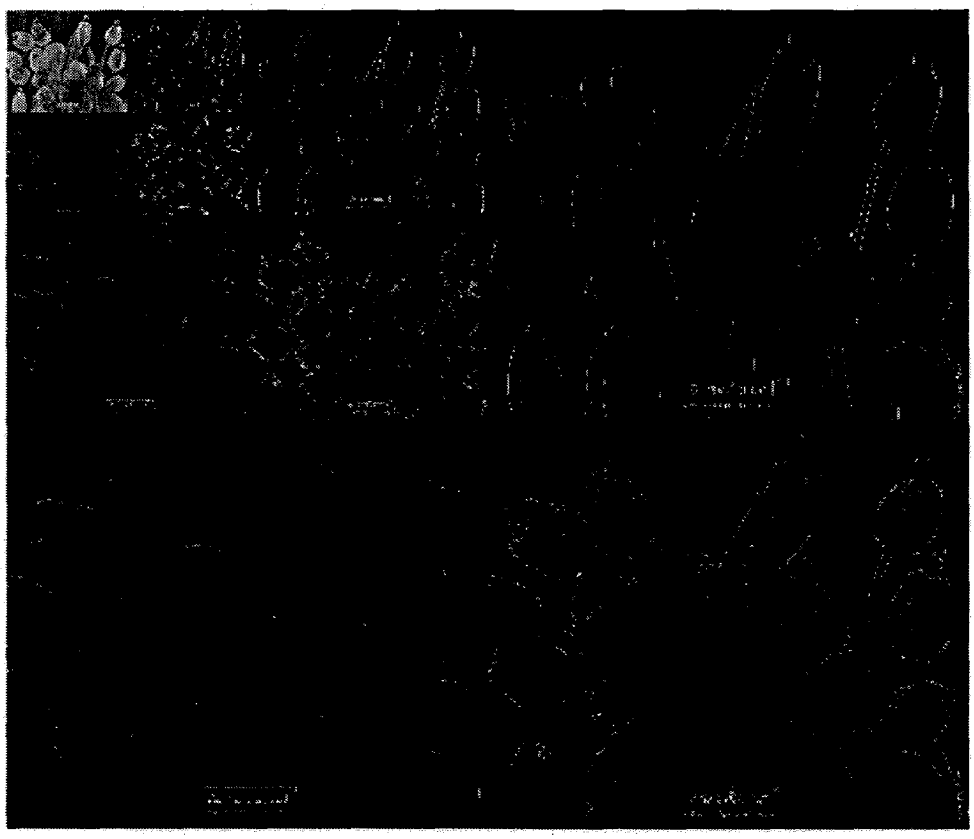


Fig.6.18. 3-level decomposed image of image flowers.tif

6.3. Synthesis Results

Implementation results of the configurations that are used during simulation/synthesis are presented here. The logic resources of the FPGA are divided in different categories as look up tables (LUT), input/output Blocks (IOB), Flip-flops, Multipliers, etc. A specific group of a number of these resources is called configurable logic blocks (CLB) and a group of CLBs forms a slice. The routing resource usage is not given by the place and route tools - higher device utilization implies greater routing resource utilization. The total number of CLBs used is a function of the device resources used and how densely they are packed. For example, on a CLB only a flip flop could be used, but still it adds to the CLB count.

A LUT Flip Flop pair for this architecture represents one LUT paired with one Flip Flop within a slice. A control set is unique combination of clock, reset, set, and enables signals for a registered element. The synthesis details, which indicate amount of logic resources used by our design, are discussed for Prototype CDF Wavelet processor of 3 levels.

Device selected: Spartan3s500efg320-4

Table.6.1. Resource utilization on Spartan3s500efg320-4 for first level decomposition

Slice logic utilization			
Unit Name	**Number used**	**Total Number**	**Percent**
Slice Flip Flops	484	9,312	5
4 input LUTs	966	9,312	10
Logic distribution			
Unit Name	**Number used**	**Total Number**	**Percent**
Occupied Slices	663	4,656	14
Slices containing only related logic	663	663	100

Slices containing unrelated logic	0	663	0
4 input LUTs	1,068	9,312	11
Number used as logic		966	
Number used as a route-thru		102	
I/O Utilization			
Unit Name	**Number used**	**Total Number**	**Percent**
Bonded IOBs	1	232	1
Specific Feature Utilization			
Unit Name	**Number used**	**Total Number**	**Percent**
RAMB16s	3	20	15
BUFGMUXs	4	24	16
BSCANs	1	1	100

Table.6.2. Resource utilization on Spartan3s500efg320-4 for second level decomposition.

