2003

# Efficient Implementations of Discrete Wavelet Transforms Using Fpgas

Deepika Sripathi

THE FLORIDA STATE UNIVERSITY

COLLEGE OF ENGINEERING

# EFFICIENT IMPLEMENTATIONS OF DISCRETE WAVELET TRANSFORMS USING FPGAs

By

DEEPIKA SRIPATHI

A Thesis submitted to the
Department of Electrical and Computer Engineering
in partial fulfillment of the
requirements for the degree of
Master of Science

Degree Awarded:
Fall Semester, 2003

The members of the committee approve the thesis of Deepika Sripathi defended on November 18th, 2003.

Simon Y. Foo
Professor Directing Thesis

Uwe Meyer-Baese
Committee Member

Anke Meyer-Baese
Committee Member

Approved:

Reginald J. Perry, Chair, Department of Electrical and Computer Engineering

The office of Graduate Studies has verified and approved the above named committee members

# ACKNOWLEDGEMENTS

I would like to express my gratitude to my major professor, Dr. Simon Foo for his guidance, advice and constant support throughout my thesis work. I would like to thank him for being my advisor here at Florida State University. I would like to thank Dr. Uwe Meyer-Baese for his guidance and valuable suggestions. I also wish to thank Dr. Anke Meyer-Baese for her advice and support. I would like to thank my parents for their constant encouragement. I would like to thank my husband for his cooperation and support. I wish to thank the administrative staff of the Electrical and Computer Engineering Department for their kind support. Finally, I would like to thank Dr. Shonda Walker, all the members of Machine Intelligence Lab, and my friends here at Florida State University.

# TABLE OF CONTENTS

# LIST OF TABLES

# LIST OF FIGURES

# ABSTRACT

Recently the Wavelet Transform has gained a lot of popularity in the field of signal processing. This is due to its capability of providing both time and frequency information simultaneously, hence giving a time-frequency representation of the signal. The traditional Fourier Transform can only provide spectral information about a signal. Moreover, the Fourier method only works for stationary signals. In many real world applications, the signals are non-stationary. One solution for processing non-stationary signals is the Wavelet Transform.

Currently, there is tremendous focus on the application of Wavelet Transforms for real-time signal processing. This leads to the demand for efficient architectures for the implementation of Wavelet Transforms. Due to the demand for portable devices and real-time applications, the design has to be realized with very low power consumption and a high throughput.

In this thesis, different architectures for the Discrete Wavelet Transform filter banks are presented. The architectures are implemented using Field Programmable Gate Array devices. Design criteria such as area, throughput and power consumption are examined for each of the architectures so that an optimum architecture can be chosen based on the application requirements. In our case study, a Daubechies 4-tap orthogonal filter bank and a Daubechies 9/7-tap biorthogonal filter bank are implemented and their results are discussed. Finally, a scalable architecture for the computation of a three-level Discrete Wavelet Transform along with its implementation using the Daubechies length-4 filter banks is presented.

# CHAPTER 1

# INTRODUCTION

## 1.1 Introduction

In general, signals in their raw form are time-amplitude representations. These time-domain signals are often needed to be transformed into other domains like frequency domain, time-frequency domain, etc., for analysis and processing. Transformation of signals helps in identifying distinct information which might otherwise be hidden in the original signal. Depending on the application, the transformation technique is chosen, and each technique has its advantages and disadvantages.

## 1.2 Why Wavelet Transforms?

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

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

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

## 1.3 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.

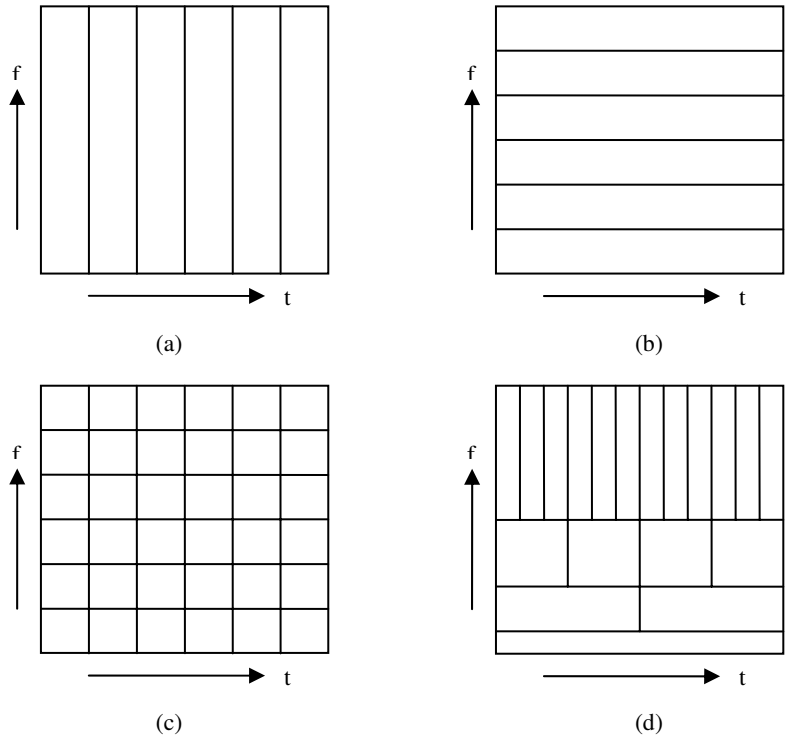



Figure 1.1 The Time-Frequency tiling for (a) Time-Domain (b) Frequency-Domain (c) STFT (d) DWT.

Figure 1.1(a) shows the time-frequency tiling in the time-domain plane and figure 1.1(b) shows the tiling in frequency-domain plane. It is seen that figure 1.1(a) does not give any frequency information and figure 1.1(b) does not give any time information. Similarly figure 1.1(c) shows the tiling in STFT and figure 1.1(d) shows the tiling in Wavelet 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.

## 1.4 The Need for Efficient DWT Architecture

The properties of Wavelet Transform allow it to be successfully applied to non-stationary signals for analysis and processing, e.g., speech and image processing, data compression, communications, etc. Due to its growing number of applications in various areas, it is necessary to explore the hardware implementation options of the Discrete Wavelet Transform (DWT).

An efficient design should take into account aspects such as area, power consumption, throughput, etc. Techniques such as pipelining, distributed arithmetic, etc., help in achieving these requirements. For most applications such as speech, image, audio and video, the most crucial problems are the memory storage and the global data transfer. Therefore, the design should be such that these factors are taken into consideration.

In this thesis, Field Programmable Gate Arrays (FPGAs) are used for hardware implementation of the DWT. FPGAs have application specific integrated circuits (ASICs) characteristics with the advantage of being reconfigurable. They contain an array of logic cells and routing channels (called interconnects) that can be programmed to suite a specific application. At present, the FPGA based ASIC market is rapidly expanding due to demand for DSP applications. FPGA implementation could be challenging as they do not have good arithmetic capabilities when compared with the general purpose DSP processors. However, the most important advantage of using an FPGA is because it is reprogrammable. Any modifications can be easily accomplished and additional features can be added at no cost which is not the case with traditional ASICs.

## 1.5 Outline of Thesis

This thesis is organized as follows. Chapter 2 presents multi-resolution analysis and Continuous Wavelet Transform. The Discrete Wavelet Transform is described in detail, and finally some wavelet families and applications are discussed.

Most DWT architectures are implemented using filter banks. Chapter 3 describes the various filter bank structures and provides a comparison of the filter banks. The distributed arithmetic technique and its application to filter banks are described in detail.

In Chapter 4, the Daubechies length-4 orthogonal and the 9/7 biorthogonal filter banks are implemented as a case study using the methods presented in Chapter 3 and their results are analyzed.

Chapter 5 describes the architecture for computing higher octave DWT. As a case study, the implementation of 3-level architecture is presented and the results are studied.

Chapter 6 concludes the thesis and provides suggestions for future work.

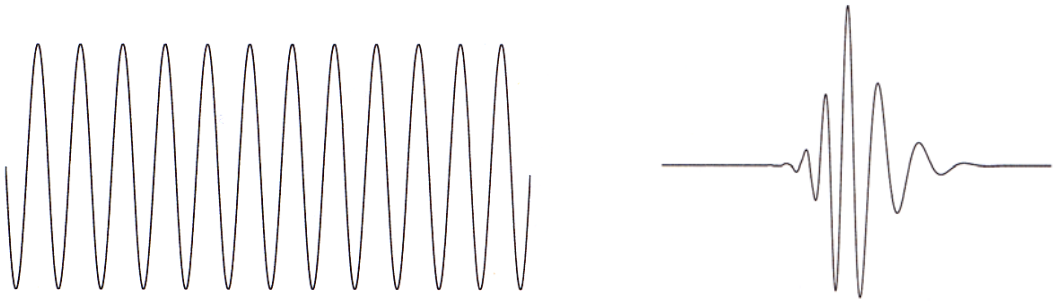
# CHAPTER 2

# THE DISCRETE WAVELET TRANSFORM

## 2.1 Introduction

The transform of a signal is just another form of representing the signal. It does not change the information content present in the signal. The Wavelet Transform provides a time-frequency representation of the signal. It was developed to overcome the short coming of the Short Time Fourier Transform (STFT), which can also be used to analyze non-stationary signals. While STFT gives a constant resolution at all frequencies, the Wavelet Transform uses multi-resolution technique by which different frequencies are analyzed with different resolutions.

A wave is an oscillating function of time or space and is periodic. In contrast, wavelets 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.



(a) (b)

Figure 2.1 Demonstration of (a) a Wave and (b) a Wavelet [2].

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.

## 2.2 The Continuous Wavelet Transform and the Wavelet Series

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

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

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.

## 2.3 The Discrete Wavelet Transform

The Wavelet Series is just a sampled version of CWT and its computation may consume significant amount of time and resources, depending on the resolution required. The Discrete Wavelet Transform (DWT), 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 [5]. 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.

## 2.4 DWT and Filter Banks

### 2.4.1 Multi-Resolution Analysis using Filter Banks

Filters are one of the most widely used signal processing functions. Wavelets can be realized by iteration of filters with rescaling. The resolution of the signal, which is a measure of the amount of detail information in the signal, is determined by the filtering operations, and the scale is determined by upsampling and downsampling (subsampling) operations[5].

The DWT is computed by successive lowpass and highpass filtering of the discrete time-domain signal as shown in figure 2.2. 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].

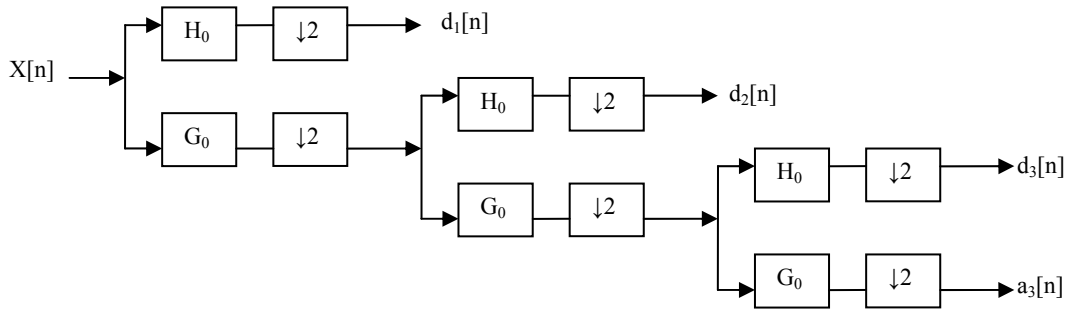



Figure 2.2 Three-level wavelet decomposition tree.

At each decomposition level, the half band filters produce signals spanning only half the frequency band. This doubles the frequency resolution as the uncertainity 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 ω/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 figure 1.1(d) of Chapter 1. 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.

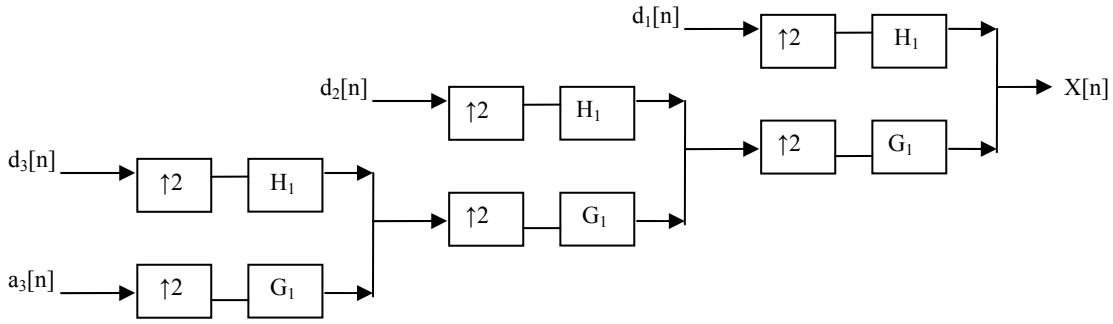



Figure 2.3 Three-level wavelet reconstruction tree.

Figure 2.3 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, $G_1$ and $H_1$.

### 2.4.2 Conditions for Perfect Reconstruction

In most Wavelet Transform applications, it is required that the original signal be synthesized from the wavelet coefficients. To achieve perfect reconstruction the analysis and synthesis filters have to satisfy certain conditions. Let $G_0(z)$ and $G_1(z)$ be the low pass analysis and synthesis filters, respectively and $H_0(z)$ and $H_1(z)$ the high pass analysis and synthesis filters respectively. Then the filters have to satisfy the following two conditions as given in [4] :

$$G_0(-z)\,G_1(z) + H_0(-z).\,H_1(z) = 0 \qquad 2.2$$

$$G_0(z)\,G_1(z) + H_0(z).\,H_1(z) = 2z^{-d} \qquad 2.3$$

The first condition implies that the reconstruction is aliasing-free and the second condition implies that the amplitude distortion has amplitude of one. It can be observed that the perfect reconstruction condition does not change if we switch the analysis and synthesis filters.

