ON SOME GENERALIZED TRANSFORMS FOR SIGNAL DECOMPOSITION AND RECONSTRUCTION

YUMNAM KIRANI SINGH

COMPUTER VISION AND PATTERN RECOGNITION UNIT
INDIAN STATISITICAL INSTITUTE
203 B.T ROAD
KOLKATA-700 108
INDIA

A THESIS SUBMITED TO
THE INDIAN STATISTICAL INSTITUTE
IN PARTIAL FULFILMENTOF THE REQUIREMENTS FOR
THE DEGREE
OF
DOCTOR OF PHILOSOPHY
January, 2004

ACKNOWLEDGEMENTS

First of all, I am grateful to Prof. Swapan Kumar Parui for his guidance and help towards the completion of this thesis. I appreciate Prof. B.B. Chaudhury, Head of Computer Vision and Pattern Recognition Unit for his benevolence and encouraging nature during my Ph. D. work. I also like to thank Utpal Garain for his comments and suggestions on some parts of a draft of the thesis. Lastly, I would like to thank Shuvransu Banerjee and other members of the Unit for their help and co-operation.

Date: 28th January, 2004 Yumnam Kirani Singh

Dedicated to

My

Father, (Late) Yumnam Bino Singh

Mother, Yumnam Brajarani Devi

And

Eldest Brother, Yumnam Arun Singh

Contents

3. ISITRA FOR SIGNAL DECOMPOSITION AND RECONSTRUCTION

List of Figures

List of Tables

Chapter 1

INTRODUCTION

Abstract: *The main theme of the present thesis is the development of some generalized transforms for signal decomposition and reconstruction. Broadly two such transforms, namely, ISITRA and YKSK transform, are proposed here. They are further classified as ISITRA-1, ISITRA-2, YKSK-1 and YKSK-2 transforms. Signal decomposition here is based on Bino's model of multiplication, which gives a generalized model for number multiplication, polynomial multiplication and signal convolution. The transforms proposed here are, therefore, convolution based transforms. Both the decomposition and reconstruction of ISITRA is based on convolution whereas the decomposition of YKSK transform are based on convolution and the reconstruction is based on de-convolution. These two transforms are much generalized than DWT (Discrete Wavelet Transforms) and filter bank and can be used in various applications for better performance. We apply ISITRA for lossy image compression and YKSK transform for signal encryption.*

1.1 History of Filter Bank and Wavelet

We process signals for various purposes. Fourier transform has long been used as a classical tool for signal processing tasks. Fourier transform lacks time resolution in frequency domain because its window size extends from minus infinity to plus infinity. Time resolution in frequency domain is desirable for certain applications, e.g., in the analysis of transient signals. This led to the development of a small window transform, namely, Gabor transform [49] in which mainly a Gaussian function is used as the filter window. This enables us to do piecewise analysis of the signal. However, it fails to provide multi-resolution analysis because its window size is fixed. Then with the emergence of the variable window transforms, like filter bank and wavelets, the limitation on the size of the window is

overcome so that its size can now vary through multilevel decomposition (or scaling in the language of wavelet). And since then, filter bank and wavelet have occupied an important place in the field of signal processing.

The history of filter bank dates back to the late 70's with the proposal of 2-channel QMF (quadrature mirror filter) filter bank by Croisiere, Esteban, Galland [38] for use in speech and image compression. Because of its anti-aliasing property and non-redundant outputs, it has become a popular model for data compression. In fact, the 2-channel filter bank was a big step forward in the field of signal processing. Then, many researchers studied the perfect reconstruction theory in filter bank and as a consequence, several developments occurred. Conjugate quadrature filters (CQF) were introduced by Minzer [89], Smith et al, [127], bi-orthogonal filter bank, multi-dimensional filter bank were introduced by Vetterli [153,155] and M-band or multi-channel filter bank was introduced by Vaidyanathan [147] and many others. Now the 2-channel filter bank has become the basis of study for multi-rate signal processing and moreover, it has become an important part of discrete wavelet transform.

The history of wavelet can be traced back to the early part of last century [52]. But the current popular wavelet theory has started from the later part of the last century with Grossman [51] introducing the concept of approximating a signal at various resolutions through scaling and translation of some basis functions known as wavelets. The concept is based on the quantization of a signal at various step sizes. If we quantize the signal at larger step sizes we get coarser resolution and if we quantize the signal at smaller step sizes, we get finer resolution of signal with less quantization errors. If the step size is infinitesimally small, we will be able to exactly reconstruct the original signal. In a similar line of interpretation, the representation of a signal using wavelets at different scales followed. This way of representation of a signal at various resolutions through scaling and translation of wavelet function evoked immense interests among many scientists and mathematicians, and various types of wavelets were developed [14, 15, 39, 74, 86]. Among these wavelets, Daubechies wavelets developed in the line of multi-resolution concept and formalized by S. G. Mallat [81] have become popular in the signal processing community. The main reason for this popularity is that Daubechies orthogonal wavelets form a QMF filter bank and can be used in filter bank for perfect reconstruction. This established a link between wavelets and filter bank. Since then developments, more or less similar to those in the case of filter

bank, occurred in the wavelets. Cohen, Daubechies and Feaveau [34] developed bi-orthogonal wavelets, Steffen, Heller, Gopinath and Burrus [133] developed M-band wavelets, J. Kovacebic and Vitterli [71] developed multi-dimensional wavelets. Interestingly, wavelets and filter banks seem intermingled. Now, the 2-channel multilevel decomposition scheme has become a model of study in discrete wavelets. From the study of basis functions for signal representation, the study of filters also has become an important part in wavelets. In the filter bank community also, discrete wavelet transform is studied as dyadic band filter bank. Some of the commonly used books in the study of wavelets and filter banks are [5, 23, 30, 31, 40, 48, 58, 64, 82, 90, 100, 137, 151, 159]. And some more works that deal with the theory, design and properties of wavelets and filter banks are available elsewhere [8, 16, 57, 106, 107, 114, 115,128, 136, 146, 149, 151, 156, 158].

In some other developments, Burt and Adelson [22] proposed pyramid coding which has similar multiresolution concept for an efficient image coding purpose. Wim Sweldens [139, 140], developed the lifting scheme as a new and easier method for constructing new wavelet filters from the existing wavelet filters. Then Daubechies and Sweldens showed that any existing wavelet transform could be realized using the lifting scheme [41]. Another development that is of interest to researchers in the area of image compression is the integer wavelet transform. The first known integer wavelet transform, which is usually interpreted as the integer transform of Haar's wavelet, is the S-transform [55, 97]. After a gap of some years, more integer transforms appeared [70, 166] as extensions to the S-transform. Amir Said and Pearlman also proposed S+P transform [110] as an extension to the S-transform. Later on, Calderbank, Wim Swelden and Daubechies [24] showed that the lifting scheme could also be used to get integer transforms of the wavelet transforms. Mention may also be made of Michael Adam's work on integer transforms [1].

In addition, mention may be made of the development of multi-wavelets [80, 138] where two or more scaling or wavelets functions are used in the place of one scaling and one wavelet function, and the development of complex wavelets [33, 47, 72, 165] for image compression and motion estimation. More developments on wavelets such as phaselets [50], ridgelets [43], curvelets [132] have been coming up for different applications.

1.2 Motivation of the Work

During the last two decades, wavelet has been one of the most popular areas of research among mathematicians, scientists and engineers. When we study wavelet, it is found that there is no apparent relation between continuous wavelet transform (CWT) and discrete wavelet transform (DWT) with respect to perfect reconstruction. The continuous wavelet transform is a way of signal representation using a basis function known as wavelet function. There is no mention of scaling function or any need of using another basis function in CWT for perfect reconstruction or inversibility specified as admissiblity condition by Daubechies [39]. Then, with a brief introduction of multi-resolution theory of Mallat [81], a scaling function appears for the development of DWT. Then the following questions arise. If CWT does not require a scaling function why should DWT require one? Are the multi-resolution theories in CWT and DWT different? What are the inversibility conditions in DWT? Does admissibility condition of CWT has anything to do with the issue of inversibility in DWT? What could be the probable logic behind introducing a new scaling function in DWT to establish a link between 2-channel filter bank and DWT? The curiosity to get satisfying answers for these questions motivates us to do a deeper analysis between DWT and filterbank.

Another source of similar motivation to do a deeper analysis between DWT and filterbank comes from the application point of view. In most of the applications people use wavelet filters that have already been derived. The most commonly used wavelet filters are Daubechies's orthogonal filters of length 4, biorthogonal filters of length 10, commonly known as 9/7 filters. However, it is not guaranteed that these filters will be the best ones. It is a general notion that different filters will suit differently for different applications. We also assume that different filters will give different performance in different applications. We are interested in finding filters that will be suitable for any particular application in hand. For this purpose, we require an easier way to find the wavelet filters of any desired length.

We trace the property of DWT that led to perfect reconstruction and find that perfect reconstruction of wavelet filters is mainly based on perfect reconstruction theory of 2-channel filter bank, which is primarily based on the convolution operation. Interestingly,

Daubechies wavelet filters are of QMF types. We analyze step by step the decomposition and reconstruction processes of 2-channel filter bank using Bino's model of multiplication and find that the filters being of QMF type is not a necessary condition for perfect reconstruction. The main reason why we use Bino's model of multiplication is that it is easier to analyze the convolution operation involved in filtering operation because the model itself can perfectly represent the convolution operation besides polynomial and number multiplication. This model enables us to derive the necessary and sufficient conditions required for finding perfect reconstruction in 2-channel filter bank scheme. There are several ways in which we get perfect reconstruction in a 2-channel filter bank scheme. We establish in the present thesis actual conditions for getting perfect reconstruction known as member conditions from which one can easily find filters in various ways. Thus, we develop ISITRA (Indian Statistical Institute Transform) as a more generalized subband decomposition and reconstruction scheme. This also removes the difficulty in finding perfect reconstruction filters in wavelet and filter banks and provides ways of finding better filter coefficients for different purposes. We then extend the 2-channel case of ISITRA to multi-channel filter bank and multi-channel transmultiplexer scheme quite different from their existing counterparts.

In order to achieve perfect reconstruction in ISITRA, filters are to be so chosen as to satisfy the member condition. Our next aim is to see if perfect reconstruction of the original signal is possible even with quite arbitrarily chosen filters. We develop in this context a new theory of perfect reconstruction where this is possible. Here, we decompose a signal in the same way as in ISITRA using Bino's model of multiplication, and then reconstruct the signal by solving the decomposition equations in a different way as we do in the deconvolution process. To generalize the concept, we introduce two terms, known as information carriers and modifiers, in the decomposition scheme. Information carriers are to ensure perfect reconstruction and modifiers to modify the decomposed components in various ways. As long as the information carriers are intact, we can get perfect reconstruction no matter how we choose the modifiers. We call such a decomposition and reconstruction scheme YKSK (Y Kirani and S Kumar) transform. YKSK transform is found to be useful for signal encryption.

We are interested in using ISITRA for lossy image compression and YKSK transform for signal encryption. For lossy image compression purpose, we develop two subband coders entitled ISPIHT (Improved-SPIHT) and DPC (Differential Position Coder). Improved-SPIHT (Set Partitioning In Hierarchical Tree) as its name suggests is similar to SPIHT [109] in its coding approach but is modified in such a way as to reduce the computational time and memory requirement while keeping its compression performance intact. In other words, ISPIHT gives the same compression performance as SPIHT but it takes less processing time and less memory. However, the difference in processing time and memory requirement is not very significant between ISPIHT and SPIHT for lower compression ratio at 8:1 or lower. From the hardware implementation point of view, a simpler coding scheme and lower memory requirement is desired. Since SPIHT and JPEG2000 [11, 26, 28] are all complex algorithms requiring large memory space, they are difficult to implement in hardware. So, we develop DPC as a totally different new coding scheme. It is very simple and fast with considerably low memory requirement. However, its performance is marginally less than SPIHT and JPEG2000. But considering its speed and low memory requirement, it would be a suitable coding scheme from the hardware implementation point of view. Moreover, there is still scope for improving DPC.

YKSK transform has the ability to distort a signal and hence can find application in signal encryption. For encryption, simply distorting a signal is not sufficient. We require a system that would prevent the unintended people from retrieving the information hidden in the distorted signal components. For this purpose, we have developed meitei lock sequence (MLS). MLS acts as the lock and key of the distorted decomposed components of the YKSK transform and gives security to our encryption scheme.

1.3 Overview of the Thesis

The present thesis proposes two new subband transforms, namely, ISITRA and YKSK transform, and their potential applications. These two transforms are developed based on Bino's model of multiplication which in turn is based on a new system of number representation called representa that allows numbers to be treated in the same ways as arrays. Representa simplifies the manipulation of large numbers in a computer. A brief introduction to the concept of representa, which is required for a proper understanding of Bino's model of multiplication, is given in the beginning of Chapter-2. The proposed model

of multiplication is a general or common model for numbers, polynomials and arrays. The multiplication of two arrays or polynomials is known as linear convolution in signal processing, a basic filtering operation. Convolution can be preformed in an easier way using Bino's model of multiplication, which is described with examples in Chapter 2. Convolution is usually done for two arrays, which corresponds to level-2 multiplication is described in detail. Level-2 multiplication is sufficient to understand the convolution operation. General formulations for multilevel multiplication are also given which will be helpful in understanding multilevel decomposition in chapters 3 and 4.

Using Bino's model of multiplication we develop a new transform in Chapter 3. The main motivation for developing such a transform is to reduce or remove the difficulty that is commonly encountered in obtaining perfect reconstruction filters in discrete wavelet transform or 2-channel filter bank. In this chapter, we describe how the conditions on the filter coefficients and perfect reconstruction are related. The conditions that are formulated allow one to choose filters easily. In addition to making the choice of the filter coefficients easier, we also explicitly provide the decomposition and reconstruction equations which can be directly implemented in a computer. We can get several perfect reconstruction schemes and their corresponding decomposition and reconstruction equations depending on how we decompose and reconstruct a signal. In Chapter 3, we describe only the perfect reconstruction theory, where the method of decomposition and reconstruction is similar to that of the existing 2-channel filter banks. We also give the decomposition and reconstruction schemes of multilevel decomposition that corresponds to multi-resolution signal decomposition scheme. The means of getting a decomposed component at a particular decomposition level is also suggested which would eliminate the intermediate successive decomposition steps to obtain the decomposed component. From the 2-channel, our decomposition and reconstruction scheme is extended to multi-channel case. We then proposed a simpler perfect reconstruction transmultiplxer in our case. We also give decomposition and reconstruction equations for 2-dimentional signals.

In ISITRA, the decomposition and reconstruction filters need to satisfy the member condition to ensure perfect reconstruction. We are now interested in finding a perfect reconstruction subband scheme where the filters can be chosen almost without any conditions or restrictions. From the decomposition scheme of ISITRA, we see that a signal

can be decomposed into two components whose lengths are greater than or equal to the half of the length of original signal. From the theory of linear equations, we know that we can solve for *n* unknown variables from *n* linear equations. For a signal of length 2n, the two decomposed components can be thought of as a set of *n* linear equations. Thus, it is quite possible to solve for each term of the signal from the two decomposed components. This way of solution leads to YKSK transform. Various decomposition and reconstruction equations of YKSK transform are given in Chapter 4. Interestingly, YKSK transform is found to be more general than Wim Sweldens' lifting scheme. Integer transforms can easily be obtained from YKSK transform. Moreover, most of the existing integer wavelet transforms are found to be special cases of YKSK integer transforms. Integer transforms are mainly used for lossless and lossy image compression. The popularity of integer transforms for image compression is their simplicity and ability to yield perfect reconstruction after coding, which is essential for lossless compression. In the case of non-integer subband transforms, such as wavelets or ISITRA lossless compression is almost impossible because we cannot exactly code the floating-point coefficients of the decomposed components in a finite number of bits.

After the development of ISITRA and YKSK transform, we develop ISPIHT and DPC for lossy image compression and MLS for signal encryption. They are described in Chapter 5.. ISPIHT is a simplified form of SPIHT coding scheme to reduce computational complexity and memory requirement. However, its performance remains the same as SPIHT. We compare the computational complexity of ISPIHT and SPIHT at various compression ratios. We find that the difference ISPIHT and SPIHT is not very significant at lower compression ratios. The high computational complexity and requirement of large memory space are the hindrance of the most of the subband coder for hardware implementation point of view. We then develop DPC as a coding scheme, which is less complex and requires comparatively less memory space. We show the compression performance between DPC, SPIHT and JPEG2000 for various images. It is found that the compression performance of DPC is only marginally less than the standard SPIHT and JPEG2000. However, considering its simplicity, speed and low memory requirement, DPC would be very suitable for hardware implementation. In the later part of the chapter 5, we discuss MLS and the properties of a random sequence generated from it, which can be used for secure encryption of a signal. We

use MLS for encryption of speech and image signals after distorting the signals with YKSK transform. Lastly, a brief conclusion is given in Chapter 6, with some directions for further possible research.

1.4 Chapter-wise Contribution

In Chapter 2, we introduce Bino's model of multiplication based on a new number representation system known as representa. Representa is a number in an array form. This form of number representation has certain advantages especially in manipulating large numbers. The properties of representas, their similarities with numbers, their differences with arrays are also briefly described. Understanding of representa will be helpful in understanding Bino's model of Multiplication and subsequently the transforms developed by using the model. Bino's model of multiplication is a generalized model of multiplication for numbers and polynomials, and discrete convolution. We describe here how multiplication is performed according to the model and the basic rules of performing it. Some advantages of Bino's model of multiplication over conventional multiplication are also illustrated with examples. How the model can be used for multiplication of two polynomials, two arrays are also described in simpler ways. Later on, we also explain briefly how FFT (fast Fourier transform) can be used for large number multiplication. General formulations for Bino's model of multiplication for multilevel multiplication of numbers, polynomials or arrays are also provided, which will be helpful in understanding multilevel decomposition in the next two subsequent chapters.

In Chapter 3, we develop a new generalized subband decomposition and reconstruction scheme entitled ISITRA. In this chapter, we discuss the differences between ISITRA, the direct filter bank and filter bank based on polyphase representation. We establish a new perfect reconstruction theory based on ISITRA. It is found that for perfect reconstruction, the decomposition filters and reconstruction filters have to satisfy certain conditions known as member conditions. Any set of decomposition and reconstruction filters, which satisfy these conditions is defined as PRF (perfect reconstruction filter) set. We also describe how to find PRF sets of a particular length from the member conditions of a decomposition and reconstruction scheme. There are several ways in which a signal can be decomposed and reconstructed depending on different member conditions. Only the member conditions that correspond to the decomposition and reconstruction schemes of 2-channel filter bank or

discrete wavelet transform are discussed in this Chapter. Other member conditions and their corresponding decomposition and reconstruction equations are given in Appendix A. Once we find a PRF set, we can easily obtain more number of PRF sets from its framework. A framework of PRF sets can be obtained from a single vector or from two distinct vectors, and accordingly ISITRA is classified into two types, namely, ISITRA-1 and ISITRA-2. We find that the PRF sets for Daubechies' wavelet are subsets of certain PRF sets of a particular reconstruction scheme of ISITRA. The frameworks of PRF sets of Daubechies' orthogonal wavelet filters are special cases of ISITRA-1 and bi-orthogonal wavelet filters are special cases of ISITRA-2. More frameworks of PRF sets of wavelet filters (orthogonal and bi-orthogonal) are given in Appendix-B. We can find infinitely many PRF sets from a framework. To find a PRF set from a given framework that will give good performance in all applications is difficult. However, it is possible to find a PRF set that will give optimum performance for any specific application by using some optimization based search techniques. A systematic search technique for finding optimal PRF sets for a given framework based on minimization of mean square error using genetic algorithm is provided in Appendix C. In ISITRA, different types of PRF sets can be obtained easily, and accordingly, a signal can be decomposed into different ways. A signal can be decomposed into two approximations or into two details or into one approximation and one detail. By "an approximation" we mean a decomposed component having less variation in the values of its data elements (usually obtained with a low pass filter) and by "a detail" we mean such a component having high variations in the values of its data elements (usually obtained with a high pass filter). The decomposition scheme is then extended to multilevel decomposition which is further classified into two types; namely, multilevel half and full decomposition schemes that will enable multi-resolution analysis. We also illustrate how multi-resolution signal representation is obtained in the multilevel decomposition scheme of ISITRA.

In another direction, we extend the 2-channel decomposition scheme to multi-channel decomposition scheme using techniques similar to those of the existing multi-channel filter bank. From the multi-channel filter bank scheme, we also propose a new multi-channel perfect reconstruction transmultiplexer. A method to obtain the filter coefficients for perfect reconstruction filters of various lengths is also described. In the last section, we provide the decomposition and reconstruction schemes of 2-D signals.

In Chapter 4, we describe YKSK transform. YKSK transform is of two types, namely, YKSK-1 and YKSK-2 transforms. Each type is further classified into three types, namely, fixed length, semi infinite and infinite length types. Possible forms of each type of YKSK-1 are described in the first part of the chapter. Multi-stage decomposition of YKSK-1 transform is then proposed which may be useful to perform decomposition and reconstruction of a signal with longer filters in one stage to do in several stages with shorter filters. Multi-stage decomposition scheme of YKSK-1 transform is a generalized version of Sweldens' lifting scheme [139]. Then, an integer version of YKSK-1 transform is proposed. All the integer transforms suggested in [1,24] are found to be special cases of YKSK-1 integer transform. A generalization of these integer transforms using YKSK-1 transform is also given so that different integer transforms can be generated easily. Then, we suggest YKSK-2 transform as a symmetrical form of YKSK transform. It is more general than YKSK-1 transform in the sense that the latter can be obtained as a special case of the former. Like YKSK-1 transform, YKSK-2 transform also is of three types. Multi-stage decomposition is possible in YKSK-2 transform also. There are two different ways of reconstruction in YKSK-2 transform. Correspondingly, there are two integer versions of YKSK-2 transform. The second integer version is similar to that of YKSK-1 transform in approach. We then describe a special form of YKSK-2 transform known as Simplet. The properties and advantages of Simplet are provided.

In Chapter 5, we describe the use of ISITRA in image compression and of YKSK transform in signal encryption. In the first part of the chapter, we discuss on the image compression model and the parameters used to measure the performance of an image compression scheme. For lossy image compression, commonly, MSE (Mean Square Error) and PSNR (Peak Signal to Noise Ratio) values are used for error quantification as a measure of performance of a lossy compression scheme. Then, we describe ISPIHT as an improved coding scheme to SPIHT. The difference between ISPIHT and SPIHT in their coding strategy and memory requirement are discussed in this chapter. The coding and decoding of DPC is described. The compression performance between DPC and standard subband coders SPIHT and JPEG2000 are given to show that the compression performance is comparable to these two standard subband coders in spite of its simplicity. The coding time and decoding time of DPC and SPIHT at various compression ratios are given for codes written in

MATLAB to show the significant difference in speed between DPC and SPIHT. Then we compare the compression performance between commonly used 9/7 wavelet filters and corresponding ISITRA filters of length-10. It is found that some ISITRA-2 filters of length-10 gives marginally better performance than commonly used 9/7 bi-orthogonal filters in some images. From the compression performance, it can be observed that it is possible to find a better filter sets than the existing ones as we cannot claim that the existing filters are the best ones.

We, then, describe meitei lock sequence and some of its properties. Meitei lock sequence is a random sequence generated from an arbitrarily chosen nonzero array, whose elements are not equal. We use MLS for encryption of a chirp signal and an image. For encryption, we apply Simplet, a special form of YKSK-2 transform, for its simplicity and ability to distort a signal fast. To show the encryption performance we measure the correlation coefficient between the original signal and their decrypted signals for various (more or less similar) decryption keys. It is found that finding distinguishable signal is almost impossible when there is a slight difference between the encryption and decryption key vectors. Our encryption scheme is found to be very tight for encryption of speech and image signals.

1.5 Summary

In this thesis, we propose two new subband transforms entitled ISITRA and YKSK transforms and their possible applications in image compression and encryption. Both these transforms are developed based on a common model of multiplication known as Bino's model of multiplication. ISITRA is a convolution based transforms i.e., that both forward and inverse transform of ISITRA is based on convolution as in DWT or 2-channel filter bank. However, it is much more general than the existing DWT or 2-channel filter bank scheme in the sense that it we can get different kinds of filters in addition to the filters specified for perfect reconstruction in the existing 2-channel filter bank. ISITRA eliminates the difficulty of finding perfect reconstruction filters in 2-channel subband scheme and provides the ways of finding better filter coefficients for different purposes. Also, we can generate different forms of decomposition and reconstruction in ISITRA. Multi-resolution signal decomposition is also possible in ISITRA using multilevel decomposition scheme.

Moreover, we can also get perfect reconstruction multi-channel filter bank and multi-channel transmultiplexer.

YKSK transform is also a generalized subband transform like ISITRA. Its forward transform is based on convolution but its inverse transform is based on deconvolution and hence filters can be chosen arbitrarily. YKSK transform has three main types, namely, fixed length type, semi-infinite length type and infinite length type. Multi-stage decomposition is possible in YKSK transform. Integer version of the YKSK transform can easily be obtained. Both multilevel half and multilevel full decomposition can also be performed in YKSK transform. Multilevel decomposition of fixed length type gives multi-resolution signal decomposition as in ISITRA. Multilevel decomposition in the case of infinite length case gives distorted signal components.

Lastly, we use ISITRA for lossy image compression and YKSK transform for image encryption. Every lossy compression scheme requires good coding schemes besides the subband transform used. We propose two coding schemes ISPIHT and DPC for subband based lossy image compression scheme. ISPIHT is an improved SPIHT in terms of speed and memory requirement. However, its performance remains the same as that of SPIHT. DPC is a very simple and significantly faster new subband coding scheme with low memory requirement. The popular subband coders, EZW, SPIHT, EBCOT and JPEG2000 are yet to find a proper place in the hardware or chip world because of their complex algorithm and requirement of large memory space. DPC, however, it is marginally less in compression performance as compared to SPIHT and JPEG2000, has no such problems and hence would be of immense interest for hardware designers. We also suggest some PRF sets of length-10 having the same frameworks as that of 9/7 filters, which perform better in some images. For the use of YKSK transform for encryption, we develop meitei lock sequence to provide tight security for our encryption scheme. Meitei lock sequence is a good random number sequence generator and may become an essential part of every secure encryption scheme in future.

Chapter 2

BINO'S MODEL OF MULTIPLICATION

Abstract: *We propose here a new way of multiplication entitled Bino's model of multiplication. It is a unified model for multiplication of numbers, polynomials and arrays. This model, based on a number representation system known as representa, allows us to perform multiplication of large numbers in small computers faster than the conventional way of multiplication. It is better and more generalized than that of Karatsuba and Ofman's algorithm for fast number multiplication. The multiplication of two arrays (discrete sequences) is known as convolution in signal processing. The same convolution (or filtering) operation is well represented by this model of multiplication. In other words, convolution can also be used for number multiplication and this model of multiplication can be used for convolution. Using this model, convolution operation can be performed in a much simpler way. It enables us in deriving a generalized perfect reconstruction theory of 2-channel filterbank. It is a recursive process and can be easily generalized to multilevel convolution, which in combination with downsampling operations gives rise to multiresolution representation as discussed in the next chapter. Multinomial expansions and various types of convolution triangles can also be obtained from this model.*

2.1 Introduction

Multiplication is generally done between two numbers. It is considered as a shortened form of addition. Because multiplication of two numbers, say x and y, gives the same result as y times addition of x or x times addition of y. In early times, by multiplication, we used to mean multiplication of numbers, i.e., integer multiplication. With the development of concept of variables in mathematics, we now consider polynomial multiplication as well, which is normally assumed to be different from number multiplication. In our conventional

way of multiplication, generally n^2 multiplication operations are required for multiplying two n-digit numbers or n-coefficient polynomials. The number of multiplication operations required, can be reduced to much less than n^2 under different conditions, e.g., when the two numbers are the same or when one of the numbers is made of the same digit etc. Karatsuba and Ofman [65] proposed an algorithm to reduce the number of multiplication operations. The same algorithm is described in detail in [161]. But their algorithm requires a lot of addition operations in exchange of some reduction in multiplication operations and cannot be generalized.

The method of multiplication we generally use enables us to multiply only two numbers at a time. We cannot multiply three or more different numbers at a time. Also we treat polynomial multiplication and number multiplication as different multiplications and the way of multiplication cannot be generalized. Bino's model of multiplication [116] takes care of all such inconveniences and offers an efficient way of multiplication. The model can lead to a general formula for multiplying several multi-digit numbers at a time; the numbers may not necessarily be equal. Also, this model represents polynomial multiplication, discrete convolution and integer multiplication. Using this model, we can perform convolution in an easier way. There are different approaches to perform discrete convolutions based on number theoretic transforms [3, 4, 99] but they are not suitable to analyze the perfect reconstruction theory of 2-channel filterbank or discrete wavelet transforms. The way Bino's model of multiplication represents discrete convolution is very suitable to analyze the filter bank scheme. Hence, in the thesis, we use this model mainly for performing discrete convolution to derive a generalized way of perfect reconstruction in the case of 2-channel filter bank.

In this chapter, we describe the basics of the model in relation to number multiplication for easy understanding of the model. Once the model is understood, it can be easily applied to polynomial multiplication and discrete convolution. For number multiplication we use a different way of variable representation for numbers known as representa (Represented in Array), which is described in the subsection 2.2. Representing a multi-digit number in the form of representa allows us to treat numbers as if they are arrays. Moreover, it saves quite some memory space for representing very large numbers. In the rest of the section we will denote a representa by an array for simplicity. We would make the difference between a

representa and an array by mentioning a base for a representa. Also we will use the term length of a representa to denote the number of elements contained in it. Also, we define level of multiplication. If two representas are multiplied, we call it level-2 multiplication, multiplication of three representas is called level-3 multiplication and so on.

2.2 Representa (Representation of a number in array form)

In mathematics, we generally use two types of entities, namely, constants and variables. The constants are numbers we generally use for actual calculation. And variables are the substitutes for constants for easy and efficient manipulation. A variable can represent any number regardless of its type and magnitude. However, the existing number representation system has certain problems in dealing with large numbers in a computer. A variable refers to a memory location in a computer. A memory location can hold a number within a certain range. That is, we cannot store a larger number in a memory location than its allowable maximum value. This problem is referred to as byte size limitation in a computer. So, we cannot handle large numbers in the same way as we do with smaller numbers that can be stored in a memory location. In some applications like cryptography, handling of large numbers is required. For handling large numbers [68], one has to treat numbers as polynomials and has to write long and complicated programs which in turn slow down the computing speed. We find that arrays can be treated in the same way as we treat numbers, if we impose place values on the elements of arrays. Thus, we present a new way of number representation, known as representa (represented in array). A representa is a number in an array form and all the properties of a number are preserved in it. We can treat representas in the same way as we treat numbers.

An array is a list of numbers. Arrays may be multi-dimensional. But here by an array we will mean a one-dimensional array. The numbers contained in an array are known as the elements of the array. There are many ways in which arrays are represented. Here we will denote an array by a list of numbers enclosed in bracket. We will also use commas to separate the individual elements of an array. Bold letters will denote array variables. For example, $\mathbf{A} = [1,3,6]$ is an array of three elements, whose name is $\mathbf{A}$. Using the array name with an index we can refer to every individual element of an array. The index associated with an array name indicates which element of the array is being referred to. There are two

ways for array indexing; 0-offset index and 1-offset index. In the former case, the first element of an array $\mathbf{A}$ is denoted by $\mathbf{A}(0)$ and in the later case by $\mathbf{A}(1)$. We will use the later case, as it is in conformity to our counting of things starting from 1. Using this convention $\mathbf{A}(i)$ will denote the i^{th} element of the array $\mathbf{A}$. There is no limitation on how many elements an array can contain. An array can have as many elements as possible. This is true in the programming aspect as well. But the size of an element cannot exceed the byte size limitation. But this presents no problem for the representation of a large number using an array.

Arrays can be used to represent numbers. Arrays with proper bases that are used to represent numbers will be known as representas to distinguish them from the ordinary arrays. They are designed to computationally manipulate large numbers in an easier way. All representas are arrays but all arrays are not representas. The characteristics of representas are different from those of arrays but are similar to those of numbers. The elements of a representa except the first element should be positive. The sign of the first element indicates the sign of a representa in the same way as the sign of the first digit of a number represents the sign of the number. Each element of a representa should be less than its base as each digit of a number is less than its base. Thus, representas are very similar to numbers.

Let us now see what kind of an array a representa is and how it can be used to store large numbers. An array is made of individual elements as a number is made of digits. The digits in a number have place values in terms of the corresponding system base. If we want to express numbers as arrays and want to treat the arrays in the same way as we treat numbers, then we have to define place values for the elements of the arrays. Let us see how 143759 in decimal system can be represented in an array. A simple way is to make each digit of the number an element of an array $\mathbf{A} = [1,4,3,7,5,9]$. Then we can use this array to represent the number as $143759 = \sum_{i=1}^{6} \mathbf{A}(i)(10)^{6-i}$.

Here the base of the array $\mathbf{A}$ is also 10. We see that the number can exactly be represented in an array. The array $\mathbf{A}$ is then known as a representa and is denoted by the symbol $\mathbf{A} : b$ where b is the base. The optimal value of b is its highest value that a computer can accommodate within its byte size limitation. As an array can have any number of elements

(in a computer as long as sufficient memory is available), we would be able to represent any arbitrarily large number by using a representa. Now, it is not imperative that each digit be made an element of a representa. We can combine two or more consecutive digits and make them an element of a representa. In that case, the base of the representa should also be changed as described in Example 1. The base of a representa depends on how many consecutive digits will be combined in each element of a representa.

For example, the decimal number 14325678 can be represented as different representas with different bases.

$\mathbf{A}:10=[1,4,3,2,5,6,7,8]$, a representa with base 10.

$\mathbf{A}:100=[14,32,56,78]$, a representa with base 100.

$\mathbf{A}:1000=[14,325,678]$, a representa base 1000.

$\mathbf{A}:10000=[1432,5678]$, a representa with base 10000, and so on.

The representation of a number in a representa of higher bases (than 10) has two main advantages. The first is that it saves a lot of memory storage. This is because, in most programming languages, all integers take the same amount of memory irrespective of their magnitudes. Suppose storing an integer takes 8 bytes of memory. Then the representa [1, 4, 3, 2, 5, 6, 7, 8] (base 10) would require 64 bytes of memory. But the representas [14, 325, 678] (base 1000) and [1432, 5678] (base 10000) would require only 24 and 16 bytes respectively, thus saving 40 and 48 bytes of memory respectively for representing an 8-digit number. More memory would be saved for storing larger numbers. The second advantage is saving of computation time, since less number of elements requires less accessing and processing time.

We have seen that we can use representas to store numbers in an array form. The value of the number represented by a representa $\mathbf{A}:b$ will be denoted by $(\mathbf{A})_b$ where b is the base. In general, if $\mathbf{A}:b$ is a representa having m elements, then we can write

$$(\mathbf{A})_b = \sum_{i=1}^{m} \mathbf{A}(i) b^{m-i}$$

This has a similarity with the representation of a number in terms of its base. The only difference is that digits are replaced by elements.

All representas are arrays but not all arrays are representas. They have the following differences.

1. In a representa all the elements except the first element should be positive and each element should be less than the base in magnitude. But there is no such restriction in arrays. An array may have two or more negative numbers as its elements.
2. The sign of the first element of an array represents the sign of the representa. There is no concept of sign in an array.
3. The elements of a representa have place values. But the elements of an array do not.
4. Representas can be operated in the same way as numbers but arrays cannot.

This way of number representation can be applied for any number system octal, hexadecimal etc, which has a base. Also, we can convert easily from one system to another using this representation. More on conversion from of a number from one system to a number in another systems, readers are referred to [13, 20, 46, 92,96]. We can perform mathematical operations such as addition, subtraction, multiplication division etc, in the same way as we do in a number system by taking care of the role of appropriate place values. Representa can handle floating-point numbers as well. The simplest way, to handle floating point number is to treat the floating-point numbers as if they are integers, noting their decimal positions. After performing necessary mathematical operations, we can restore the decimal point at the appropriate position in the result. More on floating point numbers and their fast arithmetic operations are discussed in [134, 162]. But we can perform multiplication in a simpler way by using a common model of multiplication for numbers, polynomial and arrays, which is discussed next.

2.3 Bino's Model of Multiplication

Bino's model of multiplication (named after the author’s father late Yumnam Bino Singh) is described diagrammatically (Fig.2.1) along with the rules given below. Fig.2.1 shows the model of multiplication of two representas of length 3. Here the empty circles are the digits or elements to be multiplied. The filled circles are the multiplication terms, which will give the result of the operation.

Rule 1: When two digits/elements are multiplied, a straight line called multiplication line will connect them.

Rule 2: The result of multiplication, called a multiplication term, comes from the middle of the multiplication line.

Rule 3: If two or more multiplication lines bisect one another, the multiplication term comes from the point of bisection and the result of each multiplication line is summed up to obtain the multiplication term.

Rule 4: Each multiplication term is connected to the middle of each multiplication line by a dotted line known as result line.

Rule 5: If each element in each number has a place value, then each multiplication term has a corresponding place value, otherwise not.

Let $\mathbf{A}_1 = [a_{11}, a_{12}, a_{13}]$ and $\mathbf{A}_2 = [a_{21}, a_{22}, a_{23}]$ be any two representas with base b or any two arrays or sequences then following the above rules, the model of multiplication is shown in Fig.2.1.

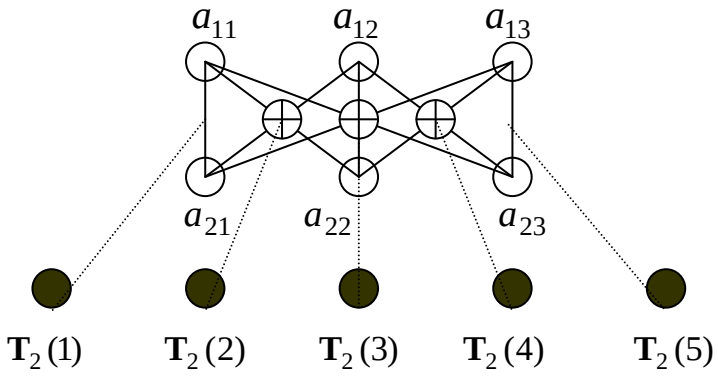


Figure 2.1 3-digit level-2 Bino's model of multiplication.

The result of multiplication of two representas yields in a representa. Multiplication terms are stored in the array $\mathbf{T}$ with the subscript denoting the level of multiplication. In the above case, $\mathbf{T}_2$ denotes the array of multiplication terms in level 2 multiplication. For easy reference, we will call the array of multiplication terms a result array. The elements of a result array are the multiplication terms and each multiplication term can be accessed with an index. For example, $\mathbf{T}_2(3)$ denotes the third multiplication term in the level 2 multiplication. The result array for multiplication of representas or arrays is the same if the

elements of the representas are also the elements of the arrays. Once we get a result array we can get different results by manipulating it in different ways (Subsection 2.3.2).

Thus, the essential components of the proposed model of multiplication are :

Multiplication term: A term connected with a result line.

Result-array: The array of multiplication terms of multiplying two or more representas or arrays.

Result of Multiplication: The result of multiplication depends on the entities to be multiplied. If they are representas with a base, the result of multiplication is obtained from the result array by converting it into a representa with the same base. If the entities are arrays or sequences the result array itself is the result of multiplication.

Let us consider the values of the multiplication terms obtained from multiplication of two representas or arrays $\mathbf{A}_1$ and $\mathbf{A}_2$, and how they are obtained. There are five multiplication terms in the result array of the model. They are obtained from the diagram as

$$\mathbf{T}_2(5) = a_{23}a_{13}$$
$$\mathbf{T}_2(4) = a_{23}a_{12} + a_{22}a_{13}$$
$$\mathbf{T}_2(3) = a_{23}a_{11} + a_{22}a_{12} + a_{21}a_{13}$$
$$\mathbf{T}_2(2) = a_{22}a_{11} + a_{21}a_{12}$$
$$\mathbf{T}_2(1) = a_{21}a_{11}$$

The first and the last terms involve no addition because they are multiplication terms from non-bisecting multiplication lines. The second and the fourth multiplication terms involve one addition because two multiplication lines bisect each other and the corresponding multiplication terms come from the respective points of bisection which are indicated by a plus sign in Fig.2.1. And the third multiplication term involves two additions because three multiplication lines bisect one another and the corresponding multiplication term comes from the point of bisection of the three multiplication lines.

The result of the above level-2 multiplication, can be written mathematically as

$$(\mathbf{T}_2)_b = \sum_{r=1}^{5} b^{5-r}\mathbf{T}_2(r), \quad \text{(for direct multiplication)}$$

$$(\mathbf{T}_2)'_b = \sum_{r=1}^{5} b^{r-1}\mathbf{T}_2(r), \quad \text{(for reverse multiplication)}$$

From these two equations, we can get the result of multiplication of two representas and also the result of multiplication of the reversed representas from the result array. By a reversed representa, we mean the representa obtained after reversing the elements. For example, the reversed representa of $\mathbf{A}_1 = [a_{11}, a_{12}, a_{13}]$ is $\mathbf{A}'_1 = [a_{13}, a_{12}, a_{11}]$. The results of multiplication (direct or reversed) can also be obtained manually easily from the result array in two ways, namely, forward slant addition and backward slant addition, as discussed in the following sub-section.

2.3.1 Forward slant addition

In forward slant addition of multiplication terms, the multiplication terms are written vertically with the last multiplication term at the top, as if we are going to do conventional addition of multiplication terms. Slant lines are drawn starting from the top rightmost element as shown in Fig.2.2a. Those elements/values crossed by a slant line will be added together and placed in proper position of the corresponding base. The result of multiplication corresponding to Fig.2.2a gives the result of direct multiplication.

We can also get the result of multiplication of reversed representas from the result array of direct multiplication without actually doing multiplication of reversed representas using forward slant addition. For this purpose, we write the multiplication terms vertically, with the first multiplication term at the top, and forward slant lines are drawn as shown in Fig.2.2b. Elements crossed by a slant line are added and the sum is kept at the corresponding position of the slant line.

Suppose $\mathbf{T}_2(7) = x_{11}x_{12}x_{13}$, $\mathbf{T}_2(6) = x_{21}x_{22}x_{23}$, $\mathbf{T}_2(5) = x_{31}x_{32}x_{33}x_{34}$, $\mathbf{T}_2(4) = x_{41}x_{42}$, $\mathbf{T}_2(3) = x_{511}x_{52}x_{53}$, $\mathbf{T}_2(2) = x_{61}x_{62}x_{63}$ and $\mathbf{T}_2(1) = x_{71}$ are the multiplication terms of the of a level-2 result array. Then the forward slant addition is shown in Fig.2.2.

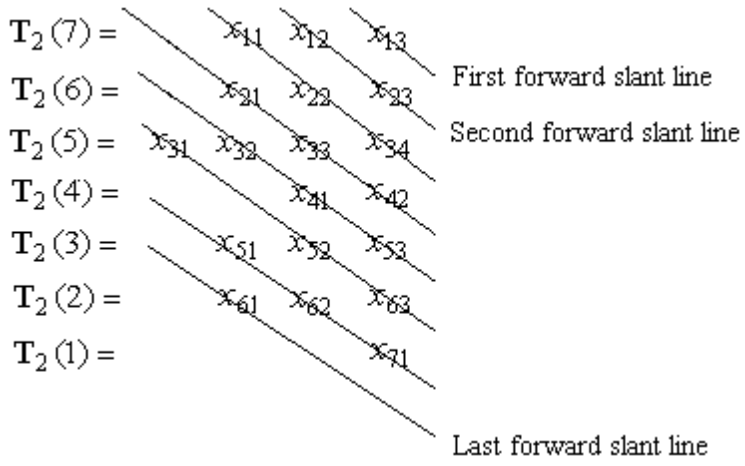


Figure 2.2a Forward slant addition (direct multiplication)

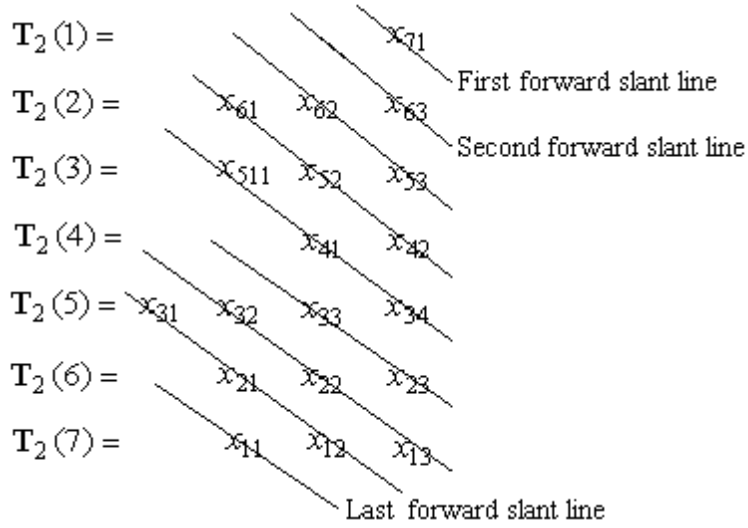


Figure 2.2b Forward slant addition (reverse multiplication)

2.3.2 Backward slant addition

As in forward slant addition, we can get the result of direct and reverse multiplications using backward slant addition. For finding the result of multiplication of reversed representas, the multiplication terms are written vertically with the last multiplication term at the top, as if we are going to do conventional addition of multiplication terms. Then, backward slant lines are also drawn similarly as shown in the Fig.2.3a. The value crossed by the first slant line corresponds to the first value of the result, i.e., the value of the result of multiplication in the unit's place. Addition is done along the remaining slant lines to get the result. Note here that if we perform forward slant addition in Fig.2.3a, we get the result of direct multiplication.

The backward slant addition can also be done to obtain the result of direct multiplication of representas from the result array of direct multiplication. For this purpose, we arrange the multiplication terms vertically with the first term at the top, and then the backward slant lines are drawn as shown in Fig.2.3b. Note here also that if we do forward slant addition in Fig.2.3b we would get the result of reverse multiplication. That is, we can say that forward slant addition is the reverse process of backward slant addition. Thus, a convenient way of finding the results of multiplication of two representas and their reverses is to do backward slant addition and then forward slant addition or vice versa.

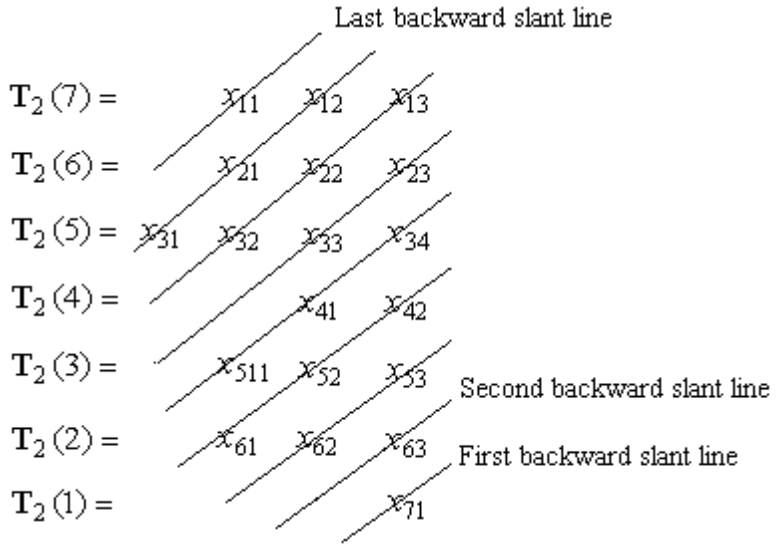


Figure 2.3a Backward slant addition (for reverse multiplication)

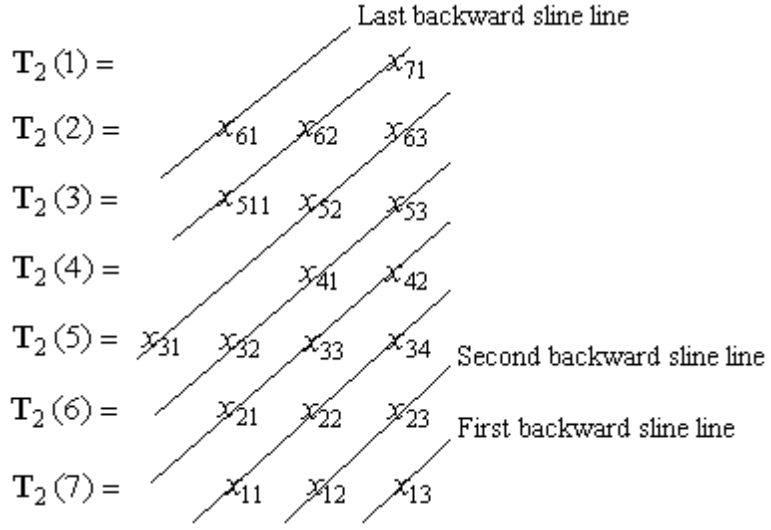
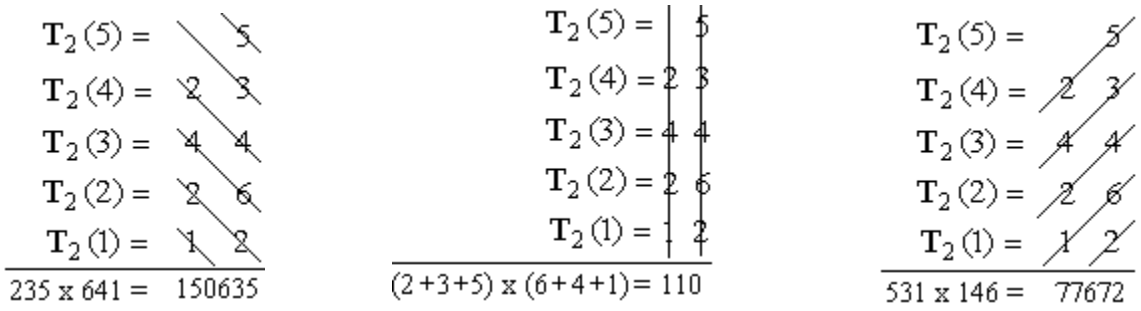


Figure 2.3b Backward slant addition (for direct multiplication)

For example, suppose 235 and 641 are to be multiplied. For simplicity and ease of explanation, let the two numbers be represented as two representas under the same base 10 as $\mathbf{A}_1 = [2,3,5]$ and $\mathbf{A}_2 = [6,4,1]$. The corresponding multiplication terms are $\mathbf{T}_2(5) = 5; \mathbf{T}_2(4) = 23; \mathbf{T}_2(3) = 44; \mathbf{T}_2(2) = 26; \mathbf{T}_2(1) = 12$. As the multiplication terms for this 2-level multiplication are known, then by doing forward slant addition of the multiplication terms we get the result of multiplication of 235 and 641 (Fig.2.4a). And by doing backward slant addition of the same multiplication terms we get the result of multiplication of the reversed numbers, i.e., 532 and 146 (Fig.2.4c). Also, by simply adding up the multiplication terms, we can get the value of polynomial multiplication (Fig.2.4b). And the result array itself is the result of convolution of two arrays as if the representas were arrays.

Using the multiplication terms of 2-level multiplication of three digit numbers derived earlier, we get the corresponding multiplication terms of multiplication of 235 and 641 as



(a) (b) (c)

Figure 2.4 Multiplication results using slant additions.

If the two representas were arrays, then $\mathbf{A}_1 \otimes \mathbf{A}_2 = [12, 26, 44, 23, 5]$, i.e., by simply putting the multiplication terms. This is what is generally called convolution [54,82] and is denoted by the symbol ⊗. And if the elements in a representa are separated by a plus sign, we will denote the corresponding representa by putting a positive sign in the upper right corner, i.e., $A_1^+ = (2+3+5)$ and $A_1^+ = (6+4+1)$. Then we get $A_1^+ \times A_2^+ = (12+26+44+23+5)$, i.e., by adding up the multiplication terms. This is what is popularly known as polynomial multiplication. Thus, by simply finding the multiplication terms once we can find not only the result of multiplication of two numbers but also the result of

multiplication of the reverse numbers, convolution of the digits in vector forms and polynomial multiplication of the digits in polynomial forms. This shows how Bino's model of multiplication can be used for performing number multiplication, discrete convolution and polynomial multiplication. More details about convolution and polynomial multiplication are discussed in the next section.

2.4 General Level-2 Multiplication

Any number or representa or array can be thought of as the result of multiplication of the number or representa or array with the multiplication identity 1. This can be called level-1 multiplication. Multiplication of two numbers or representas means multiplication of two level-1 result terms. So multiplication of two numbers or representas is known as level-2 multiplication. The level number can also be thought as the power of the representa when the representas are the same.

Let $\mathbf{A}_1 = [a_{11}, a_{12}, a_{13}, \ldots, a_{1k_1}]$ and $\mathbf{A}_2 = [a_{21}, a_{22}, a_{23}, \ldots, a_{2k_2}]$ be any two representas of base b having lengths k_1 and k_2 respectively. Then, according to Bino's model of multiplication

$$\mathbf{A}_1 \times \mathbf{A}_2 : b = (\mathbf{T}_2)_b = \sum_{r=1}^{L_2} b^{L_2 - r}\ \mathbf{T}_2(r) \cdots\cdots\cdots (1)$$

where $L_2 = (k_1 + k_2) - 1$ is the total number of multiplication terms in level-2 multiplication of two numbers of lengths k_1 and k_2. Assuming $k_1 > k_2$ each multiplication term is given by

$$\mathbf{T}_2(L_2 + 1 - r) = \sum_{i=1}^{k_2} \mathbf{A}_2(k_2 + 1 - i)\mathbf{A}_1(k_1 - r + i) \cdots\cdots\cdots (2)$$

The values of the elements of the arrays or representas are zero outside their range, e.g, $\mathbf{A}_1(r) = 0$ for $r < 1$ or $r > k_1$. This is the usual way of multiplication that starts from the right end. If we start multiplication from the left end, the level-2 multiplication terms can be written as

$$\mathbf{T}_2(r) = \sum_{i=1}^{k_2} \mathbf{A}_2(i)\mathbf{A}_1(r + 1 - i) \cdots\cdots\cdots (3)$$

This seems to be simpler than equation (2) and is similar to the equation of convolution of two finite sequences of lengths k_1 and k_2 as discussed in Section 2.5. Both equations (2) and (3) require k_2 multiplications and $k_2 - 1$ additions for each multiplication term. But for any multiplication term $\mathbf{T}_2(r)$ for $r < k_2$, requires less number of multiplications than k_2. When $r \le k_2 \le L_2 - r$, the computation of $\mathbf{T}_2(r)$ requires k_2 multiplications and when $L_2 - r < k_2$, it requires less than k_2 multiplications. In all the above equations, the upper limit of the summation is the minimum of the two lengths. As we have assumed $k_1 > k_2$, the upper limit is made k_2.

So equation (2) can be written in the following way

$$\left.\begin{aligned}
\mathbf{T}_2(L_2+1-r) &= \sum_{i=1}^{r} \mathbf{A}_2(k_2+1-i)\mathbf{A}_1(k_1-r+i) \ \textit{for } r < k_2 \\
&= \sum_{i=1}^{k_2} \mathbf{A}_2(k_2+1-i)\mathbf{A}_1(k_1-r+i) \ \textit{for } r \le k_2 \le L_2 - r \\
&= \sum_{i=1}^{L_2-r} \mathbf{A}_2(k_2+1-i)\mathbf{A}_1(k_1-r+i) \ \textit{for } L_2 - r < k_2
\end{aligned}\right\} \quad \cdots\cdots\cdots (4)$$

Similarly, we can write equation (3) as

$$\left.\begin{aligned}
\mathbf{T}_2(r) &= \sum_{i=1}^{r} \mathbf{A}_2(i)\mathbf{A}_1(r+1-i) \ \textit{for } r < k_2 \\
&= \sum_{i=1}^{k_2} \mathbf{A}_2(i)\mathbf{A}_1(r+1-i) \ \textit{for } r \le k_2 \le L_2 - r \\
&= \sum_{i=1}^{L_2-r} \mathbf{A}_2(i)\mathbf{A}_1(r+1-i) \ \textit{for } L_2 - r < k_2
\end{aligned}\right\} \quad \cdots\cdots\cdots (5)$$

Any one of equations (4) and (5) can be conveniently used for multiplication of any two multi-digit numbers manually or otherwise.

For obtaining the result of multiplication of the reverse numbers from the multiplication terms of the two given numbers, we have to simply reverse the multiplication terms as given by the following equation.

$$\left.\begin{aligned}\mathbf{A}_1^{'} \times \mathbf{A}_2^{'} &= \sum_{r=1}^{L_2} b^{r-1}\, \mathbf{T}_2(r) \\ &= \sum_{r=1}^{L_2} b^{L_2 - r}\, \mathbf{T}_2^{'}(r)\end{aligned}\right\} \cdots\cdots\cdots(6)$$

where $\mathbf{A}_1^{'}$, $\mathbf{A}_2^{'}$ and $\mathbf{T}_2^{'}$ denote the reversed versions of $\mathbf{A}_1$, $\mathbf{A}_2$ and $\mathbf{T}_2$ respectively. Suppose the two representas to be multiplied are in the form of a zero-order polynomial, i.e.,

$A_1^+ = (a_{11} + a_{12} + a_{13} + \ldots + a_{1k_1})$, $A_2^+ = (a_{21} + a_{22} + a_{23} + \ldots + a_{2k_2})$.

The multiplication terms are the same as given by any of the above formulas for $T_2(r)$. But their result is obtained by adding up the multiplication terms together. That is,

$$A_1^+ \times A_2^+ = \sum_{r=1}^{L_2} \mathbf{T}_2(r) \quad \cdots\cdots\cdots(7)$$

If the two representas are arrays, then the multiplication operation performed using Bino's model multiplication is known as the convolution operation and their result is given simply by the result array itself. That is, the convolution of two arrays $\mathbf{A}_1$ and $\mathbf{A}_2$ having the same elements as those of the two representas, is given by the following relation.

$\mathbf{A}_1 \otimes \mathbf{A}_2 = \mathbf{T}_2 \cdots\cdots\cdots(8)$

Here, again $\mathbf{T}_2$ is the result array of the multiplication of the two representas.

From the equations (6), (7) and (8), we know that what is required is the multiplication terms. Once the multiplication terms are found, we can use them for number multiplication, polynomial multiplication and convolution by simply manipulating the terms in different ways. In other words, we can say that number multiplication, polynomial multiplication and convolution are essentially the same process.

We will now consider some interesting conditions from equation (4), which will be useful for performing multiplication quickly easily and in short, as usually we start multiplication from the right end.

2.4.1 Some conditions for easy multiplication

The easiness or quickness of multiplication also lies in the pattern of the elements of the representa or arrays. We consider the following cases.

Condition 1: If the elements of $\mathbf{A}_2$ are the same, i.e., $a_{21} = a_{22} = \ldots = a_{2k_2} = a_2$, say, then from equation (4), we have

$$\left.\begin{aligned} \mathbf{T}_2(L_2+1-r) &= a_2 \sum_{i=1}^{r} \mathbf{A}_1(k_1-r+i) \ \textit{for } r < k_2 \\ &= a_2 \sum_{i=1}^{k_2} \mathbf{A}_1(k_1-r+i) \ \textit{for } r \le k_2 \le L_2 - r \\ &= a_2 \sum_{i=1}^{L-r} \mathbf{A}_1(k_1-r+i) \ \textit{for } L_2 - r < k_2 \end{aligned}\right\} \quad \cdots\cdots\cdots(9)$$

Here each multiplication term requires only one multiplication operation. So, for multiplying two sequences or numbers having lengths k_1 and k_2 only $(k_1 + k_2 - 1)$ multiplication need to be performed which would otherwise require $k_1 \times k_2$ multiplication operations. Also, there is no apparent increase in the number of addition operations.

Condition 2: If the elements of $\mathbf{A}_1$ are the same, i.e., $a_{11} = a_{12} = \ldots = a_{1k_1} = a_1$, say, then we would have the similar multiplication terms, i.e.,

$$\left.\begin{aligned} \mathbf{T}_2(L_2+1-r) &= a_1 \sum_{i=1}^{r} \mathbf{A}_2(k_2+1-i) \ \textit{for } r < k_2 \\ &= a_1 \sum_{i=1}^{k_2} \mathbf{A}_2(k_2+1-i) \ \textit{for } r \le k_2 \le L_2 - r \\ &= a_1 \sum_{i=1}^{L_2-r} \mathbf{A}_2(k_2+1-i) \ \textit{for } L_2 - r < k_2 \end{aligned}\right\} \quad \cdots\cdots\cdots(10)$$

Here also each multiplication term requires only one multiplication. And the number of multiplication operations required is $(k_1 + k_2 - 1)$.

So, if one of the two representas is made up of the same element then the multiplication can be done with much less multiplication operations than the conventional method of multiplication.

Condition 3: If both the representas were made up of only one element, say a, then the multiplication terms become

$$\left.\begin{aligned}\mathbf{T}_2(L_2+1-r) &= a^2 r \ \ for\ r < k_2\\ &= a^2 k \ \ for\ r \le k_2 \le L_2 - r\\ &= \mathbf{T}_2(L_2 - r) \ \ for\ L_2 - r < k_2\end{aligned}\right\} \quad \cdots\cdots\cdots(11)$$

We see that the multiplication terms are much more simplified. Under this condition we require to calculate only k_2 multiplication terms, others multiplication terms are simply obtained from them. Each multiplication terms require one multiplication operation and we require to calculate only k_2 multiplication terms, so the total number of multiplication operations required under this condition is only k_2.

Condition 4: If $k_1 = k_2 (= k$, say) in condition (3), then

$$\left.\begin{aligned}\mathbf{T}_2(L_2+1-r) &= a^2 r \quad for\ \ r \le k\\ &= T_2(r) \quad for\ \ r > k\end{aligned}\right\} \cdots\cdots\cdots(12)$$

Here, also we need to calculate only k multiplication terms each of which requires only one multiplication.

Condition 5: If the representas were the same, that is

$a_{11} = a_{21} (= a_1), a_{12} = a_{22} (= a_2), \ldots, a_{1k1} = a_{2k2} (= a_k)$, that is, $\mathbf{A}_1 = \mathbf{A}_2 = \mathbf{A}$, say,

$$\mathbf{T}_2(L_2+1-r) = \sum_{i=1}^{t} \mathbf{A}(k+1-i)\mathbf{A}(k-r+i) \cdots\cdots\cdots(13)$$

where $t = r$ if $r \le k$

$= 2k - r$ if $r > k$

Each multiplication term in equation (13) requires t multiplication operations and $t-1$ additions. Writing the equation (13) in the following form can reduce the multiplication and addition operations.

$$\mathbf{T}_2(L_2+1-r) = 2\sum_{i=1}^{t} \mathbf{A}(k+1-i)\mathbf{A}(k-r+i) + (r\%2)\mathbf{A}^2(k-t) \cdots\cdots\cdots(14)$$

where $t = \lfloor r/2 \rfloor$ if $r \le k$ and $= \lfloor k - r/2 \rfloor$ if $r > k$.

From equation (14), we see that the upper limit of the summation is reduced to half and a factor 2 comes as a multiplier of the term. It indicates that the addition and multiplication operations are reduced to approximately half of what they should be in equation (13).

Now we can write square of polynomial of length *k* as follows

$$(A^+)^2 = (a_1 + a_2 + \ldots + a_k)^2 = \sum_{r=1}^{L_2} \mathbf{T}_2(r) \cdots\cdots\cdots (15)$$

where $\mathbf{T}_2(r)$ is given by either equation (13) or equation (14).

Condition 6: If $\mathbf{A}_2 = \mathbf{A}_1^{'}$, i.e., one of the representas is the reverse of the other, then the multiplication term is given by

$$\left.\begin{aligned} \mathbf{T}_2(L_2 + 1 - r) &= \sum_{i=1}^{r} \mathbf{A}(i)\mathbf{A}(k - r + i) \;\; for\; r \le k \\ &= T_2(2k - r) \quad for\; r > k \end{aligned}\right\} \cdots\cdots\cdots (16)$$

Here we need to calculate only the first *k* multiplication terms each requiring r multiplication and $r-1$ addition operations. Here also the multiplication and addition operations required are reduced to a half.

2.4.2 Further reduction in multiplication operations

If the multiplication is to be done manually, then the above given formula for multiplication terms is better. If the multiplication is to be done by computer, then we can further reduce the number of multiplication operations at the expense of addition operations.

We know multiplication terms of $[a_{11}, a_{12}, a_{13}, \ldots, a_{1k_1}] \times [a_{21}, a_{22}, a_{23}, \ldots, a_{2k_2}]$ and $(a_{11} + a_{12} + a_{13} + \ldots + a_{1k_1}) \times (a_{21} + a_{22} + a_{23} + \ldots + a_{2k_2})$ are the same.

Also, we know $(a_{11} + a_{12} + a_{13} + \ldots + a_{1k_1}) \times (a_{21} + a_{22} + a_{23} + \ldots + a_{2k_2}) = \sum_{r=1}^{L_2} \mathbf{T}_2(r)$

where $L_2 = k_1 + k_2 - 1$.

If $k_2 > k_1$, a multiplication term cannot have more than k_2 multiplication operations. Also, we can find $\mathbf{T}_2(r)$ from the previous multiplication terms without doing k_2 multiplication operations in the following way.

$$\mathbf{T}_2(r) = (a_{11} + a_{12} + \ldots + a_{1k_1}) \times (a_{21} + a_{22} + \ldots + a_{2k_2}) - \sum_{i=1}^{L_2} \mathbf{T}_2(i) \quad for\; r \ne k_2 \quad \cdots\cdots (17)$$

From the equation (17), we know that for finding $\mathbf{T}_2(r)$ we require only one multiplication operation and others are addition of the remaining multiplication terms. This concept can be applied in all cases of multiplying two multi-digit numbers except in the case where one or both of the numbers are composed of the same digit to further reduce the number of multiplication operations.

So, the r_{th} multiplication term in equations (14) and (15) can be found in the following way

$$\mathbf{T}_2(r) = (a_1 + a_2 + \ldots + a_k)^2 - \sum_{i=1}^{2k-1} \mathbf{T}_2(i) \quad \text{for} \quad r \neq k \cdots\cdots\cdots(18)$$

For finding $\mathbf{T}_2(k)$ of equations (14) and (15), the respective values of $\mathbf{T}_2(r)$ should be replaced in equation (18). Using equation (18), the number of multiplication operations in calculating multiplication terms using equations (14) and (4) is further reduced by $k-1$ multiplication operations at the expense of $2k$ addition operations.

2.5 Bino's Model of Multiplication and Polynomial Multiplication

We have seen in the previous section that the multiplication of zero order polynomials can be performed easily using Bino's model of multiplication. We can easily extend it to multiplication of two polynomials in general. Suppose p_1 and p_2 are any two polynomials in x of degrees $k_1 - 1$ and $k_2 - 1$, whose coefficients are the elements of two arrays $\mathbf{A}_1$ and $\mathbf{A}_2$ of lengths k_1 and k_2 respectively. That is,

$$p_1 = \mathbf{A}_1(1) + \mathbf{A}_1(2)x + \mathbf{A}_1(3)x^2 + \cdots + \mathbf{A}_1(k_1)x^{k_1-1}$$

$$p_2 = \mathbf{A}_2(1) + \mathbf{A}_2(2)x + \mathbf{A}_2(3)x^2 + \cdots + \mathbf{A}_2(k_2)x^{k_2-1}$$

Then their product will be a polynomial p in x of degree $L-1$ whose coefficients are given by $\mathbf{T}_2$ and hence it can be written as

$$p = p_1 \times p_2 = \sum_{r=1}^{L_2} \mathbf{T}_2(r)x^{r-1} \cdots\cdots\cdots(19)$$

where $L_2 = k_1 + k_2 - 1$ is the length of $\mathbf{T}_2$ which can be obtained from equations (4) or (5) depending on from which side the multiplication starts. This clearly shows that Bino's model of multiplication can be used for polynomial multiplication.

In both equations (4) and (5), the multiplier is the array $\mathbf{A}_2$, the coefficients of p_2. We can make $\mathbf{A}_1$ the multiplier, which is the case of $p_2 \times p_1$. Then the corresponding multiplication terms $\mathbf{T}_2$ are given by equations (4) and (5) respectively.

$$\left.\begin{aligned}
\mathbf{T}_2(L+1-r) &= \sum_{i=1}^{r} \mathbf{A}_1(k_1-r+i)\mathbf{A}_2(k_2+1-i) \quad if \quad r < k_2 \\
&= \sum_{i=1}^{k_2} \mathbf{A}_1(k_1-r+i)\mathbf{A}_2(k_2+1-i) \quad if \quad k_2 \le r \le L-r \\
&= \sum_{i=1}^{L-r} \mathbf{A}_1(k_1-r+i)\mathbf{A}_2(k_2+1-i) \quad if \quad L-r < k_2
\end{aligned}\right\} \cdots\cdots\cdots (20)$$

$$\left.\begin{aligned}
\mathbf{T}_2(r) &= \sum_{i=1}^{r} \mathbf{A}_1(i)\mathbf{A}_2(r+1-i) \quad if \quad r \le k_2 \\
&= \sum_{i=1}^{k_2} \mathbf{A}_1(i)\mathbf{A}_2(r+1-i) \quad if \quad k_2 < r \le L_2 - r \\
&= \sum_{i=1}^{L-r} \mathbf{A}_1(i)\mathbf{A}_2(r+1-i) \quad if \quad L-r < k_2
\end{aligned}\right\} \cdots\cdots\cdots (21)$$

Equation (20) gives the multiplication terms of $p_2 \times p_1$ when the multiplication starts from the right end. And equation (21) gives the multiplication terms of $p_2 \times p_1$ when the multiplication starts from the left end. All these four equations (4), (5), (20), (21) are exactly the same as regards multiplication terms of multiplying two representas $\mathbf{A}_1$ and $\mathbf{A}_2$, which clearly shows that polynomial multiplication and number multiplication process are the same.

2.6 Bino's Model of Multiplication and Convolution

Linear convolution is one of the most important operations used in signal processing. It requires two signals as its input. The input signals may be continuous or discrete. Here we will consider convolution for discrete time signals or sequences only (a discrete signal will be denoted by an array). Convolution is a basic operation in signal processing. It is used to compute auto and cross-correlation functions, to design and implement FIR and IIR digital filters, to solve difference equations and to compute power spectra [59]. Linear convolution can be performed in two ways. One is in the time domain and the other is in the frequency domain. Convolution in the time domain is also known as the direct method. Convolution in

frequency domain is known as indirect method. Various ways of performing discrete convolution are described in [73, 90]. We will describe, in a simpler way, how linear convolution of two discrete signals is performed in direct method.

Let $\mathbf{y} = \mathbf{a} \otimes \mathbf{b}$ denote the result of convolution of two discrete signals $\mathbf{a}$ and $\mathbf{b}$. For clarity in explanation, we will call the sequence that comes after the convolution symbol as the convolver, similar to the multiplier in multiplication. The process of convolution requires two main steps. First, reversing the convolver and placing it at the beginning or end of the signal to be convolved with. Then moving the convolver to the right or left by one position each time depending on where (beginning or end) the convolution starts from and find the sum of the product of the elements of the two signals in the overlapping region.

To illustrate the convolution process, let $\mathbf{a} = [a_1, a_2, a_3, a_4, \cdots, a_m]$ and $\mathbf{b} = [b_1, b_2, b_3, b_4]$ be any two discrete signals. We will find the convolution of $\mathbf{a}$ by $\mathbf{b}$, starting the convolution process from the left end. First reverse the signal $\mathbf{b}$ and place it at the beginning of the signal $\mathbf{a}$ as shown below in Fig.2.5. Then every element of $\mathbf{y}$ is given by the sum of the product of the elements of $\mathbf{a}$ by $\mathbf{b}$ in the overlapping region.

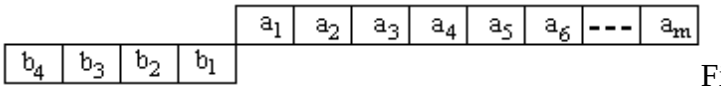


Figure 2.5a

To find $\mathbf{y}(1)$, $\mathbf{b}$ is shifted right by one position and the sum of the product of the elements of the two signals in the overlapping region is found.

Thus , $\mathbf{y}(1) = \mathbf{a}(1)\mathbf{b}(1)$ as only $\mathbf{a}(1)$ and $\mathbf{b}(1)$ overlap as shown in Fig.2.5b.

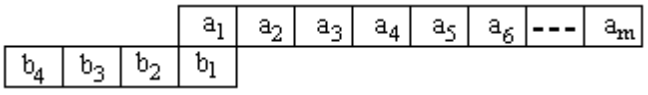


Figure 2.5b

$\mathbf{b}$ is then shifted right by one position again as shown in Fig.2.5c and $\mathbf{y}(2)$ is found as $\mathbf{y}(2) = \mathbf{b}(1)\mathbf{a}(2) + \mathbf{b}(2)\mathbf{a}(1)$, as a_1, a_2 overlap with b_2, b_1 in this position.

