Design and VLSI Implementation of Multirate Filter Banks

by

Esam Mostafa M. Abdel-Raheem
B.Sc. (Hon.), Ain Shams University, 1984
M.Sc., Ain Shams University, 1989

A Dissertation Submitted in Partial Fulfillment of the
Requirements for the Degree of
DOCTOR OF PHILOSOPHY
in the Department of Electrical and Computer Engineering

We accept this thesis as conforming
to the required standard

Dr. F. El-Guibaly, Supervisor (Dept. of Electrical and Computer Eng.)

Dr. A. Antoniou, Co-Supervisor (Dept. of Electrical and Computer Eng.)

Dr. K. F. Li, Departmental Member (Dept. of Electrical and Computer Eng.)

Dr. G. W. Vickers, Outside Member (Dept. of Mechanical Eng.)

Dr. T. Aboulnasr, External Examiner (Dept. of Electrical Eng., U. of Ottawa)

Examiners:

Dr. F. El-Guibaly, Supervisor (Dept. of Electrical and Computer Eng.)

Dr. A. Antonjou, Co-Supervisor (Dept. of Electrical and Computer Eng.)

Dr. K. F. Li, Departmental Member (Dept. of Electrical and Computer Eng.)

Dr. G. W. Vickers, Outside Member (Dept. of Mechanical Eng.)

Dr. T. Aboulnasr, External Examiner (Dept. of Electrical Eng., U. of Ottawa)

Supervisors: Dr. F. El-Guibaly and Dr. A. Antoniou

Abstract

New design approaches of two-channel perfect-reconstruction finite-duration impulse-response (FIR) quadrature mirror-image filter (QMF) banks with low delays are presented. The approaches depend on constraint optimization methods. The designs are considered superior relative to the existing approaches from the quality of reconstruction perspective. A computationally efficient iterative technique is applied to design cosine-modulated, M-channel pseudo-QMF banks. Moreover, a new weighted least-squares (WLS) method is applied to the design of two-channel linear-phase FIR QMF banks.

An algebraic mapping methodology to map decimator and interpolator algorithms onto very large scale integration (VLSI) array-processor structures is developed. Array-processor implementations for finite-duration impulse response as well as infinite-duration impulse response (IIR) filter banks are obtained based on the polyphase structures of decimators and interpolators. New and efficient array-processor implementations for FIR filter banks are also obtained based on direct-form structures of decimators and interpolators. All the implementations are considered modular and regular. Furthermore, some of the implementations have local communication features.

Improved and new-array processor implementations for FIR filters and linear-phase FIR filters are developed. New high-speed area-efficient fixed-point processors are developed to be incorporated in the resulting implementations.

Contents

List of Tables

List of Figures

List of Abbreviations

AAM	Adder-accumulator-multiplier
AFA	AND gate followed by a full adder
AHA	AND gate followed by a half adder
AM	Accumulator-multiplier
AMA	Adder-multiplier-accumulator
A^T	The transpose of matrix A
CLS	Constrained least squares
CPU	Central processing unit
CSA	Carry-save adder
DA	Distributed arithmetic
DFP	Davidson-Fletcher-Powell
DFT	Discrete Fourier transform
DG	Dependence graph
DSP	Digital signal processing
FA	Full adder
FDM	Frequency devision multiplexing
FIR	Finite-duration impulse response
MFLOPS	Millions of floating point operations
IDFT	Inverse discrete Fourier transform

IIR	Infinite-duration impulse response
LAG	Lagrange
LSB	Least significant byte
MO	Multiple operand
MSB	Most significant byte
MUX	Multiplexer
NFA	NAND gate followed by a full adder
NHA	NAND gate followed by a half adder
NOI	Number of iterations
PE	Processing element
PR	Perfect reconstruction
QMF	Quadrature mirror-image filter
SFG	Signal flow graph
Sg	Speedup gain
SNR_r	Signal-to-reconstruction noise ratio
TDM	Time devision multiplexing
VLSI	Very-large-scale integration
WLS	Weighted least squares

Acknowledgements

I would like to thank my supervisor, Professor F. El-Guibaly of the Department of Electrical and Computer Engineering, for his close supervision, continuous encouragement, patience and advice during the course of this research, and for his help in the preparation of this dissertation.

I also would like to thank my co-supervisor, Professor A. Antoniou of the Department of Electrical and Computer Engineering, for his supervision, advice, and valuable comments during the course of the research. Financial assistance provided by Micronet, Network of Centres of Excellence Program, and by NSERC, Natural Sciences and Engineering Research Council of Canada is also gratefully acknowledged.

I thank my family for their continuous support, patience, and encouragements throughout my studies.

In the name of Allah, Most
Gracious, Most Merciful

بِسْمِ اللَّهِ الرَّحْمَٰنِ الرَّحِيمِ

﴿ وَقُلِ اعْمَلُوا فَسَيَرَى اللَّهُ عَمَلَكُمْ
وَرَسُولُهُ وَالْمُؤْمِنُونَ ﴾ سورة التوبة : ١٠٥

And say: "Work (righteousness):
Soon will Allah observe your work,
And His Messenger, and the Believers".
(Qur'an, 9:105)

To my parents and my brother.
To Egypt, the land of my roots.
To United Arab Emirates, the land
of my best memories.
To Canada ... Thanks for everything.

إلى والدىّ و أخى
إلى مصر ... أرض أجدادى
إلى الإمارات ... أرض أحلى الذكريات
إلى كندا ... البلد الذى إ ستضافنى

Chapter 1

Introduction

The decomposition of a signal into contiguous frequency bands and reconstruction of the signal based on the subband components are fundamental concepts in digital signal processing (DSP) [1, 2]. Partitioning of the input signal into several frequency bands is done by the so called analysis filter bank and reconstruction is done by the synthesis filter bank. The subband components of the input signal are usually decimated to reduce the amount of computational load in applications where the subband components need to be processed.

1.1 Multirate Filter Banks

Multirate filter banks have been used extensively in many areas such as speech coding [2], image coding [3, 4], transmultiplexing [5]-[7], wavelet transform [8], frequency-domain speech scrambling [9], and short-time spectral analysis [10, 11].

1.1.1 Two-Channel Quadrature Mirror-Image Filter Bank

The two-channel quadrature mirror-image filter (QMF) bank, shown in Fig. 1.1(a), is one of the earliest and most commonly employed structures since it was introduced

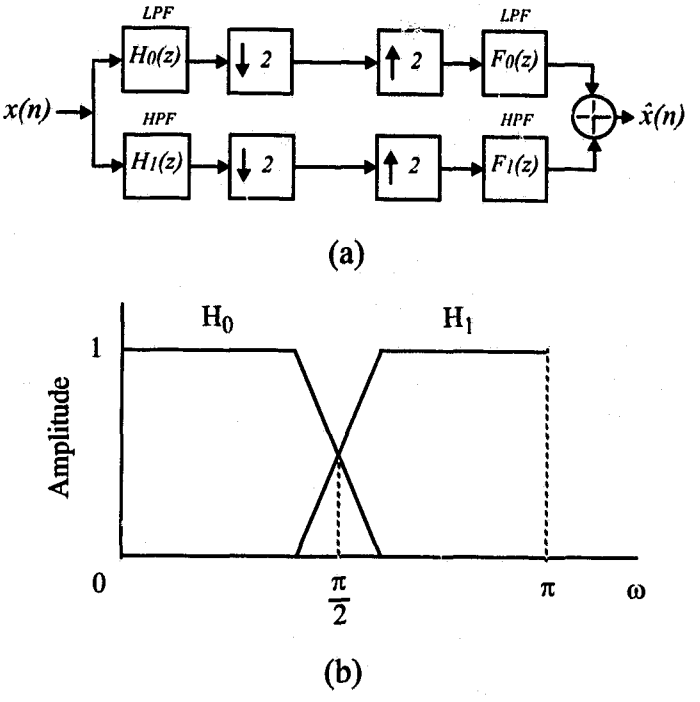


Figure 1.1: Two-channel QMF bank. (a) Analysis/synthesis banks. (b) Ideal amplitude response.

by Croisier, Esteban, and Galand in [1, 2]. In the analysis bank, the signal $x(n)$ is split into two subbands by passing through a lowpass filter H_0 and a highpass filter H_1 of transfer functions $H_0(z)$ and $H_1(z)$, respectively. Each subband signal is downsampled by a factor of two. In the synthesis bank, each decimated subband signal is upsampled by a factor of two. Then, the expanded signals are passed through the synthesis filters of transfer functions $F_0(z)$ and $F_1(z)$. The purpose of the synthesis filters is to eliminate the images. As a result, the reconstructed signal $\hat{x}(n)$ closely resembles the input signal $x(n)$. The name quadrature mirror filter derives from the fact that the response of filter H_1 is the mirror-image of the response of H_0, with respect to frequency $\pi/2$ which is a quarter of the sampling frequency

(assuming a normalized sampling period T=1 s) as indicated in Fig. 1.1(b).

Several approaches have been proposed for the design of two-channel QMF banks [1], [12]-[38]. The reconstructed signal in the two-channel QMF system of Fig. 1.1(a) is related to the input signal by

$$\hat{X}(z) = T(z)X(z) + A(z)X(-z) \tag{1.1}$$

where

$$T(z) = \tfrac{1}{2}[H_0(z)F_0(z) + H_1(z)F_1(z)] \tag{1.2}$$

and

$$A(z) = \tfrac{1}{2}[H_0(-z)F_0(z) + H_1(-z)F_1(z)] \tag{1.3}$$

are the channel and aliasing transfer functions, respectively.

According to the type of filter used, QMF banks are categorized into two classes: (a) finite-impulse-response (FIR) QMF banks, and (b) infinite-impulse-response (IIR) QMF banks. The advantages and disadvantages of one category over the other are related to the classical advantages and disadvantages of FIR and IIR filters [39]. QMF banks can introduce three types of distortion: (a) aliasing, (b) amplitude distortion, and (c) phase distortion. The first step of the design process is to cancel aliasing effects. Amplitude distortion and/or phase distortion can be minimized or eliminated according to the filter type. A perfect-reconstruction (PR) filter bank is one that is free from all errors and distortions, and the reconstructed signal is, therefore, just a delayed version of the input signal.

In general, FIR QMF banks are categorized into:

1. Classical FIR QMF banks

2. PR FIR QMF banks with nonlinear phase filters

3. PR FIR QMF banks with linear phase filters

QMF banks of the first category are filter banks in which aliasing is removed and the phase distortion is eliminated by using analysis/synthesis filters characterized by

$$F_0(z) = H_1(-z),\ F_1(z) = -H_0(-z)$$
$$H_1(z) = H_0(-z)$$

where $H_0(z)$ is the transfer function of a lowpass filter with even length N and a symmetrical impulse response. The linear time-variant system of Fig. 1.1(a) thus becomes a linear time-invariant system with a frequency response

$$T(e^{j\omega}) = \tfrac{1}{2}e^{-j\omega(N-1)}\{|\ H_0(e^{j\omega})\ |^2\ +\ |\ H_0(e^{j(\omega+\pi)})\ |^2\} \tag{1.4}$$

Filter H_0 is designed by minimizing an error measure of the form

$$E = \sum_{\omega=0}^{\pi/2}[|\ H_0(e^{j\omega})\ |^2 + |\ H_0(e^{j(\omega+\pi)})\ |^2 - 1]^2 + \alpha \sum_{\omega=\omega_s}^{\pi} |\ H_0(e^{j\omega})\ |^2 \tag{1.5}$$

The first term denotes the reconstruction error and the second term denotes the stopband error where α is a positive constant and ω_s is the stopband edge. The design was carried out in [12] using nonlinear optimization. QMF design in the time domain was introduced in [14] and QMF structures were reported in [40]. An efficient iterative technique to solve the minimization problem was introduced in [17] and minimax designs of FIR QMF banks were reported in [16]-[18].

An efficient implementation of the analysis/synthesis system requires polyphase structures. In either the analysis or synthesis bank, the polyphase filters of the lowpass filter H_0 are sufficient to implement the system [27].

QMF banks of the second category were reported in [19, 20]. These banks are called conjugate quadratic filters and the requirements imposed on the analysis/synthesis filters are given by

$$F_0(z) = H_1(-z),\ F_1(z) = -H_0(-z)$$
$$H_1(z) = -H_0(-z^{-1})\ z^{-(N-1)}$$

where $H_0(z)$ is the transfer function of a lowpass filter of even length N and nonlinear phase response. In this approach the linear time-invariant system transfer function is of the form

$$\begin{aligned} T(z) &= \tfrac{1}{2}\{H_0(z)H_0(z^{-1}) + H_0(-z)H_0(-z^{-1})\}\ z^{-(N-1)} \\ &= \tfrac{1}{2}\{G(z)\ +\ G(-z)\}\ z^{-(N-1)} \end{aligned} \tag{1.6}$$

where $G(z)$ is the transfer function of an odd-length $N_g = 2N - 1$, zero-phase half-band filter with a symmetrical impulse response and nonnegative frequency response. The design procedure depends on obtaining a spectral factor $H_0(z)$ of $G(z)$. A structure to implement the analysis filters taking advantage of the relationship between H_0 and H_1 is presented in [27]. However, all the arithmetic operations are performed at the high rate. To obtain more efficient implementation, two structures have to be employed to implement H_0 and H_1.

QMF banks of the third category, i.e. PR filter banks using linear-phase FIR analysis/synthesis filters, were introduced in [22]-[28]. Referring to (1.1) and ensuring that the relations

$$F_0(z) = 2H_1(-z), \quad F_1(z) = -2H_0(-z)$$

hold, we have

$$T(z) = H_0(z)H_1(-z)\ -\ H_0(-z)H_1(z) \tag{1.7}$$

If the pure-delay constraint

$$T(z) = z^{-2k+1}, \quad k = \frac{N_0 + N_1}{4} \tag{1.8}$$

is imposed, it is possible to obtain a PR system where the output is a delayed replica of the input. N_0 and N_1 are the lengths of H_0 and H_1, respectively.

By combining the constraint in (1.8) and the linear-phase condition, it has been shown in [23] that only two types of systems yield nontrivial analysis filters:

(a) Both filters have even length and opposite symmetry.

(b) Both filters have odd length and are symmetric.

In [23], PR is achieved by applying the lattice structure to the filters and forming an error criterion which can be minimized using optimization. In [24], the QMF problem is solved using the Lagrange multiplier and Lagrange-Newton approaches. In [25], the QMF problem is solved using a constrained least-squares (CLS) optimization approach. The analysis and synthesis filters of this design category have to be implemented separately. It is noted that if the analysis or synthesis filter is of linear phase with odd length, its polyphase components are of linear phase with odd and even lengths.

The strategy in designing IIR QMF banks is to eliminate aliasing and amplitude distortion and minimize phase distortion if it is severe depending on the application [28]. The set of filters is the same as in classical QMF banks with the constraint that $T(z)$ is an allpass transfer function. This can be done by constraining the transfer function of the lowpass analysis filter to be of the form

$$H_0(z) = \frac{a_0(z^2)}{2} + z^{-1}\frac{a_1(z^2)}{2} \tag{1.9}$$

where $a_0(z)$ and $a_1(z)$ are allpass transfer functions. Thus, $P_0(z) = a_0(z)/2$ and $P_1(z) = a_1(z)/2$ are the polyphase components. The procedure is to obtain an odd-order elliptic transfer function $H_0(z)$ of specified attenuation and transition width with passband ripple δ_p and stopband ripple δ_s such that $\delta_s^2 = 1 - (1 - 2\ \delta_s)^2$. This can be done using the procedure in [39]. This category of filter banks can be implemented efficiently using polyphase structures. A similar procedure can be performed using lattice wave digital filters [41] as these structures produce two outputs for the lowpass and the highpass signals. This can be done using the techniques in [35, 36].