Slice logic utilization			
Unit Name	**Number used**	**Total Number**	**Percent**
Slice Flip Flops	483	9,312	5
4 input LUTs	971	9,312	10
Logic distribution			
Unit Name	**Number used**	**Total Number**	**Percent**
Occupied Slices	700	4,656	15
Slices containing only related logic	700	700	100
Slices containing unrelated logic	0	700	0

Number of 4 input LUTs	1,072	9,312	11
Number used as logic		971	
Number used as a route-thru		101	
I/O Utilization			
Unit Name	**Number used**	**Total Number**	**Percent**
Bonded IOBs	1	232	1
Specific Feature Utilization			
Unit Name	**Number used**	**Total Number**	**Percent**
RAMB16s	3	20	15
BUFGMUXs	4	24	16
BSCANs	1	1	100

Table.6.3. Resource utilization on Spartan3s500efg320-4 for third level decomposition

Slice logic utilization			
Unit Name	**Number used**	**Total Number**	**Percent**
Slice Flip Flops	482	9,312	5
4 input LUTs	968	9,312	10
Logic distribution			
Unit Name	**Number used**	**Total Number**	**Percent**
Occupied Slices	666	4,656	14
Slices containing only related logic	666	666	100
Slices containing unrelated logic	0	666	0

Number of 4 input LUTs	1,068	9,312	11
Number used as logic		968	
Number used as a route-thru		100	
I/O Utilization			
Unit Name	**Number used**	**Total Number**	**Percent**
Bonded IOBs	1	232	1
Specific Feature Utilization			
Unit Name	**Number used**	**Total Number**	**Percent**
RAMB16s	3	20	15
BUFGMUXs	4	24	16
BSCANs	1	1	100

The numbers given in resource usage above, show that Spartan3s500efg320-4 device can easily accommodate the design. Larger practical picture sizes merely require an FPGA with more internal or external RAM, i.e., the amount of required logic does not grow, as seen in Tables. For cost reduction, one might want to consider using the smallest possible FPGA that can accommodate the necessary resources and providing an external dual port RAM large enough for the desired picture dimension. Since only about 14% of the FPGA is used, wavelet implementations using other filters like the Daubechies wavelet filter can also be targeted for implementation on the Spartan3s500efg320-4. For any image the resource utilization on FPGA is fixed for 3-levels because of fixed design.

Chapter 7

CONCLUSIONS AND FUTURE SUGGESTIONS

7.1. Conclusions

- In this thesis, an improved parallel architecture for implementation of lifting based CDF Wavelet is proposed. This architecture is first tested using MATLAB software, then implemented on Spartan3E xc3s500e-4fg320. Initially, CDF biorthogonal Wavelet Transform has been studied and its hardware logic has been realized in Matlab simulator using System generator. Simulations have been carried out successfully on standard images. The software and hardware outputs are validated successfully.

- Compared to previous parallel architecture [26], proposed architecture can perform level 1 decomposition of an $N \times N$ image in $N^2/4$ working clock cycles.

7.2. Future Suggestions

- The applications of FPGA in different domains are growing rapidly. In the Digital Image processing field, Wavelet based Image Compression Applications on grayscale images has been discussed. The other image processing techniques like, color image processing, Image Enhancement, Morphological operations, Video processing can be examined by taking FPGA as a platform.
- Similarly, an inverse Wavelet transform can be implemented for extracting the original image from compressed image.
- Architecture for other wavelets can also be developed similarly with reduced clock cycles.

REFERENCES

[1] C. Christopoulos, A. Skodras and T. Ebrahimi, "The JPEG2000 still image coding system: an overview," IEEE Trans. on Consumer Electronics, vol.46, no. 4, pp.1103-1127, 2000.

[2] I. Daubechies, W. Sweldens, 1998, "Factoring wavelet transforms into lifting schemes," J. Fourier Anal. Appl., vol.4, pp.247-269, 2000.

[3] A .R. Calderbank, I. Daubechies, W. Sweldens, and B.L. Yeo, "Wavelet transform that map integers to integers," Applied and Computational Harmonic Analysis (ACHA), vol.5, no.3, pp.332-369, 1998.

[4] K. Andra, C. Chakrabarti and T. Acharya, " VLSI architecture for lifting-based forward and inverse wavelet transform", IEEE Trans. on Signal Processing,vol.50, no.4,pp.966-977, 2002.

[5] C. Chrysafis, and A. Ortega, 2000, "Line-based, reduced memory, wavelet image compression," IEEE Trans. on Image Processing, vol.9,no.3, pp.378-389.

[6] P. Wu, and L. Chen, 2001, "An efficient architecture for two-dimensional discrete wavelet transform," IEEE Trans. on Circuits and Systems for Tech., vol.11, no.4, pp.536-545.