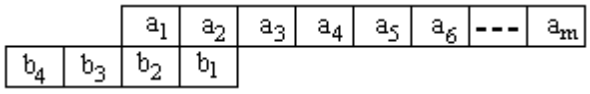


Figure 2.5c

$\mathbf{b}$ is further shifted right by one position as shown in Fig.2.5d and find $\mathbf{y}(3)$ accordingly.

$$\mathbf{y}(3) = \mathbf{b}(1)\mathbf{a}(3) + \mathbf{b}(2)\mathbf{a}(2) + \mathbf{b}(3)\mathbf{a}(1)$$

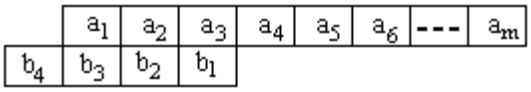


Figure 2.5d

Similarly, $\mathbf{y}(4)$ is found from Fig.2.5e as

$$\mathbf{y}(4) = \mathbf{b}(1)\mathbf{a}(4) + \mathbf{b}(2)\mathbf{a}(3) + \mathbf{b}(3)\mathbf{a}(2) + \mathbf{b}(4)\mathbf{a}(1)$$

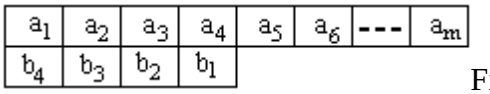


Figure 2.5e

Proceeding in this way, we can write from Fig.2.5f

$$\mathbf{y}(m) = \mathbf{b}(1)\mathbf{a}(m) + \mathbf{b}(2)\mathbf{a}(m-1) + \mathbf{b}(3)\mathbf{a}(m-2) + \mathbf{b}(4)\mathbf{a}(m-3)$$

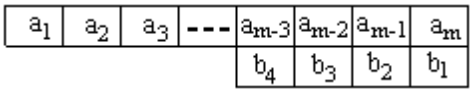


Figure 2.5f

The other terms are found in a similar way.

$$\mathbf{y}(m+1) = \mathbf{b}(2)\mathbf{a}(m) + \mathbf{b}(3)\mathbf{a}(m-1) + \mathbf{b}(4)\mathbf{a}(m-2)$$

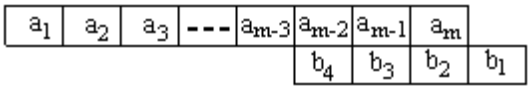


Figure 2.5g

$$\mathbf{y}(m+2) = \mathbf{b}(3)\mathbf{a}(m) + \mathbf{b}(4)\mathbf{a}(m-1)$$

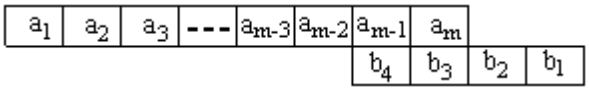


Figure 2.5h

$$\mathbf{y}(m+3) = \mathbf{b}(4)\mathbf{a}(m)$$

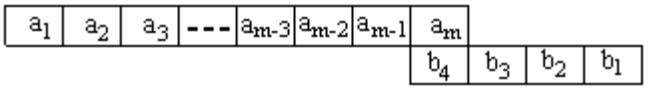


Figure 2.5i

$$\mathbf{y}(m+4) = 0$$

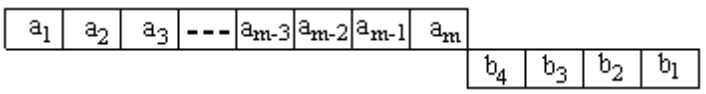


Figure 2.5j

Figure 2.5 Convolution operation using sliding windows

Thus, the length of $\mathbf{y}$ becomes $m+3$. This is how convolution is performed between two sequences. We can also get the same result of convolution by starting the convolution from

the right end, i.e., taking Fig.2.5j as the initial position and then shifting the convolver to the left by one position each time until one arrives at Fig.2.5a position.

The above values of $\mathbf{y}$ can be expressed using equation (3) as

$$\mathbf{y}(r) = \sum_{i=1}^{4} \mathbf{b}(i)\mathbf{a}(r+1-i) \cdots\cdots\cdots (22)$$

Or, by using equation (5) as

$$\left.\begin{aligned} \mathbf{y}(r) &= \sum_{i=1}^{r} \mathbf{b}(i)\mathbf{a}(r+1-i) \quad if \quad r \le 4 \\ &= \sum_{i=1}^{4} \mathbf{b}(i)\mathbf{a}(r+1-i) \quad if \quad 4 < r \le L-r \\ &= \sum_{i=1}^{L-r} \mathbf{b}(i)\mathbf{a}(r+1-i) \quad if \quad L-r < 4 \end{aligned}\right\} \cdots\cdots\cdots (23)$$

This clearly shows that polynomial multiplication and convolution represent the same process. In other words, linear convolution and Bino's model of multiplication also represent the same process. Thus, in general we can say, if $\mathbf{A}_1$ and $\mathbf{A}_2$ are any two sequences of lengths k_1 and k_2 respectively, then the convolution of the two sequences is given by $\mathbf{T}_2$ obtainable from any of the four equations (4), (5), (20) or (21), depending on from which end the convolution starts. Also, the convolution is a commutative process as multiplication is. That is,

$$\mathbf{A}_1 \otimes \mathbf{A}_2 = \mathbf{A}_2 \otimes \mathbf{A}_1 = \mathbf{T}_2$$

Usually, convolution is performed between two sequences. For the convolution of three or more sequences, we can use the formulations of three or more representas described in Section 2.8.

2.7 Fast Fourier Transform for Large Number Multiplication

From Bino's model of multiplication, we know that convolution can be used for number multiplication. As convolution has close relation with Fourier transforms, we can use Fourier transforms for finding the result array of two representas. Also, we know that FFT can be used for fast convolution. Hence FFT can be used for fast large number multiplication. The procedure in our case is given below.

1. Convert the two large numbers as representas with appropriate base.
2. Find the result array from convolution of the two representas using FFT.
3. Convert the result array in Step-2 as representa with the corresponding base.

The representa in Step-3 represents the result of the two large numbers.

FFT is found to be slower as compared to the conventional way of multiplication in two cases. First, when the lengths of the representas are less than 128. And secondly, when one of the representas is much shorter as compared to the other. Under these two conditions, it is advisable to use Bino's model of multiplication. FFT can be used for faster multiplication when the lengths of the two representas are more or less equal and greater than 128. Use of FFT for large number multiplication is also described in [168]. But it is based on polynomial multiplication and is not as fast as our method described here.

2.8 Multilevel Multiplication

Bino's model of multiplication can also be used for the multiplication of three or more representas or arrays. The multiplication of three or more representas or arrays is known as multilevel multiplication. They are just an extension of level-2 multiplication.

2.8.1 Length of level-*n* result array

We have seen that the length of level-2 result-array for multiplication of two representas or arrays of lengths k_1 and k_2 is given by $L_2 = k_1 + k_2 - 1$. Bino's model of multiplication can be extended up to a desired level. The multiplication of three representas or arrays of lengths k_1, k_2 and k_3 is called level-3 multiplication. The length of the result array of a level-3 multiplication of three representas or arrays is given by $L_3 = k_1 + k_2 + k_3 - 2$. Proceeding in this way, the length of level-*n* result array of multiplication of n representas or arrays of lengths $k_1, k_2, k_3, \cdots, k_n$ is given by the following equation.

$$L_n = k_1 + k_2 + k_3 + \cdots + k_n - (n-1) \quad \cdots\cdots\cdots (24)$$

If the lengths of all representas or arrays are equal, i.e., $k_1 = k_2 = k_3 = \cdots = k_n = (k, say)$, then the above equation becomes

$$L_n = n(k-1) + 1 \cdots\cdots\cdots (25)$$

From equations (24) and (25), we can easily find the number of multiplication terms in a result array at any level.

2.8.2 Level-3 multiplication

The multiplication of three numbers or representas is known as level-3 multiplication. The Level-3 multiplication is obtained using two successive multiplication of level-2 multiplications. First we multiply the first two representas representing the first two numbers. Their result array can be thought as a multiplicand which is again multiplied with the third representa representing the third number.

Let $\mathbf{A}_1 = [a_{11}, a_{12}, a_{13}, \ldots, a_{1k_1}]$, $\mathbf{A}_2 = [a_{21}, a_{22}, a_{23}, \ldots, a_{2k_2}]$, $\mathbf{A}_3 = [a_{31}, a_{32}, a_{33}, \ldots, a_{3k_3}]$ be any three representas of base b having lengths k_1, k_2 and k_3 respectively. Then, according to Bino's model of multiplication, starting from the right end, we have

$$\mathbf{A}_1 \times \mathbf{A}_2 \times \mathbf{A}_3 = (\mathbf{T}_3)_b = \sum_{r=1}^{L_3} b^{L_3 - r} \mathbf{T}_3(r) \cdots\cdots\cdots (26)$$

where $L_3 = (k_1 + k_2 + k_3) - 2$ is the total number of multiplication terms in level-3 multiplication of three numbers of length k_1, k_2 and k_3. And each term of $T_3(r)$ can be recursively obtained from the following relation.

$$\mathbf{T}_3(L + 1 - r) = \sum_{i=1}^{k3} \mathbf{A}_3(k_3 + 1 - i)\mathbf{T}_2(r + 1 - i) \quad \cdots\cdots\cdots (27)$$

Replacing the $\mathbf{T}_2$ terms in the right hand side of equation (27), we can write

$$\mathbf{T}_3(L + 1 - r) = \sum_{i=1}^{k3} \mathbf{A}_3(k_3 + 1 - i) \sum_{i_1=1}^{k_2} \mathbf{A}_2(k_2 + 1 - i_1)\mathbf{A}_1(k_1 - r - 1 + i + i_1) \cdots\cdots (28)$$

If we start the multiplication process from the left end, then we can write equations (27) and (28) respectively as

$$\mathbf{T}_3(r) = \sum_{i=1}^{k3} \mathbf{A}_3(i)\mathbf{T}_2(r + 1 - i) \quad \cdots\cdots\cdots (29)$$

$$\mathbf{T}_3(r) = \sum_{i=1}^{k1} \mathbf{A}_1(i) \sum_{i_1=1}^{k_2} \mathbf{A}_2(i_1)\mathbf{A}_3(k_1 + r + 1 - i - i_1) \quad \cdots\cdots\cdots (30)$$

Once we get $\mathbf{T}_3$ we can get the following results.

$$\left.\begin{aligned}\mathbf{A}_1^{'}\times\mathbf{A}_2^{'}\times\mathbf{A}_3^{'} &= \sum_{r=1}^{L_3} b^{r-1}\ \mathbf{T}_3(r)\\ &= \sum_{r=1}^{L_3} b^{L_3-r}\ \mathbf{T}_3^{'}(r)\end{aligned}\right\}\cdots\cdots\cdots(31)$$

$$A_1^{+}\times A_2^{+}\times A_3^{+} = \sum_{r=1}^{L_3}\mathbf{T}_3(r)\ \cdots\cdots\cdots(32)$$

If the representas were arrays, then we can write the level-3 convolution as

$$\mathbf{A}_1\otimes\mathbf{A}_2\otimes\mathbf{A}_3=\mathbf{T}_3\cdots\cdots(33)$$

Thus, from equation (28) or (30), we can easily perform multiplication of three numbers, polynomials or arrays.

2.8.3 Level-*n* multiplication

We have derived the result arrays of level-2 and level-3 multiplication in different forms, depending on the side multiplication starts from. As Bino's model of multiplication can be drawn from either the left or the right side, we have two kinds of equations. Let $\mathbf{A}_1=[a_{11},\, a_{12},\, a_{13},\, \ldots, a_{1k_1}]$, $\mathbf{A}_2=[a_{21},\, a_{22},\, a_{23},\, \ldots, a_{2k_2}]\ldots.$ $\mathbf{A}_n=[a_{n1},\, a_{n2},\, a_{n3},\, \ldots, a_{nk_n}]$ be *n* different representas with base b having lengths $k_1, k_2, k_3, \cdots, k_n$ respectively. As usual, their multiplication term can be recursively obtained from the following relation.

$$\mathbf{T}_n(L+1-r)=\sum_{i=1}^{k_n}\mathbf{A}_{k_n}(k_n+1-i)\mathbf{T}_{n-1}(r+1-i)\cdots\cdots\cdots(34)$$

The level-*n* result array is written in terms of level-(*n*-1) result array. Successively representing the result arrays on the right hand side with successive lower level result arrays, we can write the level-*n* result array in terms of the representas as

$$\left.\begin{aligned}&\mathbf{T}_n(L+1-r)=\sum_{i=1}^{k_n}\mathbf{A}_n(k_n+1-i)\sum_{i_1}^{k_{n-1}}\mathbf{A}_{n-1}(k_{n-1}+1-i_1)\cdots\cdots\cdots\\ &\sum_{i_{n-2}}^{k_2}\mathbf{A}_2(k_2+1-i_{n-2})\mathbf{A}_1(k_1-r-(r-1)+i+i_1+i_2+\cdots+i_{n-2})\end{aligned}\right\}\ \cdots\cdots\cdots(35)$$

If we start finding the multiplication terms from the left-hand side, i.e., if we begin multiplication from the left-hand side, then the general level-*n* multiplication term becomes

$$\mathbf{T}_n(r) = \sum_{i=1}^{kn} \mathbf{A}_{k_n}(i)\mathbf{T}_{n-1}(r+1-i) \cdots\cdots\cdots (36)$$

Expanding (36), we have

$$\mathbf{T}_n(r) = \sum_{i=1}^{k_1} \mathbf{A}_1(i) \sum_{i_1}^{k_2} \mathbf{A}_2(i_1) \cdots\cdots \sum_{i_{n-2}}^{k_{n-2}} \mathbf{A}_2(i_{n-2})\mathbf{A}_1(r-(n-1)-i-i_1-i_2-\cdots-i_{n-2}) \quad \cdots (37)$$

From any of the two equations (35), (37), we can find the multiplication terms of result array at any level. Once we find the level-*n* multiplication terms, we can have the following results.

From the level–*n* result array, we can write the following equations.

Direct multiplication of *n* representas :

$$(\mathbf{A}_1 \times \mathbf{A}_2 \times \cdots \times \mathbf{A}_n)_b = \sum_{i=1}^{L_n} b^{L_n - i} \mathbf{T}_n(i) \cdots\cdots\cdots (38)$$

Reverse multiplication of *n* representas :

$$(\mathbf{A}_1^{'} \times \mathbf{A}_2^{'} \times \cdots \times \mathbf{A}_n^{'})_b = \sum_{i=1}^{L_n} b^{i-1} \mathbf{T}_n(i) \cdots\cdots\cdots (39)$$

If the representas were arrays,

$$\mathbf{A}_1 \otimes \mathbf{A}_2 \otimes \cdots \otimes \mathbf{A}_n = \mathbf{T}_n \cdots\cdots\cdots (40)$$

If each element of every representa is separated by a positive sign, then

$$A_1^{+} \times A_2^{+} \times \cdots \times A_n^{+} = \sum_{i=1}^{L_n} \mathbf{T}_n(i) \cdots\cdots\cdots (41)$$

Equation (35) or (37) gives the general multiplication term for level-n multiplication or convolution. The multiplication terms may be more easily obtained when all the n multiplicands are the same.

2.9 Discussion and Remarks

Bino's model of multiplication is a generalized model for multiplication of numbers, polynomial and 1-D arrays or sequences. It is a symmetric model in the sense that we can start the multiplication process from the left or the right end. Using this model we can find the result array of multiplication of two representas or arrays easily. We have discussed how the result of multiplication of two representas could be obtained manually from the result array using backward slant addition and forward slant addition. We have derived various

formulations for multiplication of two numbers or in general of two representas depending on the side the multiplication operation starts from. Also, from the model we know that if we know the multiplication terms of multiplication of two numbers, then we can find the result of multiplication of the corresponding reversed numbers from the multiplication terms without actually doing multiplication again. We have also shown that linear convolution, polynomial multiplication can also be represented by Bino's model of multiplication. In other words, Bino's model of multiplication is the common model of representing number multiplication, polynomial multiplication and convolution, which are usually dealt with in different ways. Thus, as in multiplication of two representas, we can have four different forms of convolution depending on the side convolution starts from. The Bino's model of multiplication simplifies the convolution operation. This allows us to do element-wise analysis of convolution of two arrays. This helps us immensely in understanding the perfect reconstruction theory of existing 2-channel filter bank scheme and to the developments of generalized signal decomposition and reconstruction schemes proposed in chapter 4 and 5. As the convolution process can also be done with fast Fourier transforms, we have described the process of how multiplication can be done using FFT and have indicated when it can be used for faster convolution or multiplication. The multiplication process is extended first to multiplication of three numbers and then to multiplication of n numbers, in general. The same can be applied to find trinomial expansions and in general polynomial expansions from which we can find various types of convolution triangles [35, 53, 92, 126] to design symmetric averaging filters.

2.10 Conclusions

Bino's model of multiplication is a generalized way of multiplication. It can exactly represent the multiplication process involved in number multiplication, polynomial multiplication and array multiplication or convolution and hence it unifies the three multiplication processes. This allows us to perform large number multiplication fast using representas. It also represents the FIR filtering operation and helps in a better understanding of the filtering process. Moreover, multilevel multiplication, polynomial expansion or convolution can easily be performed using Bino's model of multiplication.

Chapter 3

ISITRA FOR SIGNAL DECOMPOSITION AND RECONSTRUCTION

Abstract: *Proposed here is a generalized transform entitled ISITRA for subband signal decomposition and reconstruction. Unlike in wavelet and filter bank decomposition schemes, here a signal can be decomposed into (a) two approximations or (b) two details or (c) one approximation and one detail using different decomposition filters which are not necessarily of QMF type. Here the issue of perfect reconstruction in a subband decomposition scheme is well defined in terms of member condition, a condition to be satisfied by a set of decomposition filters and their corresponding reconstruction filters. We define a PRF set as a set of decomposition and reconstruction filters that satisfy the member condition. Also, we define a framework of PRF sets as a pattern of PRF sets from which infinitely many actual PRF sets can easily be generated. There are various ways for finding different frameworks of PRF sets which provide us much more flexibility in the choice of decomposition and reconstruction filters than the wavelet and filter bank schemes. The theory of ISITRA is simple as it does not require any complex transform such as Z-transform in filter bank or Fourier transform and can be easily implemented in a computer. Moreover, perfect reconstruction multi-channel filter bank and transmultiplexer can easily be obtained from the 2-channel case.*

3.1 Introduction

In the last few decades we have witnessed a strong wave of wavelets fluxed in various fields of research and application domains. The most apparent strength of wavelet seems to lie in its multi-resolution signal representation and its ability to produce non-redundant

transformed coefficients in its discrete version based on 2-channel filter bank scheme. Such features are found to be suitable for data compression and hence it is commonly used for image compression [12, 42, 109,110,112]. However, it has also been used in various other applications like computer vision [108], pattern recognition [164], statistical parameter estimation [45], denoising [36, 44] etc. If we trace the history of the popular scheme of so called discrete wavelet transform (DWT), we will want to say that it is a hybrid scheme of wavelet and filter bank. It is not a direct discrete version of the continuous transform (CWT), which Grossman et al [51] introduced for multi-resolution signal representation through dilation and translation of a basis function known as mother wavelet. In DWT, multi-resolution is achieved using two basis functions respectively known as scaling function (or father wavelet) and wavelet function (or mother wavelet). It seems to be a mystery why scaling function would be required in DWT when there is no requirement of the same in CWT. Interesting enough, scaling function has more importance than wavelet function in DWT, in the sense that most features of a signal are in the transform coefficients of scaling function. One might imagine or assume that scaling function as a special case of wavelet function as the former can be obtained from the latter in orthogonal wavelet case through a relation known as quadrature mirror filter (QMF) relation. However, if we study further, we see that there is no relationship between scaling function and wavelet function in bi-orthogonal wavelets. But still multi-resolution is possible in the case of bi-orthogonal wavelets. It is not correct to assume that only mother wavelet or any of its derivatives is required for multi-resolution in the line that multi-resolution in CWT is achieved from only the mother wavelet.

Another mystery in DWT is the issue of inversiblity or perfect reconstruction. Theoretically, perfect reconstruction of CWT is given on admissibility condition [39], even though it is not possible to get perfect reconstruction in CWT in practice. This could be a reason why CWT is not so popular. On the other hand, perfect reconstruction is possible in DWT. Multi-resolution will have no role in perfect reconstruction, because perfect reconstruction is not possible in CWT which is inherent with multi-resolution. The perfect reconstruction in DWT is possible because DWT filters satisfy perfect reconstruction theory of 2-channel filter bank. To ensure perfect reconstruction in orthogonal filter bank, making the low pass and high pass decomposition filters are to be of QMF type is sufficient. The same relation is also required between a scaling function and a wavelet function to ensure perfect

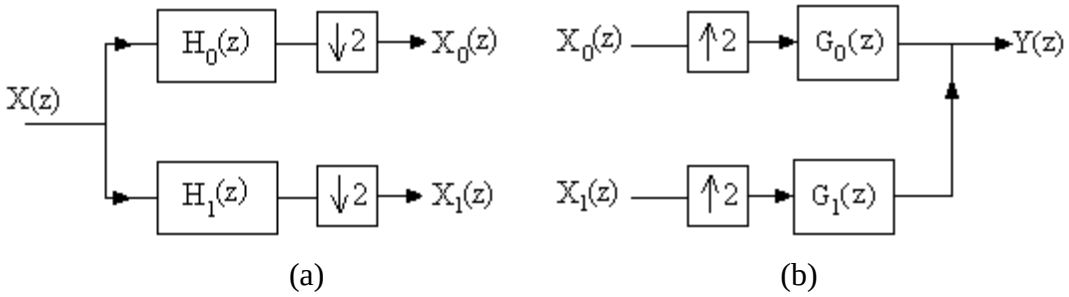
reconstruction. So, the main reason behind introducing scaling function in DWT is to establish a relation between wavelet and filter bank, which at the same time would allow perfect reconstruction. The main advantage of such an establishment is the possibility of using short length perfect reconstruction wavelet filters in the filter bank scheme. This makes DWT very popular and the term wavelet transform becomes synonymous to DWT.
DWT is based on 2-channel filter bank, we analyze step by step of decomposition and reconstruction scheme of the filter bank using Bino's model of multiplication. We could establish the necessary and sufficient conditions for perfect reconstruction in 2-channel filter bank scheme [125]. We could also know that the condition of perfect reconstruction is dependent on how reconstruction is performed. That is, different reconstruction schemes give rise to different conditions for perfect reconstruction. The existing perfect reconstruction theory of 2-channel filter bank is also one possible way of reconstruction. The transform ISITRA (Indian Statistical Institute Transform) proposed in this chapter provides a generalized model for the subband decomposition and reconstruction scheme. We will briefly discuss below 2-channel filter bank and its shortcomings, which may be useful for a better understanding of ISITRA and the differences between ISITRA and filter bank.

In Section 3.2, we describe the basic features of ISITRA and present a comparative study of computational complexities of filter bank (in direct and polyphase form) and ISITRA. In Section 3.3, the decomposition and perfect reconstruction schemes and ways of obtaining perfect reconstruction filter (PRF) sets of ISITRA are described. Two cases of ISITRA, namely, ISITRA-1 and ISITRA-2, are then considered. In ISITRA-1, the two decomposition and two reconstruction filters are obtained from a single array of length L while in ISITRA-2, they are obtained from two separate arrays of length L. In Section 3.4, first ISITRA-1 is described and then it is shown how Daubechies' orthogonal wavelets become special cases of ISITRA-1. In Section 3.5, ISITRA-2 is described and it is shown how Daubechies' bi-orthogonal wavelets become special cases of ISITRA-2. In Section 3.6, the multilevel (both half and full) decomposition and reconstruction schemes of ISITRA are presented. In Section 3.7, the multi-resolution decomposition scheme of ISITRA is discussed where the decomposed components of a signal at any decomposition level can be obtained through a single level decomposition of the original signal. This is done using a special filter called effective filter which depends on the decomposition filter used at every level and the depth

of the decomposition level. This saves a considerable computation time. In Section 3.8, multi-channel decomposition and reconstruction schemes of ISITRA are presented. In Section 3.9, a multi-channel transmultiplexer of ISITRA and the perfect reconstruction theory of 2-channel transmultiplexer are described. In Section 3.10, decomposition and reconstruction schemes of ISITRA in 2 dimensions are discussed. In Section 3.11, multilevel (both half and full) decomposition and reconstruction schemes of ISITRA in 2 dimensions are discussed. Discussions are given in Section 3.12 and conclusions in Section 3.13.

3.1.1 Filter bank scheme and its computational complexity

In the filter bank scheme, a signal is decomposed into two components using a high pass filter and a low pass filter. Then each of the filtered outputs is down sampled by a factor of 2 as shown in Fig.3.1a, to give the outputs. The decomposition filters need to be of QMF or CQF (conjugate quadrature filters) type to ensure perfect reconstruction. The down-sampling operations performed in the decomposition process are useful for data reduction. In most of the literature on filter bank, it is mentioned that the down-sampling in 2-channel filterbank introduces aliasing effect. But that is not the case as explained in [120] and later in ISITRA. The process of down-sampling is required for both perfect reconstruction and multi-resolution analysis. The decomposition scheme in Fig.3.1a performs unnecessary computation in the filtering operations, which would be simply discarded in the succeeding down-sampling process. As filter bank is usually studied using *z*-transform, signals and filters are given in *z*-domain.



(a) (b)

Figure 3.1 (a) Filter bank decomposition (b) Filter bank reconstruction

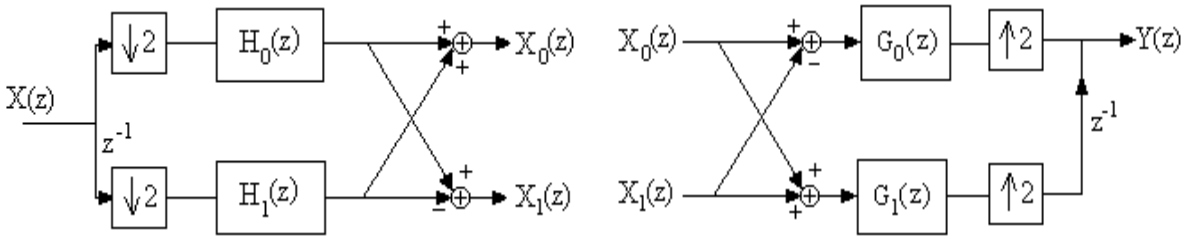
Suppose the input signal $X(z)$ is a polynomial of degree $(m-1)$ and the decomposition filters are polynomials of degree $(L-1)$. Each of the filtered outputs will be a polynomial of

degree $(L+m-2)$ with $(L+m-1)$ coefficients. Consider now that the computation of a coefficient of a filtered output requires L multiplication operations and $(L-1)$ addition operations. Then the arithmetic operations in the computation of a filtered output are $(L+m-1)*L$ multiplication and $(L+m-1)*(L-1)$ addition operations. Down-sampling by 2 in the succeeding stage rejects every odd or even coefficients of a filtered output thereby making the size of the decomposed component $X_0(z)$ or $X_1(z)$ reduced to $\lceil (L+m-1)/2 \rceil$.

In the reconstruction process, the decomposed components are up-sampled by 2 as in Fig.3.1b, thus making their length $(L+m-1)$ before filtering with the reconstruction filters. The reconstruction filters are of polynomial of degree $(L-1)$, a reconstruction filtered output would be a polynomial of degree $(2L+m-3)$. Considering that the computation of a coefficient of the filtered output requires L multiplication operations and $(L-1)$ addition operations, $(2L+m-2)*L$ multiplication operations and $(2L+m-2)*(L-1)$ addition operations would be required for the computation of a filtered output. The two filtered outputs are again added together. This requires $(2L+m-2)$ additions, both being polynomials of degree $(2L+m-3)$, to finally give the reconstructed signal $Y(z)$. Thus, the total numbers of multiplication and addition operations in the reconstruction process are $(2L+m-2)*2L$ and $(2L+m-2)*(2L-1)$ respectively.

The computationally efficient polyphase scheme [29, 48] of 2-channel filter bank is shown in Fig.3.2. In the decomposition process, the signal is first divided into two, even and odd, components using two down-samplers, one coupled with a delay operator. Multiplication of the signal which is a polynomial of degree $(m-1)$, with z^{-1} requires m multiplication operations. The number of coefficients in each of the components (even or odd) becomes $m/2$. These components are filtered with the two decomposition filters of length L, yielding outputs of length $L+m/2-1$. Now, the computation of a coefficient of filtered output requires L multiplication and $(L-1)$ addition operations. Then the computation of a filtered output requires $(L+m/2-1)*L$ multiplication and $(L+m/2-1)*(L-1)$ addition operations. The two filtered outputs are added together to get the component $X_0(z)$ and one

of them is subtracted from the other to give the component $X_1(z)$. Since the two filtered outputs are polynomials of length $L+m/2-1$, addition of the two outputs actually requires $(L+m/2-1)$ additions. Thus, the total number of addition (as well as multiplication) operations in the computation of $X_0(z)$ or $X_1(z)$ is $(L+m/2-1)*L$.



(a) (b)

Figure 3.2. (a) Polyphase decomposition (b) Polyphase reconstruction

In the reconstruction process, first the decomposed components of length $(L+m/2-1)$ are added and subtracted before inputting to the corresponding reconstruction filters. The two inputs are then passed through the reconstruction filters to give outputs of length $(2L+m/2-2)$. Thus $(L-1)*(2L+m/2-2)$ addition operations and $L*(2L+m/2-2)$ multiplication operations are required to find a filtered output. The two filtered outputs are then up-sampled by 2 making the length of each output doubled, that is, $(4L+m-4)$. Then one of the up-sampled components is multiplied with a delay operator, which requires $(4L+m-4)$ multiplication operations and as many addition operations to finally give the reconstructed signal $Y(z)$.

In both filter banks (direct and polyphase), the equality of $X(z)$ and $Y(z)$ depends on how the decomposition filters $H_0(z)$, $H_1(z)$ and the reconstruction filters $G_0(z)$, $G_1(z)$ are chosen. With the proper choice of decomposition and the reconstruction filters, it is possible to make $X(z)$ equal to $Y(z)$, in that case, the corresponding filter bank scheme is known as perfect reconstruction filter bank. Several scientists worked on the perfect reconstruction theory of filter banks. Mention may be made of Smith-Barnwell [127], Mintzer [89], Vaidyanathan [148, 150], Vetterli [153, 155, 157]. Perfect reconstruction theory of filter banks are also dealt with in [5, 29, 48, 152, 159].

Later on wavelets people connected wavelet theory with filter bank [34, 39, 81]. For that purpose, they introduced an extra wavelet function known as scaling function to correspond to the low pass filter in filter bank. With this introduction the inversible discrete wavelet transforms can be developed based on the perfect reconstruction theory of filter bank. Perfect reconstruction theory of 2-channel filter bank involves reconstruction filters. However, reconstruction filters are not explicitly studied about their roles in perfect reconstruction in DWT, nor any explicit names were given to wavelet functions that would correspond to the reconstruction filters. There are certain conditions to be satisfied by wavelet filters. The decomposition filters $\mathbf{h}_0$, corresponding to the scaling function, and $\mathbf{h}_1$ corresponding to wavelet function, should satisfy the following equation.

$$\left.\begin{aligned} \sum \mathbf{h}_0(n) &= \sqrt{2} \\ \sum \mathbf{h}_1(n) &= 0 \end{aligned}\right\} \quad \cdots\cdots\cdots (1)$$

Equation (1) is neither a necessary condition for perfect reconstruction nor a sufficient condition. For perfect reconstruction two different conditions are separately required for orthogonal and bi-orthogonal cases. For orthogonal wavelet filters, the following relation is required to ensure perfect reconstruction.

$$\left.\begin{aligned} \mathbf{h}_1(n) &= (-1)^n \mathbf{h}_0(L+1-n) \\ \mathbf{g}_0(n) &= \mathbf{h}_0(L+1-n) \\ \mathbf{g}_1(n) &= \mathbf{h}_1(L+1-n) \end{aligned}\right\} \quad \cdots\cdots\cdots (2)$$

This condition is generally known as QMF relation in filter bank community [5, 48,152]. Usually the first equation in (2) is used to specify the QMF relation. But for perfect reconstruction, the first relation is not sufficient. The relation of the reconstruction filters with the decomposition filters is also required and hence the reconstruction filters $\mathbf{g}_0$ and $\mathbf{g}_1$ are also given in equation (2). For bi-orthogonal filters, the relationship between filters is given as

$$\left.\begin{aligned} \mathbf{g}_0(n) &= (-1)^{n-1} \mathbf{h}_1(L+1-n) \\ \mathbf{g}_1(n) &= (-1)^n \mathbf{h}_0(L+1-n) \end{aligned}\right\} \quad \cdots\cdots\cdots (3)$$

The conditions given above are sufficient conditions to get perfect reconstruction in 2-channel filter bank scheme. But they are not a necessary condition. Later on, we will find

various decomposition and reconstruction filters which do not satisfy the above conditions but give perfect reconstruction. Also, it is not straightforward to find filters that satisfy the above conditions. Neither equation 1 nor 2 suggests anything about how to find filter h_0, which is required to find h_1, g_0, g_1. Also, equation 3 does not suggest how to find h_0, h_1 which are required to find g_0, g_1.

Some drawbacks of the filter bank or DWT

1) Finding perfect reconstruction filters is difficult.
2) Perfect reconstruction issues are not well addressed. There are several possible ways of perfect reconstruction in the 2-channel filter bank. Mainly QMF structure is addressed.
3) The condition for perfect reconstruction being studied in *z*-domain is not very helpful for finding filter coefficients.

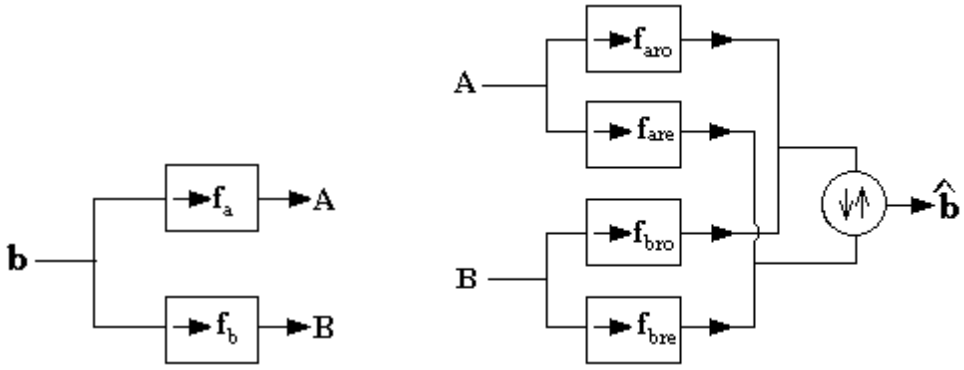
3.2 ISITRA

ISITRA is a new generalized transform for signal decomposition and reconstruction. ISITRA is developed to overcome the drawbacks in a filter bank scheme. Unlike in wavelets or filter-bank decomposition scheme where a signal is always decomposed into an approximation and a detail, here a signal can be decomposed into all possible ways. That is, a signal can be decomposed into an approximation and a detail, or into two details or into two approximations and the coefficients of the filters can be chosen quite arbitrarily. Such arbitrariness in choosing the filter coefficients is a unique feature of ISITRA.

ISITRA is similar to the filter bank scheme in its decomposition scheme as shown in Fig.3.3a. A signal array $\mathbf{b}$ of length m is decomposed into two component arrays $\mathbf{A}$ and $\mathbf{B}$ using the decomposition filter coefficients $\mathbf{f}_a$ and $\mathbf{f}_b$ of length L. The arrow symbol that precedes the filter denotes advancing of one index which avoids performing extra multiplication operations in the filtering process. In other words, we skip alternate data points of the signal in the computation of a decomposed component in the filtering process. The length of each decomposed component thus becomes $m/2+L/2-1$. Now, the computation of a coefficient of a decomposed component requires L multiplications and $(L-1)$ additions. As a result, $(m/2+L/2-1)*L$ multiplications and $(m/2+L/2-1)*(L-1)$ additions will be required for the computation of a decomposed component. A significant

amount of computation would be saved if the length of the decomposition filters is large. Here we do not require down-sampling. It is done in the filtering process itself.

The reconstruction scheme of ISITRA, shown in Fig.3.3b, is computationally more advantageous. Here, we do not up sample the decomposed components. Instead, we do the filtering operation with only the halves of the length of the reconstruction filters. Let $\mathbf{f}_{aro}$ denote the odd parts and $\mathbf{f}_{are}$, the even parts of the reconstruction filter $\mathbf{f}_{ar}$. Similar is the case for the other reconstruction filter $\mathbf{f}_{br}$. Filtering of a decomposed component with even or odd parts of a reconstruction filter requires $L/2*(m/2+L/2-1)$ multiplications and $(L/2-1)*(m/2+L/2-1)$ additions. Addition of the outputs of the two odd filters gives the odd indexed components of the reconstructed signal. And addition of the two outputs of the even filters gives the even indexed components of the reconstructed signal. Thus the numbers of multiplication and addition operations required in the computation of odd components or even components of the reconstructed signal are respectively $L*(m/2+L/2-1)$ and $(L-1)*(m/2+L/2-1)$. The odd and even components are simply inter-merged to obtain the reconstructed signal. It is denoted by the symbol  in the Fig.3.3b. In actual computation this requires no arithmetic operation.



(a) (b)

Figure.3.3 (a) ISITRA decomposition (b) ISITRA reconstruction

A summary of the arithmetic operations required during decomposition and reconstruction processes in direct filter bank, its polyphase form and ISITRA scheme is given in Table 3.1.

Subband Scheme	Total number of		Extra operations Required
	Multiplications	Additions	
1. Direct FB	$2L*(3L+2m-3)$	$2L*(3L+2m-3)-(4L+3m-4)$	Up-sampling, Down-sampling
2. Polyphase	$2L*(3L+m-2)+m-2$	$2L*(3L+m-1)+m-4$	Up-sampling, Down-sampling , Delay Operators
3. ISITRA	$2L*(L+m-2)$	$2(L-1)*(L+m-2)$	None

Table 3.1 Number of multiplication and addition operations in different subband schemes.

As compared to the direct and polyphase filter bank schemes, ISITRA is simpler and requires less number of computational operations. Also, neither a down-sampler in the decomposition process nor an up-sampler in the reconstruction process is required. In both the filter bank schemes described above, the equality of $X(z)$ and $Y(z)$ is much dependent on the decomposition and the reconstruction filters. From the perfect reconstruction theory of 2-channel filter bank using polyphase representation it is found that the orthogonal decomposition and reconstruction filters should be of two types, viz, QMF type or CQF type to ensure perfect reconstruction [48, 127, 152]. The perfect reconstructibilty of orthogonal Daubechies' wavelet filters can also be interpreted as its decomposition and reconstruction filters being of QMF type. But they are not a necessary condition for filters to ensure perfect reconstruction in 2-channel filter bank [121]. We will also show in the Section 3, that filters being QMF and CQF are not the only ways to achieve perfect reconstruction in a subband decomposition scheme. Moreover, there is more flexibility in ISITRA in choosing filter coefficients as compared to polyphase filter bank, which will be clear in the following section.

3.3 Decomposition and Reconstruction Schemes of ISITRA

In this section we will describe the theoretical and computational aspects of the decomposition and perfect reconstruction schemes of ISITRA. We consider here single level decomposition of a 1-dimensional signal or array but it can be easily extended to 2-dimensional signals or arrays. In the derivation, bold letters denote arrays and others scalars. Similar to 2-channel filter-bank decomposition scheme, here also we decompose an array or

a signal into two components using two arrays known as decomposition filters. In the former case, one of the decomposed components corresponds to approximation or low-pass component and the other to detail or high pass components, and the filters are basically of QMF types. But in our case, we have a wide choice of decomposition filters, filters need not be of QMF type and hence we cannot specify a decomposed component as detail or approximation as both may correspond to approximations or to details. Thus instead of referring a decomposed component as an approximation or a detail, we would simply refer them as **A** or **B** component. The decomposition scheme is simple and is as follows.

3.3.1 Decomposition scheme of ISITRA

Let $\mathbf{f}_a$ and $\mathbf{f}_b$ be any two decomposition-filters of even length L. Then the decomposition of a signal $\mathbf{b}$ of size *m* into two components **A** and **B**, is as follows

$$\mathbf{A}(n) = \sum_{i=1}^{L} \mathbf{f}_a(i)\mathbf{b}(2n+1-i) \cdots\cdots\cdots (4)$$

$$\mathbf{B}(n) = \sum_{i=1}^{L} \mathbf{f}_b(i)\mathbf{b}(2n+1-i) \cdots\cdots\cdots (5)$$

where $n = 1,2,3,\cdots,\ len_1$.

The above equations (4) and (5) are based on Bino's model of multiplication beginning from the left end, same in form as in equation (3) of Chapter 2 except the factor 2 used for skipping alternate data points. The length of each decomposed component in the above decomposition scheme is $len_1 = \lceil m/2 \rceil + L/2 - 1$. The subscript '1' in len_1 indicates the first level decomposition.

In the above decomposition equations, the signal is known to us but the filters used are unknown. How do we find the decomposition filters? What are their characteristics? Can any two arrays of even length L be used as decomposition filters? For perfect reconstruction from the decomposed components, the coefficients of the decomposition filters should satisfy certain condition known as *member condition*. They are generally obtained from the reconstruction schemes.

Definition 3.1 A member condition is a set of equations to be satisfied by a set of decomposition and their corresponding reconstruction filters in order to ensure perfect

reconstruction in a subband decomposition scheme. They are generally obtained from the reconstruction scheme**.**

Definition 3.2 A PRF (Perfect Reconstruction Filter) set is a set of decomposition and the reconstruction filters which satisfies a member condition. A PRF set can be used in the decomposition and the reconstruction scheme of a subband decomposition scheme.

3.3.2 Reconstruction scheme of ISITRA

Any reconstruction is usually done using reconstruction filters. There are various ways of reconstruction of the signal from its decomposed components. The reconstruction scheme used by the wavelet and filter bank community is one of them. The decomposition and the reconstruction filters are different for different reconstruction schemes. We discuss here four different types of reconstruction schemes.

Reconstruction Scheme-1:

This reconstruction scheme is done in time domain.

Let $\mathbf{f}_{ar}$ and $\mathbf{f}_{br}$ be the corresponding reconstruction filters of even length L. Then from equations (4) and (5), we get the following using the odd terms of the reconstruction filters.

$$\sum_{i=1}^{L/2}\{\mathbf{f}_{ar}(2i-1)\mathbf{A}(n+1-i)+\mathbf{f}_{br}(2i-1)\mathbf{B}(n+1-i)\}$$

$$=\sum_{r=1}^{L-1}[\sum_{i=1}^{r}\{\mathbf{f}_{a}(2i-1)\mathbf{f}_{ar}(2r+1-2i)+\mathbf{f}_{b}(2i-1)\mathbf{f}_{br}(2r+1-2i)\}\mathbf{b}(2n+2-2r)+$$

$$\sum_{i=1}^{r}\{\mathbf{f}_{a}(2i)\mathbf{f}_{ar}(2r+1-2i)+\mathbf{f}_{b}(2i)\mathbf{f}_{br}(2r+1-2i)\}\mathbf{b}(2n+1-2r)\ ] \quad \cdots\cdots\cdots(6)$$

And, using the even terms of the reconstruction filters, we get

$$\sum_{i=1}^{L/2}\{\mathbf{f}_{ar}(2i)\mathbf{A}(n+1-i)+\mathbf{f}_{br}(2i)\mathbf{B}(n+1-i)\}$$

$$=\sum_{r=1}^{L-1}[\sum_{i=1}^{r}\{\mathbf{f}_{a}(2i-1)\mathbf{f}_{ar}(2r+2-2i)+\mathbf{f}_{b}(2i-1)\mathbf{f}_{br}(2r+2-2i)\}\mathbf{b}(2n+2-2r)+$$

$$\sum_{i=1}^{r}\{\mathbf{f}_{a}(2i)\mathbf{f}_{ar}(2r+2-2i)+\mathbf{f}_{b}(2i)\mathbf{f}_{br}(2r+2-2i)\}\mathbf{b}(2n+1-2r)] \quad \cdots\cdots\cdots(7)$$

where any j-th filter coefficient = 0 for j outside the range $0 \le j \le L$ and $\mathbf{b}(k)=0$ for k outside the range $0 \le k \le m$. Now putting

$$\sum_{i=1}^{r}\{\mathbf{f}_a(2i-1)\mathbf{f}_{ar}(2r+1-2i)+\mathbf{f}_b(2i-1)\mathbf{f}_{br}(2r+1-2i)\}=0 \quad \cdots\cdots(8)$$

for all r in equation (6), we get only the odd components of $\mathbf{b}$ on the right hand side, i.e.,

$$\sum_{i=1}^{L/2}\{\mathbf{f}_{ar}(2i-1)\mathbf{A}(n+1-i)+\mathbf{f}_{br}(2i-1)\mathbf{B}(n+1-i)\}=\sum_{r=1}^{L-1}[\sum_{i=1}^{r}\{\mathbf{f}_a(2i)\mathbf{f}_{ar}(2r+1-2i)+$$
$$\mathbf{f}_b(2i)\mathbf{f}_{br}(2r+1-2i)\}\mathbf{b}(2n+1-2r)]$$

Thus, we can find odd components of $\mathbf{b}$ as

$$\mathbf{b}(2n+1-L)=\frac{\sum_{i=1}^{L/2}\{\mathbf{f}_{ar}(2i-1)\mathbf{A}(n+1-i)+\mathbf{f}_{br}(2i-1)\mathbf{B}(n+1-i)\}}{\sum_{i=1}^{L/2}\{\mathbf{f}_a(2i)\mathbf{f}_{ar}(L+1-2i)+\mathbf{f}_b(2i)\mathbf{f}_{br}(L+1-2i)\}} \quad \cdots\cdots(9)$$

provided

$$\sum_{i=1}^{r}\{\mathbf{f}_a(2i)\mathbf{f}_{ar}(2r+1-2i)+\mathbf{f}_b(2i)\mathbf{f}_{br}(2r+1-2i)\}=0, \quad \forall\ r\neq L/2\cdots\cdots\cdots(10)$$

Equation (8) is a condition for making the coefficients of all the even components of the signal in equation (6) zero while computing one odd component of the signal. Equation (10) is a condition to make the coefficients of all the odd components except the one being calculated, zero. Similarly, putting

$$\sum_{i=1}^{r}\{\mathbf{f}_a(2i)\mathbf{f}_{ar}(2r+2-2i)+\mathbf{f}_b(2i)\mathbf{f}_{br}(2r+2-2i)\}=0\cdots\cdots(11)$$

for all r in equation (7), we can find the even components of $\mathbf{b}$ as

$$\mathbf{b}(2n+2-L)=\frac{\sum_{i=1}^{L/2}\{\mathbf{f}_{ar}(2i)\mathbf{A}(n+1-i)+\mathbf{f}_{br}(2i)\mathbf{B}(n+1-i)\}}{\sum_{i=1}^{L/2}\{\mathbf{f}_a(2i-1)\mathbf{f}_{ar}(L+2-2i)+\mathbf{f}_b(2i-1)\mathbf{f}_{br}(L+2-2i)\}} \quad \cdots\cdots(12)$$

provided

$$\sum_{i=1}^{r}\{\mathbf{f}_a(2i-1)\mathbf{f}_{ar}(2r+2-2i)+\mathbf{f}_b(2i-1)\mathbf{f}_{br}(2r+2-2i)\}=0,\ \forall\ r\neq L/2\cdots\cdots(13)$$

Equation (11) is a condition which is required for making the coefficients of all the odd components of the signal in equation (7) zero while evaluating an even component of the

signal. Equation (13) is a condition to make the coefficients of all the even components except the one being calculated, zero.

Member Conditions:

In the above reconstruction scheme, we have seen that in order to reconstruct the signal perfectly from its decomposed components, the decomposition and reconstruction filters should satisfy the restrictions given by equations (8), (10), (11) and (13). Equations (8) and (11) together will be known as member condition-I and equations (10) and (13) as member condition-II, respectively given in equations (14) and (15) below.

$$\left.\begin{aligned}\sum_{i=1}^{r}\{\mathbf{f}_a(2i-1)\mathbf{f}_{ar}(2r+1-2i)+\mathbf{f}_b(2i-1)\mathbf{f}_{br}(2r+1-2i)\}=0\\ \sum_{i=1}^{r}\{\mathbf{f}_a(2i)\mathbf{f}_{ar}(2r+2-2i)+\mathbf{f}_b(2i)\mathbf{f}_{br}(2r+2-2i)\}=0\end{aligned}\right\}\quad \forall\ r \cdots\cdots(14)$$

$$\left.\begin{aligned}\sum_{i=1}^{r}\{\mathbf{f}_a(2i)\mathbf{f}_{ar}(2r+1-2i)+\mathbf{f}_b(2i)\mathbf{f}_{br}(2r+1-2i)\}=0\\ \sum_{i=1}^{r}\{\mathbf{f}_a(2i-1)\mathbf{f}_{ar}(2r+2-2i)+\mathbf{f}_b(2i-1)\mathbf{f}_{br}(2r+2-2i)\}=0\end{aligned}\right\}\quad \forall\ r \neq L/2 \cdots\cdots\cdots(15)$$

Equations (14) and (15) give the relations between decomposition filters and their corresponding reconstruction filters. Any set of decomposition and reconstruction filters that satisfy the member condition is known as a Perfect Reconstruction Filter (PRF) set. By "the member condition", we mean both member conditions I and II. We can find the coefficients of a member of a PRF set from equations (14) and (15). The members of a PRF set need not be orthogonal or orthonormal unlike in the case of wavelets or filter banks.

On the issue of perfect reconstruction in the case of continuous wavelet transform, Daubechies suggested admissibility condition. For discrete wavelet transform, Daubechies orthogonal wavelet filters are of QMF types. But the member condition of ISITRA given in equations (14) and (15) does not suggest anything about the requirement of the filters to be of QMF type or about admissibility condition. If one can find decomposition and reconstruction filters that satisfy the member condition, then they can be used for perfect reconstruction without having to concern about whether they are of QMF type or not. Also, we can find infinite number of filter coefficients of filters of QMF type of any particular

length. This can be easily proved from the restriction given by equations (10) and (13) which have infinitely many solutions. This explains briefly why we can have an infinite number of filter coefficients for any particular length L. Once we find a PRF set we can decompose a signal using the decomposition equations (4) and (5), and then reconstruct the original signal using the reconstruction equations (9) and (12).

Filters of QMF types always satisfy the member condition and hence can be used for perfect reconstruction in a subband decomposition scheme.

Proof: Let $\{\mathbf{f}_a, \mathbf{f}_b, \mathbf{f}_{ar}, \mathbf{f}_{br}\}$ be a set of filters of QMF type. Then, from equation (2), we get

$$\mathbf{f}_b(i) = (-1)^i \mathbf{f}_a(L+1-i)$$
$$\mathbf{f}_{ar}(i) = \mathbf{f}_a(L+1-i)$$
$$\mathbf{f}_{br}(i) = (-1)^{L+1-1}\mathbf{f}_a(i) = (-1)^{i-1}\mathbf{f}_a(i)$$

The QMF filters will form a PRF set if they satisfy the member condition of ISITRA. If the filters satisfy member condition-I, and also satisfy the inequality of the member condition-II, they are said to form a PRF set.

First part of member condition-I

$$\sum_{i=1}^{r}\{\mathbf{f}_a(2i-1)\mathbf{f}_{ar}(2r+1-2i) + \mathbf{f}_b(2i-1)\mathbf{f}_{br}(2r+1-2i)\}$$
$$= \sum_{i=1}^{r}\{\mathbf{f}_a(2i-1)\mathbf{f}_a(L-2r+2i) + (-1)^{2i-1}\mathbf{f}_a(L+2-2i)(-1)^{2r-2i}\mathbf{f}_a(2r+1-2i)\}$$
$$= \sum_{i=1}^{r}\{\mathbf{f}_a(2i-1)\mathbf{f}_a(L-2r+2i) - \mathbf{f}_a(L+2-2i)\mathbf{f}_a(2r+1-2i)\}$$
$$= 0, \quad \forall\ r$$

Second part of the member condition-I

$$\sum_{i=1}^{r}\{\mathbf{f}_a(2i)\mathbf{f}_{ar}(2r+2-2i)+\mathbf{f}_b(2i)\mathbf{f}_{br}(2r+2-2i)\}$$

$$=\sum_{i=1}^{r}\{\mathbf{f}_a(2i)\mathbf{f}_a(L-1-2r+2i)+(-1)^{2i}\mathbf{f}_a(L+1-2i)(-1)^{2r+1-2i}\mathbf{f}_a(2r+2-2i)\}$$

$$=\sum_{i=1}^{r}\{\mathbf{f}_a(2i)\mathbf{f}_a(L-1-2r+2i)-\mathbf{f}_a(L+1-2i)\mathbf{f}_a(2r+2-2i)\}$$

$$=0, \quad \forall\ r$$

First part of the member condition-II

$$\sum_{i=1}^{r}\mathbf{f}_a(2i)\mathbf{f}_{ar}(2r+1-2i)+\mathbf{f}_b(2i)\mathbf{f}_{br}(2r+1-2i)$$

$$=\sum_{i=1}^{r}\mathbf{f}_a(2i)\mathbf{f}_a(L-2r+2i)+(-1)^{2i}\mathbf{f}_a(L+1-2i).(-1)^{2r-2i}\mathbf{f}_a(2r+1-2i)$$

$$=\sum_{i=1}^{r}\mathbf{f}_a(2i)\mathbf{f}_a(L-2r+2i)+\mathbf{f}_a(L+1-2i)\mathbf{f}_a(2r+1-2i)\cdots\cdots(16)$$

(when $r=L/2$, the above expression becomes)

$$=\sum_{i=1}^{L/2}\mathbf{f}_a(2i)\mathbf{f}_a(2i)+\mathbf{f}_a(L+1-2i)\mathbf{f}_a(L+1-2i)$$, always a positive number.

Second part of member condition-II

$$\sum_{i=1}^{r}\{\mathbf{f}_a(2i-1)\mathbf{f}_{ar}(2r+2-2i)+\mathbf{f}_b(2i-1)\mathbf{f}_{br}(2r+2-2i)\}$$

$$=\sum_{i=1}^{r}\{\mathbf{f}_a(2i-1)\mathbf{f}_a(L-1-2r+2i)+(-1)^{2i-1}\mathbf{f}_a(L+2-2i)(-1)^{2r+1-2i}\mathbf{f}_a(2r+2-2i)\}$$

$$=\sum_{i=1}^{r}\{\mathbf{f}_a(2i-1)\mathbf{f}_a(L-1-2r+2i)+\mathbf{f}_a(L+2-2i)\mathbf{f}_a(2r+2-2i)\}\cdots\cdots(17)$$

(when $r=L/2$, the above expression becomes)

$$=\sum_{i=1}^{L/2}\{\mathbf{f}_a(2i-1)\mathbf{f}_a(2i-1)+\mathbf{f}_a(L+2-2i)\mathbf{f}_a(L+2-2i)\}$$ which is again a positive number.

In other words, filters of QMF type satisfy the member condition of ISITRA. The other values of $r\neq L/2$ in member condition-II allow one to choose perfect reconstruction filter coefficients. Expressions (16) and (17) yield the same set of expression and they should be made equal to zero to obtain filter coefficients required for perfect reconstruction.

$$\sum_{i=1}^{r}\{\mathbf{f}_a(2i)\mathbf{f}_a(L-2r+2i)+\mathbf{f}_a(L+1-2i)\mathbf{f}_a(2r+1-2i)\}=0\ \forall\ r\neq L/2\cdots\cdots(18)$$

When $r=1,$ equation (18) gives $\mathbf{f}_a(1)\mathbf{f}_a(L-1)+\mathbf{f}_a(2)\mathbf{f}_a(L)$ =0. That is, the first two and the last two elements of a QMF should not have the same sign, which explains why in all coefficients in a QMF filter cannot have all positive coefficients.

When $r=2,$ equation (18) gives

$\mathbf{f}_a(1)\mathbf{f}_a(L-3)+\mathbf{f}_a(2)\mathbf{f}_a(L-2)+\mathbf{f}_a(3)\mathbf{f}_a(L-1)+\mathbf{f}_a(4)\mathbf{f}_a(L)=0$.

In general, the set of linear equations required for finding a filter of QMF type of length L can be obtained from equation (18) as

$$\sum_{i=1}^{L/2}\mathbf{f}_a(i)\mathbf{f}_a(L-2j+i)=0\cdots\cdots\cdots(19)$$

From (19), one can easily find a filter of QMF type, other remaining filters can be easily found from QMF relations between the filters given in equation (2). Some methods of designing of quadrature mirror filters in time domain are described in [61]. However, they are relatively complex as compared with the methods of finding PRF sets of QMF types described in [117].

The reconstruction and the decomposition filters being of QMF type is not a necessary condition for perfect reconstruction.

Proof: This is evident from the reconstruction scheme-1.

Reconstruction Scheme-2:

Alternatively, in equation (6), we can make the coefficients of all the odd components and the coefficients of all the even components except $\mathbf{b}(2n+2-L)$, zero. Similarly, in equation (7), we can make the coefficients of all the even components and the coefficients of all the odd components except $\mathbf{b}(2n+1-L)$, zero. Equations (20) and (21) below give new member conditions I and II respectively.

$$\left.\begin{aligned}&\sum_{i=1}^{r}\{\mathbf{f}_a(2i)\mathbf{f}_{ar}(2r+1-2i)+\mathbf{f}_b(2i)\mathbf{f}_{br}(2r+1-2i)\}=0\\&\sum_{i=1}^{r}\{\mathbf{f}_a(2i-1)\mathbf{f}_{ar}(2r+2-2i)+\mathbf{f}_b(2i-1)\mathbf{f}_{br}(2r+2-2i)\}=0\end{aligned}\right\}\quad\forall\ r\ \cdots\cdots\cdots(20)$$

$$\left.\begin{aligned}\sum_{i=1}^{r}\{\mathbf{f}_a(2i-1)\mathbf{f}_{ar}(2r+1-2i)+\mathbf{f}_b(2i-1)\mathbf{f}_{br}(2r+1-2i)\}=0\\ \sum_{i=1}^{r}\{\mathbf{f}_a(2i)\mathbf{f}_{ar}(2r+2-2i)+\mathbf{f}_b(2i)\mathbf{f}_{br}(2r+2-2i)\}=0\end{aligned}\right\} \quad \forall\ r \neq L/2 \quad \cdots\cdots\cdots(21)$$

And the corresponding reconstruction equations are

$$\mathbf{b}(2n+1-L)=\frac{\sum_{i=1}^{L/2}\{\mathbf{f}_{ar}(2i)\mathbf{A}(n+1-i)+\mathbf{f}_{br}(2i)\mathbf{B}(n+1-i)\}}{\sum_{i=1}^{L/2}\{\mathbf{f}_a(2i)\mathbf{f}_{ar}(L+2-2i)+\mathbf{f}_b(2i)\mathbf{f}_{br}(L+2-2i)\}} \quad \cdots\cdots(22)$$

$$\mathbf{b}(2n+2-L)=\frac{\sum_{i=1}^{L/2}\{\mathbf{f}_{ar}(2i-1)\mathbf{A}(n+1-i)+\mathbf{f}_{br}(2i-1)\mathbf{B}(n+1-i)\}}{\sum_{i=1}^{L/2}\{\mathbf{f}_a(2i-1)\mathbf{f}_{ar}(L+1-2i)+\mathbf{f}_b(2i-1)\mathbf{f}_{br}(L+1-2i)\}} \quad \cdots\cdots(23)$$

Thus, we can also find a PRF set corresponding to the member condition equations (20) and (21) which can be used to decompose a signal using equations (4) and (5) and reconstruct the signal from its decomposed components using equations (22) and (23). Care must be taken here, as a PRF set under one member condition cannot be used in the reconstruction equations under a different member condition. For example, the reconstruction filters of a PRF set obtained from equations (14) and (15) cannot be used in the reconstruction equations (22) and (23) to reconstruct the original signal. Some more reconstruction schemes of ISITRA are given in Appendix A.

Property 1: If $\{\mathbf{f}_a,\mathbf{f}_b,\mathbf{f}_{ar},\mathbf{f}_{br}\}$ is a PRF set under a reconstruction scheme, then the sets $\{-\mathbf{f}_a,\mathbf{f}_b,-\mathbf{f}_{ar},\mathbf{f}_{br}\},\{\mathbf{f}_a,-\mathbf{f}_b,\mathbf{f}_{ar},-\mathbf{f}_{br}\},\{-\mathbf{f}_a,-\mathbf{f}_b,\mathbf{f}_{ar},\mathbf{f}_{br}\},\{\mathbf{f}_a,\mathbf{f}_b,-\mathbf{f}_{ar},-\mathbf{f}_{br}\}$ and $\{-\mathbf{f}_a,-\mathbf{f}_b,-\mathbf{f}_{ar},-\mathbf{f}_{br}\}$ also are PRF sets under the same reconstruction scheme.

This can be easily proved from the member condition. This enables us to find more PRF sets once we find one PRF set.

Property 2: The role of the decomposition and reconstruction filters can be interchanged. That is, we can use the reconstruction filters as the decomposition filters and vice versa. This is evident from the member condition equations. This property also enables us to find more PRF sets.

3.3.3 Computation of PRF sets of ISITRA

PRF sets form the core of a subband decomposition and reconstruction scheme. The theory of wavelet transform is also to find PRF sets. Here we will discuss how the PRF sets can be calculated from the member condition of ISITRA. We will mainly use equations (14) and (15) for the purpose because these member conditions is a generalized case perfect reconstruction scheme used in discrete wavelet transform based on filter bank model. A PRF set of length L will be denoted by PRF-L. Since the length of a PRF set is always positive and even, the PRF set of length 2, i.e., PRF-2 is the shortest. The computation of a PRF set involves finding the members of it which satisfy member condition-I and member condition-II. In practice, we first find the framework of a PRF-L set satisfying the member condition. By the framework of a PRF set, we mean the PRF set whose members have variable coefficients. And we get an actual PRF set when these variables are replaced by numerical values. In order to find a framework of a PRF set satisfying the member condition, we first find members that satisfy member condition-I and then apply member condition-II to further specify the members. If member condition-II is vacuously satisfied for the framework of a PRF set, the coefficients of the members of the PRF set can be chosen arbitrarily. Otherwise, the coefficients of the members of the PRF sets are obtained by solving the linear equations of member condition-II.

Next we will describe two schemes of ISITRA, namely, ISITRA-1 and ISITRA-2. In ISITRA-1, only the coefficients of a single non-zero array (whose elements are non-negative) of length L are provided and the coefficients of $\{\mathbf{f}_a, \mathbf{f}_b, \mathbf{f}_{ar}, \mathbf{f}_{br}\}$ are obtained from them. In ISITRA-2, the coefficients of two non-zero arrays (whose elements are non-negative) of length L are provided and the coefficients of $\{\mathbf{f}_a, \mathbf{f}_b, \mathbf{f}_{ar}, \mathbf{f}_{br}\}$ are obtained from them. Various frameworks of PRF sets of ISITRA-1 and ISITRA-2 are presented in Sections 3.4 and 3.5 respectively.

3.4 ISITRA-1

ISITRA-1 is nothing but ISITRA whose PRF sets are obtained from a single non-zero array of length L where its elements are non-negative real numbers. The decomposition and reconstruction schemes of ISITRA-1 are based on this type of PRF sets. Here the members of a PRF set are interrelated. The reconstruction filters can be obtained from the

decomposition filters and vice versa. The orthogonal wavelet filter coefficients form PRF sets of ISITRA-1. We will now discuss the frameworks of PRF sets of length 2 and 4 of ISITRA-1.

PRF-2 Sets:

When the length of the PRF set is 2, member condition-I of equation (14) simply becomes

$$\left.\begin{aligned}\mathbf{f}_a(1)\mathbf{f}_{ar}(1)+\mathbf{f}_b(1)\mathbf{f}_{br}(1)=0\\ \mathbf{f}_a(2)\mathbf{f}_{ar}(2)+\mathbf{f}_b(2)\mathbf{f}_{br}(2)=0\end{aligned}\right\}\cdots\cdots\cdots(24)$$

and the member condition-II of equation (15) is vacuously satisfied. The corresponding reconstruction equations are

$$\left.\begin{aligned}\mathbf{b}(2n-1)&=\frac{\mathbf{f}_{ar}(1)\mathbf{A}(n)+\mathbf{f}_{br}(1)\mathbf{B}(n)}{\mathbf{f}_a(2)\mathbf{f}_{ar}(1)+\mathbf{f}_b(2)\mathbf{f}_{br}(1)}\\ \mathbf{b}(2n)&=\frac{\mathbf{f}_{ar}(2)\mathbf{A}(n)+\mathbf{f}_{br}(2)\mathbf{B}(n)}{\mathbf{f}_a(1)\mathbf{f}_{ar}(2)+\mathbf{f}_b(1)\mathbf{f}_{br}(2)}\end{aligned}\right\}\cdots\cdots\cdots(25)$$

Choice of $\mathbf{f}_a, \mathbf{f}_b, \mathbf{f}_{ar}$ and $\mathbf{f}_{br}$ satisfying equation (24) can be made easily. In the case of ISITRA-1, let $\mathbf{l}=[l_1, l_2]$ be any non-zero array (i.e., l_1, l_2 are any two non-negative real numbers, both not zero) whose elements are non-negative. Then we can find $\mathbf{f}_a, \mathbf{f}_b, \mathbf{f}_{ar}$ and $\mathbf{f}_{br}$ satisfying equation (24) from it. For example, $\mathbf{f}_a=[l_1,\ l_2]$, $\mathbf{f}_{ar}=[l_2,\ l_1]$, $\mathbf{f}_b=[-l_2,\ l_1]$ and $\mathbf{f}_{br}=[l_1,\ -l_2]$ satisfy equation (24) and hence form a framework of a PRF-2 set. Since the member condition-II vanishes, there is no restriction on the values of l_1, l_2 except that both should not be zero to avoid division by zero. Thus, we can find infinitely many actual PRF-2 sets, just by choosing any two real numbers, both not zero. The above PRF-2 set of ISITRA-1 has variable coefficients as its members' coefficients. This type of PRF sets whose members have variable coefficients will be known as the framework of the PRF set. In practice, the coefficients should be real constants. By an actual PRF set we mean the PRF set whose members have real number constants as coefficients. If we choose, $l_1=l_2=1/\sqrt{2}$, we get Daubechies' wavelet filters of length 2, in short DB1 or we can choose some other values as $l_1=0.25$ and $l_2=0.625$ to get a different PRF-2 set. The above relation between $\mathbf{f}_a$ and $\mathbf{f}_b$ in the above framework of PRF-2 is known as QMF type.

We may find some other different frameworks that satisfy equation (24). For example, $\mathbf{f}_a = [l_1, -l_2]$, $\mathbf{f}_{ar} = [-l_2, l_1]$, $\mathbf{f}_b = [l_2, -l_1]$ and $\mathbf{f}_{br} = [l_1, -l_2]$ are the members of a non-orthogonal and non-QMF PRF set which satisfy equation (24). If we take $\mathbf{f}_a = [.125, -.124]$ and $\mathbf{f}_b = [.124, -.125]$ and decompose the image in Fig.3.4a with these filters, we get the decomposed image components shown in Fig.3.4b. All these components are details. Similarly, if we choose $\mathbf{f}_a = [.125, .124]$ and $\mathbf{f}_b = [.124, .125]$ as decomposition filters, all the decomposed components are approximations as shown in Fig.3.4c. In both the cases, we can get perfect reconstruction using their corresponding reconstruction filters. However, decomposition of an image into all details or into all approximations is not possible in wavelet or filter bank scheme. Decomposing a signal into details hides the distinguishable features of the signal. This may be employed for encryption as discussed in [119]. Various other possibilities of length-2 PRF sets are given in [118]. Note here that we are free to choose filter coefficients easily without any restriction. But we cannot choose other filter coefficients than $f_a = [1/\sqrt{2}, 1/\sqrt{2}]$ in the case of DWT because of the restriction in equation (1). So we cannot make the range of pixel values of the decomposed components of a signal decreased or increased as we desired in the case of wavelets. But in our case it is easily possible [125].

Figure.3.4a Original image

Figure.3.4b Decomposition of Fig.3.4a into details

Figure.3.4c Decomposition of Fig.3.3a into approximations

Once we find a PRF set, which satisfies the member condition, we can use it for decomposition and perfect reconstruction without having to concern about the orthogonality or orthonormality or about the types of the filters. Also, we can find more frameworks of PRF-2 sets once we get one framework of PRF-2 using Property-1 or Property-2. Note here that for every framework of a PRF-L set, there exist infinitely many actual PRF-L sets. Here lies the difference between ISITRA and other existing transforms.

We may have a different reconstruction scheme for filters of length 2, which is free of the any member condition from equations (4) and (5), by solving for $\mathbf{b}(2n)$ and $\mathbf{b}(2n-1)$.

That is, when $L=2$, from equations (4) and (5), we have the following decomposed components.

$$\left.\begin{aligned}\mathbf{A}(n) &= \mathbf{f}_a(1)\mathbf{b}(2n) + \mathbf{f}_a(2)\mathbf{b}(2n-1)\\ \mathbf{B}(n) &= \mathbf{f}_b(1)\mathbf{b}(2n) + \mathbf{f}_b(2)\mathbf{b}(2n-1)\end{aligned}\right\} \cdots\cdots\cdots (26)$$

From equation (26), we can easily solve for $\mathbf{b}(2n)$ and $\mathbf{b}(2n-1)$. For PRF sets of length greater than 2, we cannot have equations similar to equation (26) from which the values of $\mathbf{b}(2n)$ and $\mathbf{b}(2n-1)$ can be solved directly. Instead they have to be solved differently from the member condition-I and member condition-II equations as described below.

PRF-4 sets:

For the evaluation of PRF-4 sets of ISITRA-1, first we choose the filter coefficients in such a way as to satisfy member condition-I. Once we find the filters that satisfy member condition-I, the actual filter coefficients are then evaluated from member-condition-II.

Member condition-I of PRF-4 from equation (14) can be obtained as

$$\mathbf{f}_a(1)\mathbf{f}_{ar}(1) + \mathbf{f}_b(1)\mathbf{f}_{br}(1) = 0$$
$$\mathbf{f}_a(1)\mathbf{f}_{ar}(3) + \mathbf{f}_b(1)\mathbf{f}_{br}(3) + \mathbf{f}_a(3)\mathbf{f}_{ar}(1) + \mathbf{f}_b(3)\mathbf{f}_{br}(1) = 0$$
$$\mathbf{f}_a(3)\mathbf{f}_{ar}(3) + \mathbf{f}_b(3)\mathbf{f}_{ar}(3) = 0$$

$$\mathbf{f}_a(2)\mathbf{f}_{ar}(2) + \mathbf{f}_b(2)\mathbf{f}_{br}(2) = 0$$
$$\mathbf{f}_a(2)\mathbf{f}_{ar}(4) + \mathbf{f}_b(2)\mathbf{f}_{br}(4) + \mathbf{f}_a(4)\mathbf{f}_{ar}(2) + \mathbf{f}_b(4)\mathbf{f}_{br}(2) = 0$$
$$\mathbf{f}_a(4)\mathbf{f}_{ar}(4) + \mathbf{f}_b(4)\mathbf{f}_{ar}(4) = 0$$

We have to choose $\mathbf{f}_a, \mathbf{f}_b, \mathbf{f}_{ar}$ and $\mathbf{f}_{br}$ in such a way that they satisfy the above equations.

Let $\begin{aligned}\mathbf{f}_a &= [-l_1 \quad l_2 \quad l_3 \quad l_4]\\ \mathbf{f}_{ar} &= [l_4 \quad l_3 \quad l_2 \quad -l_1]\\ \mathbf{f}_b &= [l_4 \quad -l_3 \quad l_2 \quad l_1]\\ \mathbf{f}_{br} &= [l_1 \quad l_2 \quad -l_3 \quad l_4]\end{aligned}$ be a trial framework of PRF-4 sets of ISITRA-1.

The above set of filters satisfies member condition–I. It will be a PRF set if member condition–II also is satisfied. Member condition-II for $L=4$ obtained from equations (10) and (13) is given by

$\mathbf{f}_a(2)\mathbf{f}_{ar}(1)+\mathbf{f}_b(2)\mathbf{f}_{br}(1)=0$
$\mathbf{f}_a(4)\mathbf{f}_{ar}(3)+\mathbf{f}_b(4)\mathbf{f}_{ar}(3)=0$
$\mathbf{f}_a(1)\mathbf{f}_{ar}(2)+\mathbf{f}_b(1)\mathbf{f}_{br}(2)=0$
$\mathbf{f}_a(3)\mathbf{f}_{ar}(4)+\mathbf{f}_b(3)\mathbf{f}_{ar}(4)=0$

All these equations of member condition-II give $-l_1l_3+l_2l_4=0\cdots\cdots\cdots(27)$.

All the terms in equation (27) are not of the same sign. Hence, member condition-II is said to be satisfied with the above filters. Thus the above trial PRF-4 set is said to form a framework of a PRF-4 set of ISITRA-1. The actual PRF-4 set for this framework is found by finding the values of l_1, l_2, l_3 and l_4 from equation (27). We can choose any three coefficients arbitrarily and find the fourth from the relation. For, example we can choose $l_1=1, l_2=2, l_3=6$ then from equation (27), we get $l_4=3$. And the actual PRF set of the above framework is $\mathbf{f}_a=[-1, 2, 6, 3]$, $\mathbf{f}_{ar}=[3, 6, 2, -1]$, $\mathbf{f}_b=[3, -6, 2, 1]$ and $\mathbf{f}_{br}=[1, 2, -6, 3]$. The decomposition and reconstruction filters of this PRF set do not satisfy equation (1) or (2) or (3) and are not acceptable filters in wavelet transform or in filter bank. But still we can get perfect reconstruction with these filters in ISITRA. In this way, we can find infinitely many actual PRF-4 sets of ISITRA-1 of the above framework of PRF-4 sets. Again, let us consider the following framework of PRF-4 sets
$\mathbf{f}_a=[l_1, l_2, l_3, l_4]$, $\mathbf{f}_b=[l_4, l_3, -l_2, -l_1]$, $\mathbf{f}_{ar}=[l_4, -l_3, -l_2, l_1]$ and $\mathbf{f}_{br}=[-l_1, l_2, -l_3, l_4]$.
This set is again not a possible PRF set in wavelet because $\mathbf{f}_a$ whose elements are all positive cannot be a wavelet filter. More other frameworks of PRF-4 sets are available in [125].

We can generate other frameworks of PRF-4 set just using Property-1 or Property-2. If one of the reconstruction filters is used as a decomposition filter, then the other reconstruction filter will also be used as the decomposition filters. We may also find different frameworks under the same member condition of a reconstruction scheme. Or we may find different frameworks under different member condition of a different reconstruction scheme.

Proceeding in this way, we can find frameworks of PRF-6 and PRF-8 sets which are given in Appendix B. In general, we can find frameworks of PRF-*L* sets from the member condition equations (14) and (15). The number of possible frameworks that a PRF-*L* set can

have, depends on the value of length L. The higher the length of a PRF set, the higher is the number of the frameworks one can find in a reconstruction scheme. The PRF sets are different for different member conditions of different reconstruction schemes. We could have obtained different PRF sets if we had used the member conditions of equations (21) and (22). It may be remembered that for every framework of a PRF-*L* set, there are infinitely many actual PRF-*L* sets. This shows the flexibility of ISITRA-1 in choosing filter coefficients and hence in forming PRF sets. The space of PRF sets of ISITRA-1 is infinitely large where the space of PRF set of compactly supported orthogonal wavelet filters occupies a limited space in it. The frameworks of PRF sets of commonly used compactly supported wavelets are given below.