There are a number of filters which satisfy these conditions. But not all of them give accurate Wavelet Transforms, especially when the filter coefficients are quantized. The accuracy of the Wavelet Transform can be determined after reconstruction by calculating the Signal to Noise Ratio (SNR) of the signal. Some applications like pattern recognition do not need reconstruction, and in such applications, the above conditions need not apply.

### 2.4.3 Classification of wavelets

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

**(a)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

$$H_0(z) = z^{-N} G_0(-z^{-1}) \qquad 2.4$$

The two filters are alternated flip of each other. The alternating flip automatically gives double-shift orthogonality between the lowpass and highpass filters [1], i.e., the scalar product of the filters, for a shift by two is zero. i.e., $\sum G[k]\, H[k-2l] = 0$, where $k,l \in Z$ [4]. Filters that satisfy equation 2.4 are known as Conjugate Mirror Filters (CMF). Perfect reconstruction is possible with alternating flip.

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.

**(b)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 antisymmetric 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.

## 2.5 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.

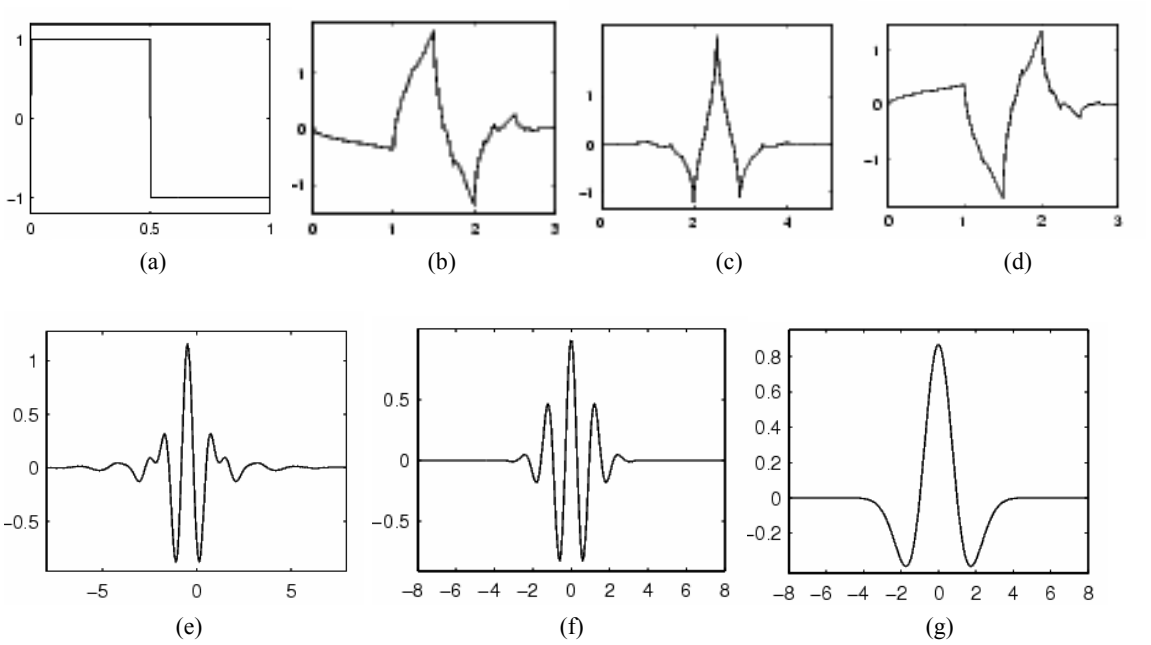



Figure 2.4 Wavelet families (a) Haar (b) Daubechies4 (c) Coiflet1 (d) Symlet2 (e) Meyer (f) Morlet (g) Mexican Hat.

Figure 2.4 illustrates some of the commonly used wavelet functions. Haar wavelet is one of the oldest and simplest wavelet. Therefore, any discussion of wavelets starts with the Haar wavelet. Daubechies wavelets are the most popular wavelets. They represent the foundations of wavelet signal processing and are used in numerous applications. These are also called Maxflat wavelets as their frequency responses have maximum flatness at frequencies 0 and π. This is a very desirable property in some

applications. The Haar, Daubechies, Symlets and Coiflets are compactly supported orthogonal wavelets. These 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.

## 2.6 Applications

There is a wide range of applications for Wavelet Transforms. They are applied in different fields ranging from signal processing to biometrics, and the list is still growing. One of the prominent applications is in the FBI fingerprint compression standard. Wavelet Transforms are used to compress the fingerprint pictures for storage in their data bank. The previously chosen Discrete Cosine Transform (DCT) did not perform well at high compression ratios. It produced severe blocking effects which made it impossible to follow the ridge lines in the fingerprints after reconstruction. This did not happen with Wavelet Transform due to its property of retaining the details present in the data.

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

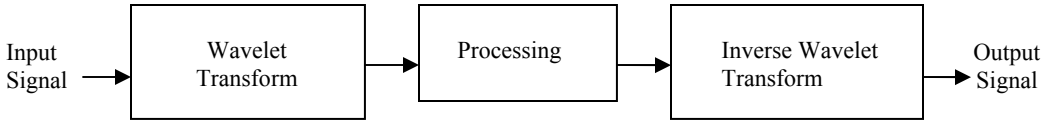



Figure 2.5 Signal processing application using Wavelet Transform.

Figure 2.5 shows the general steps followed in a signal processing application. Processing may involve compression, encoding, denoising etc. The processed signal is either stored or transmitted. For most compression applications, processing involves quantization and entropy coding to yield a compressed image. During this process, all the wavelet coefficients that are below a chosen threshold are discarded. These discarded coefficients are replaced with zeros during reconstruction at the other end. To reconstruct the signal, the entropy coding is decoded, then quantized and then finally Inverse Wavelet Transformed.

Wavelets also find application in speech compression, which reduces transmission time in mobile applications. They are used in denoising, edge detection, feature extraction, speech recognition, echo cancellation and others. They are very promising for real time audio and video compression applications. Wavelets also have numerous applications in digital communications. Orthogonal Frequency Division Multiplexing (OFDM) is one of them. Wavelets are used in biomedical imaging. For example, the ECG signals, measured from the heart, are analyzed using wavelets or compressed for storage. The popularity of Wavelet Transform is growing because of its ability to reduce distortion in the reconstructed signal while retaining all the significant features present in the signal.

# CHAPTER 3

# WAVELET FILTER BANK ARCHITECTURES

## 3.1 Filter Bank Architectures

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. We will consider the following filter bank structures and their properties, specifically with reference to DWT.

(1) Direct form structure

(2) Polyphase structure

(3) Lattice structure

(4) Lifting structure

### 3.1.1 Direct Form Structure

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

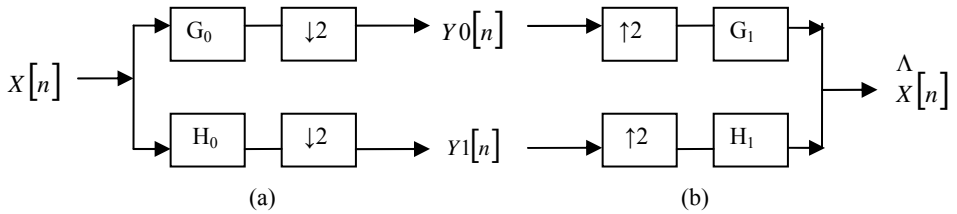



Figure 3.1 Direct form structure of (a) analysis filter bank and (b) synthesis filter.

In the analysis filter bank, x[n] is the discrete input signal, $G_0$ is the low pass filter and $H_0$ is the high pass filter. ↓2 represents decimation by 2 and ↑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 equations 3.1 and 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. For perfect output the filter banks must obey the conditions for perfect reconstruction.

### 3.1.2 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 equation 3.3:

$$\begin{bmatrix} G_{0\,even} & G_{0\,odd} \\ H_{0\,even} & H_{0\,odd} \end{bmatrix} \times \begin{bmatrix} X_{even} \\ z^{-1} X_{odd} \end{bmatrix} = H_p \begin{bmatrix} X_{even} \\ z^{-1} X_{odd} \end{bmatrix} = \begin{bmatrix} Y_0 \\ Y_1 \end{bmatrix} \qquad 3.3$$

The filters with $G_{0even}$ and $G_{0odd}$ are half as long a $G_0$, 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 figure 3.2 illustrates polyphase analysis and synthesis filter banks.

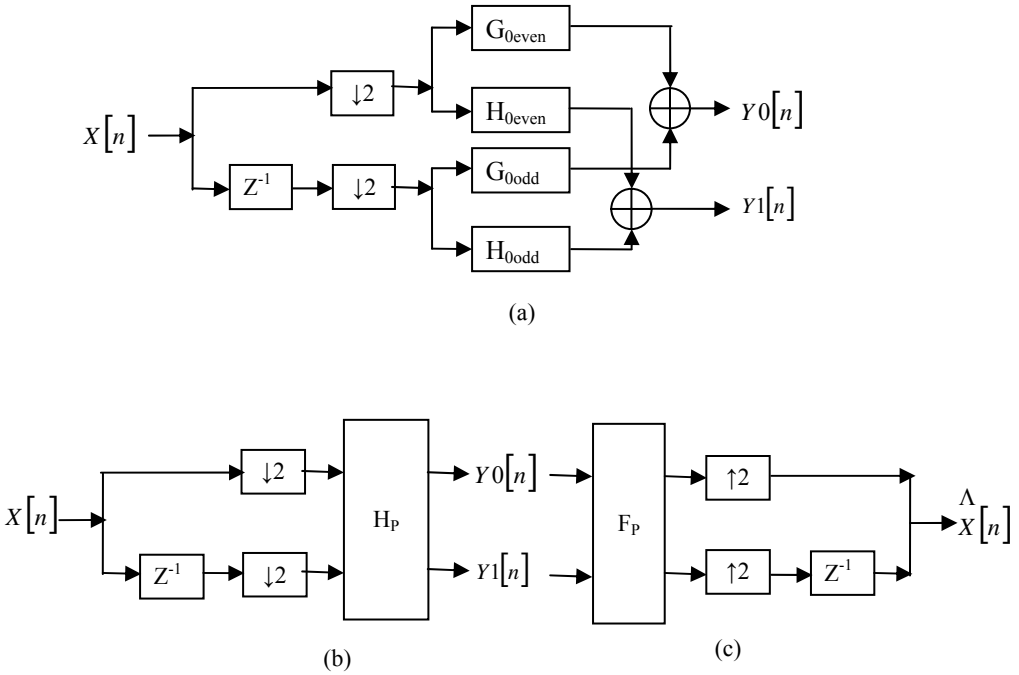



Figure 3.2 Polyphase structure of (a) analysis filter bank (b) equivalent representation of analysis filter bank and (c) synthesis filter bank.

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 are reduced by half in both the analysis and synthesis filter banks, the overall efficiency is increased by 50%. Thus, the polyphase form allows efficient hardware realizations.

### 3.1.3 Lattice Structure

In the above structure, the polyphase matrix, $H_P(z)$ can be replaced by a lattice structure. The filter bank, $H_P(z)$ can be obtained if the filters $G_0(z)$ and $H_0(z)$ are known. Similarly, if $H_P(z)$ is known, the lattice structure can be derived by representing it as a product of simple matrices. The wavelet filter banks have highly efficient lattice structures which are easy to implement.

The lattice structure reduces the number of coefficients and this reduces the number of multiplications. The structure consists of a design parameter *k* and a single overall multiplying factor. The factor *k* is collected from all the coefficients of the filter. For any *k*'s, a cascade of linear phase filters is linear phase and a cascade of orthogonal filters is orthogonal.[1] The complete lattice structure for an orthogonal filter bank is shown in figure 3.3, where β is the overall multiplying factor of the cascade.

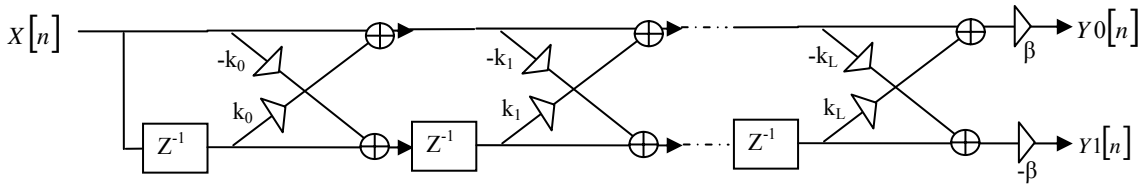



Figure 3.3 Lattice structure of an orthogonal filter bank.

The lattice structure improves the filter bank efficiency as it reduces the number of computations performed. If the direct form requires 4L multiplications, the polyphase requires 2L multiplications, and the lattice requires just L+1 multiplications. The number of additions are also reduced in the lattice form.

### 3.1.4 Lifting Structure

The lifting scheme proposed independently by Herley and Swelden [1] is a fast and efficient method to construct two-channel filter banks. It consists of two steps: lifting and dual lifting. The design starts with the Haar filter or the Lazy filter which is a perfect reconstruction filter bank with $G_0(z) = H_1(z)=1$ and $H_0(z) = G_1(z) = z^{-1}$. The lifting steps as given in [4] are :

Lifting: $H^{'}(z) = H(z) + G(-z)\, S(z^2)$ for any $S(z^2)$.

Dual Lifting: $G^{'}(z) = G(z) + H(-z)\, T(z^2)$ for any $T(z^2)$.