[7] D. Bhatia, 1997, "Reconfigurable computing," 10th International Conference on VLSI Design, Hyderabad, India, pp. 356-359.

[8] How much information? 2003 By UC Berkeley's School of Information Management and Systems.

http://www.sims.berkeley.edu/research/projects/how-much-info-2003

[9] Pennebaker, W. B. and Mitchell, J. L. JPEG - Still Image Data Compression Standards, Van Nostrand Reinhold, 1993.

[10] R. Polikar, "Wavelet tutorial",

http://users.rowan.edu/~polikar/WAVELETS/WT part1.html

[11] A. Abbate, C. M. DeCusatis and P. K. Das, "Wavelets and Subbands: Fundamentals and Applications", Brikhäuser publications, 2002.

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

[13] S. G. Mallat, "Multifrequency Channel Decompositions of Images and Wavelet Models," IEEE Trans. on Acoustics, Speech and Signal Processing, vol.37,no. 12, pp. 2091-2110, 1989.

[14] I. Daubechies, "Orthonormal Bases of Compactly Supported Wavelets," Communications on Pure and Applied Mathematics, John Wiley and Sons, pp. 909-996, 1988.

[15] I. Daubechies, "Biorthogonal Bases of Compactly Supported Wavelets," Communications on Pure and Applied Mathematics, John Wiley and Sons, pp. 485-560, 1992.

[16] David Salomon, "Data Compression" spinger, Fourth Edition2007.

[17] R. C. Gonzalez, R. E. Woods, "Digital Image Processing," *Pearson Education*, II Edition, 2003.

[18] T. Archarya, P. S. Tsai, "JPEG 2000 Standard for Image Compression: Concepts, Algorithms and VLSI Architectures," Wiley Interscience, 2005.

[19] W. Sweldens, "The lifting scheme: A new philosophy in biorthogonal wavelet constructions," in Proc. SPIE, vol.2569, pp.68-79, 1995.

[20] M. D. Adams, F. Kossentine, "Reversible Integer-To-Integer Wavelet Transform of Image Compression: Performance Evaluation and Analysis," IEEE Trans. on Image Processing, vol.9, pp.1010-1024, 2000.

[21] Mathworks URL: http://www.mathworks.com

[22] Xilinx. "*Spartan-3E FPGA Family: Complete Data Sheet*".

[23] Xilinx Inc., "System Generator User Guide", URL: http://www.xilinx.com/products/software/sysgen/app_docs/user_guide.htm

[24] G.E. Martinez-Torres, J.M. Luna-Rivera and R.E. Balderas-Navarro, "FPGA-Based Educational Platform for Wireless Transmission Using System Generator," Proc. of IEEE International Conference on Reconfigurable Computing and FPGA's, ReConFig 2006, vol., no., pp.1-9, Sept. 2006.

[25] Luiz Pires and Deepa Pandalai, Honeywell Technology Center, Minnepoli, MN 55412, "compress.c, " October 1988.

[26] C. Xiong, J. Tian and J. Liu, "Efficient parallel architecture for lifting-based row-dimensional discrete wavelet transform," IEEE Int. Workshop VLSI Design and Video Tech., Suzhou, China, pp.75-78, 2005.

[27] System Generator for DSP performing Hardware-in-the-Loop with Spartan-3E starter kit, available at: www.xilinx.com, date: 19/06/2009.

[28] Spartan-3E starter Kit/Board, User guide available at: http://www.digilentinc.com.

[29] G. K. Kharate, A. A. Ghatol and P.P. Rege, "Image Compression Using Wavelet Packet Tree", ICGST-GVIP Journal, Volume (5), Issue (7), July 2005, pp. 37-40.

APPENDIX-A

1. Procedures for System Generator software installation

Software installation is important stage as incorrect installation may result in errors during synthesis. The order shown below must be followed for successful installation.

Steps

- Windows XP is installed in PC (some Xilinx tools do not work in vista environment).
- Matlab R2006b or above versions is installed.
- Then Xilinx ISE 9.2i is installed.
- Xilinx ISE 9.2i is updated to service pack 2 to using Xilinx service pack 2.
- Then Xilinx System Generator for DSP 9.2i is installed.

  Xilinx Blockset is configured to Matlab Simulink.
- Next, in order for setting up the Spartan™-3E starter kit to enable hardware-in-the-loop verification with JTAG co-simulation via the USB configuration port, we must install Xilinx System Generator Board Description Builder.