3.4.1 ISITRA-1 frameworks of PRF sets of well-known orthogonal wavelets

In this section, we will produce the frameworks of the decomposition and reconstruction filters of commonly used orthogonal wavelets. The wavelet filters are obtained from a predefined function called wavelets using the FIR filter relation. But in ISITRA we do not need any predefined function for obtaining the PRF sets. We can directly find them from the member condition as described in the previous section. We will not show the derivation of frameworks but will give the frameworks directly. We know for every framework of PRF set, there are associated infinitely many actual PRF sets. Thus, from a single framework of any PRF-*L* set, we can have infinitely many actual PRF-*L* sets. Also, from one framework one can obtain different other frameworks by using Property-1 or Property-2. We will now find the framework of PRF sets of decomposition and reconstruction filters of well-known wavelets, mainly the Daubechies' orthogonal wavelets.

Framework of PRF set of DB1:

DB wavelets are wavelets named after Daubechies. DB1 wavelet filters are of length 2. They have the following framework.

$\mathbf{f}_a = [l_1, l_2]$ $\mathbf{f}_{ar} = [l_2, l_1]$ $\mathbf{f}_b = [-l_2, l_1]$ $\mathbf{f}_{br} = [l_2, -l_1]$

where l's are any real numbers not all zeros. In the filter coefficients of DB1, $l_1 = l_2 = 1/\sqrt{2}$ and it is the only PRF set of length 2 in the case of wavelet. But in the case of ISITRA-1 one can choose any two non-zero real numbers and use it in the above

framework as a PRF set. That is, we can find infinitely many actual PRF2 sets corresponding to the above framework. Thus, wavelet filter coefficients of length-2 form an actual PRF2 set out of infinite number of actual PRF2 sets. This flexibility allows us to choose our own PRF-2 sets instead of using filter coefficients of DB1, if one likes to test an application with their own filter coefficients.

Framework of PRF set of DB2:

DB2 wavelet filters are of length 4 and belong to a PRF-4 set. They have the following framework.

$\mathbf{f}_a = [-l_1, l_2, l_3, l_4]$ $\mathbf{f}_{ar} = [-l_4, l_3, l_2, l_1]$

$\mathbf{f}_b = [-l_4, l_3, -l_2, -l_1]$ $\mathbf{f}_{br} = [-l_1, -l_2, l_3, -l_4]$

Member condition II here yields $-l_1.l_3 + l_2.l_4 = 0$, same as equation (27).

So, we have to find any four real numbers such that they satisfy equation (27). DB2 wavelet filter coefficients that satisfy equation (28) are

$l_1 = -(1-\sqrt{3})/4\sqrt{2}$, $l_2 = (3-\sqrt{3})/4\sqrt{2}$, $l_3 = (3+\sqrt{3})/4\sqrt{2}$ and $l_4 = (1+\sqrt{3})/4\sqrt{2}$. If we convert these values into floating point numbers, $\mathbf{f}_a$ of DB2 becomes

$\mathbf{f}_a$ =[-0.12940952255092 0.22414386804186 0.83651630373747 0.48296291314469].

The coefficients of $\mathbf{f}_a$ do not exactly satisfy equation (27). It yields 2.870204074412186e-013, not exactly zero, hence the DB2 wavelet filter coefficients cannot be used for perfect reconstruction of a signal. It will always have reconstruction errors if we use the above filter coefficients. Theoretically, for perfect reconstruction, the filter coefficients have to exactly satisfy member condition-II.

We can easily find many PRF sets, which satisfy equation (27). For example, let us choose $l_1 = 0.125$, $l_2 = 0.225$, $l_4 = 0.483$. Then $l_3 = 0.8694$. If we use this in the above framework and use them in the decomposition and reconstruction equations, we would be able to obtain almost the same result and at the same time we can perfectly reconstruct the original signal from its decomposed components. Similarly, we can find infinitely many actual PRF-4 sets corresponding to the above framework of DB-2. The filters of Sym2 (Symlets or symmetric Daubechies' wavelets of length 4) [40] have the same framework. One can also find different frameworks from the above frameworks using Property-1 or

Property-2. Frameworks of PRF sets of some other popular wavelets are given in Appendix B. Every framework is associated with infinitely many actual PRF sets. It is not certain, which on of filters in a framework will give good performance for a particular application. Searching an optimal filter from the possible PRF sets of a framework can be done in a systematic way based on the optimization of certain criteria with respect to the application at hand using genetic algorithm. Such a search technique, for finding optimal PRF sets for a given framework of length 4 and 6, is described in the Appendix C. The same technique can be extended to a framework of any length of ISITRA-1.

3.5 ISITRA-2

ISITRA-2 is nothing but ISITRA whose PRF sets are obtained from two distinct and unrelated non-zero arrays. Like in ISITRA-1, the elements of these arrays are non-negative real numbers. By two distinct and unrelated arrays, we mean one cannot be obtained from the other by simple change in signs or positions of the elements or by multiplying with a constant. We will consider here how PRF sets are obtained from two distinct and unrelated arrays from equations (14) and (15) of the member condition of ISITRA.

PRF-2 sets:

Let us begin with the consideration of PRF-2 of ISITRA-2. When the length of the PRF set is 2, the member condition-I of equation (14) simply becomes

$$\left.\begin{aligned}\mathbf{f}_a(1)\mathbf{f}_{ar}(1)+\mathbf{f}_b(1)\mathbf{f}_{br}(1)=0\\ \mathbf{f}_a(2)\mathbf{f}_{ar}(2)+\mathbf{f}_b(2)\mathbf{f}_{br}(2)=0\end{aligned}\right\}\cdots\cdots\cdots(28)$$

and member condition-II of equation (15) vanishes (i.e., is vacuously satisfied).

Let $\mathbf{l}=[l_1,l_2]$ and $\mathbf{k}=[k_1,k_2]$ be any two unrelated and distinct non-zero arrays whose elements are non-negative. If we choose our $\mathbf{f}_a,\mathbf{f}_b,\mathbf{f}_{ar}$ and $\mathbf{f}_{br}$ in the following way, i.e., $\mathbf{f}_a=[l_1,l_2]$, $\mathbf{f}_{ar}=[-k_2,k_1]$, $\mathbf{f}_b=[k_2,-k_1]$ and $\mathbf{f}_{br}=[l_1,l_2]$ then the equation (28) is satisfied. Hence they form a framework of PRF-2 sets of ISITRA-2. The actual PRF set or simply the PRF set is obtained by substituting the variable coefficients by real coefficients.

As the member condition-II vanishes, there is no restrictions on the values of l's and k's which means that we can choose their values arbitrarily. In other words, we can find

infinitely many PRF-2 sets of the above framework. We may also find (i) different other frameworks of PRF-2 sets from the above framework by using Property-1 or Property-2 or (ii) different frameworks altogether from the member condition of equation (28).

PRF-4 sets:

For the evaluation of PRF-4 sets of ISITRA-2, first we choose the filter coefficients in such a way as to satisfy member condition-I of equation (14). Then the actual filter coefficients are evaluated from member condition-II. The member condition-I of equation (14) simply becomes the following set of equations.

$$\left.\begin{aligned}
&\mathbf{f}_a(1)\mathbf{f}_{ar}(1)+\mathbf{f}_b(1)\mathbf{f}_{br}(1)=0\\
&\mathbf{f}_a(1)\mathbf{f}_{ar}(3)+\mathbf{f}_b(1)\mathbf{f}_{br}(3)+\mathbf{f}_a(3)\mathbf{f}_{ar}(1)+\mathbf{f}_b(3)\mathbf{f}_{br}(1)=0\\
&\mathbf{f}_a(3)\mathbf{f}_{ar}(3)+\mathbf{f}_b(3)\mathbf{f}_{ar}(3)=0
\end{aligned}\right\} \cdots\cdots\cdots (29a)$$

$$\left.\begin{aligned}
&\mathbf{f}_a(2)\mathbf{f}_{ar}(2)+\mathbf{f}_b(2)\mathbf{f}_{br}(2)=0\\
&\mathbf{f}_a(2)\mathbf{f}_{ar}(4)+\mathbf{f}_b(2)\mathbf{f}_{br}(4)+\mathbf{f}_a(4)\mathbf{f}_{ar}(2)+\mathbf{f}_b(4)\mathbf{f}_{br}(2)=0\\
&\mathbf{f}_a(4)\mathbf{f}_{ar}(4)+\mathbf{f}_b(4)\mathbf{f}_{ar}(4)=0
\end{aligned}\right\} \cdots\cdots\cdots (29b)$$

We have to choose $\mathbf{f}_a, \mathbf{f}_b, \mathbf{f}_{ar}$ and $\mathbf{f}_{br}$ in such a way that they satisfy the above two sets of equations from two arrays $\mathbf{l}=[l_1, l_2, l_3, l_4]$ and $\mathbf{k}=[k_1, k_2, k_3, k_4]$.

Let $\begin{aligned}\mathbf{f}_a &= [-1_1 \quad l_2 \quad l_3 \quad l_4]\\ \mathbf{f}_{ar} &= [k_4 \quad k_3 \quad k_2 \quad -k_1]\\ \mathbf{f}_b &= [k_4 \quad -k_3 \quad k_2 \quad k_1]\\ \mathbf{f}_{br} &= [l_1 \quad l_2 \quad -l_3 \quad l_4]\end{aligned}$ be a trial framework which satisfies equation (29)

Member condition-II of equation (15), when $L=4$, yields the following set of equations.

$$\begin{aligned}
&\mathbf{f}_a(2)\mathbf{f}_{ar}(4)+\mathbf{f}_b(2)\mathbf{f}_{br}(4)=0\\
&\mathbf{f}_a(4)\mathbf{f}_{ar}(2)+\mathbf{f}_b(4)\mathbf{f}_{ar}(2)=0\\
&\mathbf{f}_a(1)\mathbf{f}_{ar}(3)+\mathbf{f}_b(1)\mathbf{f}_{br}(3)=0\\
&\mathbf{f}_a(3)\mathbf{f}_{ar}(1)+\mathbf{f}_b(3)\mathbf{f}_{ar}(1)=0
\end{aligned}$$

Then the corresponding equations of member condition–II for the above trial framework of PRF-4 set yields the following equations.

$$\left.\begin{aligned}
&-l_1k_3+l_2k_4=0\\
&-l_3k_1+l_4k_2=0
\end{aligned}\right\} \cdots\cdots\cdots (30)$$

All the terms in these two equations are not of the same sign. Therefore, member condition-II is said to be satisfied with the above filters. And hence the above trial PRF-4 set is said to

form a framework of a PRF-4 set of ISITRA-2. The actual PRF-4 set for this framework is found by evaluating l's and k's satisfying these two equations. This can be done in infinite number of ways. The bi-orthogonal wavelet filters of length 4 satisfy these two equations and hence they can be used for perfect reconstruction. We may find some other filters of length 4 and use them as a PRF-4 set for decomposition and reconstruction schemes. Also, the decomposition filter is crossed-related to its reconstruction filter in the sense that $\mathbf{f}_a$ and $\mathbf{f}_{br}$ consist of l's, and $\mathbf{f}_b$ and $\mathbf{f}_{ar}$ consist of k's. This allows us to find more PRF sets easily. That is, a bi-orthogonal filter set of length 4 can be used as four different PRF-4 sets.

PRF-6 sets:

When $L = 6$, we get the following two sets of equations from the equation (14) of member condition-I.

$$\left.\begin{aligned}
&\mathbf{f}_a(1)\mathbf{f}_{ar}(1) + \mathbf{f}_b(1)\mathbf{f}_{br}(1) = 0\\
&\mathbf{f}_a(1)\mathbf{f}_{ar}(3) + \mathbf{f}_b(1)\mathbf{f}_{br}(3) + \mathbf{f}_a(3)\mathbf{f}_{ar}(1) + \mathbf{f}_b(3)\mathbf{f}_{br}(1) = 0\\
&\mathbf{f}_a(1)\mathbf{f}_{ar}(5) + \mathbf{f}_b(1)\mathbf{f}_{br}(5) + \mathbf{f}_a(3)\mathbf{f}_{ar}(3) + \mathbf{f}_b(3)\mathbf{f}_{br}(3) + \mathbf{f}_a(5)\mathbf{f}_{ar}(1) + \mathbf{f}_b(5)\mathbf{f}_{br}(1) = 0\\
&\mathbf{f}_a(3)\mathbf{f}_{ar}(5) + \mathbf{f}_b(3)\mathbf{f}_{br}(5) + \mathbf{f}_a(5)\mathbf{f}_{ar}(3) + \mathbf{f}_b(5)\mathbf{f}_{br}(3) = 0\\
&\mathbf{f}_a(5)\mathbf{f}_{ar}(5) + \mathbf{f}_b(5)\mathbf{f}_{br}(5) = 0
\end{aligned}\right\}$$

and

$$\left.\begin{aligned}
&\mathbf{f}_a(2)\mathbf{f}_{ar}(2) + \mathbf{f}_b(2)\mathbf{f}_{br}(2) = 0\\
&\mathbf{f}_a(2)\mathbf{f}_{ar}(4) + \mathbf{f}_b(2)\mathbf{f}_{br}(4) + \mathbf{f}_a(4)\mathbf{f}_{ar}(2) + \mathbf{f}_b(4)\mathbf{f}_{br}(2) = 0\\
&\mathbf{f}_a(2)\mathbf{f}_{ar}(6) + \mathbf{f}_b(2)\mathbf{f}_{br}(6) + \mathbf{f}_a(4)\mathbf{f}_{ar}(4) + \mathbf{f}_b(4)\mathbf{f}_{br}(4) + \mathbf{f}_a(6)\mathbf{f}_{ar}(2) + \mathbf{f}_b(6)\mathbf{f}_{br}(2) = 0\\
&\mathbf{f}_a(2)\mathbf{f}_{ar}(6) + \mathbf{f}_b(2)\mathbf{f}_{br}(6) + \mathbf{f}_a(6)\mathbf{f}_{ar}(4) + \mathbf{f}_b(6)\mathbf{f}_{br}(4) = 0\\
&\mathbf{f}_a(6)\mathbf{f}_{ar}(6) + \mathbf{f}_b(6)\mathbf{f}_{br}(6) = 0
\end{aligned}\right\}$$

We have to choose $\mathbf{f}_a, \mathbf{f}_b, \mathbf{f}_{ar}$ and $\mathbf{f}_{br}$ which satisfy the above two sets of equations.

Let $\begin{matrix}\mathbf{f}_a = [l_1, -l_2, -l_3, l_4, l_5, l_6], \quad \mathbf{f}_{ar} = [k_6, k_5, k_4, -k_3, -k_2, k_1]\\ \mathbf{f}_b = [-k_6, k_5, -k_4, -k_3, k_2, k_1], \mathbf{f}_{br} = [l_1, l_2, -l_3, -l_4, l_5, -l_6]\end{matrix}$ be a trial framework of a PRF-6 set which satisfy the above two sets of equations.

The sets of equations for member condition-II of equation (15) are

$$\left.\begin{aligned}
&\mathbf{f}_a(2)\mathbf{f}_{ar}(1)+\mathbf{f}_b(2)\mathbf{f}_{br}(1)=0\\
&\mathbf{f}_a(2)\mathbf{f}_{ar}(3)+\mathbf{f}_b(2)\mathbf{f}_{br}(3)+\mathbf{f}_a(4)\mathbf{f}_{ar}(1)+\mathbf{f}_b(4)\mathbf{f}_{br}(1)=0\\
&\mathbf{f}_a(4)\mathbf{f}_{ar}(5)+\mathbf{f}_b(4)\mathbf{f}_{br}(5)+\mathbf{f}_a(6)\mathbf{f}_{ar}(3)+\mathbf{f}_b(6)\mathbf{f}_{br}(3)=0\\
&\mathbf{f}_a(6)\mathbf{f}_{ar}(5)+\mathbf{f}_b(6)\mathbf{f}_{br}(5)=0
\end{aligned}\right\} \quad \text{and}$$

$$\left.\begin{aligned}
&\mathbf{f}_a(1)\mathbf{f}_{ar}(2)+\mathbf{f}_b(1)\mathbf{f}_{br}(2)=0\\
&\mathbf{f}_a(1)\mathbf{f}_{ar}(4)+\mathbf{f}_b(1)\mathbf{f}_{br}(4)+\mathbf{f}_a(3)\mathbf{f}_{ar}(2)+\mathbf{f}_b(3)\mathbf{f}_{br}(2)=0\\
&\mathbf{f}_a(3)\mathbf{f}_{ar}(6)+\mathbf{f}_b(3)\mathbf{f}_{br}(6)+\mathbf{f}_a(5)\mathbf{f}_{ar}(4)+\mathbf{f}_b(5)\mathbf{f}_{br}(4)=0\\
&\mathbf{f}_a(5)\mathbf{f}_{ar}(6)+\mathbf{f}_b(5)\mathbf{f}_{br}(6)=0
\end{aligned}\right\}$$

The above sets of equations of member condition-II of ISITRA yield the following equations as member condition-II of ISITRA-2 for the above framework of PRF-6 set. The corresponding member condition-II restriction yields

$$\left.\begin{aligned}
&l_1k_5-l_2k_6=0\\
&-l_1k_3-l_2k_4-l_3k_5+l_4k_6=0
\end{aligned}\right\}\cdots\cdots\cdots(31)$$

Since all the terms in the two sets of equations in equation (31) are not of the same sign, the above trial framework forms a framework of PRF6 set. The actual PRF-6 sets can be found by computing the values of l_1, l_2, l_3, l_4, l_5 and l_6 from equation (31). This can be done in infinitely many ways.

PRF-8 sets:

Proceeding in this way, we can have the following as a framework of a PRF8 set.

$\mathbf{f}_a=[-l_1, l_2, l_3, -l_4, -l_5, l_6, l_7, l_8]$, $\mathbf{f}_{ar}=[k_8, k_7, k_6, -k_5, -k_4, k_3, k_2, -k_1]$

$\mathbf{f}_b=[-k_8, k_7, -k_6, -k_5, k_4, k_3, -k_2, -k_1]$, $\mathbf{f}_{br}=[-l_1, -l_2, l_3, l_4, -l_5, -l_6, l_7, -l_8]$

The corresponding member condition-II from equation (15) yields

$$\left.\begin{aligned}
&-l_1k_7+l_2k_8=0\\
&l_1k_5+l_2k_6+l_3k_7-l_4k_8=0\\
&-l_1k_3-l_2k_4-l_3k_5-l_4k_6-l_5k_7+l_6k_8=0
\end{aligned}\right\}\cdots\cdots\cdots(32)$$

From the above equations we can find the values of l's and k's and then replace in the framework to get an actual PRF8 set. We can find many frameworks of PRF8 sets from the member condition of equations (14) and (15). Proceeding in this way, we can find various frameworks of PRF-L sets of ISITRA-2 for higher values of filter length L.

Also note here that when all l's and k's are equal, the above frameworks of ISITRA-2 still satisfy member condition equations (14) and (15). That is, they can still be used for perfect reconstruction. When all $|l_i| = |k_{L+1-i}|, \forall\ i = 1,2,\cdots,L$, all the decomposition and reconstruction filters are interrelated and hence one can be obtained from the other. In other words, we can say all the members of a PRF set can be obtained from one array. This type of PRF set whose members can be obtained from or specified using an array is known as ISITRA-1. For example, the decomposition and the reconstruction filters of compactly supported wavelets are the PRF sets of ISITRA-1.

3.5.1 ISITRA-2 frameworks of PRF sets of well-known bi-orthogonal wavelets

In this Subsection we will provide the frameworks of the decomposition and reconstruction filters of commonly used bi-orthogonal wavelets, mainly that of Daubechies. The frameworks of bi-orthogonal wavelet filters belong to the frameworks of PRF sets of ISITRA-2. That is, here all the members of a framework are obtained from two arrays. We briefly produce here the frameworks of the commonly used bi-orthogonal wavelet filters so that one can choose one's own PRF sets. The decomposition filters and their corresponding reconstruction filters are crossed related e.g., as in equation (3).

Framework of Bior1.1:

$\mathbf{f}_a = [l_1, l_2]\quad \mathbf{f}_{ar} = [l_2, l_1]\quad \mathbf{f}_b = [-l_2, l_1]\quad \mathbf{f}_{br} = [l_2, -l_1]$

In the strict sense, Bior1.1 is not bi-orthogonal. This is the only case where a PRF set of a bi-orthogonal wavelet filter coefficients does not belong to the PRF set of ISITRA-2.

Framework of PRF set of Bior1.3:

$\mathbf{f}_a = [-l_1, l_1, l_2, l_2, l_1, -l_1],\quad \mathbf{f}_{ar} = [0, 0, l_2, l_2, 0, 0]$
$\mathbf{f}_b = [0, 0, -l_2, l_2, 0, 0],\quad \mathbf{f}_{br} = [-l_1, -l_1, l_2, -l_2, l_1, l_1]$

Member condition-II vanishes and hence any two real numbers, l_1 and $l_2 \neq 0$, can be chosen as coefficients of the above PRF set. Thus, we see we can find infinitely many actual PRF-6 sets, instead of using the PRF-6 set of the Bior1.3 developed by Daubechies. Choosing the filter coefficients of a member of PRF set here is nothing but a mere fun every layman can enjoy.

Framework of PRF set of Bior1.5:

The filters of Bior1.5 are of length 10. The framework of the PRF set of Bior1.5 is given below.

$\mathbf{f}_a = [l_1, -l_1, -l_2, l_2, l_3, l_3, l_2, -l_2, -l_1, l_1]$, $\mathbf{f}_{ar} = [0, 0, 0, 0, l_3, l_3, 0, 0, 0, 0]$

$\mathbf{f}_b = [0, 0, 0, 0, -l_3, l_3, 0, 0, 0, 0]$, $\mathbf{f}_{br} = [l_1, l_1, -l_2, -l_2, l_3, -l_3, l_2, l_2, -l_1, -l_1]$

As in Bior1.3, the member condition II of the PRF set vanishes, and hence we can choose any three real numbers, l_1, l_2 and $l_3 \neq 0$.

Framework of PRF set of Bior2.2:

The filters of Bior2.2 are of length 6. They belong to the PRF6 sets of ISITRA-2 and their framework is given below.

$\mathbf{f}_a = [0, -l_1, l_2, l_3, l_2, -l_1]$ $\mathbf{f}_{ar} = [0, k_1, k_2, k_1, 0, 0]$

$\mathbf{f}_b = [0, k_1, -k_2, k_1, 0, 0]$ $\mathbf{f}_{br} = [0, l_1, l_2, -l_3, l_2, l_1]$

Member condition-II yields

$$-l_1k_2 + l_2k_1 = 0 \cdots\cdots\cdots (33)$$

As the member condition-II yields equation (33), we have to find filter coefficients from this equation. Equation (33) consists of four variables, we have to find any four real numbers which satisfy it. This can be done in infinitely many ways. In the above framework, l_3 has no role in the reconstruction process. That is, we can choose any real number as its value.

Frameworks of 9/7 Bi-orthogonal filters:

The decomposition and the reconstruction filters of the popularly used 9/7 bi-orthogonal filters have the following frameworks.

$\mathbf{f}_a = [0, l_1, -l_2, -l_3, l_4, l_5, l_4, -l_3, -l_2, l_1]$ $\mathbf{f}_{ar} = [0, -k_1, -k_2, k_3, k_4, k_3, -k_2, -k_1, 0, 0]$

$\mathbf{f}_b = [0, -k_1, -k_2, k_3, k_4, k_3, -k_2, -k_1, 0, 0]$ $\mathbf{f}_{br} = [0, -l_1, -l_2, l_3, l_4, -l_5, l_4, l_3, -l_2, -l_1]$

This framework satisfies member condition-I given in equation (14).

Member condition-II of equation (15) yields

$$\left.\begin{aligned} l_1k_4 - l_2k_3 + l_3k_2 - l_4k_1 &= 0 \\ l_1k_2 + l_2k_3 + l_3k_4 - l_4(k_3 - k_1) + l_5k_2 &= 0 \end{aligned}\right\} \cdots\cdots\cdots (34)$$

The actual PRF sets are computed by finding the values of the filter coefficients from equation (34).

Thus, from the filter-evaluation point of view, the proposed scheme is easier than the wavelet scheme. Application specific filters can also be found by using some optimization techniques as described in appendix C. The frameworks of PRF sets obtained here are more generalized than frameworks of ISITRA-1. The PRF sets of ISITRA-1 can be thought as a special case of ISITRA-2. The PRF sets of all the bi-orthogonal wavelets are also the PRF sets of ISITRA-2 of the reconstruction scheme-1. Frameworks of PRF sets of some other bi-orthogonal wavelet filters are given in Appendix B. Once we find the PRF sets, we can use it for single level decomposition using equations (4) and (5) or for multilevel decomposition described in the next section.

3.6 Multilevel Decomposition and Reconstruction Schemes of ISITRA

In Section 3.3, we have discussed the theoretical background of perfect reconstruction in a single level subband decomposition and reconstruction scheme. The single level decomposition scheme can be easily extended to multilevel decomposition scheme to allow multi-resolution analysis. The multilevel decomposition scheme is of two types, namely, the multilevel half decomposition scheme (or simply half decomposition) and the multilevel full decomposition scheme (or simply full decomposition). In the multilevel half decomposition scheme, a signal is first decomposed into two components and successive decomposition is applied on one of the two decomposed components. It is similar to discrete wavelet transform or octave band filter bank in its decomposition and reconstruction scheme. A signal $\mathbf{b}$ is decomposed into two components $\mathbf{A}$ and $\mathbf{B}$ using the decomposition filter coefficients $\mathbf{f}_a$ and $\mathbf{f}_b$. The process of decomposition continues in one of the components, in our case component $\mathbf{A}$, up to certain desired level. The decomposed components at a particular level are the two decomposed components at that level plus the left out decomposed components of $\mathbf{B}$ at the previous levels. Thus, if a signal is decomposed up to a level-k, we get $k+1$ decomposed components. The level-3 half decomposition scheme is shown in Fig.3.5a. The reconstruction process starts from the last two decomposed components backwards in the same way as in 2-channel filter bank using single-level reconstruction scheme at each reconstruction level. In the case of full decomposition scheme, a signal is first decomposed into two components, and each of the decomposed components is again decomposed into two components. The process continues up to a

certain desired level. The decomposition scheme is similar to wavelet packet decomposition scheme. The decomposed components at a particular level are the decomposed components at that level. The level-3 full decomposition scheme is shown in Fig.3.5b. In this case if we decompose a signal up to a certain level k, we get 2^k decomposed components.

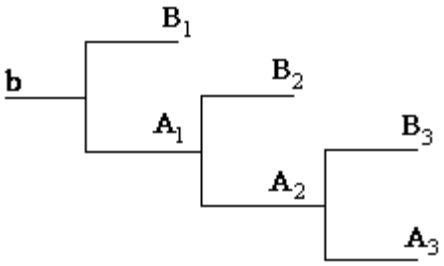


Figure 3.5a. Level-3 half decomposition

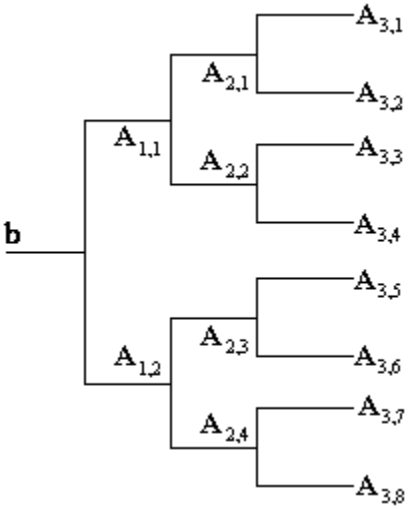


Figure 3.5b. Level-3 full decomposition

Here also, the reconstruction scheme is just the reverse process of decomposition. Reconstruction starts from the last level backward. From the 2^k components at level k, we reconstruct 2^{k-1} decomposed components of level $k-1$. The process continues till the original signal is reconstructed from the two decomposed components at level-1. The multilevel half and full decomposition schemes are described respectively in the following subsections 3.6.1 and 3.6.2.

3.6.1 Multilevel half decomposition and reconstruction

Suppose $\{\mathbf{f}_a, \mathbf{f}_b, \mathbf{f}_{ar}, \mathbf{f}_{br}\}$ is a PRF-L set of any framework of ISITRA-1 or ISITRA-2. Then in the half decomposition scheme, a signal is first decomposed into two components, say, $\mathbf{A}_1$ and $\mathbf{B}_1$, using two different decomposition filters $\mathbf{f}_a$ and $\mathbf{f}_b$. Then one of the two decomposed components (say, $\mathbf{A}$'s components) is again decomposed into two components at each successive level using the same decomposition filters. The decomposed components at a particular level-k are denoted by $\mathbf{A}_k$ and $\mathbf{B}_k$.

Let $\mathbf{b} = [b_1, b_2, b_3, \cdots, b_m]$ be a discrete signal and $\mathbf{f}_a$, $\mathbf{f}_b$ be the decomposition filters. Then we can decompose $\mathbf{b}$ up to level-k using the following half decomposition scheme.

$$\left.\begin{aligned} \mathbf{A}_k(n) &= \sum_{i=1}^{L} \mathbf{f}_a(i)\mathbf{A}_{k-1}(2n+1-i) \\ \mathbf{B}_k(n) &= \sum_{i=1}^{L} \mathbf{f}_b(i)\mathbf{A}_{k-1}(2n+1-i) \end{aligned}\right\} \cdots\cdots\cdots (35)$$

with $\mathbf{A}_1(n) = \sum_{i=1}^{L} \mathbf{f}_a(i)\mathbf{b}(2n+1-i)$ and $\mathbf{B}_1(n) = \sum_{i=1}^{L} \mathbf{f}_b(i)\mathbf{b}(2n+1-i)$

where L is the length of the decomposition filters, $n = 1, 2, \cdots, len_k$ and len_k is the length of a decomposed component at level-*k*, which can be recursively obtained from the following relation.

$$len_k = ceil\left((len_{k-1})/2\right) \quad \cdots\cdots (36)$$

And $len_1 = ceil(m/2) = \lceil m/2 \rceil$ is the length of a decomposed component at level-1. Thus, from equation (35), we can get the components of multilevel half decomposition. In the above decomposition scheme, only $\mathbf{A}$'s components are successively decomposed up to a certain level-k. The maximum decomposable level in multilevel decomposition is fixed and it depends on the length of the signal and the length of the decomposition filters. Suppose we use decomposition filters of length L at each decomposition level. Then we will say the signal is decomposable only up to level-k if $len_k < L \le len_{k-1}$.

All the $\mathbf{B}$'s components, $\mathbf{B}_1, \mathbf{B}_2, \cdots, \mathbf{B}_k$ and $\mathbf{A}_k$ represent the decomposed components at level-k. Thus we get $k+1$ decomposed components at level-k decomposition. In other words, the signal is decomposed into $k+1$ components. Once the signal is decomposed up to a desired level into several components, the next task is reconstruction of the original signal from those decomposed components. The reconstruction of a signal from its decomposed components requires reconstruction filters $\mathbf{f}_{ar}$ and $\mathbf{f}_{br}$ of the corresponding decomposition filters $\mathbf{f}_a$ and $\mathbf{f}_b$. The multilevel reconstruction equation can be easily obtained from the reconstruction equations (9) and (12) in the following way. Let

$$D_1 = \sum_{i=1}^{L} \mathbf{f}_a(2i)\mathbf{f}_{ar}(2i) + \mathbf{f}_b(2i)\mathbf{f}_{br}(2i) \neq 0$$

$$D_2 = \sum_{i=1}^{L} \mathbf{f}_a(2i-1)\mathbf{f}_{ar}(2i-1) + \mathbf{f}_b(2i-1)\mathbf{f}_{br}(2i-1) \neq 0$$

Then recursive reconstruction relations are given by the following equations by extending equations (9) and (12) as follows.

$$\mathbf{A}_{k-1}(2n+1-L) = \frac{\sum_{i=1}^{L/2} \mathbf{f}_{ar}(L+2-2i)\mathbf{A}_k(n+1-i) + \mathbf{f}_{br}(L+2-2i)\mathbf{B}_k(n+1-i)}{D_1} \quad \cdots\cdots(37)$$

$$\mathbf{A}_{k-1}(2n+2-L) = \frac{\sum_{i=1}^{L/2} \mathbf{f}_{ar}(L+1-2i)\mathbf{A}_k(n+1-i) + \mathbf{f}_{br}(L+1-2i)\mathbf{B}_k(n+1-i)}{D_2} \quad \cdots\cdots(38)$$

To make the index of the reconstructed components start from 1, we can write equations (37) and (38) as

$$\mathbf{A}_{k-1}(2n-1) = \frac{\sum_{i=1}^{L/2} \mathbf{f}_{ar}(L+2-2i)\mathbf{A}_k(n+L/2-i) + \mathbf{f}_{br}(L+2-2i)\mathbf{B}_k(n+L/2-i)}{D_1} \quad \cdots(39)$$

$$\mathbf{A}_{k-1}(2n) = \frac{\sum_{i=1}^{L/2} \mathbf{f}_{ar}(L+1-2i)\mathbf{A}_k(n+L/2-i) + \mathbf{f}_{br}(L+1-2i)\mathbf{B}_k(n+L/2-i)}{D_2} \quad \cdots(40)$$

In case $D_1 = D_2$ ($= D$, say), as in the cases of the frameworks of PRF sets of most commonly used orthogonal transforms, we can combine equations (39) and (40) as

$$\mathbf{A}_{k-1}(n) = \frac{1}{D}\{\mathbf{f}_{ar}(p-2i)\mathbf{A}_k(n) + \mathbf{f}_{br}(p-2i)\mathbf{B}_k(n)\} \cdots\cdots (41)$$

where $n = 1, 2, \cdots, len_{k-1}$; $p = L+1+n\%2$; $A_k(n) = B_k(n) = 0$ for $len_k < n < 1$ and $s\%t$ denotes the remainder when s is divided by t.

Equation (41) gives a generalized way of recursive reconstruction in the half decomposition scheme in ISITRA. This way of reconstruction avoids the necessity of up sampling the decomposed components and hence avoids the convolution of longer components with the reconstruction filters.

3.6.2 Multilevel full decomposition and reconstruction

Under this decomposition scheme, a signal is first decomposed into two parts using two decomposition filters. And each decomposed component is again decomposed into two parts at the next level. The process is continued up to a certain level and the maximum decomposable level is given by equation (36). Hence if the signal is decomposed up to level-*k*, then the number of decomposed components is 2^k. Here, the decomposed components will be denoted by a single symbol with two indices suffixed to it. The first suffix denotes the level of decomposition and the second suffix denotes the component number, i.e., $\mathbf{A}_{i,j}$ denotes j^{th} component of the i^{th} level decomposition.

Let $\mathbf{b}$ be a discrete signal of length *m* and $\mathbf{f}_1, \mathbf{f}_2$ be the decomposition filters. Then the level-*k* full decomposition scheme can be written mathematically as

$$\mathbf{A}_{k,j}(n) = \sum_{i=1}^{L} \mathbf{f}_{(j-1)\%2+1}(i)\mathbf{A}_{k-1,\lceil j/2 \rceil}(2n+1-i) \quad \cdots\cdots\cdots (42)$$

with $\mathbf{A}_{1,1}(n) = \sum_{i=1}^{L} \mathbf{f}_1(i)\mathbf{b}(2n+1-i)$ and $\mathbf{A}_{1,2}(n) = \sum_{i=1}^{L} \mathbf{f}_2(i)\mathbf{b}(2n+1-i)$,

where $j = 1,2,\cdots,2^{k-1}$; $n = 1, 2, \cdots, len_k$;

$\mathbf{A}_{k,j}(n) = \mathbf{B}_{k,j}(n) = 0$ for $len_k < n < 1$

In this decomposition scheme, the decomposed components at the level-k are the 2^k components at that level. In other words, the signal is decomposed into 2^k components. The reconstruction scheme as in the half decomposition scheme, requires reconstruction filters $\mathbf{f}_{1r}$ and $\mathbf{f}_{2r}$ of the corresponding decompositions filters $\mathbf{f}_1$ and $\mathbf{f}_2$. For simplicity, let us consider the case where $D_1 = D_2 = D$.

Then the recursive reconstruction in the case of full decomposition scheme is given by

$$\mathbf{A}_{k-1,\,j}(n) = \frac{1}{D}\{\sum_{i=1}^{L/2} \mathbf{f}_{1r}(p-2i)\mathbf{A}_{k,2j-1}(n+L/2-i) + \mathbf{f}_{2r}(p-2i)\mathbf{A}_{k,2j}(n+L/2-i)\} \cdots\cdots (43)$$

Equation (42) gives the generalized recursive full decomposition scheme of ISITRA. And Equation (43) gives the generalized recursive full reconstruction scheme of ISITRA.

3.7 Multi-resolution Decomposition Scheme of ISITRA

The representation of a signal in terms of a basis of different scales is known as multi-resolution signal representation. In the case of wavelet the scale of resolution is infinite as it is derived in the case of a continuous signal. Practically, there is a lot of difference from the theoretical scale of resolution. The scale cannot be infinite; it always has a finite range. In the case of ISITRA, the scale has fixed limits as the derivation of ISITRA is based on array context. There is nothing in between two consecutive array elements. Hence the shortest scale we can have is 1 corresponding to the first level of decomposition where the index of the signal (array) advances two elements for the computation of a term of a decomposed component. This is the finest resolution that can be achieved in a decomposed component in ISITRA. The multi-resolution representation of a signal in ISITRA is achieved in multilevel decomposition. The level of decomposition corresponds to the scale of resolution. The lower the scale, the finer is the resolution. The level zero represents the original signal. We cannot represent a signal in a finer resolution than that of the original signal. A higher level of decomposition corresponds to a coarser resolution. The decomposition process of ISITRA represents a signal from a finer resolution to a coarser resolution. The reconstruction process is the representation of a signal from a coarser to a finer resolution.

The multi-resolution of a signal in an octave band filter-bank decomposition scheme is obtained because of the down-sampling operation used at every level of decomposition. In the case of ISITRA, it is obtained through the indexing technique used during decomposition. The decomposition filters get elongated by certain amount at each successive decomposition level because of down-sampling operation in filter bank or indexing in ISITRA. Thus, the decomposition at higher levels can be thought of as filtering operations with longer filters at large translation steps. Before deriving how much amount a filter get elongated at each successive level, let us have a closer look at the filtering operation or convolution in terms of sliding windows.

The decomposition filter can also be thought of as a window which moves uniformly (by a certain fixed amount each time) to cover the whole signal. The length of the window increases with an increase in decomposition level. There is a fixed amount by which a window shifts at a particular decomposition level irrespective of the length of the window.

At the first level, the window shifts by 2 positions each time it shifts from left to right to cover the whole signal. At the second level, the window shifts by 4 positions each time from left to right. In general, at *k*-th level the window shifts from left to right by 2^k positions each time. The shifting positions of a filter at the first, second and third levels of decomposition are shown in Fig.3.6.

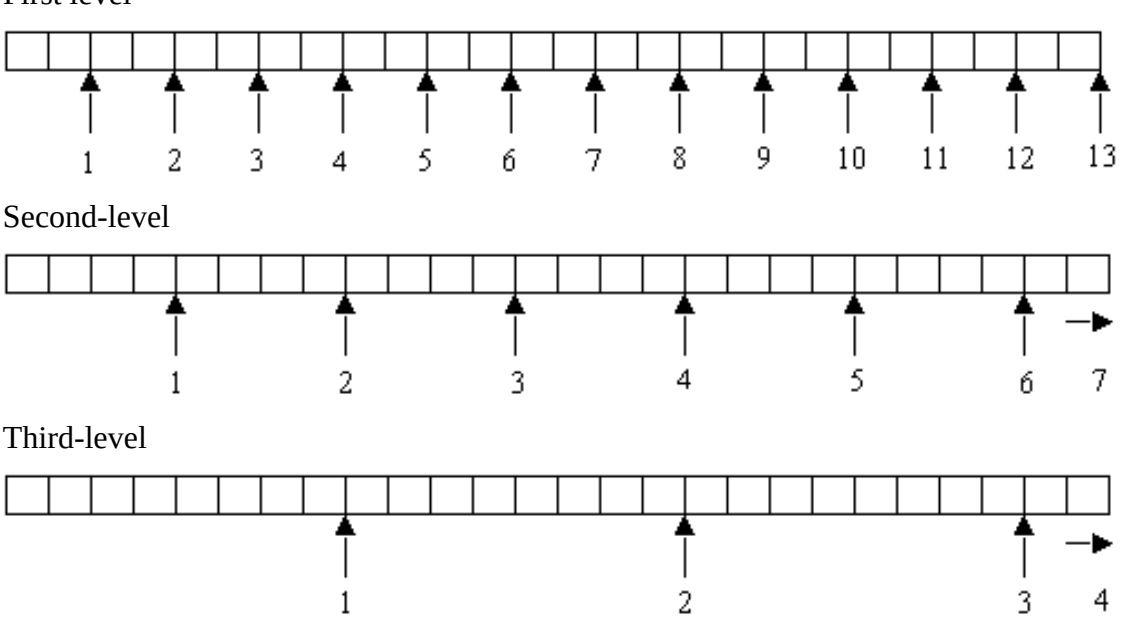


Figure 3.6 Position of shifts of an effective filter at different decomposition levels.

Thus, we get different resolutions at different levels of decomposition. The vertical arrows show the position right end of the filter window at each shift corresponding to the computation of a term of decomposed component at a particular level. The horizontal arrows show the next position of the window goes beyond the length of the signal. In the decomposition scheme shown above, the filtering operation starts from left end of the signal. The filtering operation may also start from the right side. We may get different decomposed components if we start the filtering operation from the right to the left in the decomposition scheme, as the window may cover different components of the signal at each shift.

Definition 3.3 *An effective filter* of a decomposition filter $\mathbf{f}$ at a particular decomposition level is the filter obtained from $\mathbf{f}$, which can be directly used on the original signal to get the decomposed components at that decomposition level. In other words, the decomposed

components at level- k can be obtained from the original signal in a single level decomposition by ${}^{e}\mathbf{f}_k$ where ${}^{e}\mathbf{f}_k$ is the effective filter at level-*k*.

Theorem 3.1 The length of an effective filter of a decomposition filter $\mathbf{f}$ at level-*k*, denoted by $L_e(k)$ is a function of k and is given by

$$L_e(k) = 1 + (2^k - 1)(L - 1) \quad \cdots\cdots\cdots (44)$$

where L is the length of the decomposition filter $\mathbf{f}$.

Proof. We will use induction method to prove this.

We know, the level-1 decomposed components using filter $\mathbf{f}_a$ of length L is

$$\mathbf{A}_1(n) = \sum_{i=1}^{L} \mathbf{f}_a(i)\mathbf{b}(2n + 1 - i)$$

The last term in the decomposed component is $LT_1 = \mathbf{f}_a(L)\mathbf{b}(2n - (L-1))$. And the actual effective length of the filter is L.

The decomposed components at level-2 using the same decomposition filter is

$$\mathbf{A}_2(n) = \sum_{i=1}^{L} \mathbf{f}_a(i)\mathbf{A}_1(2n + 1 - i)$$

$$= \sum_{i=1}^{L} \mathbf{f}_a(i) \sum_{i_1=1}^{L} \mathbf{f}_a(i_1)\mathbf{b}(4n + 3 - 2i - i_1)$$

The last term in the second level decomposition occurs when $i = i_1 = L$ and is

$$LT_2 = \mathbf{f}_a^2(L)\mathbf{b}(4n - 3(L-1))$$

And the actual effective length at the level-2 decomposition is $1 + 3(L-1)$.

Similarly, the last terms in the level-3 and level-4 decomposition are

$$LT_3 = \mathbf{f}_a^3(L)\mathbf{b}(8n - 7(L-1))$$

$$LT_4 = \mathbf{f}_a^4(L)\mathbf{b}(16n - 15(L-1))$$

And their actual effective filter lengths are $1 + 7(L-1)$ and $1 + 15(L-1)$ respectively.

Proceeding in this way, the last term at the *k*-th level decomposition is

$$LT_k = \mathbf{f}_a^k(L)\mathbf{b}(2^k n - (2^k - 1)(L-1))$$

Hence the effective length of the decomposition filter at level-*k* is

$$L_e(k) = 1 + (2^k - 1)(L - 1)$$

Alternatively, we can express the effective length of a decomposition filter at level-*k* recursively as

$$L_e(k) = 2^{k-1}(L-1) + L_e(k-1) \quad \cdots\cdots\cdots (45)$$

This can be proved easily.

This is how the length of a filter changes with the level of decomposition to give different resolutions at different levels. The following Table 3.2 shows that the effective length of the filter increases with increase in level of decomposition. The first column Lev–1gives the original lengths of the filters and the other columns give the lengths of the effective filters at the corresponding decomposition levels. For example, the original filter of length 10 gets elongated to lengths 28, 64, 136, … in the second, third, fourth, … decomposition levels.

Lev-1	Lev-2	Lev-3	Lev-4	Lev-5	Lev-6	Lev-7	Lev-8	Lev-9	Lev-10
2	4	8	16	32	64	128	256	512	1024
4	10	22	46	94	190	382	766	1534	3070
6	16	36	76	156	316	636	1276	2556	5116
8	22	50	106	218	442	890	1786	3578	7162
10	28	64	136	280	568	1144	2296	4600	9208

Table 3.2 Lengths of effective filters at different decomposition levels

Theorem 3.2 Let $\mathbf{f}$ be a decomposition filter of length 2, then the effective filter at level-*k* using this decomposition filter is given by`

$${}^e\mathbf{f}_k = [\mathbf{f}(1) * \mathbf{f}_{k-1}, \mathbf{f}(2) * {}^e\mathbf{f}_{k-1}]$$

Proof. This can be proved easily from the decomposition scheme of ISITRA or from Theorem 1.3 below.

Theorem 3.3 Let $\mathbf{f}$ be a decomposition filter of length *L*. Then the effective filter at level-*k* using this decomposition filter is given by

$${}^e\mathbf{f}_k(j) = \sum_{i=0}^{L} \mathbf{f}(i)\, {}^e\mathbf{f}_{k-1}(j + 2^{k-1}(1-i))$$

where ${}^{e}\mathbf{f}_k$ is the effective filter at level-k and $j = 1,2,3,\cdots, L_e(k)$ for the corresponding decomposition filter.

Proof. First level decomposition of using $\mathbf{f}$ is

$$\mathbf{A}_1(n) = \sum_{i=1}^{L} \mathbf{f}(i)\, \mathbf{b}(2n+1-i)$$

The effective filter at level-1 is the filter itself. That is, $\mathbf{f} = {}^{e}\mathbf{f}_1$. So the above equation can also be written as

$$\mathbf{A}_1(n) = \sum_{i=1}^{L} {}^{e}\mathbf{f}_1(i)\, \mathbf{b}(2n+1-i)\,.$$

Second level decomposition of using $\mathbf{f}$ is

$$\mathbf{A}_2(n) = \sum_{i=1}^{L} \mathbf{f}(i)\ \mathbf{A}_1(2n+1-i)$$

$$= \sum_{i=1}^{L} \mathbf{f}(i) \sum_{i_1=1}^{L} {}^{e}\mathbf{f}_1(i_1)\, \mathbf{b}(2(2n+1-i)+1-i_1)$$

$$= \sum_{i=1}^{L} \mathbf{f}(i) \sum_{i_1=1}^{L} {}^{e}\mathbf{f}_1(i_1)\, \mathbf{b}(4n+3-2i-i_1)$$

We know that the level-2 decomposed components can be obtained from the original signal using the effective filter at level-2, by shifting the filter window 4 positions each time. That is, we can also write the decomposed component at level-2 as

$$\mathbf{A}_2(n) = \sum_{i=1}^{L} {}^{e}\mathbf{f}_2(i)\, \mathbf{b}(4n+1-i)$$

Comparing the coefficients of various terms of $\mathbf{b}$'s in the above equation, we have

$${}^{e}\mathbf{f}_2 = [\mathbf{f}^2(1), \mathbf{f}(1)\mathbf{f}(2), \mathbf{f}(1)\mathbf{f}(3) + \mathbf{f}(1)\mathbf{f}(2), \cdots, \mathbf{f}(L-1)\mathbf{f}(L), \mathbf{f}^2(L)]$$

Third level decomposition of using $\mathbf{f}$ is

$$\mathbf{A}_3(n) = \mathbf{f}(1){}^{e}\mathbf{f}_2(1)b(8n) + \mathbf{f}(1){}^{e}\mathbf{f}_2(2)\mathbf{b}(8n-1) + \mathbf{f}(1){}^{e}\mathbf{f}_2(3)\mathbf{b}(8n-2) +$$
$$\mathbf{f}(1){}^{e}\mathbf{f}_2(4)\mathbf{b}(8n-3) + \{\mathbf{f}(1){}^{e}\mathbf{f}_2(5) + \mathbf{f}(2){}^{e}\mathbf{f}_1(1)\}\mathbf{b}(8n-4) + \{\mathbf{f}(1){}^{e}\mathbf{f}_2(6) +$$
$$\mathbf{f}(2){}^{e}\mathbf{f}_2(2)\}\mathbf{b}(8n-5) + \{\mathbf{f}(1){}^{e}\mathbf{f}_2(7) + \mathbf{f}(2){}^{e}\mathbf{f}_2(3)\}\mathbf{b}(8n-6) + \cdots +$$
$$\mathbf{f}(L){}^{e}\mathbf{f}_2(1+3(L-1)\}\mathbf{b}(8n-(2^3-1)(L-1))$$

The effective filter at level-3 is the array of the coefficients of $\mathbf{b}$'s in the above equation. That is,

$${}^{e}\mathbf{f}_3 = [\mathbf{f}(1){}^{e}\mathbf{f}_2(1), \mathbf{f}(1){}^{e}\mathbf{f}_2(2), \mathbf{f}(1){}^{e}\mathbf{f}_2(3), \mathbf{f}(1){}^{e}\mathbf{f}_2(4), \mathbf{f}(1){}^{e}\mathbf{f}_2(5)+\mathbf{f}(2){}^{e}\mathbf{f}_2(1),$$
$$\mathbf{f}(1){}^{e}\mathbf{f}_2(6)+\mathbf{f}(2){}^{e}\mathbf{f}_2(2), \cdots, \mathbf{f}(L-1){}^{e}\mathbf{f}_2(3L-2), \mathbf{f}(L){}^{e}\mathbf{f}_2(1-3(L-1))]$$

Proceeding in this way, we can write ${}^{e}\mathbf{f}_k$ in terms of ${}^{e}\mathbf{f}_{k-1}$ as

$${}^{e}\mathbf{f}_k(j) = \sum_{i=1}^{L} \mathbf{f}(i)\ {}^{e}\mathbf{f}_{k-1}(j + 2^{k-1}(1-i)) \quad \cdots\cdots\cdots (46)$$

Theorem 3.4 If ${}^{e}\mathbf{f}_k$ is an effective filter at level-*k*, for a decomposition filter $\mathbf{f}_a$, then the decomposed component of a signal at level-*k* due to successive decomposition using $\mathbf{f}_a$, is the same as the decomposed component of the signal obtained by using the ${}^{e}\mathbf{f}_k$ as the filter window with 2^k shifts at each move of the filter.

In the case of wavelet transform the scale of resolution extends from negative infinity to positive infinity, this may be true for continuous signal. But it is meaningless in an array context because there is no element between two successive elements. So the length of the filter cannot be less than 1. Also the number of decomposable level is infinite which is quite contrary to the practical reality. The meaning of resolution at negative scale is an irrelevant concept in array context. As data are manipulated in array form in a computer, the theory given here can be implemented easily. And there is no discrepancy between theoretical and implementation point of view.

3.8 Multi-channel Decomposition of ISITRA

The decomposition scheme so far we have discussed is done on the entire signal. Sometimes we may want decomposition on some selective elements of the signal or we may want to employ parallel processing in the decomposition scheme. For that purpose we divide the signal into several segments. The lengths of the segments may or may not be equal. The process of segmenting a signal into several segments and processing each segment separately is known as a multi-channel scheme. In this scheme, a signal is divided into several input segments and the usual decomposition is performed in each segment. Both half and full level decompositions are possible in some or all of the segments using filters of

ISITRA-1 or ISITRA-2. Moreover, the process of multi-channel decomposition may be repeated in some or all of the decomposed components. This way of decomposition may be useful in various applications, such as analysis, compression, data encryption etc.

3.8.1 Channel decomposition

The process of the division of signal $\mathbf{b}$ of length m into two or more segments will be known as channel decomposition. It can be done in many different ways but we discuss the following two ways. Here, the number of segments, M, is assumed to be even.

First method: Here we simply divide the signal into M segments, by taking $\lceil m/M \rceil$ successive elements in the following way.

Channel-1, $\mathbf{a}_1 = [\mathbf{b}(1) : \mathbf{b}(\lceil m/M \rceil)]$;

Channel-2, $\mathbf{a}_2 = [\mathbf{b}(\lceil m/M \rceil + 1) : \mathbf{b}(2 * \lceil m/M \rceil)]$

Channel-3, $\mathbf{a}_3 = [\mathbf{b}(2 * \lceil m/M \rceil + 1) : \mathbf{b}(3 * \lceil m/M \rceil)]$

Proceeding in this way, we can write

Channel-r, $\mathbf{a}_r = [\mathbf{b}((r-1) * \lceil m/M \rceil + 1) : \mathbf{b}(r * \lceil m/M \rceil)]$, $r = 1,2,\cdots,M$

And the last channel-M is

Channel-M, $\mathbf{a}_M = [\mathbf{b}((M-1) * \lceil m/M \rceil + 1) : \mathbf{b}(m)]$

Second method: Here we segment the signal in such a way that, first segment consists of $\mathbf{a}_1(n) = \mathbf{b}((n-1) * M + 1)$ elements of the signal, 2nd channel consists of $\mathbf{a}_2(n) = \mathbf{b}((n-1) * M + 2)$ elements of the signal and so on as shown below.

Channel-1, $\mathbf{a}_1(n) = \mathbf{b}((n-1) * M + 1)$

Channel-2, $\mathbf{a}_2(n) = \mathbf{b}((n-1) * M + 2)$

Channel-3, $\mathbf{a}_3(n) = \mathbf{b}((n-1) * M + 3)$

Proceeding in this way in this way

Channel-r, $\mathbf{a}_r(n) = \mathbf{b}((n-1) * M + r)$

and

Channel-M, $\mathbf{a}_M(n) = \mathbf{b}((n-1) * M + M)$

where $n = 1,2,3,\cdots,\lceil (m+1-r)/M \rceil$ for any channel r of M channels.

The above two methods are nothing but taking rows and columns, when a signal or an array of length m is converted into a matrix of order $M \times \lceil m/M \rceil$. In the first method, each channel corresponds to each row of the matrix. And in the second method, each channel corresponds to each column of the matrix.

3.8.2 Channel interpolation

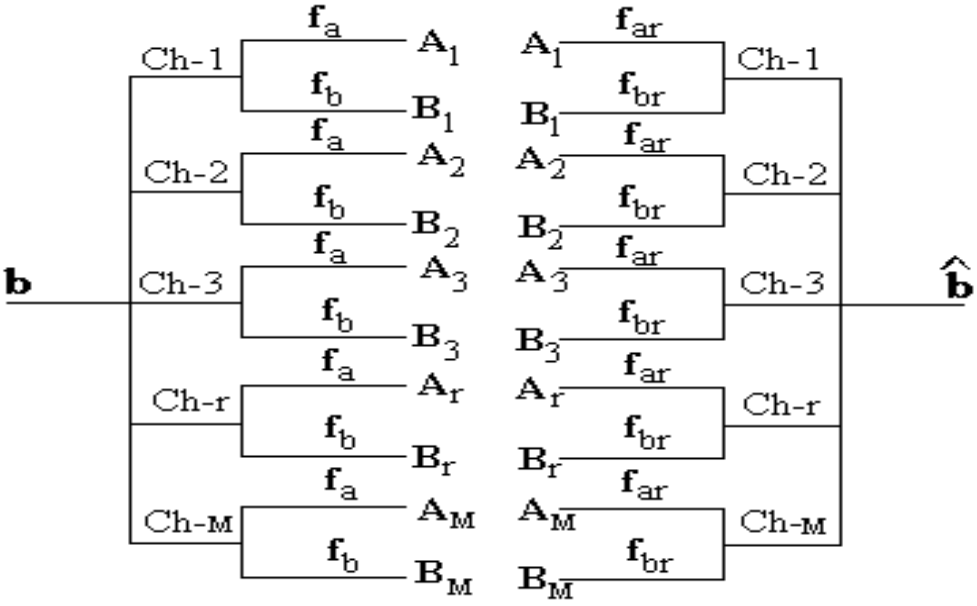
The reconstruction of a signal from two or more segments will be known as channel interpolation. We discuss the channel interpolation scheme of the above two channel decomposition methods. As the first method of channel decomposition corresponds to segmenting the signal by keeping $\lceil m/M \rceil$ successive segments, the reconstruction is very simply obtained by putting the segments side by side in the proper order. That is, if $\mathbf{a}_1, \mathbf{a}_2, \cdots, \mathbf{a}_M$ are the *M* segments of the signal $\mathbf{b}$ of length m using the first method, then we can simply obtain signal $\mathbf{b}$ as $\mathbf{b} = [\mathbf{a}_1, \mathbf{a}_2, \cdots, \mathbf{a}_M]$. For the second method, it needs two steps. Form a matrix $\mathbf{A}$ of order $M \times \lceil m/M \rceil$ with *M* channel components as $\mathbf{A} = [\mathbf{a}_1, \mathbf{a}_2, \cdots, \mathbf{a}_M]'$, the original signal is reconstructed by putting the transpose of the columns as the elements. Let $\mathbf{A}(:, c)'$ denote the transpose of *c*-th column of the matrix, then we can write, $\mathbf{b} = [\mathbf{A}(:,1)', \mathbf{A}(:,2)', \mathbf{A}(:,3)', \cdots, \mathbf{A}(:, M)']$.

Or from a matrix of order $M \times \lceil m/M \rceil$ with the transpose of *M* channel components as $\mathbf{A} = [\mathbf{a}'_1, \mathbf{a}'_2, \mathbf{a}'_3, \cdots, \mathbf{a}'_M]$, the original signal is reconstructed by putting the columns as the elements as $\mathbf{b} = [\mathbf{A}(:,1), \mathbf{A}(:,2), \mathbf{A}(:,3), \cdots, \mathbf{A}(:, M)]$.

This is how we can reconstruct the original signal from its *M* channel segments in time domain. Note that the *M*-channel components in the second method of channel decomposition correspond to the coefficients of *M*-channel polyphase representation.

3.8.3 2M-Channel filter bank of ISITRA

Once a signal is divided into a desired number of segments, the usual decomposition and reconstruction schemes of ISITRA can be applied in each segment of each channel. The single level 2*M*-channel decomposition and reconstruction schemes are shown in Fig.3.7 where $\mathbf{A}_r$ and $\mathbf{B}_r$ denote the decomposed components of the segment of the signal corresponding to channel-*r*.



Decomposition Scheme Reconstruction Scheme

Figure 3.7 2*M*-Channel filter bank of ISITRA

The input signal $\mathbf{b}$ is first divided into *M* segment inputs using a channel decomposition scheme. The input of each channel is decomposed into two components using the decomposition filters $\mathbf{f}_a$ and $\mathbf{f}_b$. Decomposing a signal in this way enhances the performance of the decomposition scheme as we perform filtering operations on shorter signals. Another plus point of such a multi-channel filter bank is that we don't need to deal with a new perfect reconstruction theory as in the existing multi-channel filter bank where finding aliasing free filters are difficult to find [48,152,159]. The perfect reconstruction theory of 2-channel ISITRA still holds in multi-channel case also. So, we do not need to find different filters for such multi-channel filter bank. Any PRF set of 2-channel ISITRA can be used in the multi-channel filter bank of ISITRA also.

3.9 Multi-Channel Transmultiplexer of ISITRA

A subband filter bank is an analysis/synthesis structure. Its dual, that is, the synthesis/analysis structure is known as transmultiplixer. Such a structure was studied by Bellanger [17] for telephony application. In a transmultiplixer, the front-end corresponds to a synthesis section. In this section several time-division-multiplexed signals are combined into a composite signal which is then transmitted in a single channel. At the receiver end, the composite signal is separated into corresponding segments using analysis filters. In the

existing model of *M*-channel tranmultiplexer [152, 159], it is difficult to get perfect reconstruction filters due to three main types of error sources, amplitude distortion, phase distortions and cross talks. More on transmultiplexer and the sources of errors for achieving perfect reconstruction can be found elsewhere [6, 7, 18, 69, 101, 102, 103, 104, 111, 154].

From the multi-channel filter bank scheme of ISITRA, we know that achieving perfect reconstruction is not difficult. If we use the dual of it as a tranmultiplexer, then the perfect reconstruction would not be difficult to achieve. Such a tranmultiplexer is shown in Fig. 3.8 below. It is not difficult to get prefect reconstruction in such a transmultiplexer, quite different from the *M*-channel transultiplexer. In the front end/transmitter end, $\mathbf{A}_i$'s and $\mathbf{B}_i$'s are the individual signals. For each pair of $\mathbf{A}_i$'s and $\mathbf{B}_i$'s we construct a signal $\mathbf{b}_i$ using the reconstruction filters $\mathbf{f}_{ar}$ and $\mathbf{f}_{br}$. The signal $\mathbf{b}_i$'s are combined together to produce a composite signal $\mathbf{y}$ using a proper channel interpolation. At the receiver end, the

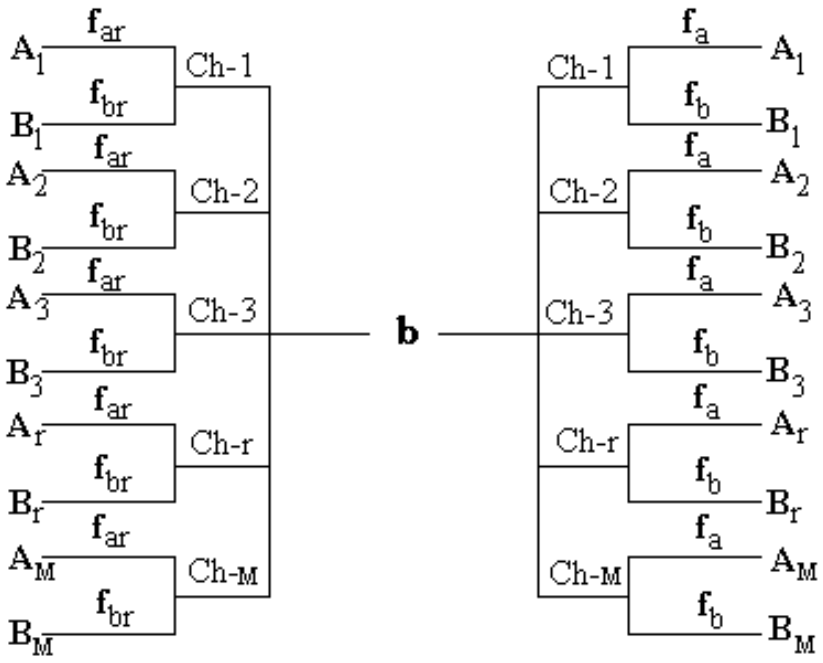


Figure 3.8 2M-Channel Transmultiplexer

composite signal $\mathbf{b}$ is divided into various channel components $\mathbf{b}_i$ using their corresponding channel decomposition scheme. From each channel input $\mathbf{b}_i$, we decompose the

corresponding individual signals $\mathbf{A}_i$ and $\mathbf{B}_i$ using the decomposition filters $\mathbf{f}_a$ and $\mathbf{f}_b$. We can exactly reconstruct the original individual signals without any distortion in the receiver end if the filters $\mathbf{f}_a$, $\mathbf{f}_b$, $\mathbf{f}_{ar}$ and $\mathbf{f}_{br}$ satisfy the member condition of perfect reconstruction theory of transmultiplexer as describing next.

3.9.1 Perfect reconstruction theory of 2-channel transmultiplexer

Let $\mathbf{A}$ and $\mathbf{B}$ be any two signals to be multiplexed. For simplicity, let us assume that the length of each signal is M. And $\mathbf{b}$ is the multiplexed signal obtained from the following relations.

$$\mathbf{b}(2n-1) = \sum_{i=1}^{L/2}\{\mathbf{f}_{ar}(2i)\mathbf{A}(n-1+i) + \mathbf{f}_{br}(2i)\mathbf{B}(n-1+i)\} \quad \cdots\cdots\cdots (47)$$

$$\mathbf{b}(2n) = \sum_{i=1}^{L/2}\{\mathbf{f}_{ar}(2i-1)\mathbf{A}(n-1+i) + \mathbf{f}_{br}(2i-1)\mathbf{B}(n-1+i)\} \quad \cdots\cdots\cdots (48)$$

These two equations can be combined as

$$\mathbf{b}(n) = \sum_{i=1}^{L/2}\{\mathbf{f}_{ar}(2i-(n-1)\%2)\mathbf{A}(\lceil n/2\rceil - 1 + i) + \mathbf{f}_{br}(2i-(n-1)\%2)\mathbf{A}(\lceil n/2\rceil - 1 + i)\} \quad \cdots\cdots (49)$$

Equation (49) gives the transmultiplexed signal. This signal is to be transmitted from the transmitter end. The next task is retrieval of the signal components $\mathbf{A}$ and $\mathbf{B}$ from its composite transmultiplexed signal.

From ISITRA, we know that we can decompose a signal into two components from two decomposition filters. Let $\mathbf{f}_a$ and $\mathbf{f}_b$ be the two decomposition filters with which we decompose the composite signal into two components $\hat{\mathbf{A}}$ and $\hat{\mathbf{B}}$. That is,

$$\left.\begin{aligned}\hat{\mathbf{A}}(n) &= \sum_{i=1}^{L}\mathbf{f}_a(i)\mathbf{b}(2n+1-i)\\ \hat{\mathbf{B}}(n) &= \sum_{i=1}^{L}\mathbf{f}_a(i)\mathbf{b}(2n+1-i)\end{aligned}\right\} \quad \cdots\cdots\cdots (50)$$

Using equation (47), we can write equation (48) as

$$\hat{\mathbf{A}}(n) = \sum_{j=1}^{L}\mathbf{f}_a(j)\sum_{i=1}^{L/2}\{\mathbf{f}_{ar}(t_1)\mathbf{A}(t_2) + \mathbf{f}_{br}(t_1)\mathbf{B}(t_2)\} \quad \cdots\cdots\cdots (51)$$

where $t_1 = 2i-(2n-j)\%2$ and $t_2 = \lceil (2n+1-j)/2\rceil - 1 + i$.

Similarly, we can write

$$\hat{\mathbf{B}}(n) = \sum_{j=1}^{L} \mathbf{f_b}(j) \sum_{i=1}^{L/2} \{\mathbf{f}_{ar}(t_1)\mathbf{A}(t_2) + \mathbf{f}_{br}(t_1)\mathbf{B}(t_2)\} \quad \cdots\cdots\cdots (52)$$

Equations (51) and (52) can be written as

$$\hat{\mathbf{A}}(n) = \sum_{r=1}^{L-1} \{\sum_{i=1}^{2r} \mathbf{f}_a(i)\mathbf{f}_{ar}(L-2r+i)\mathbf{A}(n+L/2-r)$$

$$+ \sum_{i=1}^{2r} \mathbf{f}_a(i)\mathbf{f}_{br}(L-2r+i)\mathbf{B}(n+L/2-r)\} \quad \cdots\cdots\cdots (53)$$

$$\hat{\mathbf{B}}(n) = \sum_{r=1}^{L-1} \{\sum_{i=1}^{2r} \mathbf{f}_b(i)\mathbf{f}_{ar}(L-2r+i)\mathbf{A}(n+L/2-r)$$

$$+ \sum_{i=1}^{2r} \mathbf{f_b}(i)\mathbf{f}_{br}(L-2r+i)\mathbf{B}(n+L/2-r)\} \quad \cdots\cdots\cdots (54)$$

Putting $\sum_{i=1}^{2r} \mathbf{f}_a(i)\mathbf{f}_{br}(L-2r+i) = 0 \quad \forall\, r \cdots\cdots (55)$

in equation (54), we can eliminate **B** 's components and we have

$$\hat{\mathbf{A}}(n) = \sum_{i=1}^{2r} \mathbf{f}_a(i)\mathbf{f}_{ar}(L-2r+i)\mathbf{A}(n+L/2-r) \quad \cdots\cdots (56)$$

From equation (55) we can reconstruct **A** 's components as

$$\mathbf{A}(n) = \frac{\hat{\mathrm{A}}(n)}{\sum_{i=1}^{L} \mathbf{f}_a(i)\mathbf{f}_{ar}(i)} \quad \cdots\cdots (57)$$

provided $\sum_{i=1}^{2r} \mathbf{f}_a(i)\mathbf{f}_{ar}(L-2r+i) = 0, \quad \forall\, r \neq L/2 \quad \cdots\cdots (58)$

We can retrieve **B** 's components from equation (54) in a similar way.

Putting $\sum_{i=1}^{2r} \mathbf{f}_b(i)\mathbf{f}_{ar}(L-2r+i) = 0 \quad \cdots\cdots (59)$

in equation (54), we have

$$\hat{\mathbf{B}}(n) = \sum_{i=1}^{2r} \mathbf{f}_b(i)\mathbf{f}_{br}(L-2r+i)\mathbf{B}(n+L/2-r) \quad \cdots\cdots (60)$$

From equation (59) we can reconstruct **B** 's components as

$$\mathbf{B}(n) = \frac{\hat{\mathbf{B}}(n)}{\sum_{i=1}^{L} \mathbf{f}_b(i)\mathbf{f}_{br}(i)} \quad \cdots\cdots(61)$$

provided $\sum_{i=1}^{2r} \mathbf{f}_b(i)\mathbf{f}_{br}(L-2r+i) = 0, \quad \forall\, r \neq L/2 \quad \cdots\cdots(62)$

Equations (55) and (59) together will be known as member condition-I of transmultiplexer. And equations (58) and (62) will be known as its member condition-II. As in ISITRA, the decomposition and reconstruction filters satisfying both member conditions will be known as the PRF set of the transmultiplexer. Thus, if we use decomposition and reconstruction filters satisfying these two member conditions, then we would be able to get perfect reconstruction of the signal components from its composite transmultiplexed signal.

There are various ways to choose the decomposition and reconstruction filters in order to satisfy the member conditions. One simple way is as follows.

$\mathbf{f}_a = \mathbf{f}_{ar}, \mathbf{f}_b = \mathbf{f}_{br}$ and $\mathbf{f}_b(i) = (-1)^i \mathbf{f}_a(L+1-i)$ or $\mathbf{f}_b(i) = (-1)^{i-1} \mathbf{f}_a(L+1-i)$.

With such a choice, the filters $\mathbf{f}_a$ and $\mathbf{f}_b$ are said to be Quadrature Mirror Filters. Other ways of finding PRF sets of transmultiplexer are similar to those of ISITRA. We first choose filters that satisfy member condition-I, and then find actual PRF set members from member condition-II.

With the above choice of QMF type PRF sets of transmultiplexer, member condition-I is easily satisfied. We have to find the coefficients of filters from member condition-II. Let us consider the set of equations of the member condition-II for the first few short-length filters.

For filters of length 2:

For filters of length-2, member condition-II vanishes. Hence we can choose any two real numbers (both not zero) as the coefficients of a filter.

For filters of length 4:

For filters of length-4, both the sets of equations of member condition-II yield the relation $\mathbf{f}_a(1)\mathbf{f}_a(3) + \mathbf{f}_a(2)\mathbf{f}_a(4) = 0$.

We have to choose the coefficients of $\mathbf{f}_a$ such that they satisfy the above equation. As the choice can be made in infinitely many ways, we can choose infinitely many $\mathbf{f}_a$'s of length-4. We will find here mainly the frameworks of the PRF sets of the transmultiplexers whose coefficients are independent of member condition-II. For length-4, there are various possibilities of such frameworks. A few are given below.

Case-1: $\mathbf{f}_a(1) = \mathbf{f}_a(2)$ and $\mathbf{f}_a(3) = -\mathbf{f}_a(4)$

Then the corresponding $\mathbf{f}_a$ is $[l_1 \quad l_1 \quad l_2 \quad -l_2]$, for any two non-zero real numbers l_1 and l_2.

Case-2: $\mathbf{f}_a(1) = -\mathbf{f}_a(4)$ and $\mathbf{f}_a(2) = \mathbf{f}_a(3)$

Then, the corresponding $\mathbf{f}_a$ is $[l_1 \quad l_2 \quad l_2 \quad -l_1]$.

These filters are nearly symmetric. Other frameworks can also be easily obtained in the case of length-4 from the member condition-I and member condition-II equations.

For filters of length 6:

$\mathbf{f}_a(1)\mathbf{f}_a(5) + \mathbf{f}_a(2)\mathbf{f}_a(6) = 0$,

$\mathbf{f}_a(1)\mathbf{f}_a(3) + \mathbf{f}_a(2)\mathbf{f}_\mathbf{a}(4) + \mathbf{f}_a(3)\mathbf{f}_a(5) + \mathbf{f}_a(4)\mathbf{f}_a(6) = 0$

Here we have to satisfy two linear equations. For the first one we can choose

$\mathbf{f}_a(1) = -\mathbf{f}_a(6)$, $\mathbf{f}_a(2) = \mathbf{f}_a(5)$

substituting the values of $\mathbf{f}_a(5)$ and $\mathbf{f}_a(6)$, we get

$\mathbf{f}_a(1)[\mathbf{f}_a(3) - \mathbf{f}_a(4)] + \mathbf{f}_a(2)[\mathbf{f}_a(3) + \mathbf{f}_a(4)] = 0$

putting $\mathbf{f}_a(3) = \mathbf{f}_a(4)$ makes the first term zero.

To make the second term zero, the only possibility is to make $\mathbf{f}_a(2) = 0$. Thus, the framework becomes

$\mathbf{f}_a = [l_1 \quad 0 \quad l_2 \quad l_2 \quad 0 \quad -l_1]$. Since $\mathbf{f}_a(2) = 0$, we can write the framework as

$\mathbf{f}_a = [l_1 \quad 0 \quad l_2 \quad l_2 \quad 0 \quad l_1]$. This gives the symmetric filter.

For filters of length 8:

$\mathbf{f}_a(1)\mathbf{f}_a(7) + \mathbf{f}_a(2)\mathbf{f}_a(8) = 0$, $\mathbf{f}_a(1)\mathbf{f}_a(5) + \mathbf{f}_a(2)\mathbf{f}_a(6) + \mathbf{f}_a(3)\mathbf{f}_a(7) + \mathbf{f}_a(4)\mathbf{f}_a(8) = 0$

$\mathbf{f}_a(1)\mathbf{f}_a(3) + \mathbf{f}_a(2)\mathbf{f}_a(4) + \mathbf{f}_a(3)\mathbf{f}_a(5) + \mathbf{f}_a(4)\mathbf{f}_a(6) + \mathbf{f}_a(5)\mathbf{f}_a(7) + \mathbf{f}_a(6)\mathbf{f}_a(8) = 0$

Following are the required conditions for frameworks independent of member condition-II.