The overall delay of a QMF bank is determined by the lengths of the filters used. Two-channel QMF banks are widely used for tree-structured subband speech coding systems, octave-band structures, and the wavelet transform. In these systems, delays of more than 1/4 second in full-duplex systems degrade subjective performance. Thus, the design of two-channel QMF banks with low delays is desired. A low reconstruction delay system is defined to be a system with filters of length N and with a reconstruction delay k which is smaller than $N-1$. Such designs are presented in [37, 38] but result in a relatively low signal-to-reconstruction noise ratio (SNR_r).

In this thesis, two approaches for the design of two-channel PR FIR QMF banks with low delays are proposed. The approaches are based on constrained optimization methods. Also, a minimax design of the two-channel linear-phase FIR QMF banks is developed using a weighted least-squares (WLS) technique.

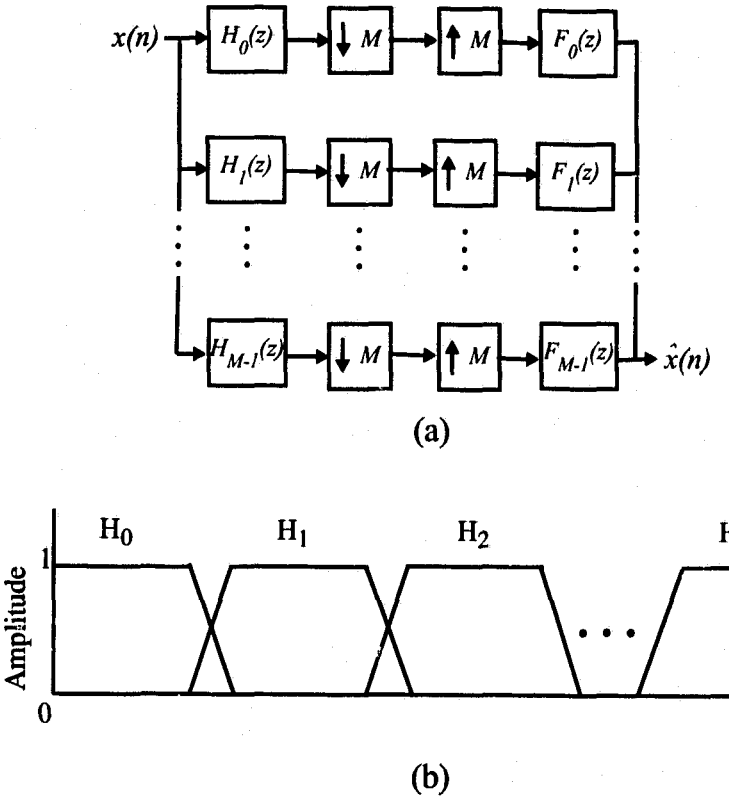


Figure 1.2: The M-channel filter bank. (a) Analysis/synthesis banks. (b) Ideal amplitude response.

1.1.2 M-Channel Filter Banks

A two-channel filter bank can be extended to an M-channel filter bank shown in Fig. 1.2(a), where $H_0(z)$, $H_1(z)$, ..., $H_{M-1}(z)$ are the transfer functions of analysis bank filters, and $F_0(z)$, $F_1(z)$, ..., $F_{M-1}(z)$ represent the synthesis filters. In the analysis section, the incoming signal $x(n)$ is split into M frequency bands by filtering, and each subband signal is maximally decimated, i.e., decimated by a factor of M. The M decimated signals are then processed in the synthesis bank by interpo-

lating each signal, filtering, and then adding the M filtered signals. In maximally decimated filter banks, each subband component is represented by the minimum number of samples per unit time. Such filter banks are of particular interest in frequency-domain coders. In speech coding applications, the primary objective is to reduce the bit rate while maintaining the perceived quality of the coding system. In frequency domain coders, the bit rate is directly related to the number of frequency-domain samples per unit time. Thus it is important to represent the output of each channel with the minimum number of samples which is achieved in maximally decimated filter banks. The benefit of utilizing maximally decimated channels is not strictly limited to coding applications. In other processing areas, frequency bands are often maximally decimated in order to reduce the complexity of the frequency-domain algorithms.

The reconstructed signal is expressed as

$$\hat{X}(z) = \frac{1}{M} \sum_{\ell=0}^{M-1} X(zW^{\ell}) \sum_{k=0}^{M-1} H_k(zW^{\ell}) F_k(z) \tag{1.10}$$

where $W = e^{-j2\pi/M}$. The above equation can be written as

$$\hat{X}(z) = X(z)T(z) + \sum_{\ell=1}^{M-1} X(zW^{\ell}) A_{\ell}(z) \tag{1.11}$$

where

$$T(z) = \frac{1}{M} \sum_{k=0}^{M-1} H_k(z) F_k(z) \tag{1.12}$$

and

$$A_{\ell}(z) = \frac{1}{M} \sum_{k=0}^{M-1} H_k(zW^{\ell}) F_k(z), \quad \ell \neq 0 \tag{1.13}$$

are the overall channel transfer function and the ℓth aliasing transfer function, respectively. The design of M-channel filter banks is considered in [42]-[50]. Cosine-modulated pseudo-QMF banks were introduced by Nussbaumer [42] and developed

by others [43]-[45], [48]-[50]. In these systems, the analysis and synthesis filters are chosen so that only *adjacent-channel aliasing* is cancelled, and the channel function is approximately a delay [27]. These filters are attractive from the perspective of design and implementation. From the design point of view, only a prototype filter needs to be designed. From the implementation point of view, the cost of the analysis or synthesis bank is equal to that of one filter plus the cost of a fast discrete-cosine transformer. The design of cosine-modulated pseudo-QMF banks can be performed using any unconstrained nonlinear optimization.

In this thesis, a computationally efficient design of cosine-modulated pseudo-QMF banks is achieved using an iterative technique.

1.1.3 Filter Bank Structures

QMF banks can be formed in different structures, i.e., uniform and nonuniform QMF banks. The M-channel filter bank in Fig. 1.2(a) is called a uniform QMF bank. An idealized amplitude response of the analysis filters is shown in Fig. 1.2(b). The resulting filter bank is regular and the outputs from each analysis channel have the same sample rate. Another way to build a uniform filter bank, with M a power of two is to use tree structures with two-channel QMF banks as shown in Fig. 1.3.

M-channel nonuniform QMF banks can be built either directly using M different analysis/synthesis filters with M different decimation/interpolation ratios as shown in Fig. 1.4(a), or by using the two-channel QMF banks in the so-called octave-band structure as shown in Fig. 1.5. Either way, the idealized amplitude response of the analysis filters is of the form shown in Fig. 1.4(b). Other nonuniform filter banks using fractional compression/expansion factors are reported in [51].

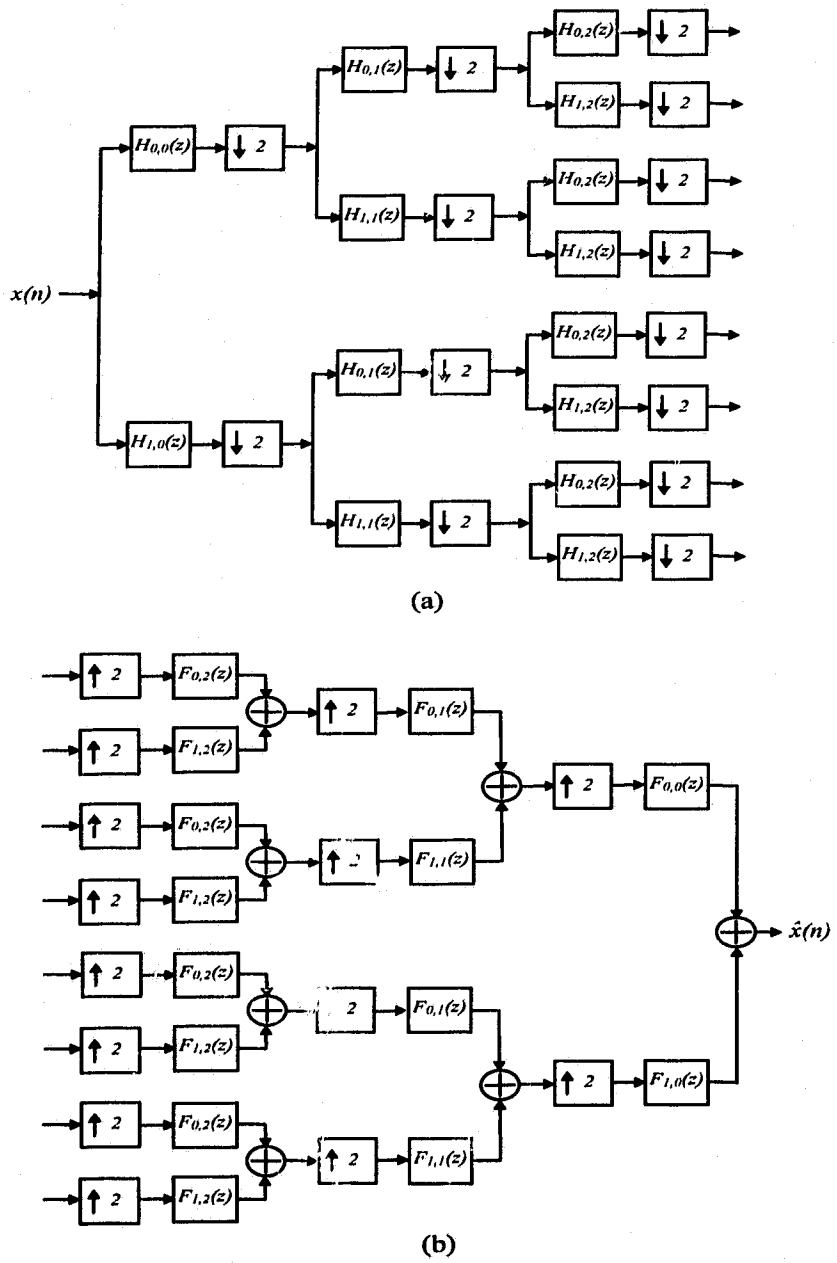


Figure 1.3: 8-channel tree-structured system. (a) Analysis bank. (b) Synthesis bank.

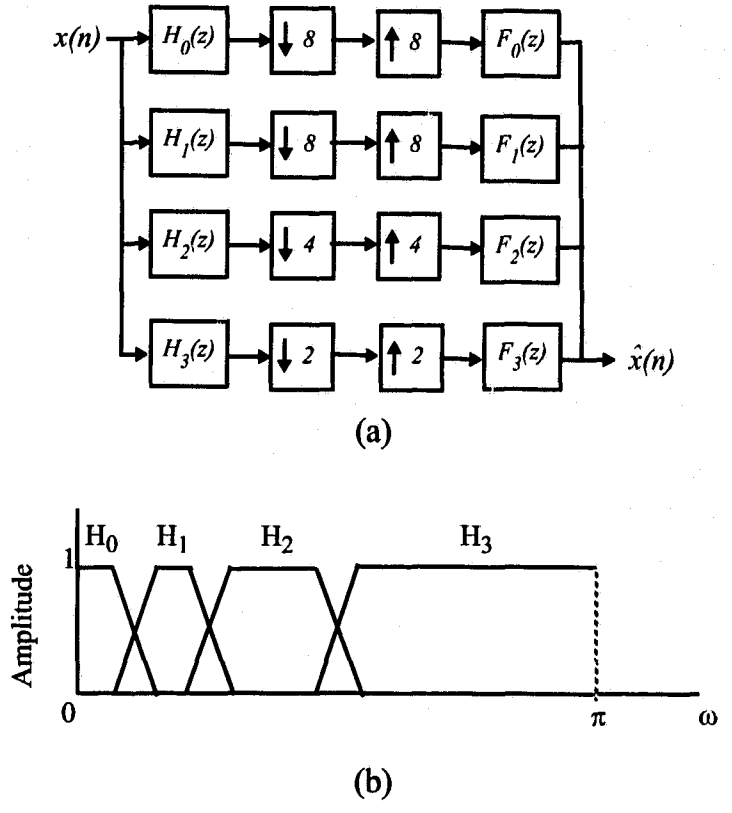


Figure 1.4: 4-channel nonuniform filter bank. (a) Analysis/synthesis banks. (b) Ideal amplitude response.

1.2 VLSI Array Processors

The practicality of algorithms for many real-time DSP applications is determined by the computational load. This critically depends on the amount of parallel processing, sampling rate and the volume of data. The availability of low-cost, high-density, high-speed VLSI devices, and the emergence of computer-aided design facilities, presages a major breakthrough in the design of massively parallel processors.

Real-time signal processing can be achieved by using special-purpose array pro-

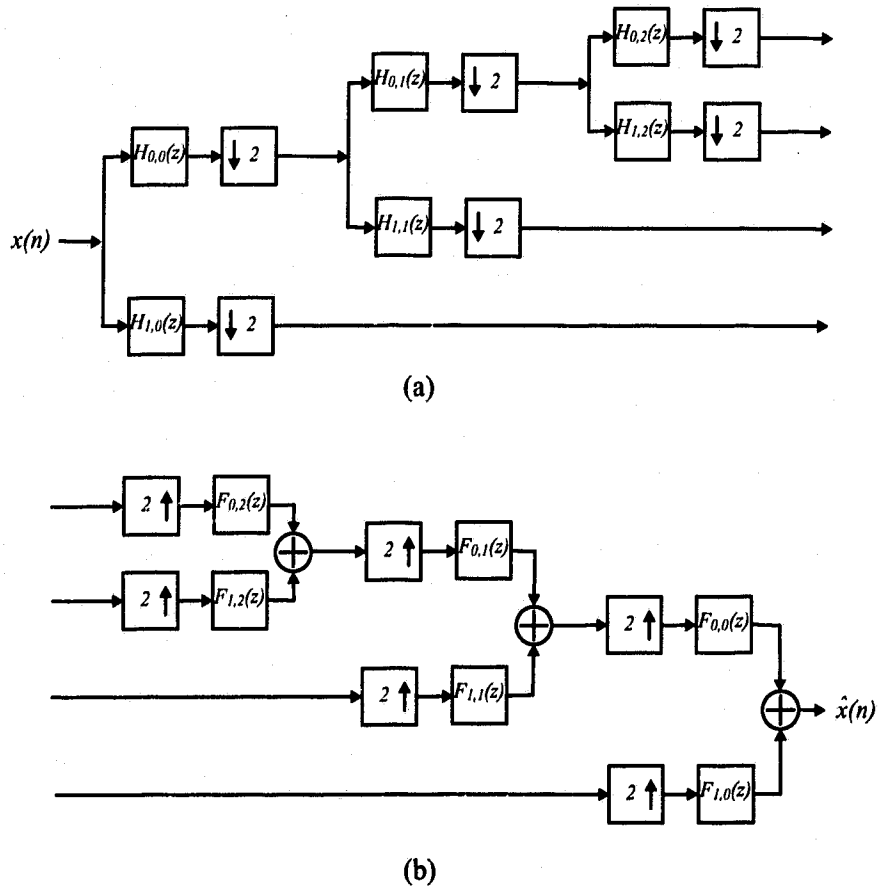


Figure 1.5: 4-channel octave-band system. (a) Analysis bank. (b) Synthesis bank.

cessors, and maximizing the processing concurrency by either pipeline processing or parallel processing or both [52]. A systolic system consists of a set of interconnected cells, each capable of performing some simple operation. Information in a systolic system flows between cells in a pipelined fashion, and communication with the outside world occurs only at the *boundary cells* [53]. A systolic array is very amenable to VLSI implementation by taking advantage of its regular and localized data flow [52]. It is especially suitable to a special class of compute-bound algorithms in which the total number of operations is larger than the total number of input and output elements [52]-[57]. A systolic array often represents a direct mapping of computations onto processor arrays. Consequently, the systolic array features the important properties of modularity, local interconnection, as well as a high degree of pipelining and highly synchronized multiprocessing.