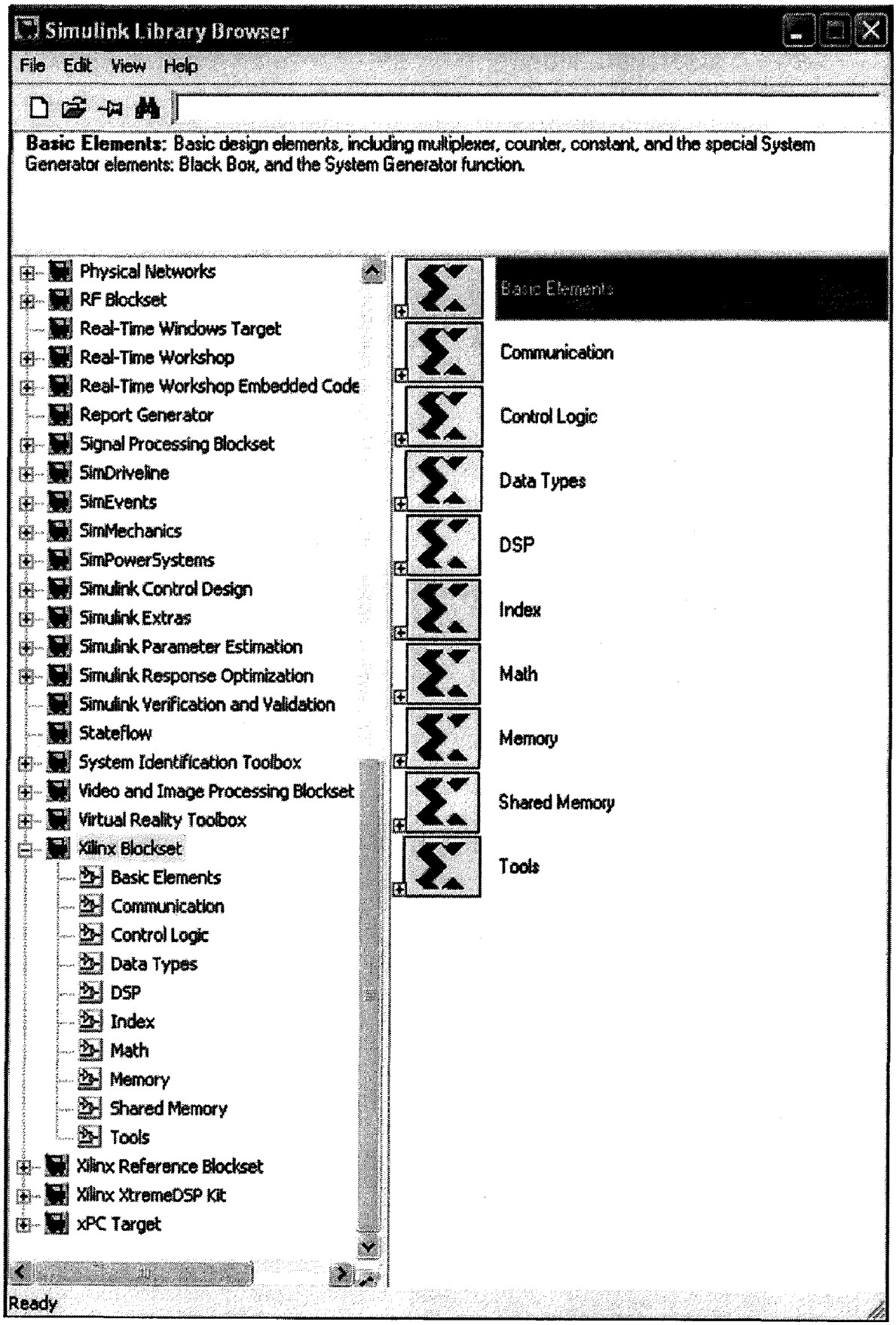


Fig.A.1. Xilinx Blockset in Simulink

The steps for installing the Xilinx System Generator Board Description Builder are

- Double click on **system generator token** and select new compilation target as show below.