$\mathbf{f}_a(1) = -\mathbf{f}_a(8)$, $\mathbf{f}_a(2) = \mathbf{f}_a(7)$

Substituting the respective values in the last two equations, we get

$\mathbf{f}_a(1)[\mathbf{f}_a(5) - \mathbf{f}_a(4)] + \mathbf{f}_a(2)[\mathbf{f}_a(3) + \mathbf{f}_a(6)] = 0$

This suggests that $\mathbf{f}_a(4) = \mathbf{f}_a(5)$ and $\mathbf{f}_a(2) = 0$.

$\mathbf{f}_a(1)[\mathbf{f}_a(3) - \mathbf{f}_a(6)] + \mathbf{f}_a(4)[\mathbf{f}_a(3) + \mathbf{f}_a(6)] = 0$

That is, $\mathbf{f}_a(3) = \mathbf{f}_a(6)$ and $\mathbf{f}_a(4) = 0$.

Thus the framework of $\mathbf{f}_a = [l_1 \quad 0 \quad l_2 \quad 0 \quad 0 \quad l_2 \quad 0 \quad -l_1]$

With slight modifications, we have the following symmetric framework.

$\mathbf{f}_a = [l_1 \quad 0 \quad 0 \quad l_2 \quad l_2 \quad 0 \quad 0 \quad l_1]$

For filters of length 10:

$\mathbf{f}_a(1)\mathbf{f}_a(9) + \mathbf{f}_a(2)\mathbf{f}_a(10) = 0$,

$\mathbf{f}_a(1)\mathbf{f}_a(7) + \mathbf{f}_a(2)\mathbf{f}_a(8) + \mathbf{f}_a(3)\mathbf{f}_a(9) + \mathbf{f}_a(4)\mathbf{f}_a(10) = 0$

$\mathbf{f}_a(1)\mathbf{f}_a(5) + \mathbf{f}_a(2).\mathbf{f}_a(6) + \mathbf{f}_a(3)\mathbf{f}_a(7) + \mathbf{f}_a(4)\mathbf{f}_a(8) + \mathbf{f}_a(5)\mathbf{f}_a(9) + \mathbf{f}_a(6)\mathbf{f}_a(10) = 0$

$\mathbf{f}_a(1)\mathbf{f}_a(3) + \mathbf{f}_a(2)\mathbf{f}_a(4) + \mathbf{f}_a(3)\mathbf{f}_a(5) + \mathbf{f}_a(4)\mathbf{f}_a(6) + \mathbf{f}_a(5)\mathbf{f}_a(7) + \mathbf{f}_a(6)\mathbf{f}_a(8) +$
$\mathbf{f}_a(7)\mathbf{f}_a(9) + \mathbf{f}_a(8)\mathbf{f}_a(10) = 0$

$\mathbf{f}_a(1) = -\mathbf{f}_a(10)$, $\mathbf{f}_a(2) = \mathbf{f}_a(9)$

Putting the above two restrictions in the second equation, we get

$\mathbf{f}_a(1)[\mathbf{f}_a(7) - \mathbf{f}_a(4)] + \mathbf{f}_a(2)[\mathbf{f}_a(3) + \mathbf{f}_a(8)] = 0$

That is, $\mathbf{f}_a(4) = \mathbf{f}_a(7)$ and $\mathbf{f}_a(2) = 0$.

From the third equation, we get

$\mathbf{f}_a(1)[\mathbf{f}_a(5) - \mathbf{f}_a(6)] + \mathbf{f}_a(4)[\mathbf{f}_a(3) + \mathbf{f}_a(8)] = 0$

From this we get, $\mathbf{f}_a(5) = \mathbf{f}_a(6)$ and $\mathbf{f}_a(4) = 0$.

From the fourth equation, we get

$\mathbf{f}_a(1)\mathbf{f}_a(3) - \mathbf{f}_a(8)\mathbf{f}_a(1) = 0$

$\mathbf{f}_a(1)[\mathbf{f}_a(3) - \mathbf{f}_a(8)] + \mathbf{f}_a(5)[\mathbf{f}_a(3) + \mathbf{f}_a(8)] = 0$

That is, $\mathbf{f}_a(3) = \mathbf{f}_a(8)$ and $\mathbf{f}_a(5) = 0$.

Then, the framework becomes $\mathbf{f}_a = [l_1 \quad 0 \quad 0 \quad l_2 \quad 0 \quad 0 \quad l_2 \quad 0 \quad 0 \quad l_1]$.

The theory of 2*M*-channel transmultiplexer is simple. The PRF sets of transmultiplexer can easily be obtained from the member condition in various ways. The frameworks of filters given above are simple, one can choose any real values for the non-zero coefficients. Also we see that the symmetric filters can also be obtained. The transmultiplexer theory given here is a generalized one and is much simpler than the *M*-channel transmultiplexer. The main significance of this type of tranmultiplexer is its theoretical simplicity of perfect reconstruction. In the existing theory of *M*-channel transmultiplexer, perfect reconstruction is difficult to achieve because of three different types of distortion occur in the reconstructed signal, namely, phase distortion, amplitude distortion and cross-talk distortion. Such kinds of distortion are easily eliminated in our 2*M*-channel transmultiplexer scheme.

3.10 Decomposition Scheme of ISITRA in 2 Dimensions

The decomposition of 2-D signals is just the extension of 1-D signals. A 2-D signal can be thought as an array of 1-D signals of same lengths. For the decomposition of a 2-D signal, usually we first decompose the signal column-wise into two components using two decomposition filters. Each of the two decomposed components is again into two components using two decomposition filters. Usually the same decomposition filters are used for row-wise decomposition too. The order of row-wise or column-wise decomposition is immaterial. That is, one can do row-wise decomposition first followed by column-wise decomposition. Altogether, we get four decomposed components when a 2-D signal is decomposed.

Let $\mathbf{b}$ be a 2-D signal of size $M \times N$ and $\mathbf{f}_a$ and $\mathbf{f}_b$ be any two decomposition filters of length L. We will first decompose the signal column-wise into two decomposed

components. Let $\mathbf{A}$ and $\mathbf{B}$ be the respective decomposed components due to decomposition filters $\mathbf{f}_a$ and $\mathbf{f}_b$. Then we can write

$$\mathbf{A}(m,n) = \sum_{i=1}^{L} \mathbf{f}_a(i)\mathbf{b}(m,2n+1-i) \quad \text{and} \quad \mathbf{B}(m,n) = \sum_{i=1}^{L} \mathbf{f}_b(i)\mathbf{b}(m,2n+1-i)$$

where $m = 1,2,\cdots,M$ and $n = 1,2,\cdots,\lceil (N+L-1)/2 \rceil$.

Now, the decomposed components $\mathbf{A}$ and $\mathbf{B}$ have the same size $M \times (\lceil (N+L-1)/2 \rceil$. We have to decompose each of the decomposed components row-wise into two components using the same decomposition filters $\mathbf{f}_a$ and $\mathbf{f}_b$. Let $\mathbf{A}_a$ and $\mathbf{A}_b$ be the decomposed components due to row-wise decomposition of $\mathbf{A}$ using $\mathbf{f}_a$ and $\mathbf{f}_b$. And also let $\mathbf{B}_a$ and $\mathbf{B}_b$ be the decomposed components due to row-wise decomposition of $\mathbf{B}$ using the same set of decomposition filters. Then we can write

$$\mathbf{A}_a(m,n) = \sum_{i=1}^{L} \mathbf{f}_a(i)\mathbf{A}(2m+1-i,n) \quad \text{and} \quad \mathbf{A}_b(m,n) = \sum_{i=1}^{L} \mathbf{f}_b(i)\mathbf{A}(2m+1-i,n)$$

where $m = 1,2,\cdots,\lceil (M+L-1)/2 \rceil$ and the range of n remains the same as defined above. Substituting the values of $\mathbf{A}$, we have

$$\mathbf{A}_a(m,n) = \sum_{i=1}^{L} \mathbf{f}_a(i) \sum_{i_1=0}^{L} \mathbf{f}_a(i_1)\mathbf{b}(2m+1-i,2n+1-i_1)$$

$$\mathbf{A}_b(m,n) = \sum_{i=1}^{L} \mathbf{f}_b(i) \sum_{i_1=0}^{L} \mathbf{f}_a(i_1)\mathbf{b}(2m+1-i,2n+1-i_1)$$

Similarly, we can write $\mathbf{B}_a$ and $\mathbf{B}_b$ as

$$\mathbf{B}_a(m,n) = \sum_{i=1}^{L} \mathbf{f}_a(i) \sum_{i_1=0}^{L} \mathbf{f}_b(i_1)\mathbf{b}(2m+1-i,2n+1-i_1)$$

$$\mathbf{B}_b(m,n) = \sum_{i=1}^{L} \mathbf{f}_b(i)\mathbf{b}(2m+1-i,2n+1-i_1)$$

Thus, we get four decomposed components, namely, $\mathbf{A}_a, \mathbf{A}_b, \mathbf{B}_a$ and $\mathbf{B}_b$, each of size $\lceil (M+L-1)/2 \rceil \times \lceil (N+L-1)/2 \rceil$ when a 2-D signal $\mathbf{b}$ of size $M \times N$ is decomposed using the decomposition filters $\mathbf{f}_a$ and $\mathbf{f}_b$ of length L.

3.10.1 Reconstruction in 2 dimensions

The process of reconstruction here is also the reverse process of the decomposition. For reconstruction process, we employ reconstruction filters of the corresponding decomposition filters. Let $\mathbf{f}_{ar}$ and $\mathbf{f}_{br}$ be the reconstruction filters of the decomposition filters $\mathbf{f}_a$ and $\mathbf{f}_b$. The process of reconstruction involves two main stages. First, we reconstruct the row-wise decomposed components from their corresponding row-wise decomposed components. That is, first $\mathbf{A}$ is reconstructed from $\mathbf{A}_a$ and $\mathbf{A}_b$ using the reconstruction filters $\mathbf{f}_{ar}$ and $\mathbf{f}_{br}$. Similarly we reconstruct $\mathbf{B}$ from $\mathbf{B}_a$ and $\mathbf{B}_b$ using the same reconstruction filters. Then the original signal $\mathbf{X}$ is reconstructed from $\mathbf{A}$ and $\mathbf{B}$ using $\mathbf{f}_{ar}$ and $\mathbf{f}_{br}$. Let

$$D_1 = \sum_{i=1}^{L} \mathbf{f}_a(2i)\mathbf{f}_{ar}(2i) + \mathbf{f}_b(2i)\mathbf{f}_{br}(2i) \neq 0$$

$$D_2 = \sum_{i=1}^{L} \mathbf{f}_a(2i-1)\mathbf{f}_{ar}(2i-1) + \mathbf{f}_b(2i-1)\mathbf{f}_{br}(2i-1) \neq 0$$

Then the reconstruction of $\mathbf{A}$ from $\mathbf{A}_a$ and $\mathbf{A}_b$ is done as follows.

$$\mathbf{A}(2m-1,n) = \frac{\sum_{i=1}^{L/2} \mathbf{f}_{ar}(L+2-2i)\mathbf{A}_a(t_m - i, n) + \mathbf{f}_{br}(L+2-2i)\mathbf{A_b}(t_m - i, n)}{D_1}$$

$$\mathbf{A}(2m,n) = \frac{\sum_{i=1}^{L/2} \mathbf{f}_{ar}(L+1-2i)\mathbf{A}_a(t_m - i, n) + \mathbf{f}_{br}(L+1-2i)\mathbf{A}_b(t_m - i, n)}{D_2}$$

putting $t_m = m + L/2$.

Similarly, we reconstruct $\mathbf{B}$ from $\mathbf{B}_a$ and $\mathbf{B}_b$ as

$$\mathbf{B}(2m-1,n) = \frac{\sum_{i=1}^{L/2} \mathbf{f}_{ar}(L+2-2i)\mathbf{B}_a(t_m - i, n) + \mathbf{f}_{br}(L+2-2i)\mathbf{B}_b(t_m - i, n)}{D_1}$$

$$\mathbf{B}(2m,n) = \frac{\sum_{i=1}^{L/2} \mathbf{f}_{ar}(L+1-2i)\mathbf{B}_a(t_m - i, n) + \mathbf{f}_{br}(L+1-2i)\mathbf{B}_b(t_m - i, n)}{D_2}$$

Then, we reconstruct $\mathbf{b}$ from $\mathbf{A}$ and $\mathbf{B}$ using $\mathbf{f}_{ar}$ and $\mathbf{f}_{br}$ as

$$\mathbf{b}(m,2n-1)=\frac{\sum_{i=1}^{L/2}\mathbf{f}_{ar}(t_2-2i)\mathbf{A}(m,t_n-i)+\mathbf{f}_{br}(t_2-2i)\mathbf{B}(m,t_n-i)}{D_1}$$

$$\mathbf{b}(m,2n)=\frac{\sum_{i=1}^{L/2}\mathbf{f}_{ar}(t_1-2i)\mathbf{A}(m,t_n-i)+\mathbf{f}_{br}(t_1-2i)\mathbf{B}(m,t_n-i)}{D_2}$$

where $t_n=n+L/2$.

The above two equations of $\mathbf{A}$ can be combined together and written as

$$\mathbf{A}(m,n)=\frac{\sum_{i=1}^{L/2}\mathbf{f}_{ar}(s_m-2i)\mathbf{A}_a(t_m-i,n)+\mathbf{f}_{br}(s_m-2i)\mathbf{A}_b(t_m-i,n)}{D_{2-m\%2}}$$

putting $s_m=L+(m-1)\%2$.

Similarly, we can write $\mathbf{B}$ and $\mathbf{b}$ as

$$\mathbf{B}(m,n)=\frac{\sum_{i=1}^{L/2}\mathbf{f}_{ar}(s_m-2i)\mathbf{B}_a(t_m-i,n)+\mathbf{f}_{br}(s_m-2i)\mathbf{B}_b(t_m-i,n)}{D_{2-m\%2}}$$

$$\mathbf{b}(m,n)=\frac{\sum_{i=1}^{L/2}\mathbf{f}_{ar}(s_n-2i)\mathbf{A}(m,t_n-i)+\mathbf{f}_{br}(s_n-2i)\mathbf{B}(m,t_n-i)}{D_{2-n\%2}}$$

Expressing $\mathbf{b}$ in terms of $\mathbf{A}_a,\mathbf{A}_b,\mathbf{B}_a$ and $\mathbf{B}_b$, we have

$$\mathbf{b}(m,n)=\frac{1}{D_{2-m\%2}D_{2-n\%2}}\sum_{i=1}^{L/2}\mathbf{f}_{ar}(s_n-2i)\{\sum_{i_1=1}^{L/2}\mathbf{f}_{ar}(s_m-2i_1)\mathbf{A}_a(t_m-i_1,t_n-i)+$$

$$\mathbf{f}_{br}(s_m-2i_1)\mathbf{A}_b(t_m-i_1,t_n-i)\}+\mathbf{f}_{br}(s_n-2i)\{\sum_{i_1=1}^{L/2}\mathbf{f}_{ar}(s_m-2i_1)\mathbf{B}_a(t_m-i_1,t_n-i)$$

$$+\mathbf{f}_{br}(s_m-2i_1)\mathbf{B}_b(t_m-i_1,t_n-i)\}$$

Using this equation, we can directly reconstruct the original signal from its four decomposed components. With a proper choice of filter coefficients, we can make the denominator terms equal, i.e., $D_1=D_2=D$, say. Then the above reconstruction equation becomes much simpler. Moreover, we can eliminate the denominator terms by making them equal to unity.

3.11 Multilevel Decomposition Scheme of ISITRA in 2 Dimensions

The single level decomposition scheme described in the previous section can be further extended to a multilevel decomposition scheme. The multilevel decomposition scheme is further divided into two types, namely, multilevel half decomposition and multilevel full decomposition schemes as in the case of 1-D signal decomposition.

3.11.1 Half decomposition

For ease of reference, we put level index in each of the decomposed component. The level-1 decomposed components will be denoted by $\mathbf{A}_{1a}, \mathbf{A}_{1b}, \mathbf{B}_{1a}$ and $\mathbf{B}_{1b}$ and the level-*k* decomposed components will be denoted by $\mathbf{A}_{ka}, \mathbf{A}_{kb}, \mathbf{B}_{ka}$ and $\mathbf{B}_{kb}$. Figure 3.9, shows the decomposed components of level-2 half decomposition of a 2D signal. In the figure a 2D signal first decomposed into four components $\mathbf{A}_{1a}, \mathbf{A}_{1b}, \mathbf{B}_{1a}$ and $\mathbf{B}_{1b}$ at level-1. In the second level only the left–top decomposed component $\mathbf{A}_{1a}$ is decomposed into four components.

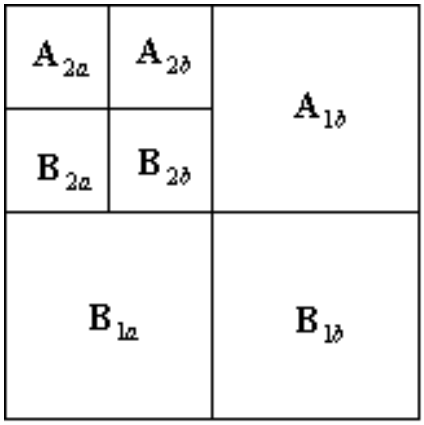


Fig.3.9 Level-2 half decomposition

The recursive decomposition scheme is given below

$$\mathbf{A}_{ka}(m,n) = \sum_{i=1}^{L} \mathbf{f}_a(i) \sum_{i_1=0}^{L} \mathbf{f}_a(i_1) \mathbf{A}_{(k-1)a}(2m+1-i, 2n+1-i_1)$$

$$\mathbf{A}_{kb}(m,n) = \sum_{i=1}^{L} \mathbf{f}_b(i) \sum_{i_1=0}^{L} \mathbf{f}_a(i_1) \mathbf{A}_{(k-1)a}(2m+1-i, 2n+1-i_1)$$

$$\mathbf{B}_{ka}(m,n) = \sum_{i=1}^{L} \mathbf{f}_a(i) \sum_{i_1=0}^{L} \mathbf{f}_b(i_1) \mathbf{A}_{(k-1)a}(2m+1-i, 2n+1-i_1)$$

$$\mathbf{B}_{kb}(m,n) = \sum_{i=1}^{L} \mathbf{f}_b(i) \sum_{i_1=0}^{L} \mathbf{f}_b(i_1) \mathbf{A}_{(k-1)a}(2m+1-i, 2n+1-i_1)$$

where $m = 1,2,\cdots, lem_k$ and $n = 1,2,\cdots, len_k$

The values of lem_k and len_k are recursively obtained as follows

$lem_k = ceil((lem_{k-1})/2)$ and $len_k = ceil((len_{k-1})/2)$

with $lem_1 = \lceil (M + L - 1)/2 \rceil$ and $len_1 = \lceil (N + L - 1)/2 \rceil$.

Thus, the size of each decomposed components at level-*k* is $lem_k \times len_k$.

Note here that a signal is said to be decomposable up to level-*k*, if $1 \le \min(lem_k, len_k) < L$.

3.11.2 Half reconstruction

The multilevel half reconstruction scheme can be obtained from the single level reconstruction scheme. As we have used the same decomposition filters at each level of decomposition, we will also use the same reconstruction filters $\mathbf{f}_{ar}$ and $\mathbf{f}_{br}$ at each level of reconstruction. The reconstruction is done from bottom upwards. That is, reconstruction starts from the last four decomposed components. Without going into derivation detail, we can directly write the recursive reconstruction equation as

$$\mathbf{A}_{(k-1)a}(m,n) = \frac{1}{D_{2-m\%2} D_{2-n\%2}} \sum_{i=1}^{L/2} \mathbf{f}_{ar}(s_n - 2i) \{ \sum_{i_1=1}^{L/2} \mathbf{f}_{ar}(s_m - 2i_1) \mathbf{A}_{ka}(t_m - i_1, t_n - i) +$$

$$\mathbf{f}_{br}(s_m - 2i_1) \mathbf{A}_{kb}(t_m - i_1, t_n - i) \} + \mathbf{f}_{br}(s_n - 2i) \{ \sum_{i_1=1}^{L/2} \mathbf{f}_{ar}(s_m - 2i_1) \mathbf{B}_{ka}(t_m - i_1, t_n - i)$$

$$+ \mathbf{f}_{br}(s_m - 2i_1) \mathbf{B}_{kb}(t_m - i_1, t_n - i) \}$$

As only $\mathbf{A}_{ra}$ component is decomposed at level-*r*, we reconstruct only the one component at each reconstruction level.

3.11.3 Full decomposition

In this case, an image is decomposed into four components. Each of the decomposed components is again decomposed into four components at each successive level of decomposition. As each of the decomposed components is decomposed into four components, it is not feasible to refer the decomposed components as $\mathbf{A}$-components or $\mathbf{B}$-components. We will denote the decomposition filters by $\mathbf{F}_1$ and $\mathbf{F}_2$. Each of the decomposed components is referenced by using two suffixes- first suffix denotes the level of decomposition and the second suffix to denote component number. In short, by $\mathbf{A}_{i,j}$, we will denote the j-th decomposed component at i-th level. The level-2 full decomposition of a 2D signal is shown in Figure 3.10. Here, first a 2D signal is first decomposed into four components at the first level and each of the decomposed components is successively decomposed into four components at the 2nd level.

$\mathbf{A}_{2,1}$	$\mathbf{A}_{2,2}$	$\mathbf{A}_{2,5}$	$\mathbf{A}_{2,6}$
$\mathbf{A}_{2,3}$	$\mathbf{A}_{2,4}$	$\mathbf{A}_{2,7}$	$\mathbf{A}_{2,8}$
$\mathbf{A}_{2,9}$	$\mathbf{A}_{2,10}$	$\mathbf{A}_{2,13}$	$\mathbf{A}_{2,14}$
$\mathbf{A}_{2,11}$	$\mathbf{A}_{2,12}$	$\mathbf{A}_{2,15}$	$\mathbf{A}_{2,16}$

Figure 3.10 Level-2 full decomposition of a 2D signal.

Assuming, for every decomposition process we use $\mathbf{F}_1$ and $\mathbf{F}_2$ filters, we can get the decomposed components at level-1as follows.

$$\mathbf{A}_{1,1}(m,n) = \sum_{i=1}^{L} \mathbf{F}_1(i) \sum_{i_1=0}^{L} \mathbf{F}_1(i_1)\mathbf{b}(2m+1-i, 2n+1-i_1)$$

$$\mathbf{A}_{1,2}(m,n) = \sum_{i=1}^{L} \mathbf{F}_2(i) \sum_{i_1=0}^{L} \mathbf{F}_1(i_1)\mathbf{b}(2m+1-i, 2n+1-i_1)$$

$$\mathbf{A}_{1,3}(m,n) = \sum_{i=1}^{L} \mathbf{F}_1(i) \sum_{i_1=0}^{L} \mathbf{F}_2(i_1)\mathbf{b}(2m+1-i, 2n+1-i_1)$$

$$\mathbf{A}_{1,4}(m,n) = \sum_{i=1}^{L} \mathbf{F}_2(i) \sum_{i_1=0}^{L} \mathbf{F}_2(i_1)\mathbf{b}(2m+1-i, 2n+1-i_1)$$

In the second level, we will get 16 decomposed components as each decomposed component is to be decomposed into four components as shown in Fig.3.10.

Thus, writing the decomposed components at each level of decomposition becomes difficult for higher level of decomposition. In order to avoid this disadvantage, we use a different way of writing decomposed components at a particular level by introducing suffixes to filters. Using this technique, the above level-1 decomposed components can be written in short as

$$\mathbf{A}_{1,j}(m,n) = \sum_{i=1}^{L} \mathbf{F}_{(2-j\%2)}(i) \sum_{i_1=1}^{L} \mathbf{F}_{2-(\lceil j\%2 \rceil \%2)}(i_1)\mathbf{b}(2m+1-i, 2n+1-i_1)$$

where $j = 1,2,3,4.$

The suffix associated with filters on the right hand side allows one to choose appropriate filters to find a particular decomposed component at a particular level. Generalizing this way, we can recursively get the decomposed components at level-*k* from the following relation.

$$\mathbf{A}_{k,j}(m,n) = \sum_{i=1}^{L} \mathbf{F}_{(2-j\%2)}(i) \sum_{i_1=1}^{L} \mathbf{F}_{2-(\lceil j\%2 \rceil \%2)}(i_1)\mathbf{A}_{k-1,\lceil j\%4 \rceil}(2m+1-i, 2n+1-i_1)$$

where $j = 1,2,\cdots,4^k$ for any $1 \le \min(lem_k, len_k) < L$.

3.11.4 Full reconstruction

The reconstruction scheme is also almost the same as in multilevel half reconstruction. The difference here is that instead of reconstructing only one component at every reconstruction level, we reconstruct 4^{r-1} components from the 4^r decomposed components. Let $\mathbf{F}_{1r}$ and $\mathbf{F}_{2r}$ be the corresponding reconstruction filters of the decomposition filters $\mathbf{F}_1$ and $\mathbf{F}_2$. Then we can write the recursive reconstruction equation as

$$\mathbf{A}_{k-1,j}(m,n) = \frac{1}{D_{2-m\%2}D_{2-n\%2}}[\sum_{i=1}^{L/2}\mathbf{F}_{1r}(s_n-2i)\{\sum_{i_1=1}^{L/2}\mathbf{F}_{1r}(s_m-2i_1)\mathbf{A}_{k,4j-3}(t_m-i_1,t_n-i)+$$

$$\mathbf{F}_{2r}(s_m-2i_1)\mathbf{A}_{k,4j-2}(t_m-i_1,t_n-i)\}+\mathbf{F}_{2r}(s_n-2i)\{\sum_{i_1=1}^{L/2}\mathbf{F}_{1r}(s_m-2i_1)\mathbf{A}_{k,4j-1}(t_m-i_1,t_n-i)$$

$$+\mathbf{F}_{2r}(s_m-2i_1)\mathbf{A}_{k,4j}(t_m-i_1,t_n-i)\}]$$

Note here that the reconstruction level-0, which corresponds to $k=1$ gives the original 2-D signal. Thus, we can reconstruct the original signal from its decomposed components using full decomposition scheme. We may have several decomposition schemes in the case of 2-D signals as in 1-D signals corresponding to the different decomposition and the reconstruction equations of different PRF sets. The applicability of a particular decomposition or the reconstruction scheme depends on the types of application one has in hand. The general decomposition scheme is the same for all cases.

3.12 Discussion and Remarks

In this chapter, we develop a new generalized subband decomposition and reconstruction scheme entitled ISITRA based on Bino's model of multiplication. We compare the computational complexities of 2-channel filter bank (in direct and polyphase forms) with ISITRA. It is found that ISITRA is significantly better than the direct 2-channel filter bank (2-FB) and is marginally better than the polyphase form of the filter bank. The perfect reconstruction theory of ISITRA is much simpler than the 2-channel filter bank. Perfect reconstruction theory provides us the relations between filters in a subband scheme to ensure perfect reconstruction. Discrete wavelet transforms is based on the perfect reconstruction theory of filter bank and hence perfect reconstruction is possible in DWT while the same is not possible in the continuous wavelet transforms. Perfect reconstruction theory of 2-channel filter bank being derived in *z*-domain is difficult to obtain PRF sets in DWT or 2-FB. Perfect reconstruction theory of ISITRA removes the difficulty of finding PRF sets. It suggests some conditions known as member condition to be satisfied by decomposition and reconstruction filters. Any set of decomposition and reconstruction filters that satisfies member condition can be used for perfect reconstruction. Also, it is very easy to find PRF sets from the member condition. Member condition can be easily satisfied in various different ways bypassing the wavelet filter conditions (1) or (2) or (3). This indicates that

neither of the wavelet filter conditions is a prerequisite for perfect reconstruction in the present context of DWT.

There are two different ways in which we can find PRF sets from the member condition in ISITRA. A PRF set can be obtained from an array or from 2 different arrays. PRF sets obtained from a single array is known as PRF sets of ISITRA-1. Similarly, PRF sets obtained from 2 distinct arrays are known as PRF sets of ISITRA-2. All the PRF sets of orthogonal wavelet filters belong to PRF sets of ISITRA-1. And all the PRF sets of bi-orthogonal wavelets belong to PRF sets of ISITRA-2. In ISITRA, once we know a PRF set, infinitely many PRF sets can be easily obtained from it. Frameworks of PRF sets of orthogonal and bi-orthogonal wavelet are also provided so that one can easily find PRF sets for different applications.

After illustrating how to find PRF sets, we then extend single level decomposition scheme to multilevel decomposition scheme that leads to multi-resolution signal representation. There are two types of multilevel decomposition scheme; half decomposition scheme and full decomposition scheme. Half decomposition scheme is similar to dyadic filter bank or DWT and full decomposition scheme is similar to wavelet packet decomposition scheme or octave band filter bank. The decomposition and reconstruction equations are provided for both multilevel decomposition schemes can be easily implemented. To better understand the multi-resolution concept in ISITRA, we explain multi-resolution in terms of some filters known as effective filters. Effective filters are useful to directly get a decomposed component of a signal at any particular level without having to pass through successive decomposition stages. This will save significant amount of time in multilevel signal decomposition.

From the 2-channel decomposition scheme, we have proposed multi-channel decomposition scheme. In our multi-channel scheme, a signal is first decomposed into several segments using a scheme known as channel decomposition scheme and then each segment is decomposed into two components using 2-channel decomposition scheme of ISITRA. Hence, we do not require a different perfect reconstruction theory unlike in the existing multi-channel filter bank scheme where a separate perfect reconstruction theory is required. We can use PRF sets of 2-channel ISITRA in multi-channel case also. From the multi-

channel filter bank scheme of ISITRA, we have proposed multi-channel transmultiplexer scheme. Our transmultiplexer scheme is simple and perfect reconstruction, which is difficult to achieve in the existing multi-channel transmultiplexer, can easily be achieved. The possibility of getting perfect reconstruction in our proposed multi-channel transmultiplexer is based on the fact that in our case, only perfect reconstruction theory of 2-channel transmultiplexer is required for perfect reconstruction. We have described the perfect reconstruction theory of 2-channel transmultiplexer and the ways of finding PRF sets of various lengths. Lastly, we describe decomposition and reconstruction scheme of 2D signals.

3.13 Conclusions

ISITRA is a generalized subband decomposition and reconstruction scheme where the perfect reconstruction is possible. It is simple and easy to implement being given in array context. The perfect reconstruction theory is simple and different types of perfect reconstruction filters can easily be obtained. Hence a signal can be decomposed into all possible combinations of detail and approximation components without disturbing perfect reconstruction. Multi-resolution signal representation is achieved in ISITRA through multilevel decomposition scheme, which can be also be achieved directly with effective filters. Moreover, the multi-channel filter bank and multi-channel transmultiplexer of ISITRA are simple and perfect reconstruction is possible in both cases. ISITRA, being a generalized case of 2-channel filter bank, may be used in place of discrete wavelet transform in various applications.

Chapter 4

YKSK TRANSFORM FOR SIGNAL DECOMPOSITION AND RECONSTRUCTION

Abstract: *We propose here a generalized transform entitled YKSK transform. The motivation of developing such a transform is to give a generalized transform that can be used for various signal and image processing applications such as compression, analysis, encryption, etc. The decomposition scheme of such a transform consists of two parts, information carriers and the modifiers. Information carriers store necessary information required for perfect reconstruction of the signal from its decomposed components. Modifiers are used to modify the decomposed component in various ways through arbitrary choice of decomposition filters. By adjusting the modifiers, one can use it as different types of transforms using subband decomposition scheme. The similar result of analysis obtained from using the wavelet or ISITRA can also be made obtainable from this transform. Unlike the case in wavelet and ISITRA, which uses decomposition and reconstruction filters of even length, it uses only the decomposition filters and the length of the filters may be even or odd. Also the filter coefficients can be chosen quite arbitrarily without concerning about the types of the filters. Moreover, we can easily get integer version of the YKSK transform easily.*

4.1 Introduction

In the existing subband transforms such as wavelet and filter bank used for signal decomposition and reconstruction, it is difficult to obtain the filter coefficients. This difficulty has been greatly reduced with the emergence of ISITRA (Chapter 3). But there are still some restrictions in the choice of filter coefficients. All the filter coefficients are to be found from their corresponding member condition equations to ensure perfect reconstruction of the original signal from its decomposed components. In the case of wavelet and filter

bank, a signal is decomposed into two components, one corresponds to high pass or detail and the other corresponds to low pass or approximation. But in the case of ISITRA, it is quite different where a signal can be decomposed in all possible ways, because of the possibility of choosing different types of filters other than the QMF types. Even though we can decompose a signal in various possible ways, in ISITRA too all filters need to be of even length and the coefficients of the filters cannot be chosen arbitrarily for filters of length greater than 2. In order to remove the restrictions in the choice of filter coefficients and to make the length of each decomposed component independent of the length of the decomposition filters, we propose YKSK transform in this chapter.

It is also a kind of subband decomposition scheme as that of the filter bank or ISITRA. However, the decomposition and reconstruction schemes here are quite different from them. Here the decomposition and reconstruction processes involve two parts, namely, the information carriers and the modifiers. The information carriers retain the necessary information for reconstruction of the signal from its decomposed components. The modifiers have great influence to make efficacious change on the characteristics of the decomposed components. Modifiers may be constants or functions of the decomposition filters and components of the input signal or its decomposed components. By appropriately selecting the modifiers one can decompose a signal into various possible ways as in ISITRA. That is, a signal can be decomposed into two approximations or into two details or into one approximation and a detail by using appropriate modifiers finding of which is possible through quite an arbitrary choice of the decomposition filters. Also, the length of a decomposed component is independent of the length of the decomposition filters and is always the ceiling of half of the length of the input signal. This enables us to save a significant amount of storage and processing time when the lengths of the filters are large. Moreover, unlike in subband transforms such as filter bank, wavelet or ISITRA where separate reconstruction filters are required for the reconstruction propose, here no reconstruction filters are explicitly required. The same decomposition filters are used as the reconstruction filters.

The decomposition scheme can be continued up to any desired level, but the maximum decomposable level is fixed depending on the length of the signal. The multilevel decomposition scheme is discussed later.

4.1.1 Types of YKSK transform

YKSK transform consists of two parts, the information carriers and the modifiers. Depending on the number of elements of the signal used for information carrier, YKSK transform is divided into two classes, namely, YKSK-1 and YKSK-2 transforms. In YKSK-1, the information carrier part consists of a single term of the signal. In YKSK-2, information carrier part consists of two terms of the signal. In both classes of YKSK transform, the characteristics of a decomposed component are dependent on the form of the corresponding modifier. As modifiers can be chosen arbitrarily, we can make the number of terms of the input signal involved in the computation of a term of a decomposed component (i) fixed at a certain value or (ii) increasing till the last term of the decomposed component is obtained. Depending on whether this number for a modifier is fixed or increasing, each class of YKSK transform is divided into three main types, 1) Fixed Length Type, 2) Semi-Infinite Length Type and 3) Infinite Length Type. In the fixed length type, the numbers of terms of the signal involved in the computation of both the decomposed components are fixed. In the semi-infinite case, the number of terms of the signal involved in the computation of a decomposed component is fixed while the number of terms of a signal involved in the computation of the other decomposed component is increasing. In the infinite case, the numbers of terms of the signal involved in the computation of both the decomposed components are increasing. YKSK transforms can easily be generalized to integer transforms.

In short, the main features of YKSK transform are :

1. It provides a subband decomposition scheme where each decomposed component has two parts, namely, an information carrier part and a modifier part.
2. Information carriers store necessary information required for perfect reconstruction of the signal from its decomposed components. Modifiers are used to modify the decomposed component in various ways through arbitrary choice of decomposition filters.
3. Perfect reconstruction of the signal from the decomposed components is possible and simple. The restrictions on the choice of filter coefficients for perfect reconstructibility are soft. No separate reconstruction filters are necessary for the purpose.

4. In YKSK transform, the decomposed components can be either both approximations or both details or one approximation and one detail.
5. The length of the decomposed components does not depend on the length of the filters. It is always the ceiling of the half of the length of the signal.
6. One can get integer transforms from YKSK transform. Existing integer transforms are special cases of YKSK transform.

4.2 YKSK-1 Transform

Here only a single term in each decomposition equation constitutes the information carrier part and the remaining terms constitute the modifier. Usually, the first term of a decomposition equation constitutes the information carrier. The first term of one decomposed component is made of an odd component of the signal while the first term of the other decomposed component is made of an even component of the signal. The reconstruction is thus simple, just evaluating back odd and even components of the signal from the decomposed components. And hence it requires no extra condition for perfect reconstruction and no separate reconstruction filters. The same decomposition filters are used for reconstruction process also. Only requirement for perfect reconstruction is determinability of the modifiers both in the decomposition and reconstruction process. The length of each decomposed component is equal to the ceiling of the half of the length of the input signal. Another interesting feature of this transform is that we can easily make the values of decomposed components integers even if we use arbitrarily chosen floating point numbers as filters for decomposition. Most of the integer wavelet transforms such as S-transform[55], interpolating transforms [24], lifting scheme [139] can be explained as special cases of YKSK-1 transform.

4.2.1 Decomposition scheme of YKSK-1 transform

In this decomposition scheme, a signal is decomposed into two components using arbitrarily chosen decomposition filters. Like in ISITRA, a signal can be decomposed into two approximations, two details or one approximation and one detail and hence we denote the decomposed components simply as $\mathbf{A}$ and $\mathbf{B}$, and the associated decomposition filters as $\mathbf{f}_a$ and $\mathbf{f}_b$. The decomposition filters are some arbitrarily chosen non-zero and non-empty arrays of numbers. The coefficients of these decomposition filters may be real or imaginary

numbers, but we consider here only real numbers. We will consider only the single-level decomposition of 1-dimensional signal. The decomposition scheme for higher-level or higher dimensional cases can easily be obtained from it as in the case of ISITRA. All arrays are all unity offset arrays, that is, the lowest index of an array is 1.

Let $\mathbf{b}$ be a 1-dimensional signal of even length m and $\mathbf{A}$ and $\mathbf{B}$ be its two decomposed components. Then we can write the decomposition scheme of YKSK-1 transform as

$$\left.\begin{aligned}\mathbf{A}(n) &= \mathbf{f}_a(1)\mathbf{b}(2n-1) + M_a(n)\\ \mathbf{B}(n) &= \mathbf{f}_b(1)\mathbf{b}(2n) + M_b(n)\end{aligned}\right\} \quad \cdots\cdots\cdots(1)$$

where $n = 1,2,\cdots,m/2$ and M_a and M_b are modifiers. Modifiers may be constants or functions of decomposition filters $\mathbf{f}_a$ and $\mathbf{f}_b$ or components of the input signal or decomposed components. Modifiers have great influence on the characteristics of a decomposed component. By choosing the modifiers in different ways we can decompose a signal into two approximations or two details or one approximation and one detail. The modifiers may or may not be a function of n. The role of the decomposition filters is mainly to obtain suitable modifiers which would transform the signal in certain desired ways. The decomposition filters may be of any length, and their coefficients can be chosen quite arbitrarily. But the first element of each decomposition filter should be non-zero in order not to destroy the information carriers and to avoid division by zero.

4.2.2 Reconstruction scheme of YKSK-1 transform

YKSK transform is a perfect reconstruction transform even if the filters are chosen arbitrarily because we choose only determinable modifiers. The reconstruction scheme is very simple and involves no complex mathematical operations. It is just finding the even and odd components of the signal from the decomposed components. That is, finding $\mathbf{b}(2n-1)$ and $\mathbf{b}(2n)$ from the decomposition equations, which is always possible if we have a means to find the respective modifiers. This is how we can get perfect reconstruction from its decomposed components in YKSK-1 transform. We can get the original signal from its decomposed components from the following relations.

$$\left.\begin{aligned}\mathbf{b}(2n) &= (\mathbf{B}(n) - M_b(n))/\mathbf{f}_b(1)\\ \mathbf{b}(2n-1) &= (\mathbf{A}(n) - M_a(n))/\mathbf{f}_a(1)\end{aligned}\right\} \quad \cdots\cdots\cdots(2)$$

Thus, we can easily get the original signal from the decomposed components using equation (2). The only restriction is that $\mathbf{f}_a(1) \neq 0$ and $\mathbf{f}_b(1) \neq 0$. From both the decomposition and reconstruction equations, we know that modifiers are involved. It is already mentioned that the modifiers should be determinable for both the decomposition and reconstruction processes. Modifiers are said to be determinable when the modifiers have constant values or when the modifiers can be generated during the decomposition and reconstruction processes. For modifiers to be determinable, at least one of the modifiers should be free from information carrier terms.

4.3 Types of YKSK-1 Transform

Depending on the number of terms of the signal involved in the computation of a term of the decomposed components, YKSK-1 transform has three types, namely fixed length type, semi-infinite length type and infinite length type. They are described below. Let $N_a(n)$ be the number of terms of the signal $\mathbf{b}$ involved in the computation of a decomposed component $\mathbf{A}(n)$ and $N_b(n)$ be the number of term of the signal $\mathbf{b}$ involved in the computation of a decomposed component $\mathbf{B}(n)$.

4.3.1 Fixed length type YKSK-1 transform

Here the number of terms of a signal involved in the computation of a decomposed component is fixed at a value less than the length of the signal. That is, $N_a(n)$ and $N_b(n)$ are constants, not dependent on *n*. The fixed length type YKSK-1 transform is similar to sliding window transform like ISITRA or filter-bank. That is, any term of a decomposed component is computed from the elements of the signal covered by a sliding window, which is here the length of the transform. It differs from ISITRA or filter bank in that the sliding window here does not go beyond the signal's end point. It stops as soon as the forwarding end of the window reaches the end point of the signal. Hence, the length of a decomposed component here is dependent on the length of the input signal alone and not on the length of the sliding window. Length of the sliding window here does not mean the length of a filter as in ISITRA or filter bank, but the number of terms of the signal, which we want to involve in the computation of a term of a decomposed component. This length may or may not be equal to the length of the filters used in the decomposition.

Suppose, $N_a(n) = N_a$ and $N_b(n) = N_b$. The following type of modifiers gives rise to fixed length YKSK-1 transform.

$$\left.\begin{aligned} M_a(n) &= \sum_{i=2}^{La} \mathbf{f}_a(i)\mathbf{b}(2n + c_1 \pm i) \\ M_b(n) &= \sum_{i=2}^{Lb} \mathbf{f}_b(i)\mathbf{A}(n + c_2 \pm i) \end{aligned}\right\} \quad \cdots\cdots\cdots(3)$$

where c_1 and c_2 are some integer constants (zero, positive or negative) for adjusting the indices of the arrays, L_a is the number of coefficients in $\mathbf{f}_a$ and L_b is the number of coefficients in $\mathbf{f}_b$. The number terms of $\mathbf{b}$ involved in the computation of $\mathbf{A}(n)$ here is L_a. And since $M_b(n)$ is a function of $\mathbf{A}(n)$, the number of terms of $\mathbf{b}$ involved in the computation of $\mathbf{B}(n)$ is fixed. Hence a set of decomposition equations with modifiers of this type is known as fixed length type. Also, since the modifier $M_b(n)$ is a function of the decomposed components, reconstruction starts from the associated information carrier, here $\mathbf{b}(2n)$, knowledge of which is required for the reconstruction of the other information carrier $\mathbf{b}(2n-1)$. Depending on the values of N_a and N_b, there are five different types of fixed length types, namely, equal and even fixed length type, equal and odd fixed length type, unequal and even fixed length type, unequal and odd fixed length type and mixed fixed length type.

Equal and even fixed length type: Here N_a and N_b are even and equal. For example, consider the following modifiers

$$\left.\begin{aligned} M_a(n) &= \mathbf{f}_a(2)\mathbf{b}(2n) + \sum_{i=3}^{La} \mathbf{f}_a(i)\mathbf{b}(2n + 1 - i) \\ M_b(n) &= \sum_{i=2}^{Lb} \mathbf{f}_b(i)\mathbf{A}(n + 2 - i) \end{aligned}\right\} \quad \cdots\cdots\cdots(4)$$

When $L_b = 2$, the number of terms of $\mathbf{b}$ involved in the computation of $\mathbf{B}(n)$ will be equal to L_a. And if L_a is even, we get equal and even fixed length type. For example, if $L_a = 4$ and $L_b = 2$, our decomposition equations become

$$\mathbf{A}(n) = \mathbf{f}_a(1)\mathbf{b}(2n-1) + \mathbf{f}_a(2)\mathbf{b}(2n) + \mathbf{f}_a(3)\mathbf{b}(2n-2) + \mathbf{f}_a(4)\mathbf{b}(2n-3)$$

$$\mathbf{B}(n) = \mathbf{f}_b(1)\mathbf{b}(2n) + \mathbf{f}_b(2)\mathbf{A}(n)$$
$$= (\mathbf{f}_b(1) + \mathbf{f}_a(2)\mathbf{f}_b(2))\mathbf{b}(2n) + \mathbf{f}_b(2)(\mathbf{f}_a(1)\mathbf{b}(2n-1) + \mathbf{f}_a(3)\mathbf{b}(2n-2) + \mathbf{f}_a(4)\mathbf{b}(2n-3))$$

Thus, computation of $\mathbf{A}(n)$ or $\mathbf{B}(n)$ requires at most 4 terms of $\mathbf{b}$. Note here that the modifier $M_a(n)$ contains one information carrier and hence we cannot start reconstruction from $\mathbf{b}(2n-1)$. We start reconstruction from $\mathbf{b}(2n)$, as its corresponding modifier $M_b(n)$ is a function of decomposed components only and is free of information carriers. Also, the reconstruction should start from the beginning or the left end of the signal as the modifier $M_a(n)$ has terms of $\mathbf{b}$ whose indices are less than $2n-1$.

Unequal and even fixed length type: Here N_a and N_b are even but unequal. For example, when $L_a = 4$, $L_b = 3$ in equation (4), the number of terms of $\mathbf{b}$ involved in the computation of $\mathbf{A}(n)$ is 4 while the same in the computation of $\mathbf{B}(n)$ is 6.

Equal and odd fixed length type: Here N_a and N_b are odd and equal. For example, if L_a is odd and $L_b = 2$ for the modifiers in equation (4), we get this fixed length type. If L_a =3, the decomposition equations become

$$\mathbf{A}(n) = \mathbf{f}_a(1)\mathbf{b}(2n-1) + \mathbf{f}_a(2)\mathbf{b}(2n) + \mathbf{f}_a(3)\mathbf{b}(2n-2)$$

$$\mathbf{B}(n) = \mathbf{f}_b(1)\mathbf{b}(2n) + \mathbf{f}_b(2)\mathbf{A}(n)$$
$$= (\mathbf{f}_b(1) + \mathbf{f}_a(2)\mathbf{f}_b(2))\mathbf{b}(2n) + \mathbf{f}_b(2)(\mathbf{f}_a(1)\mathbf{b}(2n-1) + \mathbf{f}_a(3)\mathbf{b}(2n-2))$$

Thus, the computation of $\mathbf{A}(n)$ or $\mathbf{B}(n)$ requires only three terms, namely, $\mathbf{b}(2n), \mathbf{b}(2n-1)$ and $\mathbf{b}(2n-2)$.

Unequal and odd fixed length type: Here N_a and N_b are odd but unequal. An example is described now. In the previous three cases, the modifier $M_a(n)$ consists of terms of $\mathbf{b}$ whose indices are less than $2n-1$. We can also form a modifier $M_a(n)$ with terms of $\mathbf{b}$ whose indices are greater than $2n-1$ as

$$\left.\begin{aligned} M_a(n) &= \sum_{i=2}^{La} \mathbf{f}_a(i)\mathbf{b}(2n-2+i) \\ M_b(n) &= \sum_{i=2}^{Lb} \mathbf{f}_b(i)\mathbf{A}(n+2-i) \end{aligned}\right\} \quad \cdots\cdots\cdots(5)$$

The decomposition equations for $L_a = 3$ and $L_b = 3$ are

$$\mathbf{A}(n) = \mathbf{f}_a(1)\mathbf{b}(2n-1) + \mathbf{f}_a(1)\mathbf{b}(2n) + \mathbf{f}_a(1)\mathbf{b}(2n+1)$$

$$\mathbf{B}(n) = \mathbf{f}_b(1)\mathbf{b}(2n) + \mathbf{f}_b(2)\mathbf{A}(n) + \mathbf{f}_b(3)\mathbf{A}(n-1) + \mathbf{f}_b(4)\mathbf{A}(n-2)$$

Computation of $\mathbf{A}(n)$ requires 3 terms of $\mathbf{b}$, i.e., $\mathbf{b}(2n-1), \mathbf{b}(2n)$ and $\mathbf{b}(2n+1)$ whereas the computation of $\mathbf{B}(n)$ requires 7 terms of $\mathbf{b}$, i.e., $\mathbf{b}(2n-5), \mathbf{b}(2n-4), \cdots \mathbf{b}(2n), \mathbf{b}(2n+1)$.

Note here that the reconstruction of $\mathbf{b}$ starts from the right end of the signal, i.e., from the highest values of n to the lowest values, as $M_a(n)$ has terms of $\mathbf{b}$ whose indices are greater than $2n$.

Mixed fixed length type: Here one of N_a and N_b is even and the other odd. For example, consider the modifiers below

$$\left.\begin{aligned} M_a(n) &= \sum_{i=2}^{La} \mathbf{f}_a(i)\mathbf{b}(2n-2+i) \\ M_b(n) &= \sum_{i=2}^{Lb} \mathbf{f}_b(i)\mathbf{b}(2n-2+i) \end{aligned}\right\} \quad \cdots\cdots\cdots(6)$$

When L_a is odd and L_b is even or vice versa in equation (6) , we get the mixed fixed length type. Note here that reconstruction starts from the maximum values of n, as the indices of $\mathbf{b}$ in both $M_a(n)$ and $M_b(n)$ are greater than $2n$. The modifiers are still determinable, if we reconstruct the signal $\mathbf{b}$ from its last term. But these modifiers are not determinable if we are to reconstruct the signal starting from its first term.

4.3.2 Semi-infinite length type YKSK-1 transform

In this case, one of $N_a(n)$ and $N_b(n)$ is fixed while the other increases with *n*. For example, consider the following modifiers.

$$\left.\begin{aligned} M_a(n) &= \sum_{i=2}^{La} \mathbf{f}_a(i)\mathbf{b}(2n + c_1 \pm i) \\ M_b(n) &= \sum_{i=2}^{Lb'} \mathbf{f}_b(i)\mathbf{B}(n+1-i) + \sum_{j=Lb'+1}^{Lb} \mathbf{f}_b(j)\mathbf{A}(n + c_2 \pm j) \end{aligned}\right\} \quad \cdots\cdots\cdots(7)$$

where c_1 and c_2 are some integer constants (zero, positive or negative) and $L_b^{'}$ is any positive integer such that $2 \le L_b^{'} \le L_b$. The number terms of $\mathbf{b}$ involved in the computation of $\mathbf{A}(n)$ here is L_a. But since $M_b(n)$ is a function of $\mathbf{B}(n-1)$, $\mathbf{B}(n-2)$, …etc., the number of terms of $\mathbf{b}$ involved in the computation of $\mathbf{B}(n)$ keeps changing for different values of n. A set of decomposition equations with modifiers of this type is known as semi infinite length type. Also, since the modifier $M_b(n)$ is a function of the decomposed components, reconstruction starts from the associated information carrier, here $\mathbf{b}(2n)$, knowledge of which is required for the reconstruction of the other information carrier $\mathbf{b}(2n-1)$.

With a proper choice of modifiers we can make number of terms of $\mathbf{b}$ involved in the computation of $\mathbf{B}(n)$ odd or even. Thus, we can define three different types of semi-infinite type, namely, even semi-infinite, odd semi-infinite and mixed semi-infinite.

Even semi-infinite length type: In this type, one of $N_a(n)$ and $N_b(n)$ is even and fixed while the other is even and increases with n till the last term of the corresponding decomposed component is computed. For example, if $L_a = 2$, $L_b = 3$, $L_b^{'} = 2$, $c_1 = 2, c_2 = 3$ and i, j have '-' signs in the expression of modifiers in equation (7), the decomposition equations become

$$\mathbf{A}(n) = \mathbf{f}_a(1)\mathbf{b}(2n-1) + \mathbf{f}_a(2)\mathbf{b}(2n)$$

$$\mathbf{B}(n) = \mathbf{f}_b(1)\mathbf{b}(2n) + \mathbf{f}_b(2)\mathbf{B}(n-1) + \mathbf{f}_b(3)\mathbf{A}(n)$$

Here $N_a(n)$ is fixed at 2 while $N_b(n)$ increases with n as shown below.

$$\mathbf{B}(1) = (\mathbf{f}_b(1) + \mathbf{f}_a(2)\mathbf{f}_b(3))\mathbf{b}(2) + \mathbf{f}_b(3)\mathbf{f}_a(1)\mathbf{b}(1)$$

$$\mathbf{B}(2) = [\mathbf{f}_b(1) + \mathbf{f}_a(2)\mathbf{f}_b(3)]\mathbf{b}(4) + \mathbf{f}_a(1)\mathbf{f}_b(3)\mathbf{b}(3) + \mathbf{f}_b(2)[\mathbf{f}_b(1) + \mathbf{f}_a(2)\mathbf{f}_b(3)]\mathbf{b}(2) + \mathbf{f}_b(2)\mathbf{f}_a(1)\mathbf{f}_b(3)\mathbf{b}(1)$$

and so on.

Thus, $N_b(n)$, the number of terms of $\mathbf{b}$ involved in the computation of $\mathbf{B}(n)$ increases as 2, 4, 6, … for $n = 1,2,3$ ... There are several other ways in which we can obtain even-semi-infinite length type of decomposition.

Odd semi-infinite length type: In this type, one of $N_a(n)$ and $N_b(n)$ is odd and fixed while the other is odd and increases with *n*. For example, suppose our decomposition equations are defined as follows.

$$\left.\begin{aligned}\mathbf{A}(n) &= \mathbf{f}_a(1)\mathbf{b}(2n) + \sum_{i=2}^{La}\mathbf{f}_a(i)\mathbf{b}(2n+1-i) \\ \mathbf{B}(n) &= \mathbf{f}_b(1)\mathbf{b}(2n-1) + \mathbf{f}_b(2)\mathbf{b}(2n-2) + \sum_{i=3}^{Lb}\mathbf{f}_b(i)\mathbf{B}(n+2-i)\end{aligned}\right\} \quad \cdots\cdots\cdots(8)$$

Here, $N_a(n)$ is fixed at an odd number L_a. But the number of terms of the signal $\mathbf{b}$ involved in the computation of $\mathbf{B}(n)$ is odd and increasing. If $L_b = 3$, $N_b(n)$ increases as shown below.

$\mathbf{B}(1) = \mathbf{f}_b(1)\mathbf{b}(1)$

$\mathbf{B}(2) = \mathbf{f}_b(1)\mathbf{b}(3) + \mathbf{f}_b(2)\mathbf{b}(2) + \mathbf{f}_b(3)\mathbf{f}_b(1)\mathbf{b}(1)$

$\mathbf{B}(3) = \mathbf{f}_b(1)\mathbf{b}(5) + \mathbf{f}_b(2)\mathbf{b}(4) + \mathbf{f}_b(3)[\mathbf{f}_b(1)\mathbf{b}(3) + \mathbf{f}_b(2)\mathbf{b}(2) + \mathbf{f}_b(3)\mathbf{f}_b(1)\mathbf{b}(1)]$

and so on.

There are several other ways in which we can obtain odd semi-infinite length type of decomposition.

Mixed semi-infinite length type: Here one of $N_a(n)$ and $N_b(n)$ is odd while the other is even. Also, one of them is fixed while the other increases with *n*. For example, when $L_a = 4$, $L_b = 3$ in equation (8), $N_a(n)$ is fixed at 4 while $N_b(n)$ is odd and increasing.

4.3.3 Infinite length type YKSK-1 transform

In this type, both $N_a(n)$ and $N_b(n)$ increase with n until the last terms of the decomposed components are computed. The modifiers below give rise to infinite length YKSK-1 transforms.

$$\left.\begin{aligned} M_a(n) &= \mathbf{f}_a(2)\mathbf{b}(2n+c_1) + \sum_{i=3}^{La}\mathbf{f}_a(i)\mathbf{A}(n+2-i) \\ M_b(n) &= \sum_{i=2}^{Lb'}\mathbf{f}_b(i)\mathbf{B}(n+1-i) + \sum_{j=Lb'+1}^{Lb}\mathbf{f}_b(j)\mathbf{A}(n+c_2 \pm j) \end{aligned}\right\} \quad \cdots\cdots\cdots(9)$$

where c_1 and c_2 are some integer constants (zero, positive or negative) and L_b' is such that $2 \le L_b' \le L_b$.

$N_a(n)$ keeps on increasing as n increases since the modifier $M_a(n)$ involves some previous terms of the decomposed component $\mathbf{A}$. The same is true for $N_b(n)$ also. Similar to the fixed length type, we have five different infinite length types, namely, equal and even infinite length type, equal and odd infinite length type, unequal and even infinite length type, unequal and odd infinite length type and mixed infinite length type.

Equal and even infinite length type: Here $N_a(n)$ and $N_b(n)$ are equal and even and increase with n. For example, when $L_a = 3$, $L_b = 3$, $L_b' = 2$, $c_2 = 2$ and j has '-' sign in equation (9), the modifiers become

$$M_a(n) = \mathbf{f}_a(2)\mathbf{b}(2n) + \mathbf{f}_a(3)\mathbf{A}(n-1)$$

$$M_b(n) = \mathbf{f}_b(2)\mathbf{B}(n-1) + \mathbf{f}_b(3)\mathbf{A}(n-1)$$

Each of $N_a(n)$ and $N_b(n)$ here increases as 2, 4, 6, 8, ... for $n = 1,2,3,4$, ….

Unequal and even infinite length type: Here $N_a(n)$ and $N_b(n)$ are unequal but even and increase with n. An example of this type is based on the following decomposition equations.

$$\mathbf{A}(n) = \mathbf{f}_a(1)\mathbf{b}(2n-1) + \mathbf{f}_a(2)\mathbf{b}(2n) + \mathbf{f}_a(3)\mathbf{A}(n-1)$$

$$\mathbf{B}(n) = \mathbf{f}_b(1)\mathbf{b}(2n) + \mathbf{f}_b(2)\mathbf{B}(n-1) + \mathbf{f}_b(3)\mathbf{A}(n+2)$$

$N_a(n)$ is even and increases as 2, 4, 6, 8, … for $n = 1,2,3,4$, …. And $N_b(n)$ is also even and increases as 6, 8, 10, 12, … for $n = 1,2,3,4$, … because of the presence of the term $\mathbf{A}(n+2)$ in the modifier. Clearly, the two sequences of numbers are different.

Equal and odd infinite length type: Here $N_a(n)$ and $N_b(n)$ are equal and odd and increase with n. For example, when $L_a = L_b = 3, L_b^{'} = 2,\ c_1 = -2, c_2 = 2$ and j has '-' sign in the expression for modifiers in the equation (9), each of $N_a(n)$ and $N_b(n)$ here increases as 1, 3, 5, 7, … for the 1st, 2nd, 3rd, 4th , … terms of the decomposed components.

Unequal and odd infinite length type: Here $N_a(n)$ and $N_b(n)$ are unequal. But both are odd and increasing with n. The following decomposition equations produce an example of this type.

$$\left.\begin{aligned}\mathbf{A}(n) &= \mathbf{f}_a(1)\mathbf{b}(2n-1) + \mathbf{f}_a(2)\mathbf{b}(2n-2) + \mathbf{f}_a(3)\mathbf{A}(n-1)\\ \mathbf{B}(n) &= \mathbf{f}_b(1)\mathbf{b}(2n) + \mathbf{f}_b(3)\mathbf{A}(n+1)\end{aligned}\right\} \quad \cdots\cdots\cdots(10)$$

In the decomposition equation for $\mathbf{B}(n)$, we assume that $\mathbf{f}_b(2) = 0$. $N_a(n)$ increases as 1, 3, 5, … for $n = 1,2,3$, ... But $N_b(n)$ increases as 3, 5, 7, ... for $n = 1,2,3$, ….

Mixed infinite length type: In this type, the values of $N_a(n)$ and $N_b(n)$ are neither all odd numbers nor all even numbers. In other words, this is an infinite length type which is not any of the above four types. For example, consider the following decomposed components.

$$\left.\begin{aligned}\mathbf{A}(n) &= \mathbf{f}_a(1)\mathbf{b}(2n) + \mathbf{f}_a(2)\mathbf{b}(2n-2) + \mathbf{f}_a(3)\mathbf{A}(n-1)\\ \mathbf{B}(n) &= \mathbf{f}_b(1)\mathbf{b}(2n-1) + \mathbf{f}_b(2)\mathbf{B}(n-1) + \mathbf{f}_b(3)\mathbf{A}(n-1)\end{aligned}\right\} \quad \cdots\cdots\cdots(11)$$

$N_a(n)$ increases as 1, 2, 3, 4, … for $n = 1,2,3,4$, ... while $N_b(n)$ increases as 1, 3, 5, 7, ... for $n = 1,2,3,4$, ….

Replacing the filter coefficients by 1's, we can easily check the number of elements of $\mathbf{b}$ involved in the computation of a decomposed component. The above decomposition equations then become

$$\mathbf{A}(n) = \mathbf{b}(2n) + \mathbf{b}(2n-2) + \mathbf{A}(n-1)$$

$$\mathbf{B}(n) = \mathbf{b}(2n-1) + \mathbf{B}(n-1) + \mathbf{A}(n-1)$$

Thus, we can easily check the required number of elements of $\mathbf{b}$ for various values of n as

$$\mathbf{A}(1) = \mathbf{b}(2)$$
$$\mathbf{A}(2) = \mathbf{b}(4) + 2\mathbf{b}(2)$$
$$\mathbf{A}(3) = \mathbf{b}(6) + 2[\mathbf{b}(4) + \mathbf{b}(2)]$$
$$\mathbf{A}(4) = \mathbf{b}(8) + 2[\mathbf{b}(6) + \mathbf{b}(4) + \mathbf{b}(2)]$$

and so on.

$$\mathbf{B}(1) = \mathbf{b}(1)$$
$$\mathbf{B}(2) = \mathbf{b}(3) + \mathbf{b}(2) + \mathbf{b}(1)$$
$$\mathbf{B}(3) = \mathbf{b}(5) + \mathbf{b}(4) + \mathbf{b}(3) + 3\mathbf{b}(2) + \mathbf{b}(1)$$
$$\mathbf{B}(4) = \mathbf{b}(7) + \mathbf{b}(6) + \mathbf{b}(5) + 3\mathbf{b}(4) + \mathbf{b}(3) + 5\mathbf{b}(2) + \mathbf{b}(1)$$

and so on.

The number of terms of $\mathbf{b}$ involved in the computation of $\mathbf{A}(n)$ is n while the same in the case of $\mathbf{B}(n)$ is $2n-1$.

4.4 Multi-Stage Decomposition of YKSK-1Transform

The decomposition equations can also be subdivided into several intermediate stages as long as the information carriers are retraceable from the decomposed components for reconstruction purpose. Multi-stage decomposition scheme will allow us to do the decomposition of a signal with long filters in a single stage in several stages with shorter filters as in [41]. Let us first see 2-stage decomposition where there are two successive stages to find the decomposed components.

Stage-1: Let the decomposed components in stage-1 are denoted by $\mathbf{A}^1$ and $\mathbf{B}^1$, and the corresponding modifiers are $M_a^1(n)$ and $M_b^1(n)$. Then we can write the intermediate stage decomposition equation as

$$\left.\begin{aligned}\mathbf{A}^1(n) &= \mathbf{f}_a(1)\mathbf{b}(2n-1) + M_a^1(n)\\ \mathbf{B}^1(n) &= \mathbf{f}_b(1)\mathbf{b}(2n) + M_b^1(n)\end{aligned}\right\} \quad \cdots\cdots\cdots(12)$$

Suppose $M_a^1(n) = \mathbf{f}_a(2)\mathbf{b}(2n) + \mathbf{f}_a(3)\mathbf{b}(2n+1)$, $M_b^1(n) = \mathbf{f}_b(2)\mathbf{A}^1(n-1) + \mathbf{f}_b(3)\mathbf{A}^1(n)$ are the modifiers in stage-1.

Stage-2: The actual decomposed components are obtained in this stage from the stage-1 decomposed components as

$$\left.\begin{aligned}\mathbf{A}^2(n) &= \mathbf{f}_a(4)\mathbf{A}^1(n) + M_a^2(n)\\ \mathbf{B}^2(n) &= \mathbf{f}_b(4)\mathbf{B}^1(n) + M_b^2(n)\end{aligned}\right\} \quad \cdots\cdots\cdots(13)$$

Suppose the modifiers here in this stage are $M_a^2(n) = \mathbf{f}_a(5)\mathbf{B}^1(n-1) + \mathbf{f}_a(6)\mathbf{B}^1(n)$, and $M_b^2(n) = \mathbf{f}_b(5)\mathbf{A}^1(n-1) + \mathbf{f}_b(6)\mathbf{A}^1(n)$. These are valid determinable modifiers since we know the values of $\mathbf{A}^1$ and $\mathbf{B}^1$ from stage-1.

The decomposed components are given by $\mathbf{A}^2$ and $\mathbf{B}^2$. We can get back the original signal from these two decomposed components by reversing the equations (12) and (13) backwards. So the reconstruction also involves two stages.

Stage-1: Reconstruction of $\mathbf{A}^1$ and $\mathbf{B}^1$ from $\mathbf{A}^2$ and $\mathbf{B}^2$ as

$$\left.\begin{aligned}\mathbf{B}^1(n) &= (\mathbf{B}^2(n) - M_b^2(n))/\mathbf{f}_b(4)\\ \mathbf{A}^1(n) &= (\mathbf{A}^2(n) - M_a^2(n))/\mathbf{f}_a(4)\end{aligned}\right\} \quad \cdots\cdots\cdots(14)$$

Stage-2: Reconstruction of $\mathbf{b}(2n-1)$ and $\mathbf{b}(2n)$ from $\mathbf{A}^1$ and $\mathbf{B}^1$ as

$$\left.\begin{aligned}\mathbf{b}(2n) &= (\mathbf{B}^1(n) - M_b^1(n))/\mathbf{f}_b(1)\\ \mathbf{b}(2n-1) &= (\mathbf{A}^1(n) - M_a^1(n))/\mathbf{f}_a(1)\end{aligned}\right\} \quad \cdots\cdots\cdots(15)$$

Thus, we can easily reconstruct the original signal in a 2-stage decomposition scheme.

There is no restriction on the number of intermediate stages in the decomposition scheme. Also, the number of decomposition stages may not be equal for the two decomposed components. In other words, we are free to choose or modify our decomposed components in various ways. In general, we can find a decomposed component through $(j+1)$ stages and the other component through $(k+1)$ stages. Suppose the number of the filter coefficients required in Stage-1 is l_1, that in Stage-2 is l_2, that in Stage-3 is l_3 and so on. And the last stage gives the decomposed component. We can write the $(k+1)$st stage decomposed component $\mathbf{A}^{k+1}$ using the decomposition filter $\mathbf{f}_a$ as

$$\left.\begin{aligned}\mathbf{A}^1(n) &= \mathbf{f}_a(1)\mathbf{b}(2n-1) + M_a^1(n)\\ \mathbf{A}^2(n) &= \mathbf{f}_a(1+l_1)\mathbf{A}^1(n) + M_a^2(n)\\ \mathbf{A}^3(n) &= \mathbf{f}_a(1+l_1+l_2)\mathbf{A}^2(n) + M_a^3(n)\\ &\cdots\ \cdots\ \cdots\ \cdots\cdots\ \cdots\ \cdots\ \cdots\\ \mathbf{A}^{k+1}(n) &= \mathbf{f}_a(1+l_1+l_2+\cdots+l_k)\mathbf{A}^k(n) + M_a^{k+1}(n)\end{aligned}\right\} \quad \cdots\cdots\cdots(16)$$

In equation (16), we assume that the modifier in each stage is determinable. Once the modifiers are known, we can recover the information carrier $\mathbf{b}(2n-1)$ from the decomposed component $\mathbf{A}^{k+1}$ very easily just by reversing the decomposition equation in bottom up fashion as

$$\left.\begin{aligned}
&\mathbf{A}^{k}(n) = (\mathbf{A}^{k+1}(n) - M_a^{k+1}(n))/\mathbf{f}_a(1 + l_1 + l_2 + \cdots + l_k)\\
&\cdots \quad \cdots \quad \cdots \quad \cdots\cdots \quad \cdots \quad \cdots \quad \cdots\\
&\mathbf{A}^{2}(n) = (\mathbf{A}^{3}(n) - M_a^{3}(n))/\mathbf{f}_a(1 + l_1 + l_2)\\
&\mathbf{A}^{1}(n) = (\mathbf{A}^{2}(n) - M_a^{2}(n))/\mathbf{f}_a(1 + l_1)\\
&\mathbf{b}(2n-1) = (\mathbf{A}^{1}(n) - M_a^{1}(n))/\mathbf{f}_a(1)
\end{aligned}\right\} \quad \cdots\cdots\cdots(17)$$

Similarly, we can find the other decomposed component $\mathbf{B}^{j+1}$ through $(j+1)$ stages using the decomposition filter $\mathbf{f}_b$ as

$$\left.\begin{aligned}
&\mathbf{B}^{1}(n) = \mathbf{f}_b(1).\mathbf{b}(2n) + M_b^{1}(n)\\
&\mathbf{B}^{2}(n) = \mathbf{f}_b(1 + l_1).\mathbf{B}^{1}(n) + M_b^{2}(n)\\
&\mathbf{B}^{3}(n) = \mathbf{f}_b(1 + l_1 + l_2).\mathbf{B}^{2}(n) + M_b^{3}(n)\\
&\cdots \quad \cdots \quad \cdots \quad \cdots\cdots \quad \cdots \quad \cdots \quad \cdots\\
&\mathbf{B}^{j+1}(n) = \mathbf{f}_b(1 + l_1 + l_2 + \cdots + l_j).\mathbf{B}^{j}(n) + M_b^{j+1}(n)
\end{aligned}\right\} \quad \cdots\cdots\cdots(18)$$

$$\left.\begin{aligned}
&\mathbf{B}^{j}(n) = (\mathbf{B}^{j+1}(n) - M_b^{j+1}(n))/\mathbf{f}_b(1 + l_1 + l_2 + \cdots + l_j)\\
&\cdots \quad \cdots \quad \cdots \quad \cdots\cdots \quad \cdots \quad \cdots \quad \cdots\\
&\mathbf{B}^{2}(n) = (\mathbf{B}^{3}(n) - M_b^{3}(n))/\mathbf{f}_b(1 + l_1 + l_2)\\
&\mathbf{B}^{1}(n) = (\mathbf{B}^{2}(n) - M_b^{2}(n))/\mathbf{f}_b(1 + l_1)\\
&\mathbf{b}(2n) = (\mathbf{B}^{1}(n) - M_b^{1}(n))/\mathbf{f}_b(1)
\end{aligned}\right\} \quad \cdots\cdots\cdots(19)$$

Thus, we see that even if the decomposed components are obtained through different stages, it is always possible to reconstruct the original signal provided the modifiers are determinable. Also note that there is no restriction on the choice of the filter coefficients for perfect reconstruction except that the filter coefficients corresponding to the first terms of the decomposition equation should be non-zero.

4.5 Integer Version of YKSK-1 Transform

From the decomposition equations, we know that a signal can be decomposed into two components using arbitrarily chosen filters. We can get the original signal back from its decomposed components using reconstruction equation (2), if we can determine the values of the modifiers required in the reconstruction process. We have also discussed various

types of modifiers in the previous section. Let us now see how simple the decomposition and reconstruction schemes are, when the filter coefficients associated with the information carriers are unity.

So, when $\mathbf{f}_a(1) = 1$, $\mathbf{f}_b(1) = 1$, the decomposition equation (1) becomes

$$\left.\begin{aligned}\mathbf{A}(n) &= \mathbf{b}(2n-1) + M_a(n)\\ \mathbf{B}(n) &= \mathbf{b}(2n) + M_b(n)\end{aligned}\right\} \quad \cdots\cdots\cdots(20)$$

Similarly, the reconstruction equation (2) becomes

$$\left.\begin{aligned}\mathbf{b}(2n) &= \mathbf{B}(n) - M_b(n)\\ \mathbf{b}(2n-1) &= \mathbf{A}(n) - M_a(n)\end{aligned}\right\} \quad \cdots\cdots\cdots(21)$$

Suppose our input signal is an integer array and we are interested in obtaining the decomposed components as integer values regardless of whether the decomposition filter coefficients are integers or real (i.e., floating point) numbers. We can easily do it by making the modifiers' values integer either by ceiling or flooring or rounding off to the nearest integer. In most of the cases flooring is used to make the decomposed components integers. Thus, the integer form of YKSK-1 transform is defined by decomposition equation (22) and reconstruction equation (21) given below.

$$\left.\begin{aligned}\mathbf{A}(n) &= \mathbf{b}(2n-1) + \lfloor M_a(n) \rfloor\\ \mathbf{B}(n) &= \mathbf{b}(2n) + \lfloor M_b(n) \rfloor\end{aligned}\right\} \quad \cdots\cdots\cdots(22)$$

$$\left.\begin{aligned}\mathbf{b}(2n) &= \mathbf{B}(n) - \lfloor M_b(n) \rfloor\\ \mathbf{b}(2n-1) &= \mathbf{A}(n) - \lfloor M_a(n) \rfloor\end{aligned}\right\} \quad \cdots\cdots\cdots(23)$$

Thus, we can get integer decomposed components even if we have floating point modifier values. It is a general notion [1], [21] that there will be information loss due to the flooring operations and hence 0.5 is added to the modifier terms before flooring off to the nearest integer as

$$\left.\begin{aligned}\mathbf{A}(n) &= \mathbf{b}(2n-1) + \lfloor M_a(n) + 0.5 \rfloor\\ \mathbf{B}(n) &= \mathbf{b}(2n) + \lfloor M_b(n) + 0.5 \rfloor\end{aligned}\right\} \quad \cdots\cdots\cdots(24)$$

But in fact there is no loss or gain of information due to flooring or ceiling or rounding off to an integer in the decomposition or reconstruction process because any modifier value added

during decomposition is subtracted during reconstruction. So we can add or subtract any value to or from our modifier terms before making them integers and still we will be able to reconstruct the original signal perfectly. That is, we can even write the decomposition equation (22) as equation (25) and the reconstruction equation (23) as equation (26) given below.

$$\left.\begin{aligned}\mathbf{A}(n) &= \mathbf{b}(2n-1) + \lfloor M_a(n) \pm r_1 \rfloor \\ \mathbf{B}(n) &= \mathbf{b}(2n) + \lfloor M_b(n) \pm r_2 \rfloor\end{aligned}\right\} \quad \cdots\cdots\cdots(25)$$

$$\left.\begin{aligned}\mathbf{b}(2n) &= \mathbf{B}(n) - \lfloor M_b(n) \pm r_1 \rfloor \\ \mathbf{b}(2n-1) &= \mathbf{A}(n) - \lfloor M_a(n) \pm r_2 \rfloor\end{aligned}\right\} \quad \cdots\cdots\cdots(26)$$

where r_1, r_2 are any two real constants (positive or negative or zero).