1.2.1 Techniques for Mapping Algorithms onto Hardware

Several techniques for mapping algorithms onto processor arrays have been discussed in the literature [52], [58]-[61]. Kung [52] presented the signal flow graph (SFG) approach which is derived from the dependence graph (DG). The SFG can be mapped directly onto a systolic array by mapping nodes onto processing elements (PEs) and edges onto interconnections. Timing and data movements are derived from a linear timing function applied to the DG. Rao and Kailath [58] represent an iterative algorithm as a reduced dependence graph derived from a class of algorithms called regular iterative algorithms. They proved that a regular iterative algorithm can be mapped onto a processor array using a transformational approach. Moldovan [59] expresses an iterative algorithm as a set of dependence vectors. In this scheme,

processor assignment and system timing are obtained by transforming the dependence vectors using a transformation matrix. Quinton [60] proposed an algorithm mapping method based on expressing a problem as a set of uniform recurrence equations over a domain consisting of a set of index points. A valid timing function is determined subject to constraints set by the algorithm dependence vectors. A processor allocation function is chosen to project the points in the index set of the recurrences. Once the timing and allocation functions are known, the systolic array can be systematically generated. A similar method was proposed by Rajopadhye [61].

1.2.2 VLSI Implementation of Multirate Filter Banks

Decimators and interpolators are the most basic elements of multirate filter banks. The polyphase representation leads to computationally efficient implementations of decimator and interpolator structures in which all the multiplications and additions are performed at a low rate [62]-[63]. Rational sampling rate conversion was considered in [63]. In [27, 63, 64, 65] structures for fractional decimators and interpolators were considered. Systolic implementation of linear-phase FIR decimators and interpolators were reported in [66]. The implementation is complex and involves the use of programmable switches arranged in a hierarchical manner. This results in a large silicon area and complex control overhead. Moreover, the number of multipliers involved is large especially for high decimation and interpolation factors. Systolic implementations of IIR decimators and interpolators were reported in [67]. However, no systematic methodology was used to map the decimator and the interpolator equations onto systolic structures. As a consequence, the resulting systolic

structures involve a fractional delay for the signal processed at the low rate. From a hardware point of view, this requires special and elaborate techniques for control and signal timing. Moreover, other, and perhaps more efficient, structures could not be explored.

The study of multirate systems using the technique of Kung [52] is awkward since some edges in the DG correspond to high-rate signals while other edges correspond to low-rate signals. Simple application of a projection vector and different scheduling vectors would result in highly inefficient architectures. The techniques presented by Rao and Kailath [58] and Moldovan [59] are not amenable to multirate systems since only one scheduling vector is defined within the structure of the transformation matrix. The methods by Quinton [60] and Rajopadhye [61] are applicable to single-rate systems only. However, it is not obvious how these techniques can be adapted to multirate systems.

FIR and IIR filter bank implementations are reported in [68]-[70]. The implementation in [68] is not suitable for high-speed applications since it depends on *c-slow* circuits. Moreover, the implementations in [69, 70] are not suitable for FIR PR filter banks.

In this thesis, new efficient VLSI array processors for decimators, interpolators, and filter banks are obtained using an algebraic approach. The implementations are based on polyphase FIR/IIR decimator/interpolator structures and on direct-form FIR decimator/interpolator structures. Since polyphase filters are integral parts of the former implementation, array-processor implementations for digital filters are obtained. It should be mentioned that throughout the thesis, the term systolic array stands for a fully pipelined array processor. A partially pipelined array processor is being referred to by a semi-systolic array or by simply an array processor.

1.3 Design of Fixed-Point Processors for DSP Applications

In considering the design of any array processor system, it is important to consider the design of the PEs involved. Multiplier and adder delays are the dominant factors which determine the speed of the array structure. Minimizing these delays results in a high-speed array processor suited for high-speed DSP applications.

In this thesis, two new designs of fixed-point processors are presented. The designs are based on parallel multipliers for 2's-complement arithmetic [71]-[74]. A new inner-product processor in which both high-speed and double-precision operations are maintained is presented. The new processor enhances the speed of operation with a slight increase in the area. This processor can be incorporated in FIR filter, FIR decimator, and filter bank structures. A special processor for linear-phase FIR filter structures is presented. The processor performs an add-multiply-accumulate operation in the same time as a simple multiplier. The module enhances the speed of operation without incurring extra silicon area or introducing extra latency to the system.

1.4 Thesis Outline

This thesis is organized in three parts. The first part, comprising Chapters 2 and 3, deals with the design of multirate filter banks. The second part, comprising Chapters 4 and 5, deals with VLSI array-processor implementation of digital filters along with the new designs of fixed-point inner-product and adder-multiplier-accumulator processors. The last part, comprising Chapter 6 deals with the VLSI array-processor implementation of decimators, interpolators, and filter banks.

In Chapter 2, two constrained optimization approaches are applied for the design of two-channel PR FIR QMF banks with low delays. The first approach is based on the Lagrange-multiplier method and can be used to design banks with filters of unequal lengths. The approach is simple, efficient, and flexible and leads to a closed-form exact solution. The second approach can be used to design filters with equal as well as unequal lengths. In this approach, the design is formulated as a quadratic constrained least-squares minimization problem which can be solved using standard optimization approaches.

In Chapter 3, two design approaches for filter banks are presented. In the first approach, a computationally efficient approach is applied to the design of cosine-modulated pseudo-QMF banks. In the second approach, a modified WLS method is employed to obtain a weighted minimax design of linear-phase FIR QMF banks.

In Chapter 4, array processors for FIR filters and linear-phase FIR filters are developed using the SFG approach. Those implementations are considered integral parts of the polyphase structures of decimators, interpolators, and filter banks.

In Chapter 5, a new inner-product processor is presented. The new processor enhances the speed of operation of the FIR filter implementation but entails a slight increase in the area. A new processor module to perform an add-multiply-accumulate operation is also presented which enhances the speed of operation of the linear-phase FIR implementations without increasing the silicon area or introducing extra latency in the system.

In Chapter 6, an algebraic technique is applied to obtain new array-processor implementations for FIR/IIR decimators, interpolators, and filter banks. The implementations are based on FIR/IIR decimator and interpolator polyphase structures and on FIR direct-form structures. Fully pipelined systolic implementations

with modular and regular PEs for polyphase decimators/interpolators with integer/fraction compression/expansion factors are presented.

The conclusion of the thesis and suggestions for further research are given in Chapter 7.

Chapter 2

Design of Low-Delay Two-Channel FIR Filter Banks

2.1 Introduction

Two-channel QMF banks are widely used for tree-structured subband speech coding systems [20], octave-band structures, and the wavelet transform [8]. In these systems, delays of more than 1/4 second in a full-duplex system degrade subjective performance. Thus, the design of two-channel QMF banks with low delays is highly desirable. A low reconstruction delay system is defined to be a system with filters of length N and with a reconstruction delay k which is smaller than $N-1$. Such designs are presented in [37, 38] but result in a relatively low SNR_r for the reconstructed signals.

In this chapter, two approaches for the design of two-channel PR FIR QMF banks with low delays are proposed. In the first approach, a low-order filter is first designed and the objective function of the filter bank is formulated as a quadratic programming problem with linear constraints. Then the Lagrange-multiplier method [75] is used to design a higher-order filter. The method is simple, efficient, flexible, and

leads to a closed-form exact solution. The second approach can be used to design filters of equal as well as unequal lengths. In this approach, the filter bank design is formulated as a quadratic-constrained least-squares minimization problem which can be solved using standard minimization algorithms [25, 76].

2.2 Perfect-Reconstruction System

The reconstructed signal in the two-channel QMF system of Fig. 1.1(a) is related to the input signal by Eq. (1.1), i.e.,

$$\hat{X}(z) = T(z)X(z) + A(z)X(-z) \tag{2.1}$$

where

$$T(z) = \tfrac{1}{2}[H_0(z)F_0(z) + H_1(z)F_1(z)]$$

and

$$A(z) = \tfrac{1}{2}[H_0(-z)F_0(z) + H_1(-z)F_1(z)]$$

are the channel and aliasing transfer functions, respectively. The aliasing term is cancelled by choosing $F_0(z) = 2H_1(-z)$ and $F_1(z) = -2H_0(-z)$. If

$$T(z) = z^{-k} \tag{2.2}$$

where k is a positive integer, then

$$\hat{X}(z) = z^{-k}X(z)$$

and the output signal is a delayed replica of the input signal and, therefore, a PR system is obtained.

Let $H_0(z)$ and $H_1(z)$ be the transfer functions of a lowpass filter of length N_0 and a highpass filter of length N_1, respectively. The desired (or ideal) frequency responses of these filters can be expressed as

$$\tilde{H}_0(e^{j\omega T}) = \mid \tilde{H}_0(e^{j\omega T}) \mid e^{-j\omega k_0 T}, \quad \tilde{H}_1(e^{j\omega T}) = \mid \tilde{H}_1(e^{j\omega T}) \mid e^{-j\omega k_1 T} \tag{2.3}$$

where $k_0 < (N_0 - 1)/2$ and $k_1 < (N_1 - 1)/2$ are the desired passband group delays of the lowpass and the highpass filters, respectively [77, 78], and T is the sampling period. It is easy to verify that $k = k_0 + k_1$ is the desired total system delay which is assumed to be an odd integer. Let $h_0(nT)$ and $h_1(nT)$ be the impulse responses of the causal lowpass and highpass filters, respectively. Assuming a normalized sampling period $T = 1$ s, (2.2) can be expressed in the time domain as

$$\sum_{r=0}^{2i-1} (-1)^r \; h_0(2i - 1 - r) h_1(r) = \tfrac{1}{2}\delta(i - k'), \quad i = 1, 2, \ldots, R \tag{2.4}$$

where

$$k' = \frac{k_0 + k_1 + 1}{2}$$

$$R = \begin{cases} \frac{N_0+N_1}{2} - 1 & \text{if } N_0 + N_1 \text{ is even} \\ \\ \frac{N_0+N_1-1}{2} & \text{if } N_0 + N_1 \text{ is odd} \end{cases}$$

2.3 Lagrange-Multiplier Approach

Let us consider the design of filter banks using filters of unequal lengths. The design process starts with the design of a lowpass filter of transfer function $H_0(z)$, length N_0, and group delay k_0 s. The error function to be minimized is of the form

$$\Psi_0 = \alpha_0 \int_0^{\omega_{p0}} \mid \tilde{H}_0(e^{j\omega}) - H_0(e^{j\omega}) \mid^2 \; d\omega + \beta_0 \int_{\omega_{s0}}^{\pi} \mid H_0(e^{j\omega}) \mid^2 \; d\omega \tag{2.5}$$

where w_{p0} and w_{s0} are the passband and stopband edges, respectively, and α_0 and β_0 are weights. If we let

$$\mathbf{y}_0 = [\ h_0(0) \quad h_0(1) \quad \cdots \quad h_0(N_0-1)\ ]^T$$

the error measure can be formulated as

$$\Psi_0 = \tfrac{1}{2}\mathbf{y}_0^T\mathbf{Q}_0\mathbf{y}_0 + \mathbf{p}_0^T\mathbf{y}_0 + d_0 \tag{2.6}$$

where d_0 is a constant given by

$$d_0 = \alpha_0\omega_{p0}$$

and $\mathbf{Q}_0$ is a real, symmetric, and positive definite $N_0 \times N_0$ matrix with entries

$$\begin{aligned} Q_0(n,m) = {} & 2\alpha_0 \int_0^{w_{p0}} [\cos(n\omega)\cos(m\omega) + \sin(n\omega)\sin(m\omega)]\ d\omega + \\ & 2\beta_0 \int_{w_{s0}}^{\pi} [\cos(n\omega)\cos(m\omega) + \sin(n\omega)\sin(m\omega)]\ d\omega \end{aligned} \tag{2.7}$$

for $0 \le n,\ m \le N_0 - 1$, and $\mathbf{p}_0$ is a column vector with entries

$$p_0(n) = -2\alpha_0 \int_0^{w_{p0}} [\cos(k_0\omega)\cos(n\omega) + \sin(k_0\omega)\sin(n\omega)]\ d\omega \tag{2.8}$$

for $0 \le n \le N_0 - 1$. The design can be performed using the approach in [77].

The next step is to form an error measure for the design of a highpass filter of transfer function $H_1(z)$, length N_1, and group delay k_1 s. Defining

$$\mathbf{y}_1 = [\ h_1(0) \quad h_1(1) \quad \cdots \quad h_1(N_1-1)\ ]^T$$

the error measure can be put, as above, in the form

$$\Psi_1 = \tfrac{1}{2}\mathbf{y}_1^T\mathbf{Q}_1\mathbf{y}_1 + \mathbf{p}_1^T\mathbf{y}_1 + d_1 \tag{2.9}$$

where d_1 is a constant given by

$$d_1 = \alpha_1(\pi - w_{p1})$$

$\mathbf{Q}_1$ is a real, symmetric, and positive definite $N_1 \times N_1$ matrix with entries

$$Q_1(n,m) = 2\alpha_1 \int_{w_{p1}}^{\pi} [\cos(n\omega)\cos(m\omega) + \sin(n\omega)\sin(m\omega)]\, d\omega + 2\beta_1 \int_{0}^{w_{s1}} [\cos(n\omega)\cos(m\omega) + \sin(n\omega)\sin(m\omega)]\, d\omega \qquad (2.10)$$

for $0 \leq n,\, m \leq N_1 - 1$, where w_{s1} and w_{p1} are the stopband and passband edges, respectively, and α_1 and β_1 are weights. In this case, $\mathbf{p}_1$ is a column vector with entries

$$p_1(n) = -2\alpha_1 \int_{w_{p1}}^{\pi} [\cos(k_1\omega)\cos(n\omega) + \sin(k_1\omega)\sin(n\omega)]\, d\omega \qquad (2.11)$$

for $0 \leq n \leq N_1 - 1$.

Equation (2.4) can be expressed in matrix form as

$$\mathbf{C}\mathbf{y}_1 = \mathbf{m} \qquad (2.12)$$

where $\mathbf{C}$ is an $R \times N_1$ matrix, which can be obtained by inspection from (2.4), and

$$\mathbf{m} = [\, m_1 \quad m_2 \quad \cdots \quad m_{k'} \quad \cdots \quad m_R \,]^T$$

with

$$m_i = \begin{cases} \frac{1}{2} & i = k' \\ 0 & i \neq k' \end{cases}$$

Matrix $\mathbf{C}$ can assume four distinct forms depending on whether each of the filter lengths N_0 and N_1 is odd or even.