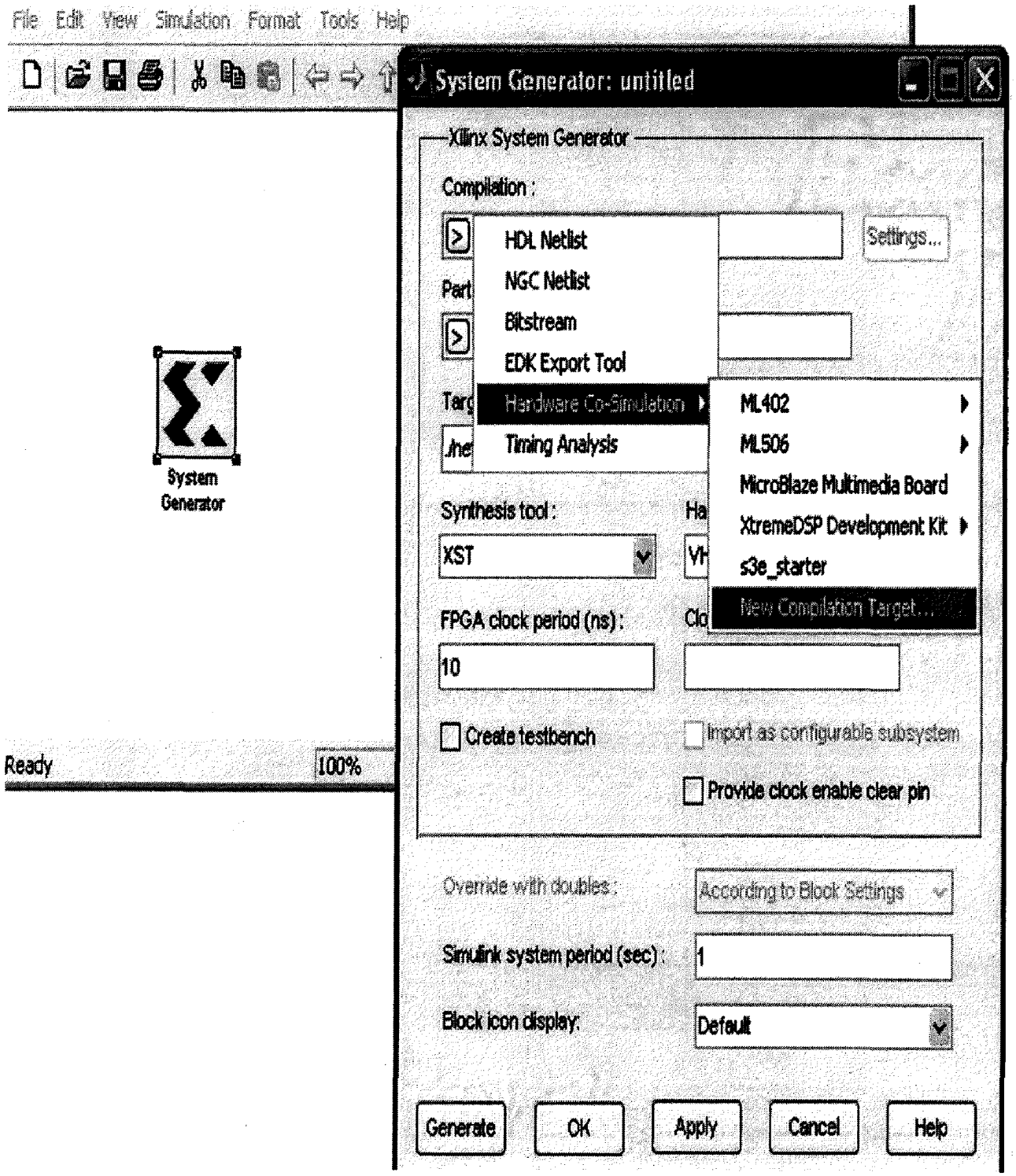


Fig.A.2. Invoke SBD Builder to Create New Hardware Compilation Target

- Enter target board information and clock sections as below.