Thus, addition or subtraction of any real number to or from a modifier before making it an integer, does not affect perfect reconstruction.

Also, it is not necessary to use the same flooring or ceiling function for the two modifiers to make them integers. We can use different types of operators to make different modifiers integers. For example, we can use flooring operator for one modifier and ceiling operator for the other modifier as shown below.

$$\left.\begin{aligned}\mathbf{A}(n) &= \mathbf{b}(2n-1) + \lfloor M_a(n) \pm r_1 \rfloor \\ \mathbf{B}(n) &= \mathbf{b}(2n) + \lceil M_b(n) \pm r_2 \rceil\end{aligned}\right\} \quad \cdots\cdots\cdots(27)$$

$$\left.\begin{aligned}\mathbf{b}(2n) &= \mathbf{B}(n) - \lfloor M_b(n) \pm r_1 \rfloor \\ \mathbf{b}(2n-1) &= \mathbf{A}(n) - \lceil M_a(n) \pm r_2 \rceil\end{aligned}\right\} \quad \cdots\cdots\cdots(28)$$

One can also use rounding off operation in combination with ceiling or flooring operations for making modifiers integers.

The multi-stage decomposition described earlier can also be made integer transforms. The only requirement is that the filter coefficient associated with the information carrier at each stage should be unity and the modifier term should be made an integer by flooring or ceiling or rounding off to the nearest integer. For example, in 2-stage case, the decomposition equations become

$$\left.\begin{aligned}\mathbf{A}^1\mathbf{(n)} &= \mathbf{b}(2n-1) + \lfloor M_a^1(n) \rfloor \\ \mathbf{B}^1(n) &= \mathbf{b}(2n) + \lfloor M_b^1(n) \rfloor\end{aligned}\right\} \quad \cdots\cdots\cdots(29)$$

$$\left.\begin{aligned}\mathbf{A}^2(n) &= \mathbf{A}^1(n) + \left\lfloor M_a^2(n) \right\rfloor \\ \mathbf{B}^2(n) &= \mathbf{B}^1(n) + \left\lfloor M_b^2(n) \right\rfloor\end{aligned}\right\} \quad \cdots\cdots\cdots(30)$$

The reconstruction equations can easily be obtained from equations (29) and (30) in the usual manner. In this way, we can get integer transform for multi-stage decomposition also. Thus, we can get integer value decomposed components for an integer value signal in spite of the floating point values of the decomposition filter coefficients used in the modifiers.

4.6 Generalization of Some Well-known Integer Transforms Through YKSK-1 Transform

In this section we reproduce some of the integer-to-integer transforms commonly used for image compression. But these transforms were produced in several occasions without any proper generalization. Some tried to explain them in terms of wavelets, others tried to explain in terms of polyphase filter bank (PFB). For example, S-transform is considered to be the integer version of Haar's wavelet [55]. Wim Swelden's lifting scheme, which was explained on the basis of polynomial factorization, does not strictly lead to integer transform. One can by brute force obtain integer transform from it by rejecting the scaling factor in the last step of polynomial factorization. The theory of perfect reconstruction lifting scheme based on 2-channel PFB in [41] is not in actual case necessary, very much evident from the perfect reconstruction theory discussed in YKSK-1 transform and from [24]. And strictly lifting scheme cannot be taken as a generalization of discrete wavelet transform. The results obtained by using wavelet filters cannot in any way be obtained by using lifting scheme. YKSK-1 transform is the generalized form of the lifting scheme and some of the existing integer transforms. We can obtain the same result of the lifting scheme or any of the integer transform in YKSK-1. Some of the existing integer transforms are generalized here. Note here that in the generalization we removed the factor (1/2) which is not necessary for perfect reconstruction. We find most of the existing integer transforms as special cases of fixed length YKSK-1 transform.

1. S-transform:

S-transform is usually considered as the integer version of Haar's transform. This is special case of equal and even fixed length type of YKSK-1 transform. The number of terms of the signal $\mathbf{b}$ involved in the computation of each decomposed component is fixed at 2.

The generalized decomposition equations of S-transform become

$$\left.\begin{aligned}\mathbf{A}(n) &= \mathbf{f}_a(1)\mathbf{b}(2n-1) + \mathbf{f}_a(2)\mathbf{b}(2n)\\ \mathbf{B}(n) &= \mathbf{f}_b(1)\mathbf{b}(2n) + \mathbf{f}_b(2)\mathbf{A}(n)\end{aligned}\right\} \quad \cdots\cdots\cdots(31)$$

where $n = 1,2,\cdots, m/2$

In the above decomposition equations, modifiers are functions of n, hence we write them as $M_a(n) = \mathbf{f}_a(2)\mathbf{b}(2n)$, $M_b(n) = \mathbf{f}_b(2)\mathbf{A}(n)$.

Reconstruction of the signal can be obtained from equation (31) as

$$\left.\begin{aligned}\mathbf{b}(2n) &= (\mathbf{B}(n) - M_b(n))/\mathbf{f}_b(1)\\ \mathbf{b}(2n-1) &= (\mathbf{A}(n) - M_a(n))/\mathbf{f}_a(1)\end{aligned}\right\} \quad \cdots\cdots\cdots(32)$$

We can obtain the integer version of equations (31) and (32) by putting $\mathbf{f}_a = [1, \mathbf{f}_a(2)]$, and $\mathbf{f}_b = [1, \mathbf{f}_b(2)]$ as given below.

$$\left.\begin{aligned}\mathbf{A}(n) &= \mathbf{b}(2n-1) + \lfloor \mathbf{f}_a(2)\mathbf{b}(2n) \rfloor\\ \mathbf{B}(n) &= \mathbf{b}(2n) + \lfloor \mathbf{f}_b(2)\mathbf{A}(n) \rfloor\end{aligned}\right\} \quad \cdots\cdots\cdots(33)$$

$$\left.\begin{aligned}\mathbf{b}(2n) &= \mathbf{B}(n) - \lfloor \mathbf{f}_b(2)\mathbf{A}(n) \rfloor\\ \mathbf{b}(2n-1) &= \mathbf{A}(n) - \lfloor \mathbf{f}_a(2)\mathbf{b}(2n) \rfloor\end{aligned}\right\} \quad \cdots\cdots\cdots(34)$$

When $\mathbf{f}_a = [1, -1]$ and $\mathbf{f}_b = [1, 0.5]$, the above YKSK-1 transform becomes S-transform [55]. We can choose any real values for the second elements of the decomposition filters and hence the resulting integer transform becomes more general than S-transform. As we are interested in integer transforms, we will provide only the generalized integer decomposition equations in the rest of this section. Also, we will not provide reconstruction equations as they are quite straightforward.

2. TS-transform:

It is a modification of S-transform by adding a modification stage in one of the decomposed components. The generalized integer decomposition equations of TS-transform can be written as.

$$\left.\begin{aligned}\mathbf{A}^1(n) &= \mathbf{b}(2n) + \lfloor \mathbf{f}_a(2)\mathbf{b}(2n-1) \rfloor \\ \mathbf{B}(n) &= \mathbf{b}(2n-1) + \lfloor \mathbf{f}_b(2)\mathbf{A}^1(n) \rfloor \\ \mathbf{A}(n) &= \mathbf{A}^1(n) + \lfloor \mathbf{f}_a(4)\mathbf{B}(n-2) + \mathbf{f}_a(5)\mathbf{B}(n) \rfloor\end{aligned}\right\} \quad \cdots\cdots\cdots(35)$$

In equation (35), $\mathbf{A}$ and $\mathbf{B}$ are the decomposed components. This is a case of unequal and even fixed length type where the number of terms of the signal $\mathbf{b}$ involved in the computation of $\mathbf{B}(n)$ is fixed at even number 2 and the same for $\mathbf{A}(n)$ is 4. For making integer transform, there is some restriction on choosing the coefficients of the decomposition filters. The restriction here is that the first elements of the decomposition filters should be 1. The third element of the decomposition filter $\mathbf{f}_a$ is also 1. But the rest of the elements, which will be the parts of the modifiers, can be chosen arbitrarily. Equation (23) becomes the TS transform (21) when $\mathbf{f}_a = [1, -1, 1, 0.25, -0.25]$ and $\mathbf{f}_b = [1, 0.5]$.

3. S+P transform:

It is also a modification of S-transform. The first two equations of equation (35) are kept intact and a different modification is made in the third equation as follows.

$$\left.\begin{aligned}\mathbf{A}^1(n) &= \mathbf{b}(2n) + \lfloor \mathbf{f}_a(2)\mathbf{b}(2n-1) \rfloor \\ \mathbf{B}(n) &= \mathbf{b}(2n-1) + \lfloor \mathbf{f}_b(2)\mathbf{A}^1(n) \rfloor \\ \mathbf{A}(n) &= \mathbf{A}^1(n) + \lfloor M_a(n) \rfloor\end{aligned}\right\} \quad \cdots\cdots\cdots(36)$$

$$M_a(n) = \mathbf{f}_a(4)\mathbf{B}(n-2) + \mathbf{f}_a(5)\mathbf{B}(n-1) + \mathbf{f}_a(6)\mathbf{B}(n) + \mathbf{f}_a(7)\mathbf{B}(n+1) + \mathbf{f}_a(8)\mathbf{A}^1(n+1)$$

This is also a case of unequal and even fixed length type. Computation of $\mathbf{B}(n)$ requires 2-terms of $\mathbf{b}$ and the computation of $\mathbf{A}(n)$ requires 8 terms. When $\mathbf{f}_a = [1, -1, 1, 0, 0.25, 0, -0.25, 0]$, equation (36) becomes TS transform. When $\mathbf{f}_a = [1, -1, 1, 0, 0.25, 0.125, -0.375, 0.25]$ equation (36) becomes SPB transform [1]. And it becomes SPC integer transform [1] when $\mathbf{f}_a = [1, -1, 1, -1/16, 5/16, 1/4, -1/2, 3/8]$. Note here that in the case of SPB, computation of $\mathbf{A}(n)$ requires 6 terms of $\mathbf{b}$. Said and Pearlman [110] suggested SPB for natural image compression.

4. (2,2) interpolating transform or (5,3) biorthogonal filter transform:

This is a case of unequal odd fixed length type where the computation of $\mathbf{B}(n)$ requires 5 terms of $\mathbf{b}$ and computation of $\mathbf{A}(n)$ requires 3 terms.

The generalized decomposition equations in its integer version is given by

$$\left.\begin{aligned}\mathbf{A}(n) &= \mathbf{b}(2n) + \lfloor \mathbf{f}_a(2)\mathbf{b}(2n-1) + \mathbf{f}_a(3)\mathbf{b}(2n+1) \rfloor \\ \mathbf{B}(n) &= \mathbf{b}(2n-1) + \lfloor \mathbf{f}_b(2)\mathbf{A}(n-1) + \mathbf{f}_b(3)\mathbf{A}(n) \rfloor \end{aligned}\right\} \quad \cdots\cdots\cdots(37)$$

When $\mathbf{f}_a = [1, -1/2, -1/2]$ and $\mathbf{f}_b = [1, 1/4, 1/4]$, equation (37) becomes the (2,2) interpolating transform [21].

5. (4,2) interpolating transform:

It is a modification of (2,2) interpolating transform. The modification is made only on the first decomposition equation. But this modification also affects the number of terms of $\mathbf{b}$ involved in the computation of $\mathbf{B}(n)$. The generalized decomposition equations in its integer version is given by

$$\left.\begin{aligned}\mathbf{A}(n) &= \mathbf{b}(2n) - \lfloor \mathbf{f}_a(2)\mathbf{b}(2n-1) + \mathbf{f}_a(3)\mathbf{b}(2n+1) + \mathbf{f}_a(4)\mathbf{b}(2n+2) + \mathbf{f}_a(5)\mathbf{b}(2n+3) \rfloor \\ \mathbf{B}(n) &= \mathbf{b}(2n-1) + \lfloor \mathbf{f}_b(2)\mathbf{A}(n-1) + \mathbf{f}_b(3)\mathbf{A}(n) \rfloor \end{aligned}\right\} \cdots\cdots(38)$$

This is a case of unequal and even fixed length case where the computation of $\mathbf{B}(n)$ requires 7 terms of $\mathbf{b}$ and computation of $\mathbf{A}(n)$ requires 4 terms.

When $\mathbf{f}_a = [1, 1/16, -9/16, -9/16, 1/16]$ and $\mathbf{f}_b = [1, 1/4, 1/4]$, equation (26) becomes the (4,2) interpolating transform [24] which is also referred to as 9/7M [1].

6. (2,4) interpolating transform:

It is also a modification of the (2,2) interpolating transform. In this case modification is made on the second decomposition equation. This modification results in the unequal and odd fixed length type where the computation of $\mathbf{B}(n)$ requires 9 terms of $\mathbf{b}$ and the computation of $\mathbf{A}(n)$ requires 3 terms.

The integer version of the generalized decomposition equations is given by

$$\left.\begin{aligned}\mathbf{A}(n) &= \mathbf{b}(2n) - \lfloor \mathbf{f}_a(2)\mathbf{b}(2n-1) + \mathbf{f}_a(3)\mathbf{b}(2n+1) \rfloor \\ \mathbf{B}(n) &= \mathbf{b}(2n-1) + \lfloor \mathbf{f}_b(2)\mathbf{A}(n-2) + \mathbf{f}_b(3)\mathbf{A}(n-1) + \mathbf{f}_b(4)\mathbf{A}(n) + \mathbf{f}_b(5)\mathbf{A}(n+1) \rfloor \end{aligned}\right\} \cdots\cdots(39)$$

When $\mathbf{f}_a = [1, -1/2, -1/2]$ and $\mathbf{f}_b = [1, -3/64, 19/64, 19/64, -3/64]$, it becomes the (2,4) interpolating transform [24].

7. (2/6) integer transform:

It is a case of unequal and even fixed length type where the computation of $\mathbf{B}(n)$ requires 2 terms of $\mathbf{b}$ and computation of $\mathbf{A}(n)$ requires 6 terms. The generalized integer version of the transform is given by

$$\left.\begin{aligned} \mathbf{A}^1(n) &= \mathbf{b}(2n) + \lfloor \mathbf{f}_a(2)\mathbf{b}(2n-1) \rfloor \\ \mathbf{B}(n) &= \mathbf{b}(2n-1) + \lfloor \mathbf{f}_b(2)\mathbf{A}^1(n) \rfloor \\ \mathbf{A}(n) &= \mathbf{A}^1(n) + \lfloor \mathbf{f}_a(4)\mathbf{B}(n-1) + \mathbf{f}_a(5)\mathbf{B}(n+1) \rfloor \end{aligned}\right\} \quad \cdots\cdots\cdots(40)$$

When $\mathbf{f}_a = [1, -1, 1, 1/4, -1/4]$ and $\mathbf{f}_b = [1, 1/2]$, it becomes the (2/6) integer transform [1].

8. (2/10) integer transform:

It is a modification of (2/6) case where the modification is made only in the last stage of decomposition equation of $\mathbf{A}(n)$. This case is also of unequal and even fixed length type where the computation of $\mathbf{B}(n)$ requires 2 terms of $\mathbf{b}$ and computation of $\mathbf{A}(n)$ requires 10 terms. The generalized integer version of the transform is given by

$$\left.\begin{aligned} \mathbf{A}^1(n) &= \mathbf{b}(2n) + \lfloor \mathbf{f}_a(2)\mathbf{b}(2n-1) \rfloor \\ \mathbf{B}(n) &= \mathbf{b}(2n-1) + \lfloor \mathbf{f}_b(2)\mathbf{A}^1(n) \rfloor \\ \mathbf{A}(n) &= \mathbf{A}^1(n) + \lfloor \mathbf{f}_a(4)\mathbf{B}(n-2) + \mathbf{f}_a(5)\mathbf{B}(n-1) + \mathbf{f}_a(6)\mathbf{B}(n+1) + \mathbf{f}_a(7)\mathbf{B}(n+2) \rfloor \end{aligned}\right\} \cdots\cdots(41)$$

When $\mathbf{f}_a = [1, -1, 1, -3/64, 22/64, -22/64, 3/64]$ and $\mathbf{f}_b = [1, 1/2]$, it becomes the (2/10) integer transform [1].

9. (6,2) interpolating transform:

It is also a modification of (2,2) interpolating transform. The modification is made only in the first decomposition equation. It is a case of unequal and odd fixed length type where the computation of a term of $\mathbf{A}(n)$ requires 7 terms of $\mathbf{b}$ and the computation of a term of $\mathbf{B}(n)$ requires 9 terms. The integer version of the generalized decomposition equations is given by

$$\left.\begin{aligned}\mathbf{A}(n) &= \mathbf{b}(2n) + \lfloor M_a(n) \rfloor \\ \mathbf{B}(n) &= \mathbf{b}(2n-1) + \lfloor \mathbf{f}_b(2)\mathbf{A}(n-1) + \mathbf{f}_b(3)\mathbf{A}(n) \rfloor\end{aligned}\right\} \quad \cdots\cdots\cdots(42)$$

$$M_a(n) = \mathbf{f}_a(2)\mathbf{b}(2n-5) + \mathbf{f}_a(3)\mathbf{b}(2n-3) + \mathbf{f}_a(4)\mathbf{b}(2n-1) + \mathbf{f}_a(5)\mathbf{b}(2n+1) + \mathbf{f}_a(6)\mathbf{b}(2n+3) + \mathbf{f}_a(7)\mathbf{b}(2n+5)$$

When $\mathbf{f}_a = [1, -3/256, 25/256, -75/128, -75/128, 25/256, -3/256]$ and $\mathbf{f}_b = [1, 1/4, 1/4]$, equation (42) becomes (6,2) interpolating transform [24].

10. (4,4) interpolating transform:

It is a modification of (4,2), interpolating transform where the modification is made in the second decomposition equation. The modification leads to the unequal and odd fixed length type where the computation of $\mathbf{B}(n)$ requires 13 terms of $\mathbf{b}$ and computation of $\mathbf{A}(n)$ requires 7 terms. The integer version of the generalized decomposition equations is given by

$$\left.\begin{aligned}\mathbf{A}(n) &= \mathbf{b}(2n) + \lfloor M_a(n) \rfloor \\ \mathbf{B}(n) &= \mathbf{b}(2n-1) + \lfloor M_b(n) \rfloor\end{aligned}\right\} \cdots\cdots(43)$$

where $M_a(n) = \mathbf{f}_a(2)\mathbf{b}(2n-3) + \mathbf{f}_a(3)\mathbf{b}(2n-1) + \mathbf{f}_a(4)\mathbf{b}(2n+1) + \mathbf{f}_a(5)\mathbf{b}(2n+3)$

$M_b(n) = \mathbf{f}_b(2)\mathbf{A}(n-2) + \mathbf{f}_b(3)\mathbf{A}(n-1) + \mathbf{f}_b(4)\mathbf{A}(n) + \mathbf{f}_b(5)\mathbf{A}(n+1)$.

When $\mathbf{f}_a = [1, 1/16, -9/16, -9/16, 1/16]$ and $\mathbf{f}_b = [1, -1/32, 9/32, 9/32, -1/32]$, then equation (43) becomes (4,4) interpolating transform [24], which is also referred to as (13/7-T) [1]. And when $\mathbf{f}_b = [1, -1/16, 5/16, 5/16, -1/16]$, it is known as (13/7-C) integer transform [1].

11. (2+2,2) interpolating transform:

It is also a modification of (2,2) interpolating transform where the modification is made for first decomposed component through two successive modifications. This is also a case of unequal fixed length type where the computation of a term of $\mathbf{B}(n)$ requires 5 terms and computation of $\mathbf{A}(n)$ requires 11 terms of the signal $\mathbf{b}$. The integer version of the generalized decomposition equations is given by

$$\left.\begin{aligned}\mathbf{A}^1(n) &= \mathbf{b}(2n) + \lfloor \mathbf{f}_a(2)\mathbf{b}(2n-1) + \mathbf{f}_a(3)\mathbf{b}(2n+1) \rfloor \\ \mathbf{B}(n) &= \mathbf{b}(2n-1) + \lfloor \mathbf{f}_b(2)\mathbf{A}^1(n-1) + \mathbf{f}_b(3)\mathbf{A}^1(n) \rfloor \\ \mathbf{A}(n) &= \mathbf{A}^1(n) + \lfloor M_a(n) \rfloor\end{aligned}\right\} \quad \cdots\cdots\cdots(44)$$

where $M_a(n) = \mathbf{f}_a(5)\mathbf{B}(n-1) + \mathbf{f}_a(6)\mathbf{B}(n) + \mathbf{f}_a(7)\mathbf{B}(n+1) + \mathbf{f}_a(8)\mathbf{B}(n+2) + \mathbf{f}_a(9)\mathbf{A}^1(n+1)$.

It is reported in [21] that when $\mathbf{f}_a = [1, -1/2, -1/2, 1, 1/8, 3/8, -3/8, -1/8, 1]$ and $\mathbf{f}_b = [1, 1/4, 1/4]$, it works better.

If the last element of the decomposition filter $\mathbf{f}_a$ is zero, i.e., $\mathbf{f}_a(9) = 0$, then it becomes the decomposition equation of 5/11 integer transform [1].

When $\mathbf{f}_a = [1, -1/2, -1/2, 1, 1/16, -1/16, -1/16, 1/16, 0]$, it is referred to as 5/11-C. And when $\mathbf{f}_a = [1, -1/2, -1/2, 1, 1/32, -1/32, -1/32, 1/32, 0]$, it is referred to as 5/11-B.

12. DB2 integer transform:

This integer transform corresponds to discrete wavelet transform using Daubechies' orthogonal wavelet filters of length-4. But is not strictly similar to it. This is a case of equal and even fixed length type where the computation of a term of each decomposed component requires 4 terms of the signal $\mathbf{b}$. The integer version of the decomposition equations corresponding to DB2 in a generalized form is given by

$$\left.\begin{aligned} \mathbf{A}^1(n) &= \mathbf{b}(2n) - \lfloor \mathbf{f}_a(2)\mathbf{b}(2n-1) \rfloor \\ \mathbf{B}(n) &= \mathbf{b}(2n-1) + \left\lfloor \mathbf{f}_b(2)\mathbf{A}^1(n-1) + \mathbf{f}_b(3)\mathbf{A}^1(n) \right\rfloor \\ \mathbf{A}(n) &= \mathbf{A}^1(n) + \lfloor \mathbf{f}_a(4)\mathbf{B}(n+1) \rfloor \end{aligned}\right\} \quad \cdots\cdots\cdots(45)$$

When $\mathbf{f}_a = [1, -\sqrt{3}, 1, 1]$ and $\mathbf{f}_b = [1, -(\sqrt{3}-2)/4, \sqrt{3}/4]$, it becomes the integer version of the DB2 transform through lifting scheme [41].

13. (9,7) integer transform:

The integer version of the transform through lifting scheme is generalized here. But this is again in no way similar to the decomposition scheme of (9,7) bi-orthogonal wavelet filters. This is a case of unequal and odd fixed length type where the computation of $\mathbf{A}(n)$ requires 7 terms of $\mathbf{b}$ and computation of $\mathbf{B}(n)$ requires 9 terms of $\mathbf{b}$. The generalized decomposition equation is

$$\left.\begin{aligned}
\mathbf{A}^1(n) &= \mathbf{b}(2n-1) + \lfloor \mathbf{f}_a(2)\mathbf{b}(2n) + \mathbf{f}_a(3)\mathbf{b}(2n+1) \rfloor \\
\mathbf{B}^1(n) &= \mathbf{b}(2n) + \lfloor \mathbf{f}_b(2)\mathbf{A}^1(n-1) + \mathbf{f}_b(3)\mathbf{A}^1(n) \rfloor \\
\mathbf{A}(n) &= \mathbf{A}^1(n) + \lfloor \mathbf{f}_a(5)\mathbf{B}^1(n) + \mathbf{f}_a(6)\mathbf{B}^1(n+1) \rfloor \\
\mathbf{B}(n) &= \mathbf{B}^1(n) + \lfloor \mathbf{f}_b(5)\mathbf{A}(n-1) + \mathbf{f}_b(6)\mathbf{A}(n) \rfloor
\end{aligned}\right\} \quad \cdots\cdots\cdots(46)$$

When $\mathbf{f}_a = [1, -1.58613434, -1.58613434, 1, 0.88291107, 0.88291107]$ and $\mathbf{f}_b = [1, -0.05298011, -0.05298011, 1, 0.44350885, 0.44350685]$, equation (32) becomes the integer version of the (9,7) biorthogonal symmetric transform through lifting scheme [24] This decomposition scheme is one of the most popular schemes for subband image compression and is referred to as (9/7-F) in [1].

14. 6/14 integer transform:

This is a case of unequal and even fixed length type where the computation of $\mathbf{B}(n)$ requires 6 terms of $\mathbf{b}$ and computation of $\mathbf{A}(n)$ requires 14 terms. The generalized integer version of the transform is

$$\left.\begin{aligned}
\mathbf{A}^1(n) &= \mathbf{b}(2n) + \lfloor \mathbf{f}_a(2)\mathbf{b}(2n-1) \rfloor \\
\mathbf{B}(n) &= \mathbf{b}(2n-1) + \lfloor \mathbf{f}_b(2)\mathbf{A}^1(n-1) + \mathbf{f}_b(3)\mathbf{A}^1(n) + \mathbf{f}_b(4)\mathbf{A}^1(n+1) \rfloor \\
\mathbf{A}(n) &= \mathbf{A}^1(n) + \lfloor \mathbf{f}_a(4)\mathbf{B}(n-2) + \mathbf{f}_a(5)\mathbf{B}(n-1) + \mathbf{f}_a(6)\mathbf{B}(n+1) + \mathbf{f}_a(7)\mathbf{B}(n+2) \rfloor
\end{aligned}\right\} \quad \cdots(47)$$

When $\mathbf{f}_a = [1, -1, 1, -1/16, 3/8, -3/8, 1/16]$ and $\mathbf{f}_b = [1, 1/16, 1/2, -1/16]$, it becomes (6/14) integer transform [1].

15. Cubic B-splines integer transform:

This transform is most commonly used in computer graphics. This is also a case of unequal and odd fixed length case where the computation of $\mathbf{B}(n)$ requires 5 terms of $\mathbf{b}$ and computation of $\mathbf{A}(n)$ requires 7 terms. The generalized integer version decomposition equations as obtained from lifting scheme is given by

$$\left.\begin{aligned}
\mathbf{A}^1(n) &= \mathbf{b}(2n-1) + \lfloor \mathbf{f}_a(2)\mathbf{b}(2n) + \mathbf{f}_a(3)\mathbf{b}(2n+1) \rfloor \\
\mathbf{B}(n) &= \mathbf{b}(2n) + \lfloor \mathbf{f}_b(2)\mathbf{A}^1(n-1) + \mathbf{f}_b(3)\mathbf{A}^1(n) \rfloor \\
\mathbf{A}(n) &= \mathbf{A}^1(n) + \lfloor \mathbf{f}_a(5)\mathbf{B}(n) + \mathbf{f}_a(6)\mathbf{B}(n+1) \rfloor
\end{aligned}\right\} \quad \cdots\cdots\cdots(48)$$

When $\mathbf{f}_a = [1, 1/4, 1/4, 1, -3/16, -3/16]$ and $\mathbf{f}_b = [1, 1, 1]$, equation (48) becomes the integer version of cubic B-splines obtained through lifting scheme [41].

4.7 Some Experimental Results

Three of the integer wavelet transforms described in Section 4.6 are considered here. These are (i) (4,4) interpolating transform, (ii) (2+2,2) interpolating transform and (iii) (9,7) integer transform. For each of them, the corresponding YKSK-1 generalized version is considered and a search technique is employed to find the optimal YKSK-1 filter giving the smallest value of the Mean Square Error (MSE). The MSE is calculated between the original image and the image reconstructed from only the approximation component from the first decomposition. level The search is based on a genetic algorithm. The images used for the experiment are lena128, lena256, barbara128, barbara256, boat128 and boat256 where "128" and "256" indicate a size of 128x128 and 256x256 respectively. Zero padding is used in the decomposition scheme.

4.7.1 (4,4) interpolating transform

In the (4,4) interpolating transform, there are 8 free filter coefficients, namely, $\mathbf{f_a}(2)$, $\mathbf{f_a}(3)$, $\mathbf{f_a}(4)$ $\mathbf{f_a}(5)$ and $\mathbf{f}_b(2)$, $\mathbf{f}_b(3)$, $\mathbf{f}_b(4)$, $\mathbf{f}_b(5)$. For each filter coefficient, an interval is considered to define the search space where a genetic algorithm can be applied. Let these intervals be I_1, I_2,….. I_8. Each such interval is centred around the corresponding coefficient of the 13/7-T filter set. For any filter set ($\mathbf{f_a}(2)$,… $\mathbf{f_a}(5)$, $\mathbf{f}_b(2)$,….. $\mathbf{f}_b(5)$) belonging to $I = I_1 \times I_2 \times \cdots \times I_8$ the MSE value can be computed after decomposing an image into four components and reconstructing the image from the approximation component only. Thus the MSE value can be thought of as a function $g(\mathbf{f_a}(2),\ldots \mathbf{f_a}(5), \mathbf{f}_b(2),\ldots. \mathbf{f}_b(5))$. The genetic algorithm developed here finds the optimal filter coefficients in the search space I which produces the smallest value of the function g. In this process, we get six optimizing YKSK-1 filter sets corresponding to six images. The filter set based on the average of these six filter sets is considered as a consensus YKSK-1 filter set which is in a sense the best based on the six images. The YKSK-1 filter set and the two standard integer wavelet filter sets (13/7-T and 13/7-C) in (4,4) interpolating transform are given in Table 4.1 where the MSE values produced by all these filter sets for each of the six images are also provided. It is seen that the proposed YKSK-1 filter sets perform better than the standard wavelet filter sets in terms of the MSE values.

4.7.2 (2+2,2) interpolating transform and (9,7) integer transform

In the (2+2,2) interpolating transform, there are 9 free filter coefficients, namely, $\mathbf{f_a}(2)$, $\mathbf{f_a}(3)$, $\mathbf{f_a}(5)$, …, $\mathbf{f_a}(9)$, $\mathbf{f}_b(2)$ and $\mathbf{f}_b(3)$. Here the search space I is 9-dimensional and the interval in each dimension is determined in a similar fashion using the 5/11-C filter set. The search of the optimal filter sets using a genetic algorithm is performed in the same way as in the case of (4,4) interpolating transform. The YKSK-1 filter set and the two standard wavelet filter sets (5/11-B and 5/11-C) in (2+2,2) interpolating transform are given in Table 4.2 where the MSE values produced by all these filter sets for each of the six images are also provided. It is seen that the proposed YKSK-1 filter sets perform better than the wavelet filter sets in terms of the MSE values. It is to be noted here that the standard (2+2, 2) filter set does not produce satisfactory results.

In the (9,7) integer transform, there are 8 free filter coefficients, namely, $\mathbf{f_a}(2)$, $\mathbf{f_a}(3)$, $\mathbf{f_a}(5)$, $\mathbf{f_a}(6)$ and $\mathbf{f}_b(2)$, $\mathbf{f}_b(3)$, $\mathbf{f}_b(5)$, $\mathbf{f}_b(6)$. The results obtained in a similar fashion are given in Table 4.3. Here also that the proposed YKSK-1 filter sets perform better than the (9,7) integer wavelet filter set.

In the above experiment, the intervals in the search space I are not large. This is because the goal here has been only to establish the fact that there are YKSK-1 filters that perform better than the existing filters. However, if the search is performed in a larger space, better YKSK-1 filters may emerge. Another point to note here is that the criterion for optimal filters has been taken as the MSE value in a certain context. However, any other criterion may be used for optimization with the basic algorithm remaining the same. There are several other wavelet integer filters, which have not been considered for comparison. However, corresponding optimal YKSK-1 filters can be computed in a similar fashion.

Filters	fa(2)	fa(3)	fa(4)	fa(5)	fb(2)	fb(3)	fb(4)	fb(5)
YKSK-1	0.1529	-0.8395	-0.3365	0.0269	-0.0154	0.1484	0.4527	-0.1039
13/7-T	0.0625	-0.5625	-0.5625	0.0625	-0.0312	0.2812	0.2812	-0.0312
13/7-C	0.0625	-0.5625	-0.5625	0.0625	-0.0625	0.3125	0.3125	-0.0625

Table 4.1a Filter sets of YKSK-1, 13/7-T and 13/7-C integer transforms.

Images	YKSK-1	13/7-T	13/7-C
lena128	118.38	142.10	140.62
lena256	68.56	78.19	77.35
barb128	117.97	135.71	134.38
barb256	100.44	109.92	109.18
boat128	174.91	215.81	213.68
boat256	120.95	134.90	133.52

Table 4.1b MSE values for six different images corresponding to the filter sets of YKSK-1, 13/7-T and 13/7-C integer transforms.

Images	fa(2)	fa(3)	fa(5)	fa(6)	fa(7)	fa(8)	fa(9)	fb(2)	fb(3)
YKSK-1	-0.6349	-0.3796	0.1141	-0.1551	0.0208	0.0512	-0.1411	0.1646	0.4620
5/11-C	-0.5000	-0.5000	0.0625	-0.0625	-0.0625	0.0625	0.0000	0.2500	0.2500
5/11-B	-0.5000	-0.5000	0.0312	-0.0312	-0.0312	0.0312	0.0000	0.2500	0.2500

Table 4.2a Filter sets of YKSK-1, 5/11-C and 5/11-B integer transforms.

Images	M1	M2	M3
lena128	122.14	145.00	147.52
lena256	70.51	79.51	81.21
barb128	120.89	138.34	140.08
barb256	103.66	111.67	111.26
boat128	183.01	220.86	219.82
boat256	125.28	138.02	138.96

Table 4.2b MSE values for six different images corresponding to the filter sets of YKSK-1, 5/11-C and 5/11-B integer transforms.

Filters	fa(2)	fa(3)	fa(5)	fa(6)	fb(2)	fb(3)	fb(5)	fb(6)
YKSK-1	-1.4538	-0.9892	0.5214	0.3814	-0.1828	-0.2366	0.6989	0.9892
(9,7)	-1.5861	-1.5861	0.8829	0.8829	-0.0530	-0.0530	0.4435	0.4435

Table 4.3a Filter sets of YKSK-1 and (9,7) integer transforms.

Images	M1	M2
lena128	117.11	135.24
lena256	67.03	75.75
barb128	117.77	130.93
barb256	100.55	108.31
boat128	176.93	204.89
boat256	119.74	131.48

Table 4.3b MSE values for six different images corresponding to the filter sets of YKSK-1, and (9,7) integer transforms.

4.8 YKSK-2 Transform

We propose here another new generalized subband transform known as YKSK-2 transform. The concept of decomposition and reconstruction schemes is similar to that of YKSK-1 transform. The main difference here lies in the information carrier term. Unlike in the case of YKSK-1 transform where only a single term of the signal constitutes the information carrier part in each decomposition equation, here two terms of the signal constitute the information carrier part in each decomposition equation. Hence the name YKSK-2 transform. Thus, the first two terms of each decomposition equation constitute the information carrier while the third term is the modifier. The reconstruction of the signal from the decomposed components is a little different from that in the case of YKSK-1 transform. But it is simple, just solving for the two information carriers from the two decomposition equations. Reconstruction is possible for arbitrary filter coefficients provided the four coefficients associated with the information carriers avoid a certain division by zero, as we will see later. Like in YKSK-1 transform, only the decomposition filters are required for both decomposition and reconstruction processes, and we can decompose a signal into all possible combinations, namely, into two approximations or into two details or into one approximation and one detail, by choosing appropriate modifiers. The length of a decomposed component in the case of a 1-D signal is always the ceiling of the half of the length of the signal irrespective of the length of the decomposition filters used. In other words, the length of a decomposed component is independent of the length of the decomposition filters. Another interesting feature of this transform is that we can easily make the values of the decomposed components integers even if we use arbitrary floating point numbers as the filter coefficients for decomposition.

4.8.1 Decomposition scheme of YKSK-2 transform

It is a subband decomposition scheme similar to that of YKSK-1 transform. Here also the decomposition and the reconstruction processes involve two main entities: the information carriers and the modifiers. But the decomposition and the reconstruction schemes are more general than those of YKSK-1 transform. Unlike in YKSK-1 transform where a decomposition equation contains only one term for the information carrier, here a decomposition equation contains two information carrier terms. The modifiers have a great influence to make efficacious change on the characteristics of the decomposed components.

The information carriers retain the necessary information for signal reconstruction. They are used for the reconstruction of the signal from its decomposed components.

Let $\mathbf{b} = [b_1, b_2, \cdots, b_m]$ be a discrete signal and $\mathbf{f}_a$ and $\mathbf{f}_b$ be any two arbitrarily chosen decomposition filters. Then we can decompose $\mathbf{b}$ into two components in the following way.

$$\left.\begin{aligned}\mathbf{A}(n) &= \mathbf{f}_a(1)\mathbf{b}(2n) + \mathbf{f}_a(2)\mathbf{b}(2n-1) + M_a(n)\\ \mathbf{B}(n) &= \mathbf{f}_b(1)\mathbf{b}(2n) + \mathbf{f}_b(2)\mathbf{b}(2n-1) + M_b(n)\end{aligned}\right\} \cdots\cdots(49)$$

where $n = 1,2,\cdots,\lceil m/2 \rceil$ and $\mathbf{A}(n)$ and $\mathbf{B}(n)$ are zeros outside the range $1 \le n \le \lceil m/2 \rceil$.

The terms $M_a(n)$ and $M_b(n)$ are known as modifiers of the decomposed components. The first two terms on the right hand side of each decomposition equation in equation (49) are the information carriers. In other words, the terms other than the modifiers constitute the information carriers. Modifiers have great influence on the characteristics of a decomposed component. By choosing the modifiers in different ways we can decompose a signal into different types of components. The modifiers may be a function of n or a constant. The decomposition filters may be of any length, but only the first two of their elements are meant for the information carriers. The others are used for the modifiers. There is also no restriction on the choice of the coefficients of the decomposition filters, except the ones to be used for the information carrier terms. The length of the decomposed components is independent of the length of the filter. The length of each decomposed component is equal to the ceiling of the half of the length of the input signal. The decomposition process can be applied successively in one or both of the decomposed components at each subsequent level of decomposition depending on whether one wants multilevel half decomposition or multilevel full decomposition, as in the case of ISITRA.

4.8.2 Reconstruction scheme of YKSK-2 transform

YKSK transform is a perfect reconstruction transform even if the filters are chosen arbitrarily. The reconstruction scheme is very simple and involves no complex mathematical operations. It is just solving for the even and odd components of the information carriers from the decomposed components. The decomposition equation (49) consists of two equations with two unknowns, namely, $\mathbf{b}(2n)$ and $\mathbf{b}(2n-1)$. Hence, solving for these two unknowns is always possible if we somehow know the values of the modifiers as the

decomposed components and the decomposition filters are known to us. That is why we can always get a perfect reconstruction of a signal from its decomposed components in YKSK-1 transform. There are three main reconstruction schemes by which we can reconstruct the original signal. These schemes are given below. We can get the original signal from its decomposed components using the decomposition filters from the following relations.

$$\left.\begin{aligned}\mathbf{b}(2n-1) &= D^{-1}\{\mathbf{f}_b(1)(\mathbf{A}(n)-M_a(n))-\mathbf{f}_a(1)(\mathbf{B}(b)-M_b(n))\}\\ \mathbf{b}(2n) &= -D^{-1}\{\mathbf{f}_b(2)(\mathbf{A}(n)-M_a(n))-\mathbf{f}_a(2)(\mathbf{B}(n)-M_b(n)\}\end{aligned}\right\} \cdots\cdots(50)$$

where $D=\mathbf{f}_a(1)\mathbf{f}_b(2)-\mathbf{f}_a(2)\mathbf{f}_b(1)\neq 0$ $\cdots\cdots\cdots(51)$.

Equation (50) is the generalized reconstruction scheme of the subband decomposition scheme of YKSK-2. And equation (51) is the only condition to be satisfied while choosing the filter coefficients for reconstruction. As equation (51) can be easily satisfied, we can say that the coefficients of the decomposition filters can be chosen quite arbitrarily.

Instead of solving $\mathbf{b}(2n)$ and $\mathbf{b}(2n-1)$ separately, we can first find one of them from one decomposition equation and then find the other from the remaining decomposition equation, as in equations (52) and (53) below.

$$\left.\begin{aligned}\mathbf{b}(2n) &= \{\mathbf{f}_b(2)(\mathbf{A}(n)-M_a(n))-\mathbf{f}_a(2)(\mathbf{B}(n)-M_b(n))\}/D\\ \mathbf{b}(2n-1) &= \left.\begin{aligned}&\{\mathbf{B}(n)-M_b(n)-\mathbf{f}_b(1)\mathbf{b}(2n)\}/\mathbf{f}\\ &or\\ &\{\mathbf{A}(n)-M_a(n)-\mathbf{f}_a(1)\mathbf{b}(2n)\}/\mathbf{f}_a(2)\end{aligned}\right\}\end{aligned}\right\} \cdots\cdots\cdots(52)$$

$$\left.\begin{aligned}\mathbf{b}(2n-1) &= \{\mathbf{f}_b(1)(\mathbf{A}(n)-M_a(n))-\mathbf{f}_a(1)(\mathbf{B}(b)-M_b(n))\}/D1\\ \mathbf{b}(2n) &= \left.\begin{aligned}&\{\mathbf{B}(n)-M_b(n)-\mathbf{f}_b(2)\mathbf{b}(2n-1)\}/\mathbf{f}_b(1)\\ &or\\ &\{\mathbf{A}(n)-M_a(n)-\mathbf{f}_a(2)\mathbf{b}(2n-1)\}/\mathbf{f}_a(1)\end{aligned}\right\}\end{aligned}\right\} \cdots\cdots\cdots(53)$$

where $D1=-D$.

We can reconstruct the original signal by any of the above three reconstruction schemes given by equations (50), (52) and (53). Note that in the above reconstruction equations, we do not use any reconstruction filters explicitly. We use the same decomposition filters instead. As the perfect reconstruction is possible for almost any arbitrarily chosen filter coefficients, we need not be much concerned about the choice of the filter coefficients for

perfect reconstruction. But we need a careful choice of filter coefficients which will give us the desired results.

When $\mathbf{f}_a(1) = 0, \mathbf{f}_b(2) = 0$, the decomposition equation (49) becomes the decomposition equation of YKSK-1 (i.e., equation (1)) as shown below.

$$\left.\begin{aligned}\mathbf{A}(n) &= \mathbf{f}_a(2)\mathbf{b}(2n-1) + M_a(n)\\ \mathbf{B}(n) &= \mathbf{f}_b(1)\mathbf{b}(2n) + M_b(n)\end{aligned}\right\} \quad \cdots\cdots\cdots(54)$$

All the above three reconstruction equations (50), (52) and (53) become

$$\left.\begin{aligned}\mathbf{b}(2n-1) &= \{\mathbf{A}(n) - M_a(n)\} / \mathbf{f}_a(2)\\ \mathbf{b}(2n) &= \{\mathbf{B}(n) - M_b(n)\} / \mathbf{f}_b(1)\end{aligned}\right\} \cdots\cdots(55)$$

Similarly, when $\mathbf{f}_a(2) = 0, \mathbf{f}_b(1) = 0$, the decomposition equation (49) again becomes the decomposition equation of YKSK-1 transform.

$$\left.\begin{aligned}\mathbf{A}(n) &= \mathbf{f}_a(1)\mathbf{b}(2n) + M_a(n)\\ \mathbf{B}(n) &= \mathbf{f}_b(2)\mathbf{b}(2n-1) + M_b(n)\end{aligned}\right\} \quad \cdots\cdots\cdots(56)$$

The three reconstruction equations (50), (52) and (53) become

$$\left.\begin{aligned}\mathbf{b}(2n-1) &= \{\mathbf{B}(n) - M_b(n)\} / \mathbf{f}_b(2)\\ \mathbf{b}(2n) &= \{\mathbf{A}(n) - M_a(n)\} / \mathbf{f}_a(1)\end{aligned}\right\} \cdots\cdots(57)$$

Thus, YKSK-1 transform is a special case of YKSK-2 transform.

4.9 Types of YKSK-2 Transform

Depending on the number of terms of the signal involved in the computation of a term of the decomposed components, YKSK-2 transform, like YKSK-1 transform, is classified into three types, namely, fixed length type, semi-infinite length type and infinite length type. They are described below. Let, like in YKSK-1 transform, $N_a(n)$ be the number of terms of the signal $\mathbf{b}$ involved in the computation of a term of the decomposed component $\mathbf{A}(n)$ and $N_b(n)$ be the number of terms of $\mathbf{b}$ involved in the computation of a term of the decomposed component $\mathbf{B}(n)$.

4.9.1 Fixed length type YKSK-2 transform

In this decomposition scheme, we choose modifiers in such a way that both $N_a(n)$ and $N_b(n)$ are fixed. That is, neither of them depends on *n*. Like in YKSK-1 transform, fixed length type is classified into 5 types, namely, equal and even fixed length type, unequal and even fixed length type, equal and odd fixed length type, unequal and odd fixed length type and mixed fixed length type.

Equal and even fixed length type: In this type, N_a and N_b are even and equal. The modifiers in this case have the following form.

$$\left.\begin{aligned} M_a(n) &= \sum_{i=3}^{La} \mathbf{f}_a(i)\mathbf{b}(2n + c_1 \pm i) \\ M_b(n) &= \sum_{i=3}^{Lb} \mathbf{f}_b(i)\mathbf{b}(2n + c_2 \pm i) \end{aligned}\right\} \quad \cdots\cdots\cdots(58)$$

Or

$$\left.\begin{aligned} M_a(n) &= \sum_{i=3}^{La} \mathbf{f}_a(i)\mathbf{b}(2n + c_1 \pm i) \\ M_b(n) &= \sum_{i=3}^{Lb} \mathbf{f}_b(i)\mathbf{A}(n + c_2 \pm i) \end{aligned}\right\} \quad \cdots\cdots\cdots(59)$$

where c_1, c_2 are integer constants (zero, positive or negative) for adjusting the indices of the arrays. This type of YKSK-2 transform can be achieved in a number of ways by appropriately choosing the modifiers. Two simple cases of this type are :

Case-1: When the decomposition filters have length 2 and the modifier terms vanish.

Let $\mathbf{f}_a = [l_1, l_2]$ and $\mathbf{f}_b = [k_1, k_2]$ be the decomposition filters such that $D \neq 0$.

When $L_a = L_b = 2$, the modifier terms vanish, i.e., $M_a(n) = 0$; $M_b(n) = 0$. Then the decomposition equation (49) becomes

$$\left.\begin{aligned} \mathbf{A}(n) &= \mathbf{f}_a(1)\mathbf{b}(2n) + \mathbf{f}_a(2)\mathbf{b}(2n-1) \\ \mathbf{B}(n) &= \mathbf{f}_b(1)\mathbf{b}(2n) + \mathbf{f}_b(2)\mathbf{b}(2n-1) \end{aligned}\right\} \cdots\cdots(60)$$

From equation (58), we see that $N_a = N_b = 2$. This is the case of equal and even fixed length type. The corresponding decomposition scheme becomes similar to that of

decomposition scheme of ISITRA-2 of length 2. If $l_1 = l_2 = 1/\sqrt{2}$ and $k_1 = -k_2 = 1/\sqrt{2}$, the decomposed components obtained would be the same as those of decomposed components using DB1 or Haar's wavelet filters. The reconstruction scheme here is obtained from equation (50) by putting $M_a(n) = 0;\ M_b(n) = 0$, i.e.,

$$\left.\begin{aligned}\mathbf{b}(2n-1) &= D^{-1}\{\mathbf{f}_b(1)\mathbf{A}(n) - \mathbf{f}_a(1)\mathbf{B}(n)\}\\ \mathbf{b}(2n) &= -D^{-1}\{\mathbf{f}_b(2)\mathbf{A}(n) - \mathbf{f}_a(2)\mathbf{B}(n)\}\end{aligned}\right\} \cdots\cdots(61)$$

Modifiers have no role either in the decomposition and the reconstruction process and hence the characteristics of the decomposed components are solely dependent on the decomposition filters.

Case-2: When $L_a = L_b = L$ (an even number greater than 2) in equation (58), we can write the modifier terms as

$$\left.\begin{aligned}M_a(n) &= \sum_{i=3}^{La}\mathbf{f}_a(i)\mathbf{b}(2n+1-i)\\ M_b(n) &= \sum_{i=3}^{Lb}\mathbf{f}_b(i)\mathbf{b}(2n+1-i)\end{aligned}\right\} \cdots\cdots(62)$$

We can get the decomposed components by substituting the modifier terms in the decomposition equation (49). Then $N_a = N_b = L$ and the decomposition equations become

$$\left.\begin{aligned}\mathbf{A}(n) &= \sum_{i=1}^{L}\mathbf{f}_a(i)\mathbf{b}(2n+1-i)\\ \mathbf{B}(n) &= \sum_{i=1}^{L}\mathbf{f}_b(i)\mathbf{b}(2n+1-i)\end{aligned}\right\} \cdots\cdots(63)$$

where $n = 1,2,3,\cdots,\lceil m/2 \rceil$

The decomposition equations in (63) here are the same as those of ISITRA or discrete wavelet transform. The only difference is that in the case of ISITRA or DWT, the range of n is: $n = 1,2,3,\cdots(\lceil m/2 \rceil + L/2 - 1)$. The decomposed components will have similar characteristics as those of the decomposed components using wavelets or ISITRA if we use the same set of decomposition filters. The difference is apparent in the trailing decomposed components. The length of a decomposed component here is independent of the length of the filter L and is always equal to the ceiling of the half of the length of the signal. But the

decomposed components using ISITRA or wavelet filters will be longer than the decomposed components using YKSK-2 transform. We can easily reconstruct the signal from its decomposed components by substituting the respective modifiers in equation (50).

Unequal and even fixed length type: Here N_a and N_b are even but unequal. For example, if $L_a = 4,\ L_b = 6,$ equation (62) provides this type.

Equal and odd fixed length type: Here N_a and N_b are odd and equal.

Unequal and odd fixed length type: Here N_a and N_b are odd but unequal. For example, if $L_a = 3,\ L_b = 5,$ equation (62) provides this type.

Mixed fixed length type: Here one of N_a and N_b is even and the other odd. For example, if $L_a = 5,\ L_b = 4,$ equation (62) provides this type.

4.9.2 Semi-infinite length type YKSK-2 transform

In this case, one of $N_a(n)$ and $N_b(n)$ is fixed while the other increases with *n*. For example, consider the following modifiers.

$$\left.\begin{aligned} M_a(n) &= \sum_{i=3}^{La} \mathbf{f}_a(i)\mathbf{b}(2n + c_1 \pm i) \\ M_b(n) &= \sum_{i=3}^{Lb} \mathbf{f}_b(i)\mathbf{B}(2n + c_2 \pm i) \end{aligned}\right\} \quad \cdots\cdots\cdots(64)$$

where c_1, c_2 are integer constants (zero, positive or negative).

There are three main types of semi-infinite length type, even semi infinite length type, odd semi- infinite length type and mixed semi-infinite length type.

Even semi-infinite length type:

$$\left.\begin{aligned} \mathbf{A}(n) &= \mathbf{f}_a(1)\mathbf{b}(2n) + \mathbf{f}_a(2)\mathbf{b}(2n-1) + \mathbf{f}_a(3)\mathbf{b}(2n-2) + \mathbf{f}_a(4)\mathbf{b}(2n-3) \\ \mathbf{B}(n) &= \mathbf{f}_b(1)\mathbf{b}(2n) + \mathbf{f}_b(2)\mathbf{b}(2n-1) + \mathbf{f}_b(3)\mathbf{B}(n-1) \end{aligned}\right\} \quad \cdots\cdots\cdots(65)$$

Odd semi-infinite length type:

$$\left.\begin{aligned}&\mathbf{A}(n)=\mathbf{f}_a(1)\mathbf{b}(2n)+\mathbf{f}_a(2)\mathbf{b}(2n-1)+\mathbf{f}_a(3)\mathbf{b}(2n-2)\\&\mathbf{B}(1)=\mathbf{f}_b(1)\mathbf{b}(2)\\&\mathbf{B}(n)=\mathbf{f}_b(1)\mathbf{b}(2n)+\mathbf{f}_b(2)\mathbf{b}(2n-1)+\mathbf{f}_b(3)\mathbf{B}(n-1),\ \forall\ 2\le n\le\lceil m/2\rceil\end{aligned}\right\}\quad\cdots\cdots\cdots(66)$$

Mixed semi-infinite length type:

$$\left.\begin{aligned}&\mathbf{A}(n)=\mathbf{f}_a(1)\mathbf{b}(2n)+\mathbf{f}_a(2)\mathbf{b}(2n-1)+\mathbf{f}_a(3)\mathbf{b}(2n-2)\\&\mathbf{B}(n)=\mathbf{f}_b(1)\mathbf{b}(2n)+\mathbf{f}_b(2)\mathbf{b}(2n-1)+\mathbf{f}_b(3)\mathbf{B}(n-1)\end{aligned}\right\}\quad\cdots\cdots\cdots(67)$$

4.9.3 Infinite length type YKSK-2 transform

In this type, as in YKSK-1 transform, both $N_a(n)$ and $N_b(n)$ increase with n until the last terms of the decomposed components are computed. Like in YKSK-1 transform, here also there are five different infinite length types, namely, equal and even infinite length type, equal and odd infinite length type, unequal and even infinite length type, unequal and odd infinite length type and mixed infinite length type. They are described below.

Equal and even infinite length type: In general, we get equal and even infinite length type if we use the modifiers $M_a(n)=\mathbf{f}_a(3)\mathbf{A}(n-1)$ and $M_b(n)=\mathbf{f}_b(3)\mathbf{B}(n-1)$ in the decomposition equation (49). Similarly, a reconstruction is obtained from the decomposed components from equation (50) by appropriately replacing the modifier terms with the ones used in the decomposition.

Unequal and even infinite length type: Here $N_a(n)$ and $N_b(n)$ are unequal but even and increase with n.

$$\left.\begin{aligned}&\mathbf{A}(n)=\mathbf{f}_a(1)\mathbf{b}(2n-1)+\mathbf{f}_a(2)\mathbf{b}(2n)+\mathbf{f}_a(3)\mathbf{A}(n-1)\\&\mathbf{B}(n)=\mathbf{f}_b(1)\mathbf{b}(2n-1)+\mathbf{f}_b(2)\mathbf{b}(2n)+\mathbf{f}_b(3)\mathbf{A}(n+1)\end{aligned}\right\}\quad\cdots\cdots\cdots(68)$$

Here $N_a(n)$ increases as 2, 4, 6, … for $n=1,2,3\ldots.$ But $N_b(n)$ increases as 4, 6, 8, ... for $n=1,2,3\ldots.$

Equal and odd infinite length type:

$$\left.\begin{aligned}&\mathbf{A}(1)=\mathbf{f}_a(1)\mathbf{b}(2)\\&\mathbf{B}(1)=\mathbf{f}_a(1)\mathbf{b}(2)\end{aligned}\right\}$$

$$\left.\begin{aligned}\mathbf{A}(n) &= \mathbf{f}_a(1)\mathbf{b}(2n) + \mathbf{f}_a(2)\mathbf{b}(2n-1) + \mathbf{A}(n-1)\\ \mathbf{B}(n) &= \mathbf{f}_a(1)\mathbf{b}(2n) + \mathbf{f}_a(2)\mathbf{b}(2n-1) + \mathbf{B}(n-1)\end{aligned}\right\} \quad \cdots\cdots\cdots(69)$$

for $2 \le n \le \lceil m/2 \rceil$

Unequal and odd infinite length type:

$$\left.\begin{aligned}\mathbf{A}(1) &= \mathbf{f}_a(1)\mathbf{b}(2)\\ \mathbf{B}(1) &= \mathbf{f}_a(1)\mathbf{b}(2)\end{aligned}\right\}$$

$$\left.\begin{aligned}\mathbf{A}(n) &= \mathbf{f}_a(1)\mathbf{b}(2n) + \mathbf{f}_a(2)\mathbf{b}(2n-1) + \mathbf{A}(n-1)\\ \mathbf{B}(n) &= \mathbf{f}_a(1)\mathbf{b}(2n) + \mathbf{f}_a(2)\mathbf{b}(2n-1) + \mathbf{A}(n+1)\end{aligned}\right\} \quad \cdots\cdots\cdots(70)$$

for $2 \le n \le \lceil m/2 \rceil$

Mixed infinite length type: In this type, the values of $N_a(n)$ and $N_b(n)$ are neither all odd numbers nor all even numbers. In other words, this is an infinite length type which is not any of the above four types. For example, we get this type if we use equation (68) for computing $\mathbf{A}(n)$ and use equation (69) for computing $\mathbf{B}(n)$.

4.10 Multi-Stage Decomposition of YKSK-2 Transform

Let us first consider for 2-stage decomposition and reconstruction scheme. Here stage means the number of decomposition steps to get the decomposed components.

Let the decomposed components in stage-1 are denoted by $\mathbf{A}^1, \mathbf{B}^1$, the modifiers by M_a^1, M_b^1 and the lengths of the decomposition filters by L_a^1, L_b^1. And let the decomposed components in stage-2 be denoted by $\mathbf{A}^2, \mathbf{B}^2$, the modifiers by M_a^2, M_b^2 and the lengths of the decomposition filters by L_a^2, L_b^2. We can write the decomposition equations in the two stages as

Stage-1: Decomposition equations:

$$\left.\begin{aligned}\mathbf{A}^1(n) &= \mathbf{f}_a(1)\mathbf{b}(2n) + \mathbf{f}_a(2)\mathbf{b}(2n-1) + M_a^1(n)\\ \mathbf{B}^1(n) &= \mathbf{f}_b(1)\mathbf{b}(2n) + \mathbf{f}_b(2)\mathbf{b}(2n-1) + M_b^1(n)\end{aligned}\right\} \cdots\cdots(71)$$

If $\mathbf{f}_a(1) = 0, \mathbf{f}_b(2) = 0$, then it becomes YKSK-1 transform.

Stage-2: Decomposition equations:

$$\left.\begin{aligned}\mathbf{A}^2(n) &= \mathbf{f}_a(L_a^1+1)\mathbf{A}^1(n)+\mathbf{f}_a(L_a^1+2)\mathbf{B}^1(n)+M_a^2(n)\\ \mathbf{B}^2(n) &= \mathbf{f}_b(L_b^1+1)\mathbf{A}^1(n)+\mathbf{f}_b(L_b^1+2)\mathbf{B}^1(n)+M_b^2(n)\end{aligned}\right\}\cdots\cdots(72)$$

Multi-stage decomposition is the decomposition of a signal through two or more intermediate stages.

We can reconstruct the original signal from its decomposed components $\mathbf{A}^2, \mathbf{B}^2$ in two stages in the following way.

Stage-1: Reconstruction equations:

$$\left.\begin{aligned}\mathbf{A}^1(n) &= D1^{-1}\{\mathbf{f}_b(r_b)\mathbf{A}^2(n)-\mathbf{f}_a(r_a)\mathbf{B}^2(n)+\mathbf{f}_a(r_a)M_b^2(n)-\mathbf{f}_b(r_b)M_a^2(n)\}\\ \mathbf{B}^1(n) &= -D1^{-1}\{\mathbf{f}_b(r_b^1)\mathbf{A}^2(n)-\mathbf{f}_a(r_a^1)\mathbf{B}^2(n)+\mathbf{f}_a(r_a^1)M_b^2(n)-\mathbf{f}_b(r_b^1)M_a^2(n)\}\end{aligned}\right\}\cdots\cdots(73)$$

where $D1=\mathbf{f}_a(r_a)\mathbf{f}_b(r_b^1)-\mathbf{f}_a(r_a^1).\mathbf{f}_b(r_b)\neq 0$, $r_a=L_a^1+1$, $r_a^1=r_a+1$, $r_b=L_b^1+1$, $r_b^1=r_b+1$.

Stage-2: Reconstruction equations:

$$\left.\begin{aligned}\mathbf{b}(2n-1) &= D^{-1}\{\mathbf{f}_b(1)\mathbf{A}^1(n)-\mathbf{f}_a(1)\mathbf{B}^1(n)+\mathbf{f}_a(1)M_b^1(n)-\mathbf{f}_b(1)M_a^1(n)\}\\ \mathbf{b}(2n) &= -D^{-1}\{\mathbf{f}_b(2)\mathbf{A}^1(n)-\mathbf{f}_a(2)\mathbf{B}^1(n)+\mathbf{f}_a(2)M_b^1(n)-\mathbf{f}_b(2)M_a^1(n)\}\end{aligned}\right\}\cdots\cdots(74)$$

where $D=\mathbf{f}_a(1)\mathbf{f}_b(2)-\mathbf{f}_a(2)\mathbf{f}_b(1)\neq 0$.

Thus, we can reconstruct the original signal from its two stage decomposed components. This can be easily extended to multi-stage decomposition scheme as in YKSK-1 transform. The number of stages of decomposition for the two decomposed components may not be the same.

4.11 Integer Version of YKSK-2 Transform

Suppose our input signal is an integer-valued signal. Then, as in YKSK-1 transform, we can make the decomposed components integers. The simplest way is just to make the coefficients of the filter associated with the information carriers integers and making the respective modifiers integers by flooring, ceiling or rounding off to the nearest integer. There is no loss of information due to making the modifier values integers, as opposed to the popular belief that the flooring function causes information loss and hence that value 0.5 is added as a compensation for it. Then the decomposition equation (49) becomes

4.11.1 First integer version of YKSK-2 transform

$$\left.\begin{aligned}\mathbf{A}(n) &= \mathbf{f}_a(1)\mathbf{b}(2n) + \mathbf{f}_a(2)\mathbf{b}(2n-1) + \lfloor M_a(n) \rfloor \rfloor \\ \mathbf{B}(n) &= \mathbf{f}_b(1)\mathbf{b}(2n) + \mathbf{f}_b(2)\mathbf{b}(2n-1) + \lfloor M_b(n) \rfloor \rfloor\end{aligned}\right\} \cdots\cdots(75)$$

For simplicity, suppose $\mathbf{f}_a(1) = 1, \mathbf{f}_a(2) = 1$ and $\mathbf{f}_b(1) = 1, \mathbf{f}_b(2) = -1$, then, we get

$$\left.\begin{aligned}\mathbf{A}(n) &= \mathbf{b}(2n) + \mathbf{b}(2n-1) + \lfloor M_a(n) \rfloor \rfloor \\ \mathbf{B}(n) &= \mathbf{b}(2n) - \mathbf{b}(2n-1) + \lfloor M_b(n) \rfloor \rfloor\end{aligned}\right\} \cdots\cdots\cdots(76)$$

$$\left.\begin{aligned}\mathbf{b}(2n-1) &= 0.5\{\mathbf{A}(n) - \mathbf{B}(n) - (\lfloor M_a(n) \rfloor - \lfloor M_b(n) \rfloor)\} \\ \mathbf{b}(2n) &= 0.5\{\mathbf{A}(n) + \mathbf{B}(n) - (\lfloor M_a(n) \rfloor + \lfloor M_b(n) \rfloor)\}\end{aligned}\right\} \cdots\cdots\cdots(77)$$

We can even combine different integer transformation operators and still get perfect reconstruction. If we combine ceiling and rounding off operators in equation (76), we get

$$\left.\begin{aligned}\mathbf{A}(n) &= \mathbf{b}(2n) + \mathbf{b}(2n-1) + \langle M_a(n) \rangle \rangle \\ \mathbf{B}(n) &= \mathbf{b}(2n) - \mathbf{b}(2n-1) + \lceil M_b(n) \rceil \rceil\end{aligned}\right\} \cdots\cdots\cdots(78)$$

In the above decomposition equation, the symbol $\langle x \rangle$ denotes the round off integer value of a real number x to the nearest integer. Then the reconstruction equations of (77) become

$$\left.\begin{aligned}\mathbf{b}(2n-1) &= 0.5\{\mathbf{A}(n) - \mathbf{B}(n) - (\langle M_a(n) \rangle - \lceil M_b(n) \rceil)\} \\ \mathbf{b}(2n) &= 0.5\{\mathbf{A}(n) + \mathbf{B}(n) - (\langle M_a(n) \rangle + \lceil M_b(n) \rceil)\}\end{aligned}\right\} \cdots\cdots\cdots(79)$$

The decomposition scheme of equations (76) and (78) have the high chance that the values of $\mathbf{A}$ are higher than 2 times of the maximum value of the signal $\mathbf{b}$, which may not be desirable for some applications particularly for compression purpose. We can solve this problem by modifying the way of making the decomposed components integers.

4.11.2 Second integer version of YKSK-2 transform

Let the decomposition scheme be

$$\left.\begin{aligned}\mathbf{A}(n) &= \lfloor \mathbf{f}_a(1)\mathbf{b}(2n) + \mathbf{f}_a(2)\mathbf{b}(2n-1) \rfloor + \lfloor M_a(n) \rfloor \rfloor \\ \mathbf{B}(n) &= \lfloor \mathbf{f}_b(1)\mathbf{b}(2n) + \mathbf{f}_b(2)\mathbf{b}(2n-1) \rfloor + \lfloor M_b(n) \rfloor \rfloor\end{aligned}\right\} \cdots\cdots\cdots(80)$$

We know from the above decomposition equation that the decomposed components will have integer values, regardless of the values of the decomposition filters. But we will not in general get perfect reconstruction from them if all the decomposition filters are floating point numbers.

If $\mathbf{f}_b(1)$ and $\mathbf{f}_b(2)$ are integers, we can write

$$\left.\begin{aligned}\mathbf{A}(n) &= \lfloor \mathbf{f}_a(1).\mathbf{b}(2n) + \mathbf{f}_a(2).\mathbf{b}(2n-1) \rfloor + \lfloor M_a(n) \rfloor \\ \mathbf{B}(n) &= \mathbf{f}_b(1).\mathbf{b}(2n) + \mathbf{f}_b(2).\mathbf{b}(2n-1) + \lfloor M_b(n) \rfloor\end{aligned}\right\} \quad \cdots\cdots\cdots(81)$$

Writing $\mathbf{b}(2n)$ in terms of $\mathbf{b}(2n-1)$ from the second equation above and then substituting $\mathbf{b}(2n)$ in the first equation, we get

$$\mathbf{A}(n) = \left\lfloor \frac{\mathbf{f}_a(1)\{\mathbf{B}(n) - \lfloor M_b(n) \rfloor\}}{\mathbf{f}_b(1)} + k\mathbf{b}(2n-1) \right\rfloor + \lfloor M_a(n) \rfloor \quad \cdots\cdots\cdots(82)$$

where $k = \dfrac{\mathbf{f}_a(2)\mathbf{f}_b(1) - \mathbf{f}_a(1)\mathbf{f}_b(2)}{\mathbf{f}_b(1)}$.

If the coefficient of $\mathbf{b}(2n-1)$, i.e., k, is an integer, then the above equation becomes

$$\mathbf{A}(n) = \left\lfloor \frac{\mathbf{f}_a(1)\{\mathbf{B}(n) - \lfloor M_b(n) \rfloor\}}{\mathbf{f}_b(1)} \right\rfloor + k\mathbf{b}(2n-1) + \lfloor M_a(n) \rfloor$$

And we get,

$$\mathbf{b}(2n-1) = \{\mathbf{A}(n) - \lfloor M_a(n) \rfloor - \left\lfloor \frac{\mathbf{f}_a(1)\{\mathbf{B}(n) - \lfloor M_b(n) \rfloor\}}{\mathbf{f}_b(1)} \right\rfloor\}/k$$

Thus, for real numbers $\mathbf{f}_a(1), \mathbf{f}_a(2)$ and integers $\mathbf{f}_b(1), \mathbf{f}_b(2)$ for which k is an integer, we get perfect reconstruction of the original integer signal from its integer decomposed components using the following reconstruction equations.

$$\left.\begin{aligned}\mathbf{b}(2n-1) &= \{\mathbf{A}(n) - \lfloor M_a(n) \rfloor - \left\lfloor \frac{\mathbf{f}_a(1)\{\mathbf{B}(n) - \lfloor M_b(n) \rfloor\}}{\mathbf{f}_b(1)} \right\rfloor\}/k \\ \mathbf{b}(2n) &= \{\mathbf{B}(n) - \lfloor M_b(n) \rfloor - \mathbf{f}_b(2)\mathbf{b}(2n-1)\}/\mathbf{f}_b(1)\end{aligned}\right\} \quad \cdots\cdots\cdots(83)$$

We consider the following special cases in terms of the choice of values of $\mathbf{f}_b(1)$ and $\mathbf{f}_b(2)$.

Case-1: $\mathbf{f}_b(1) = \mathbf{f}_b(2) \neq 0$. That is, $\mathbf{f}_a(2) - \mathbf{f}_a(1)$ is an integer. Two such examples are given below.

Example-1: $\mathbf{f}_a(1) = -0.25, \mathbf{f}_a(2) = 0.75$. Then our decomposition equation (81) becomes

$$\left.\begin{aligned}\mathbf{A}(n) &= \lfloor 0.75\mathbf{b}(2n-1) - 0.25\mathbf{b}(2n) \rfloor + \lfloor M_a(n) \rfloor \\ \mathbf{B}(n) &= \mathbf{b}(2n) + \mathbf{b}(2n-1) + \lfloor M_b(n) \rfloor\end{aligned}\right\} \quad \cdots\cdots\cdots(84)$$

And the corresponding reconstruction equation (83) becomes

$$\left.\begin{aligned}\mathbf{b}(2n-1) &= \mathbf{A}(n) - \lfloor M_a(n) \rfloor + \lfloor 0.25(\mathbf{B}(n) - \lfloor M_b(n) \rfloor) \rfloor \\ \mathbf{b}(2n) &= \mathbf{B}(n) - \lfloor M_b(n) \rfloor - \mathbf{b}(2n-1)\end{aligned}\right\} \quad \cdots\cdots(85)$$

Example-2: $\mathbf{f}_a(1) = -0.5, \mathbf{f}_a(2) = 0.5$. Then the decomposition and reconstruction equations become

$$\left.\begin{aligned}\mathbf{A}(n) &= \lfloor 0.5\mathbf{b}(2n-1) - 0.5\mathbf{b}(2n) \rfloor + \lfloor M_a(n) \rfloor \\ \mathbf{B}(n) &= \mathbf{b}(2n) + \mathbf{b}(2n-1) + \lfloor M_b(n) \rfloor\end{aligned}\right\} \quad \cdots\cdots\cdots(86)$$

$$\left.\begin{aligned}\mathbf{b}(2n-1) &= \mathbf{A}(n) - \lfloor M_a(n) \rfloor + \lfloor 0.5(\mathbf{B}(n) - \lfloor M_b(n) \rfloor) \rfloor \\ \mathbf{b}(2n) &= \mathbf{B}(n) - \lfloor M_b(n) \rfloor - \mathbf{b}(2n-1)\end{aligned}\right\} \quad \cdots\cdots(87)$$

Case-2: $\mathbf{f}_b(1) = -\mathbf{f}_b(2) \neq 0$. That is, $\mathbf{f}_a(2) + \mathbf{f}_a(1)$ is an integer. Two such examples along with the decomposition and reconstruction equations are given below.

Example-1: $\mathbf{f}_a(1) = 0.25$, $\mathbf{f}_a(2) = 0.75$,

$$\left.\begin{aligned}\mathbf{A}(n) &= \lfloor 0.25\mathbf{b}(2n) + 0.75\mathbf{b}(2n-1) \rfloor + \lfloor M_a(n) \rfloor \\ \mathbf{B}(n) &= \mathbf{b}(2n) - \mathbf{b}(2n-1) + \lfloor M_b(n) \rfloor\end{aligned}\right\} \quad \cdots\cdots\cdots(88)$$

$$\left.\begin{aligned}\mathbf{b}(2n-1) &= \mathbf{A}(n) - \lfloor M_a(n) \rfloor - \lfloor 0.25(\mathbf{B}(n) - \lfloor M_b(n) \rfloor) \rfloor \\ \mathbf{b}(2n) &= \mathbf{B}(n) - \lfloor M_b(n) \rfloor + \mathbf{b}(2n-1)\end{aligned}\right\} \quad \cdots\cdots(89)$$

Example-2: $\mathbf{f}_a(1) = 0.5$, $\mathbf{f}_a(2) = 0.5$.

$$\left.\begin{aligned}\mathbf{A}(n) &= \lfloor 0.5(\mathbf{b}(2n) + \mathbf{b}(2n-1)) \rfloor + \lfloor M_a(n) \rfloor \\ \mathbf{B}(n) &= \mathbf{b}(2n) - \mathbf{b}(2n-1) + \lfloor M_b(n) \rfloor\end{aligned}\right\} \quad \cdots\cdots\cdots(90)$$

$$\left.\begin{aligned}\mathbf{b}(2n-1) &= \mathbf{A}(n) - \lfloor M_a(n) \rfloor - \lfloor 0.5(\mathbf{B}(n) - \lfloor M_b(n) \rfloor) \rfloor \\ \mathbf{b}(2n) &= \mathbf{B}(n) + \mathbf{b}(2n-1) - \lfloor M_b(n) \rfloor\end{aligned}\right\} \quad \cdots\cdots\cdots(91)$$

In this way, we can get different integer transforms by choosing different values of $\mathbf{f}_a(1), \mathbf{f}_a(2), \mathbf{f}_b(1), \mathbf{f}_b(2)$ such that k is an integer.

In equation (83), we first reconstruct $\mathbf{b}(2n-1)$ and then reconstruct $\mathbf{b}(2n)$. The order is important as the computation of $\mathbf{b}(2n)$ requires the value of $\mathbf{b}(2n-1)$. Alternatively, we can first reconstruct $\mathbf{b}(2n)$ and then reconstruct $\mathbf{b}(2n-1)$ in the following manner.

$$\left.\begin{aligned}\mathbf{b}(2n) &= \frac{\mathbf{f}_b(2)}{\mathbf{f}_b(1).k}\{\lfloor M_a(n)\rfloor - \mathbf{A}(n) + \left\lfloor \frac{\mathbf{f}_a(2)\{\mathbf{B}(n) - \lfloor M_b(n)\rfloor\}}{\mathbf{f}_b(2)} \right\rfloor\}\\ \mathbf{b}(2n-1) &= \{\mathbf{B}(n) - \lfloor M_b(n)\rfloor - \mathbf{f}_b(1)\mathbf{b}(2n)\}/\mathbf{f}_b(2)\end{aligned}\right\} \quad \cdots\cdots\cdots(92)$$

provided $\frac{\mathbf{f}_b(1)}{\mathbf{f}_b(2)}k$ is an integer.