The optimization problem for designing filter H_1 becomes

$$\text{minimize } \Psi_1 \text{ subject to } \mathbf{C}\mathbf{y}_1 = \mathbf{m} \qquad (2.13)$$

which can be solved by forming the Lagrangian function

$$L(\mathbf{y}_1, \lambda) = \tfrac{1}{2}\mathbf{y}_1^T\mathbf{Q}_1\mathbf{y}_1 + \mathbf{p}_1^T\mathbf{y}_1 + d_1 - \boldsymbol{\lambda}^T(\mathbf{C}\mathbf{y}_1 - \mathbf{m}) \qquad (2.14)$$

where

$$\boldsymbol{\lambda} = [\ \lambda_1 \quad \lambda_2 \quad \cdots \quad \lambda_R\ ]^T$$

is the Lagrange-multiplier column vector. The necessary and sufficient conditions for the solution of the problem in (2.13) form the set of linear equations

$$\begin{bmatrix} -\mathbf{Q}_1 & \mathbf{C}^T \\ \mathbf{C} & \mathbf{0} \end{bmatrix} \begin{bmatrix} \mathbf{y}_1 \\ \boldsymbol{\lambda} \end{bmatrix} = \begin{bmatrix} \mathbf{p}_1 \\ \mathbf{m} \end{bmatrix} \tag{2.15}$$

Hence, a closed-form solution can be readily obtained [79].

2.4 Constrained Least-Squares Approach

In this section we deal with the design of filter banks using filters of equal as well as unequal lengths. If

$$\mathbf{h} = [\ h_0(0) \quad \cdots \quad h_0(N_0-1) \quad h_1(0) \quad \cdots \quad h_1(N_1-1)\ ]^T$$

then the error measure for the design of filters H_0 and H_1 can be put in the form

$$\Psi = \tfrac{1}{2}\mathbf{h}^T\mathbf{Q}\mathbf{h} + \mathbf{p}^T\mathbf{h} + d \tag{2.16}$$

where d is a constant given by

$$d = d_0 + d_1$$

and $\mathbf{Q}$ is an $(N_0 + N_1) \times (N_0 + N_1)$ matrix of the form

$$\mathbf{Q} = \begin{bmatrix} \mathbf{Q}_0 & \mathbf{0} \\ \mathbf{0} & \mathbf{Q}_1 \end{bmatrix} \tag{2.17}$$

where $\mathbf{Q}_0$ and $\mathbf{Q}_1$ are the matrices formulated in (2.7) and (2.10), respectively, and $\mathbf{p}$ is a column vector given by

$$\mathbf{p} = [\ \mathbf{p}_0^T \quad \mathbf{p}_1^T\ ]^T \tag{2.18}$$

where $\mathbf{p}_0$ and $\mathbf{p}_1$ are the vectors formulated in (2.8) and (2.11), respectively.

The constraints in (2.4) can be expressed as

$$\begin{aligned} \mathbf{h}^T\mathbf{D}_i\mathbf{h} &= 0, \qquad i = 1, 2, \ldots, R,\ i \neq k' \\ \mathbf{h}^T\mathbf{D}_{k'}\mathbf{h} - 0.5 &= 0 \end{aligned} \tag{2.19}$$

where $\mathbf{D}_i$ is an $(N_0 + N_1) \times (N_0 + N_1)$ matrix of the form

$$\mathbf{D}_i = \begin{bmatrix} \mathbf{0} & \mathbf{C}_i \\ \mathbf{0} & \mathbf{0} \end{bmatrix} \tag{2.20}$$

with the entries of the $N_0 \times N_1$ matrix $\mathbf{C}_i$ being

$$C_i(n,m) = \begin{cases} (-1)^{n+1} & \text{if } n + m = 2i - 1 \\ 0 & \text{otherwise} \end{cases} \tag{2.21}$$

for $0 \leq n \leq N_0 - 1$, $0 \leq m \leq N_1 - 1$, and $i = 1, 2, \ldots, R$. The optimization problem becomes

$$\text{minimize } \Psi \text{ subject to (2.19)} \tag{2.22}$$

The PR QMF bank design procedure can be summarized as follows [25]:

1. Given N_0, N_1, k_0, k_1 and the passband and stopband edges in $H_0(z)$ and $H_1(z)$, compute $\mathbf{Q}$ and $\mathbf{p}$.

2. Compute $\mathbf{D}_i$, $i = 1, 2, \ldots, R$, and form the set of constraints in (2.19).

3. Design lowpass and highpass filters with the same specifications using the approach in [77]. Use their coefficients as initial values in the minimization problem.

4. Use a standard nonlinear optimization subroutine to solve the minimization problem in (2.22). The IMSL subroutine *DNCONF* [76] was found to give good results.

2.5 Design Examples and Comparisons

The Lagrange-multiplier approach can, in theory, be used to design equal-length filters since there is one degree of freedom for the design problem (i.e., the number of design parameters N_1 exceeds the number of constraints R by 1). Unfortunately, such designs turn out to be quite unsatisfactory in practice. However, by using unequal filter lengths such that $N_1 > N_0 + 2$, good designs can be achieved. The Lagrange-multiplier approach is essentially a suboptimal method since the lowpass filter may not be optimal. However, it has been found that a highpass filter with good frequency response is obtained by choosing a narrow transition width for the lowpass filter [24, 80, 81]. The design using the least-squares approach is optimal and more flexible since the two filters are designed simultaneously [82].

To check the reconstruction performance of the designed filter banks, the SNR_r in dB, which is defined as

$$\begin{aligned} \text{SNR}_r &= 10 \log \left(\frac{\text{signal energy}}{\text{reconstruction noise energy}} \right) \\ &= 10 \log \left\{ \frac{\sum_n x^2(n)}{\sum_n [x(n) - \hat{x}(n+k)]^2} \right\} \end{aligned} \tag{2.23}$$

has been computed for the case of a 100-sample ramp input signal. In the following, two examples are given in detail for the design of QMF banks with a delay of 7 s to illustrate the design approaches. Other examples are given for the design of low-delay QMF banks with higher-order filters. Finally, comparisons are carried out between the proposed approaches with other methods reported in the literature.

Example 2.1: Design LAG1220-7

A low-delay QMF bank with $N_0 = 12$, $N_1 = 20$, $\alpha_j = 2$, $\beta_j = 1$, $j = 0, 1$, $k_0 = 3$ s, and $k_1 = 4$ s was designed using the Lagrange-multiplier approach. The 12-tap lowpass filter was designed using the approach reported in [77], assuming bandedge frequencies $\omega_{p0} = 0.47\pi$, $\omega_{s0} = 0.62\pi$ and $k_0 = 3$ s. Then matrix $\mathbf{C}$ was formed. Matrix $\mathbf{Q}_1$ and vector $\mathbf{p}_1$ were then calculated assuming bandedge frequencies $\omega_{p1} = 0.63\pi$ and $\omega_{s1} = 0.37\pi$, and $k_1 = 4$ s. The coefficients of $H_1(z)$ were, in turn, obtained by solving (2.15). The coefficients of $H_0(z)$ and $H_1(z)$ are listed in Table 2.1 while the amplitude responses and the delay characteristics of the analysis filters are shown in Figs. 2.1 and 2.2, respectively. The SNR_r for a ramp input was found to be 280.34 dB, using double-precision floating-point accuracy. This value is in the range of the signal-to-reconstruction noise ratios for the QMF banks designed in [24]. The high SNR_r, together with the low ripple in the amplitude response and the low error in the delay characteristic of the channel bank in Fig. 2.3(a) and (b) demonstrate the PR quality of the design. The system delay is 7 s as opposed to 15 s for the linear-phase case.

Table 2.1: Coefficients of the analysis filters of Example 2.1.

n	$h_0(n)$	$h_1(n)$
0	−0.05352392759088	0.03252179821733
1	−0.03604898479695	0.02190380755440
2	0.27745906824176	−0.03212977889457
3	0.54786731838976	−0.24098504790310
4	0.34395622842965	0.53839195243112
5	−0.05434275032940	−0.38395673990167
6	−0.11351898427312	−0.04193841971583
7	0.05351223965582	0.15184586425018
8	0.05330699945173	0.03987338767202
9	−0.04568786417107	−0.08467846321947
10	−0.02108658469195	−0.03341761649297
11	0.03326214006498	0.04339265091636
12		0.02551827338073
13		−0.00753605052017
14		−0.01080075937535
15		0.00922530593043
16		0.00368756412226
17		−0.00433805350299
18		−0.00081204036347
19		0.00128091868374

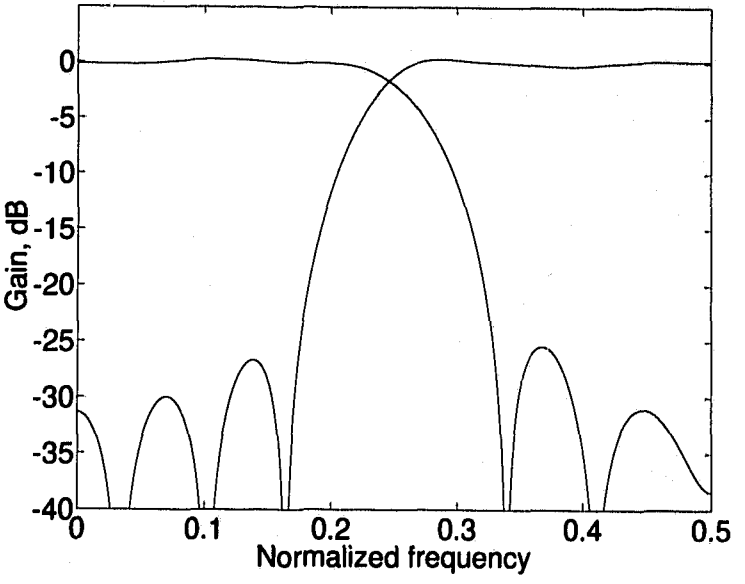
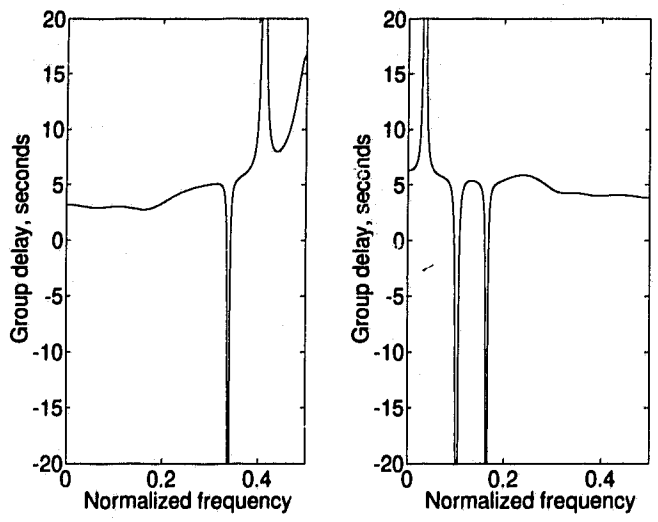


Figure 2.1: Amplitude responses of the analysis filters of Example 2.1.

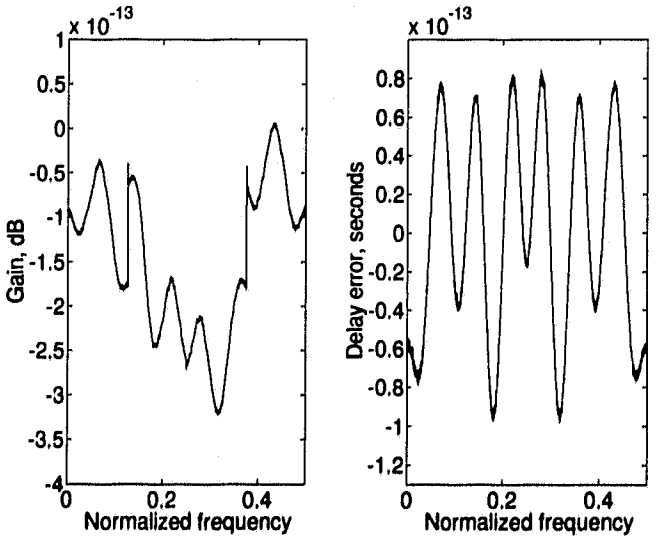
Example 2.2: Design CLS22-7

A low-delay QMF bank with $N_0 = N_1 = 22$, $\alpha_j = 2$, $\beta_j = 1$ for $j = 0, 1$, $k_0 = 3$ s, and $k_1 = 4$ s was designed using the constrained least-squares approach. The bandedge frequencies were chosen to be $\omega_{p0} = \omega_{s1} = 0.35\pi$, $\omega_{s0} = \omega_{p1} = 0.65\pi$. The algorithm converged in 41 iterations. Extra constraints in the transition bands have been imposed to control undesirable artifacts in these regions. This can be done by adding extra components to the error measure in (2.16). The coefficients of the analysis filters are listed in Table 2.2 and their amplitude responses and delay characteristics are shown in Figs. 2.4 and 2.5. The SNR_r was found to be 181.30 dB which is in the range of the signal-to-reconstruction noise ratios for the QMF banks designed in [25]. The high SNR_r, together with the low ripple in the amplitude response and the low error in the delay characteristic of the channel bank



(a) (b)

Figure 2.2: Group-delay characteristics of the analysis filters of Example 2.1. (a) Group delay of H_0. (b) Group delay of II_1.



(a) (b)

Figure 2.3: Performance of the QMF bank of Example 2.1. (a) Channel amplitude response. (b) Channel delay error.

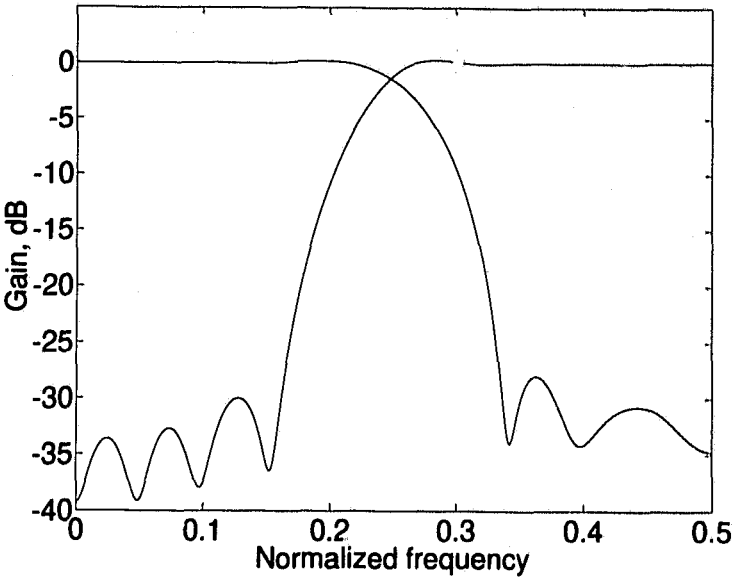


Figure 2.4: Amplitude responses of the analysis filters of Example 2.2.

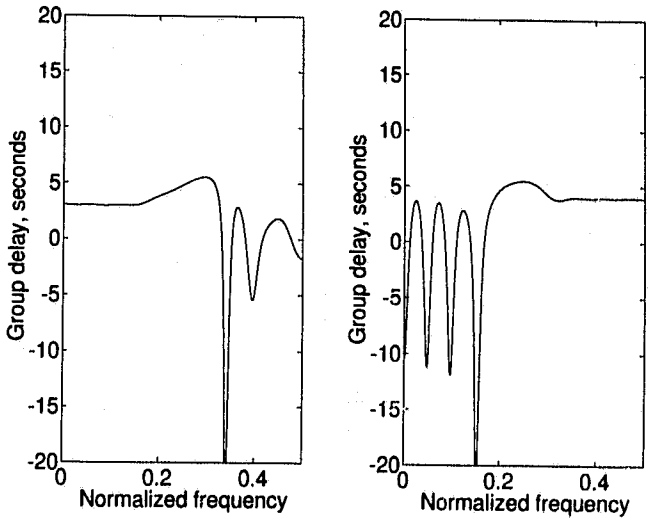
in Fig. 2.6(a) and (b) demonstrate the PR quality of the design. The system delay is 7 s as opposed to 21 s for the linear-phase case.