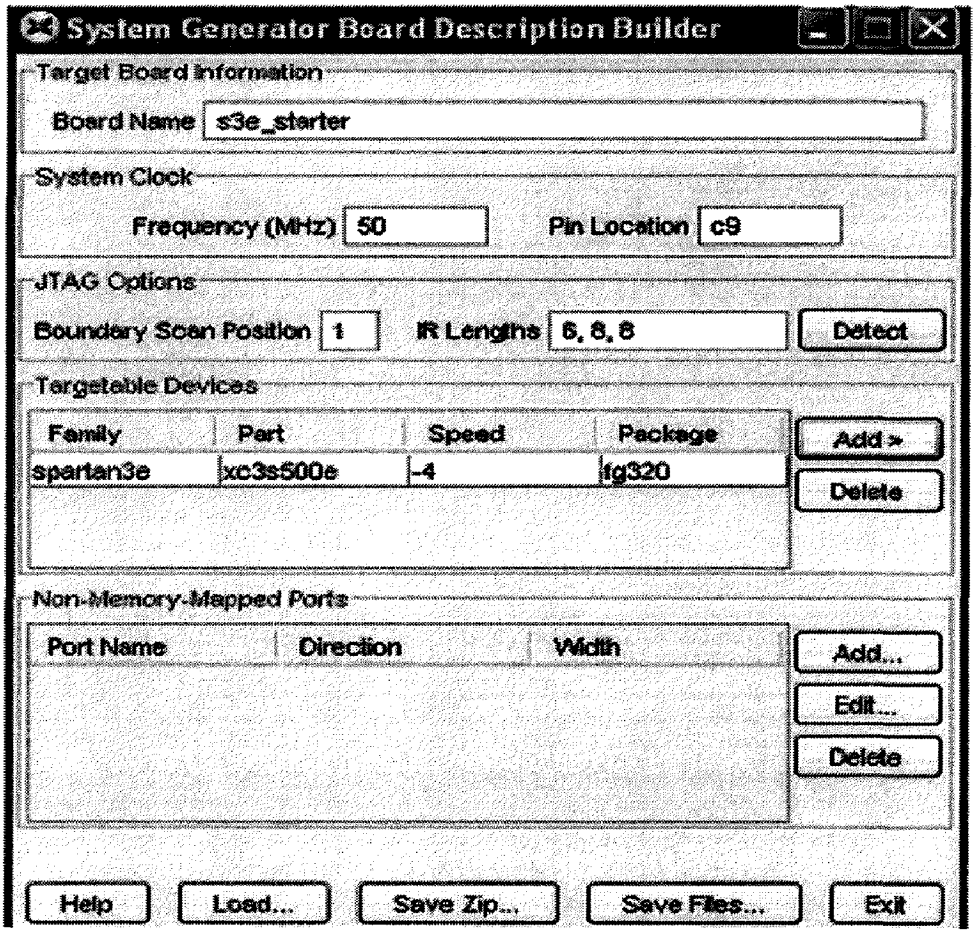


Fig.A.3. *Specify SBD Builder Options for Spartan-3E board*

- Click the Save Files button and save the zip file and exit SBD Builder.
- Now the kit is ready for hardware-in-the-loop verification with JTAG co-simulation via the USB configuration port.

2. FPGA configuration options [28]

A few system-level design trade-offs were required in order to provide the Spartan-3E Starter Kit board with the most functionality.

➢ Download FPGA designs directly to Spartan-3E FPGA via JTAG (**Joint Test Action Group (JTAG)** is the usual name used for the IEEE 1149.1 standard), using the onboard USB interface. .The on-board USB-JTAG logic also provides in-system programming for the on-board platform Flash PROM and the Xilinx XC2C64A CPLD.

- Program the on-board 4Mbit Xilinx XCF04S serial Platform Flash PROM, then configure the FPGA from the image stored in the Platform Flash PROM using Master Serial mode.

- Program the on-board 16 Mbit ST Microelectronics SPI serial Flash PROM, then configure the FPGA from the image stored in the SPI serial Flash PROM using SPI mode.

- Program the on-board 128Mbit Intel StrataFlash parallel NOR Flash PROM, then configures the FPGA from the image stored in the Flash PROM using BPI Up or BPI Down configuration modes. Further, an FPGA application can dynamically load two different FPGA configurations using the Spartan-3E FPGA's MultiBoot mode.