For example, reconstruction equation (91) then becomes

$$\left.\begin{aligned}\mathbf{b}(2n) &= \mathbf{A}(n) - \lfloor M_a(n)\rfloor + \lfloor 0.5(\mathbf{B}(n) - \lfloor M_b(n)\rfloor)\rfloor\\ \mathbf{b}(2n-1) &= \mathbf{b}(2n) - (\mathbf{B}(n) - \lfloor M_b(n)\rfloor)\end{aligned}\right\} \quad \cdots\cdots\cdots(93)$$

Instead of the floor function to get integer decomposed components we can use ceiling and rounding off to the nearest integer also. Moreover, we can mix them as

$$\left.\begin{aligned}\mathbf{A}(n) &= \lfloor 0.5(\mathbf{b}(2n) + \mathbf{b}(2n-1))\rfloor + \lceil M_a(n)\rceil\\ \mathbf{B}(n) &= \mathbf{b}(2n) - \mathbf{b}(2n-1) + \langle M_b(n)\rangle\end{aligned}\right\} \quad \cdots\cdots\cdots(94)$$

Then our reconstruction equations becomes

$$\left.\begin{aligned}\mathbf{b}(2n-1) &= \mathbf{A}(n) - \lceil M_a(n)\rceil - \lfloor 0.5(\mathbf{B}(n) - \langle M_b(n)\rangle)\rfloor\\ \mathbf{b}(2n) &= \mathbf{B}(n) + \mathbf{b}(2n-1) - \langle M_b(n)\rangle\end{aligned}\right\} \quad \cdots\cdots\cdots(95)$$

or

$$\left.\begin{aligned}\mathbf{b}(2n) &= \mathbf{A}(n) - \lceil M_a(n)\rceil + \lfloor 0.5(\mathbf{B}(n) - \langle M_b(n)\rangle)\rfloor\\ \mathbf{b}(2n-1) &= \mathbf{b}(2n) - \mathbf{B}(n) + \langle M_b(n)\rangle\end{aligned}\right\} \quad \cdots\cdots\cdots(96)$$

Thus, we can combine the integer transformation operators like flooring, ceiling, and rounding in various ways to get integer transformed decomposed components in YKSK-2. Also, one can add or subtract real number constants to or from the modifiers as in YKSK-1 without affecting perfect reconstruction. Note that when $\mathbf{f}_a(1) = 0, \mathbf{f}_b(2) = 0$, or $\mathbf{f}_a(2) = 0, \mathbf{f}_b(1) = 0$, the above integer version transform for YKSK-2 becomes the integer version transform of YKSK-1.

The same integer transformation process is extendible to the multi-stage decomposition scheme as well. For a 2-stage scheme, we have

Stage-1 decomposition equations:

$$\left.\begin{aligned}\mathbf{A}^1(n) &= \lfloor 0.5(\mathbf{b}(2n)+\mathbf{b}(2n-1))\rfloor + \lfloor M_a^1(n)\rfloor \\ \mathbf{B}^1(n) &= \mathbf{b}(2n)-\mathbf{b}(2n-1)+\lfloor M_b^1(n)\rfloor\end{aligned}\right\} \cdots\cdots(97)$$

Stage-2 decomposition equations:

$$\left.\begin{aligned}\mathbf{A}^2(n) &= \lfloor 0.5(\mathbf{A}^1(n)+\mathbf{B}^1(n))\rfloor + \lfloor M_a^2(n)\rfloor \\ \mathbf{B}^2(n) &= \mathbf{A}^1(n)-\mathbf{B}^1(n)+\lfloor M_b^2(n)\rfloor\end{aligned}\right\} \cdots\cdots(98)$$

Reconstruction starts from the reconstruction of Stage-1 decomposed components from the Stage-2 decomposed components using the following reconstruction equations.

$$\left.\begin{aligned}\mathbf{A}^1(n) &= \mathbf{A}^2(n)-\lfloor M_a^2(n)\rfloor + \lfloor 0.5(\mathbf{B}^2(n)-\lfloor M_b^2(n)\rfloor)\rfloor \\ \mathbf{B}^1(n) &= \mathbf{A}^1(n)-\mathbf{B}^2(n)+\lfloor M_b^2(n)\rfloor\end{aligned}\right\} \cdots\cdots(99)$$

Once the Stage-1 decomposed components are reconstructed from the Stage-2 decomposed components, we can reconstruct the original signal using the following reconstruction equations.

$$\left.\begin{aligned}\mathbf{b}(2n-1) &= \mathbf{A}^1(n)-\lfloor M_a^1(n)\rfloor - \lfloor 0.5(\mathbf{B}^1(n)-\lfloor M_b^1(n)\rfloor)\rfloor \\ \mathbf{b}(2n) &= \mathbf{B}^1(n)+\mathbf{b}(2n-1)-\lfloor M_b^1(n)\rfloor\end{aligned}\right\} \cdots\cdots(100)$$

Thus we can reconstruct the original signal from its 2-stage decomposed components $\mathbf{A}^2$ and $\mathbf{B}^2$. We can also use the integer decomposition and reconstruction schemes of equations (76) and (77) for multi-stage integer decomposition and reconstruction schemes.

4.12 Simplet as a Special Case of YKSK-2 Transform

Simplet stands for simple transform. It is also a subband decomposition and reconstruction scheme. Decomposition is based on the addition or subtraction of some successive elements of the signal. There is no complex multiplication involved in the decomposition scheme or maximum only one multiplication. It can be thought as a special case of YKSK-2 transform, where the filter coefficients are 1's and –1's. And the modifiers are so chosen that the decomposition equations are formed with addition and subtraction operations only. Depending on the number of successive elements used in a decomposition equation, Simplet can be divided into three types, namely, fixed length Simplet, semi-infinite length Simplet and infinite length Simplet. Note here that Simplet was first developed before YKSK

transform and there were only two types of Simplet, viz , fixed length Simplet and Infinite length Simplet. [112, 114]. The concept of semi-infinite length Simplet is brought in from the YKSK transform.

4.12.1 Fixed length Simplet

The number of successive terms of a discrete input signal for the evaluation of any term of a decomposed component is fixed at a certain value. This number may be even or odd, equal or unequal for both the decomposed components. And accordingly fixed length Simplet is further classified into five types, namely, equal and even fixed length Simplet, unequal and even fixed length Simplet, equal and even fixed length Simplet, unequal and even fixed length Simplet and mixed fixed length Simplet. All these can be easily obtained. We discuss below only the first type.

Equal and even fixed length Simplet:

Here the number of successive terms of the signal $\mathbf{b}$ involved in the computation of a term of each of the decomposed components is equal and even. That is, $N_a(n) = N_b(n)$ is even. There can be two forms of the decomposition equations in this type of Simplet. One is given in equation (101) and the other in equation (102).

$$\left.\begin{aligned} \mathbf{A}(n) &= \sum_{i=1}^{L} \mathbf{b}(2n+1-i) \\ \mathbf{B}(n) &= \sum_{i=1}^{L} (-1)^{i-1} \mathbf{b}(2n+1-i) \end{aligned}\right\} \text{ for } 1 \le n \le \lceil m/2 \rceil \quad \cdots\cdots\cdots(101)$$

$$\left.\begin{aligned} \mathbf{A}(n) &= \sum_{i=1}^{L} \mathbf{b}(2n-2+i) \\ \mathbf{B}(n) &= \sum_{i=1}^{L} (-1)^{i-1} \mathbf{b}(2n-2+i) \end{aligned}\right\} \text{ for } 1 \le n \le \lceil m/2 \rceil \quad \cdots\cdots\cdots(102)$$

Equation (101) is the process when the decomposition is started from the left end, and equation (102) is the decomposition process started from the right end and reversing the index of decomposed components. In both cases, the number of terms of $\mathbf{b}$ involved in the computation of a term of decomposed component is L. When L is even it is the case of equal and even fixed length type Simplet.

The reconstruction scheme corresponding to equation (101) is

$$\left.\begin{aligned}\mathbf{b}(2n) &= 0.5(\mathbf{A}(n)+\mathbf{B}(n)) - \sum_{i=1}^{L/2-1}\mathbf{b}(2(n-i)) \\ \mathbf{b}(2n-1) &= 0.5(\mathbf{A}(n)-\mathbf{B}(n)) - \sum_{i=1}^{L/2-1}\mathbf{b}(2(n-i)-1)\end{aligned}\right\} \text{ for } 1 \le n \le \lceil m/2 \rceil \quad \cdots\cdots\cdots (103)$$

We can get the reconstruction scheme corresponding to equation (102) in a similar way.

4.12.2 Improved fixed length Simplet

Here, the proposed improvement is only in terms of computation time. In the above fixed length Simplet, the number of additions in the evaluation of a term of a decomposed component depends on the length of the Simplet. We can make the decomposition scheme independent of the length of the Simplet with a slight modification in the above decomposition scheme as follows.

$$\left.\begin{aligned}\mathbf{A}(n) &= \mathbf{b}(2n)+\mathbf{b}(2n-1)+\mathbf{A}(n-1) \\ \mathbf{B}(n) &= \mathbf{b}(2n)-\mathbf{b}(2n-1)+\mathbf{B}(n-1)\end{aligned}\right\} \text{ for } 1 \le n \le L/2 \cdots\cdots\cdots (104)$$

$$\left.\begin{aligned}\mathbf{A}(n) &= \mathbf{b}(2n)+\mathbf{b}(2n-1)+\mathbf{A}(n-1)-\mathbf{b}(2n-L)-\mathbf{b}(2n-L-1) \\ \mathbf{B}(n) &= \mathbf{b}(2n)-\mathbf{b}(2n-1)+\mathbf{B}(n-1)-\mathbf{b}(2n-L)+\mathbf{b}(2n-L-1)\end{aligned}\right\} \quad \text{for } n > L/2 \quad \cdots\cdots\cdots (105)$$

With this modification, we can reduce the number of addition/subtraction operations involved in the evaluation of a term of a decomposed component. Here it takes only 4 addition/subtraction operations in the evaluation of any $\mathbf{A}(n)$ or $\mathbf{B}(n)$ and is independent of the length of the Simplet. This modification is useful for Simplets with length greater than 4. Modifications of fixed length decomposition scheme in the form of equations (104,105) are not only useful in the decomposition process, but also simplify the reconstruction scheme of fixed even-length Simplet and make it computationally faster. We can write the reconstruction scheme as

$$\left.\begin{aligned}&\left.\begin{aligned}\mathbf{b}(2n) &= 0.5\{\mathbf{A}(n)+\mathbf{B}(n)-\mathbf{A}(n-1)-\mathbf{B}(n-1)\} \\ \mathbf{b}(2n-1) &= 0.5\{\mathbf{A}(n)-\mathbf{B}(n)-\mathbf{A}(n-1)+\mathbf{B}(n-1)\}\end{aligned}\right\} \quad \textit{for } 1 \le n \le L/2 \\ &\left.\begin{aligned}\mathbf{b}(2n) &= 0.5\{\mathbf{A}(n)+\mathbf{B}(n)-\mathbf{A}(n-1)-\mathbf{B}(n-1)\}+\mathbf{b}(2n-L) \\ \mathbf{b}(2n-1) &= 0.5\{\mathbf{A}(n)-\mathbf{B}(n)-\mathbf{A}(n-1)+\mathbf{B}(n-1)\}+\mathbf{b}(2n-L-1)\end{aligned}\right\} \quad \textit{for } n > L/2\end{aligned}\right\} \cdots\cdots (106)$$

Thus, we see that the reconstruction scheme becomes a constant time process. The reconstruction of an element of the original signal requires four addition/subtraction

operations and one division. As the reconstruction scheme can easily be obtained from the corresponding decomposition scheme, only the decomposition schemes are provided for the remaining types of Simplet.

4.12.3 Semi-infinite length Simplet

When computation of a term of a decomposed component requires fixed number of terms of the signal and the computation of a term of the other decomposed component requires more and more terms of the signal. There are three types, even semi-infinite length type, odd semi-infinite length type and mixed semi-infinite length type.

Even semi-infinite length Simplet:

The number of successive elements of signal $\mathbf{b}$ involved in the computation of a term of one of the component, say $\mathbf{A}(n)$, is fixed and even, while the number of successive elements of $\mathbf{b}$ involved in the computation of a term of the other component $\mathbf{B}(n)$ increases continuously in terms of even numbers.

$$\left.\begin{aligned}\mathbf{A}(n) &= \sum_{i=1}^{4}\mathbf{b}(2n-2+i)\\ \mathbf{B}(n) &= \sum_{i=1}^{2}(-1)^{i-1}\mathbf{b}(2n-2+i)+\mathbf{B}(n-1)\end{aligned}\right\} \quad \cdots\cdots\cdots(107)$$

Odd semi-infinite length Simplet:

The number of successive elements of signal $\mathbf{b}$ involved in the computation of a term of one of the component, say $\mathbf{A}(n)$ is fixed and odd, while the number of successive elements of $\mathbf{b}$ involved in the computation of a term of the other component $\mathbf{B}(n)$ increases continuously in terms of odd numbers.

$\mathbf{A}(1) = \mathbf{b}(1), \mathbf{B}(1) = \mathbf{b}(1),$

$$\left.\begin{aligned}\mathbf{A}(n) &= \sum_{i=1}^{3}\mathbf{b}(2n+1-2*i)\\ \mathbf{B}(n) &= \sum_{i=1}^{2}(-1)^{i-1}\mathbf{b}(2n+1-2*i)+\mathbf{B}(n-1)\end{aligned}\right\} \quad \cdots\cdots\cdots(108) \quad \text{for} \quad n>1$$

Computation of $\mathbf{A}(n)$ requires at most 3 elements of $\mathbf{b}$, whereas the computation of $\mathbf{B}(n)$ is not fixed at a maximum but requires $(2n-1)$ elements of $\mathbf{b}$.

Mixed semi-infinite length Simplet:

This is the case where the number of elements of $\mathbf{b}$ involved in the computation of both the decomposed components is not simultaneously even or odd.

$$\left.\begin{aligned}\mathbf{A}(n) &= \sum_{i=1}^{3}\mathbf{b}(2n+1-i)\\ \mathbf{B}(n) &= \sum_{i=1}^{2}(-1)^{i-1}\mathbf{b}(2n+1-i)+\mathbf{B}(n-1)\end{aligned}\right\} \quad \cdots\cdots\cdots(109)$$

Computation of $\mathbf{A}(n)$ requires at most 3 elements of $\mathbf{b}$, whereas the computation of $\mathbf{B}(n)$ is not fixed at a maximum but requires even number, $2n$ elements of $\mathbf{b}$.

4.11.4 Infinite length Simplet

This is the case where the number of terms of the signal involved in the computation of a term of each of the decomposed components is increasing. There are five different types of IL-Simplet, namely, equal and even infinite length type, unequal and even infinite length type, equal and odd infinite length type, unequal and odd infinite length type and mixed infinite length type. One example for each of these types is given below.

Equal and even infinite length Simplet:

$$\left.\begin{aligned}\mathbf{A}(n) &= \mathbf{b}(2n)+\mathbf{b}(2n-1)+\mathbf{A}(n-1)\\ \mathbf{B}(n) &= \mathbf{b}(2n)-\mathbf{b}(2n-1)+\mathbf{B}(n-1)\end{aligned}\right\} \quad \cdots\cdots\cdots (110)$$

Unequal and even length Simplet:

$$\mathbf{B}(1) = \mathbf{b}(1)-\mathbf{b}(2)+\mathbf{b}(3)-\mathbf{b}(4)$$
$$\mathbf{B}(2) = \mathbf{b}(1)-\mathbf{b}(2)+\mathbf{b}(3)-\mathbf{b}(4)+\mathbf{b}(5)-\mathbf{b}(6)$$

$$\left.\begin{aligned}\mathbf{A}(n) &= \mathbf{b}(2n)+\mathbf{b}(2n-1)+\mathbf{A}(n-1), 1\le n\le\lceil m/2\rceil\\ \mathbf{B}(n) &= \mathbf{b}(2n-1)-\mathbf{b}(2n)+\mathbf{b}(2n+1)-\mathbf{b}(2n+2)+\mathbf{B}(n-2), 1\le n\le\lceil m/2\rceil\end{aligned}\right\} \quad \cdots\cdots(111)$$

Equal and odd infinite length Simplet:

$$\left.\begin{aligned}&\mathbf{A}(1)=\mathbf{b}(1),\ \mathbf{B}(1)=\mathbf{b}(1)\\&\mathbf{A}(n)=\mathbf{b}(2n-1)+\mathbf{b}(2n-2)+\mathbf{A}(n-1),\ 2\le n\le\lceil m/2\rceil\\&\mathbf{B}(n)=\mathbf{b}(2n-1)+\mathbf{b}(2n-2)+\mathbf{B}(n-1),\ 2\le n\le\lceil m/2\rceil\end{aligned}\right\}\quad\cdots\cdots\cdots(112)$$

Unequal and odd infinite length Simplet:

$$\left.\begin{aligned}&\mathbf{A}(1)=\mathbf{b}(1)\\&\mathbf{A}(n)=\mathbf{b}(2n-1)+\mathbf{b}(2n-2)+\mathbf{A}(n-1),\ 2\le n\le\lceil m/2\rceil\\&\mathbf{B}(n)=\mathbf{b}(2n-1)-\mathbf{b}(2n-1)+\mathbf{A}(n+1)\end{aligned}\right\}\quad\cdots\cdots\cdots(113)$$

Mixed infinite length Simplet:

$$\left.\begin{aligned}&\mathbf{A}(1)=\mathbf{b}(1)\\&\mathbf{A}(n)=\mathbf{b}(2n-1)+\mathbf{b}(2n-2)+\mathbf{A}(n-1),\ 2\le n\le\lceil m/2\rceil\\&\mathbf{B}(n)=\mathbf{b}(2n)-\mathbf{b}(2n-1)+\mathbf{B}(n-1),\ 1\le n\le\lceil m/2\rceil\end{aligned}\right\}\quad\cdots\cdots\cdots(114)$$

Care must be taken for reconstruction in this case. The odd and even components are to be successively reconstructed using the substitution methods. The odd components are reconstructed from the first decomposition equation and the even components from the second decomposition equation.

4.12.5 Properties and advantages of Simplet

We have discussed various decomposition and reconstruction schemes in Simplet. The following are some of the properties and advantages of these schemes.

1. Both decomposition and perfect reconstruction are computationally fast.
2. No need of decomposition or reconstruction filters and the length of Simplet may be even or odd.
3. The lengths of the decomposed components are independent of the length of the Simplet. This saves a lot of storage space and processing time as compared to the filter bank and discrete wavelet transforms where the length of a decomposed component is dependent on the length of the analysis filters.
4. The characteristics of the decomposed components can be modified by selecting the distorter terms in different ways, which may be useful for different applications.

5. Applying multi-level decomposition on one or both of the decomposed components, multi-resolution signal analysis may be obtained in the case of fixed length Simplet.
6. Applying multi-level decomposition on one or both of the decomposed components, a signal can be distorted to a considerable extent in the case of infinite length Simplet.

4.13 Multilevel Decomposition and Reconstruction in YKSK transform

As in ISITRA, multilevel decomposition is equally possible in YKSK transform also. Both half and full decomposition schemes shown in Figs.3.5a and 3.5b are possible. In multilevel half decomposition scheme, a signal is decomposed into two components and one of the decomposed components at each level is successively decomposed into two components up to a certain desired level. The decomposed components at a particular level are the two decomposed components at that level plus the left out decomposed components at the previous levels. In full decomposition scheme, each of the decomposed components at each level is successively decomposed into two components. The decomposed components at a particular level in the case of full decomposition are the decomposed components at that particular level. The reconstruction in multilevel decomposition scheme is also similar in approach as discussed in the case of ISITRA. Note here that the number of decomposable levels of a signal may not be similar to that of ISITRA because here the length of a decomposed component is independent of the lengths of the decomposition filters used. The length of a decomposed component at a particular level of decomposition can be recursively obtained from the following recursive relation.

$$len_k = \lceil len_{k-1} / 2 \rceil \quad \cdots\cdots\cdots (115)$$

If the length of original signal is a power of 2, then the length of a decomposed components is at each level is half of a decomposed component at the previous level. And the maximum number of decomposable levels is $\log_2(m)$, where m is the length of the signal.

4.14 Discussion and Remarks

YKSK transform is a very simple subband scheme from the decomposition and as well as reconstruction point of view. Its perfect reconstruction theory is simple just like solving two unknowns from two linear equations, quite different from the point of perfect reconstruction theory of ISITRA based on the member condition obtained from the decomposition and the reconstruction filters relations. Hence, there is no member condition in YKSK transforms

and the filters can be chosen arbitrarily, which is not the case in other subband transforms, such filter bank, wavelet or ISITRA where the decomposition and the reconstruction filters have to satisfy their member condition in order to ensure perfect reconstruction.

The decomposition equations of YKSK transform consist of two terms, namely information carriers and modifiers. Information carriers are used for the perfect reconstruction and modifiers are for modification of the signals in certain ways. There are two types of YKSK transform: YKSK-1 transform where each decomposition equation consists of one information carrier and YKSK-2 transform where there are two information carriers in each decomposition equation. Each of YKSK transforms is divided into fixed length type, semi infinite length type and infinite length type according to modifiers involved in each decomposition equation. The fixed length-type of YKSK transform has some similarities with the other subband scheme such as wavelet or ISITRA in its decomposition scheme. But the semi-infinite length types and infinite length types are a new way of decomposition scheme which has no similarities with any of the subband decomposition schemes. Also, multi-stage decomposition scheme is possible in both types of YKSK transform. Moreover, YKSK transform can be easily made integer transforms, which is not possible in the case of ISITRA when the filter coefficients are floating point numbers. The lifting scheme and most of the popular integer transforms used in image compression have been found to be special cases of fixed length type YKSK-1 transform. YKSK-1 transform is not symmetric, because each decomposition equation consists of one information carrier only. The decomposition equation of YKSK-2 transform can be made symmetric by choosing symmetric decomposition filters. There are two ways in which we can reconstruct a signal from its decomposed components in YKSK-2 transform and hence, accordingly, we have two versions of integer transform. YKSK-2 transform can be thought of as a generalized case of YKSK-1 transform as the latter can be obtained from the former. Also, the integer version of YKSK-1 transform is a generalization of S-transform, S+P transforms etc. Application specific optimal filter coefficients for YKSK transforms can be chosen based on optimization of certain criteria of the applications at hand. In addition, multilevel signal decomposition is also possible for each type of YKSK-1 and YKSK-2 transforms.

We may use fixed length type of YKSK transform in place of wavelet or ISITRA, for example, in image compression. Infinite length types are found to be good in distorting

signals, which may find application in encryption for signal hiding point of view. Another reason why YKSK transform can be used for encryption is its noise sensitivity in its reconstruction process. When there is significant noise in its decomposed components or the filters are slightly different, the reconstructed signal is indistinguishably noisy. The noise sensitivity is more pronounced in the case of infinite length types of YKSK transform including Simplet. Wavelet or ISITRA has good reconstruction property in the sense that we can get well distinguishable reconstructed signal even if there is significant noise in the decomposed components and hence they are not suitable for signal encryption. But we are yet to find areas where semi-infinite length can be applied.

4.15 Conclusions

YKSK transform is a simple and generalized subband scheme from the perfect reconstruction point of view. Here we can choose arbitrary filters of any pattern that would be suitable for a particular application. Fixed length type of YKSK transform can be used in place of wavelet or ISITRA in certain applications. The multi-stage case of fixed length type YKSK-1 transform is a simpler and more generalized form of lifting scheme. YKSK-2 transform, being a symmetric version of YKSK-1 transform may find some applications in image processing where symmetric filters are preferred. Its infinite length type is found to be useful in encryption. Moreover, its different integer versions may find a place in lossless image compression as well.

Chapter 5

SOME APPLICATIONS OF ISITRA AND YKSK TRANSFORMS

Abstract: *Some applications of ISITRA and YKSK transforms are given here. ISITRA is used for lossy image compression and YKSK transform for signal encryption. We propose here two new subband coding schemes entitled ISPIHT (Improved SPIHT) and DPC (Differential Position Coding) for image compression. In ISPIHT, the coding part of SPIHT is improved to reduce the number of search operations and memory requirement during coding while preserving the compression performance the same as in SPIHT. This modification results in the coding speed of ISPIHT about 3 to 5 times faster than SPIHT. The decoding part remains the same both in ISPIHT and SPIHT. The other coding scheme, DPC, is a very simple, fast and memory efficient coding scheme. However, its compression performance in terms of PSNR value is marginally lower than those of standard subband coding schemes of SPIHT and JPEG2000 in most of the images. Considering its speed and low memory requirement, DPC is a better coding scheme than the two standard subband coding schemes. Both ISPIHT and DPC can be used for lossless as well as lossy image compression. But we use them for lossy image compression. Also, we propose some ISITRA-2 filters of length 10 having the same framework as that of the popular bi-orthogonal 9/7 wavelet filters. The proposed filters give better compression performance than the 9/7 filters in some images. For signal encryption purpose, we use infinite length YKSK transform to distort a signal to hide the information content in the signal. Then, we apply a new random sequence known as meitei lock sequence to add security to the distorted signals. Experimental results show that the proposed encryption scheme has a very tight security.*

5.1 Introduction

In this chapter, some applications of ISITRA and YKSK transform are provided. ISITRA may be used in various fields of application in place of discrete wavelets. But we will use ISITRA in lossy image compression. In lossy image compression, the decompressed image is not exactly equal to the original image. Hence, compression performance of a coding scheme is usually measured in terms of mean square error (MSE) or peak signal to noise ratio (PSNR) at a given compression ratio. In addition to these, the time complexity and the memory requirement of a compression scheme are also used to compare a compression scheme to the other. A compression scheme which gives lower MSE or higher PSNR value at a given compression ratio in less time using less memory is considered as a better compression scheme. Most of the research works on lossy image compression [2, 12, 22, 24, 42, 63, 66, 67, 91, 95, 97, 105, 109, 112, 113, 130, 143, 145, 163] use MSE and PSNR value as a measure of compression performance. We will also use PSNR values to compare the performance of our coding schemes to the standard coding schemes. For our compression scheme, we propose here two coding schemes known as ISPIHT (Improved SPIHT) and DPC (Differential Position Coding). The former is similar to SPIHT [109] in its approach, but is simpler and a faster than SPIHT without any compromise in the performance. The second one is a new coding scheme, which is significantly much faster and occupies considerably less memory space than SPIHT and JPEG2000 [60]. But its performance in terms of PSNR value is a marginally lower than SPIHT and JPEG2000 at compression ratios 16:1 and lower. At the higher compression ratios, it gives similar or even marginally better performance than SPIHT without arithmetic coding and JPEG2000 in some images. Our compression scheme is based subband decomposition scheme plus coding scheme as described in Section 5.2. In ISPIHT based compression scheme, entropy-coding scheme may or may not be used as in SPIHT. However, in DPC based compression scheme, an entropy-coding scheme is required to code the differential position information of the subband coefficients. We use arithmetic-coding as our entropy coding scheme in DPC.

For lossy image compression based on wavelet or filter bank decomposition scheme, a proper choice of filter has considerable impact on the compression performance of different images. Some criteria on the choice of orthogonal wavelet filters for lossy image compression are suggested in [35, 144], based on regularity of filters. However, it is mentioned [157] that the importance of regularity on image compression is still an open

question. In [160], the regularity property of bi-orthogonal wavelet filters and their impact on image compression are discussed and it is shown that filters with high regularity do not always give better compression performance. Also, compression performance of various commonly used integer transforms is discussed with the conclusion that different filters with high vanishing moments give different performance for different images [1]. A suitable criterion for evaluation of filters to be used in image compression can be MSE or PSNR value. A filter set that gives lower MSE or higher PSNR value for an image can be considered as a better filter set. The flexibility in choosing filters in ISITRA and YKSK transform can be employed to find optimal filters based on the optimization of certain criteria for any specific application. For the compression purpose, we use genetic algorithm to find filters that gives minimum MSE at level-1 decomposition and reconstruction [124]. We suggest some ISITRA-2 filters of length 10 having the same framework as that of commonly used 9/7 bi-orthogonal wavelet filters for lossy image compression. The proposed ISITRA filters give marginally better performance than 9/7 filters in some images. This shows that the higher degree of freedom in choosing filters in ISITRA and YKSK transform will enable us to find better filters for compression as well as for other applications. Of the two subband transforms ISITRA and YKSK transform, ISITRA would be more suitable for lossy image compression because its reconstruction process results in smoother signal as it is based on convolution. We know that integer transforms [24,110,113] etc. are used for lossless and lossy image compression. Undoubtedly, YKSK transform, being a more generalized case of these integer transforms, may also be used for image compression. But, here we use YKSK transform for signal encryption.

Infinite length type YKSK transform has the ability to produce signal distortion. This is important for the signal hiding point of view. We can use either infinite length type YKSK-1 transform or YKSK-2 transform for signal distortion. But we use Simplet, a special case of YKSK-2 because of its simplicity and ability to distort a signal fast [123]. Also, it has another important property useful for encryption, that is, reconstruction difficulty in the presence of noise. During decomposition, the distorting ability hides original signal by transforming it into quite a different signal, while the reconstruction difficulty (in the presence of noise) makes it more desirable for encryption as it may add security to the encryption scheme. However, Simplet itself is not sufficient for encryption because we can

easily get back the original signal from its distorted components. We have to devise some mechanism to add noise to the distorted components so that it cannot be easily reconstructed unless the noises in the distorted components are totally removed. Addition and removal of noise is done using a random sequence known as meitei lock sequence (MLS) [122]. The distorted components plus MLS form encrypted components from which the reconstruction of the original signal is not possible unless one knows the exact MLS used in the encryption. MLS is generated from any arbitrary array and is empirically found to be unique for an array. Another important property of an MLS is its non-periodic nature. These uniqueness and aperiodicity features of MLS, separately discussed in section 5.7, form desirable properties of MLS for use in signal encryption. In our encryption scheme, we first distort the signal using Simplet and then encrypt the distorted components using MLS. We compute the correlation coefficients between the original signal and decrypted signal for various possible keys. It is found that when there is slight different in any of the encryption and decryption key, the correlation coefficients is negligible small, which means that the decrypted signals have no similarity with the original signal as shown in subsection 5.8.2. In other word, our proposed encryption scheme is a secure one.

5.2 Compression Scheme

Our proposed compression scheme consists of two models, compression and decompression model as shown in Fig.5.1. The compression model is shown in Fig.5.1a in which the original signal $\mathbf{b}$ is first decomposed into subbands using some transform such as ISITRA or YKSK transform. The decomposed components or subbands are then coded with a coding scheme, which may or may not include entropy coding to give the coded bit-stream $\tilde{\mathbf{b}}$.

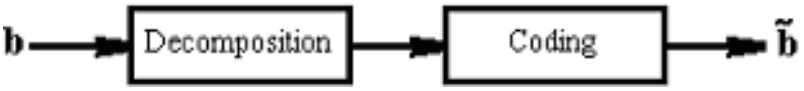


Figure 5.1a Compression model

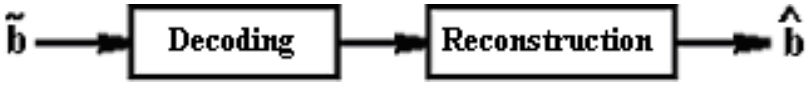


Figure 5.1b Decompression model

The decompression model is shown in Fig.5.1b. It is in some sense the reverse process of the compression model. Firstly, the bit-streams are converted back into appropriate subbands using appropriate decoding scheme. From the decoded subbands, we get the decompressed image or reconstructed image $\hat{\mathbf{b}}$ using the suitable reconstruction scheme of ISITRA or YKSK transform.

If the signal $\mathbf{b}$ is exactly the same as $\hat{\mathbf{b}}$, then the compression scheme is called lossless compression. For lossless compression, only the speed and the compression ratio are the two main measures to make comparison between two or more lossless compression schemes. The higher the speed and the compression ratio the better is the lossless compression scheme. In lossless compression it is very difficult to get higher compression rate and most of them are very slow. Some sources of lossless image compression based on subband decomposition schemes are [1,110,113] etc. When $\mathbf{b}$ is not exactly equal $\hat{\mathbf{b}}$, then the compression scheme is called lossy compression. In lossy image compression, we can get higher compression ratio or rate. The logic behind the lossy image compression is based on the visual imperceptibility of image defects introduced due to coarse quantization or coding. For example, when a gray level image with pixel value range 1-255 at 8 bpp, is represented with a pixel range of 32-255 at 6 bpp replacing those values less than 32 with zeros, there will be hardly any visual defects in the two images shown in Fig.5.2a and Fig.5.2b.

Figure 5.2a Original image (1-255)

Figure 5.2b Modified image (32-255)

But numerically they are different. This is a simple example of lossy image compression. In the lossy image compression, because of rejection of some pixel values or because of coarse quantization, errors are introduced in the decompressed image. Generally, MSE and PSNR values are used for error measurement, which reflect the compression performance in a lossy

image compression. The lower the MSE or the higher the PSNR value, the higher is the image quality. But their value cannot measure image quality as discussed in the next subsection.

5.2.1 Mean square error (MSE)

It is generally used to measure quantization or reconstruction error between a signal and its quantized or reconstructed signal. Suppose $\mathbf{b}$ is an image and $\hat{\mathbf{b}}$ is the reconstructed image through some quantization or reconstruction process. Then the MSE between the two images is mathematically defined as

$$\text{MSE}=\frac{1}{MN}\sum_{i=1}^{M}\sum_{j=1}^{N}(\mathbf{b}(i,j)-\hat{\mathbf{b}}(i,j))^2 \quad \text{.....(1)}$$

where M is the number of rows and N is the number of columns of the two images. Generally it is assumed that the lower the MSE value, the lower is the reconstruction error. But the MSE value between two images cannot determine the amount of perceptual difference between the two images.

Suppose $\mathbf{b}_1$ is an image. If we replace the last row and last column of the image with zeros and call the resulting image $\mathbf{b}_2$, then we will find much higher MSE and lower PSNR values between these two images. But this does not mean that the two images have much difference as a whole. The error is introduced only because of the last row and the last column. This cannot be taken for error of the whole image $\mathbf{b}_2$ as the remaining parts of the image have no error. This indicates that MSE value cannot be taken as a faithful measure of error.

5.2.2 Peak signal to noise ratio (PSNR)

Another measure of quantization error is PSNR value. It is the ratio in decibel of the peak value of a reconstructed signal to the Mean Square Error (MSE) between the reconstructed signal and its original signal. For lossy compression of images, the peak value in PSNR value is the peak value of the decompressed image. It is taken as a measure of error introduced in the compression process. It is mathematically defined as

$$PSNR_r = 20 \log_{10}\left(\frac{P_r}{\sqrt{MSE}}\right), \ldots\ldots\ldots..(2)$$

where P_r is the maximum or peak value of the decompressed or reconstructed image.

For a gray scale image, the pixel values range from 0 –255. Hence the most often peak value is taken as 255, i.e.,

$$PSNR = 20 \log_{10}\left(\frac{255}{\sqrt{MSE}}\right), \ldots\ldots\ldots(3)$$

We would rather call equation (3) as the PSNR value with respect to the original image, because generally the original image has the peak pixel value as 255. But it is not always true that the original image will have the peak value as 255. Hence we can modify equation (3) as

$$PSNR_o = 20 \log_{10}\left(\frac{P_o}{\sqrt{MSE}}\right), \ldots\ldots.(4)$$

where P_o is the peak value of the original image.

Most often the peak value of the decompressed image or the original image is sometimes higher or sometimes lower than 255. However, in most lossy compression schemes, equation (3) is the most common form of measuring PSNR value.

5.3 Improved SPIHT (ISPIHT)

Lossy image compression is the major research area in image compression because high compression rates can be achieved without introducing much perceptible visual distortion in the compressed signal. Of the lossy image compression techniques, subband image compressions based on wavelet transform are notably the most popular ones. There are numerous papers on wavelet based subband image compression. But the popular ones, which gave an impetus to lossy image compression, are EZW [112] and SPIHT [109]. SPIHT is a simplification and generalization of EZW. A simple and well-explained SPIHT coding scheme is also given in [100]. Another coding scheme, which becomes popular in the recent years, is EBCOT [141] and is the main coding scheme in the JPEG2000 [60]. EBCOT is algorithmically is similar to SPIHT in its coding approach except that it is block based. It reduces the buffer memory size during the coding but it does not exploit the

redundancy of between subbands. Moreover, EBCOT or JPEG2000 is inherent with an entropy-coding scheme and it is difficult to achieve exact target bit rate during the compression and hence post-processing operations are required to keep the target bit rate. That is, JPEG2000 always performs extra coding which will be discarded during the rate control post processing state. This tends to make JPEG2000 encoders much slower than decoders.

In terms of compression performance JPEG2000 is significantly improved than its predecessor JPEG. It has added various new features like, scalability, error resilience, coding based on region of interest etc. More literature on JPEG2000 can be had from [79, 98, 142] and free software is also available from [169, 170]. The compression performance of JPEG2000, however, is not very significant than that of SPIHT in spite of its increased complexity. On the other hand, SPIHT is relatively simple and straightforward and it is still considered as a benchmark for lossy image compression scheme. It performs only necessary coding to achieve a given target compression rate and it can stand alone as a coder without using any entropy coder. Entropy coder is optional, it is used only when a little more compression is desired. However, the main drawbacks of SPIHT are the requirement of large buffer memory during coding and large number of search operations. The ISPIHT coding scheme proposed here is an attempt to reduce the memory requirement and number of search operations during the SPIHT coding without affecting its performance.

SPIHT coding uses three lists LIP (List of Insignificant Pixels), LSP (List of Significant Pixels) and LIS (List of Insignificant Sets). The significance information from of the pixels corresponding to the co-ordinates stored in LIP and LSP can be generated even if we store the pixel values instead of the co-ordinates. Hence, we can directly store only the pixel values in the LIP and LSP. If we store the pixel values in the LIP and LSP, we would be able to reduce half of the memory space required by LIP and LSP. It may simplify the algorithm as well, as we can get the pixel values directly from the LIP and LSP instead of getting the pixel values indirectly from the original image matrix by using the co-ordinates stored in the LIP or LSP. Another improvement that can be made on the SPIHT is the searching of the significant sets in LIS. The significance information of the LIS gives the position information of the pixels in the image. In SPIHT, an LIS is repeatedly searched if it is insignificant at the previous one or more threshold levels. Testing a significance of an LIS requires testing of all pixels in all sets in the corresponding hierarchical tree. This can be

avoided if we store the maximum pixel values of all sets in an array within the LIS in the first significant test. If a hierarchical tree of an LIS is found to be insignificant in the first testing, in subsequent testing instead of testing all pixels in all sets, it will simply test only the maximum values of the sets, which are already stored in an array. If we implement these two modifications in the SPIHT scheme, it will give the same result as SPIHT but its memory requirement and computational time will be less. We implement these two modifications using a different hierarchical tree scheme known as ISPIHT (Improved SPIHT) to simplify the coding scheme further.

5.3.1 Spatial ISPIHT

In the spatial tree of SPIHT, the top leftmost pixel is taken as root. The root has three immediate children. And each pixel other than the root has four pixels as its children. It is difficult to trace and visualize which groups of pixels are the descendents of which pixels. This requires tracing back and forth along the tree path. We eliminate such awkwardness in our tree generation method. In our 2-dimensional tree structure, we take the upper leftmost element as root. The root has three descendent trees formed by successive nodes. Each node is associated with a sub-matrix whose size is dependent on the level of the nodes. In other words, the node at any level n is associated with a sub-matrix of size $2^{n-1} \times 2^{n-1}$. The schematic tree is shown in the Fig.5.3.a. A typical spatial tree having up to level 4 is shown in Fig.5.3.b.

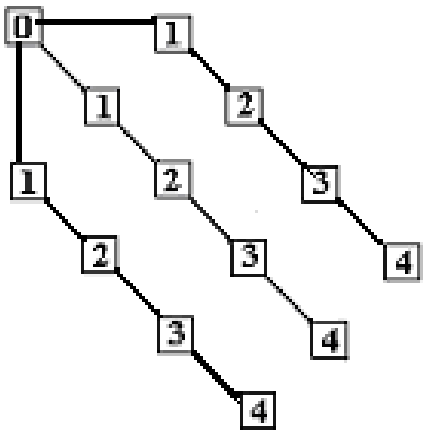


Figure 5.3.a ISPIHT Structure

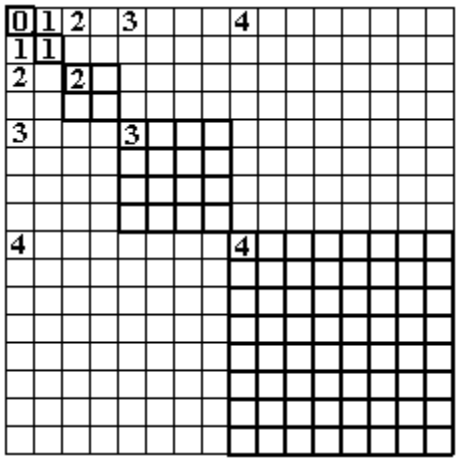


Figure 5.3.b Spatial ISPIHT

Some terms associated with spatial ISPIHT structure are described below.

Node Identification:

The upper leftmost corner pixel co-ordinate identifies the root. The middle node at level l is identified with (p, p) co-ordinate, where $p = 2^{l-1} + 1$. The left node and the right node at the same level are identified with $(1, p)$ and $(p,1)$ co-ordinates.

Number of levels in a descendent tree:

Let $\mathbf{A}$ be a matrix of size $m \times n$, then the maximum possible number of levels in the matrix-tree is given by λ, where $\lambda = \lceil \log_2(\min(m, n)) \rceil$. For simplicity, we assume $\mathbf{A}$ as a square matrix having its size in terms of power of 2.

Number of pixels associated with a node:

Each node is associated with a set of pixels. The number of pixels associated with a node depends on the level of the node. The nodes in the first level are associated with one pixel each. The nodes in other levels are associated with 4^{l-1} pixels, where l is the level of the node.

5.3.2. ISPIHT generation

We know that each node is identified with distinct a co-ordinate and each node is associated with a set of pixels. The set of pixels associated with a node of ISPIHT will be known as node matrix.

Let $\mathbf{A}$ be an image matrix of size $m \times n$. We denote the pixels associated with the co-ordinate (x, y) of a node at level l by $\mathbf{N}_l(x, y)$. Suppose the image matrix has λ levels. For a unity offset system of arrays, the root of the tree is (1,1) and the pixel associated with it is denoted by $\mathbf{N}_0(1,1) = \mathbf{A}(1,1)$.

For level-1, the pixels associated with each node are as given below

$\mathbf{N}_1(2,2) = \mathbf{A}(2,2), \mathbf{N}_1(1,2) = \mathbf{A}(1,2), \mathbf{N}_1(2,1) = \mathbf{A}(2,1)$

For any other level l from 2 to λ, the pixels associated with each node are

$\mathbf{N}_l(x, y) = \mathbf{A}(C_{l-1}(x, y))$

where $C_j(x,y) = \begin{bmatrix} (0,0) & (0,1) & (0,2) & \cdots & (0,j) \\ (1,0) & (1,1) & (1,2) & \cdots & (1,j) \\ (2,0) & (2,1) & (2,2) & \cdots & (2,j) \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ (2^j-1,0) & (2^j-1,1) & (2^j-1,2) & \cdots & (2^j-1,2^j-1) \end{bmatrix} + (x,y)$

Thus, we can easily generate the sub-matrices associated with the nodes of a descendent tree for a matrix **A**, if we know the co-ordinates of the top leftmost pixels of the sub-matrices.

We employ two types of descendent trees, namely type-0 and type-1, while testing for significance of pixels associated with a hierarchical tree. A type-0 descendent tree associated with the co-ordinate (x,y), will be denoted by T(x,y,0) and similarly type-1 by T(x,y,1). A type-0 and a type-1 tree corresponding to a pixel co-ordinate are defined as follows.

T(x,y,0)=$\{N_2(x,y) \cup N_3(2x-1,2y-1) \cup \cdots \cup N_\lambda(2^{\lambda-1}x+1,2^{\lambda-1}x+1)\}$

And

T(x,y,1)= $\{N_3(2x-1,2y-1) \cup \cdots \cup N_\lambda(2^{\lambda-1}x+1,2^{\lambda-1}x+1)\}$

We usually generate the type-0 tree for a given co-ordinate. Type-1 can be easily obtained from the type-0 by neglecting the first 2x2 sub-matrix.

In order to reduce the number comparisons while the testing of significance of a descendent tree, a descendent tree is represented by an array known as value tree whose elements are the maximum values of the node matrices in the tree. That is, the value tree of a descendent tree of type-a, V(T(x,y,a)) is the set of maximum values of the node matrices in the tree T(x,y,a).

V(T(x,y,a))=$\{Max(N_i(x,y)), \forall\ i = 2,3,...,\lambda\ if\ \ a = 0, i = 3,...,\lambda\ if\ \ a = 0\}$

In short, a valued tree (VT) of a co-ordinate of l nodes is an array of l pixels of the maximum absolute values of the node matrices in the corresponding descendent tree.

Algorithm to generate the VT of any co-ordinate (x,y)

Input: Data matrix **A** and any co-ordinate (x,y)

Output: V(T(x,y,0)), maximum values of the sub-matrices in the type-0 descendent tree of the given co-ordinate.

lc_1, lc_2, the last coordinates of the first sub-matrix.

Initialize a counter, $k = 1$ and Output to an empty array.

While (lc_1, lc_2 <= $Size(A)$)

Generate the coordinate matrix $\mathbf{C}$ of the sub-matrix of size 2^k

Find the maximum value of $A(\mathbf{C})$ and append it to the Output.

Increment k

Update lc_1, lc_2 with the last co-ordinate of $\mathbf{C}$.

End while

5.3.3 Lists LIS, LIP and LSP

In the coding scheme, we use three lists as in SPIHT. LIS is generated using the VT algorithm. It is used to find the significant information of the node matrices in the descendent trees. LIS is the list of valued Trees (VTs).

A VT is said to be significantwith respect to a threshold, Thr if any value V_i in the value of VT is greater than or equal to the threshold. I.e., if $V_i >= Thr \ \ for\ any\ V_i \in VT$. We denote the significance of VT by 1 and that of insignificance by 0. If S denotes a significance function, then

$$S(VT) = \begin{cases} 1, \ if\ V_i >= Thr \\ 0, \ Otherwise \end{cases}$$

LIP is a list of insignificant pixels, which are insignificant for a given threshold, Thr. That is,

$$\text{LIP} = \{A(x_i, y_j) : \left|A(x_i, y_j)\right| < Thr \ \ for\ some\ x_i, y_j\}.$$

The significance of the pixels $A(x_i, y_j)$ in the LIP is tested using the significance function S as

$$S(A(x_i, y_j)) = \begin{cases} 1, \ if\ \left|A(x_i, y_j)\right| >= Thr, \ \ for\ any\ A(x_i, y_j) \in LIP \\ 0, \ Otherwise \end{cases}$$

The third list is the LSP, list of significant pixels. It is the set or list of pixels, which are found significant in the previous thresholds. I.e.,

$$\text{LSP} = \{A(x_l, y_m) : \left|A(x_l, y_m)\right| >= Thr, \ \ for\ any\ A(x_l, y_m) \in LSP\}$$

LSP is used to send the less significant bit information of significant pixels in the succeeding threshold levels.

5.3.4. ISPIHT Coding algorithm

In this coding scheme, we extend the tree structure hierarchically in terms of node matrices so that tree may be represented in a simpler way. The coding scheme consists of three stages, namely, initialization, Sorting Pass and Refinement Pass as in SPIHT. So, the basic idea of coding remains the same but the actual coding approach is significantly different. We store pixel values instead of the pixel co-ordinates in LIP and LSP so that the searching operation may be done in a faster way and the memory requirement may be reduced. The LIS is a list that stores not only the pixel co-ordinates of the hierarchical trees but also of the values of the tree. This saves us a significant number of searching operations while testing the significance of type-1 hierarchical trees and any type-0 trees, which were found insignificant in the previous significant testing.

Initialization:

The initialization stage consists of assigning the three lists to be used in the coding with appropriate values. First LIP is initialized with the pixel values associated with the co-ordinates (1,1), (1,2), (2,1) and (2,2), LSP to an empty matrix. Then we generate the list of LIS structure having (1,3), (3,1) and (3,3) as coordinates and the maximum values of the node matrices in their respective descendent trees. Then, the initial threshold is computed from the maximum values of the VTs and the elements of the LIP as

M=floor(log2(Vm));

Where Vm=maximum value of the VTs in LIS and the pixels in LIP.

Threshold=2^M;

Sorting Pass:

It involves testing of significance of the pixels in the LIP (LIP testing) and testing of significance of the descendent tree in the LIS structure (LIS testing).

LIP testing:

It tests whether a pixel in the LIP is greater than or equal to the current threshold value. A pixel whose absolute value is greater than or equal to the threshold value is assumed to be significant. Otherwise, the pixel is assumed to be insignificant. To indicate the significance, we send 1 each time a pixel in the LIP is found to be significant followed by a sign bit

information. Then, we append the absolute value to the LSP and remove the same from the LIP. For an insignificant pixel, we simply send a zero.

LIS Testing:

The LIS testing is a little more involved than LIP testing. Here, we first check the type of the VTs and then test their significance. A descendent tree as represented by a VT is said to be significant, if there is any value in VT, which is greater than or equal to the threshold. Otherwise, the descendent tree is said to be insignificant. If a type-0 VT is significant, we send 1, then we test the significance of the four pixels associated with the topmost node of the tree. For a significant pixel, we send a 1 followed by a sign bit and append its absolute value in the LSP. For insignificant pixels, we send 0s and the pixel values are appended to LIP. We then test if the number of elements in the corresponding VT is greater than 1. If so, we append the corresponding VT to LIS as type-1 by removing the first element of the VT. For an insignificant VT of type -0, we simply send a 0.

If a type-1 VT is significant, we send a 1. Then we generate VT of type-0 corresponding to the co-ordinates of the top leftmost corner pixels of the four 2x2 sub-matrices of 4x4 node matrix in the current node. Then, we remove the corresponding VT from the LIS. For an insignificant VT of type-1, we simply send a zero.

Refinement Pass:

Refinement pass is used to reduce the error introduced during the decoding process. The bits we send during the LIP testing and LIS testing give only information whether the pixel value is greater than threshold or not and its position in the image matrix. But they do not give any information about how much the pixel value is greater than the threshold. The refinement pass supplements this information during the decoding to upgrade the pixel values at each threshold levels to reduce the decoding error. The process of refinement pass remains the same as in SPIHT.

After each completion of sorting pass and refinement pass, the threshold value is reduced by halve. And then same process repeated for more threshold levels. As the information is coded in the form of bit stream, we can stop the coding process at any desired length of the bit-stream.

Algorithm: ISPIHT Coding

Output: Bit stream

Input: Wavelet co-efficient or data matrix to be coded, A, the number of threshold levels, N.

Assign LIP={ A(1,1), A(1,2), A(2,1), A(2,2)}

Assign LIS with VTs for coordinates (1,3), (3,1) and (3,3) as type-0 descendent trees.

Compute Vm, M, Threshold (as described in the Initialization).

Assign Lp1=0;

Comment: Sorting Pass

For I=1 to N

Comment: LIP Testing

For each pixel in the LIP

If a pixel is significant

Send a 1, followed by sign bit.

Delete the pixel form LIP and append its absolute value to LSP.

Else

Send a 0

End **if**

End **For each**

Comment: LIS Testing

For each VT in LIS

If the type of VT is 0

If VT is significant

Send a 1

For each of the four pixels associated with the node of the VT.

If a pixel is significant,

Send a 1 followed by sign bit.

Append the absolute value of the pixel to LSP.

Else

Send a 0

Append the pixel to LIP

End **If**

End **For each**

If VT has more than 1 element

Neglect the first element, change its type to 1 and append to LIS.

End **If**

Else

Send a 0

End **If**

Else

If VT (of type-1) is significant

Send a 1

Generate VT corresponding to the four top leftmost pixel co-ordinates of the four 2x2 sub-matrices associated with current node.

Delete the VT from the LIS.

Else

Send a 0

End **If**

End **If**

End **For each**

Comment: Refinement Pass

For r=1 to Lp1

If (LSP(r) –Threshold)>=Threshold/2

Send a 1

Else

Send a 0

End **if**

End **For**

Lp1=No. of pixels in the LSP

Threshold=Threshold/2;

End **For**

The decoding algorithm is just the reverse of the coding algorithm. But a difference with coding algorithm is that the LIP and LSP are stored as co-ordinates. And the LIS stores only

the pixel co-ordinates of the topmost nodes of the descendent trees and does not store VTs. Because during the decoding, the significance of a descendent tree can be ascertained by testing whether the corresponding bit is 1 or 0. In other words, it does not require exhaustive search for testing the significance of a descending tree. This makes the decoding much faster than coding in SPIHT. The decoding part in ISPIHT remains the same as in SPIHT except the way of representation of the tree structure. So, we can also directly use the decoding algorithm of SPIHT.

5.3.5 Experimental results

The algorithm gives the same compression performance and the image quality as SPIHT at any compression ratio. But it requires comparatively less memory space and less number of comparison operations during coding. We compare the memory requirement and number of comparison operations required for compressing an image at various compression ratios. First an image is decomposed up to all possible decomposition levels using 9/7 bi-orthogonal filters. Then, we code the decomposed components using ISPIHT and SPIHT. Here, we use two popular gray images, Lena and Barbara of size 512x512 pixels. The comparison is made for five different compression ratios 8:1, 16:1, 32:1, 64:1 and 128:1. We compute the number of comparison operations at each threshold level passed before achieving the target rate. For Lena image, the target rate is more or less matched with the threshold value. That is, for achieving 128:1 compression ratio, we have to code at least up to the threshold value 128. But it requires a little more bits from the threshold level 64 to get the exact 128:1 compression ratio. Similarly, to get the compression ratio of 64:1, we require all bits obtained from the threshold level 64 plus a little more from the next lower level threshold. Similar is the case for the other compression ratios.

The first column of the table represents the threshold levels. The second and the third columns represent the number of comparison operations required to test the significance of the descendent trees of type-0 and type-1 at the particular threshold level in SPIHT. The fourth and fifth columns represent the number of comparison required to perform the test of significance of type-0 and type-1 descendent tree at the particular threshold level in ISPIHT. The LIP column represents the number of pixels in the LIP before the start of LIP testing and Sorting Pass at any threshold level. The last LSP column represents the number of

significant pixels after LIP testing and Sorting Pass at the current threshold level. The memory requirement for LIP and LSP in ISPIHT is just the half of the memory requirement in SPIHT.

CR	Threshold	LIS of SPIHT		LIS of ISPIHT		LIP	LSP
		Type-0	Type-1	Type-0	Type-1		
L E N A 128:1 64:1 32:1 16:1 8:1	32768	262140	0	24	0	4	1
	16384	262140	0	24	0	3	2
	8192	262140	0	24	0	3	2
	4096	262140	174752	24	14	3	3
	2048	415028	420448	164	245774	9	21
	1024	338516	311104	368	127157	39	74
	512	312420	300512	882	144812	98	232
	256	324056	270512	1961	129204	256	557
	128	332084	269600	4490	131318	703	1512
	64	252688	210240	2980	93940	2024	3703
	32	262188	212416	11010	90350	4317	8186
	16	252688	210240	16018	94213	16139	16969
	8	295788	236624	28911	148691	36211	35497

Table 5.1 Number of comparisons in LIS testing and the number of elements in LIP and LSP at various threshold levels for compressing 512x512 Lena image at various compression ratios.

CR	Threshold	LIS-SPIHT		LIS -ISPIHT		LIP	LSP
B A R B A R A 128,64:1 32:1 16:1 8:1	32768	87380	0	24	0	4	2
	16384	87380	0	24	0	3	2
	8192	87380	0	24	0	3	2
	4096	87380	0	24	0	3	2
	2048	567948	414992	156	229453	3	18
	1024	376744	345232	382	168126	38	65
	512	297428	302896	803	118202	107	203
	256	294004	297984	1792	135050	289	553
	128	298228	279648	4663	121019	675	1612
	64	329172	300928	12816	155562	2464	7349
	32	275204	236960	15536	115756	11719	18501
	16	240028	198672	19624	99626	18815	35851

Table 5.2 Number of comparisons in LIS testing and the number of elements in LIP and LSP at various threshold levels for compressing 512x512 Barbara image at various compression ratios.

From the two tables, we see that the number of comparison operations is significantly less in the case of ISPIHT than in SPIHT to code the same number pixel values. The number of comparison operations is significantly less in the case ISPIHT at higher compression ratio

but the significance level reduces at lower compression ratio. Experimentally, ISPIHT becomes faster by 3 to 5 times than SPIHT during coding of different images. Also, the memory requirement is for be about 2 times less in ISPIHT than in SPIHT.

5.4 Differential Position Coding (DPC)

In SPIHT and ISPIHT, the use of lists, LIP, LIS and LSP requires large amount of memory space. Also, sorting pass requires a considerable amount of processing time. We propose here a new coding scheme, which requires less memory and less processing time. In this coding scheme, we code the position information of the significant pixels at a given threshold. The main idea behind DPC is that the positions of the significant coefficients are more or less correlated for most images, which can be compressed significantly using a entropy coding such as Huffman or arithmetic coding. The coding scheme consists only two stages, position coding and refinement pass. The position coding finds the position information of the significant pixels and find the successive difference of the pixels positions. Position coding is simple and takes a constant time. The refinement pass that followed is the same as that used in SPIHT. During coding we require only a list to store the significant pixels for refinement pass. Here no sorting and LIP testing is required. Thus, it saves not only memory but also the processing time significantly. In spite of its simplicity, its performance is more or less comparable to JPEG2000 and SPIHT in terms of PSNR value at a compression ratio. However, considering its speed and memory requirement, it is significantly better than both SPIHT and JPEG2000. This coding scheme can be used for both lossy and lossless compression. But we use here for lossy image compression purpose.

5.4.1 Encoding in DPC

In DPC, we first find the initial threshold value from the maximum values of the data to be coded as in SPIHT or ISPIHT. Then, we test the significance of the pixels or coefficients at various thresholds. A coefficient is said to be significant with respect to a threshold if its value is greater than or equal to the threshold value. We find the position information of the coefficients of the subbands, which are significant with respect to the threshold. The difference in the successive positions of the significant coefficients, eg 1st from the 2nd, 2nd from the 3rd and so on are sent to the output. The significant pixels are put to LSP and the corresponding significant pixels are deleted from the input data. The sign bits of the

significant values are appended as binary string to an output array known as sign and refinement bit (SRB). This is followed by refinement pass, which is the same as described in the case of ISPIHT. That is, if 2^m is the current threshold, then in the refinement pass, we send m-th bit of the values in the LSP which were significant in the previous thresholds. The threshold is decreased by a factor of 2 and the whole process is repeated for a specified number of threshold levels in the case of lossy compression or for all possible levels of thresholds for lossless compression. The encoding algorithm is given below.

Encoding Algorithm:

Input: X→ Image /Data to be coded, N→ Desired number of threshold levels

Output: Px→ Difference of the significant pixels positions

K→ No of significant pixels at each threshold level.

SRB→ Sign and Refinement Bits

Step-1: Find the initial threshold, Thr. Assign Px, K, SRB and LSP to empty arrays.

Step-2: Find the number of significant pixels at the current threshold.

Step-3: Find the successive difference of the pixel positions at step-2 and append it to Px.

Step-4: Append the number of significant pixels to K.

Step-5: Send the sign bits of the significant pixels to SRB. Append the significant pixels to LSP and delete the corresponding pixels from X.

Step-6: Perform refinement pass of the pixels in the LSP, which were significant in the previous threshold levels and append the resulting bits to SRB.

Step-7: Decrease the Thr by a factor of 2.

Step-8: Repeat the step 2 to 7 for the specified number of times, viz N.

5.4.2 Decoding in DPC

Decoding requires the information of the size of the original data, the initial threshold value in addition to Px, K, SRB and N. We first set the initial threshold value. From Px, we reconstruct the actual pixel positions and assign the value of 1.5 times current Threshold. Then, from the knowledge of K, we separate the sign bits and the refinement bits from SRB at each level of threshold. The reconstructed values are given proper sign and updated using the sign and the refinement bits. The process is repeated for some or all specified number of threshold levels.

Decoding Algorithm:

Input: Sz→Size of the original image/data.

M→ Initial Threshold index.

N→ Desired number of threshold levels for decoding.

Px→ Difference of the significant pixels positions

K→ No of significant pixels at each threshold level.

SRB→ Sign and Refinement Bits

Output: Y→ Reconstructed or decoded image/data.

Step-1: Set Thr to the initial threshold. Assign LSP to a null array and values of Y having the size Sz to zeros.

Step-2: Reconstruct the actual pixel positions, which were significant at the current threshold, from the difference pixel positions Px and the knowledge of K.

Step-3: Append the pixel positions to LSP and delete the values from Px from which actual pixel positions are constructed.

Step-4: Assign the corresponding positions of Y obtained in step-1 with values 1.5* Thr.

Step-5: Separate the sign and refinement bits from SRB using the knowledge of K.

Step-6: Give proper sign to the reconstructed values corresponding to the positions added in the current threshold and update the values corresponding to the positions which were added in the previous thresholds properly using refinement bits.

Step-7: Decrease the threshold by a factor of 2.

Step-8: Repeat the step 2 to 7 for the specified number of times, viz, N.

5.4.3. Experimental results

We show here the compression performance of three subband based lossy compression techniques, namely SPIHT (with and without arithmetic coding), JPEG2000 and DPC, being proposed. DPC gives marginally better compression performance than SPIHT without arithmetic coding in terms of PSNR value for some images and marginally less PSNR values for some other images. However, its performance is marginally less than SPIHT with arithmetic coding and JPEG2000. The main reason is that in the proposed DPC, arithmetic coding is applied only to the differential positions; the sign and refinement bit are left uncompressed. Applying any entropy-coding scheme to SRB results in loss rather than gain because bit patterns in SRB are quite random in nature. However, it is significantly much

faster than SPIHT and JPEG2000 and requires comparatively less memory space during coding and decoding. We apply the compression schemes to four standard images, namely, Barbara, Boat, Goldhill and Lena each of size 512x512. Table 5.3 gives the PSNR values of the four images at various compression ration when compressed with DPC, SPIHT without arithmetic coding, SPIHT(AC) with arithmetic coding and JPEG2000.

Images	CR	Peak Signal to Noise Ratio (PSNR) in Decibel			
		DPC	SPIHT	SPIHT(AC)	JPEG2000
Barbara (512x512)	8:1	36.77	36.46	37.95	38.05
	16:1	31.68	31.67	32.11	32.85
	32:1	27.61	27.76	28.14	28.82
	64:1	25.03	24.98	25.38	25.74
	128:1	23.85	23.65	23.77	23.75
Boat (512x512)	8:1	38.32	38.46	39.12	39.28
	16:1	33.93	33.90	34.46	34.58
	32:1	30.65	30.48	30.97	30.99
	64:1	28.09	27.78	28.16	27.97
	128:1	25.72	25.65	25.90	25.44
Goldhill (512x512)	8:1	35.82	36.00	36.55	36.57
	16:1	32.62	32.71	33.13	33.23
	32:1	30.25	30.22	30.56	30.54
	64:1	28.04	28.27	28.48	28.38
	128:1	26.32	26.54	26.73	26.54
Lena (512x512)	8:1	39.87	40.03	40.45	40.41
	16:1	36.83	36.88	37.25	37.30
	32:1	33.67	33.72	34.14	34.12
	64:1	30.55	30.72	31.10	30.94
	128:1	27.97	28.00	28.38	27.92

Table 5.3 PSNR values of DPC, SPIHT, SPIHT(AC) and JPEG2000 at various compression ratios.

From the table, we see that the PSNR values of the DPC are marginally less than those of SPIHT and JPEG2000 in case of Goldhill and Lena images. But it is marginally better than

SPIHT for Barbara and Boat images. In most of the cases it is marginally less than SPIHT with arithmetic coding and JPEG2000. The difference in the PSNR values of DPC and SPIHT and JPEG2000 are found to higher at lower compression ratio than at higher compression ratio. It is because the sign bits and refinement bits, which are uncompressible in DPC becomes very large at the lower compression ratio. The original Barbara, Boat, Goldhill and Lena images are shown in Figures 5.4 to 5.7. The decompressed images at 32:1, 64:1 and 128:1 of each of the images corresponding to the coding schemes DPC, SPIHT, SPIHT(AC) and JPEG2000 are shown in Figures 5.8 to 5.23. From the decompressed figures, it can be observed that the image qualities of the decompressed images using DPC are better than those of SPIHT and JPEG2000, especially for Barbara and Boat images.

5.4.4. Computational time Analysis:

Both coding and decoding algorithms take constant time. However the decoding time is slightly higher than the coding time in contrast to SPIHT and JPEG2000 in which coding time is higher than the decoding time. Coding time is higher in case of SPIHT because testing the significance of a subband or subbands in a descendent tree requires testing of all the coefficients in the subbands in the corresponding descendent tree. Whereas at the time of decoding, testing the significance of a subband or a descendent tree requires testing of only a bit whether it is 1 or 0. In the case of JPEG2000, coding time is more than decoding time because coding requires post processing to achieve a target bit rate. Post processing is not required during decoding. The overall coding and decoding time are more in JPEG2000 than in SPIHT [84]. All these programs do not take much time when written in complier language like C, just a fraction of a second. Physically, it would be difficult to experience the significance of improvement in speed in terms of milliseconds or microseconds in DPC over SPIHT or JPEG200. However, when such a program is written in a technical language like MATLAB, it takes significantly different computation time. We compare the computation time between DPC and SPIHT to show the significant difference in speed between them. For this purpose, we run the program for the same number of thresholds. The number of thresholds, here, corresponds to the number of sorting passes performed during coding or decoding process. Table 5.4a shows the coding time between DPC and SPIHT at

different number of threshold levels. Table 5.b shows the decoding time between DPC and SPIHT at the corresponding number of thresholds used in the coding.

No. of Threshold	DPC	SPIHT	Times
10	0.31	10.2850	33.17
11	0.35	38.4860	109.96
12	0.40	171.15690	427.89
13	0.44	807.1210	1834.36
14	0.55	3499.1510	6350.54

Table 5.4a Comparison of coding time between DPC and SPIHT at various thresholds

No. of Threshold	DPC	SPIHT	Times
10	3.36	1.5120	0.45
11	3.50	5.6070	1.43
12	4.1960	26.2680	6.26
13	4.667	87.8470	18.82
14	5.35	471.5580	88.14

Table 5.4b Comparison of decoding time between DPC and SPIHT at various thresholds

From these tables, we see that both coding and decoding time take almost constant time even if the number of threshold levels of coding or decoding is increased. However, it is not the case in both SPIHT and JPEG2000. The coding and decoding time increases significantly with increase in the number of thresholds, with increase in the number of sorting pass. But in the case of DPC coding, the time for coding differential positions decreases with the increase in the number of threshold. The slight increase in the coding time is due to the increase in time for sending refinement bits. The number of refinement bits increases with the number of threshold levels during the coding. The decoding time is 10 times more than the coding time. However, the decoding time is also almost independent of the number of threshold levels. The increase in the decoding time as compared with the coding time is because of the increase in the number of reconstructed pixels, which are to be updated at every threshold value during decoding progresses. But it is significantly much

less than the decoding time of the SPIHT. Since SPIHT is faster than JPEG2000 [84], we can conclude DPC will be significantly faster than JPEG2000 written in MATLAB code.

5.4.5 Advantages and disadvantages of DPC

DPC is very simple algorithm as compared with the existing popular subband coding schemes. Also, it requires comparatively less memory. However, it has also certain disadvantages. We list the advantages and disadvantages of DPC as follows.

Advantages:

1. The algorithm is simple and easy to understand and implement.
2. It requires less memory space during coding and decoding as compared to SPIHT and JPEG2000.
3. The coding and decoding speed is significantly faster than the SPIHT and JPEG2000.
4. The compression performance is marginally less than the SPIHT and JPEG2000 in terms of PSNR values at compression ratio less than 32:1 but it gives higher PSNR values in some images at higher compression ratios.
5. It can be easily implemented into a fast hardware based coding scheme considering its simplicity and low memory requirement.
6. It can be used both for lossy and lossless compression.

Disadvantages:

1. It requires an entropy-coding scheme to code the differential positions of the significant pixels at various thresholds.
2. The sign and refinement bit stream are random in nature and difficult to compress.
3. It is difficult to get the exact target bit rate during the coding.

5.5. Comparison of 9/7 Wavelet filter and Some PRF-10 of ISITRA-2

The compression performance of an image compression scheme is dependent not only on the coding scheme but also on the filters used in the subband or wavelet decomposition and reconstruction scheme. Good filters give good reconstructed image. That is, the reconstructed image using good filters after decompression has less reconstruction error or higher PSNR values. There are several wavelet filters, however, the most popular one used

in lossy compression is the bi-orthogonal filters of length 10 commonly known as 9/7 filter set. This filter set has been used in most of standard subband coding schemes such as EZW, SPIHT, EBCOT and JPEG2000. In other words, the bi-orthogonal 9/7 is the accepted filter set for use in subband lossy image compression. Our attempt here is to show that using the flexibility in choosing filter coefficients in ISITRA, it would be possible to find better filter sets than the popular 9/7 filter set for use subband image compression. We suggest two new PRF sets of ISITRA-2 of length-10, which are designated respectively as PRF2.1, and PRF2.2. These PRF sets are found by using genetic algorithm based on the MSE or PSNR values, either of which is a suitable criterion for finding a suitable PRF set for used in image compression. The criteria may be different for different applications. We consider those PRF sets, which give smaller MSE or higher PSNR values than the 9/7 filter set when an image is reconstructed from only the approximation component in a single level decomposition as described in Appendix C. The decomposition and reconstruction filter sets of the popular 9/7, PRF2.1 and PRF2.2 PRF sets are respectively given in Table 5.5a, 5.5b and 5.5c. The coefficients of the filters in the two PRF sets are significantly different from the coefficients of the filers in the bi-orthogonal 9/7 filter set. We compare the compression performance of these length 10 filter set of ISITRA-2 with 9/7 filter set using SPIHT coding. The reason why we choose SPIHT coding for the performance evaluation of the filters in Table 5.5 is that it is popular and it is not very difficult to write a SPIHT program to test any new PRF set. The compression performance of these PRF sets (Table 5.5) on Barbara, Boat, Goldhill and Lena images, each of size 512x512, at various compression ratios is given in Table 5.6. The compression performance of PRF2.1 and PRF2.2 is poor for Goldhill and Lena images especially at higher compression ratio. However, it can be seen from Table 5.6 that the newly proposed PRF sets (PRF2.1, PRF2.2) perform better for Barbara and Boat images at a particular compression ratio. This suggests that the bi-orthogonal 9/7 filter set is not the best filter set to compress all kinds of images using a subband based lossy compression scheme.

Decomposition filters		Reconstruction filters	
fa	**f**b	**f**ar	**f**br
0	0	0	0
0.03782845550726	-0.06453888262870	-0.06453888262870	-0.03782845550726
-0.02384946501956	0.04068941760916	-0.04068941760916	-0.02384946501956
-0.11062440441844	0.41809227322162	0.41809227322162	0.11062440441844
0.37740285561283	-0.78848561640558	0.78848561640558	0.37740285561283
0.85269867900889	0.41809227322162	0.41809227322162	-0.85269867900889
0.37740285561283	0.04068941760916	-0.04068941760916	0.37740285561283
-0.11062440441844	-0.06453888262870	-0.06453888262870	0.11062440441844
-0.02384946501956	0	0	-0.02384946501956
0.03782845550726	0	0	-0.03782845550726

Table 5.5a Decomposition and reconstruction filters for 9/7 PRF set.

Decomposition filters		Reconstruction filters	
fa	**f**b	**f**ar	**f**br
0	0	0	0
0.04902540842008	-0.07503948850118	-0.07503948850118	-0.04902540842008
-0.02423517392248	0.03709494960852	-0.03709494960852	-0.02423517392248
-0.13660532123680	0.41683159914039	0.41683159914039	0.13660532123680
0.42784667322751	-0.75756664772082	0.75756664772082	0.42784667322751
0.83101341748950	0.41683159914039	0.41683159914039	-0.83101341748950
0.42784667322751	0.03709494960852	-0.03709494960852	0.42784667322751
-0.13660532123680	-0.07503948850118	-0.07503948850118	0.13660532123680
-0.02423517392248	0	0	-0.02423517392248
0.04902540842008	0	0	-0.04902540842008

Table 5.5b Decomposition and reconstruction filters for PRF2.1 PRF set.

Decomposition filters		Reconstruction filters	
fa	**f**b	**f**ar	**f**br
0	0	0	0
0.04902540842008	-0.07503948850118	-0.07503948850118	-0.04902540842008
-0.02423517392248	0.03709494960852	-0.03709494960852	-0.02423517392248
-0.13660532123680	0.41683159914039	0.41683159914039	0.13660532123680
0.42784667322751	-0.75756664772082	0.75756664772082	0.42784667322751
0.83101341748950	0.41683159914039	0.41683159914039	-0.83101341748950
0.42784667322751	0.03709494960852	-0.03709494960852	0.42784667322751
-0.13660532123680	-0.07503948850118	-0.07503948850118	0.13660532123680
-0.02423517392248	0	0	-0.02423517392248
0.04902540842008	0	0	-0.04902540842008

Table 5.5c Decomposition and reconstruction filters for PRF2.2 PRF set.

PRF sets	CR	Peak Signal to Noise Ratio (PSNR) in Decibel			
		Barbara	Boat	Goldhill	Lena
9/7 filter	8:1	36.46	38.46	36.00	40.03
	16:1	31.67	33.90	32.71	36.88
	32:1	27.76	30.48	30.22	33.72
	64:1	24.98	27.78	28.27	30.72
	128:1	23.65	25.65	26.54	28.00
PRF2.1	8:1	36.95	38.60	35.97	40.07
	16:1	31.70	33.88	32.60	36.80
	32:1	27.68	30.41	30.11	33.54
	64:1	24.98	27.77	28.08	30.47
	128:1	23.74	25.67	26.27	27.73
PRF2.2	8:1	37.04	38.60	36.01	40.10
	16:1	31.79	33.95	32.64	36.88
	32:1	27.77	30.49	30.12	33.63
	64:1	25.09	27.75	28.01	30.54
	128:1	23.78	25.69	26.11	27.73

Table 5.6 Compression performance of decomposition and reconstruction filters of 9/7, PRF2.1, and PRF2.2 PRF sets on Barbara, Boat, Goldhill, and Lena images of size 512x512 at various compression ratios.