Example 2.3: Design LAG2836-15

A QMF bank with $N_0 = 28$, $N_1 = 36$, $\alpha_j = 2$, $\beta_j = 1$, $j = 0, 1$, $k_0 = 6$ s, and $k_1 = 9$ s (overall delay of 15 s) was designed using the Lagrange-multiplier approach. The bandedge frequencies were chosen to be $\omega_{p0} = 0.48\pi$, $\omega_{s0} = 0.6\pi$, $\omega_{p1} = 0.6\pi$, and $\omega_{s1} = 0.4\pi$. The coefficients of the analysis filters are listed in Table 2.3 and their amplitude responses are shown in Fig. 2.7. The SNR_r was computed as 277.84 dB.

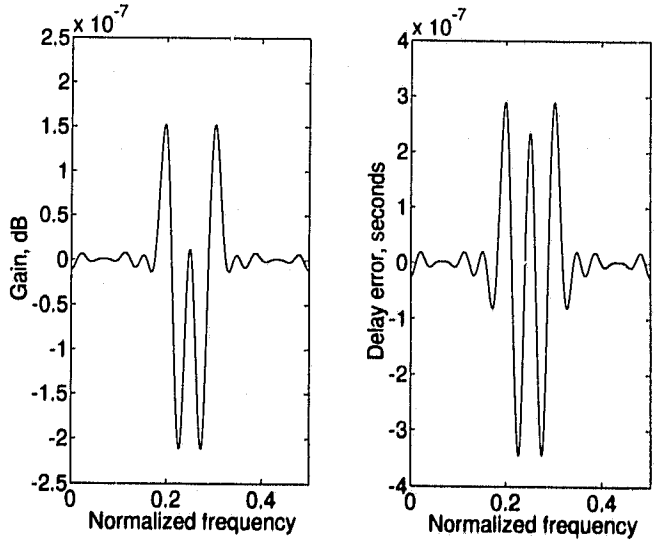
Table 2.2: Coefficients of the analysis filters of Example 2.2.

n	$h_0(n)$	$h_1(n)$
0	−0.01517670761659	0.02583764155501
1	−0.02748086900231	0.04678490494258
2	0.23331201429783	−0.04245202126045
3	0.53564016361092	−0.26954571706871
4	0.39274475667857	0.55756057678187
5	−0.04427374946690	−0.34978218807769
6	−0.15595780827251	−0.06391248196065
7	0.04534948530154	0.11682583251928
8	0.08191041475485	0.05932570374243
9	−0.03618701819961	−0.05524538065712
10	−0.03687832005056	−0.04560791903447
11	0.02612075654280	0.02531628225385
12	0.00572978973131	0.02702388663295
13	−0.01126674651201	−0.01182736639632
14	0.00572796985631	−0.01087434500804
15	0.00132677615654	0.00781912830834
16	−0.00481217097949	0.00263665222623
17	0.00091690112817	−0.00547941307446
18	0.00133725052965	−0.00104796512125
19	0.00006241929893	0.00019562160864
20	0.00064774085623	−0.00235198635705
21	0.00007183375029	−0.00026082721088



(a) (b)

Figure 2.5: Group-delay characteristics of the analysis filters of Example 2.2. (a) Group delay of H_0. (b) Group delay of H_1.



(a) (b)

Figure 2.6: Performance of the QMF bank of Example 2.2. (a) Channel amplitude response. (b) Channel delay error.

Table 2.3: Coefficients of the analysis filters of Example 2.3.

n	$h_0(n)$	$h_1(n)$
0	−0.01726989469213	0.00085560929309
1	0.02030905341524	−0.00100617954806
2	0.02762931377654	−0.00020376848923
3	−0.06422984777432	0.00181205212249
4	−0.03904502679218	0.00667079700065
5	0.28191208849025	−0.01739559711696
6	0.54952786543939	−0.05938054880821
7	0.34532531600061	0.01691408867766
8	−0.05738794970506	0.27656142625214
9	−0.12071784135312	−0.51752354040609
10	0.06131513339612	0.35195442449110
11	0.06435452029168	0.01668771384230
12	−0.06064533664926	−0.12952462873520
13	−0.03252893116069	−0.01416878858099
14	0.05550431596366	0.07536597668317
15	0.01082015206346	0.01001904927630
16	−0.04677586491392	−0.04292503313019
17	0.00384527983516	0.00296605398460
18	0.03590011258880	0.03140259365474
19	−0.01248815304573	−0.00657954267343
20	−0.02455362637762	−0.02246435919617
21	0.01597510472080	0.00762833153308
22	0.01429231861813	0.01554257450307
23	−0.01546358066289	−0.00740094760838
24	−0.00624212020699	−0.01043401539473
25	0.01235531815551	0.00614276830158
26	0.00088736966787	0.00672832986381
27	−0.00788723984040	−0.00480585318996
28		−0.00466904166662
29		0.00265664994518
30		0.00154541504617
31		−0.00555203312016
32		−0.00030474459288
33		0.00121865033943
34		0.00003065847825
35		−0.00027250285860

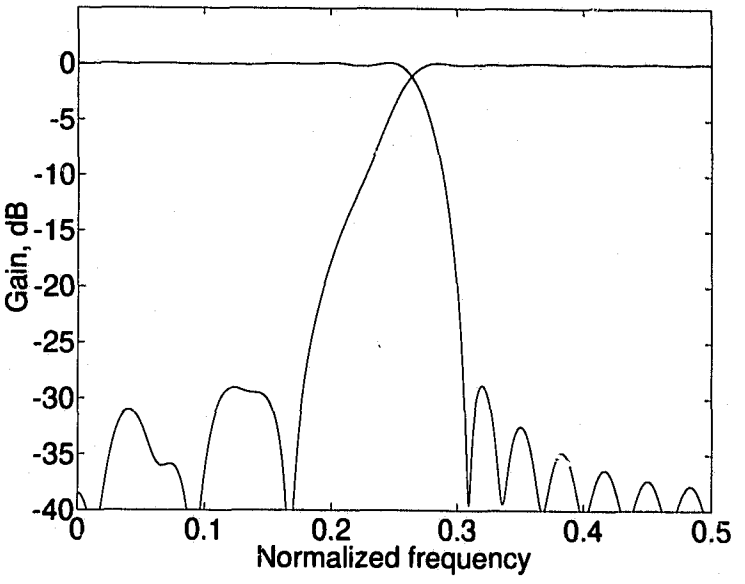


Figure 2.7: Amplitude responses of the analysis filters of Example 2.3.

Example 2.4: Design CLS32-15a

A QMF bank with $N_0 = N_1 = 32$, $\alpha_j = 2$, $\beta_j = 1$ for $j = 0$, 1, $k_0 = 7$ s, and $k_1 = 8$ s (overall delay of 15 s) was designed using the constrained least-squares approach using bandedge frequencies $\omega_{p0} = \omega_{s1} = 0.4\pi$, $\omega_{s0} = \omega_{p1} = 0.6\pi$. The algorithm converged in 50 iterations. The coefficients of the analysis filters are listed in Table 2.4 and their amplitude responses are shown in Fig. 2.8. The SNR_r was computed as 187.21 dB. The design process was used to obtain a QMF bank with the above specifications but with bandedge frequencies $\omega_{p0} = 0.38\pi$, $\omega_{s0} = 0.65\pi$, $\omega_{s1} = 0.35\pi$, and $\omega_{p1} = 0.62\pi$ (design CLS32-15b). The amplitude responses of the analysis filters are also shown in Fig. 2.8. It is noted that stopband attenuations of the individual filters are improved by increasing their transition widths.

Our experimental results have shown that reducing k_0 and increasing k_1 in the

Table 2.4: Coefficients of the analysis filters of Example 2.4 (design CLS32-15a).

n	$h_0(n)$	$h_1(n)$
0	−0.00186147724822	0.00331307418090
1	0.00020684139604	− 0.00036813819179
2	0.00986111153625	− 0.02222481820371
3	0.01297942679285	− 0.02258154784320
4	−0.05688475379803	0.02923497456067
5	−0.01400372820372	0.06826622828711
6	0.27094897935811	−0.03912319819452
7	0.51615456435928	−0.28722857826639
8	0.36039385101885	0.54723040028287
9	−0.01755177185168	−0.34017387364702
10	−0.13979778898743	−0.05239431701054
11	0.01717477470304	0.11699937003832
12	0.08780294116628	0.05353894933130
13	−0.01637517926896	−0.06319864063315
14	−0.05743771689227	−0.05064449114196
15	0.01193773701736	0.03502574897566
16	0.03779543629227	0.04446370017123
17	−0.00884732304749	−0.01754953882107
18	−0.02252963655778	−0.03586472511658
19	0.00522882924781	0.00618024152967
20	0.01203648907217	0.02573488655402
21	−0.00140697852462	0.00058556877601
22	−0.00657839472702	−0.01532676791456
23	−0.00057902957558	−0.00269163605953
24	0.00370611386302	0.00702111845438
25	0.00044009204812	0.00113943552557
26	−0.00139035395285	−0.00275847311667
27	0.00034728445142	0.00067032434250
28	−0.00023289552532	0.00112455539579
29	−0.00074479624060	−0.00041686769015
30	0.00035614588157	0.00120200623557
31	0.00039086658688	0.00131919047189

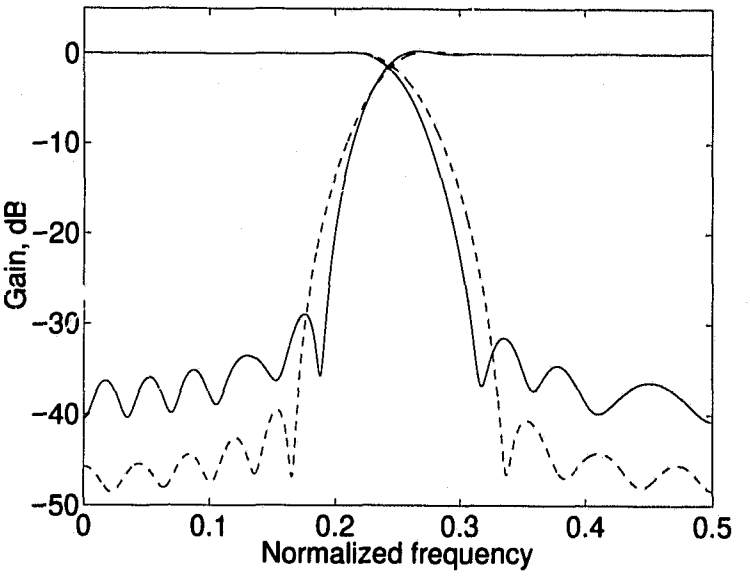


Figure 2.8: Amplitude responses of the analysis filters of Example 2.4. The responses indicated by the solid lines are those for the design CLS32-15a and the responses indicated by the dashed lines are those for design CLS32-15b.

Lagrange-multiplier approach would result in filters that are free from artifacts in the transition bands. The difference between k_0 and k_1 depends on the filter lengths. On the other hand, choosing k_0 and k_1 to be close to each other in the constrained least-squares approach (for the case where the filter lengths are equal) results in better convergence in the algorithm as well as better control over artifacts in the transition bands.

The SNR_r, minimum stopband attenuation in dB (A_{si}), maximum passband ripple amplitude (δ_{pi}), and transition width (Δ_i) of several two-channel designs are given in Table 2.5. The transition width Δ_i, $i = 0$, 1, is measured as the difference between the actual passband edge $\hat{\omega}_{pi}$ and the actual stopband edge $\hat{\omega}_{si}$, which in a lowpass filter are defined as the highest and lowest frequency at which the ripple amplitude and the attenuation are equal to the maximum passband ripple ampli-

Table 2.5: Results for two-channel designs.

Design	N_0	N_1	k (s)	A_{s0} (dB)	A_{s1} (dB)	δ_{p0}	δ_{p1}	SNR_r (dB)	Δ_0	Δ_1
LAG0812-5	8	12	5	28.80	27.14	0.0144	0.0145	275.40	0.4240π	0.4656π
LAG1220-7	12	20	7	25.40	26.67	0.0370	0.0350	280.34	0.2180π	0.2402π
LAG1826-9	18	26	9	26.50	23.24	0.0207	0.0205	269.53	0.2520π	0.2728π
LAG2024-9	20	24	9	22.25	24.20	0.0400	0.0387	292.81	0.2020π	0.2610π
LAG2028-11	20	28	11	24.92	26.10	0.0102	0.0095	285.63	0.2468π	0.2758π
LAG2836-15	28	36	15	28.80	29.03	0.0076	0.0074	277.84	0.2268π	0.2617π
CLS08-3	8	8	3	25.85	19.95	0.0143	0.0286	204.13	0.3930π	0.3641π
CLS08-5	8	8	5	25.75	23.50	0.0187	0.0185	147.44	0.3410π	0.3840π
CLS16-7	16	16	7	24.60	25.79	0.0072	0.0074	169.61	0.2353π	0.2989π
CLS22-7	22	22	7	28.00	30.00	0.0047	0.0050	181.30	0.3398π	0.3448π
CLS2024-9	20	24	9	27.50	23.35	0.0155	0.0147	243.81	0.2320π	0.2240π
CLS32-15a	32	32	15	31.52	28.91	0.0038	0.0032	187.21	0.228π	0.2198π
CLS32-15b	32	32	15	40.71	39.50	0.0016	0.0015	195.62	0.2951π	0.3168π
CLS32-15b/8	32	32	15	40.23	33.90	0.0135	0.0124	64.16	0.2625π	0.2506π
CLS32-11	32	32	11	30.84	30.66	0.0046	0.0046	190.66	0.3290π	0.2912π
Design in [37]	32	32	15	36.28	36.97	0.0016	0.0021	83.00	0.2968π	0.3046π

tude and minimum stopband attenuation, respectively. Designs obtained with the Lagrange-multiplier approach and the constrained least-squares method are identified with the prefixes LAG and CLS, respectively. From Table 2.5, it is clear that the Lagrange-multiplier approach yields higher signal-to-reconstruction noise ratios than the constrained least-squares method since the constraints in the former are exactly satisfied. For the same filter lengths and system delay, the second approach yields higher stopband attenuation and lower passband ripple. For low-delay QMF banks with system delays up to one-third that of the linear-phase case, the second approach is preferred since in the first approach it is difficult to control undesirable artifacts in the transition bands. For moderately low-delay QMF banks with filters of unequal lengths, with system delays equal to half that of the linear-phase case, the first approach provides an easy and efficient design.

Table 2.6: SNR_r under finite-precision coefficients for the design CLS32-15b and its counterpart in [37].

Wordlength (bits)	32	16	12	8
Design CLS32-15b	179.16	96.64	67.03	64.16
Design in [37]	82.93	79.07	62.32	45.00

Both approaches are considered superior relative to that reported in [37, 38] from the quality of reconstruction perspective reflected by the higher signal-to-reconstruction noise ratios. It is also sometimes possible to obtain increased minimum stopband attenuation and decreased passband ripple as can be seen from Table 2.5 by comparing the analysis filters of the design CLS32-15b with their counterparts in [37].

Coefficient quantization would affect the PR quality of QMF banks. To illustrate this, the SNR_r for the design CLS32-15b was measured for various coefficient wordlengths and the results are listed in Table 2.6 where rounding has been used. To demonstrate that the quality of reconstruction is still better than that obtained in [37], a comparison is included in Table 2.6. Moreover, coefficient quantization would affect the responses of the analysis/synthesis filters. To show the effect, the coefficients of the analysis filters of the design CLS32-15b were quantized to 8-bit accuracy. Figure 2.9 shows the amplitude responses of the filters (design CLS32-15b/8). The measured values of A_{si}, δ_{pi}, SNR_r, and Δ_i are listed in Table 2.5.