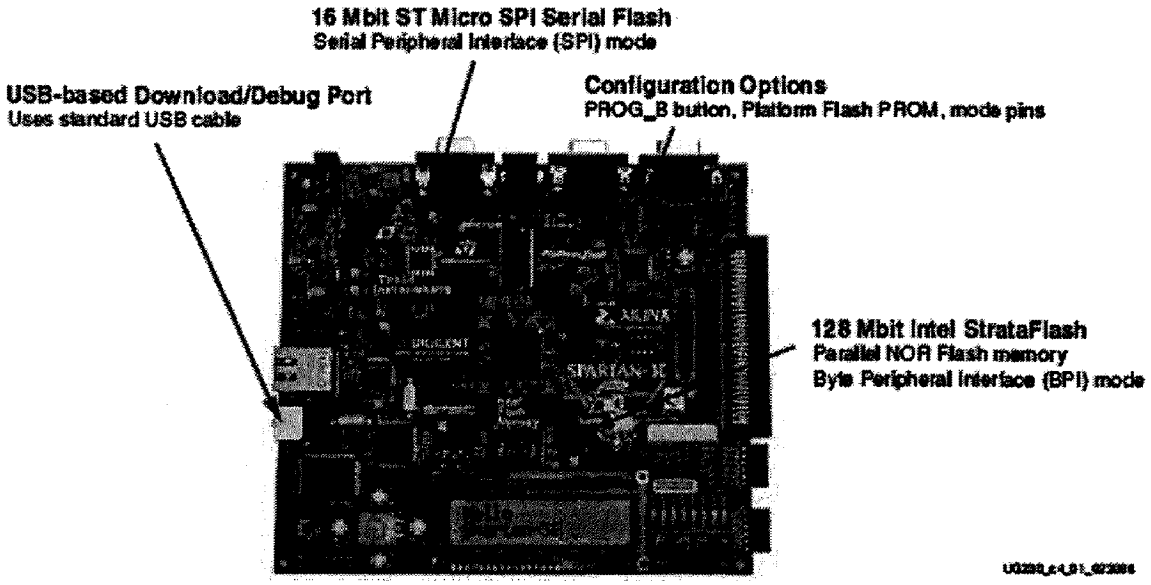


Fig.A.4. Spartan-3E Starter Kit FPGA Configuration Options

APPENDIX-B

1. M-code for Raster scanning image of size m x m

```
tic
clc;
clear all;
I=imread
I=rgb2gray(I);
img_res=imresize(I,[512 512],'bicubic');
figure,imshow(img_res)
k=0;
b=zeros(512*512,1);
c=zeros(512*512,1);
[m n]=size(img_res);
for i=1:512
    for j=1:512
      k=k+1;
      b(k)=b(k)+img_res(i,j);
   c(k)=c(k)+k;
    end;
end;
pvtin=[c b];
disp(pvtin);
toc
```

2. M-code to display output image

```
as=zeros(512,512);
as1=zeros(512,512);

k=516;
for i=1:512
    for j=1:512
```

```
        k=k+1;
        as(i,j)=as(i,j)+row4(k);
    end;
end;

for i=1:512
    m=1;n=2;%flag=0;
    for j=1:512
        if j<=256
        as1(i,j)=as(i,m);
        m=m+2;
        else
        as1(i,j)=as(i,n);
        n=n+2;
        end
    end
end
% disp(as)
%as=uint8(row3);
 figure,imshow(as/255)
 figure,imshow(as1/255)
```

3. Reconstructed images

The decomposed image of CDF wavelet transform is reconstructed using Matlab code and compared with original image and the reconstructed image is having good clarity as original image.