Figure-5.4 :
Original Barbara

Figure-5.5 :
Original Boat

Figure-5.6:
Original Goldhill

Figure-5.7:
Original Lena

Figure-5.8(a): Decompressed Barbara image at 32:1 using DPC.

Figure-5.8(b): Decompressed Barbara Image at 64:1 using DPC

Figure-5.8(c): Decompressed Barbara image at 128:1 using DPC.

Figure-5.9(a): Decompressed Boat image at 32:1 using DPC

Figure-5.9(b): Decompressed Boat image at 64:1 using DPC

Figure-5.9(c): Decompressed Boat image at 128:1 using DPC

Figure-5.10(a): Decompressed Goldhill image at 32:1 using DPC

Figure-5.10(b): Decompressed Goldhill image at 64:1 using DPC

Figure-5.10(c) : Decompressed Goldhill image at 128:1 using DPC

Figure-5.11(a): Decompressed Lena image at 32:1 using DPC.

Figure-11(b): Decompressed Lena image at 64:1 using DPC.

Figure-11(c): Decompressed Lena image at 128:1 using DPC.

Figure-5.12(a): Decompressed Barbara image at 32:1 using SPIHT

Figure-5.12(b): Decompressed Barbara image at 64:1 using SPIHT

Figure-5.12(c): Decompressed Barbara image at 128:1 using SPIHT

Figure-5.13(a): Decompressed Boat image at 32:1 using SPIHT

Figure-5.13(b): Decompressed Boat image at 64:1 using SPIHT

Figure-5.13(c): Decompressed Boat image at 128:1 using SPIHT

Figure-5.14(a): Decompressed Goldhill image at 32:1 using SPIHT

Figure-5.14(b): Decompressed Goldhill image at 64:1 using SPIHT.

Figure-5.14(c): Decompressed Goldhill image at 128:1 using SPIHT

Figure-5.15(a): Decompressed Lena image at 32:1 using SPIHT.

Figure-5.15(b): Decompressed Lena image at 64:1 using SPIHT

Figure-5.15(c): Decompressed Lena image at 128:1 using SPIHT.

Figure-5.16(a): Decompressed Barbara image at 32:1 using SPIHT(AC).

Figure-5.16(b): Decompressed Barbara image at 64:1 using SPIHT(AC)

Figure-5.16(c): Decompressed Barbara image at 128:1 using SPIHT(AC)

Figure-5.17(a): Decompressed Boat image at 32:1 using SPIHT(AC)

Figure-5.17(b): Decompressed Boat image at 64:1 using SPIHT(AC)

Figure-5.17(c): Decompressed Boat image at 128:1 using SPIHT(AC)

Figure-5.18(a): Decompressed Goldhill image at 32:1 using SPIHT(AC)

Figure-5.18(b): Decompressed Goldhill image at 64:1 using SPIHT(AC)

Figure-5.18(c): Decompressed Goldhill image at 128:1 using SPIHT(AC)

Figure-5.19(a): Decompressed Lena image at 32:1 using SPIHT(AC)

Figure-5.19(b): Decompressed Lena image at 64:1 using SPIHT(AC)

Figure-5.19(c): Decompressed Lena image at 128:1 using SPIHT(AC)

Figure-5.20(a): Decompressed Barbara image at 32:1 using JPEG2000

Figure-5.20(b): Decompressed image at 64:1 using JPEG2000

Figure-5.20(c): Decompressed Barbara image at 128:1 using JPEG2000

Figure-5.21(a): Decompressed Boat image at 32:1 using JPEG2000

Figure-5.21(b): Decompressed Boat image at 64:1 using JPEG2000

Figure-5.21(c): Decompressed Boat image at 128:1 using JPEG2000

Figure-5.22(a): Decompressed Goldhill image at 32:1 usingJPEG2000

Figure-5.22(b): Decompressed Goldhill image at 64:1 using JPEG2000

Figure-5.22(c): Decompressed Goldhill image at 128:1 using JPEG2000.

Figure-5.23(a): Decompressed Lena image at 32:1 using JPEG2000.

Figure-5.23(b): Decompressed Lena image at 64:1 using JPEG2000.

Figure-5.23(c): Decompressed image at 128:1 using JPEG2000.

5.6 Meitei Lock Sequence

Meitei Lock Sequence (MLS) is a random sequence generated from an array. It has very interesting properties useful for encryption. The array from which the sequence is generated is called the key array or, simply, the key. The elements of the key array can be chosen quite arbitrarily. It is empirically found when there is any difference in any of the elements of the key, the sequence generated is also different. That is, the sequence is unique in the sense that no two different keys can generate the same MLS. Another important feature of MLS is that any sequence generated from an array is empirically found to be random with no sign of periodicity. So far we have tested for 400-million length MLS's for periodicity by taking various keys of length 2, 3, 4 and 5. We found no trace periodicity and it is almost impossible to know the exact array from any section of the random sequence, an important property that will be useful in encryption. Moreover, the length of the key is arbitrary. This will increase the complexity in the search of the right key in the key space. Thus, this offers a reasonably secure encryption method. Another characteristic is that a longer key can generate an MLS faster. There may be many forms of MLS generator. But we present here three such forms. The MLS generator is algorithmically much simpler than the pseudo random number generators used for various cryptographic applications [10].

Purely non-deterministic random number sequences cannot be used for encryption. However, it can be used for random key generation and distribution [131]. More on the random number generators and their applications can be found in [68, 85, 86, 135]. To be useful for encryption, random number generators should have the following properties. It should be able to produce a unique random sequence. The period of such a sequence should be infinitely long. The sequence should not exhibit any peculiar features from which the keys can be deduced. MLS exhibits such desired properties for encryption.

5.6.1 Generation of meitei lock sequence

Encryption is analogous to lock and key property. In this analogy, if we can generate a sequence from an array and is unique for that array, then we can use the sequence as the lock sequence and the generating array as the key. If we can generate such a sequence, the sequence will be known as meitei lock sequence (MLS). In other words, an MLS is a

sequence generated from an arbitrarily chosen non-zero array and is unique for that array. If we denote an MLS generated from an array $\mathbf{x}$ by $MLS(\mathbf{x})$, we can express its uniqueness as follows.

For any two non-zero arrays $\mathbf{x}_1$ and $\mathbf{x}_2$, $MLS(\mathbf{x}_1) \neq MLS(\mathbf{x}_2),\ \forall\ \mathbf{x}_1 \neq \mathbf{x}_2 \quad \cdots\cdots(6)$.

This is the main property of an MLS. It ensures that no two distinct arrays can generate the same MLS, a desired property for tight encryption.

An M operator is defined as a mapping from $\Re^l$ to $\Re^l$, for any positive integer *l*. Then an MLS can be generated from it by recursively operating on it. The uniqueness of an MLS depends the uniqueness of *M*. For a given array $\mathbf{x}$ of length l, an MLS is a sequence of length $l \times t$, generated by using the following relation.

$$MLS(\mathbf{x}) = \{M(\mathbf{x}), M^2(\mathbf{x}), \cdots, M^t(\mathbf{x})\} \quad \cdots\cdots(7)$$

For a given l, one can generate a sequence of any length by choosing t properly. Here the array $\mathbf{x}$ is the key and the sequence $MLS(\mathbf{x})$ is the lock sequence. Note that each distinct key $\mathbf{x}$ will generate a unique lock sequence $MLS(\mathbf{x})$ when $M(\mathbf{x}_1) \neq M(\mathbf{x}_2),\ \forall\ \mathbf{x}_1 \neq \mathbf{x}_2$.

In this application, the elements of the key array are taken as strictly positive. Consequently, each element in the generated MLS is also positive. While generating an MLS using equation (7), one need not generate a lock sequence of the same length as that of the component to be encrypted. Instead one can generate a shorter MLS and then replicate it successively to make its length equal to the length of the signal to be encrypted. This will save time in generating very long MLS's.

5.7 Some *M* operators

We present here three forms of *M* operator. These three operators have been empirically found to be unique and non-periodic.

I. $M_1 : \mathbf{y}(i) = S / \mathbf{x}(i) + S\%\mathbf{x}(i)$, $i = 1,2,\cdots,l$ and $S = \sum_{i=1}^{l} \mathbf{x}(i)$

II. $M_2 : S/\{\mathbf{x}(i) + \mathbf{x}(i+1)\} + S\%\{\mathbf{x}(i) + \mathbf{x}(i+1)\}$, $i = 1,2,\cdots,l; \mathbf{x}(l+1) = 0$

III. $M_3 : N_1(M_1(\mathbf{x}))$; where $N_1 : \mathbf{y}(i) + \mathbf{y}(i+1),\ \mathbf{y}(l+1) = 0$

That is, for *l*=4, the *i*-th component of $M_1(\mathbf{x})$ is $\mathbf{y}(i) = S / \mathbf{x}(i) + S\%\mathbf{x}(i)$ for $\boldsymbol{i}$ = 1, 2, 3, 4. The *i*-th component of $M_2(\mathbf{x})$ is $\mathbf{z}(i) = S/\{\mathbf{x}(i) + \mathbf{x}(i+1)\} + S\%\{\mathbf{x}(i) + \mathbf{x}(i+1)\}$ for $\boldsymbol{i}$ = 1, 2, 3 and $\mathbf{z}(4) = S / \mathbf{x}(4) + S\%\mathbf{x}(4)$. The *i*-th component of $M_3(\mathbf{x})$ is $\mathbf{w}(i) = y_i + y_{i+1}$ for i = 1, 2, 3 and $\mathbf{w}(4) = \mathbf{y}(4)$.

All the three generators are found to be useful in generating satisfactory MLS's. We briefly study the uniqueness and periodicity of the first MLS generator. The same logic can be extended for the other generators as well.

5.7.1 Uniqueness of an MLS

We study here the uniqueness of M_1 operator. Uniqueness of an MLS is dependent on the uniqueness of its generator. The desired property is : $M_1(\mathbf{x}_1) \neq M_1(\mathbf{x}_2)$ for any two distinct arrays $\mathbf{x}_1$ and $\mathbf{x}_2$. However, it is very complicated to prove this property of M_1 operator. From an empirical study we find that unless $\mathbf{x}(i)$ exactly divides *S*, M_1 operator is unique.

Uniqueness of an MLS generator directly relates with the tightness of an encryption. If we are to generate an MLS from a key array using the first generator for use in encryption, the key array should be such that the sum is not divisible by each of its elements. Instead of checking whether the sum of an array is divisible by each of its element, one easy way of obtaining such an array is to make the number of odd elements in the key array odd and to make at least one element even. This would make the sum of the array elements an odd number which will not be divisible by an even number.

The 2nd and the 3rd generators are better in terms of uniqueness. We do not require such restrictions in these two generators as in the first generator. The restriction in the second generator is that all the elements in the key arrays should not be the same. For such types of arrays, the MLS's generated are in a stair case form, i.e., the sequences become non-random in nature. So, we should avoid choosing key arrays whose elements have the same value, if we are to generate an MLS using the 2nd generator. The 3rd generator has no such restriction on the choice of key arrays. In other words, the MLS's generated by the 3rd generator are more likely to be unique. The range in the long run of an MLS from the first generator is

from 2 to 30 for l = 4. For higher values of l the range becomes proportionately larger. The ranges of MLS's generated by the 2nd and the 3rd generators are much larger, which may not be desirable in some applications and hence we usually restrict the maximum value within a desired range using a modulus operator. Also, while generating the MLS using equation (7) for signal encryption, one need not generate the lock sequence to be equal to the length of the component to be encrypted. Instead we can generate a shorter MLS and then replicate it successively to make its length equal to the length of the signal to be encrypted. This will save time generating longer MLS.

5.7.2 Periodicity of an MLS

Theoretically, it is very difficult to estimate the periodicity in an MLS. Unlike the usual pseudo random number sequences, MLS's are found to be aperiodic. We have empirically studied its periodicity in an MLS by generating several MLS's of length 400 million. Interestingly, we have found none of MLS's to be periodic. This is also another desirable property so far as encryption is concerned as this will make cipher text attack more difficult. Most of the random number generators [19, 73, 75] are periodic. From the popular random number generator [73], we know that the periodicity is dependent on the divisor of the modulus operator δ in the following equation.

$$s_{n+1} = (as_n + c)\%\delta \quad \cdots\cdots\cdots(8)$$

The random sequence $\{s_n\}$ obtained from (8) cannot have a value greater than δ for integers a, c and δ. Such a random number generator cannot generate a sequence having a period greater than δ. The larger the periodicity the more is the randomness in the sequence. The usual practice is to keep the value of δ as high as possible to get a random sequence with a longest possible period. The longer the period, the better the random number generator is [56, 94]. But we cannot generate a random number sequence with a very large period in a computer because of its fixed byte size. For a 32 bit machine, we cannot make the value of δ greater than 2^{31}. That is what is done in IBM-360 random number generators [75]. The periodicity is also dependent to some extent on the choice of the parameters a and c. For IBM-360, the values of the parameters are $a = 7^5$ and $c = 0$, in addition to $\delta = 2^{31} - 1$.

Now the question is if the value of δ keeps changing randomly, can we have a random sequence of infinite period? That would be quite possible. The main reason why we get random sequence of almost infinite period in the case of our MLS can be attributed to the random variation in the value of the divisor of the modulus operator.

Some MLS's are shown in Figs.5.24 and 5.25 just to show the difference in the MLS with a slight change in any element of a key. Figs.5.24a and Fig.5.24b show two MLS's obtained from the same 1st generator. But the MLS's are quite different with a slight change in the second element of the key. Similarly, the MLS's in Figs.5.25a and Fig.5.25b are also different even when they are obtained from the same 2nd generator because of a slight difference in the second element of the key. Also, note that the maximum values of the MLS's in the case of 1st generator are less than 20 while the maximum values of the MLS's in the case of 2nd generator go much beyond 60 though we have restricted the values of the sequences within 60.

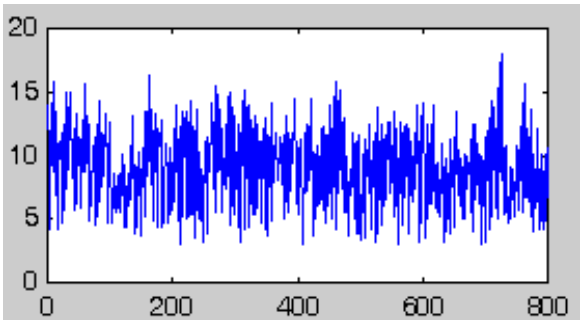


Fig.5.24a MLS for the key [2 3 10 13]

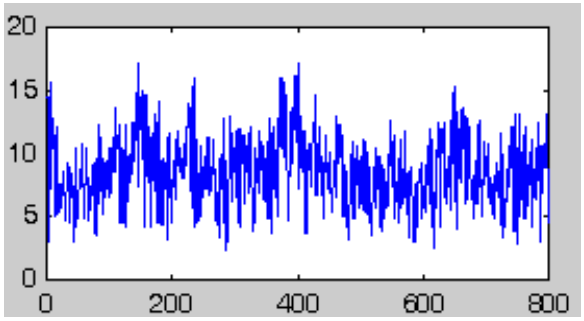


Fig.5.24b MLS for the key [2 2 10 13]

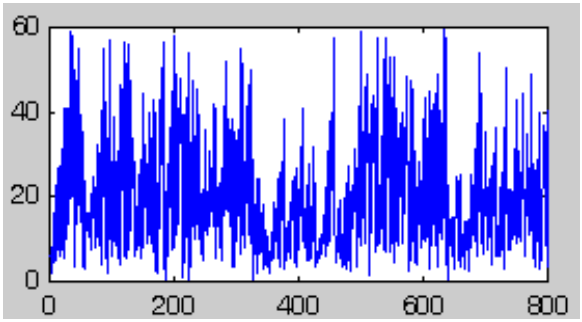


Fig.5.25a MLS for the key [2 3 10 13]

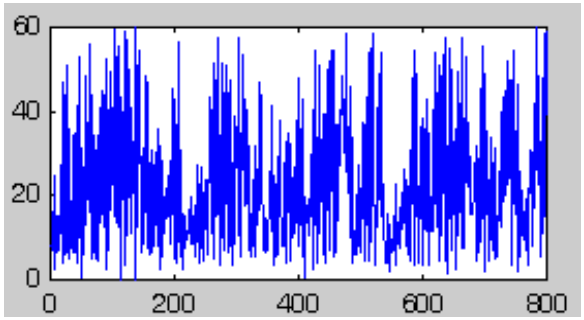


Fig.5.25b MLS for the key [2 2 10 13]

5.8 Signal Encryption using Simplet and MLS.

Encryption is generally used for protection of privacy and secure communication. The logic behind any encryption technique is to hide the information content in the signal so that no interceptors can understand and retrieve the content of the signal. For speech and image data, encryption is performed by modifying the original data usually in three different ways, namely, position permutation [21], value transformation [25, 165] and combination of both [32, 129]. Some of them have been reported to be insecure in [27, 62, 76]. The proposed encryption method here is a value transformation process. One way to hide the information content in the signal is to distort the signal such that it is perceptually unintelligible. Signal distortion can be done using infinite length type YKSK transform. Here we use equal and even infinite length Simplet, a special case of YKSK-2 transform for signal distortion. Because it is simple and fast as its decomposition scheme involves only addition and subtraction operations and its reconstruction process is also simple and fast. However, Simplet itself is not sufficient for encryption of a signal as the original signal can be easily reconstructed from its distorted components. In other words, Simplet does not have the lock and key property of an encryption scheme. For every encryption technique there should also be a provision for decryption of the signal by the intended users. Encryption is usually done with a key (password) to hide the information content in a signal. The decryption of the signal from the encrypted signal is done using the same key or a key that can be derived from it. The key is kept secret between the sender and the receiver for secure communication. For secret data storage, only the encryptor knows the key. The tightness of the encryption depends on how difficult the search process for the unknown key in the key space, is. An ideal encryption key should be such that the search for the key is computationally prohibitive. For any encryption scheme, the more difficult to crack the key the better it is. Hence the main criteria for any encryption technique are the tightness of the key and the decryptibility using the exact key. Ours is a symmetric private key cryptosystem where the encryption and the decryption keys are the same. The locking and unlocking in our encryption scheme are performed using an MLS generated from the 3rd generator.

We take advantage of the ability of infinite length Simplet for fast signal distortion purpose, which is important from the point of view of protection of signal contents. After making the signal perceptually (visually in the case of images) unintelligible in its decomposed

components, we use MLS for encrypting it using an arbitrary key which is a non-zero array of arbitrary length. An MLS is found to be a unique and non-periodic sequence for any given key (array). This property of an MLS is made use of in the encryption and decryption schemes.

In the existing subband decomposition scheme such as DWT, there is an approximation component among the decomposed components that is perceptually similar to the original signal. The approximation component, even after applying an MLS to it, largely remains perceptually similar to the original signal. Thus, an existing subband decomposition scheme along with an MLS cannot be used for encryption, since the contents of the original signal is not protected. Transforms like DFT (discrete Fourier transform), DCT (discrete cosine transform) can act as a distorter, but their reconstruction scheme is not suitable for encryption.

Simplet has the property that if there is noise in the decomposed components, the reconstructed signal will be unintelligibly different from the original signal. But the existing transforms such as DWT, DFT, DCT do not have this property. That is, even if there is significant noise in the transform coefficients, the reconstructed signal is perceptually intelligible to a great extent, which is demonstrated with examples in Subsection 5.8.2. Consequently, such transforms are not suitable for the purpose of encryption.

5.8.1 Tightness of key array

Any given MLS can be generated if and only if the key is known. Thus, the complexity of an attack is more for larger spaces of the key. The length l of the key (array) here is arbitrary. If each element consists of four bytes, then the key array will have $32l$ bits. The size of the search space becomes 2^{32l}. Hence the complexity of an attack is 2^{32l} which can be made significantly large by choosing a large value of l. Thus, this offers a reasonably secure encryption method. Several encryption methods are available to protect the content of digital images. Some are known to be insecure [9, 25, 165]. It is desirable that the complexity of an attack for a reasonably efficient encryption method, be not lower than 2^{128} [77].

In some methods of encryption, it is possible to construct a low quality approximation of the original image on the basis of a fewer bits in the key. This is not possible in the present scheme since any slight change in any element of the key leads to a large error in the reconstructed signal. This is because a slight change in any of the key elements generates a different MLS which will introduce errors in the decrypted components.

5.8.2 Experimental results

Our encryption scheme is therefore a two-stage process as shown in Fig.5.26a. First we distort a signal using equal and even infinite length (EEIL) Simplet. The distorted signal is then encrypted using an MLS. Encryption here is simply the sum of the MLS and the distorted signal components. Decryption scheme is a two-stage reverse process as shown in Fig.5.26b. It first decrypts the encrypted components using the MLS (obtained from the known key), which is nothing but subtraction of the MLS from the encrypted components and then reconstructs the original signal from the decrypted components using the reconstruction method of equal and even infinite length Simplet.

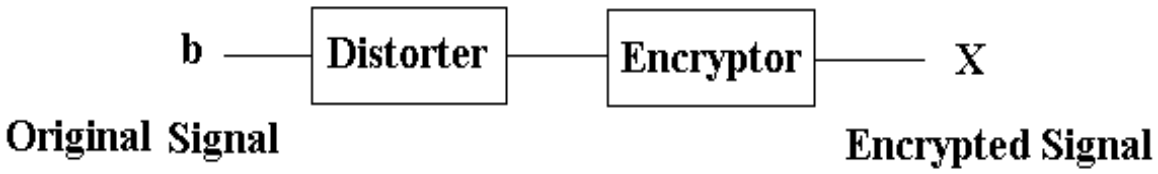


Figure 5.26a Encryption model

(Distorter: Decomposition using EEIL-Simplet; Encryptor: Distorted Components + MLS)

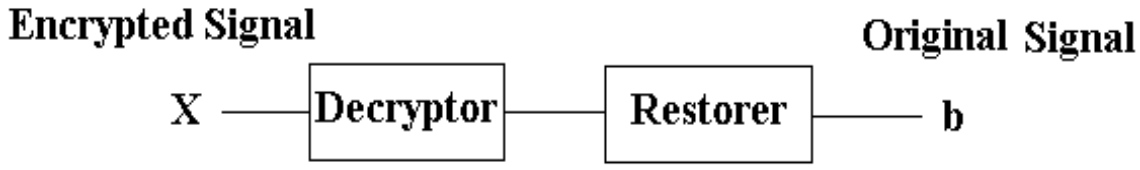


Figure 5.26b Decryption model

(Decryptor: Encrypted Components – MLS; Restorer: Reconstruction using EEIL-Simplet.)

Some examples of MLS's of the 3rd generator are shown in Fig.5.27 for four slightly different keys. Thus, even a slight change in the key leads to significant changes in MLS. An original chirp signal and the chirp signal plus MLS corresponding to the key array [7 121 7 3 33] are shown in Fig.5.28a. From Fig.5.28a, we see that MLS itself is not enough to hide

the information content in the chirp signal. The distorted decomposed components of the original chirp signal using Simplet are shown in Fig.5.28b. The encrypted components using MLS ([7 121 7 3 33]) are shown in Fig.5.29a. From these figures we see that the encrypted components are considerably different from the original chirp signal. We now examine the reconstructed signals from the encrypted components using two decryption key arrays that are very close, but not equal, to the actual encryption key array. In Fig.5.29b, we can see that the reconstructed signals differ considerably from the original chirp signal. There have been similar findings in the case of image encryption. An original image is shown in Fig.5.30a. The original image plus MLS ([7 121 7 3 33]) is shown in Fig.5.30b in which features of the original image are identifiable. The distorted decomposed components using Simplet are shown in Fig.5.31a and the corresponding encrypted components with the same MLS are shown in Fig.5.31b. The reconstructed images using three decryption key arrays that are very close (but not equal) to the actual encryption key array are shown in Fig.5.32. They are considerably different from the original image. We also show the correlation coefficients between the original signal on one hand and the encrypted components and reconstructed signals on the other, using various encryption and decryption key arrays in Tables 5.7 and 5.8. The correlation coefficient is a measure of similarity between two signals. A higher absolute value of the correlation coefficient means more similarity and a lower such value means less similarity. From the tables it can be seen that the correlation coefficients are negligibly small. From Table 5.7, one can see that the encrypted signals have little resemblance with the original signal irrespective of the keys used for encryption. For example, the correlation coefficients between the original chirp signal in Fig.5.28a and two encrypted components (Fig.5.28b) using the key [7 121 7 3 33] are 0.004752 and 0.003810 respectively. Similarly, the correlation coefficients between the original image in Fig.5.30a and four encrypted components (Fig.5.310b) using the same key [7 121 7 3 33] are 0.004202, -0.000454, 0.007425 and -0.000454 respectively.

From Table 5.7, it is clear that the reconstructed signals also have little resemblance with the original signals when the decryption key array differs (however marginally) from the encryption key array. For example, the correlation coefficients between the original chirp signal in Fig.5.28a and two reconstructed signals (Fig.5.29b) are -0.011583 and -0.001081 respectively. Similarly, the correlation coefficients between the original image in Fig.5.30a

and the three reconstructed images (Fig.5.32) are –0.009910, -0.004167 and –0.010558 respectively. This shows that if the decryption key is not exactly equal to the encryption key, it is almost impossible to get a distinguishable reconstructed signal corresponding to the original image. In other words, the encryption scheme is considerably tight.

An existing subband decomposition scheme, where usually an approximation component is present among the decomposed components, is not suitable here for the purpose of encryption. In Fig.5.33 examples are given where the image in Fig.5.30a is decomposed using Daubechies wavelet filter of length four (DB2) and two different MLS are added to the decomposed image. The features of the original image are visible in the encrypted images in Fig.5.33. The discrete cosine transform coefficients of the image in Fig.5.30a, when added to the MLS ([7 121 7 3 33]), is shown in Fig.5.34a. This is perceptually unintelligible. The same is true for discrete Fourier transform shown in Fig.5.35a. However, we can get perceptually intelligible reconstructed images from the encrypted images in Fig5.34a and Fig.5.35b even by using different MLS's. The reconstructed images using the MLS ([60 11 8 25 120]), are respectively shown in Fig.5.34b and Fig.5.35b. Thus the existing transforms cannot be used as a distorter in our encryption scheme.

5.9 Discussion and Remarks

In this chapter we have presented some applications of ISITRA and YKSK transforms. Here, we use ISITRA for lossy image compression and YKSK transform for signal encryption. In lossy image compression, the decompressed signal is not exactly equal to the original image. Some kinds of error measurements, usually MSE and PSNR values, are used for performance evaluation of any lossy image compression scheme. The lower the MSE value or higher the PSNR value, the better is the quality of the decompressed image. However, MSE and PSNR values cannot reflect the quality of a decompressed image. As no other better alternatives of error measure are available yet, we also use MSE and PSNR values as error measures.

Any subband decomposition scheme itself is not sufficient for image compression. For efficient image compression, we need good coders that will code the subbands efficiently. SPIHT and JPEG200 are standard subband coding schemes popularly used for lossy image compression in the recent years. SPIHT is much simpler in its coding scheme and it can be

used with or without any entropy coding schemes. Also, it can achieve exact compression rate. JPEG2000 as compared with SPIHT is more complex in its coding scheme and also it is difficult to get exact compression rate. To control the target compression rate extra post-processing operations are to be performed which slows down the coding part in JPEG2000. The betterment of compression performance of JPEG200 over SPIHT is only marginal. SPIHT can be improved in its coding scheme by reducing the number of searching operations and the memory requirement during the sorting pass. We propose ISPIHT as an improvement of SPIHT in its coding scheme. ISPIHT is found to be faster than SPIHT by about 3 to 5 times during coding. We also proposed another coding scheme DPC which is very simple, fast and memory efficient as compared with SPIHT and JPEG2000. However, its performance is marginally less than SPIHT and JPEG2000. It requires no buffer memory during coding and decoding and the number of search operations during coding is significantly less. As a consequence, time complexity and the memory requirement is significantly reduced in DPC than in SPIHT and JPEG2000.

The performance of a subband image compression scheme is dependent not only on the coding schemes used but also on the sets of filters used during decomposition and reconstruction. We suggest two sets of filters of ISITRA-2 of length 10 and compare their performance with the commonly used bi-orthogonal 9/7 wavelet filter set for lossy image compression. It is found that the proposed filters sets give better performance in some images than the bi-orthogonal 9/7 filter set. It indicates that bi-orthogonal 9/7 filter set is not suitable to compress all kinds of images.

As for encryption, we know that infinite length type YKSK transform can be used for signal distortion. Either of YKSK-1 of YKSK-2 transforms can be used for signal distortion purpose. But we used equal and even infinite length Simplet a special form of YKSK-2 transform because of its simplicity and ability to distort a signal fast. Simplet is a very simple transform based on addition and subtraction. Infinite length Simplet is useful for signal distortion. However, for sufficiently large length, fixed-length Simplet also can attain significant signal distortion (Fig. 5.36). The decomposition and reconstruction schemes in this case are much simpler and faster. Simplet being a perfect reconstruction transform is not of much use in encryption unless its distorted components are protected, because one can

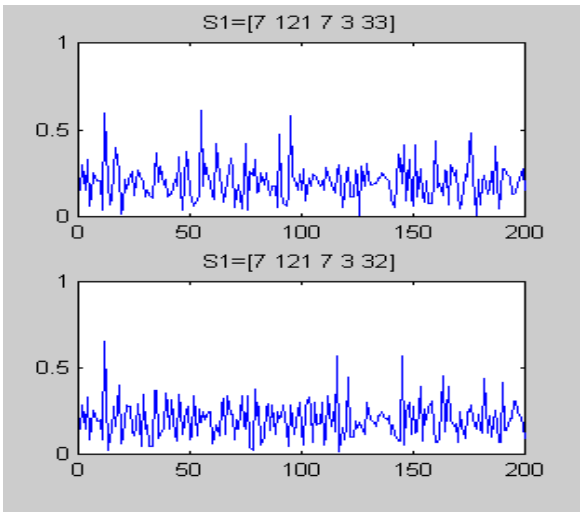
easily get back the original signal from its distorted components. To protect the distorted components from using by the unintended users we developed a random sequence known as meitei lock sequence (MLS). An MLS is a random sequence generated from an arbitrarily chosen array known as key vector. Theoretically, an MLS is unique for a key vector. That is no-two different key vectors should be able to produce MLS. We have proposed three MLS generators whose uniqueness issues are discussed well. Another interesting feature of MLS is that it shows no periodicity. We have tested the periodicity of various MLS of lengths more than 400 million and we are yet to find any sign of periodicity in any of the MLS. Because of the uniqueness and aperidicity features of MLS, it is clear that MLS can be used for secure signal encryption.

The encryption scheme proposed here is a new scheme in speech and image encryption. We use a distorter to distort the signal beyond perceptual intelligibility and then an encryptor for encrypting the distorted signal. Simplet first distorts the signal and then the distorted signal is encrypted using an MLS. An MLS sequence itself is not sufficient to protect features of the original signal as seen in Figs.5.28(a) and 5.30(b). It is used just for locking the distorted components safely so that an interceptor cannot reconstruct the original signal using the reconstruction scheme of Simplet. The role of the MLS is to cause error in the decomposed components prior to reconstruction process, if there is any small discrepancy in the decryption key. This is why the reconstructed signal is perceptually uncorrelated with the original signal if there is any small difference in an element of the decryption key vector.

We also provide the correlation coefficients between the original signal and its decrypted signal with various slightly different key vectors. It is found that the correlation is negligibly small when there is a slight difference between encryption and decryption key vectors. This indicates that it is not possible to get a distinguishable signal from its encrypted components in our encryption scheme unless the decryption key is exactly equal to the encryption key. As the key vector can be of any length and can be chosen arbitrarily, the search space for the exact key vector is infinitely large. In other words, it provides very tight security. The security would even be more, if we use infinite length type of YKSK transforms for signal distortion in our encryption scheme. Because it would amount to double locking systems; the filters of the YKSK transform would act as first lock and the key of the MLS as the second lock.

5.10 Conclusions

From the various compression results, we see that employing the flexibility of choosing filter coefficients in ISITRA and YKSK transform, it is possible to get better PRF sets than the commonly used PRF sets for image compression. The same may be true for other applications as well. The ISPIHT can be uses as a standalone coding scheme without using any entropy coding. DPC can be used preferably where the speed is of great concern than the compression ratio. Both can also be used for lossless coding with slight modification in the refinement pass. Also, meitei lock sequence is a new and interesting random number sequence with no sign of periodicity. As an MLS can be generated from any arbitrary non-zero array and as each MLS is found to be unique for any key array. It is very difficult to find the exact array through searching. Hence, the signal encryption scheme based on infinite length Simplet and meitei lock sequence is a tight encryption scheme.



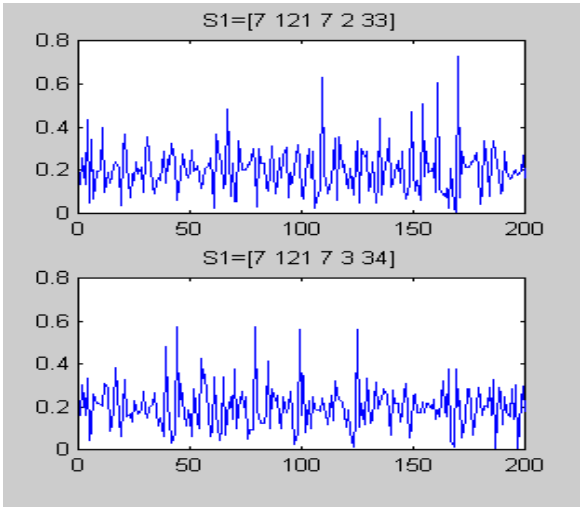


Figure 5.27 MLS's generated from key arrays [7 121 7 3 33], [7 121 7 3 32], [7 121 7 2 33] and [7 121 7 3 34]

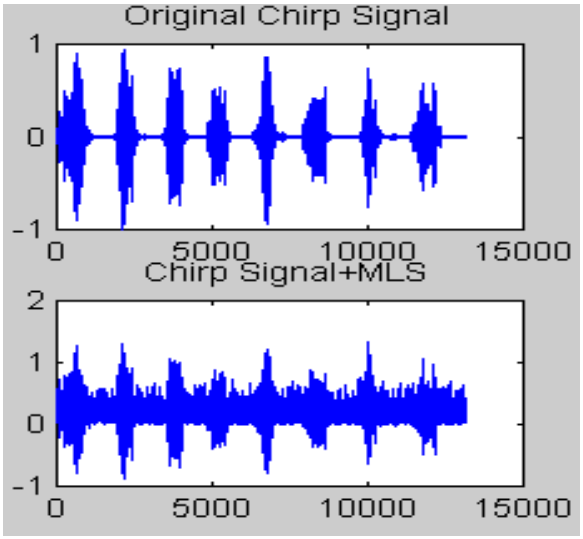
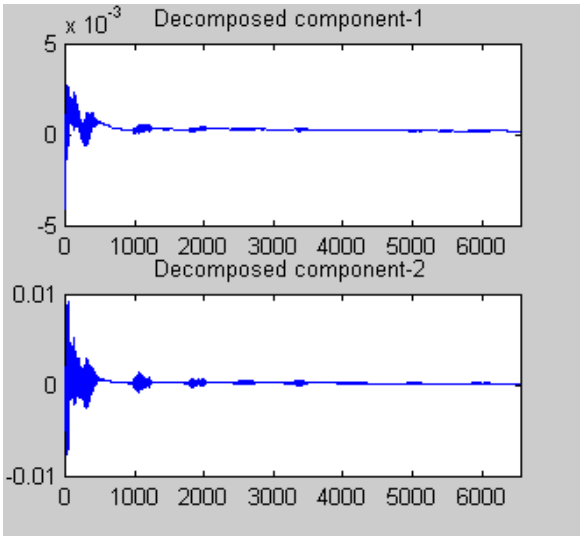
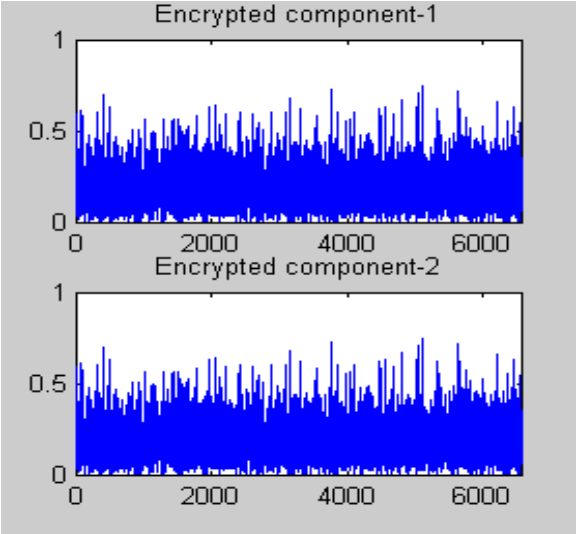


Figure 5.28a Original chirp signal and the chirp plus MLS



5.28b Decomposed components of the chirp signal using EEIL-Simplet



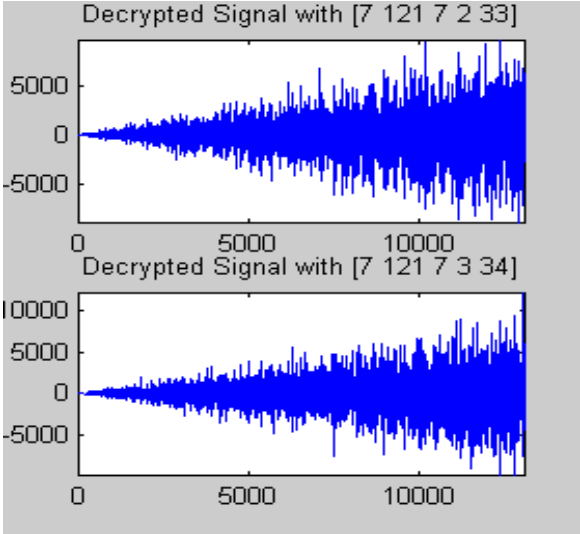


Figure 5.29a Encrypted components using key array [7 121 7 3 33]

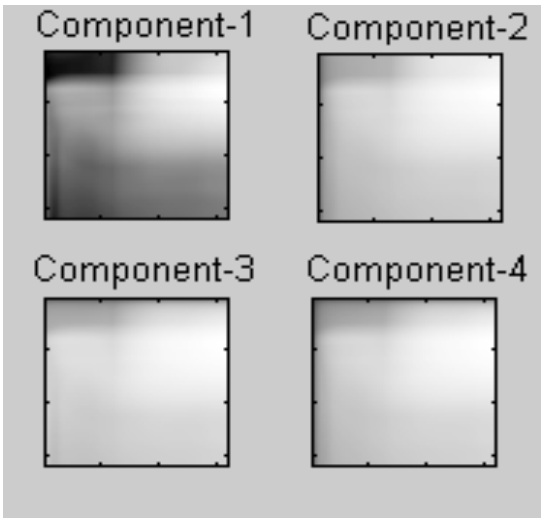
5.29b Reconstructed signals using key arrays [7 121 7 2 33] and MLS [7 121 7 3 34]

(a)

(b)

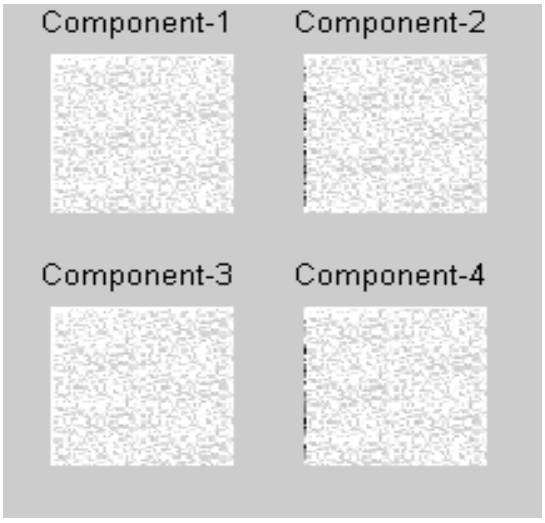
Figure 5.30a Original image

5.30b Original image plus MLS ([7 121 7 3 33])



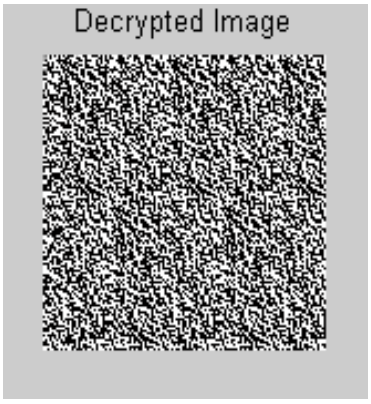
(a)

Figure 5.31a Decomposed components using EEIL Simplet

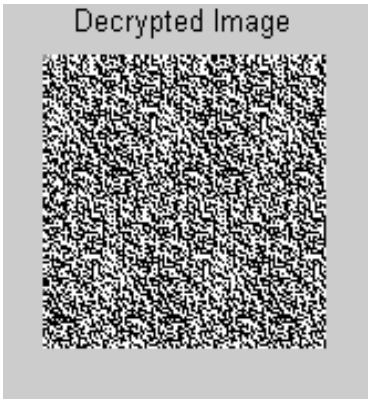


(b)

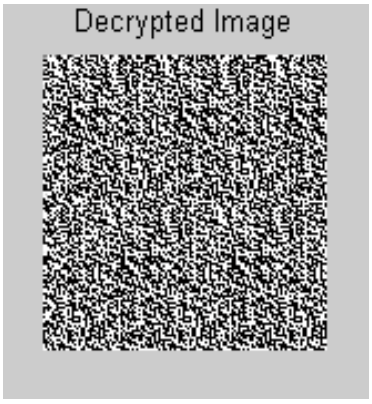
5.31b Encrypted components using key array [7 121 7 3 33]



(a)



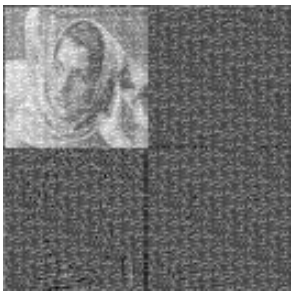
(b)



(c)

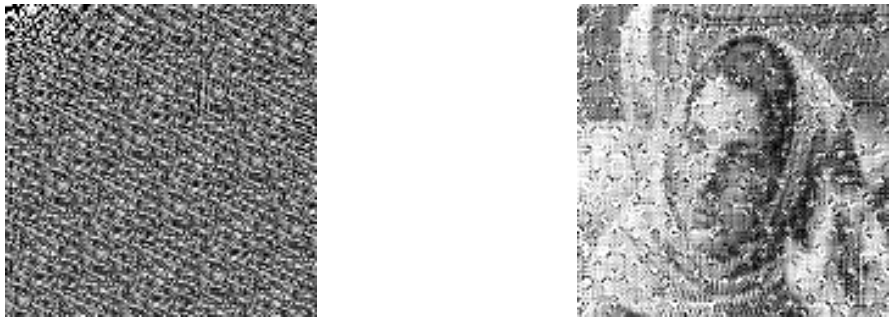
Figure 5.32 Reconstructed images using key arrays (a) [7 121 7 3 32], (b) [7 121 7 2 33] and (c) [7 121 7 3 34].

(a)



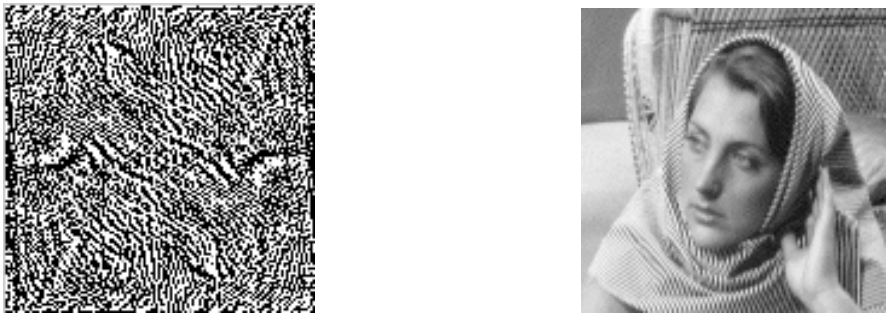
(b)

Figure 5.33 (a) DB2 decomposed components plus MLS [1 23 6 32]
(b) DB2 decomposed components plus MLS [7 121 7 3 33]



(a) (b)

Figure 5.34 (a) Encrypted image of Fig.5.30a using DCT and MLS ([7 121 7 3 33])
(b) Decrypted image from Fig.5.23a using a different MLS ([60,11,8,25,120])



(a) (b)

Figure 5.35 (a) Encrypted image of Fig5.30a using DFT and MLS ([7 121 7 3 33])
(b) Decrypted image from Fig.24a using a different MLS ([60,11,8,25,120])

(a) (b)

Figure 5.36 Decomposed components of the image in Fig.5.30a using EEFL
(a) with $L = 10$ and (b) with $L = 40$.

Key Array	Encrypted Components	
Encryption	Chirp	Image
[23 6 12]	1) –0.008925	1) 0.005090
	2) –0.009509	2) -0.006713
		3) 0.007639
		4) 0.000109
[1 23 6 32]	1) 0.000549	1) -0.006966
	2) –0.000212	2) -0.014682
		3) -0.004247
		4) -0.017333
[7 121 7 3 33]	1) 0.004752	1) 0.004202
	2) 0.003810	2) -0.000454
		3) 0.007425
		4) -0.000454

Table 5.7 Correlation coefficients between the original signals and their encrypted components.

Key Array		Reconstructed Signal	
Encryption	Decryption	Chirp	Image
[23 6 12]	[23 5 12]	-0.000898	-0.009074
	[23 7 12]	0.007193	0.003084
	[22 6 12]	0.006832	-0.020053
	[24 6 12]	0.005429	-0.006271
	[23 6 11]	0.006015	-0.019939
	[23 6 13]	0.009499	-0.010063
[1 23 6 32]	[2 23 6 32]	0.009393	0.008781
	[0 23 6 32]	0.003843	0.002053
	[1 22 6 32]	0.004260	0.013582
	[1 24 6 32]	0.002208	0.011263
	[1 23 7 32]	0.002541	-0.002960
	[1 23 5 32]	-0.001145	0.012148
	[1 23 6 33]	0.008266	0.014854
	[1 23 6 31]	-0.004883	0.016535
[7 121 7 3 33]	[7 121 7 3 32]	-0.007735	-0.009910
	[7 121 7 2 33]	-0.011583	-0.004167
	[7 121 7 3 34]	-0.001081	-0.010558
	[7 121 6 3 33]	-0.007158	-0.000691
	[7 120 7 3 33]	-0.001704	-0.007069
	[8 121 7 3 33]	-0.000492	-0.013704

Table 5.8 Correlation coefficients between the original signals and the reconstructed signals using various decryption key arrays.

Chapter 6

CONCLUSIONS

6.1 Conclusions

We have developed two new subband transforms entitled ISITRA and YKSK transforms based on the model of multiplication called Bino's Model of Multiplication. It is common model of multiplication for numbers, polynomials or discrete convolution. It gives us an insight into the filtering operation, which is nothing but a discrete linear convolution. It can be extended to multilevel multiplication or convolution. This makes us easier to the development of ISITRA and YKSK transforms.

ISITRA is a generalized form of 2-channel filter on which the discrete wavelet transform is based. ISITRA is free from any complex mathematical theories unlike wavelet transform, which are difficult to understand and implement for scientists and engineers. Being based on array mathematics, ISITRA can be easily implemented in computers. We derived various possible ways of obtaining perfect reconstruction through a 2-channel filter bank. Different reconstruction schemes yield different member condition equations from which various decomposition and reconstruction filters can be easily computed. In wavelet theory, finding filters for perfect reconstruction is a difficult task. The member condition equations of ISITRA eliminate this difficulty of finding filters for perfect reconstruction. To decompose a signal in two approximations or into two details or into one approximation and one detail, different types filters can be obtained. That is, perfect reconstruction filters can be chosen without having to worry about the conditions imposed in wavelet or filter bank to get perfect reconstruction. This flexibility in choosing filters will be of helpful in finding optimal filters for various applications. Perfect reconstruction filters can be obtained from the member condition equations either from a single array or from two different arrays and accordingly

ISITRA is classified into two types, namely, ISITRA-1 and ISITRA-2. The orthogonal filters of wavelets and filter bank are found to belong to ISITRA-1 and the bi-orthogonal ones to ISITRA-2. Also, a multi-resolution signal representation is achieved in ISITRA through a multilevel decomposition scheme where a signal is successively decomposed.

From the 2-channel filter bank scheme of ISITRA, it is generalized to 2M-channel filter bank scheme. This 2M-channel filter bank scheme is different from M-channel filter bank scheme of multi-rate signal processing in the sense that the number of channels in the purposed multi-channel case cannot be odd. This simplifies the perfect reconstruction theory in our multi-channel scheme. There is not much difficulty in getting perfect reconstruction filters as compared to the M-channel filter bank based on polyphase representation. We have also proposed the reverse of the 2M-channel filter bank, i.e., 2M-channel transmultiplexer. Our 2M channel transmultiplexer can be made free from all kinds of distortions such as cross talk, amplitude, phase distortions etc. which are the main problems in the M-channel transmultiplexer based on polyphase representation.

The decomposition scheme in ISITRA involves finding filters that satisfy the member condition. The degree of freedom in choosing filter coefficients is quite large as compared to wavelet and filter bank. But we cannot still use any arbitrary array as a decomposition filter. We are interested in finding a transform in which any arbitrary array of any length, even or odd, can be used as a filter for the decomposition of a signal. Consequently, a transform entitled YKSK transform is developed to meet the above requirements. The motivation in developing such a transform is to give a generalized transform that can be used for various signal and image processing applications. YKSK transform has two types, namely, YKSK-1 and YKSK-2 transforms. Lifting scheme of Wim Sweldens and most of the existing integer transforms developed so far are found to be the special cases of YKSK-1 transform. YKSK-2 transform is more generalized than YKSK-1 transform in the sense that the latter can be thought of as a special case of the former and can also be more symmetric than the latter. Both YKSK-1 and YKSK-2 transforms are classified into fixed length, semi-infinite length and infinite length types. Fixed length types are better suited for signal analysis, compression etc, and the infinite length types for signal distortion purpose (useful for encryption). The application of semi-infinite length types is yet to be established. Multi-

stage and multi-level decompositions are possible in YKSK transform. Also, we can easily get integer transforms from YKSK transforms.

To demonstrate the applicability of ISITRA, we use it for lossy image compression. For this purpose, we developed two coding schemes named ISPIHT and DPC. The former can be thought of as the improved or modified form of SPIHT to make it faster and to reduce the memory requirement. ISPIHT gives the same performance in terms of PSNR values at a given compression ratio. The difference between ISPIHT and SPIHT is in the coding part only, while the decoding part remains the same. That is, the same decoding algorithm of SPIHT can also work as the decoding algorithm of ISPIHT. The improvement is made only in the coding part by reducing the number of searching operations and the memory requirement in the significant testing of the descendant trees during the sorting pass. However, the improvement is not very significant at the compression ratio 8:1 or lower. Hence, we develop DPC as a faster and memory efficient subband coder. DPC is much simpler than any existing subband coding schemes. Yet, it is significantly faster than SPIHT and requires less memory. The compression performance is marginally less as compared with the two standard subband coders SPIHT and JPEG2000. But considering other factors, such as low computational complexity and low memory requirement, DPC can be considered significantly better than the existing standards. These two features of DPC,, namely, less computational complexity and low memory requirement, would also be the attractive features from the hardware implementation point of view.

The compression performance of a lossy compression scheme depends not only in the coding scheme used but also in the filters used. A systematic search technique for finding optimal filters based on the minimization of mean square error using genetic algorithm is provided. Some ISITRA-2 filters of length 10 having the same framework as that of 9/7 filters for use in lossy image compression are proposed. It is found that these filters give marginally better performance in terms of PSNR values for some images. This shows that the popular 9/7 bi-orthogonal wavelet filter set used in SPIHT and JPEG200 for lossy image compression is not the best filter set to compress all kinds of images.

To demonstrate the applicability of YKSK transform, we use it for signal encryption. Infinite length type YKSK transform has the property of achieving signal distortion and it is

very difficult to achieve reconstruction if there is any change in any of the decomposed components. We use this property for hiding the information content in the signal for encryption application. To make the encryption functional, we require a locking system. For that propose, a random sequence entitled meitei lock sequence (MLS), which can be generated from an array seed, is developed. The array seed is used as a key in our encryption model. MLS has the property that if there is any slight change in any of the elements of the key elements, there is a large change in the MLS. It is almost impossible to generate the same MLS unless one exactly knows the key elements. As the key array can be chosen arbitrarily without any restriction in its length or elements, the search space for the key is infinitely large. In other words, the uniqueness of MLS and the reconstruction difficulty of infinite length YKSK transform provide tight security in our proposed encryption scheme.

6.2 Future Directions of Research

This thesis opens up some new avenues for further research, which are given below.

1) Representa provides an easier way of manipulating large numbers in a computer. Finding new number theory results based on representa would be useful in solving problems, such as finding easier ways of testing of large prime numbers, large number factorizations etc., which have direct applications in public key cryptography.

2) Bino's model of multiplication provides a common way of number multiplication, polynomial multiplication and discrete convolution. A faster way of polynomial factorization, which will be a common model for large number division or discrete deconvolution may be developed. Moreover, the model of multiplication may be used for the study of combinatorics and may lead to the development of trinomial distribution, quadrinomial distribution etc.

3) ISITRA suggests some new decomposition and reconstruction schemes. Each reconstruction scheme generates different types of PRF frameworks. Each framework is a source of infinitely many actual PRF sets. There is no hard and fast rule on the suitability of any particular PRF set for an application. Several investigations can be carried out for finding optimal PRF sets for different frameworks for different applications. Also, the suitability of complex PRF sets for different applications can be investigated.

4) In some papers, it is reported that m-band wavelets perform better than normal wavelets. Investigation can also be carried out whether 2M-channel ISITRA can give better performance than m-band wavelets. Similar investigations can also be performed on different possible forms of multi-channel transmultiplexer schemes using ISITRA model.

5) YKSK transform is a reversible transform where the filters can be chosen arbitrarily. Investigation can be done on the methods for choosing appropriate filters for certain applications. Suitability and performance of integer version of YKSK transforms for lossless image compression can also be investigated.

6) Further work can be done on the improvement of performance of DPC and so that it can outperforms the existing standards SPIHT and JPEG2000.

7) Investigations on the properties of MLS and possible cryptanalytic attacks may lead to interesting results for symmetric key cryptography.

APPENDIX A

More Reconstruction Schemes of ISITRA

We will provide here various reconstruction schemes, which are not similar to those of wavelet and filter bank community. The PRF sets derived from these reconstruction schemes will be different from those discussed in Chapter 3. Only the member conditions and the reconstruction equations are given. One may obtain various frameworks and the associated PRF sets corresponding to different member conditions of different reconstruction schemes. In some cases, these reconstruction schemes may be easier and straightforward in computation.

A.1 First Reconstruction Scheme

Member condition I:

$$\left.\begin{aligned}\sum_{i=1}^{r}\{\mathbf{f}_a(2i)\mathbf{f}_{ar}(2r+1-2i)+\mathbf{f}_b(2i)\mathbf{f}_{br}(2r+1-2i)\}=0\\ \sum_{i=1}^{r}\{\mathbf{f}_a(2i-1)\mathbf{f}_{ar}(2r+2-2i)+\mathbf{f}_b(2i-1)\mathbf{f}_{br}(2r+2-2i)\}=0\end{aligned}\right\}\ \forall\ r \quad \cdots\cdots(a.1)$$

Member condition II:

$$\left.\begin{aligned}\sum_{i=1}^{r}\{\mathbf{f}_a(2i-1)\mathbf{f}_{ar}(2r+1-2i)+\mathbf{f}_b(2i-1)\mathbf{f}_{br}(2r+1-2i)\}=0\\ \sum_{i=1}^{r}\{\mathbf{f}_a(2i)\mathbf{f}_{ar}(2r+2-2i)+\mathbf{f}_b(2i)\mathbf{f}_{br}(2r+2-2i)\}=0\end{aligned}\right\}\ \forall\ r\neq L/2 \qquad \cdots\cdots(a.2)$$

Reconstruction equations :

$$\mathbf{b}(2n-1) = \frac{\sum_{i=1}^{L/2} \{\mathbf{f}_{ar}(2i)\mathbf{A}(n+1-i) + \mathbf{f}_{br}(2i)\mathbf{B}(n+1-i)\}}{\sum_{i=1}^{L/2} \{\mathbf{f}_{a}(2i)\mathbf{f}_{ar}(L+2-2i) + \mathbf{f}_{b}(2i)\mathbf{f}_{br}(L+2-2i)\}} \quad \cdots\cdots(a.3)$$

$$\mathbf{b}(2n) = \frac{\sum_{i=1}^{L/2} \{\mathbf{f}_{ar}(2i-1)\mathbf{A}(n+1-i) + \mathbf{f}_{br}(2i-1)\mathbf{B}(n+1-i)\}}{\sum_{i=1}^{L/2} \{\mathbf{f}_{a}(2i-1)\mathbf{f}_{ar}(L+1-2i) + \mathbf{f}_{b}(2i-1)\mathbf{f}_{br}(L+1-2i)\}} \quad \cdots\cdots(a.4)$$

Alternatively, we can also write

Member condition I:

$$\left.\begin{aligned} &\sum_{i=1}^{r} \{\mathbf{f}_{a}(2i-1)\mathbf{f}_{ar}(2r+1-2i) + \mathbf{f}_{b}(2i-1)\mathbf{f}_{br}(2r+1-2i)\} = 0 \\ &\sum_{i=1}^{r} \{\mathbf{f}_{a}(2i)\mathbf{f}_{ar}(2r+2-2i) + \mathbf{f}_{b}(2i)\mathbf{f}_{br}(2r+2-2i)\} = 0 \end{aligned}\right\} \quad \cdots\cdots(a.5)$$

Member condition II:

$$\left.\begin{aligned} &\sum_{i=1}^{r} \{\mathbf{f}_{a}(2i)\mathbf{f}_{ar}(2r+1-2i) + \mathbf{f}_{b}(2i)\mathbf{f}_{br}(2r+1-2i)\} = 0 \\ &\sum_{i=1}^{r} \{\mathbf{f}_{a}(2i-1)\mathbf{f}_{ar}(2r+2-2i) + \mathbf{f}_{b}(2i-1)\mathbf{f}_{br}(2r+2-2i)\} = 0 \end{aligned}\right\} \quad \forall\ r \neq L/2 \quad \cdots\cdots\cdots(a.6)$$

Reconstruction equations :

$$\mathbf{b}(2n-1) = \frac{\sum_{i=1}^{L/2} \{\mathbf{f}_{ar}(2i-1)\mathbf{A}(n+1-i) + \mathbf{f}_{br}(2i-1)\mathbf{B}(n+1-i)\}}{\sum_{i=1}^{L/2} \{\mathbf{f}_{a}(2i)\mathbf{f}_{ar}(L+1-2i) + \mathbf{f}_{b}(2i)\mathbf{f}_{br}(L+1-2i)\}} \quad \cdots\cdots(a.7)$$

$$\mathbf{b}(2n) = \frac{\sum_{i=1}^{L/2} \{\mathbf{f}_{ar}(2i)\mathbf{A}(n+1-i) + \mathbf{f}_{br}(2i)\mathbf{B}(n+1-i)\}}{\sum_{i=1}^{L/2} \{\mathbf{f}_{a}(2i-1)\mathbf{f}_{ar}(L+2-2i) + \mathbf{f}_{b}(2i-1)\mathbf{f}_{br}(L+2-2i)\}} \quad \cdots\cdots(a.8)$$

A.2 Second Reconstruction Scheme

Member condition I:

$$\left.\begin{aligned} &\sum_{i=1}^{r} \{\mathbf{f}_{a}(2i)\mathbf{f}_{ar}(L-1+2i-2r) + \mathbf{f}_{b}(2i)\mathbf{f}_{br}(L-1+2i-2r)\} = 0 \\ &\sum_{i=1}^{r} \{\mathbf{f}_{a}(2i-1)\mathbf{f}_{ar}(L-2r+2i) + \mathbf{f}_{b}(2i-1)\mathbf{f}_{br}(L-2r+2i)\} = 0 \end{aligned}\right\} \quad \cdots\cdots(a.9)$$

Member condition II:

$$\left.\begin{aligned} &\sum_{i=1}^{r} \{\mathbf{f}_{a}(2i-1)\mathbf{f}_{ar}(L-1+2i-2r) + \mathbf{f}_{b}(2i-1)\mathbf{f}_{br}(L-1+2i-2r)\} = 0 \\ &\sum_{i=1}^{r} \{\mathbf{f}_{a}(2i)\mathbf{f}_{ar}(L-2r+2i) + \mathbf{f}_{b}(2i)\mathbf{f}_{br}(L-2r+2i)\} = 0 \end{aligned}\right\} \quad \forall\ r \neq L/2 \quad \cdots\cdots\cdots(a.10)$$

Reconstruction Equations:

$$\mathbf{b}(2n-1) = \frac{\sum_{i=1}^{L/2} \{\mathbf{f}_{ar}(2i)\mathbf{A}(n-1+i) + \mathbf{f}_{br}(2i)\mathbf{B}(n-1+i)\}}{\sum_{i=1}^{L/2} \{\mathbf{f}_{a}(2i)\mathbf{f}_{ar}(2i) + \mathbf{f}_{b}(2i)\mathbf{f}_{br}(2i)\}} \quad \cdots\cdots(a.11)$$

$$\mathbf{b}(2n) = \frac{\sum_{i=1}^{L/2} \{\mathbf{f}_{ar}(2i-1)\mathbf{A}(n-1+i) + \mathbf{f}_{br}(2i-1)\mathbf{B}(n-1+i)\}}{\sum_{i=1}^{L/2} \{\mathbf{f}_{a}(2i-1)\mathbf{f}_{ar}(2i-1) + \mathbf{f}_{b}(2i-1)\mathbf{f}_{br}(2i-1)\}} \quad \cdots\cdots(a.12)$$

Alternatively, we can write

Member condition I:

$$\left.\begin{aligned}\sum_{i=1}^{r}\{\mathbf{f}_a(2i)\mathbf{f}_{ar}(L-2r+2i)+\mathbf{f}_b(2i)\mathbf{f}_{br}(L-2r+2i)\}=0\\ \sum_{i=1}^{r}\{\mathbf{f}_a(2i-1)\mathbf{f}_{ar}(L-2r+2i-1)+\mathbf{f}_b(2i-1)\mathbf{f}_{br}(L-2r+2i-1)\}=0\end{aligned}\right\}\quad\cdots\cdots(a.13)$$

Member condition II:

$$\left.\begin{aligned}\sum_{i=1}^{r}\{\mathbf{f}_a(2i-1)\mathbf{f}_{ar}(L-2r+2i)+\mathbf{f}_b(2i-1)\mathbf{f}_{br}(L-2r+2i)\}=0\\ \sum_{i=1}^{r}\{\mathbf{f}_a(2i)\mathbf{f}_{ar}(L-2r+2i-1)+\mathbf{f}_b(2i)\mathbf{f}_{br}(L-2r+2i-1)\}=0\end{aligned}\right\}\quad\forall\ r\neq L/2\cdots\cdots\cdots(a.14)$$

Reconstruction equations:

$$\mathbf{b}(2n-1)=\frac{\sum_{i=1}^{L/2}\{\mathbf{f}_{ar}(2i-1)\mathbf{A}(n-1+i)+\mathbf{f}_{br}(2i-1)\mathbf{B}(n-1+i)\}}{\sum_{i=1}^{L/2}\{\mathbf{f}_a(2i)\mathbf{f}_{ar}(2i-1)+\mathbf{f}_b(2i)\mathbf{f}_{br}(2i-1)\}}\quad\cdots\cdots(a.15)$$

$$\mathbf{b}(2n)=\frac{\sum_{i=1}^{L/2}\{\mathbf{f}_{ar}(2i)\mathbf{A}(n-1+i)+\mathbf{f}_{br}(2i)\mathbf{B}(n-1+i)\}}{\sum_{i=1}^{L/2}\{\mathbf{f}_a(2i-1)\mathbf{f}_{ar}(2i)+\mathbf{f}_b(2i-1)\mathbf{f}_{br}(2i)\}}\quad\cdots\cdots(a.16)$$

A.3 Third Reconstruction Scheme

Member condition I:

$$\left.\begin{aligned}\sum_{i=1}^{r}\{\mathbf{f}_a(2i)\mathbf{f}_{ar}(2r+1-2i)+\mathbf{f}_b(2i)\mathbf{f}_{br}(2r+1-2i)\}=0\\ \sum_{i=1}^{r}\{\mathbf{f}_a(2i-1)\mathbf{f}_{ar}(2r+2-2i)+\mathbf{f}_b(2i-1)\mathbf{f}_{br}(2r+2-2i)\}=0\end{aligned}\right\} \quad \cdots\cdots(a.17)$$

Member condition II:

$$\left.\begin{aligned}\sum_{i=1}^{r}\{\mathbf{f}_a(2i-1)\mathbf{f}_{ar}(2r+1-2i)+\mathbf{f}_b(2i-1)\mathbf{f}_{br}(2r+1-2i)\}=0\\ \sum_{i=1}^{r}\{\mathbf{f}_a(2i)\mathbf{f}_{ar}(2r+2-2i)+\mathbf{f}_b(2i)\mathbf{f}_{br}(2r+2-2i)\}=0\end{aligned}\right\} \quad \forall\ r\neq L/2 \quad \cdots\cdots\cdots(a.18)$$

Reconstruction equations:

$$\mathbf{b}(2n-1)=\frac{\sum_{i=1}^{L/2}\{\mathbf{f}_{ar}(2i)\mathbf{A}(n+1-i)+\mathbf{f}_{br}(2i)\mathbf{B}(n+1-i)\}}{\sum_{i=1}^{L/2}\{\mathbf{f}_a(2i)\mathbf{f}_{ar}(L+2-2i)+\mathbf{f}_b(2i)\mathbf{f}_{br}(L+2-2i)\}} \quad \cdots\cdots(a.19)$$

$$\mathbf{b}(2n)=\frac{\sum_{i=1}^{L/2}\{\mathbf{f}_{ar}(2i-1)\mathbf{A}(n+1-i)+\mathbf{f}_{br}(2i-1)\mathbf{B}(n+1-i)\}}{\sum_{i=1}^{L/2}\{\mathbf{f}_a(2i-1)\mathbf{f}_{ar}(L+1-2i)+\mathbf{f}_b(2i-1)\mathbf{f}_{br}(L+1-2i)\}} \quad \cdots\cdots(a.20)$$

Alternatively, we can also write

Member condition I:

$$\left.\begin{aligned}\sum_{i=1}^{r}\{\mathbf{f}_a(2i-1)\mathbf{f}_{ar}(2r+1-2i)+\mathbf{f}_b(2i-1)\mathbf{f}_{br}(2r+1-2i)\}=0\\ \sum_{i=1}^{r}\{\mathbf{f}_a(2i)\mathbf{f}_{ar}(2r+2-2i)+\mathbf{f}_b(2i)\mathbf{f}_{br}(2r+2-2i)\}=0\end{aligned}\right\} \cdots\cdots\cdots(a.21)$$

Member condition II:

$$\left.\begin{aligned}\sum_{i=1}^{r}\{\mathbf{f}_a(2i)\mathbf{f}_{ar}(2r+1-2i)+\mathbf{f}_b(2i)\mathbf{f}_{br}(2r+1-2i)\}=0\\ \sum_{i=1}^{r}\{\mathbf{f}_a(2i-1)\mathbf{f}_{ar}(2r+2-2i)+\mathbf{f}_b(2i-1)\mathbf{f}_{br}(2r+2-2i)\}=0\end{aligned}\right\}\quad \forall\ r\neq L/2 \quad \cdots\cdots\cdots(a.22)$$

Reconstruction equations:

$$\mathbf{b}(2n-1)=\frac{\sum_{i=1}^{L/2}\{\mathbf{f}_{ar}(2i)\mathbf{A}(n+1-i)+\mathbf{f}_{br}(2i)\mathbf{B}(n+1-i)\}}{\sum_{i=1}^{L/2}\{\mathbf{f}_a(2i)\mathbf{f}_{ar}(L+2-2i)+\mathbf{f}_b(2i)\mathbf{f}_{br}(L+2-2i)\}} \quad \cdots\cdots(a.23)$$

$$\mathbf{b}(2n)=\frac{\sum_{i=1}^{L/2}\{\mathbf{f}_{ar}(2i-1)\mathbf{A}(n+1-i)+\mathbf{f}_{br}(2i-1)\mathbf{B}(n+1-i)\}}{\sum_{i=1}^{L/2}\{\mathbf{f}_a(2i-1)\mathbf{f}_{ar}(L+2-2i)+\mathbf{f}_b(2i-1)\mathbf{f}_{br}(L+2-2i)\}} \quad \cdots\cdots(a.24)$$

APPENDIX B

More Frameworks Of PRF Sets

B.1 Frameworks of PRF-6 and PRF-8 Sets of ISITRA-1

B.1.1 PRF-6 sets

When $L = 6$, we get the following two sets of equations from equations (8) and (11) (of Chapter 3) of member condition-I.

$$\left.\begin{aligned}
&\mathbf{f}_a(1)\mathbf{f}_{ar}(1) + \mathbf{f}_b(1)\mathbf{f}_{br}(1) = 0\\
&\mathbf{f}_a(1)\mathbf{f}_{ar}(3) + \mathbf{f}_b(1)\mathbf{f}_{br}(1) + \mathbf{f}_a(3)\mathbf{f}_{ar}(1) + \mathbf{f}_b(3)\mathbf{f}_{br}(1) = 0\\
&\mathbf{f}_a(1)\mathbf{f}_{ar}(5) + \mathbf{f}_b(1)\mathbf{f}_{br}(5) + \mathbf{f}_a(3)\mathbf{f}_{ar}(3) + \mathbf{f}_b(3)\mathbf{f}_{br}(3) + \mathbf{f}_a(5)\mathbf{f}_{ar}(1) + \mathbf{f}_b(5)\mathbf{f}_{br}(1) = 0\\
&\mathbf{f}_a(3)\mathbf{f}_{ar}(5) + \mathbf{f}_b(3)\mathbf{f}_{br}(5) + \mathbf{f}_a(5)\mathbf{f}_{ar}(3) + \mathbf{f}_{\mathbf{b}}(5)\mathbf{f}_{br}(3) = 0\\
&\mathbf{f}_a(5)\mathbf{f}_{ar}(5) + \mathbf{f}_b(5)\mathbf{f}_{br}(5) = 0
\end{aligned}\right\}$$

and

$$\left.\begin{aligned}
&\mathbf{f}_a(2)\mathbf{f}_{ar}(2) + \mathbf{f}_b(2)\mathbf{f}_{br}(2) = 0\\
&\mathbf{f}_a(2)\mathbf{f}_{ar}(4) + \mathbf{f}_b(2)\mathbf{f}_{br}(4) + \mathbf{f}_a(4)\mathbf{f}_{\mathbf{ar}}(2) + \mathbf{f}_b(4)\mathbf{f}_{br}(2) = 0\\
&\mathbf{f}_a(2)\mathbf{f}_{ar}(6) + \mathbf{f}_b(2)\mathbf{f}_{\mathbf{br}}(6) + \mathbf{f}_a(4)\mathbf{f}_{ar}(4) + \mathbf{f}_b(4)\mathbf{f}_{br}(4) + \mathbf{f}_a(6)\mathbf{f}_{ar}(2) + \mathbf{f}_b(6)\mathbf{f}_{br}(2) = 0\\
&\mathbf{f}_a(4)\mathbf{f}_{ar}(6) + \mathbf{f}_b(4)\mathbf{f}_{br}(6) + \mathbf{f}_a(6)\mathbf{f}_{ar}(4) + \mathbf{f}_b(6)\mathbf{f}_{br}(4) = 0\\
&\mathbf{f}_a(6)\mathbf{f}_{ar}(6) + \mathbf{f}_b(6)\mathbf{f}_{br}(6) = 0
\end{aligned}\right\}$$

We have to choose $\mathbf{f}_a, \mathbf{f}_b, \mathbf{f}_{ar}$ and $\mathbf{f}_{br}$ which satisfy the above two sets of equations. Let $\mathbf{l} = \{l_1, l_2, l_3, l_4, l_5, l_6\}$ be a vector from which $\mathbf{f}_a, \mathbf{f}_b, \mathbf{f}_{ar}$ and $\mathbf{f}_{br}$ satisfying member condition-I are obtained. Let the following filters be a trial framework of a PRF-6 set.

$$\mathbf{f}_a = [l_1, -l_2, -l_3, l_4, l_5, l_6], \quad \mathbf{f}_{ar} = [l_6, l_5, l_4, -l_3, -l_2, l_1]$$
$$\mathbf{f}_b = [-l_6, l_5, -l_4, -l_3, -l_2, l_1], \ \mathbf{f}_{br} = [l_1, -l_2, -l_3, -l_4, l_5, -l_6]$$

We call them a framework of a PRF-6 sets if they satisfy member condition-II as well. The sets of equations for member condition-II obtained from equations (10) and (13) (of Chapter 3) are

$$\left.\begin{aligned}
&\mathbf{f}_a(2)\mathbf{f}_{ar}(1)+\mathbf{f}_b(2)\mathbf{f}_{br}(1)=0\\
&\mathbf{f}_a(2)\mathbf{f}_{ar}(3)+\mathbf{f}_b(2)\mathbf{f}_{br}(1)+\mathbf{f_a}(4)\mathbf{f}_{ar}(1)+\mathbf{f}_b(4)\mathbf{f}_{br}(1)=0\\
&\mathbf{f_a}(4)\mathbf{f}_{ar}(5)+\mathbf{f}_b(4)\mathbf{f_{br}}(5)+\mathbf{f}_a(6)\mathbf{f}_{ar}(3)+\mathbf{f}_b(6)\mathbf{f_{br}}(3)=0\\
&\mathbf{f}_a(6)\mathbf{f}_{ar}(5)+\mathbf{f}_b(6)\mathbf{f}_{br}(5)=0
\end{aligned}\right\}$$

and

$$\left.\begin{aligned}
&\mathbf{f}_a(1)\mathbf{f}_{ar}(2)+\mathbf{f}_b(1)\mathbf{f}_{br}(2)=0\\
&\mathbf{f}_a(1)\mathbf{f}_{ar}(4)+\mathbf{f}_b(1)\mathbf{f_{br}}(4)+\mathbf{f}_a(1)\mathbf{f}_{ar}(2)+\mathbf{f}_b(1)\mathbf{f}_{br}(2)=0\\
&\mathbf{f}_a(3)\mathbf{f}_{ar}(6)+\mathbf{f}_b(3)\mathbf{f_{br}}(6)+\mathbf{f}_a(5)\mathbf{f}_{ar}(4)+\mathbf{f}_b(5)\mathbf{f_{br}}(4)=0\\
&\mathbf{f}_a(5)\mathbf{f}_{ar}(6)+\mathbf{f}_b(5)\mathbf{f}_{br}(6)=0
\end{aligned}\right\}$$

The above sets of equations yield equation (b.1) as member condition-II of ISITRA-1.

$$\left.\begin{aligned}
&l_1l_5-l_2l_6=0\\
&-l_1l_3-l_2l_4-l_3l_5+l_4l_6=0
\end{aligned}\right\}\cdots\cdots\cdots(b.1)$$

Since all the terms in the two equations in (*b*.1) are not of the same sign, the above trial framework forms a framework of PRF6 set. The actual PRF-6 sets can be found by finding the values of l_1, l_2, l_3, l_4, l_5 and l_6 from equation (*b*.1). This can be done in infinite number of ways. Hence, we can find infinitely many actual PRF-6 sets of the above framework of ISITRA-1.