As indicated earlier, low-delay two-channel QMF banks are used to construct tree-structured subband coding systems. An example of an 8-channel bank using low-delay two-channel QMF banks is shown in Fig. 2.10. The first analysis filters in the structure are of lengths $N_0 = N_1 = 32$ with delays $k_0 = 7$ s and $k_1 = 8$ s,

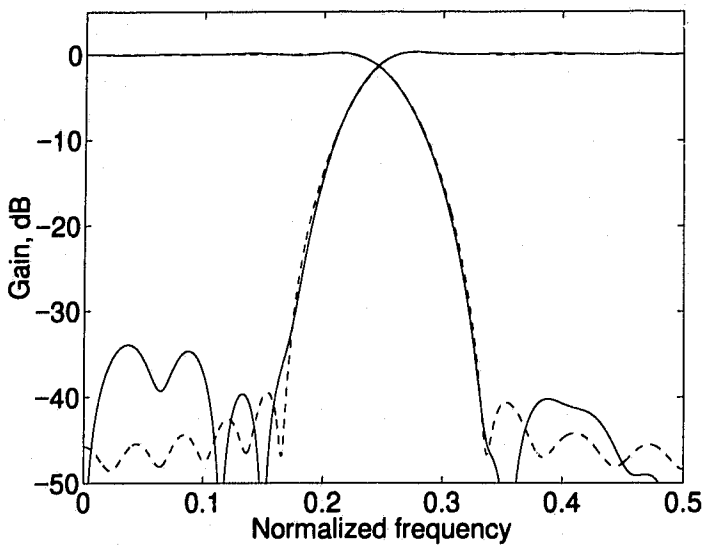


Figure 2.9: The amplitude responses of the analysis filters for the design CLS32-15b/8 (solid lines). The dashed lines show the amplitude responses using maximum machine precision.

respectively (design CLS32-15a). The second analysis filters are of lengths $N_0 = N_1 = 22$ with delays $k_0 = 3$ s and $k_1 = 4$ s, respectively (design CLS22-7). Finally, the last analysis filters are of lengths $N_0 = 8$ and $N_1 = 12$ with delays $k_0 = 2$ s and $k_1 = 3$ s, respectively (design LAG0812). The system delay is 49 s compared to 109 s for the linear-phase case. The SNR_r for a ramp input signal was determined as 175.34 dB.

To demonstrate the application of real signals to the 8-channel bank, a sound signal was applied to the input of the analysis bank. The signal was sampled at 8192 Hz. The time-domain representation and power spectrum of the sampled sound signal are shown in Fig. 2.11. Figures 2.12 and 2.13 show time-domain representations and power spectrums of the subband signals (refer to Fig. 1.3(a)). Then the subband signals were input to the synthesis bank (refer to Fig. 1.3(b)). Figure 2.14 shows the time-domain representation and the power spectrum of the reconstructed

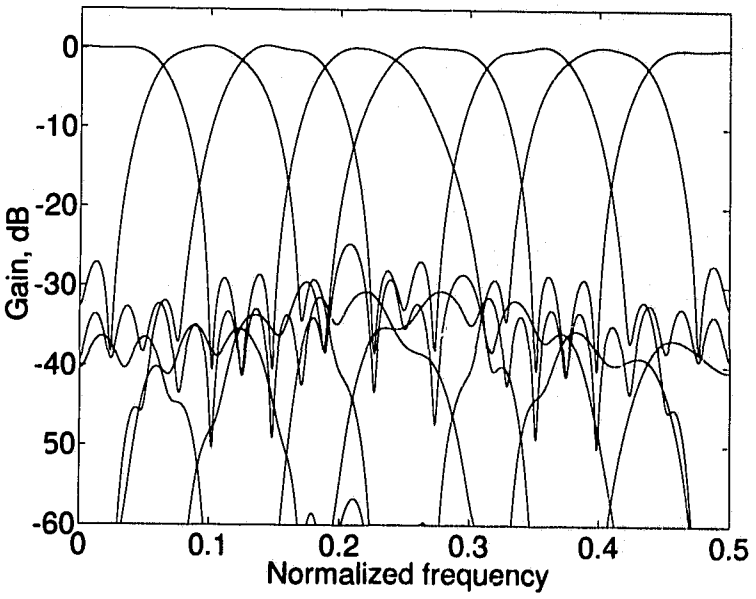
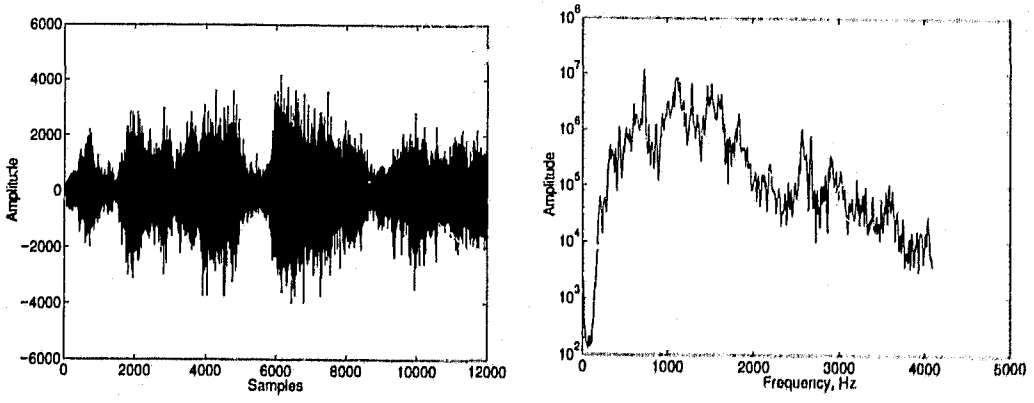


Figure 2.10: Amplitude responses of the analysis filters of an 8-channel filter bank.



(a) (b)

Figure 2.11: A *laughter* input sampled sound signal. (a) Time-domain representation. (b) Power spectrum.

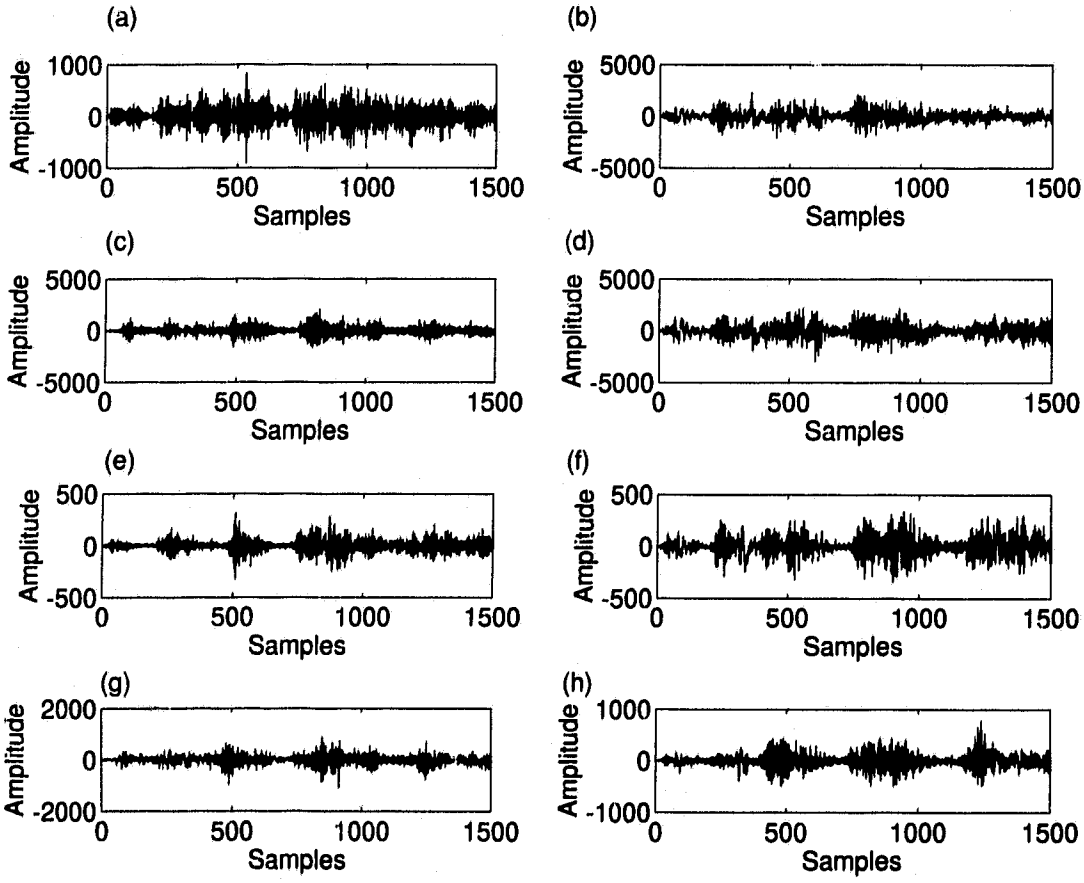


Figure 2.12: Time-domain representations of the subband signals. Label (a) refers to the first subband signal in Fig. 1.3(a) (top channel) while label (h) refers to the eighth subband signal (bottom channel).

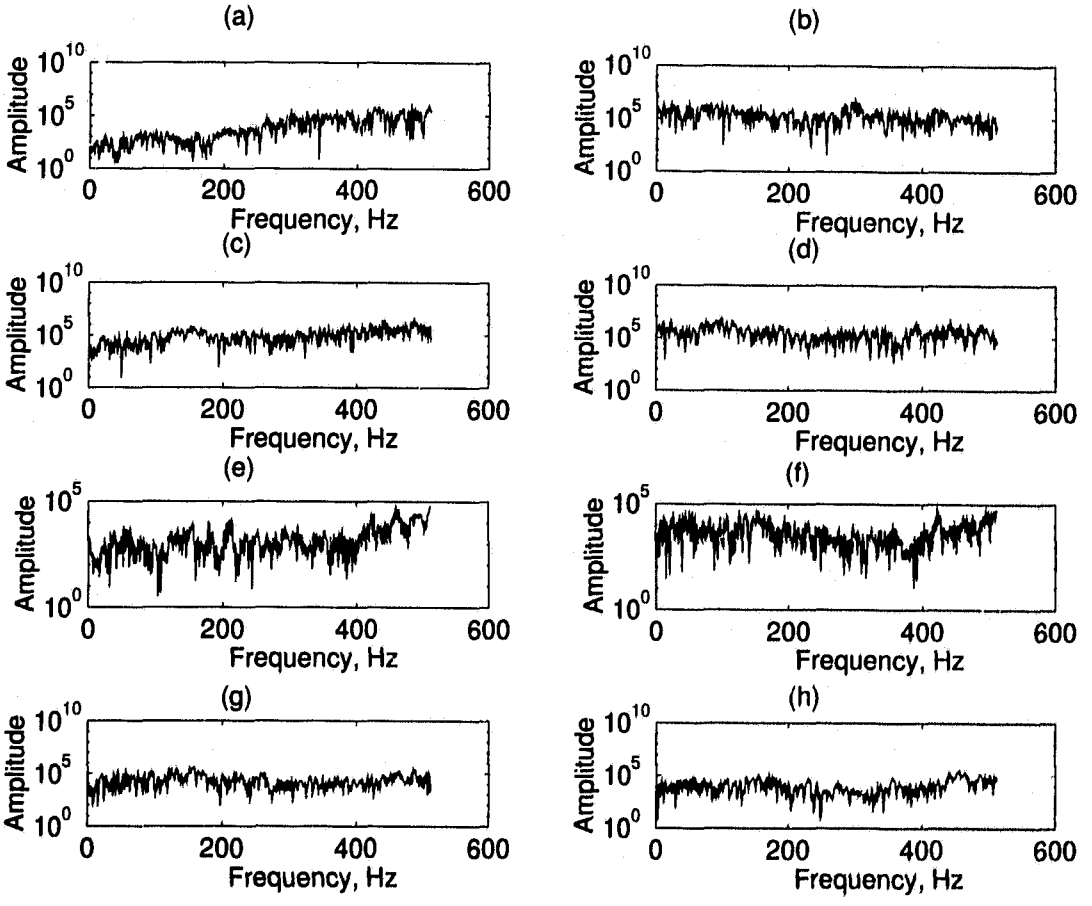
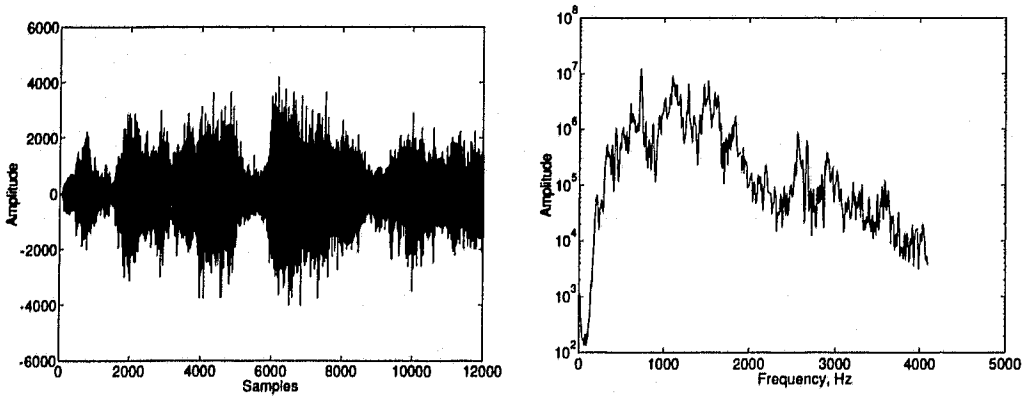


Figure 2.13: Power spectrums of the subband signals.



(a) (b)

Figure 2.14: The laughter reconstructed sampled sound signal. (a) Time-domain representation. (b) Power spectrum.

sampled signal. The SNR_r was determined as 156.97 dB which verifies that the signal is perfectly reconstructed.

2.6 Digital Transmultiplexers

In digital telephone networks, it is sometimes necessary to convert from time-division multiplexing (TDM) format to frequency-division multiplexing (FDM) format and vice versa. This is usually done using synthesis and analysis banks. However, guard bands were required between the frequency bands to minimize crosstalk and sharp cutoff filters were used to extract the desired frequency bands. A larger guard band implies larger permissible transition band (and hence lower order) for the analysis filters. The received signals suffered from magnitude and phase distortions and the transmission bandwidth could not be efficiently utilized because of the inevitable guard bands.

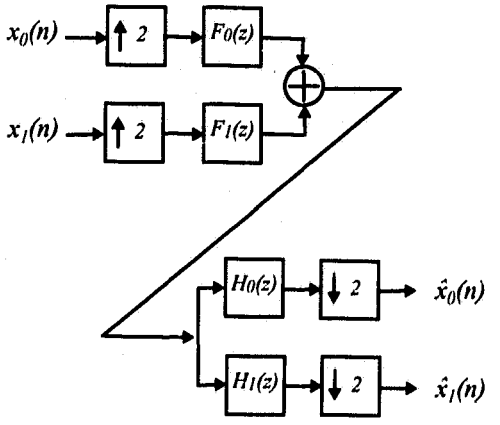


Figure 2.15: A two-input transmultiplexer.

A novel approach to transmultiplexing was proposed by Vetterli [7]. In this approach, crosstalk is permitted in the TDM/FDM converter (synthesis bank) and then cancelled by the FDM/TDM converter (analysis bank). Crosstalk terms can be completely eliminated by careful choice of the relation between the synthesis and analysis filters (similar to alias cancellation in QMF banks).