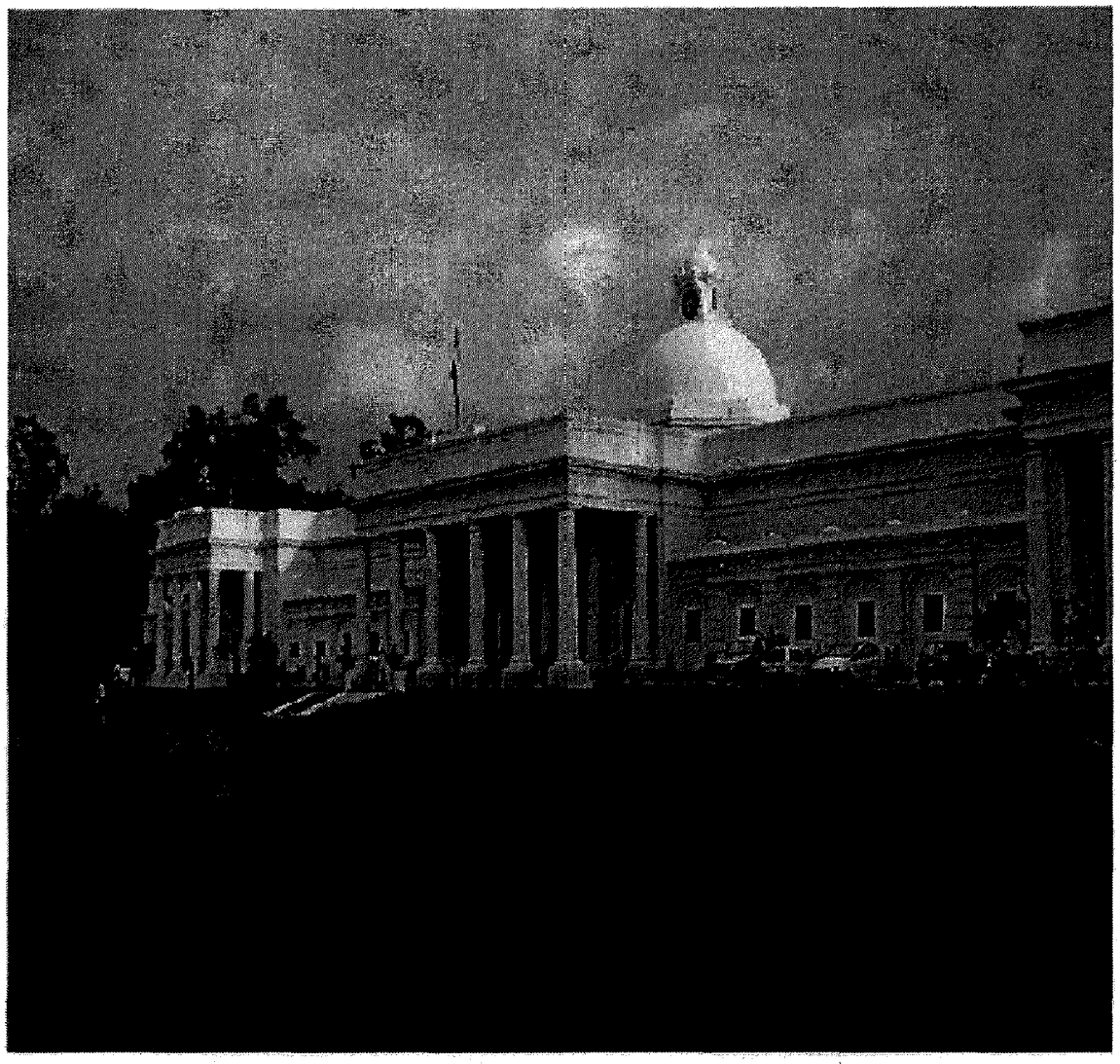


Fig.B.1. Original image IITR.TIF of size 512x512

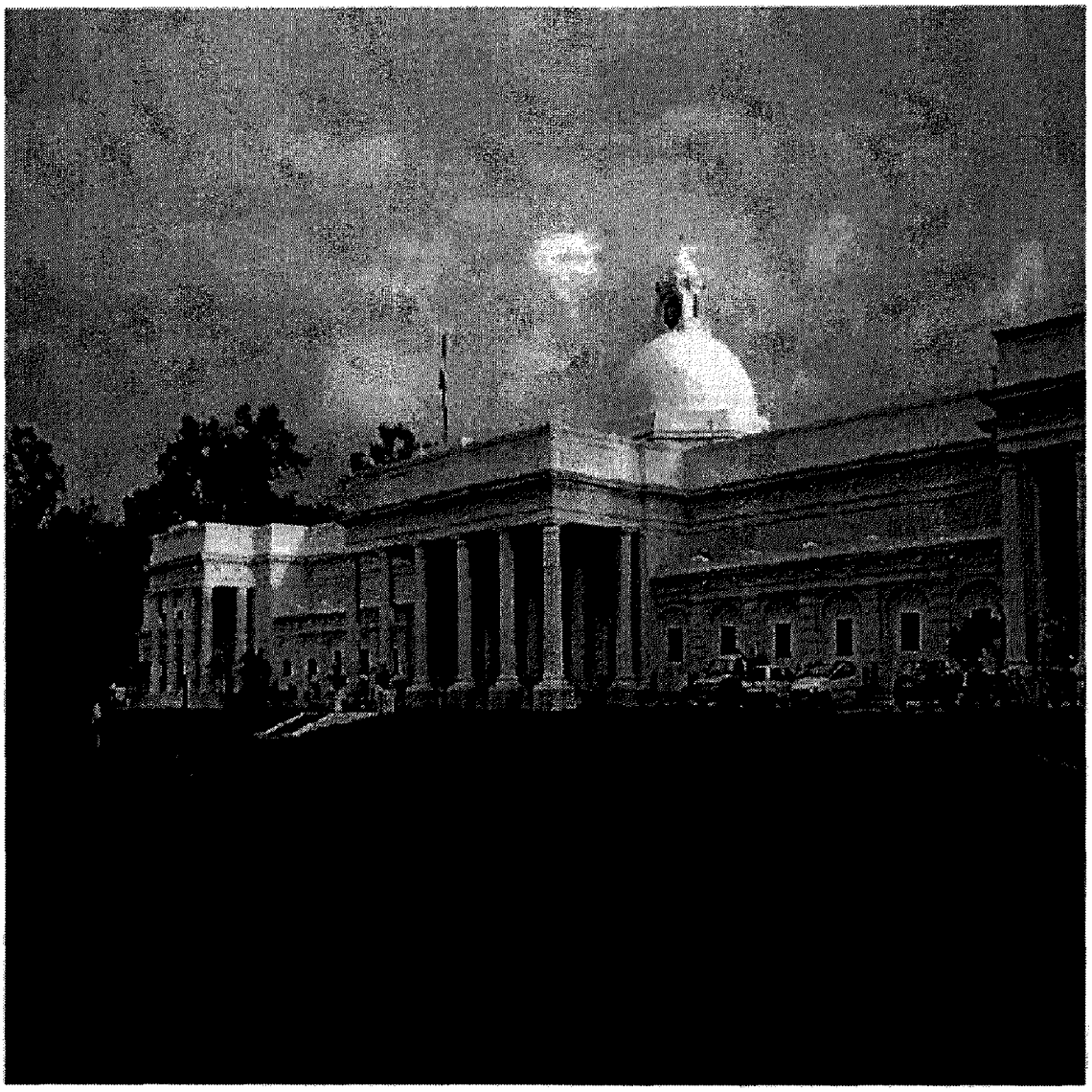


Fig. B.2. Reconstructed image IITR.TIF of size 512x512