B.1.2 PRF-8 sets

Similarly, we can find $\mathbf{f}_a, \mathbf{f}_b, \mathbf{f}_{ar}$ and $\mathbf{f}_{br}$ of a framework of a PRF-8 set which satisfy the member condition equations (14) and (15) (of Chapter 4). For example, the following is a framework of a PRF8 set of ISITRA-1.

$$\mathbf{f}_a=[-l_1,l_2,l_3,-l_4,-l_5,l_6,l_{7,}l_8],\quad \mathbf{f}_{ar}=[l_8,l_7,l_6,-l_5,-l_4,l_3,l_{2,}l_1]$$

$$\mathbf{f}_b=[-l_8,l_7,-l_6,-l_5,l_4,l_3,-l_{2,}-l_1],\ \mathbf{f}_{br}=[-l_1,-l_2,l_3,l_4,-l_5,-l_6,l_{7,}-l_8]$$

And the corresponding member condition-II restrictions reduce to

$$\left.\begin{aligned}
&-l_1l_7+l_2l_8=0\\
&l_1l_5+l_2l_8+l_3l_7-l_4l_8=0\\
&-l_1l_3-l_2l_4-l_3l_5-l_4l_6+l_5l_7+l_6l_8=0
\end{aligned}\right\}\cdots\cdots\cdots(b.2)$$

We can find infinitely many values of $l_1, l_2, l_3, l_4, l_5, l_6, l_7$ and l_8 from equation (*b*.2). Thus, we have infinitely many actual PRF-8 sets of the above PRF-8 set framework. Also, we can have various other frameworks satisfying the member condition.

B.2 Frameworks of PRF Sets of Daubechies' Wavelets, Symlets and Coiflets

B.2.1 Framework of PRF sets of filter coefficients of DB3

The filters of DB3 are of length 6. The framework of PRF-6 set of DB3 wavelet filter coefficients is given below.

$\mathbf{f}_a = [l_1, -l_2, -l_3, l_4, l_5, l_6]$ $\mathbf{f}_{ar} = [l_6, l_5, l_4, -l_3, -l_2, l_1]$

$\mathbf{f}_b = [-l_6, l_5, -l_4, -l_3, l_2, l_1]$ $\mathbf{f}_{br} = [l_1, l_2, -l_3, -l_4, l_5, -l_6]$

Member condition-II yields

$$\left.\begin{aligned} l_1 l_5 - l_2 l_6 &= 0 \\ -l_1 l_3 - l_2 l_4 - l_3 l_5 + l_4 l_6 &= 0 \end{aligned}\right\} \cdots\cdots\cdots (b.3)$$

We have to choose any six real numbers that will satisfy equation (*b*.3). In the case of DB3, one of its decomposition filters has the following coefficients.

$\mathbf{f}_a$ =[0.03522629188210 -0.08544127388224 -0.13501102001039 0.45987750211933 0.80689150931334 0.33267055295096]

Here also the coefficients of $\mathbf{f}_a$ of DB3 do no exactly satisfy member condition II. i.e., equation (b.3), and hence it would not be possible in general to obtain perfect reconstruction using these filter coefficients in the framework. Theoretically, for perfect reconstruction using the above framework of DB3, we have to find PRF-6 sets, which satisfy equation (*b*.3). It is not difficult to find filter coefficients that satisfy equation (*b*.3). For example, the following filter coefficients satisfy it.

$l_1 = 0.125$, $l_2 = 0.225$, $l_3 = 0.516$, $l_4 = 1.9888$, $l_5 = 0.8694$ and $l_6 = 0.483$.

Similarly, we can find some other PRF-6 sets corresponding to the above framework from equation (*b*.3). The filter coefficients of Sym3 have the same framework. Thus, we can find various PRF-6 sets other than that of the PRF-6 set of DB3. Also, one can find various other frameworks from DB3 framework using Property-1 or Property-2 (Chapter 3).

B.2.2 Framework of PRF sets of filter coefficients of DB4

The length of the filters here is 8. They have the following framework.

$\mathbf{f}_a = [-l_1, l_2, l_3, -l_4, -l_5, l_6, l_7, l_8]$ $\mathbf{f}_{ar} = [l_8, l_7, l_6, -l_5, -l_4, l_3, l_2, -l_1]$

$\mathbf{f}_b = [-l_8, l_7 - l_6, -l_5, l_4, l_3, -l_2, -l_1]$ $\mathbf{f}_{br} = [-l_1, -l_2, l_3, l_4, -l_5, -l_6, l_7, -l_8]$

And member condition-II yields

$$\left.\begin{aligned} &-l_1l_7 + l_2l_8 = 0 \\ &l_1l_5 + l_2l_6 + l_3l_7 - l_4l_8 = 0 \\ &-l_1l_3 - l_2l_4 - l_3l_5 - l_4l_6 - l_5l_7 + l_6l_8 = 0 \end{aligned}\right\} \cdots\cdots\cdots (b.4)$$

The DB4 filter coefficients do not exactly satisfy the equation (*b*.4). For economy of space, we have not shown the wavelet filter coefficients of DB4. The floating point values of the wavelet filter coefficients of DB4 cannot be used for perfect reconstruction from its decomposed components as they do not exactly satisfy equation (*b*.4). One can find filter coefficients, which will give similar results as those of DB4 but better reconstruction as discussed in the case of framework of DB2 or DB3. Also, we can find infinitely many PRF-8 sets of the above framework from equation (*b*.4).

B.2.3 Framework of PRF sets of filter coefficients of DB5

The framework of PRF-10 set of DB5 is shown below.

$$\mathbf{f}_a = [l_1, -l_2, -l_3, l_4, -l_5, -l_6, l_7, l_8, l_9, l_{10}]$$
$$\mathbf{f}_{ar} = [l_{10}, l_9, l_8, l_7, -l_6, -l_5, l_4, -l_3, -l_2, l_1]$$
$$\mathbf{f}_b = [-l_{10}, l_9 - l_8, l_7, l_6, -l_5, -l_4, -l_3, l_2, l_1]$$
$$\mathbf{f}_{br} = [l_1, l_2, -l_3, -l_4, -l_5, l_6, l_7, -l_8, l_9, -l_{10}]$$

And member condition-II yields

$$\left.\begin{aligned} &l_1l_9 - l_2l_{10} = 0 \\ &l_1l_7 - l_2l_8 - l_3l_9 + l_4l_{10} = 0 \\ &-l_1l_5 + l_2l_6 - l_3l_7 + l_4l_8 - l_5l_9 - l_6l_{10} = 0 \\ &-l_1l_3 - l_2l_4 + l_3l_5 - l_4l_6 - l_5l_7 - l_6l_8 + l_7l_9 + l_8l_{10} = 0 \end{aligned}\right\} \cdots\cdots\cdots (b.5)$$

The filter coefficients of DB5 do not exactly satisfy equation (b.5). Hence there will be reconstruction error. One can find better filter coefficients, which will exactly or much more closely satisfy equation (*b*.5) as discussed in the case of framework of DB2 or DB3. The more closely a PRF10 set satisfies equation (*b*.5), the less reconstruction error it will give. One can find different PRF10 sets of the same framework by finding any ten real numbers that satisfy equation (*b*.5).

B.2.4 Framework of PRF sets of filter coefficients of Sym2

The framework of Symlets of length 4 (in short Sym2) is the same as that of DB2.

B.2.5 Framework of PRF sets of filter coefficients of Sym3

Same as that of DB3.

B.2.6 Framework of PRF sets of filter coefficients of Sym4

The filters of sym4 are of length 8 but it has different framework from that of DB4. The framework of PRF set of filter coefficients of sym4 is given below.

$\mathbf{f}_a = [-l_1, -l_2, l_3, l_4, l_5, -l_6, -l_7, l_8]$ $\quad \mathbf{f}_{ar} = [l_8, -l_7, -l_6, l_5, l_4, l_3, -l_2, -l_1]$

$\mathbf{f}_b = [-l_8, -l_7, l_6, l_5, -l_4, l_3, l_2, -l_1]$ $\quad \mathbf{f}_{br} = [-l_1, l_2, l_3, -l_4, l_5, l_6, -l_7, -l_8]$

And the corresponding member condition-II restriction yields

$$\left.\begin{aligned} & l_1 l_7 - l_2 l_8 = 0 \\ & -l_1 l_5 + l_2 l_6 - l_3 l_7 + l_4 l_8 = 0 \\ & -l_1 l_3 - l_2 l_4 + l_3 l_5 - l_4 l_6 - l_5 l_7 - l_6 l_8 = 0 \end{aligned}\right\} \cdots\cdots\cdots (b.6)$$

The filter coefficients of the Sym4 also do not exactly satisfy equation (*b*.6), and hence here also reconstruction error exists. As this framework is different from that of DB4, the equation of member condition-II is also different. For economy of space, we do not show the filter coefficients. From equation (*b*.6), one can find infinitely many PRF-8 sets of the framework of Sym4. Also, one can find different frameworks from this framework using Property-1 or Property-2 (Chapter 3).

B.2.7 Framework of PRF sets of filter coefficients of Sym5

The filters of Sym5 are of length 10. The framework and their member condition-II restrictions are given below.

$\mathbf{f}_a = [l_1, l_2, -l_3, l_4, l_5, l_6, l_7, -l_8, -l_9, l_{10}]$

$\mathbf{f}_{ar} = [l_{10}, -l_9, -l_8, l_7, l_6, l_5, l_4, -l_3, l_2, l_1]$

$\mathbf{f}_b = [-l_{10}, -l_9, l_8, l_7, -l_6, l_5, -l_4, -l_3, -l_2, l_1]$

$\mathbf{f}_{br} = [l_1, -l_2, -l_3, -l_4, l_5, -l_6, l_7, l_8, -l_9, -l_{10}]$

And the corresponding member condition-II restriction yields

$$\left.\begin{aligned} & -l_1 l_9 + l_2 l_{10} = 0 \\ & l_1 l_7 - l_2 l_8 + l_3 l_9 + l_4 l_{10} = 0 \\ & l_1 l_5 + l_2 l_6 - l_3 l_7 - l_4 l_8 - l_5 l_9 + l_6 l_{10} = 0 \\ & -l_1 l_3 + l_2 l_4 - l_3 l_5 + l_4 l_6 + l_5 l_7 - l_6 l_8 - l_7 l_9 - l_8 l_{10} = 0 \end{aligned}\right\} \cdots\cdots\cdots (b.7)$$

As in the earlier cases, one can find here infinitely many PRF-10 sets other than that of DB5 by finding any 10 real numbers that satisfy equation (*b*.7).

B.2.8 Framework of PRF sets of Coif1

The framework of PRF set Coif1 (first Coiflets) is different from those of DB3 and Sym3.

$\mathbf{f}_a = [-l_1, -l_2, l_3, l_4, l_5, -l_6]$ $\mathbf{f}_{ar} = [-l_6, l_5, l_4, l_3, -l_2, -l_1]$

$\mathbf{f}_b = [l_6, l_5, -l_4, l_3, l_2, -l_1]$ $\mathbf{f}_{br} = [-l_1, l_2, l_3, -l_4, l_5, l_6]$

Member condition-II yields

$$\left.\begin{aligned} -l_1 l_5 + l_2 l_6 &= 0 \\ -l_1 l_3 - l_2 l_4 + l_3 l_5 - l_4 l_6 &= 0 \end{aligned}\right\} \quad \cdots\cdots\cdots (b.8)$$

Since the lengths of the Coiflets filters are multiples of 6, we discuss here only the framework of Coif1. Others can be found in the same fashion.

B.3 Frameworks of PRF Sets of Daubechies' Biorthogonal Wavelets

B.3.1 Framework of PRF sets of Bior2.4

The length of the filters of Bior2.4 is 10. The framework of PRF set of Bior2.4 is shown below.

$\mathbf{f}_a = [0, l_1, -l_2, -l_3, l_4, l_5, l_4, -l_3, -l_2, l_1]$ $\mathbf{f}_{ar} = [0, 0, 0, k_1, k_2, k_1, 0, 0, 0, 0]$

$\mathbf{f}_b = [0, 0, 0, k_1, -k_2, k_1, 0, 0, 0, 0]$ $\mathbf{f}_{br} = [0, -l_1, -l_2, l_3, l_4, -l_5, l_4, l_3, -l_2, -l_1]$

Member condition-II yields

$$\left.\begin{aligned} l_1 k_2 - l_2 k_1 &= 0 \\ -l_2 k_1 - l_3 k_2 + l_4 k_1 &= 0 \end{aligned}\right\} \cdots\cdots\cdots (b.9)$$

As equation (*b*.9) is free of l_5, l_5 has no role in perfect reconstruction. Thus, we can choose any real number as its value. The values of the other variable coefficients of the members of the framework are evaluated from equation (*b*.9). As this can be easily done in infinite number of ways, we can find infinitely many PRF-10 sets of the above framework.

B.3.2 Framework of PRF sets of Bior2.6

The filters of Bior2.6 are of length 14. Their framework is given below.

$\mathbf{f}_a = [0, -l_1, l_2, l_3, -l_4, -l_5, l_6, l_7, l_6, -l_5, -l_4, l_3, l_2, -l_1]$

$\mathbf{f}_{ar} = [0, 0, 0, 0, 0, k_1, k_2, k_1, 0, 0, 0, 0, 0, 0]$

$\mathbf{f}_b = [0, 0, 0, 0, 0, k_1, -k_2, k_1, 0, 0, 0, 0, 0, 0]$

$\mathbf{f}_{br} = [0, l_1, l_2, -l_3, -l_4, l_5, l_6, l-_7, l_6, l_5, -l_4, -l_3, l_2, l_1]$

And the corresponding member condition-II yields

$$\left.\begin{aligned} &l_1k_2 + l_2k_1 = 0 \\ &l_2k_1 + l_3k_2 - l_4k_1 = 0 \\ &-l_4k_1 - l_5k_2 + l_6k_1 = 0 \end{aligned}\right\} \cdots\cdots\cdots(b.10)$$

As equation (*b*.10) is free of l_7, it has no role in perfect reconstruction. Thus, we can choose any real number as its value. The values of the other variable coefficients of the members of the framework are evaluated from equation (*b*.10). As in the earlier cases, here also we can find infinitely many PRF sets of the above framework.

B.3.3 Framework of PRF sets of Bior2.8

The filters of Bior2.8 are of length 18. Their framework is shown below.

$\mathbf{f}_a = [0, l_1, -l_2, -l_3, l_4, l_5, -l_6, -l_7, l_8, l_9, l_8, -l_7, -l_6, l_5, l_4, -l_3, -l_2, l_1]$

$\mathbf{f}_{ar} = [0, 0, 0, 0, 0, 0, 0, k_1, k_2, k_1, 0, 0, 0, 0, 0, 0, 0, 0]$

$\mathbf{f}_b = [0, 0, 0, 0, 0, 0, 0, k_1, -k_2, k_1, 0, 0, 0, 0, 0, 0, 0, 0]$

$\mathbf{f}_{br} = [0, -l_1, -l_2, l_3, l_4, -l_5, -l_6, l_7, l_8, -l_9, l_8, l_7, -l_6, -l_5, l_4, l_3, -l_2, -l_1]$

And the corresponding member condition-II yields

$$\left.\begin{aligned} &l_1k_2 - l_2k_1 = 0 \\ &-l_2k_1 - l_3k_2 + l_4k_1 = 0 \\ &l_4k_1 + l_5k_2 - l_6k_1 = 0 \\ &-l_6k_1 - l_7k_2 + l_8k_1 = 0 \end{aligned}\right\} \cdots\cdots\cdots(b.11)$$

As equation (*b*.11) is free of l_9, it has no role in perfect reconstruction. Thus, we can choose any real number as its value. The values of the other variable coefficients of the members of the framework are evaluated from equation (*b*.11). And we can find infinitely many PRF-18 sets of the above framework.

B.3.4 Framework of PRF sets of Bior3.1

The filters of Bior3.1 are of length 4. One of the decomposition filter is symmetric and the other anti-symmetric. Similar is the case for the reconstruction filters. The framework of PRF sets of Bior3.1 is given below.

$\mathbf{f}_a = [-l_1, l_2, l_2, -l_1], \quad \mathbf{f}_{ar} = [k_1, k_2, k_2, k_1]$

$\mathbf{f}_b = [-k_1, k_2, -k_2, k_1], \quad \mathbf{f}_{br} = [-l_1, -l_2, l_2, l_1]$

And the corresponding member condition-II yields

$-l_1k_2 + l_2k_1 = 0 \cdots\cdots\cdots(b.12)$

All the filter coefficients of the members of the framework are obtained from equation (*b*.12).

B.3.5 Framework of PRF sets of Bior3.3

$\mathbf{f}_a = [l_1, -l_2, -l_3, l_4, l_4, -l_3, -l_2, l_1], \quad \mathbf{f}_{ar} = [0, 0, k_1, k_2, k_2, k_1, 0, 0]$

$\mathbf{f}_b = [0, 0, -k_1, k_2, -k_2, k_1, 0, 0], \quad \mathbf{f}_{br} = [l_1, l_2, -l_3, -l_4, l_4, l_3, -l_2, -l_1]$

Member condition-II yields

$$\left.\begin{aligned} l_1k_2 - l_2k_1 &= 0 \\ l_1k_1 - l_2k_2 - l_3k_2 + l_4k_1 &= 0 \end{aligned}\right\} \cdots\cdots\cdots (b.13)$$

B.3.6 Framework of PRF sets of Bior3.5

$\mathbf{f}_a = [-l_1, l_2, l_3, -l_4, \ -l_5, l_6, l_6, -l_5, -l_4, l_3, l_2, -l_1],$

$\mathbf{f}_{ar} = [0, 0, 0, 0, k_1, k_2, k_2, k_1, 0, 0, 0, 0]$

$\mathbf{f}_b = [0, 0, 0, 0, -k_1, k_2, -k_2, k_1, 0, 0, 0, 0],$

$\mathbf{f}_{br} = [-l_1, -l_2, l_3, l_4, \ -l_5, -l_6, l_6, l_5, -l_4, -l_3, l_2, l_1]$

Member condition-II yields

$$\left.\begin{aligned} -l_1k_2 + l_2k_1 &= 0 \\ -l_1k_1 + l_2k_2 + l_3k_2 - l_4k_1 &= 0 \\ l_3k_1 - l_4k_2 - l_5k_2 + l_6k_1 &= 0 \end{aligned}\right\} \cdots\cdots\cdots (b.14)$$

B.3.7 Framework of PRF sets of Bior3.7

$\mathbf{f}_a = [l_1, -l_2, -l_3, l_4, \ l_5, -l_6, -l_7, l_8, l_8, -l_7, -l_6, l_5, l_4, -l_3, -l_2, l_1]$

$\mathbf{f}_{ar} = [0, 0, 0, 0, 0, 0, k_1, k_2, k_2, k_1, 0, 0, 0, 0, 0, 0]$

$\mathbf{f}_b = [0, 0, 0, 0, 0, 0, -k_1, k_2, -k_2, k_1, 0, 0, 0, 0, 0, 0]$

$\mathbf{f}_{br} = [l_1, l_2, -l_3, -l_4, \ l_5, l_6, -l_7, -l_8, l_8, l_7, -l_6, -l_5, l_4, l_3, -l_2, -l_1]$

Member condition-II yields

$$\left.\begin{aligned} l_1k_2 - l_2k_1 &= 0 \\ l_1k_1 - l_2k_2 - l_3k_2 + l_4k_1 &= 0 \\ -l_3k_1 + l_4k_2 + l_5k_2 - l_6k_1 &= 0 \\ l_5k_1 - l_6k_2 - l_7k_2 + l_8k_1 &= 0 \end{aligned}\right\} \cdots\cdots\cdots (b.15)$$

B.3.8 Framework of PRF sets of Bior3.9

$\mathbf{f}_a = [-l_1, l_2, l_3, -l_4, \ -l_5, l_6, l_7, -l_8, l_9, l_{10}, l_{10}, l_9, -l_8, l_7, l_6, -l_5, -l_4, l_3, l_2, -l_1]$

$\mathbf{f}_{ar} = [0, 0, 0, 0, 0, 0, 0, 0, k_1, k_2, k_2, k_1, 0, 0, \ 0, 0, 0, 0, 0, 0]$

$\mathbf{f}_b = [0, 0, 0, 0, 0, 0, 0, 0, 0, -k_1, k_2, -k_2, k_1, 0, 0, 0, 0, 0, 0, 0, 0]$

$\mathbf{f}_{br} = [-l_1, -l_2, l_3, l_4, -l_5, -l_6, l_7, l_8, l_9, -l_{10}, l_{10}, -l_9, -l_8, -l_7, l_6, l_5, -l_4, -l_3, l_2, l_1]$

Member condition-II yields

$$\left.\begin{aligned} &-l_1k_2 + l_2k_1 = 0 \\ &-l_1k_1 + l_2k_2 + l_3k_2 - l_4k_1 = 0 \\ &l_3k_1 - l_4k_2 - l_5k_2 + l_6k_1 = 0 \\ &-l_5k_1 + l_6k_2 + l_7k_2 - l_8k_1 = 0 \\ &l_7k_1 - l_8k_2 + l_9k_2 + l_{10}k_1 = 0 \end{aligned}\right\} \cdots\cdots\cdots (b.16)$$

B.3.9 Framework of PRF sets of Bior4.4

$\mathbf{f}_a = [0, l_1, -l_2, -l_3, l_4, l_5, l_4, -l_3, -l_2, l_1]$, $\mathbf{f}_{ar} = [0, -k_1, -k_2, k_3, k_4, k_3, -k_2, -k_1, 0, 0]$

$\mathbf{f}_b = [0, -k_1, k_2, k_3, -k_4, k_3, k_2, -k_1, 0, 0]$, $\mathbf{f}_{br} = [0, -l_1, -l_2, l_3, l_4, -l_5, l_4, l_3, -l_2, -l_1]$

Member condition-II yields

$$\left.\begin{aligned} &-l_1k_2 + l_2k_1 = 0 \\ &l_1k_4 - l_2k_3 + l_3k_2 - l_4k_1 = 0 \\ &-l_1k_2 - l_2k_3 - l_3k_4 + l_4(k_3 - k_1) - l_5k_2 = 0 \end{aligned}\right\} \cdots\cdots\cdots (b.17)$$

B.3.10 Framework of PRF sets of Bior5.5

$\mathbf{f}_a = [0, 0, l_1, l_2, -l_3, l_4, l_5, l_4, -l_3, l_2, l_1, 0]$,

$\mathbf{f}_{ar} = [\ k_1, -k_2, -k_3, -k_4, k_5, k_6, k_5, -k_4, -k_3, -k_2, k_1, 0]$

$\mathbf{f}_b = [-k_1, -k_2, k_3, -k_4, -k_5, k_6, -k_5, -k_4, k_3, -k_2, -k_1, 0]$,

$\mathbf{f}_{br} = [0, 0, l_1, -l_2, -l_3, -l_4, l_5, -l_4, -l_3, -l_2, l_1, 0]$

Member condition-II yields

$$\left.\begin{aligned} &-l_1k_2 + l_2k_1 = 0 \\ &-l_1k_4 - l_2k_3 + l_3k_2 + l_4k_1 = 0 \\ &l_1k_6 + l_2k_5 + l_3k_4 - l_4(k_3 - k_1) - l_5k_2 = 0 \\ &-l_1k_4 + l_2(k_5 + k_1) + l_3(k_2 - k_6) + l_4(k_5 - k_3) - l_5k_4 = 0 \end{aligned}\right\} \cdots\cdots\cdots (b.18)$$

B.3.11 Framework of PRF sets of Bior6.8

The filters of bior6.8 are of length 18. Their framework is given below.

$\mathbf{f}_a = [0, l_1, -l_2, -l_3, l_4, l_5, -l_6, -l_7, l_8, l_9, l_8, -l_7, -l_6, l_5, l_4, -l_3, -l_2, l_1]$

$\mathbf{f}_{ar} = [0, 0, 0, k_1, k_2, -k_3, -k_4, k_5, k_6, k_5, -k_4, -k_3, k_2, k_1, 0, 0, 0, 0]$

$\mathbf{f}_b = [0, 0, 0, k_1, -k_2, -k_3, k_4, k_5, -k_6, k_5, k_4, -k_3, -k_2, k_1, 0, 0, 0, 0]$

$\mathbf{f}_{br} = [0, 0, -l_1, -l_2, l_3, l_4, -l_5, -l_6, l_7, l_8, -l_9, l_8, l_7, -l_6, -l_5, l_4, l_3, -l_2, -l_1, 0]$

Member condition-II yields

$$\left.\begin{aligned}
&l_1k_2 - l_2k_1 = 0\\
&-l_1k_4 + l_2k_3 - l_3k_2 + l_4k_1 = 0\\
&l_1k_6 - l_2k_5 + l_3k_4 - l_4k_3 + l_5k_2 - l_6k_1 = 0\\
&-l_1k_4 - l_2k_5 - l_3k_6 + l_4k_5 - l_5k_4 + l_6k_3 - l_7k_2 + l_8k_1 = 0\\
&l_1k_2 + l_2k_3 + l_3k_4 + l_4k_5 + l_5k_6 - l_6k_5 + l_7k_4 + l_8(k_1 - k_3) + l_9k_2 = 0\\
&-l_2k_1 - l_3k_2 - l_4k_3 - l_5k_4 - l_6(k_5 + k_1) - l_7k_2 + l_8(k_5 - k_3) - l_9k_4 = 0
\end{aligned}\right\} \cdots\cdots(b.19)$$

Here, all the members of the framework have some zero elements and all the nonzero elements are symmetric about the center of the nonzero elements. The members of the frameworks of Bior5.5 and 6.8 also have the same symmetry. The actual PRF sets of these frameworks can be found from their corresponding member condition-II equations.

APPENDIX C

Finding optimal PRF sets through optimization

Selection of appropriate filters is a common problem encountered in various applications of signal processing. Such a selection for image compression has been studied based on number of vanishing moments and regularity of filters. But it has been shown that neither the filters having higher vanishing moments nor those with regularity always give satisfactory results for compression purpose. In our case, the choice of suitable filters is made using an optimization technique, namely, genetic algorithms[♣]. A genetic algorithm finds the maximum value of a function $g(x_1, x_2, ..., x_p)$ where $x_1, x_2, ..., x_p$ are free variables. However, in the present case of a framework of PRFL sets, $f_1, f_2, \cdots, f_L$ are not free variables. For this, a framework *F* of PRFL sets is mapped to a lower dimensional space Ω with free variables such that there is a one-to-one correspondence between *F* and Ω. In other words, for each element in Ω, there is a PRF set of length *L* in *F* and vice versa. Now, if a reward function is defined for every PRF set with respect to some criterion, then it can be looked upon as a function on Ω as well. A genetic algorithm in that case can find an optimal element in Ω which will correspond to the best PRF set with respect to the same criterion. How the space Ω is determined and how a reward function is defined for a PRF set are described below.

The decomposition type considered here has an approximation component and a detail component. An image is thus decomposed into four components (one approximation and three details) and the image is reconstructed from only the approximation component by replacing the three detail components with zeros. A PRF set is assumed to be better for compression as compared to another PRF set if the MSE between original image and

[♣] D. E. Goldberg, Genetic algorithms in search, optimization and machine learning, Addison-Wesley, 1989.

reconstructed image from the approximation component is smaller for the first PRF set. The goal is to find a PRF set for which the MSE value is minimum, for a given image.

For any PRF set given by $f_1, f_2, \cdots, f_L$, a function $g(f_1, f_2, ..., f_L)$ is defined as *1/MSE* where *MSE* is the MSE value obtained after reconstructing the original image from the first decomposed (approximation) component on the basis of $f_1, f_2, \cdots, f_L$. However, employing an optimization algorithm to find the optimal PRF set with respect to such a function is not possible in wavelets or in filter bank. But in ISITRA finding such optimal PRF sets is possible as can be seen below.

In a framework *F* of PRFL sets, there are *L* scalar variables $f_1, f_2, \cdots, f_L$ and their values determine a PRFL set. However, for perfect reconstruction, the values of these *L* variables cannot be chosen independently, since they should satisfy condition C_2. So, we map the set of all PRFL sets satisfying condition C_2 to a space $\boldsymbol{\Omega}$ with a lower dimension in which the variables can be chosen independently. In that case, for any element $\boldsymbol{\theta}$ in Ω, there is a PRFL set determined by $g_1(\boldsymbol{\theta}) = f_1, f_2, \cdots, f_L$. For any given image, the value *1/MSE* arising from $g_1(\boldsymbol{\theta})$ after reconstruction can be looked upon as a function $h(\boldsymbol{\theta})$ of $\boldsymbol{\theta}$ where $h(\boldsymbol{\theta}) = g(g_1(\boldsymbol{\theta}))$. Thus, $h(\boldsymbol{\theta})$ is a real valued function defined on Ω. An optimization algorithm is used here to find $\boldsymbol{\theta}_0$ in Ω such that $h(\boldsymbol{\theta})$ is optimized at $\boldsymbol{\theta}_0$. It is easy to see that the function $h(\boldsymbol{\theta})$ is quite complex and does not have a close form. Thus genetic algorithms are a natural choice for an optimization algorithm. The method for getting $g_1(\boldsymbol{\theta})$ from $\boldsymbol{\theta}$ is described in detail below for $L = 4$, and $L = 6$.

Case-1. $L = 4$.

Here is the 2-dimensional space $\Omega\ \{\boldsymbol{\theta} = (\theta_1, \theta_2) : 0 \le \theta_1, \theta_2 \le 2\pi\}$. For any element (θ_1, θ_2) in Ω, we define $f_1 = \sin\theta_1 \sin\theta_2$, $f_2 = \sin\theta_1 \cos\theta_2$, $f_3 = \cos\theta_1$. Note that the point (f_1, f_2, f_3) lies on the surface of the unit sphere with the center at the origin. From equation member condition-II equation (c.1) of length-4 PRF set given below, we get f_4 as $f_4 = (f_1 f_2) / f_2$.

$$f_1 f_3 + f_2 f_4 = 0 \cdots\cdots (c.1)$$

Once the values of f_1, f_2, f_3, f_4 are found, the corresponding PRF4 set of the framework F_1 given in (c.2) is determined.

$$\left.\begin{aligned} \mathbf{f}_a &= [-f_1, f_2, f_3, f_4],\ \mathbf{f}_{ar} = [f_4, f_3, f_2, -f_1] \\ \mathbf{f}_b &= [f_4, -f_3, f_2, f_1],\ \mathbf{f}_{br} = [f_1, f_2, -f_3, f_4] \end{aligned}\right\} \cdots\cdots (c.2)$$

There is a one-to-one relation between Ω and F_1.

Case-2. $L = 6$.

Here is the 3-dimensional space $\Omega\{\boldsymbol{\theta} = (\theta_1, \theta_2, \theta_3) : 0 \le \theta_1, \theta_2, \theta_3 \le 2\pi\}$. For any element $(\theta_1, \theta_2, \theta_3)$ in Ω, we define $f_1 = \sin\theta_1 \sin\theta_2 \sin\theta_3$, $f_2 = \sin\theta_1 \sin\theta_2 \cos\theta_3$, $f_3 = \sin\theta_1 \cos\theta_2$, $f_4 = \cos\theta_1$. Note that the point (f_1, f_2, f_3, f_4) lies on the surface of the unit 4-dimensional sphere with the center at the origin. From the member condition-II equation (c.3) of the length-6, PRF set given below, we get the values of f_5 and f_6.

$$\left.\begin{aligned} f_1 f_5 - f_2 f_6 &= 0 \\ -f_1 f_3 - f_2 f_4 - f_3 f_5 + f_4 f_6 &= 0 \end{aligned}\right\} \cdots\cdots (c.3)$$

Once the values of $f_1, f_2, f_3, f_4, f_5, f_6$ are found, we can determine frameworks that satisfy member condition of ISITRA-1. We consider here two frameworks, F_2 and F_3 given respectively in equations (c.4) and (c.5) below. .

$$\left.\begin{aligned} \mathbf{f}_a &= [f_1, -f_2, -f_3, f_4, f_5, f_6] \\ \mathbf{f}_{ar} &= [f_6, f_5, f_4, -f_3, -f_2, f_1] \\ \mathbf{f}_b &= [-f_6, f_5, -f_4, -f_3, -f_2, f_1] \\ \mathbf{f}_{br} &= [f_1, -f_2, -f_3, -f_4, f_5, -f_6] \end{aligned}\right\} \cdots\cdots (c.4)$$

$$\left.\begin{aligned} \mathbf{f}_a &= [f_1, -f_2, -f_3, f_4, f_5, f_6] \\ \mathbf{f}_{ar} &= [f_6, f_5, f_4, -f_3, -f_2, f_1] \\ \mathbf{f}_b &= [-f_6, f_5, -f_4, -f_3, f_2, f_1] \\ \mathbf{f}_{br} &= [f_1, f_2, -f_3, -f_4, f_5, -f_6] \end{aligned}\right\} \cdots\cdots (c.5)$$

There is a one-to-one relation between Ω and F_2 or F_3.

We implement our genetic algorithm on Ω with dimensions 2 and 3 in Case-1 and Case-2 respectively. The function to be maximized is $h(\boldsymbol{\theta})$.

B.1 Experimental Results

In our genetic algorithm, the population size is taken as 20 while the crossover probability is fixed at 0.7. In every session the genetic algorithms runs for 150 generations and the mutation probability is reduced according to the schedule: 0.75 in the first 50 generations, 0.25 in the second 50 generations and 0.10 in the last 50 generations. Mutation operation is in fact exploration in the search space while crossover operation indicates exploitation. In the initial stage of search, exploration should take place to a great extent and as the search continues, its occurrence should reduce for an efficient convergence.

For our experiment, we consider three different images (Lena, Barbara and Boat) each of two different sizes (128x128 and 256x256). These six images are Lena128, Lena256, Barb128, Barb256, Boat128 and Boat256. Zero padding in images is used in our decomposition scheme.

For the framework F_1 of PRF4 sets (L=4), the genetic algorithm is employed in the 2-dimensional space Ω to find the best $\boldsymbol{\theta} = (\theta_1, \theta_2)$ giving the minimum MSE value for each of the six images. It is found that the MSE value thus obtained is smaller than that obtained by the Db2 filters for each of the six images. The optimizing values of $\boldsymbol{\theta} = (\theta_1, \theta_2)$ are not exactly the same for all the images, but are quite close to one another in the space Ω. These optimizing values along with the corresponding MSE values are given in Table C.1. The average of the six values of $\boldsymbol{\theta} = (\theta_1, \theta_2)$ is (167.9863, 215.3437) and the corresponding PRF4 set is called PRF4-OPT (given in Table C.2). PRF4-OPT can be looked upon as the best ISITRA PRF4 set found by the genetic algorithm on the basis of the six images as far as the MSE value is concerned. The six MSE values found with the PRF set PRF4-OPT are shown in Table C.3 and it is seen that in each case this MSE value also is lower than that obtained by the Db2 filters. Thus, PRF4-OPT is empirically found to be better than the Db2 filters.

In a similar fashion, for the framework F_2 of PRF6 sets (L=6), the genetic algorithm is employed in the 3-dimensional space Ω to find the best $\boldsymbol{\theta} = (\theta_1, \theta_2, \theta_3)$ giving the minimum MSE value for each of the six images. It is observed that for each image there are two optimizing values of $\boldsymbol{\theta} = (\theta_1, \theta_2, \theta_3)$ with MSE values that are significantly small. These twelve optimizing values are found to lie around two distinct 3-dimensional points in

Ω. These two sets of values and the corresponding MSE values are given in Table C.1. The MSE values obtained are significantly smaller than those obtained by the Db3 filters. From the two sets of values of $\boldsymbol{\theta} = (\theta_1, \theta_2, \theta_3)$, two averages are computed as (97.6321, 255.6036, 326.5542) and (32.4162, 29.2266, 33.4760). The two corresponding PRF6 sets are respectively called PRF6-OPT1 and PRF6-OPT2 (Table C.2). PRF6-OPT1 and PRF6-OPT2 can be thought of as the two best PRF6 sets of ISITRA found by the genetic algorithm on the basis of the six images in terms of the MSE value. The MSE values for the six images found with PRF6-OPT1 and PRF6-OPT2 filters are shown in Table C.3 and it is seen that in each case the MSE value is significantly lower than that obtained by the Db3 filters. Thus, both PRF6-OPT1 and PRF6-OPT2 are found to be better than the Db3 filters. It is to be noted that these two PRF sets are nearly reverse of each other.

In a similar way, the best $\boldsymbol{\theta} = (\theta_1, \theta_2, \theta_3)$ giving the minimum MSE value for each of the six images is found for the framework F_3 of PRF6 sets which includes the Db3 filters. The optimal MSE value thus obtained is lower than that obtained by the Db3 filters in each case . However, these MSE values are almost always marginally larger than both the optimal MSE values obtained in the framework F_2 (Table C.1). One can thus conclude on the basis of the six images that F_2 is a better framework of PRF6 filters than F_3. The average of six optimizing $\boldsymbol{\theta} = (\theta_1, \theta_2, \theta_3)$ values in F_2 is (124.2163, 12.4996, 31.2668) and the resulting PRF6 set is called PRF6-OPT3 (Table C.2). The MSE values for the six images using PRF6-OPT3 filters are given in Table C.3. It is seen that these MSE values are very close to those obtained with PRF4-OPT filters. On scrutiny, it is found that the entries in $\mathbf{f}_a$ of PRF4-OPT filters are very close to the first four entries in $\mathbf{f}_a$ of PRF6-OPT3 filters. In fact, it can be seen that if two zeros are appended at the end of $\mathbf{f}_a$ of a PRF4 set in F_1, the PRF6 set corresponding to the new $\mathbf{f}_a$ belongs to F_3. Thus, in a sense, F_1 represents a subclass of F_3 and incidentally, the best PRF6 set in F_3 in fact corresponds to the best PRF4 set in F_1. In other words, F_3 with longer PRF sets does not produce a better PRF set than F_1 with shorter PRF sets. Thus, in this respect also, F_2 is a better framework than F_3 which incidentally contains Db3. Another interesting observation is that for longer DB

filters, the MSE value becomes larger for each of the six images while in ISITRA longer optimal PRF sets tend to give smaller MSE values (Table C.3).Similar empirical observations are found with L=8 [124].

Framework		Lena128	Lena256	Barb128	Barb256	Boat128	Boat256	Average $\boldsymbol{\theta}$
F_1 (L= 4)	MSE Value	136.1400	80.0596	134.0195	107.4704	178.6759	132.1466	PRF4-OPT
	θ_1	166.5395	165.9530	167.9765	169.0469	170.1173	168.2844	167.9863
	θ_2	214.1348	213.4017	215.4447	216.2658	217.2434	215.5718	215.3437
F_2 (L= 6)	MSE Value	129.5042	73.0901	126.6304	102.5910	180.5500	125.5415	PRF6-OPT1
	θ_1	97.4975	98.3137	97.9912	97.9032	95.9784	98.1085	97.6321
	θ_2	256.8719	252.2873	254.5259	254.9560	261.2561	253.7243	255.6036
	θ_3	325.9139	328.7341	327.1163	327.5122	322.5366	327.5122	326.5542
F_2 (L= 6)	MSE Value	125.2815	73.0838	128.6302	105.8416	174.3297	127.1909	PRF6-OPT2
	θ_1	30.9286	28.7390	31.1534	34.2365	35.8748	33.5650	32.4162
	θ_2	32.5122	32.9618	29.1202	23.7478	26.6666	30.3509	29.2266
	θ_3	32.2580	30.4789	32.4828	35.0371	36.2463	34.3528	33.4760
F_3 (L= 6)	MSE Value	135.8183	79.8160	133.8937	107.4420	179.3537	132.0448	PRF6-OPT3
	θ_1	123.0791	122.1163	124.7800	125.5337	124.9853	124.8035	124.2163
	θ_2	13.7243	14.4672	12.0625	11.1749	11.6715	11.8973	12.4996
	θ_3	30.0439	28.7976	32.5366	34.0371	29.6578	32.5278	31.2668

Table C.1 Optimal MSE values with corresponding values of $\boldsymbol{\theta}$ obtained by genetic algorithm

PRF Sets	Coefficients of $\mathbf{f}_a$
Db2	-0.129409, 0.224143, 0.836516, 0.482962
PRF4-OPT	0.098936, -0.139507, -0.803678, -0.569958
Db3	0.035226, -0.085441, -0.135011, 0.459877, 0.806892, 0.332671
PRF6-OPT1	0.041418, -0.062632, -0.134208, 0.242176, 0.799056, 0.528403
PRF6-OPT2	0.527162, 0.798096, 0.245518, -0.132323, -0.071553, 0.047263
PRF6-OPT3	0.092886, -0.152970, -0.807269, -0.562284, 0.009601, 0.005830

Table C.2 Filter coefficients of $\mathbf{f}_a$ of DB filters and optimal filters of ISITRA-1

PRF Sets	Lena128	Lena256	Barb128	Barb256	Boat128	Boat256
Db2	142.1820	82.4187	142.8773	113.3956	195.8668	138.3945
PRF4-OPT	136.8728	80.8597	134.0977	107.6207	180.0317	132.1815
Db3	148.3420	83.2028	154.0570	120.1949	221.1716	146.4211
PRF6-OPT1	132.0533	76.3091	128.9876	105.0484	188.0865	128.0543
PRF6-OPT2	125.7939	73.9672	128.1042	106.1989	177.4919	127.3481
PRF6-OPT3	136.4138	80.5084	133.9818	107.6771	180.3144	132.1041

Table C.3 MSE values for DB filters and optimal filters of ISITRA-1

B.2 Discussion and Conclusion

A method has been proposed for finding optimal PRF sets in ISITRA scheme of signal decomposition and reconstruction and the optimization technique used for the purpose is genetic algorithms. In the existing schemes like filter bank or wavelets, the space of PRF sets is not appropriate for such optimization. The space of PRF sets in ISITRA is constrained and thus is not also suitable for an optimization technique like genetic algorithms. However, a lower dimensional space is constructed where the variables are unconstrained and there is a one-to-one correspondence between this space and the space of ISITRA PRF sets. Then genetic algorithms are applied to the lower dimensional space and the optimal points are found in terms of MSE values. From these optimal points the optimal PRF sets are computed. An interesting feature to be noted here is that the longer PRF sets of ISITRA result in lower MSE values, contrary to the observation seen in the case of Db filters. The criterion of optimization here is the MSE value. However, any other quantitative criterion like energy compaction can also be used. The MSE value used in the present study is defined on the basis of a particular lossy compression scheme. The MSE value using any other compression scheme can also be the basis of optimization of PRF sets of ISITRA. Optimal PRF sets of lengths 4 and 6 only have been presented and they are found to be better than Daubechies' orthonormal filters. A similar study of longer PRF sets of ISITRA can also be done. Also, for any value of L, several frameworks of PRF sets are possible of which only a few have been considered here. Other frameworks may produce optimal PRF sets that are better than the ones obtained so far.

REFERENCES

[1] M. D. Adams and F. Kossentini, "Reversible integer-to-integer wavelet transforms for image compression: Performance evaluation and analysis", *IEEE Trans. Image Processing*., vol.9, pp.1010-1024, 2000.

[2] E.H. Adelson, E. Simoncelli and R. Hingorani, " Orthogonal pyramid transforms for image coding", *Proc. SPIE*, vol.845, pp.50-58, Cambridge, MA, Oct. 1987.

[3] Ramesh C. Agarwal, Charles S. Burrus, "Fast Convolution Using Fermat Number Transforms with Applications to Digital Filtering", IEEE Trans. Acoustics, Speech and Signap Processing, vol. 22, pp.87-97, 1974.

[4] Ramesh C. Agarwal, James W. Cooley, "New Algorithms for Digital Convolution", IEEE Trans. Acoustics, Speech and Signap Processing, vol. 25, pp.392-410, 1977.

[5] A. N. Akansu and R. A. Hadad., *Multi-resolution Signal decomposition: Transforms, Subbands, wavelets*, San Diego, CA: Academic Press, 1992.

[6] A.N. Akansu, M.V. Tazebay and R.A. Haddad, "A new look at digital orthogonal transmultiplexers for CDMA communications", *IEEE Trans. Signal Processing*., vol.45, pp.263-267, 1997.

[7] A.N. Akansu, P. Duhamel, X. Lin and M.De Courville, " Orthogonal transmultiplexers in communication: a review", *IEEE Trans. Signal Processing*., vol. 46, pp. 979-995, 1998.

[8] A. Aldroubi and M. Unser, "Families of multiresolution and wavelet spaces with optimal properties", *Numer. Funct. Anal.Optim*., vol.14, pp.417-446, 1993.

[9] C. Alexopoulos, Nikolaos G. Bourbakis and N. Ioannou, "Image encryption method using a class of fractals", *J. Electronic Imaging*, vol.4, pp.251-259, 1995.

[10] American National Standard Institute, "Financial institution key management (wholesale)", *ANSIX9.17-1985*, 1985.

[11] K. Andra, C. Chakrabarti and T. Acharya, "A High performance JPEG2000 Architecture", *Proc. InternationalSymposium on Circuits and Systems (ISCAS'2002)*, Scottsdale, Arizona, U.S.A, May 2002.

[12] M. Antonini, M. Barlaud, P. Mathieu and I. Daubechies, "Image coding using wavelet transform", *IEEE Trans. Image Processing*, vol.1, pp.205-220, 1992.

[13] K. Atkinson, *Elementary Numerical Analysis*, John Wiley & Sons, 1985.

[14] G. Battle, “A block spin construction of ondelettes: Part-I, Lemarie functions”, *Commun. Math. Phys.*, vol.110, pp.601-615, 1987.

[15] G. Battle, “A block spin construction of ondelettes: Part-II, The QFT connection”, *Commun. Math. Phys.*, vol.114, pp.93-102, 1988.

[16] G. Beylkin, R. Coifman and V. Rokhlin, “Fast wavelet transforms and numerical algorithms”, *Communications in Pure and Applied Mathematics*, vol 44, pp.141-183, 1991.

[17] M.G. Bellanger and J.L.Dauget. TDM-FDM Transmultiplexer: Digital polyphase and FFT., *IEEE Trans. Commun.*, vol.22, pp.1199-1204, 1974.

[18] M. G. Bellanger , “On computational complexity in digital transmultiplexer filters”, *IEEE Trans. Commun.*, vol.30, pp.1461-1465, 1982.

[19] L. Blum, M. Blum and M. Shub, “A simple unpredictable pseudo random number generator.”, *SIAM Journal on Computing*, vol.15, pp.364-383, 1986.

[20] A. Bogomolny, “*Base converter*”, http://www.cut-the-knot.com/binary.html.

[21] N. Bourbakis and C. Alexopoulos, “Picture data encryption using scan patterns”, *Pattern Recognition*, vol.25, no.6, pp.567-581, 1992.

[22] P. J. Burt and E,H. Adelson, “The Laplacian pyramid as a compact image code”, *IEEE Trans. Commun.*, vol.31, pp.532-540, 1983.

[23] C. S. Burrus, R. A. Gopinath and H. Guo, *Introduction to wavelets and wavelet transforms: A primer*. Englewood Cliffs, NJ: Prentice Hall, 1997.

[24] R. Caulderbank, I. Daubechies. W. Sweldens and B.L. Yeo, “Wavelet transforms that maps integer to integers”, *Appl.Comput. Harmon.Anal.*, vol.5, pp.332-369, 1998

[25] Henry Ker-Chang Chang and Jiang-Long Liu, “A linear quadtree compression scheme for image encryption”, *Signal Processing: Image Communication*, vol.10, pp.279-290, 1997

[26] Kuan-Fu Chen, Chung-Jr Lian, Hong-Hui Chen, Liang-Gee Chen, “Analysis and Architecture Design of EBCOTfor JPEG2000”, *Proc. International Symposium on Circuits and Systems (ISCAS'2001)*, Sydney, Australia, May 2001.

[27] Howard Cheng and Xiaobo Li, “ Partial encryption of compressed images and videos”, *IEEE Trans. Signal Processing*, vol.48, pp.2439-2451, 2000.

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

[29] R.E. Chrochiere and L. R. Rabiner, *Mutlirate Digital Signal Processing*, Prentice-Hall, Englewood Cliffs, NJ, 1983.

[30] C.K. Chui, *An introduction to Wavelets.* Academic Press, San Diego, CA, 1992.

[31] C. K. Chui, editor. *Wavelets: A tutorial in Theory and Applications*. Academic Press, San Diego, CA, 1992.

[32] T. J. Chuang and Lin J.C., "A new multiresolution approach to still image encryption", *Pattern Recognit. Image Anal*., vol.9, pp.431-436, 1999.

[33] Diego Clonda, J. M. Lina and Bernard Goulard, "Complex Dubechies wavelets: Properties and statistical image modelling", *Signal Processing*, vol.84, pp.1-23, 2004.

[34] A. Cohen, I. Daubechies and J. Feauveau. "Biorthogonal bases of compactly supported wavelets", *Comm. Pure Appl. Math.*, vol.45, pp.485-560, 1992.

[35] R.R. Coifman and M.V. Wickerhauser, "Entropy-based algorithms for best basis selection", *IEEE Trans. Information Theory*, vol.38, pp.713-718, 1992.

[36] R. R. Coifman and D.L. Donoho, "Translation invariant denoising" in *Wavelets and Statistics*, A. Antoniadis and G. Oppenheim, Eds, San Diego CA: Springer Verlag, 1995, Lectures Notes.

[37] J. H. Conway and R.K. Guy, "Pascal Triangle" *In the book of numbers*, New York: Springer Verlag, pp. 68-70, 1996

[38] A. Crosier, D. Esteban and C. Galland, "Perfect channel splitting by use of interpolation/Decimation/tree decomposition Techniques", in Proc. Int. Symp. On Information, Circuit and Systems, (Patras, Greece), 1976.

[39] I. Daubechies. "Orthonomal bases of compactly supported wavelets". *Comm. Pure Appl. Math.*, vol.41, pp.909-996, 1988.

[40] I. Daubechies, *Ten Lectures on Wavelets*. CBMS-NSF Regional Conf. Series in Appl. Math., vol.61. Society for Industrial and Applied Mathematics, Philadelphia, PA, 1992.

[41] I. Daubechies and W. Sweldens, "Factoring wavelet transforms into lifting steps", *J. Fourier Anal. Appl*, vol.4, pp.145-267, 1998.

[42] R.A. Devore, B. Jawerth and B.J. Lucier, "Image compression through wavelet transform coding", *IEEE Trans. Inform. Theory*, vol.38, pp.719-746, 1992

[43] Minh N. Do and Martin Vetterli, "The finite ridgelet transform for image representation", *IEEE Trans, Image Processing*, vol.12, pp.16-28, 2003.

[44] D. L. Donoho, "Denoising by soft-thresholding", *IEEE Trans. Inf. Theory*, vol.41, pp.613-627, 1995.

[45] D. L. Donoho, I. M. Johnstone, G. Karkyacharian and D. Picard, "Density estimation by wavelet thresholding", *Ann. Statist*., vol.24, pp.508-539, 1996.

[46] W. Dunham, The Mathematical Universe, John Wiley & Sons, 1994.

[47] Felix C.A. Fernandes, Rutger L.C Van Spandonck and C. Sidney Burrus, "A new framework for complex wavelet transforms", *IEEE Trans. Signal Processing*, vol.51, pp.1825-1837, July 2003.

[48] N.J. Fliege, *Multirate Signal Processing*, Academic Press, 1992.

[49] D. Gabor, "Theory of communication", *Journal IEE*, vol.93, pp.429-457, 1946.

[50] R. A. Gopinath, "Phaselet-transform: an integral redundancy nearly shift invariant wavelet transform", *IEEE Trans. Signal Processing*, vol.51, pp.1792-1805, 2003.

[51] A. Grossman and J. Morlet, "Decomposition of hardy functions into square integrable wavelets of constant shapes", *SIAM J. Math. Annal*., vol.15, pp.723-736, 1984.

[52] Alfred Haar, "Zur theorie der orthogonalen funktionen-systeme", Math. Annal., vol. 9, pp.331-371,1910.

[53] Harborth, H. "Number of Odd Binomial Coefficients", *Proc. Amer. Math. Soc*, vol.62, pp.19-22, 1977.

[54] A. Harten. "Multiresolution representation of data: A general framework", *SIAM J. Numer. Anal*., vol.33, pp.1205-1256, 1996.

[55] V. K. Heer and H. E. Reinfelder, "A comparison reversible methods for data compression", in Medical Imaging IV, pp.354-365, Proc. SPIE 1233, SPIE, Bellingham, WA, 1990.

[56] P.Hellekalek, "Good random number generators are (not so) easy to find", *Mathematics and Computers in Simulation*, vol.46, pp.485-505, 1998

[57] C. Herley and M. Vetterli. "Wavelets and recursive filter banks", *IEEE Trans. Signal Processing*, vol. 41, pp. 2536-2556, 1993.

[58] E. Harnandez and G. Weiss, *A first course on wavelets*, Boca Raton, FL: CRC Press, 1996.

[59] Emmanuel C. Ifeachor and Barrie W. Jervis, *Digital Signal Processing*, Addison – Wesley Publishing Company, 1995.

[60] ISO/IEC FCD15444-1, Information technology – JPEG 2000 Image Coding System, JPEG 2000 Final Committee Draft Version 1.0, 16 March 2000

[61] V.K. Jain and R.E. Crochiere, " Quadrature mirror filter design in time domain", *IEEE Trans. ASSP*, vol. 32, pp.353-361, 1984..

[62] Jinn-Ke Jan and Yuh-Min Tseng, "On the security of image encryption method", *Information Processing Letters*, vol.60, pp.261-265, 1996.

[63] R.L. Joshi, T.R. Fischer, V. J. Crump and T. R. Fischer, " Image subband coding using arithmetic and trelis coded quantization", *IEEE Trans. Circ. & Syst. Video Tech*., vol. 5, pp.515-523, 1995.

[64] G. Kaiser, *A friendly guide to wavelets*, Boston: Birkauser, 1994.

[65] Karatsuba A. and Ofman Yu. "Multiplication of Many-Digital Numbers by Automatic", *Doklady Akad. Nauk SSSR* **145**, 293-294, 1962. Translation in *Physics-Doklady,* vol.7, pp.595-596, 1963.

[66] J. H. Kasner and M. W. Marcellin, "Adaptive wavelet coding of images", *Proc. IEEE Conf. on Image Processing*, vol.3, pp.358-362, Austin, TX, Nov. 1994.

[67] Y.H. Kim and J.W. Modestino, "Adaptive entropy coded subband coding of images", *IEEE Trans. Image Processing*, Vol.1, pp.31-48, 1992.

[68] D. Knuth, *The art of Computer Programming vol-2: Seminumerical algorithms*, Reading, MA:Addison-wesley, 1998.

[69] R. D. Koilpillai, T. Q. Nguyen and P.P. Vaidyanathan, "Some results in the theory of crosstalk-free transmultiplexers", *IEEE Trans. Signal Processing*, vol.39, pp.2174-2183, 1991.

[70] K. Komatsu and K. Sezaki, "Reversible coding of images", Proc. of SPIE, vol.2501, pp.676-684, 1995.

[71] J. Kovacevic and M. Vetterli, "Nonseparable multidimensional perfect reconstruction filterbanks and wavelets bases for R^n ", IEEE Trans. Info. Theory, vol.38, pp.533-555, 1992.

[72] W. Lawton, "Application of complex valued wavelet transform to subband decomposition", *IEEE Trans. Signal Processing*, vol.9, pp.3566-3568, 1993.

[73] D. Lehmer, "Mathematical methods in large scale computing", *Proceedings 2nd Symposium on Large scale Digital Calculating Machinery*, pp.141-146, Cambridge: Havard University Press, 1951.

[74] P. G. Lemarie, "Ondelettes à localization exponentielle", *J. Math. Pures et Appl*.,vol. 67, pp. 227-236, 1988.

[75] P. Lewis, A Goodman and J. Miller, " A pseudo random number generator for the system/360", *IBM systems Journal*, vol.8, pp.136-146, 1969.

[76] Shujun Li and Xua Zheng, “Cryptanalysis of a chaotic image encryption method”, *Proc. 2002 IEEE Intl. Symp. Circuits and Systems* (ISCAS2002), vol.2, pp.708-711, 2002.

[77] Shujun Li and Xua Zheng, “On the security of an image encryption method”, *Proc. 2002 IEEE Intl. Conf. On Image Processing* (ICIP 2002), vol.2, pp.925-928, 2002.

[78] J. M. Lina and M. Mayrand, “Complex Daubechies wavelets”, *App. Comput. Harmon Anal*., vol.2, pp.219-229, 1995.

[79] M. W. Marcellin, M. Gormish, A. Bilgin, M. Boliek, “An Overview of JPEG 2000,” *Proc. of Data Compression Conference*, Snowbird, Utah, March 2000.

[80] V. Martin and Amy E. Bell, “New image compression techniques using multi-wavelets and wavelet packets”, IEEE Trans. Image Processing, vol.10, pp.500-510, 2001.

[81] S.G. Mallat, “A Theory of multiresolution signal decomposition: The wavelet representation”, *IEEE Trans, on Pattern Analysis and Machine Intelligence*, vol.11, pp.674-693, 1989.

[82] S. Mallat, *Wavelet Tour of Signal Processing*, San Diego, CS: Accademic, 1998.

[83] James H. McClellan and Charles M. Rader, Number Theory in Digital Signal Processing, Prentice-Hall, 1979.

[84] Peter Meerwald, Roland Norcen, Andreas Uhl, Parallel JPEG2000 Image Coding on Multiprocessors, Proceedings of the International Parallel and Distributed Processing Symposium, 2002 (IPDPS'02), available at the following website. http://citeseer.ist.psu.edu/meerwald02parallel.html.

[85] A. Menezes, P. Oorshcot and S. Vanstone, *Handbook of Applied cryptography*, Boca Raton, FL: CRC Press, 1997.

[86] A. Meyer and S. Matyas, *Cryptography: A new dimension to Computer Security*, New York: Wiley 1982.

[87] Y. Meyer, Ondelettes et operateurs, Hermann Paris, 1990.

[88] Y. Meyer, *Wavelets: Algorithms and Applications*, Philadelphia, SIAM 1993.

[89] F. Mintzer ,"Filters for distortion-free two band filter banks", *IEEE Trans. ASSP*, vol.33, pp.626-630, 1985.

[90] S.K. Mitra and James F. Kaiser, Handbook for Digital Signal Processing, Wesley Interscience, 1993.

[91] Soontorn Oraintara, Tirac D. Tran and Truong Q. Nguyen, “A class of regular bi-orthogonal linear phase filter bank: theory, structure and application in image coding” *IEEE Trans. Signal Processing*, vol.51, pp.3220-3235, Dec-2003.

[92] Oyestein Ore, *Number Theory and Its History*, Dover Publications, 1976.

[93] T. Pappas, " Pascal Triangle, the Fibonacci Sequence & Binomial Formula", "Chinese Triangle", and "Probability and Pascal Triangle", *The Joy of Mathematics*, San Carlos, CA: Wide World Publication/Tetra, pp.40-41, 88 and 184-186, 1984.

[94] S. Park and K. Miller, "Random number generators; good ones are hard to find", *Communications of the ACM*, vol.31, pp.1192-1201, 1988.

[95] Keuun Hyeong Park, Chul Soo Lee and Hyun Wook Park, " A multiresolutional coding method based on SPIHT", *Signal Processing: Image Communication*, vol.17, pp.467-476, 2002.

[96] J.A. Paulos, *Beyond Numeracy*, Vintage Books, 1992.

[97] M. Rabbini and P. W. Jones, *Digital Image Compression Techniques*, SPIE, Bellingham, WA, 1991.

[98] M. Rabbini, R. Joshi, "An overview of the JPEG2000 still image compression standard.", Signal Processing: Image Communication, vol.17, pp.3–48, 2002.

[99] Charles M. Rader, Discrete Convolution via Mersenne Transforms", IEEE Trans. Comput., vol-21, pp1269-1273, 1972.

[100] Raghuveer M. Rao and Ajit S. Bopardikar, *Wavelet Transforms: Introduction to Theory and Applications*, Addison-Wesley, Delhi, 2000.

[101] R. P. Ramachandran and P. Kabal, "Transmuliplexers: perfect reconstruction compensation of channel distortion", *Signal Processing* vol.21, pp.261-274, 1990.

[102] R. P. Ramachandran and P. Kabal, "Bandwidth eifficient transmultiplexers, part-1: synthesis", *IEEE Trans. Signal Processing*, vol.40, pp.70-84, 1992.

[103] R. P. Ramachandran and P. Kabal, "Bandwidth eifficient transmultiplexers, part-2: Subband complement s performance aspects", *IEEE Trans. signal Processing*, vol.40, pp.1108-1121, 1992.

[104] C. Ribeyre, M. Prat, X. Maitre and P. Coullare, "Exploitation of transmultiplexers in telecommunication networks", *IEEE Trans. Commun.* (Special Issue on Transmultiplexers), vol.30, pp.1493-1497, 1982.

[105] Julien Reichel, Gloria Menegaz and Marcaus J. Nadenau, "Integer wavelet transform for embedded lossy to lossless image compression", *IEEE Trans. Image Processing*, vol.10, pp.383-392, 2001.

[106] O. Rioul and P. Duhamel, "Fast algorithms for discrete and continuous wavelet transforms", *IEEE trans. Information Theory*, vol.38, pp.569-586, 1992.

[107] O. Riol, "A discrete time multi-resolution theory", *IEEE Trans. Signal Processing*, vol.41, pp.2591-2606, 1993.

[108] A. Rosenfeld, *Multiresolution Techniques in Computer Vision*, Springer Verlag, New York 1984.

[109] Amar Said and William A. Pearlman, "A new fast and efficient image codec based on set partitioning in hierarchical trees", *IEEE Trans. on Circuits and Systems for Video Tech.*, vol.6, pp.243-250, 1996.

[110] A Said and W.A. Perlman, "An image multiresolution representation for lossless and lossy compression", *IEEE trans. Image Processing*, vol.5, pp.1303-1310, 1996.

[111] H. Scheuermann and H. Göckler, "A Comprehensive survey of digital transmultiplexing methods", *Proc. IEEE*, vol.69, pp.1415-1450, 1981.

[112] J.M Shapiro, "Embedded image coding using zerotrees of wavelet coefficients", *IEEE Trans. Signal Processing*, vol.41, pp.3445-3462, 1993.

[113] F. Sheng, A. Bilgin, P. J. Sementilli and M. W. Marcellin, "Lossy and lossless image compression using reversible integer wavelet transforms", in *Proceedings 1998 International Conference on Image Processing*., vol.3, pp.876-880, Los Alamitos, CA: IEEE Press, 1998.

[114] P. L. Shui and Z. Bao, "Recursive bi-orthogonal interpolating wavelets and singal adapted interpolating filter banks", *IEEE Trans. Signal Processing*, vol.47, pp.2585-2594, 2000.

[115] P. L. Shui, Z. Bao and X. D. Zhang, "M-band compactly supported orthogonal symmetric interpolating scaling functions", *IEEE Trans. Signal Processing*, vol.49, pp.1704-1713, 2001.

[116] Y.K. Singh and S.K. Parui, "On the multiplication of two multi-digit numbers using Bino's Model of Multiplication", accepted in "*The WORLDSES MULTICONFERENCE on Applied and Theoretical Mathematics*", Athens, Greece, 1-3 December, 2000. (available at, http://www.geocities.com/kiranisingh/multi.html)

[117] Y. K. Singh and S.K. Parui, " Criteria for perfect reconstructibility of filters of QMF types", accepted in the conference "*WSES/IEEE SSIP'01,MALTA*," 1-6 September 2001. (Also available in the web site http://www:geocities.com/kiranisingh/criteria.html)

[118] Y.K Singh and S.K Parui, "Multi-level decomposition scheme of ISITRA-1 of length-2", *Proceedings of XXXVI Annual Convention*, CSI-2001, pp.79-86, Science City, Kolkata, 20-24, November, 2001.

[119] Y. K Singh and S.K Parui, "Encryption of speech signal using full decomposition of ISITRA-1 of length-2.", *Proceedings of the Sixth International Workshop on Recent Trends in Speech, Music and Allied Signal Processing (IWSMSP-2001),* pp.38-45, National Physical Laboratory, New Delhi, December, 2001.

[120] Y.K. Singh "A View on Perfect Reconstruction Theory of 2-Channel Filter Bank", *http://www.geocities.com/kiranisingh/Filterbank/criticism.htm*.

[121] Y. K. Singh and S.K. Parui, "A perfect reconstruction theory of ISITRA-*1*", accepted in the conference "*2002 WSEAS Int.Conference. on ELECTRONICS, CONTROL & SIGNAL PROCESSING*" Novotel Apollo Singapore (4 Star), Singapore, December 9-12, 2002

[122] Y. K. Singh and S.K. Parui, "Speech encryption signal using Simplet and Meitei lock sequence", *Proceedings of Frontiers of Research on Speech and Music*, FRSM-2003, pp.144-153, IIT-Kanpur, February, 2003.

[123] Y.K Singh and S.K. Parui, "Simplet and its applications in signal encryption", , *Multidimensional System and signal Processing*, vol. 15, pp. 375-394, 2004.

[124] Y. K. Singh, S.K. Parui, Shuvransu Bannerjee, "A Comparative study between wavelet and ISITRA filters", IEEE INDICON-2004 conference, Dec, 2004 IIT Kharagpur.

[125] Y. K. Singh and S.K. Parui, "ISITRA: A new generalized way of signal decomposition and reconstruction", *Digital Signal Processing*, vol. 16, pp. 3-23, 2006.

[126] D. E. Smith, *A Source Book in Mathematics*, New York: Dover, p.86, 1984.

[127] M. J. T. Smith and T. P. Barnwell. "Exact reconstruction techniques for tree-structured subband coders", *IEEE Trans. ASSP*, vol.34, pp.434-441, 1986.

[128] A. K. Soman and P. P. Vaidyanathan, "On orthonormal wavelets and filter banks", *IEEE Trans. Signal Processing,* vol.43, pp.1170-1183, 1993.

[129] Sridharan S., Dawson E. and Goldburg B., "Fast Fourier transform based speech encryption system", *IEEE Proc.-I Communication, Speech, Vision*, vol.138, pp.215-223, 1991.

[130] P. Sriram and M. W. Marcellin, "Image coding using wavelet transforms and entropy constrained trellis coded quantization", *IEEE Trans. Image Processing,* vol.4, pp.25-733, 1995.

[131] William Stallings, *Cryptography and Network Security: Principles and Practice*, Prentice Hall, New Jersey; 1995.

[132] J. L. Starck, E. J. Candes and D. L. Donoho, "Curvelet transforms for image denoising", IEEE Trans. Image Processing, vol.11, pp.670-684, 2002.

[133] P. Steffen, P. N. Heller, R. A. Gopinath and C. S. Burrus, "Theory of regular m-band wavelet bases", *IEEE Trans. Signal Processing*, vol.41, pp.3497-3510, 1993.

[134] P. H. Sterbenz, *Floating Point Computation*, Upper Saddle River, NJ; Prentice Hall, 1975.

[135] D. Stinson, *Cryptography: Theory and Practice*, Boca Raton, FL:CRC Press, 1995.

[136] G. Strang, "Wavelets and dilation equations: A brief introduction", *SIAM Rev*. vol.4, pp.614-627, 1989.

[137] G. Strang and T. Nguyen. *Wavelets and Filter Banks*, Wellesley, Cambridge, 1996.

[138] V. Strela, P. N. Heller, G. Strang, P. Topiwala and C. Heil, "The application of multiwavelet filter bank to image processing", IEEE Trans. Image Processing,, vol.8, pp.548-562, 1999.

[139] W. Sweldens "The lifting Scheme: A custom-design construction of bi-orthogonal wavelets", *Appl. Comput. Harmon.Anal*., vol.3, pp.186-200, 1996.

[140] W.Sweldens. "The Lifting Scheme: A construction of second generation wavelets", *SIAM J. Math. Anal*., vol.29, pp.511-546, 1997.", N

[141] David Taubman, "High perfomance Scalable Image Compression with EBCOT", IEEE Transactions on Image Processing, vol. 9, No. 7, pp. 1158-1170, 1999.

[142] David Taubman and Michael W. Marcellin, JPEG2000, Image Compression Fundamentals, Standards and Practice, Norwell, MA, Kluwer Academic, 2002.

[143] T. Tran, R. deQueiroz and T. Nguyen, "Linear phase perfect reconstruction filter bank: Lattice structure, design and application in image coding", *IEEE Trans. Signal Processing*, vol.48, pp.133-147, 2000.

[144] A. H. Tewfik, D. Sinha and P. Jorgensen, "On optimal choice of wavelet for signal representation", *IEEE Trans. Info. Theory*, vol.38, pp.747-765, 1992.

[145] M. Unser and Thierry Blue, "Mathematical properties of the JPEG 2000 wavelet filters", *IEEE Trans. Image Processing*, vol.12, pp.1080-1090, 2003.

[146] Michael Unser, "Wavelet theory demystified", *IEEE Trans. Signal Processing*, vol.51, pp.470-483, 2003.

[147] P.P. Vaidyanathan., "Theory and design of M-channel maximally decimated quadrature mirror filters with arbitrary M, having perfect reconstruction property", *IEEE Trans. ASSP*, vol.35, pp.476-492, 1987.

[148] P.P. Vaidyanathan and P. Q. Haong, "Lattice structures for optimal design and robust implementation of two-band perfect reconstruction QMF banks", *IEEE Trans. ASSP*, vol.36, pp.81-94, 1988.

[149] P. P. Vaidyanthan and A. Kirac, "Results on optimal bi-orthogonal filter banks" *IEEE Trans. Circuit, Syst.II,* vol.45, pp.932-947, 1998.

[150] P.P. Vaidyanathan, T.Q. Nguyen, Z. Doganata and T. Saramaki, "Improved technique for design of perfect reconstruction FIR QMF banks with lossless polyphase matrices", *IEEE Trans ASSP,* vol.37, pp.1042-1055, 1989.

[151] P. P. Vaidyanathan, "Multirate digital filters, filter banks, polyphase networks, and applications: A tutorial", *Proceedings of the IEEE,* vol.78, pp.56-93, 1990.

[152] P. P. Vaidyanthan, *Multirate Systems and Filter Banks,* Englewood Cliffs, NJ: Prentice-Hall, 1992.

[153] M. Vetterli, "Filter banks allowing perfect reconstruction", *Signal Processing,* vol.10, pp.219-244, 1986.

[154] M. Vetterli, "Perfect transmultiplexer", *Proceedings of IEEE ICASSP*, pp.2567-2570, Tokyo, Japan, 1986.

[155] M. Vetterli, "A theory of multirate filterbanks", *IEEE Trans. ASSP*, vol.35, pp.356-372, 1987.

[156] M. Vetterli, "Running FIR and IIR filtering using multirate filterbanks", *IEEE Trans. Signal Processing,* vol.36, pp.730-738, 1988.

[157] M. Vetterli, D.Le Gall, "Perfect reconstruction FIR filter banks: some properties and Factorizations", *IEEE Trans. ASSP,* vol.37, pp.1057-1071, 1989.

[158] M. Vetterli and C. Harley, "Wavelets and filter banks: Theory and design", *IEEE Trans. ASSP*, vol.40, pp.2207-2232, 1992.

[159] M. Vetterli and J. Kovacevic, *Wavelets and Subband Coding*, Prentice Hall, Englewood Cliffs, NJ, 1995.

[160] J. Vilasenor, B. Belzer and J. Liao, "Wavelet filter evaluation for image compression", *IEEE Trans. Image Processing*, vol.2, pp.1053-1060, 1995.

[161] Gathen Jaochim VonZur, and Gerhard Jergen, *Modern Computer Algebra*, Cambridge University Press, 1999.

[162] S. Waser and F. Flynn, *Introduction to Arithmetic for Digital Systems Designers*, New York, Holtz, Reinhart and Winston, 1985.

[163] J.W. Woods and S.D. O'Neil, "Subband coding of images", *IEEE Trans. ASSP*, vol.34, pp.1278-1288, 1986.

[164] Patrick Wunsch and Andrew F. Laine, "Wavelet descriptors for multi-resolution recognition of handreinted characters", *Pattern Recognition,* vol.28, pp.1237-1249, 1995.

[165] Jui-Cheng Yen and Jiun-In Guo,“ A new chaotic key based design for image encryption and decryption”, *Proc. IEEE Int. Conf. Circuits and Systems*, vol. 4, pp.49-52, 2000.

[166] A. Zandi, J. D. Allen, E. L. Schwartz and M. Bolick, “CREW: Compression with reversible embedded wavelets”, in Proc. of IEEE Data Compression Conference, pp. 212-221, Snowbird, UT, USA, 1995.

[167] X. P. Zhang, M. Desai and Y. N. Peng, “Orthogonal complex filter banks and wavelets: some properties and design”, *IEEE Trans. Signal Processing* vol.47, pp.1039-1048, 1999.

[168] Zuras, D. "More on squaring and multiplying large integers." *IEEE Trans. Comput.*, vol.43, pp.899-908, 1994.

[169] http://www.ece.uvic.ca/~mdadams/jasper.

[170] http://www.kakadusoftware.com/.