The simple case of transmultiplexing two signals is shown in Fig. 2.15. The signals are related as [6]

$$\hat{X}_i(z) = \tfrac{1}{2}[H_i(z^{1/2})Y(z^{1/2}) + H_i(-z^{1/2})Y(-z^{1/2})], \quad i = 0, 1 \tag{2.24}$$

where

$$Y(z) = F_0(z)X_0(z^2) + F_1(z)X_1(z^2) \tag{2.25}$$

From (2.24)-(2.25), the conditions for PR are

$$\frac{1}{2}[H_i(z)F_j(z) + H_i(-z)F_j(-z)] = \begin{cases} z^{-2\hat{k}} & i = j \\ 0 & i \neq j \end{cases} \tag{2.26}$$

where $\hat{k}$ is a positive integer. The first condition is the pure delay constraint and the second condition is the crosstalk free constraint. Crosstalk can be exactly eliminated by choosing $F_0(z) = 2z^{-1}H_1(-z)$ and $F_1(z) = -2z^{-1}H_0(-z)$. Accordingly, (2.26) can have the form

$$z^{-1}[H_0(z)H_1(-z) - H_0(-z)H_1(z)] = z^{-2\hat{k}} \tag{2.27}$$

or

$$H_0(z)H_1(-z) - H_0(-z)H_1(z) = z^{-k} \tag{2.28}$$

where $k = 2\hat{k} - 1$.

Let $H_0(z)$ and $H_1(z)$ be the transfer functions of a lowpass filter with a desired passband group delay k_0 s and a highpass filter with a desired passband group delay k_1 s, respectively. From (2.28), $k = k_0 + k_1$, which is assumed to be an odd integer. It is easy to verify that (2.28) can be expressed in the time domain as in (2.4). Consequently, the design procedures described in Sections 2.3 and 2.4 can be applied to design filters H_0 and H_1 and hence the analysis filters designed in Examples 2.1-2.4 can be used in the transmultiplexer design. This observation agrees with the results reported in [7, 27]. It is noted that the system delay is $\hat{k}$ s.

As an example, two sampled sound signals were applied to the synthesis bank of Fig. 2.15 using the filters of Example 2.4 (design CLS32-15a) and their time-domain representations are shown in Fig. 2.16(a) and (b). Figure 2.16(c) shows the time-domain representation of the channel signal and the time-domain representations

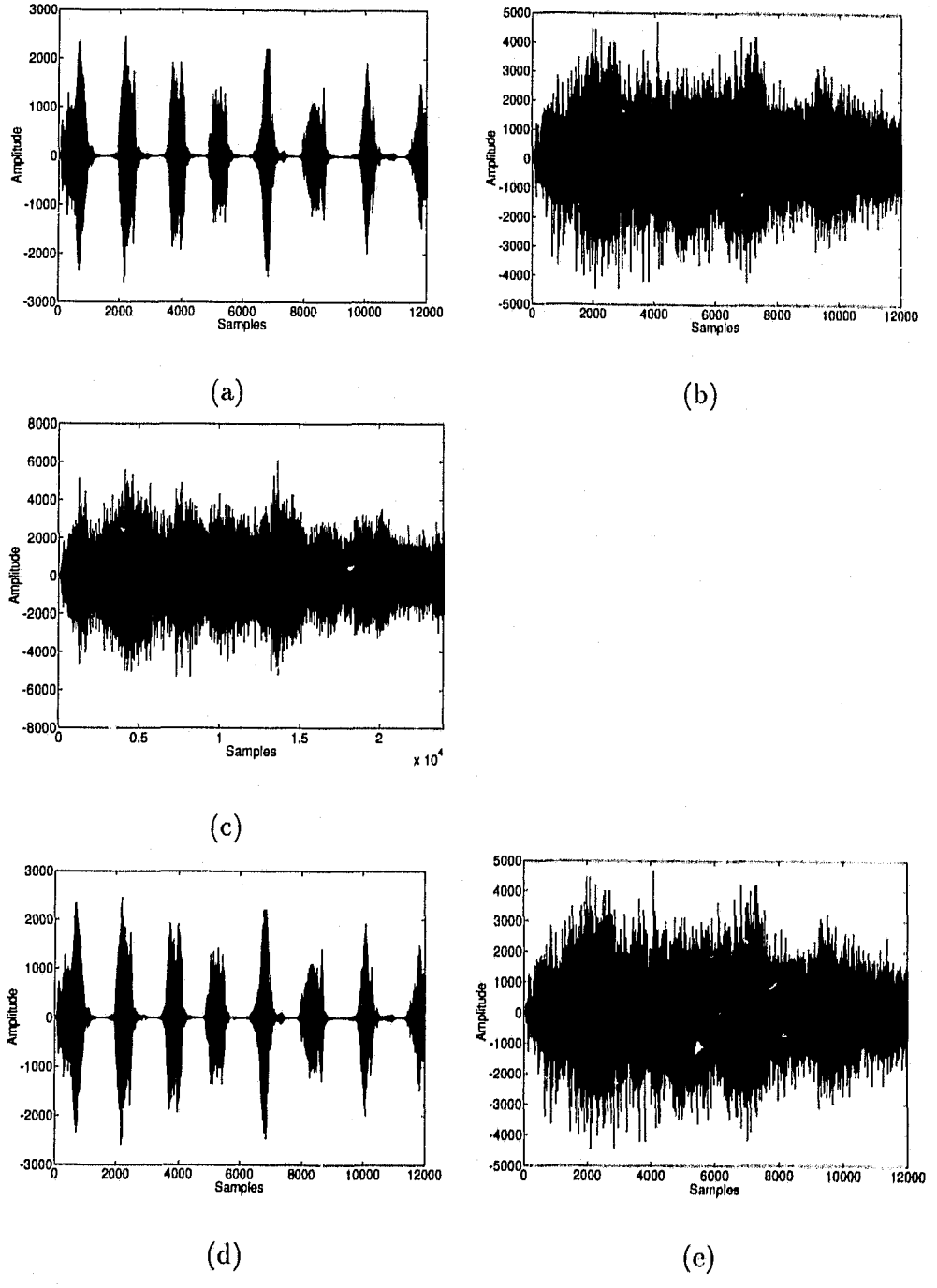


Figure 2.16: Time-domain representation of (a) a *bird chirp* sampled sound signal, (b) a *chinese gong* sampled sound signal, (c) the channel signal,
(d) the bird chirp reconstructed sampled sound signal, and (e) the chinese gong reconstructed sampled sound signal.

of the reconstructed signals are shown in Fig. 2.16(d) and (e). The SNR_r of the reconstructed signals were determined as 167.07 dB and 179.54 dB, respectively.

2.7 Conclusions

Two constrained optimization approaches have been applied for the design of two-channel PR FIR QMF banks with low delays. The first approach is based on the Lagrange-multiplier method and can be used to design banks with filters of unequal lengths. The approach is simple, efficient, flexible, and leads to a closed-form exact solution. The second approach can be used to design filters with equal as well as unequal lengths. In this approach, the design is formulated as a quadratic constrained least-squares minimization problem which can be solved using standard minimization algorithms. The two approaches have been compared and the merits of each have been identified. The two approaches have also been compared with other known methods for the design of FIR QMF banks and found to yield reduced reconstruction error and it is sometimes possible to obtain increased minimum stopband attenuation and decreased passband ripple. The results have been extended to perfect transmultiplexers.

Chapter 3

New Approaches for the Design of Filter Banks

3.1 Introduction

As mentioned earlier, cosine-modulated filter banks are attractive from the perspective of design and implementation. From the design point of view, only a prototype filter needs to be designed. From the implementation point of view, the cost of the analysis or synthesis bank is equal to that of one filter plus the cost of a fast discrete cosine transformer. The design of cosine-modulated pseudo-QMF banks can be performed using any unconstrained nonlinear optimization technique.

For some applications, linear-phase QMF banks with equiripple reconstruction error are required. A Chebyshev design of QMF banks was introduced in [16]. The WLS technique, developed by Lawson [83] and improved by others [84]-[86], produces an equiripple design if a suitable least-squares weighting function is used.

In this chapter, the iterative technique developed by Lim, Yang, and Koh [18] is reformulated and applied to design cosine-modulated pseudo-QMF banks. The technique linearizes the error measure and expresses it in the form of a set of lin-

ear equations. The algorithm results in saving in computation and design time. Design examples are included to illustrate the advantages of the design method. Also, a WLS approach for the design of two-channel, linear-phase FIR QMF banks is described. The design is carried out by using the iterative technique developed by Chen and Lee [17] along with certain modifications based on the weight update technique described by Sunder and Ramachandran [86]. The method is used to design QMF banks having equiripple reconstruction error with analysis and synthesis filters having equiripple or least-squares stopband errors. The method is then compared with other WLS methods described in the literature [83]-[85].

3.2 Design of Pseudo-QMF Banks

Generally speaking, cosine-modulated pseudo-QMF banks are designed with approximate alias cancellation [27]. Phase distortion is eliminated as the channel transfer function can be put in the form

$$T(z) = \frac{z^{-(N-1)}}{M} \sum_{k=0}^{M-1} H_k(z^{-1}) H_k(z)$$

which shows that $T(z)$ has linear phase where N is the length of the FIR filters involved. The remaining distortion, the amplitude distortion, is minimized.

The impulse responses of the analysis and synthesis filters are given by

$$h_k(n) = 2h(n)\cos\left[\frac{\pi}{M}(k+0.5)\left(n - \frac{N-1}{2}\right) + \phi_k\right]$$

$$f_k(n) = 2h(n)\cos\left[\frac{\pi}{M}(k+0.5)\left(n - \frac{N-1}{2}\right) - \phi_k\right]$$

for $0 \le n \le N-1$, $0 \le k \le M-1$, where $\phi_k = (-1)^k \pi/4$; $h(n)$ is the symmetrical impulse response of the prototype filter of length N. Thus, the design problem is reduced to designing the prototype filter.

3.2.1 Prototype Filter Design

The error measure for the design can be expressed as

$$E = \sum_{\omega=0}^{\omega_a} [\mid H(e^{j\omega}) \mid^2 + \mid H(e^{j(\omega-\pi/M)}) \mid^2 -1]^2 + \alpha \sum_{\omega=\omega_s}^{\pi} \mid H(e^{j\omega}) \mid^2 \qquad (3.1)$$

where α is a positive constant. The first component represents the flatness requirement for the channel amplitude response and the second component represents the stopband error measure.

Let $\Omega_r = \{\omega_1, \ldots, \omega_n\}$ and $\Omega_s = \{\omega_s, \ldots, \omega_m\}$ be dense sets of frequencies linearly distributed in the ranges $\omega = 0$ to $\omega = \pi/M$, and $\omega = \pi/2M + \epsilon$ to $\omega = \pi$, respectively. If

$$\mathbf{x} = \begin{cases} \left[\, 2h(\frac{N}{2}-1) \quad \cdots \quad 2h(0) \,\right]^T & \text{if } N \text{ is even} \\ \left[\, h(\frac{N-1}{2}) \quad 2h(\frac{N-3}{2}) \quad \cdots \quad 2h(0) \,\right]^T & \text{if } N \text{ is odd} \end{cases}$$

$$\mathbf{c} = \begin{cases} \left[\, \cos\frac{\omega}{2} \quad \cdots \quad \cos\frac{(N-1)\omega}{2} \,\right]^T & \text{if } N \text{ is even} \\ \left[\, 1 \quad \cos\omega \quad \cdots \quad \cos\frac{(N-1)\omega}{2} \,\right]^T & \text{if } N \text{ is odd} \end{cases}$$

$$\hat{\mathbf{c}} = \begin{cases} \left[\, \cos\frac{(\omega-\pi/M)}{2} \quad \cdots \quad \cos\frac{(N-1)(\omega-\pi/M)}{2} \,\right]^T & \text{if } N \text{ is even} \\ \left[\, 1 \quad \cos(\omega-\pi/M) \quad \cdots \quad \cos\frac{(N-1)(\omega-\pi/M)}{2} \,\right]^T & \text{if } N \text{ is odd} \end{cases}$$

then

$$\mid H(e^{j\omega}) \mid^2 = \mathbf{x}^T\mathbf{Q}\mathbf{x}$$

$$\mid H(e^{j\omega}) \mid^2 + \mid H(e^{j(\omega-\frac{\pi}{M})}) \mid^2 = \mathbf{x}^T\mathbf{P}\mathbf{x}$$

where the matrices $\mathbf{P}$ and $\mathbf{Q}$ are defined as

$$\mathbf{Q} = \mathbf{c}\mathbf{c}^T, \quad \mathbf{P} = \mathbf{c}\mathbf{c}^T + \hat{\mathbf{c}}\hat{\mathbf{c}}^T$$

and are symmetric. The error measure in (3.1) can be expressed as

$$E = \sum_{\omega \in \Omega_r} (\mathbf{x}^T \mathbf{P} \mathbf{x})^2 - \mathbf{x}^T \mathbf{T} \mathbf{x} + C \tag{3.2}$$

where C is a constant and

$$\mathbf{T} = 2 \sum_{\omega \in \Omega_r} \mathbf{P} - \alpha \sum_{\omega \in \Omega_s} \mathbf{Q}$$

To obtain the optimum solution for $\mathbf{x}$, E is differentiated with respect to elements of $\mathbf{x}$ to get

$$\left[2 \sum_{\omega \in \Omega_r} \mathbf{x}_{opt}^T \mathbf{P} \mathbf{x}_{opt} \mathbf{P} - \mathbf{T} \right] \mathbf{x}_{opt} = \mathbf{0} \tag{3.3}$$

If $\mathbf{x}_{opt}$ is expressed as $\mathbf{x}_{opt} = \mathbf{x}_j + \mathbf{e}_j$, where $\mathbf{e}_j$ is small, then substituting in (3.3) and retaining only the first-order terms of $\mathbf{e}_j$, (3.3) can be written in the form

$$\mathbf{A}_j \mathbf{e}_j = -\mathbf{B}_j \mathbf{x}_j \tag{3.4}$$

where

$$\mathbf{B}_j = 2 \sum_{\omega \in \Omega_r} \mathbf{x}_j^T \mathbf{P} \mathbf{x}_j \mathbf{P} - \mathbf{T}$$

$$\mathbf{A}_j = \mathbf{B}_j + 4 \sum_{\omega \in \Omega_r} \mathbf{P} \mathbf{x}_j \mathbf{x}_j^T \mathbf{P}$$

The set of linear equations in (3.4) can be solved by the Gaussian elimination method that avoids matrix inversion [79].

The error $\mathbf{e}_j$ can be minimized using the iterative algorithm of Lim *et al.* [18], which can be summarized as follows:

1. Set $j = 0$, choose ϵ_1 to be a small positive constant, and input $\mathbf{x}_0$.

2. Initialize frequencies Ω_r and Ω_s.

3. Using $\mathbf{x}_j$ as initial value, solve (3.4), and calculate $\mathbf{x}_{j+1} = \mathbf{x}_j + \mathbf{e}_j$.

4. If $\sum_i \mid e_j(i) \mid < \epsilon_1$, then output $\mathbf{x}_j$ and stop; otherwise, set $j = j + 1$ and go to step 3.

3.2.2 Design Examples and Comparisons

Two designs for cosine-modulated pseudo-QMF banks using the iterative technique are presented. The designs were performed using MATLAB [87] and run on a Sun SPARC station. The number of frequency points was set to $8N$ in the band 0 - π. The starting coefficients for the algorithm were those of FIR filters with the same lengths and edge frequencies designed using a window method.