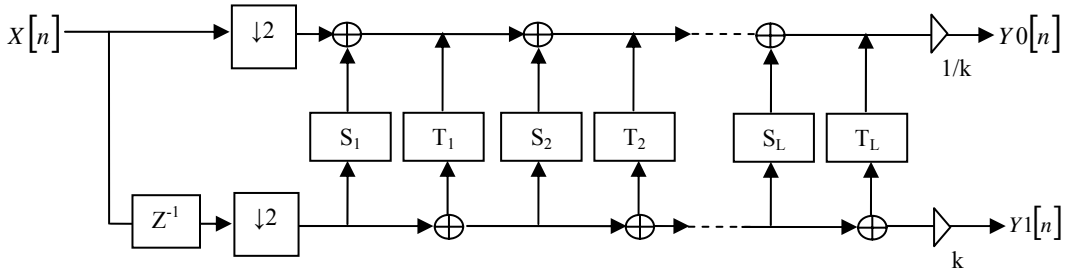



Figure 3.4 Lifting implementation.

The lifting implementation is shown in figure 3.4. The lifting and dual lifting steps are alternated to produce long filters from short ones. Filters with good properties which satisfy the perfect reconstruction properties can be built using this method.

## 3.2 Comparison of implementation options

For hardware implementation, the choice of filter bank structure determines the efficiency and accuracy of computation of the DWT. All structures have some advantages and drawbacks which have to be carefully considered and based on the application, the most suitable implementation can be selected.

It is observed that the direct form is a very inefficient method for DWT implementation. This method is almost never used for DWT computation. The polyphase structure appears to be an efficient method for DWT computation. But the lattice and lifting implementations require fewer computations than the polyphase implementation and therefore are more efficient in terms of number of computations. However, the polyphase implementation can be made more efficient than the lattice and lifting schemes in case of long filters by incorporating techniques like Distributed Arithmetic. Also, the lattice structure cannot be used for all linear phase filters and imposes restrictions on the length of the filters.

In the case of the lattice and lifting schemes, the filtering units cannot operate in parallel as each filtering unit depends on results from the previous filtering unit. In the case of convolutional polyphase implementation, the units can operate in parallel, and therefore the filtering operations have less delay. However, pipelining can be used in the other schemes to reduce the delay.

Often, for implementation purposes, the real number filter coefficients are quantized into binary digits. This introduces some quantization error. In the lifting scheme, the inaccuracy due to quantization is accumulated with each step. Thus, the lifting scheme constants must be quantized with better accuracy than the convolutional filter constants [7], i.e., the lifting constants need to be represented by more number of bits.

## 3.3 Distributed Arithmetic Technique

### 3.3.1 DA-based approach for the filter bank

Distributed Arithmetic (DA) [4] has been one of the popular techniques to compute the inner product equation in many DSP FPGA applications. It is applicable in cases where the filter coefficients are known a priori. The inner sum of products is rearranged so that the multiply and accumulate (MAC) operation is reduced to a series of

look-up table (LUT) calls, and two's complement (2C) shifts and adds. Therefore, the multipliers which occupy large areas are replaced by small tables of pre-computed sums stored on FPGA LUTs which reduce the filter hardware resources.

Consider the following inner product calculation shown in 3.4(a) where c[n] represents an N-tap constant coefficient filter and x[n] represents a sequence of B-bit inputs:

$$y = \sum_{n=0}^{N-1} c[n].x[n] \qquad 3.4(a)$$

$$= \sum_{n=0}^{N-1} c[n].\sum_{b=0}^{B-1} x_b[k].2^b \qquad 3.4(b)$$

$$= \sum_{b=0}^{B-1} 2^b.\sum_{n=0}^{N-1} c[n].x_b[n] \qquad 3.4(c)$$

In equation 3.4(a), the inputs can be replaced as in 3.4(b) where $x_b[k]$ denotes the $b^{th}$ bit of $k^{th}$ sample of x[n]. Rearranging equation 3.4(b) gives 3.4(c). All the possible values of the inner function $\sum_{n=0}^{N-1} c[n].x_b[n]$ in (c) can be pre-computed and stored in an LUT. Now, the equation can be implemented using an LUT, a shifter and an adder. The architectures for the conventional MAC operation, represented by equation 3.4(a), and the DA-based shift-add operation, represented by equation 3.4(c) are shown in figure 3.5 for a 4-tap filter.

In the DA architecture, the input samples are fed to the parallel-to-serial shift register cascade. For an N-tap filter and B-bit input samples, there are N shift registers of B-bits each. As the input samples are shifted serially through the B-bit shift registers, the bit outputs (one bit from each of N registers) of the shift register cascade are taken as address inputs by the look-up table (LUT). The LUT accepts the N bit input vector $x_b$, and outputs the value of $\sum_{n=0}^{N-1} c[n].x_b[n]$ which is already stored in the LUT. For an N-tap filter a $2^N$ word LUT is required. The LUT output is then shifted based on the weight of

$x_b$ and then accumulated. This process is followed for each bit of the input sample before a new output sample is available. Thus for a B-bit input precision a new inner product *y* is computed every B clock cycles.

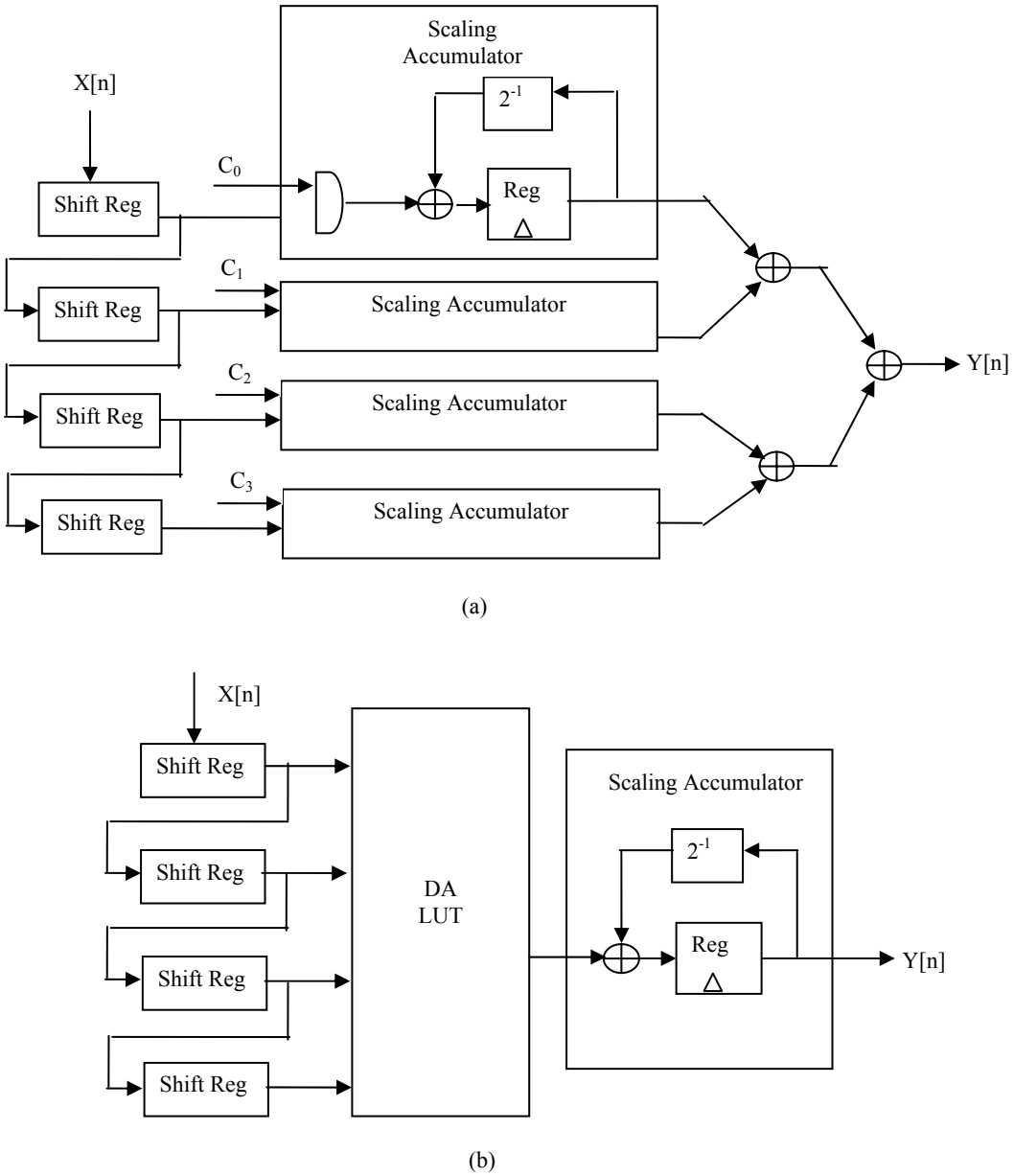



Figure 3.5 (a) Conventional MAC and (b) shift-add DA architectures.

Consider a four-tap serial FIR filter with coefficients $C_0$, $C_1$, $C_2$, $C_3$. The DA-LUT table is as shown in table 3.1. The table consists of the sums of the products of the N bit input vector $x_b$ (N = 4 in this case) and the filter coefficients for all possible combinations.

Table 3.1 DA-LUT table for a 4-tap filter.

| Address | Data |
|---|---|
| 0000 | 0 |
| 0001 | $C_0$ |
| 0010 | $C_1$ |
| 0011 | $C_0+C_1$ |
| ⋮ | ⋮ |
| ⋮ | ⋮ |
| ⋮ | ⋮ |
| 1110 | $C_1+C_2+C_3$ |
| 1111 | $C_0+C_1+C_2+C_3$ |

In conventional MAC-based filter, the throughput is based on the filter length. As the number of filter taps increase, the throughput decreases. In case of DA-based filter, the throughput depends on the input bit precision as seen above and is independent of the filter taps. Thus the filter throughput is de-coupled from the filter length. But when the filter length is increased, the throughput remains the same while the logic resources increase. In case of long filters, instead of creating a large table, it can be partitioned into smaller tables and their outputs can be combined. With this approach, the size of the circuit grows linearly with the number of filter taps rather than exponentially.

For a DWT filter bank, the equation 3.4(c) can be extended to equation 3.5(a) and 3.5(b) to define the low pass and high pass filtering operations.

$$y_0 = \sum_{b=0}^{B-1} 2^b . \sum_{n=0}^{N-1} G_0[n] . x_b[n] \qquad 3.5(a)$$

$$y_1 = \sum_{b=0}^{B-1} 2^b . \sum_{n=0}^{N-1} H_0[n] . x_b[n] \qquad 3.5(b)$$

The polyphase form of the above filters can be obtained by splitting the filters and the input, x[n] into even and odd phases to obtain four different filters. Since the length of each filter is now halved they require much smaller LUTs.

$$y_0 = \sum_{b=0}^{B-1} 2^b . \sum_{n=0}^{N/2-1} G_{0even}[n] . x_{evenb}[n] + \sum_{b=0}^{B-1} 2^b . \sum_{n=0}^{N/2-1} G_{0odd}[n] . x_{oddb}[n] \qquad 3.6(a)$$

$$y_1 = \sum_{b=0}^{B-1} 2^b . \sum_{n=0}^{N/2-1} H_{0even}[n] . x_{evenb}[n] + \sum_{b=0}^{B-1} 2^b . \sum_{n=0}^{N/2-1} H_{0odd}[n] . x_{oddb}[n] \qquad 3.6(b)$$

**Parallel Distributed Arithmetic for Increased Speed**

DA-based computations are inherently bit-serial. Each bit of the input is processed before each output is computed. For a B-bit input, it takes B clock cycles to compute one output. Thus, this serial distributed arithmetic (SDA) filter has a low throughput. The speed can be increased by partitioning the input words into smaller words and processing them in parallel. As the parallelism increases, the throughput increases proportionally, and so does the number of LUTs required.

Filters can be designed such that several bits of the input are processed in a clock period. Partitioning the input word into M sub-words requires M-times as many memory LUTs and this increases the storage requirements. But, now a new output is computed every B/M clock cycles instead of every B cycles. A fully parallel DA (PDA) filter is achieved by factoring the input into single bit sub-words which achieves maximum speed. A new output is computed every clock cycle. This method provides exceptionally high-performance, but comes at the expense of increased FPGA resources. Figure 3.6 shows a parallel DA architecture for an N-tap filter with 4-bit inputs.

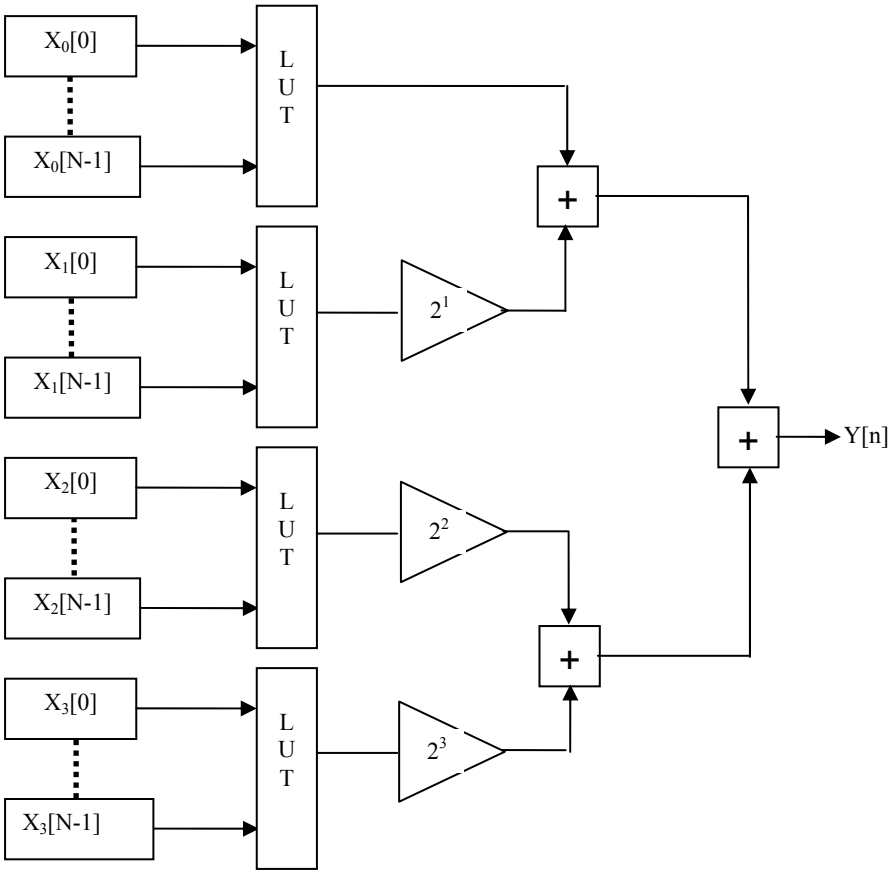



Figure 3.6 Parallel DA architecture [4].

**Processing for multiple channels**

In some applications, the same filter is applied to different inputs. In this case, instead of using two separate filters, a single filter can be shared among the different inputs. Sharing of filters decreases the filter sample rate but this method is very efficient in terms of the logic resources consumed.

A multi-channel filter can be realized using virtually the same amount of logic resources as a single channel version of the same filter. The trade-off here is between the logic resources and filter sample rate.

### 3.3.2 A Modified DA-based approach for the filter bank

This architecture is based on a novel architecture presented in [10]. Unlike in the conventional DA method where the input is distributed over the coefficients, in this case the coefficient matrix is distributed over the input. It is seen that in the previous architecture, as the input bit precision increases there is an exponential growth in the LUT size and this increases the amount of logic resources required. The advantage of the present architecture over the previous one is that, in this method we do not require any memory or LUT tables. This reduces the logic resources consumed tremendously.

Consider the following inner product equation 3.6(a) where c[n] represents the M-bit coefficients of an N-tap constant coefficient filter and x[n] represents the inputs.

$$y = \sum_{n=0}^{N-1} c[n].x[n] \qquad 3.6(a)$$

$$= \sum_{m=0}^{M-1} c_m[k]2^m \sum_{n=0}^{N-1} x[n] \qquad 3.6(b)$$

$$= \sum_{m=0}^{M-1} 2^m . \sum_{n=0}^{N-1} c_m[n].x[n] \qquad 3.6(c)$$

In equation 3.6(a) the coefficients can be replaced as in equation 3.6(b) where $c_m[k]$ denotes the $m^{th}$ bit of $k^{th}$ coefficient of c[n]. Rearranging equation 3.6(b) gives 3.6(c). The inner function, $\sum_{n=0}^{N-1} c_m[n].x[n]$ in 3.6(c) can be designed as a unique adder system based on the coefficient bits consisting of zeros and ones. The output, *y*, can then be computed by shifting and accumulating the results of the adder system accordingly based on the coefficient bit weight. Thus the whole equation can be implemented using just adders and shifters.

# CHAPTER 4

# IMPLEMENTATION OF DWT FILTER BANKS

## 4.1 Hardware Issues

### 4.1.1 Field Programmable Gate Arrays

Field Programmable Gate Arrays (FPGAs) are used to synthesize and test the architectures in this thesis. FPGAs are programmable logic devices made up of arrays of logic cells and routing channels. They have ASIC characteristics such as reduced size and power dissipation, high throughput, etc., with the added advantage that they are reprogrammable. Therefore, new features can be easily added and they can be used as a tool for comparing different architectures. Currently, Altera Corporation and Xilinx Corporation are the leading vendors of programmable devices. The architecture of the FPGAs is vendor specific.

Among the mid-density programmable devices, Altera's FLEX 10K and Xilinx XC4000 series of FPGAs are the most popular ones. They have attractive features which make them suitable for many DSP applications. FPGAs contain groups of programmable logic elements or basic cells. The programmable cells found in Altera's devices are called Logic Elements (LEs) while the programmable cells used in Xilinx's devices are called the Configurable Logic Blocks (CLBs). The typical design cycle for FPGAs using Computer Aided Design (CAD) tools is shown in figure in 4.1.

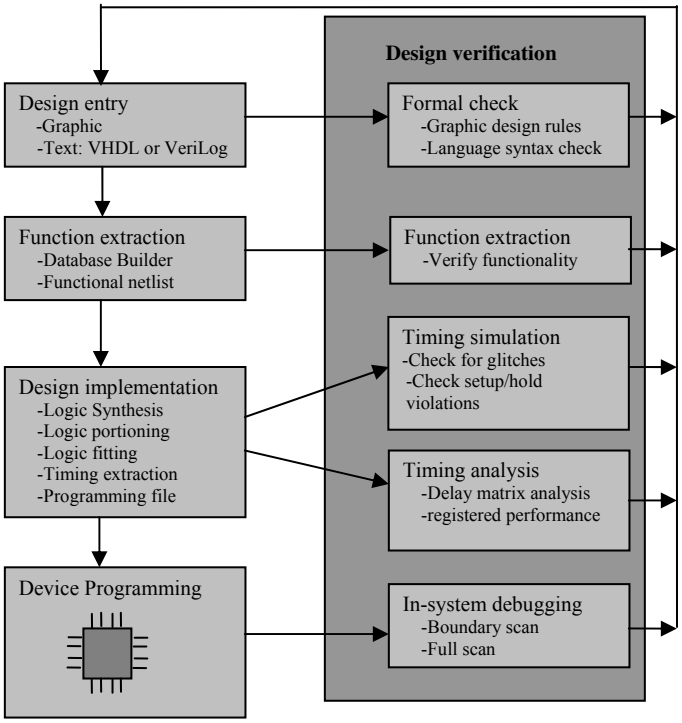



Figure 4.1 CAD design cycle [4].

The design is first entered using graphic entry or text entry. In the next stage the functionality of the design is extracted. Then the design is targeted on a selected device and its timing is extracted. Finally the actual hardware device is programmed. At every stage the appropriate verification is done to check the working of the design. For design entry, text is preferred as it allows more control over the design compared to graphic design entry.

### 4.1.2 The FLEX 10K Devices

The Wavelet filter banks are implemented on a device from the FLEX 10K family of Embedded Programmable Logic Devices provided by Altera. The Flexible Logic Element MatriX (FLEX) architecture provides good density, speed and all features necessary to implement an entire system in a single device. They are based on

Complementary Metal Oxide Semiconductor (CMOS) SRAM technology and thus are reset on power off. The block diagram of a Flex 10K device is shown in figure 4.2.

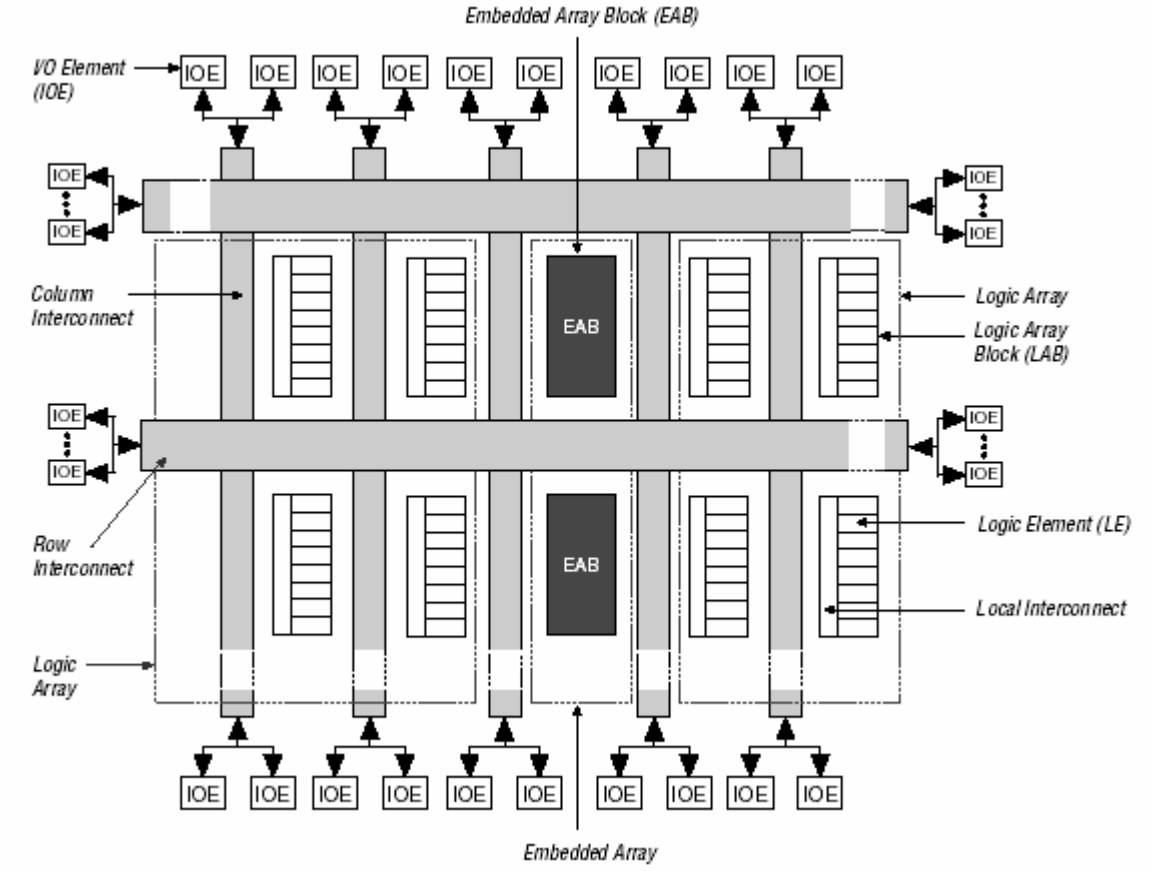



Figure 4.2 FLEX 10K device block diagram [12].

The FLEX 10K devices consist of a logic array to implement general logic functions and an embedded array for implementing efficient memory and specialized logic functions. The embedded array consists of a series of 2K-bit Embedded Array Blocks (EABs) which can be used independently or may be combined for larger functions. The logic array consists of Logic Array Blocks (LABs). Each LAB consists of eight LEs and a local interconnect. A FastTrack Interconnect is used to connect the LABs and EABs and the device pins. These are a series of fast channels that run the entire

length and width of the device. Input/Output Elements (IOEs) are located at the end of each row and column of the fast track Interconnect.

The LE is the smallest logic unit in the device. It consists of a four-input look-up table (LUT), a programmable flip-flop, and dedicated signal paths for carry and cascade functions. The carry and cascade chains provide dedicated high-speed data paths that connect all LEs in an LAB and all LABs in a row. The carry chains support high-speed counters and adders, while the cascade chains implement wide input functions with minimum delay. The LE can operate in different modes like the normal mode, arithmetic mode etc. Each mode uses the LE resources differently. For example, while the normal mode uses a four-input LUT, the arithmetic mode offers 2 three-input LUTs with additional fast carry that are ideal for implementing adders, accumulators, and comparators. Each LE can drive both the local interconnect and the Fast Track Interconnect.

**The FLEX10K30A Device**

The device chosen for implementation purpose is EPF10K30AQC240 with speed grade 1. It is a 3.3V device suitable for low power applications. Some of its features are listed in Table 4.1.

Table 4.1 Features of EPF10K30A devices.

| Feature | EPF10K30 |
|---|---|
| Typical gates (logic and RAM) | 30,000 |
| Logic Elements (LEs) | 1,728 |
| Logic Array blocks (LABs) | 216 |
| Embedded Array Blocks (EABs) | 6 |
| Total RAM bits | 12,288 |

### 4.1.3 Design Software

The implementations have been carried out using the software, MAX+plus II student edition version 10.2 provided by Altera Corporation. The hardware language used is the Very High Speed Integrated Circuit Hardware Description Language (VHDL).

VHDL is a widely used language for register transfer level description of hardware. It is used for design entry, simulation, and synthesis of digital systems.

## 4.2 Power Evaluation

Power constraints can be critical when designing a particular application. With the demand for low power applications, power estimation has become an integral part of the design process. Since most of the new portable applications such as streaming audio and video require high computational capacity, they have to be realized with very low power in order for the battery to have a satisfactory life span. Performing a power evaluation early in the design process can avoid complicated and expensive design changes later on if the power constraints are violated. Power estimation at a higher level can be used to accurately trade off power versus other design parameters such as area, performance, etc.

Many techniques can be incorporated to lower the power dissipation in an application. While lowering the supply voltage reduces the power dissipation to a large extent, other techniques at higher levels of design abstraction can achieve considerable improvement in power dissipation. At the architectural level, techniques such as pipelining and parallelism can be used to optimize area and power constraints. Also, techniques such as reducing the number of operations performed, reduce the switching capacitances, which in turn reduce the power consumption. Yet another method is operation substitution. Operations which require more computational energy should be replaced with operations that require less computational energy when possible. For example, in DSP applications, replacing the multiplication operations by additions saves both area and power.

Accurate power estimation involves estimating the average switched capacitances at the internal nodes of a circuit. Accurate estimation of switched capacitance at the architectural or behavioral level is quite difficult, while they can be more accurately estimated at the transistor circuit level [3].

Altera provides the following method for estimating the power consumption of an application implemented on the FLEX 10K devices:

$$\text{Estimated total power } P_{EST} = P_{INT} + P_{IO} \quad (4.1)$$

where $P_{INT}$ is the internal power consumption and $P_{IO}$ is the external power consumption of the device. Here, only the $P_{INT}$ is considered as it is design dependent. It is calculated as

$$P_{INT} = I_{CCINT} \times V_{CC} \quad (4.2)$$

where $V_{CC}$ is the supply voltage of the device and $I_{CCINT}$ is the total internal current which is given by

$$I_{CCINT} = I_{CCSTANDBY} + I_{CCACTIVE} \quad (4.3)$$

The standby current, $I_{CCSTANDBY}$ can be obtained from the device data sheet and is usually negligible for the CMOS devices. $I_{CCACTIVE}$ is the design dependent component. It depends on the switching frequency and the logic resources used. It can be calculated as

$$I_{CCACTIVE} = K \times f_{MAX} \times N \times tog_{LC} \quad (4.4)$$

where K is the $I_{CC}$ coefficient which can be obtained from the device data sheet, $f_{MAX}$ is the maximum operating frequency, N is the total number of logic cells used in the device and $tog_{LC}$ is the average percent of logic cells toggling at each clock which is typically 12.5%.

The above calculation provides an $I_{CC}$ estimate based on typical operating conditions with no output load [12]. Therefore, this method does not give the accurate power consumed when the application is actually run. But since the power estimation carried out here is only for the comparison of different designs, the above equations serve the purpose.

## 4.3 Daubechies Wavelet Filters

One of the major issues when using wavelet transforms, is choosing a suitable wavelet filter. The Daubechies family of wavelet filters, developed by Ingrid Daubechies, are perhaps the most popular wavelet filters due to their many desirable characteristics. They represent the foundations of wavelet signal processing and are used in numerous applications. They are also called Maxflat filters as the frequency responses of these filters have maximum flatness at frequencies 0 and $\pi$. Daubechies wavelets have the property of having the maximum number of vanishing moments for a given order which makes them suitable for compression applications. They are compactly supported orthogonal filters capable of achieving perfect reconstruction. Also, the Maxflat filters all have dyadic coefficients. Therefore, they can be easily represented as binary numbers for hardware implementation purposes.

In this chapter the implementation of the Daubechies 4-tap (db4) orthogonal filter bank and the Cohen-Daubechies-Feauveau (CDF) 9/7-tap biorthogonal filter bank which is often referred to as the Daubechies 9/7 (db9/7) filter bank will be discussed. Only the implementation of the analysis filter bank will be considered. The synthesis bank can be implemented following the same techniques.

The coefficients of the above filters are real numbers. For the sake of hardware implementation, these coefficients need to be quantized into binary digits which may result in loss of accuracy. The accuracy is measured in terms of the signal to noise ratio of the reconstructed signal. Therefore, the quantization should be done appropriately so that the quality of the wavelet transform is not compromised.

### 4.3.1 The Daubechies 4-tap orthogonal filter bank

This filter is widely used due to its orthogonal properties. It is also easy to implement as it is a short filter. It satisfies the perfect reconstruction conditions. The coefficients of the analysis filters are given in Table 4.2.

Table 4.2 Daubechies 4-tap filter coefficients.

| Tap | Low Pass Filter | High Pass Filter |
|---|---|---|
| 0 | 0.4830 | 0.1294 |
| 1 | 0.8365 | 0.2241 |
| 2 | 0.2241 | -0.8365 |
| 3 | -0.1294 | 0.4830 |

The floating point implementation of DWT consumes a lot of hardware resources making it very expensive. Another disadvantage is that the floating point operations are much slower than integer operations. Therefore, for ease of hardware implementation, the floating point coefficients are shifted by $2^8$ and the corresponding integer values are used. This gives us a good enough precision at 4 decimal places. Using integer operations allows fast computation and consumes less hardware which in turn results in less power consumption. At the end of computations, the output is again shifted by $2^{-8}$ to get the correct result.

### 4.3.2 The Daubechies 9/7-tap biorthogonal filter bank

The CDF Family of biorthogonal wavelets are perhaps the most widely used biorthogonal wavelets. These wavelets have symmetric scaling and wavelet functions, i.e., both the low pass and high pass filters are symmetric. The properties of these wavelets have made them very popular for image compression applications. For high compression ratios more zeros are needed which can be obtained by using longer filters. But, if the filter is too long, ringing occurs and this destroys the image quality. The Daubechies 9/7 filter bank has rational filter length and yields good performance. It is used in FBI Fingerprint Compression Standard and is the default filter for lossy compression in the JPEG 2000 standard. The coefficients for the analysis filter are given in Table 4.3.

As in the case of the db4 filter, for ease of hardware implementation, the floating point coefficients are shifted by $2^{13}$. This gives us a precision of up to 5 decimal places

which is quite good. At the end of computations, the output is again shifted $2^{-13}$ to get the correct results.

Table 4.3 Daubechies 9/7-tap filter coefficients.

| Taps | Low Pass Filter | Taps | High Pass Filter |
|---|---|---|---|
| 4 | 0.6029490183263579 | 3 | 1.1115087052456994 |
| 3,5 | 0.2668641184428723 | 2,4 | -0.5912717631142470 |
| 2,6 | -0.0782326652898785 | 1,5 | -0.05754352622849957 |
| 1,7 | -0.01686411844287495 | 0,6 | 0.09127176311424948 |
| 0,8 | 0.02674875741080976 | | |

## 4.4 Implementations and Results

### 4.4.1 Daubechies 4-tap orthogonal filter bank implementation

The Daubechies orthogonal wavelet filter bank is implemented using three different architectures. Section 4.4.1.1 presents the polyphase implementation of the filter bank. Section 4.4.1.2 presents the polyphase implementation using DA and section 4.4.1.3 presents polyphase implementation using the modified DA.

#### 4.4.1.1 Polyphase Implementation

For the polyphase implementation, each of the filters of the analysis bank is divided into even and odd phases to obtain the two tap filters as in Table 4.4.

Table 4.4 Polyphase filters for db4 filter bank.

| Taps | Low Pass Filters | | High Pass Filters | |
|---|---|---|---|---|
| | Even phase $G_{0even}$ | Odd phase $G_{0odd}$ | Even phase $H_{0even}$ | Odd phase $H_{0odd}$ |
| 0 | 0.4830 | 0.8365 | 0.1294 | 0.2241 |
| 1 | 0.2241 | -0.1294 | -0.8365 | 0.4830 |

The incoming input samples, x[n], are multiplexed into even samples($x_{even}$) and odd samples ($x_{odd}$). The even samples are convolved with the even filter coefficients and the odd samples with a delay are convolved with the odd filter coefficients and finally the even and odd phases are added. Low pass filter output represents the approximation or scaling coefficients and the high pass filter output represents the detail or wavelet coefficients of the DWT.

#### 4.4.1.2 Polyphase with DA Implementation

Distributed Arithmetic can be easily mapped into the look-up table (LUT) based architecture of the FPGAs. As observed in Table 4.4, in the polyphase implementation there are four two-tap filters. Each two-tap filter needs a four-input LUT as shown in Table 4.5. Since the high pass filter coefficients are mirror versions of the low pass coefficients, it is seen that, the low pass even coefficients are the same as the high pass odd coefficients. Similarly, the low pass odd coefficients are the same as the high pass even coefficients except for a change in sign. Therefore, the multiple channel technique described in the previous chapter can be applied here, and instead of four LUT tables for each of the filters, just two can be implemented which the filters can share. The speed is sacrificed here for fewer logic resources.

Table 4.5 DA-LUT table for a 2-tap filter.

| Address | Data |
|---|---|
| 00 | 0 |
| 01 | $C_0$ |
| 10 | $C_1$ |
| 11 | $C_0+C_1$ |

Also, for an 8-bit input precision, the serial distributed arithmetic (SDA) will give a very low throughput as discussed in the previous chapter, i.e., each output is obtained after 8 clock cycles which is not suitable for real-time applications. Therefore, a fully Parallel Distributed Arithmetic (PDA) architecture described in the previous chapter is implemented here to achieve a high-performance. All the 8 input bits are processed at a time and this requires eight identical LUT tables for each filter. In this case, the logic resources are traded for an increase in speed.

#### 4.4.1.3 Polyphase with modified DA Implementation

In this case, adder trees formed by the filter coefficients are used instead of LUTs. The input samples are passed through a delay line as shown in figure 4.3(a). This allows the inputs to be available to the adder tree in a parallel manner.

Since the filter coefficients are the same for the low pass even filter and the high pass odd filter, they have the same adder tree structure as shown in figure 4.3(b). The inputs $x_n$ and $x_{n+2}$ represent $x_0$ and $x_2$ for the low pass even filter computation and $x_1$ and $x_3$ for the high pass odd filter computation. The tree is formed using the 8-bit precision filter coefficients given in Table 4.6. The adder tree requires only a single adder. Also, the low pass odd filter and the high pass even filter have the same coefficients as given in Table 4.6, these filters do not require an adder tree.

Table 4.6 Binary coefficients for the db4 filter bank.

| Taps | Low pass even filter and High pass odd filter | Low pass odd filter and High pass even filter |
|---|---|---|
| 0 | 0 1 1 1 1 1 0 0 | 1 1 0 1 0 1 1 0 |
| 1 | 0 0 1 1 1 0 0 1 | 0 0 1 0 0 0 0 1 |

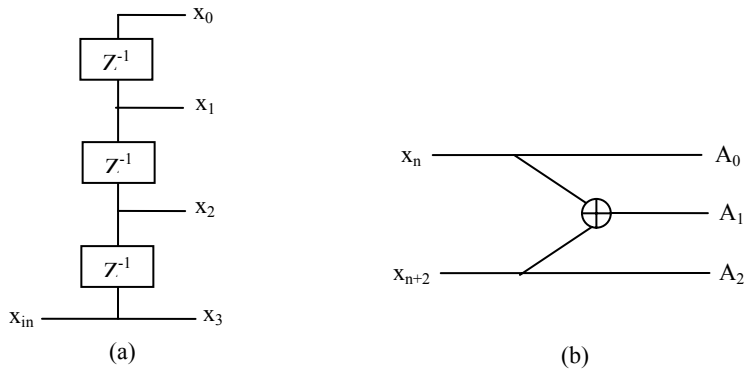



Figure 4.3 Polyphase implementation of db4 with modified DA (a) delay line and (b) adder tree.

The outputs from the adder tree are then scaled according to the respective coefficient bit-weight and then added to get the final filter outputs. A pipelined adder as shown in figure 4.4 can be used for this operation. Finally, the even and odd phases are added to obtain the final outputs. Thus the whole design is implemented with a minimal number of shift and add operations.

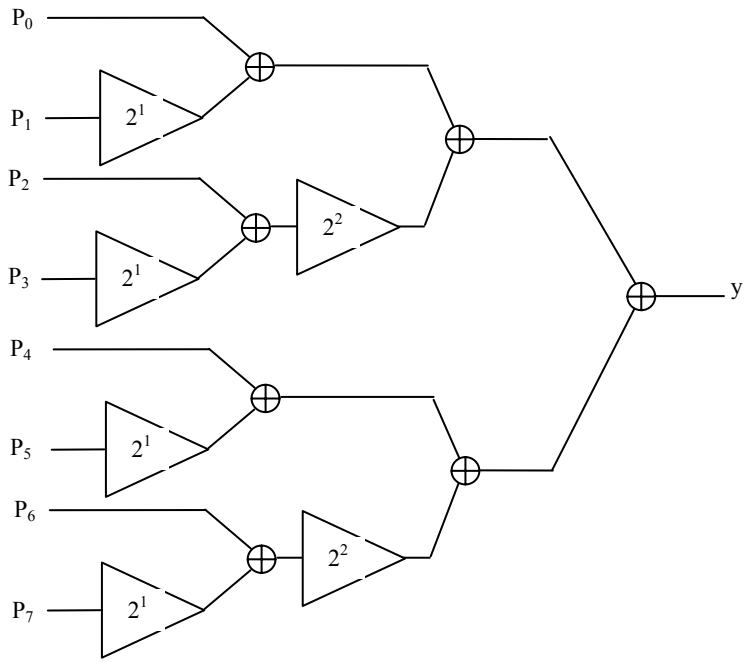
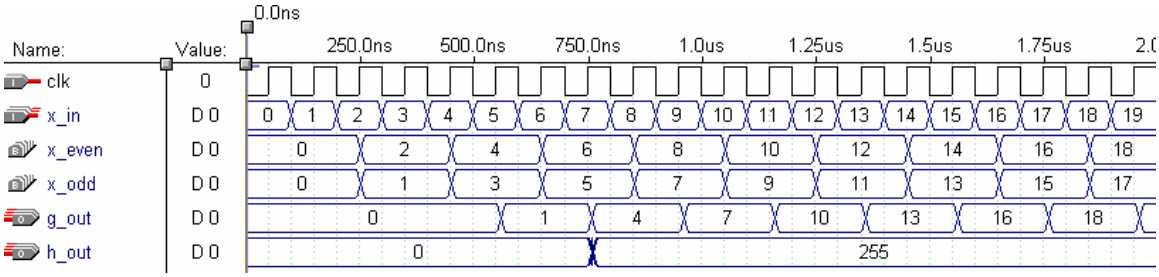


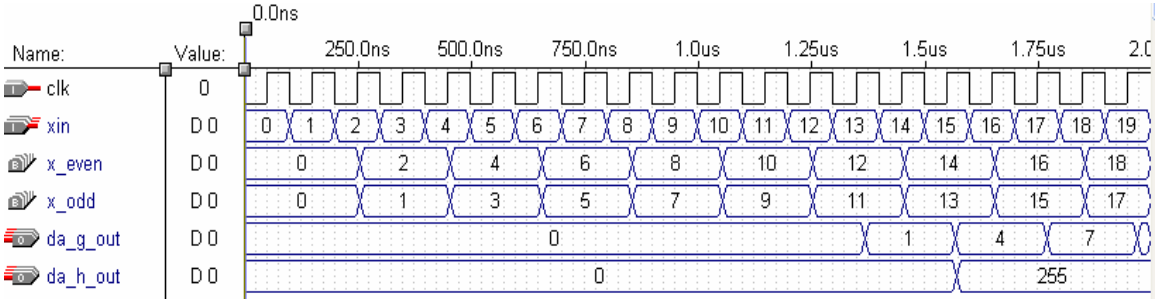

Figure 4.4 Pipelined adder-shifter.

#### 4.4.1.4 Results

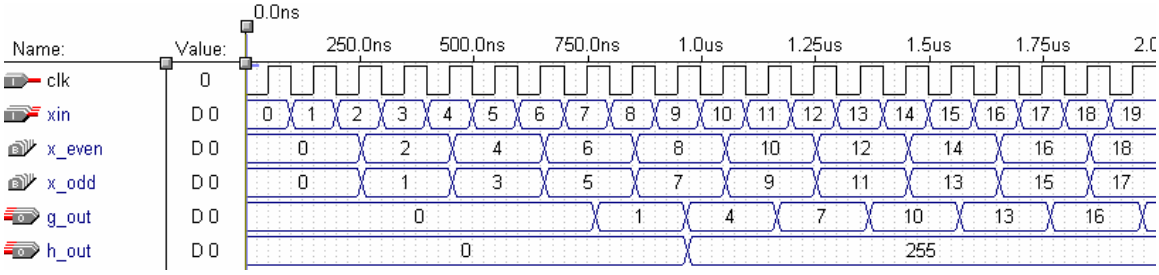
All the architectures were synthesized using the FAST synthesis style with speed 10 provided by the MAX+plus II software. The simulation waveforms generated by the MAX+plus II simulator are used to verify the functionality of the design. Figure 4.5(a) shows the simulation results for polyphase implementation, figure 4.5(b) shows the results for polyphase with DA implementation, and figure 4.5(c) shows the simulation results for polyphase with modified DA. In all cases, 8-bit samples were used as input. Also, the output signals were scaled to have the same number of bits as the input.




(a)




(b)




(c)

Figure 4.5 Simulation results of the db4 filter bank.

In the above figure, the inputs, clk and xin are the clock signal and the input samples respectively. x_even and x_odd are the even and odd samples which are obtained by multiplexing the incoming input samples. The outputs, g_out and h_out are the outputs of the low pass and high pass filters respectively. g_out represents the approximation or scaling coefficients and h_out represents the detail or wavelet coefficients. All inputs and outputs are signed 8-bit numbers.

The hardware resources used for the design can be obtained from the report file generated by the software. Figure 4.6 shows the comparisons of the number of logic cells (LCs) used by the different architectures, where A1 corresponds to the polyphase implementation, A2 corresponds to polyphase implementation with DA, and A3 corresponds to polyphase implementation with modified DA.

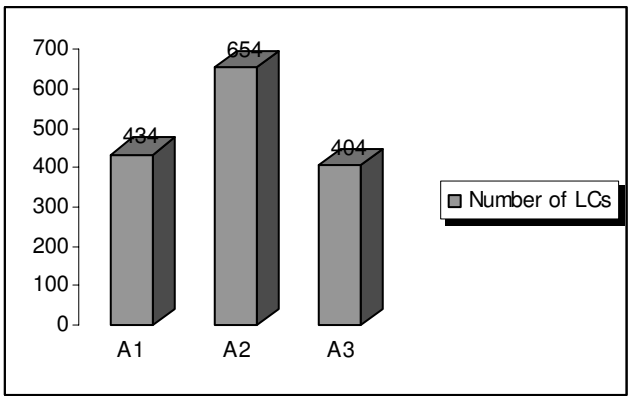



Figure 4.6 Comparison of hardware requirements for the db4 filter banks.

The results show that A1 consumes 434 logic cells, which corresponds to 25% of the total available logic cells in FLEX 10K30A device. A2 consumes 654 logic cells which is 37% of the total and A3 consumes 404 LCs which is 23% of the total LCs. A2, which is the polyphase implementation with DA consumes more hardware as it uses a fully parallel implementation of the DA architecture. It would consume much less resources if it were not a parallel implementation but this would have decreased the throughput. Thus, if area is a constraint, A3, which is the polyphase implementation with modified DA would be a good choice.

The performance of the design can be obtained using the timing analyzer provided by MAX+plus II. Figure 4.7 shows the maximum operating frequencies of the different architectures. As observed in the figure the polyphase with DA implementation is the

fastest among the three. Therefore, for high speed designs the fully parallel DA architecture should be used.

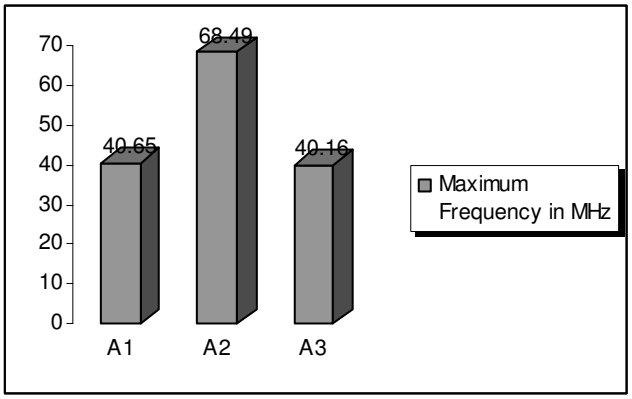



Figure 4.7 Comparison of performance for the db4 filter banks.

The power consumption can be calculated using the method presented in section 4.2. The EPF10K30A is a low power device suitable for mobile applications. Its supply voltage $V_{CC}$ is 3.3V, the standby current $I_{CCSTANDBY}$ is 0.3 mA and its $I_{CC}$ coefficient K is 17. The average ratio of logic cells toggling at each clock, $tog_{LC}$, is taken to be the typical value of 0.125. The maximum frequency $f_{MAX}$ and the number of logic cells N are obtained from the above results for the different architectures.

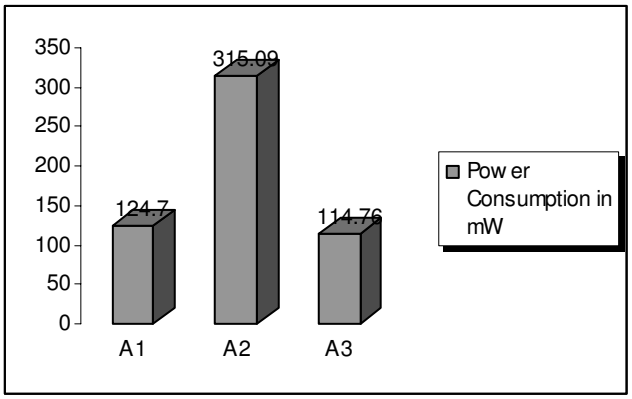



Figure 4.8 Comparison of power consumption for the db4 filter banks.

The calculated power consumption of the architectures is shown in figure 4.8. It is seen that A3 consumes the least power. Thus, for low area and power, the polyphase with modified DA architecture is the preferred choice, and for high speed, the polyphase with parallel DA architecture is a good choice.

### 4.4.2 The Daubechies 9/7-tap biorthogonal filter bank

The Daubechies biorthogonal 9/7-tap wavelet filter bank is implemented using the polyphase architecture and the polyphase with modified DA architecture. Section 4.4.2.1 presents the polyphase implementation and section 4.4.2.2 presents polyphase implementation with modified DA architecture. The conventional DA architecture is not implemented as it requires considerable hardware resources. Due to the increased filter length, the size of the LUT tables is very large which makes the design unfeasible.

#### 4.4.2.1 Polyphase Implementation

For the polyphase implementation, each of the filters of the analysis bank is divided into even and odd phases, as shown in Table 4.7.

Table 4.7 Polyphase filters for db9/7 filter bank.

| Low Pass Filters | | | | High Pass Filters | | | |
|---|---|---|---|---|---|---|---|
| Taps | Even phase $G_{0even}$ | Taps | Odd phase $G_{0odd}$ | Taps | Even phase $H_{0even}$ | Taps | Odd phase $H_{0odd}$ |
| 4 | 0.60294 | 3,5 | 0.26686 | 2,4 | 1.11150 | 3 | -0.59127 |
| 2,6 | -0.07822 | 1,7 | -0.01686 | 0,6 | -0.05754 | 1,5 | 0.09127 |
| 0,8 | 0.02674 | | | | | | |

First, the incoming input samples, x[n], are multiplexed into even samples ($x_{even}$) and odd samples ($x_{odd}$). The even samples are convolved with the even filter coefficients and the odd samples with a delay are convolved with the odd filter coefficients, and finally the even and odd phases are added.

Taking advantage of the symmetric coefficients, the inputs which are to be multiplied with the same coefficients can be first added before multiplying with the coefficients, this will decrease the number of multiplications by approximately half. Thus, the above architecture takes advantage of both decimation and linear phase property of the filter. While using polyphase technique reduces the number of multiplication operations by half, the symmetry property reduces the multiplication operations by half again.

#### 4.4.2.2 Polyphase with modified DA Implementation

The above polyphase architecture can be made even more efficient by incorporating DA into it. First, the input samples are passed through a delay line as shown in figure 4.9(a) so that they are available in a parallel fashion to the adder trees. The adder tree for the low pass even phase filter is shown in figure 4.9(b), while the low pass odd phase filter is shown in figure 4.9(c), high pass even phase filter is shown in figure 4.9(d), and high pass odd phase filter is shown in figure 4.9(e). They are formed using the following 14 bit precision coefficients of the respective filters provided in Table 4.8.

Table 4.8 Binary coefficients for the db9/7 filter bank.

(a) low pass filter

| Taps | Even Phase Filter | Taps | Odd Phase Filter |
|---|---|---|---|
| 4 | 0 0 0 0 0 0 1 1 0 1 1 0 1 1 | 3,5 | 0 0 0 0 0 0 1 0 0 0 1 0 1 0 |
| 2,6 | 0 0 0 0 1 0 1 0 0 0 0 0 0 0 | 1,7 | 0 0 1 0 0 0 1 0 0 0 1 0 1 0 |
| 0,8 | 0 1 0 0 1 1 0 1 0 0 1 0 1 1 | | |

(b) high pass filter

| Taps | Even Phase Filter | Taps | Odd Phase Filter |
|---|---|---|---|
| 2,4 | 0 0 0 0 1 0 1 1 1 0 1 0 1 1 | 3 | 0 0 0 0 0 1 1 1 0 1 0 1 1 1 |
| 0,6 | 0 1 0 0 1 0 1 1 1 0 1 0 1 1 | 1,5 | 1 0 0 0 1 1 1 0 1 0 1 1 1 0 |

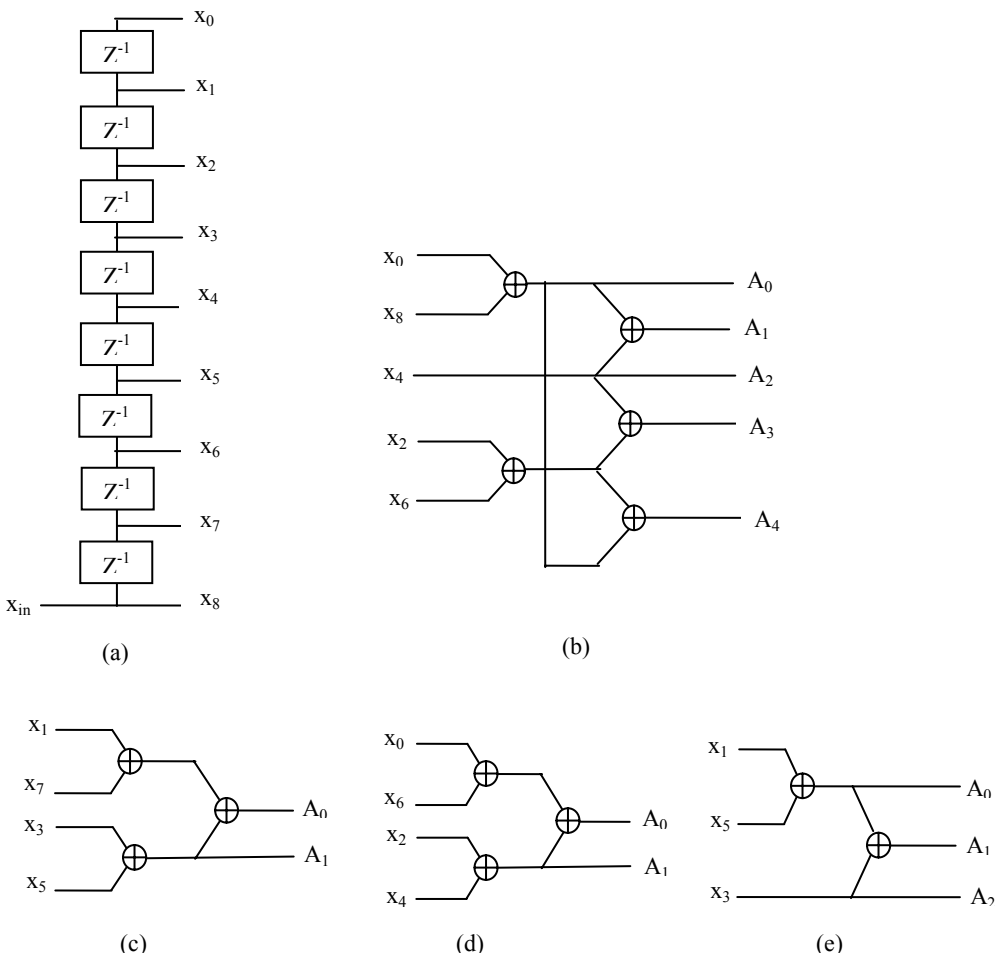


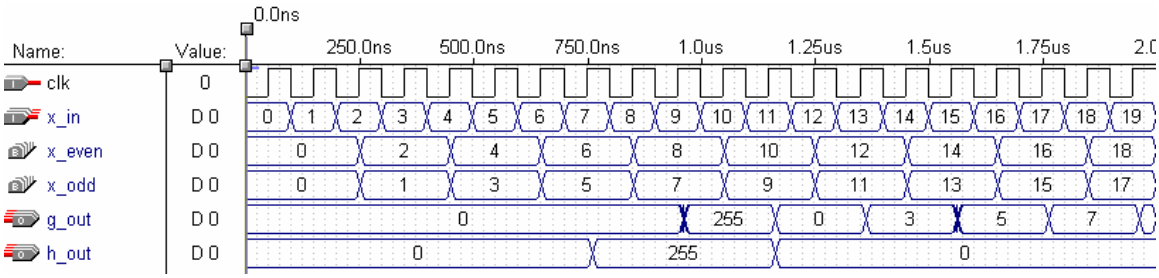

Figure 4.9 Polyphase implementation of db9/7 with modified DA (a) delay line (b) and (c) low pass even and odd adder trees (d) and (e) high pass even and odd adder trees respectively.

The outputs from the adder tree are now scaled according to their weights and then added to get the filter outputs. A pipelined adder-shifter similar to the one shown in figure 4.4 for the 8-bit coefficients can be used for the 14-bit coefficients to obtain the filter outputs. Again, the even and odd phases are added to get the final low pass and high pass filter outputs.

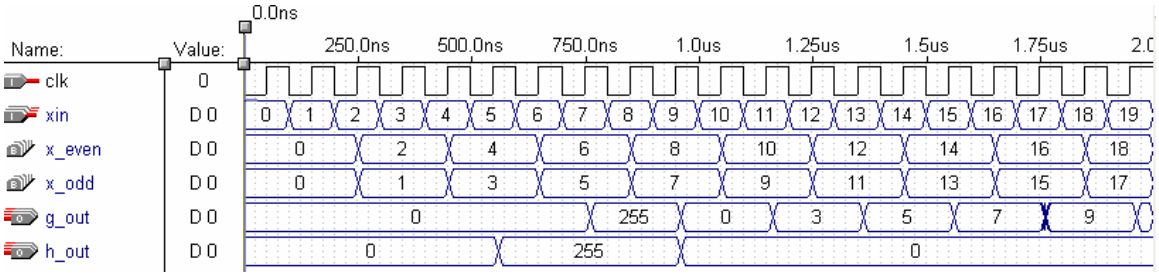
#### 4.4.2.3 Results.

The simulation waveforms generated by the MAX+plus II simulator verify the correct functioning of the design. Figure 4.10(a) shows the simulation results for polyphase implementation, figure 4.10(b) shows the simulation results for polyphase with

modified DA implementation. In all cases, 8-bit signed input samples were used. Also, the output signals were scaled to have the same number of bits as the input.




(a)




(b)

Figure 4.10 Simulation results of the db9/7 filter bank

The hardware requirements, in terms of the number of LCs utilized by the architectures, is provided in figure 4.11, where B1 corresponds to polyphase implementation and B2 corresponds to polyphase implementation with modified DA.

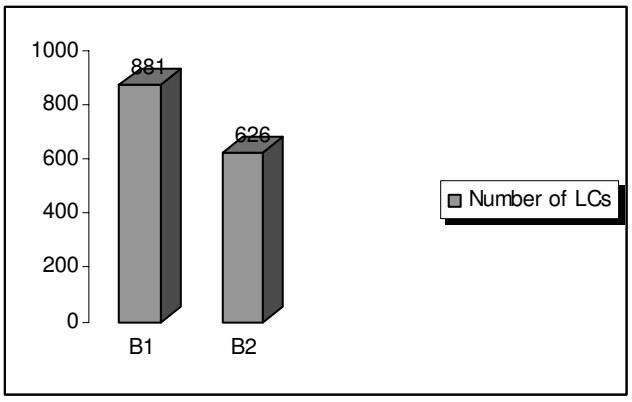



Figure 4.11 Comparison of hardware requirements for the db9/7 filter banks.

The results show that B1 consumes 809 logic cells, which corresponds to 46% of the total available logic cells in FLEX 10K30A device. B2 consumes 626 logic cells which is 36% of the total LCs. Thus, for an area efficient design, B2, which is the polyphase implementation with modified DA, is most suitable.

The performance of the architectures in terms of the maximum frequency using the timing analyzer is shown in figure 4.12. As observed, B2 has slightly lower performance than B1.

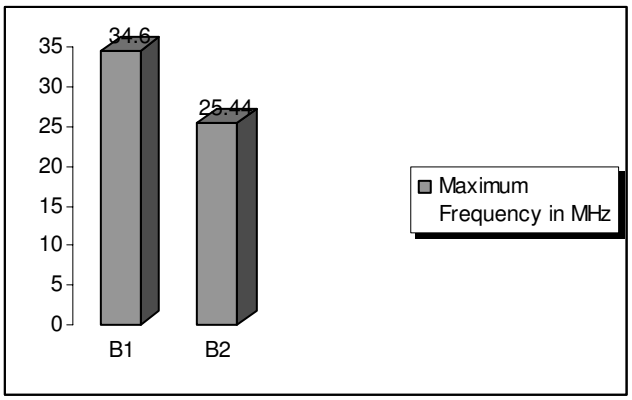



Figure 4.12 Comparison of performance for the db9/7 filter banks.

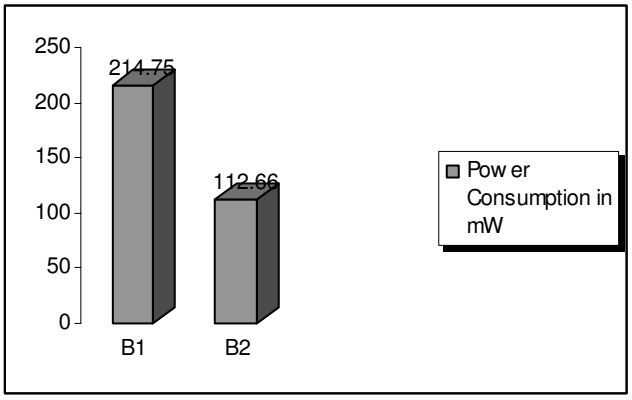



Figure 4.13 Comparison of power consumption for the db9/7 filter banks.

The estimated power consumption of the two architectures is shown in figure 4.13. It is observed that B2 consumes 23.91mW while B1 consumes 33.7mW. Therefore, B2 is more power efficient than B1. By overall comparison, it is seen that B2, i.e., the polyphase with modified DA architecture, is the better of the two, as it consumes much less area and power with a slightly lower performance than B1.

# CHAPTER 5

# IMPLEMENTATION OF HIGHER OCTAVE DWT

## 5.1 Introduction

Real-time applications of DWT such as streaming video require high performance computations. Due to computational intensity of DWT, dedicated circuits are needed for the computation of DWT. A number of architectures have been proposed recently for the implementation of 1-dimensional and 2-dimensional DWT. Each one has its advantages and drawbacks. Serial architectures are area efficient, but they have low throughput. In folded architectures, all computations of DWT at different levels are folded to the same low pass and high pass filters. This achieves a low latency, but its drawbacks are, increased hardware area with complex routing and control network.

An efficient DWT architecture should be such that it is easily scalable for different levels of wavelet decompositions and filter lengths. In this chapter, a simple and scalable architecture for computation of higher octave one dimensional DWT is described.

## 5.2 Scalable Architecture for Higher Level DWT

The block diagram of the proposed scalable architecture for higher octave DWT is shown in figure 5.1. The architecture shown here is only up to three levels; but, it can easily be scaled for higher octaves. The main unit in the architecture is the Processing Element (PE). In our case, each PE consists of an analysis filter bank. Therefore, the PE architecture depends on the chosen filter and the selected architecture of the filter bank.

For simplicity, the Daubechies length-4 filter has been used to implement the proposed architecture. The polyphase architecture of the filter bank implemented in the previous chapter is selected as it offers a good compromise among area, performance and power consumption. Furthermore, the polyphase architecture has low latency.

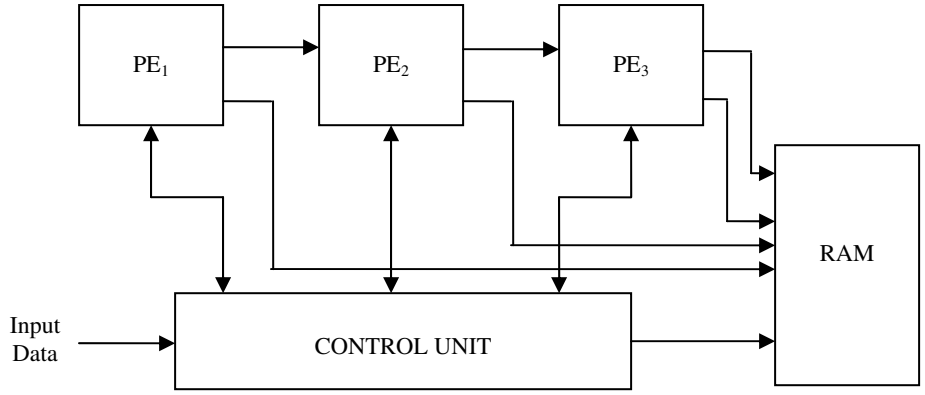



Figure 5.1 Block diagram of higher level DWT the architecture.

The input signal is first sent to the first PE, which produces the first level of approximation and detail coefficients. The approximation coefficients or the scaling coefficients, which are the outputs of the low pass filter, are then fed to the next PE for further processing. The detail coefficients or the wavelet coefficients which are the outputs of the high pass filter are stored in a RAM. The control unit provides communication between different PEs as well as loads wavelet coefficients into the RAM as they are being generated by the PEs. In the last stage, both approximation and detail coefficients are stored in the RAM. The control unit also provides clock signals to all the other units for synchronization purposes.

The above architecture enables parallel processing. The PEs can operate simultaneously thus enabling a fast computation of DWT. A more detailed description of the architecture along with the control signals for $PE_1$ is shown in figure 5.2.

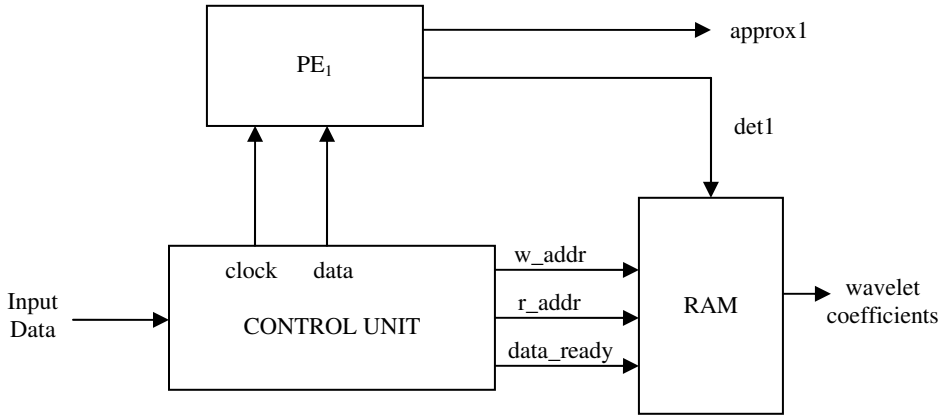



Figure 5.2 Schematic with control signals

The control unit sends the input data and the clock signal to $PE_1$. $PE_1$ receives the inputs, performs the filtering operation, and generates the first level of approximation and detail coefficients. As soon as the first coefficient is generated, the control unit enables the data ready signal which indicates the data is ready to be written into the RAM. The control unit then enables the write enable signal and sends the write address to the RAM so that the wavelet coefficients are written in the appropriate location in the RAM. Simultaneously, $PE_2$ processes the approximation coefficients as they are generated to produce the second level of coefficients. Similarly, as the second level coefficients are available, $PE_3$ starts processing them and the third level coefficients are produced. Therefore, while the first level wavelet coefficients are being written into the RAM, the next higher levels of wavelet coefficients are also being generated. Thus, the control unit has to perform appropriate scheduling, in order to store the wavelet coefficients from all levels in the memory.

The scheduling gets even more complicated due to the fact that the wavelet coefficients have to be stored such that the last level of coefficients appear first, followed by the previous level and so on. Therefore, the control unit schedules the write address of the RAM such that the third stage approximation coefficients followed by the detail coefficients are stored in the first available memory locations in the RAM. The second

stage detail coefficients are stored in the next address locations followed by the first stage detail coefficients. Therefore, when the wavelet coefficients are read back from the RAM, they are available in the correct order and can be used directly for further processing and compression.

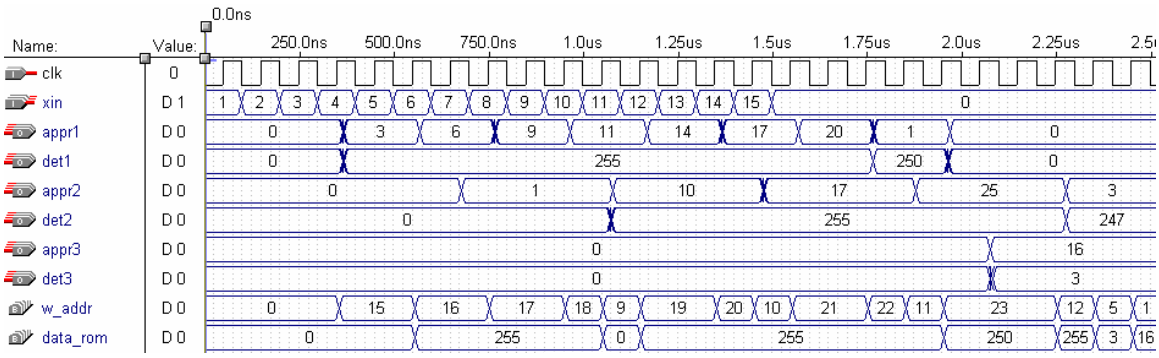
## 5.3 Implementation and Results

The Altera device EPF10K70RC240 with speed grade 2 is chosen for implementation purpose so that the whole design can fit into one device. It is a 5V device and some of its features are listed in Table 5.1.

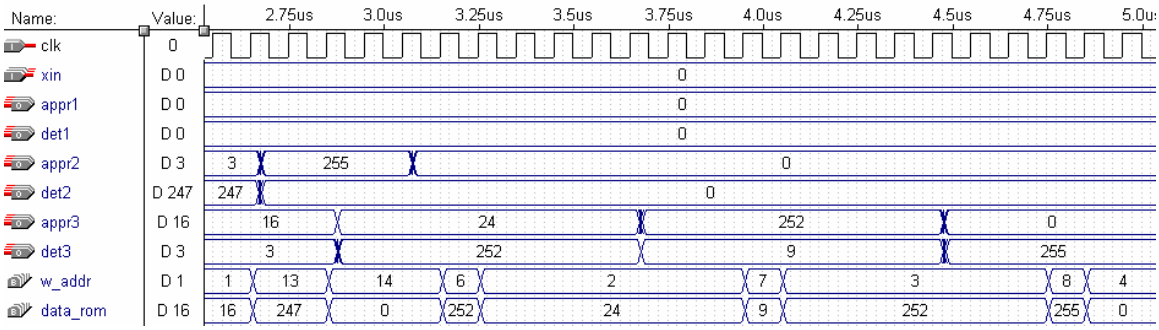
Table 5.1 Features of EPF10K70 devices.

| Feature | EPF10K70 |
|---|---|
| Typical gates (logic and RAM) | 70,000 |
| Logic Elements (LEs) | 3,744 |
| Logic Array blocks (LABs) | 468 |
| Embedded Array Blocks (EABs) | 9 |
| Total RAM bits | 18,432 |

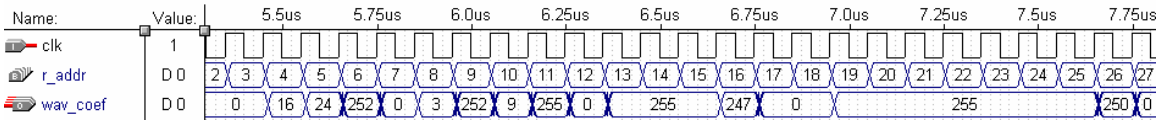
The architecture is implemented for an input signal of 15 samples using the orthogonal Daubechies length-4 filter. The simulation waveforms generated by the MAX+plus II simulator verify the functionality of the design. Figure 5.3 shows the simulation results of the implemented architecture. Input samples of 8-bit precision are used. The coefficients at every level are scaled to have the same number of bits as the input. This allows the use of the same PEs for different levels of computation of the DWT. Thus, the architecture is modular and is easily scalable to obtain higher level of octaves.




(a)




(b)




(c)

Figure 5.3 Simulation results of the 3-level DWT architecture.

The input samples enter $PE_1$ at a rate of one sample per clock cycle. The outputs appr1 and det1 which are the approximation and detail coefficients, respectively, are generated at the rate of one sample per two clock cycles. $PE_2$ receives the appr1 coefficients and produces the next level of coefficients, appr2 and det2, at the rate of one sample per four clock cycles. $PE_3$ receives the appr2 coefficients and produces the outputs appr3 and det3 at the rate of one sample per eight clock cycles. The output rate keeps decreasing as DWT level goes higher due to the decimation of the signal at each stage. The decreasing rate helps the control unit to interleave the write cycles of the RAM, so that the detail coefficients produced from different PEs can be written to the

same RAM. Figures 5.3(a) and 5.3(b) show the outputs produced at each stage and the loading of the coefficients in the appropriate locations of the RAM. Figure 5.3(c) shows the final required DWT coefficients as they are read from the RAM.

The hardware resources required for the implementation can be derived from the report file generated by Max+plus II software. The number of logic cells (LCs) used was found to be 2794, which corresponds to 74% of the total LCs available in the device. The maximum operating frequency was found to be 20.83 MHz. The power consumption calculated was 3094.32mW. The supply voltage, $V_{CC}$, of the EPF10K70 device is 5V, the standby current, $I_{CCSTANDBY}$ is 0.5 mA and its $I_{CC}$ coefficient, K is 85. The average ratio of logic cells toggling at each clock, $tog_{LC}$, is taken to be the typical value of 0.125.

## 5.4 Discussion

Applications such as image compression require signal extension prior to filtering at every stage of DWT computation. There are three types of signal extension; zero padding or extension by zeros, wraparound or periodic extension, and symmetric extension or extension by reflection. It is found that symmetric extension provides a better signal-to-noise ratio (SNR) value among the three after reconstruction as it reduces distortion at boundaries of reconstructed image.

The above proposed architecture assumes zero padding, i.e., the first coefficient of a level is computed using only the first coefficient of the previous level. The advantage of this architecture is low latency, small area, low memory requirements and low power consumption. However, the SNR of the reconstructed signal is compromised. If symmetric extension is chosen, it provides better SNR, but it has higher latency, requires large area and memory, has more complexity due to extra logic, and has higher power consumption. Thus, the trade-off has to be decided based on the application. The proposed architecture is more suitable for applications such as pattern recognition which do not require reconstruction of the signal.

# CHAPTER 6

# CONCLUSIONS AND FUTURE WORK

## 6.1 Conclusions

The Discrete Wavelet Transform provides a multiresolution representation of signals. The transform can be implemented using filter banks. In this thesis, different architectures for the Discrete Wavelet Transform have been implemented. For each of them, parameters such as area, performance and power consumption were discussed. Based on the application and the constraints imposed, the appropriate architecture can be chosen. For the Daubechies length-4 orthogonal filter, three architectures were implemented, i.e., the polyphase architecture, the polyphase with fully parallel DA architecture, and the polyphase with modified DA architecture. It is seen that, in applications which require low area and power consumption, e.g., in mobile applications, the polyphase with modified DA architecture is most suitable and for applications which require high throughput, e.g., real-time applications, the polyphase with DA architecture is more suitable.

The biorthogonal wavelets, with different number of coefficients in the low pass and high pass filters, increases the number of operations and the complexity of the design, but they have better SNR than the orthogonal filters. For the Daubechies 9/7 biorthogonal filter, two different architectures were implemented, i.e., the polyphase architecture, and the polyphase with modified DA architecture. It is seen that the polyphase architecture has better throughput while the polyphase with modified DA architecture has lower area and lower power consumption.

A scalable architecture for computation of higher octave DWT has been presented. The architecture was implemented using the Daubechies length-4 filter for a signal length of 15. The simulation results verify the functionality of the design. The proper scheduling of the wavelet coefficients written to the RAM ensures that, when the coefficients are finally read back from the RAM, they are available in the required order for further processing. The proposed architecture is simple since further levels of decomposition can be achieved using identical processing elements. It is easily scalable to different signal lengths and filter orders for use in different applications. The architecture enables fast computation of DWT with parallel processing. It has low memory requirements and consumes low power.

## 6.2 Future Work

Synthesis filter banks to compute the inverse DWT, i.e., IDWT can be implemented using similar architectures for the corresponding analysis filter banks.

The architectures of the filter banks can be further improved using techniques such as Reduced Adder Graph, Canonic Signed Digit coding and Hartley's common subexpression sharing among the constant coefficients. Also, in the case of orthogonal filters with mirror coefficients, the transpose form of the filters yields a good architecture; this can be implemented and compared with the others.

The proposed higher octave DWT architecture can be extended to include symmetric signal extension. The use of symmetric extension in image compression applications reduces the distortion at boundaries of reconstructed image and provides improved SNR.

In memory intensive applications such as image and video processing, memory accesses could be the dominant source of power dissipation, as reading and writing to memory involves switching of highly capacitive address busses. Methods such as gray

code addressing can be incorporated into the architecture to reduce this power dissipation.

As the DWT hierarchy increases, the required precision of the wavelet coefficients also increases. In the proposed architecture, the coefficients at all levels are scaled to have the same precision. While this reduces the hardware requirements, the accuracy of the coefficients is compromised as the number of levels increases. Therefore, the architecture can be modified to allow increased precision as the DWT level increases so as to achieve higher accuracy.

The proposed architecture can also be extended to 2-dimensional DWT computation. This can be achieved by computing the 1-dimensional DWT along the rows and columns separately. This operation requires large amount of memory and involves extensive control circuitry.

# BIBLIOGRAPHY


[1] Gilbert Strang and Truong Nguyen, *Wavelets and Filter Banks*, Wellesley-Cambridge Press, 1997.

[2] C. Sydney Burrus, Ramesh A. Gopinath, Haitao Guo, *Introduction to Wavelets and Wavelet Transforms: A Primer*, Prentice Hall, 1997.

[3] Kaushik Roy, Sharat C. Prasad, *Low-Power COMS VLSI Circuit Design*, John Wiley and Sons, Inc., 2000.

[4] Uwe Meyer-Baese, *Digital Signal Processing with Field Programmable Gate Arrays*, Springer-Verlag, 2001.

[5] http://engineering.rowan.edu/~polikar/WAVELETS/WTtutorial.html ; the Wavelet Tutorial by Robi Polikar.

[6] Robert D. Turney, Chris Dick, and Ali M. Reza, *Multirate Filters and Wavelets: From Theory to Implementation*, Xilinx Inc.

[7] V. Spiliotopoulos, N.D. Zervas, C.E. Androulidakis, G. Anagnostopoulos, S. Theoharis, *Quantizing the 9/7Daubechies Filter Coefficients for 2D DWT VLSI Implementations*, 14th International Conference on Digital Signal Processing, pages 227 - 231, vol.1, July 2002.

[8] J.Ramirez, A. Garcia, U. Meyer-Baese, F. Taylor, P.G. Fernendez, A. Lloris, *Design of RNS-Based Distributed Arithmetic DWT Filterbanks*, Proceedings of 2001 IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP), pages 1193 -1196, vol.2, May 2001.

[9] Xilinx Incorporation, *The Role of Distributed Arithmetic in FPGA-based Signal Processing*, Xilinx application notes, San Jose, CA.

[10] M. Alam, C.A. Rahman, W. Badawy, G. Jullien, *Efficient Distributed Arithmetic Based DWT Architecture for Multimedia Applications*, Proceedings of the 3rd IEEE International Workshop on System-on-Chip for Real-Time Applications, pages 333 -336, June 2003.

[11] M.Nibouche, A.Bouridane and O.Nibouche, *A Framework for A Wavelet-Based High Level Environment*, the 8th IEEE International Conference on Electronics, Circuits and Systems (ICECS), pages 429 -432, vol.1, Sept. 2001.

[12] Altera Corporation,"*FLEX 10K Embedded Programmable Logic Device Family Data Sheet*".

[13] Altera Corporation, "*Evaluating Power for Altera Devices*", Application Note 74, version 3.1.

[14] Philip P. Dang and Paul M. Chau, *Integer Fast Wavelet Transform and its VLSI Implementation for Low Power Applications*, IEEE Workshop on Signal Processing Systems, (SIPS), pages 93 -98, Oct. 2002.

[15] Ali M. Al-Haj, *Fast Discrete Wavelet Transformation Using FPGAs and Distributed Arithmetic*, International Journal of Applied Science and Engineering, 2003.

[16] Keshab K.Parhi and Takao Nisitani, *VLSI Architectures for Discrete Wavelet Transforms*, IEEE Transactions on VLSI Systems, pages 191 -202, vol. 1, No. 2, June 1993.

[17] Maire Francois, *Low Power Implementation for a Class of Orthogonal Wavelet Transform using Synthesizable VHDL*, M.S. thesis, Stockholm, September 1999.

[18] Sung Bum Pan and Rae-Hong Park, *New Systolic Arrays for Computation of the 1-D Discrete Wavelet Transform*, IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP), pages 4113 -4116, vol.5, April 1997.

[19] Tay-Jyi Lin, Chein-Wei Jen, *An Efficient 2-D DWT Architecture via Resource Cycling*, IEEE International Symposium on Circuits and Systems (ISCAS), pages 914 - 917, vol. 4, May 2001.

[20] M.Nibouche, A.Bouridane and O.Nibouche, *Design and FPGA Implementation of Biorthogonal Discrete Wavelet Transforms*, International Conference on trends in Communications, EUROCON'2001, pages 103 -106, vol.1, July 2001.

[21] J.Bhasker, *A VHDL Synthesis Primer*, Second edition, Star Galaxy Publishing, 2001.

# BIOGRAPHICAL SKETCH

Deepika Sripathi was born on November 1st, 1978 in Hyderabad, India. She completed her undergraduate studies in June 2001 and received her B.E. degree in Electronics and Communication Engineering from Osmania University in Hyderabad, India. She was admitted to Master's program at Florida State University in the department of Electrical and Computer Engineering in August 2001. Her areas of interest include Digital IC and ASIC design. She graduated from Florida State University in December 2003.