Example 3.1:

The design involves an 8-channel system ($M = 8$) with a prototype filter of length $N = 48$, $\omega_s = 0.13\pi$, and $\alpha = 0.2$ using $\epsilon_1 = 10^{-5}$. The design terminated within 7 iterations. The amplitude response of the prototype filter is shown in Fig. 3.1 while the amplitude responses of the analysis filters are shown in Fig. 3.2. The plot of $M \mid T(e^{j\omega}) \mid$ in Fig. 3.3, which represents a scaled channel amplitude response, is seen to be very close to unity. Figure 3.4 shows the plot of $\sqrt{\sum_{\ell=1}^{M-1} \mid A_\ell(e^{j\omega}) \mid^2}$ versus frequency, which represents the aliasing error [27].

The same design was carried out using a *quasi-Newton* nonlinear optimization technique [39, 75] based on the Davidson-Fletcher-Powell (DFP) algorithm. The design terminated within 23 iterations and 107 function evaluations. The results are the same as in the iterative design. A comparison was carried out between the two techniques in terms of the number of iterations (NOI), the number of millions of floating point operations (MFLOPS), and the central processing unit (CPU) time

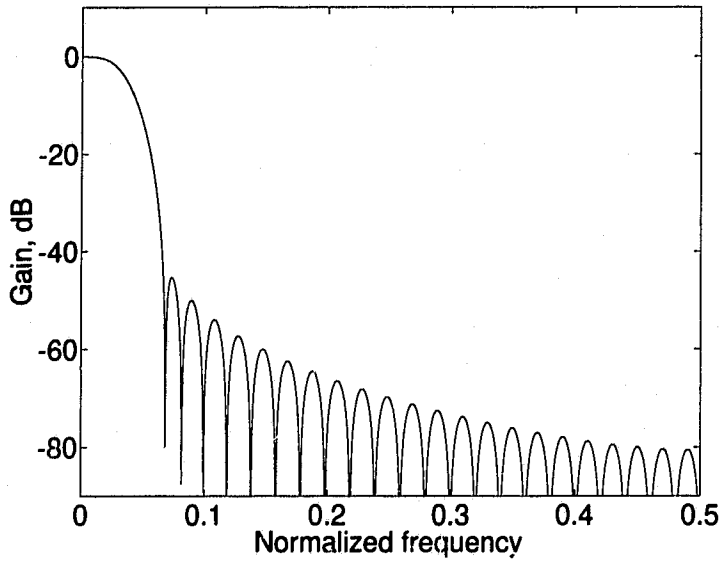


Figure 3.1: Amplitude responses of the prototype filter of Example 3.1.

in seconds. The results of the comparison are summarized in Table 3.1. It is clear that the applied iterative technique is far superior than the nonlinear optimization technique from the perspective of reducing the amount of computation and the design time.

Example 3.2:

A 17-channel filter bank (M=17) with a prototype filter of length $N = 102$ was designed using the iterative algorithm with $\omega_s = 0.075\pi$ and the same ϵ_1 and α as in Example 3.1. The design terminated within 12 iterations. The amplitude responses of the analysis filters are shown in Fig. 3.5. The plot of $M \mid T(e^{j\omega}) \mid$ is shown in Fig. 3.6. The same design was also carried out using nonlinear optimization. The design terminated within 33 iterations and 149 function evaluations. The results obtained are summarized in Table 3.1.

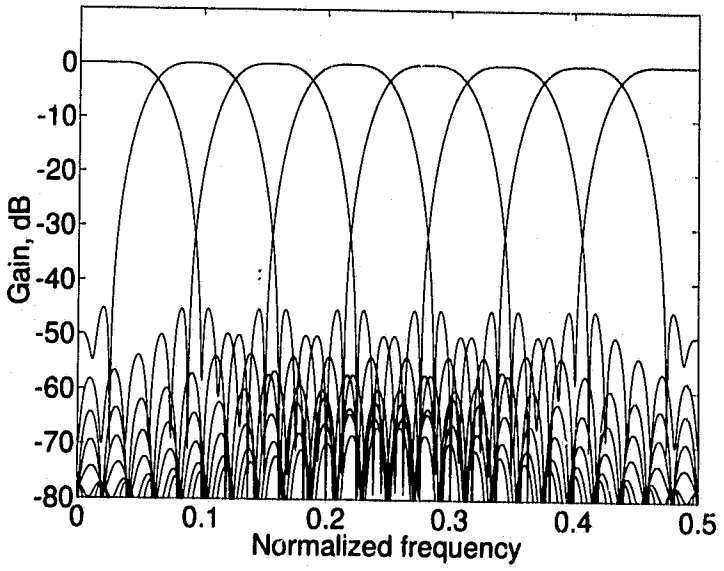


Figure 3.2: Amplitude responses of the analysis filters of Example 3.1.

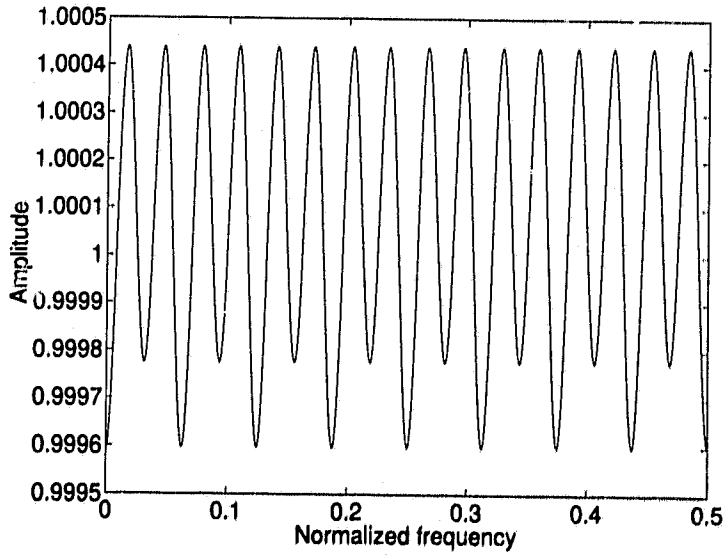


Figure 3.3: A scaled channel amplitude response for the filter bank of Example 3.1.

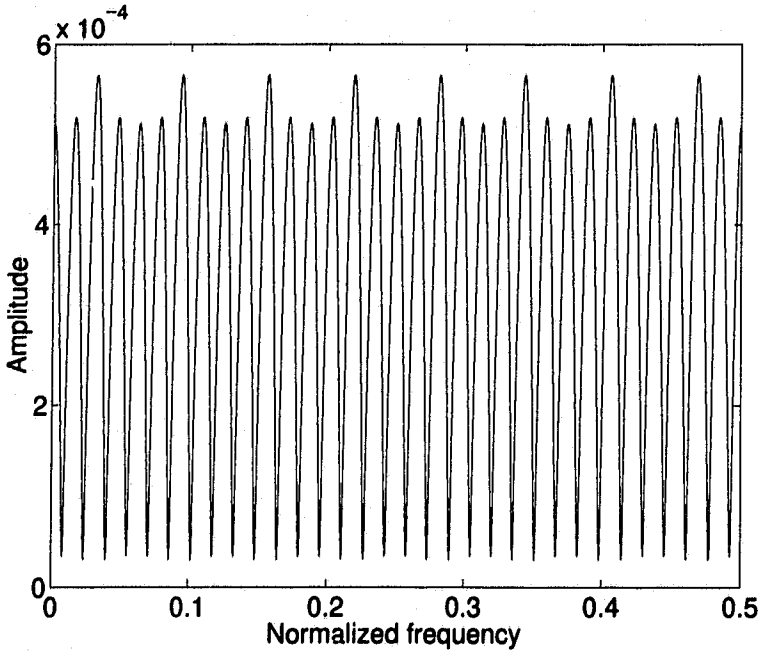


Figure 3.4: Aliasing error for the filter bank in Example 3.1.

Table 3.1: Computation comparisons. Method I is the iterative algorithm while method II is a quasi-Newton algorithm.

Example	Design	M	NOI	MFLOPS	CPU Time (s)
Example 3.1	Method I	8	7	12.9574	3.75
	Method II	8	23	179.6242	199.18
Example 3.2	Method I	17	12	182.2552	28.06
	Method II	17	33	2207.8220	1685.53

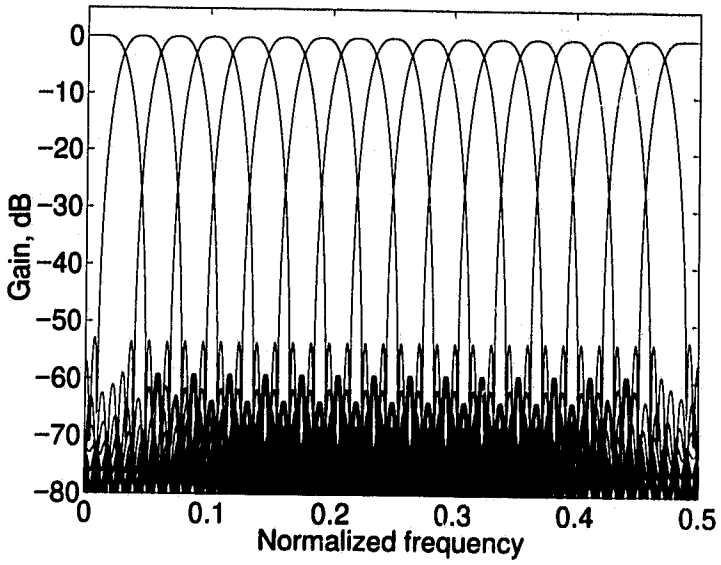


Figure 3.5: Amplitude responses of the analysis filters of Example 3.2.

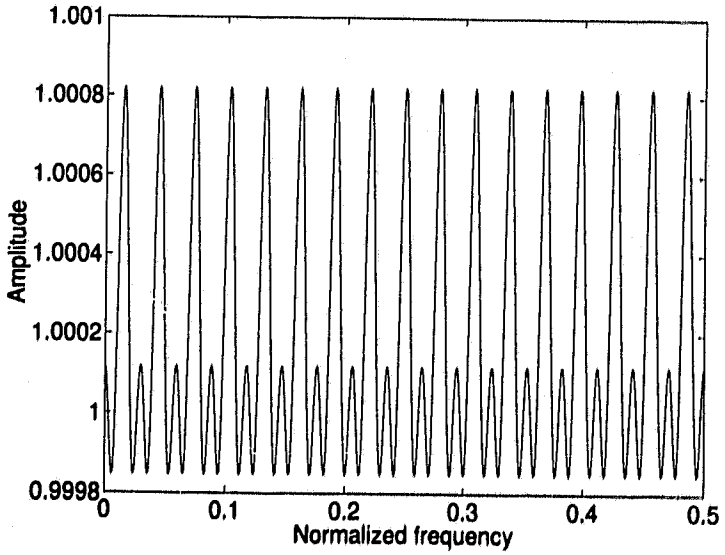


Figure 3.6: A scaled channel amplitude response for the filter bank of Example 3.2.

3.3 Weighted Minimax Design of QMF Banks

3.3.1 WLS Technique

In general, the error measure to be minimized in the WLS technique is of the form

$$E = \sum_{\omega_i \in \Omega} W(\omega_i) \mid e(\omega_i) \mid^2 \tag{3.5}$$

where $W(\omega)$ is a weighting function and $e(\omega_i)$ is the difference between a desired and an actual function value at frequency ω_i which belongs to the frequency set $\Omega = \{\omega_1, \ldots, \omega_K\}$. The weighting function is updated iteratively during the minimization procedure. The weights at the $(j+1)$th iteration are calculated using data available at the jth iteration according to the following relation

$$W^{j+1}(\omega) = \beta^j(\omega) W^j(\omega) \tag{3.6}$$

with

$$\beta^j(\omega) = \frac{\gamma^\theta(\omega)}{D} \tag{3.7}$$

where $\beta^j(\omega)$ is an updating function at the jth iteration, D is a normalization factor, and θ is a positive constant. In [83], $\gamma(\omega_i)$ is measured as the absolute error $\mid e(\omega_i) \mid$ and θ is set to 1.0. In [84], to prevent a weight of zero at the jth iteration from propagating further to the $(j+1)$th iteration, the function $[W^j(\omega)+\mu]$ is used instead of $W^j(\omega)$ in (3.6), where μ is a small positive constant. In [85], $\gamma(\omega_i)$ is measured as the value of the envelop connecting the absolute error maxima at frequency ω_i and $1.4 \leq \theta \leq 1.7$. In [86], $\gamma(\omega_i)$ is measured as the value of a local maximum of the absolute error where ω_i is located either between two local minima or between a local minimum and the band edge with $D = 1.0$ and $\theta = 1.5$.

3.3.2 Design of Linear-Phase Two-Channel QMF Banks

The design of QMF banks with equiripple reconstruction error with analysis/synthesis filters having equiripple stopband error follows that of the conventional QMF banks and the error measure to be minimized is [17]

$$E = \sum_{\omega=0}^{\pi/2} W(\omega)[\mid H_0(e^{j\omega}) \mid^2 + \mid H_0(e^{j(\omega+\pi)}) \mid^2 -1]^2 + \sum_{\omega=\omega_s}^{\pi} W(\omega) \mid H_0(e^{j\omega}) \mid^2 \quad (3.8)$$

where $H_0(e^{j\omega})$ is the frequency response of a lowpass filter of even length N with symmetrical impulse response, and ω_s is the stopband edge. Let $\Omega_r = \{\omega_1, \ldots, \omega_p\}$ and $\Omega_s = \{\omega_s, \ldots, \omega_m\}$ be dense sets of frequencies linearly distributed in the ranges $0 \leq \omega \leq \pi/2$ and $\omega_s \leq \omega \leq \pi$, respectively. In general, $W(\omega)$ can be expressed as

$$W(\omega_i) = \begin{cases} W_r(\omega_i) & \omega_i \in \Omega_r \\ W_s(\omega_i) & \omega_i \in \Omega_s \end{cases} \quad (3.9)$$

To design QMF banks with equiripple reconstruction error using analysis/synthesis filters having least-squares stopband errors, $W(\omega)$ is expressed as

$$W(\omega_i) = \begin{cases} W_r(\omega_i) & \omega_i \in \Omega_r \\ \alpha & \omega_i \in \Omega_s \end{cases} \quad (3.10)$$

where α is a positive constant. To minimize (3.8) in the weighted-minimax sense, the weighting function $W(\omega)$ must be appropriately chosen.

Considering the minimization problem in (3.8), the weights at the $(j+1)$th iteration are calculated using data available at the jth iteration according to the following relations

$$W_r^{j+1}(\omega) = \beta_r^j(\omega) W_r^j(\omega) \quad (3.11)$$

$$W_s^{j+1}(\omega) = \alpha\beta_s^j(\omega)W_s^j(\omega) \tag{3.12}$$

with

$$\beta_r^j(\omega) = \frac{p\gamma_r^\theta(\omega)}{\sum_{\Omega_r} W_r^j(\omega)\gamma_r^\theta(\omega)} \tag{3.13}$$

$$\beta_s^j(\omega) = \frac{q\gamma_s^\theta(\omega)}{\sum_{\Omega_s} W_s^j(\omega)\gamma_s^\theta(\omega)} \tag{3.14}$$

where p and q are the number of frequencies in Ω_r and Ω_s, respectively.

In this thesis, the weight updating method presented in [86] is modified in accordance with (3.11)-(3.14) and applied to the design of QMF banks. The iterative algorithm in [17] is used to minimize the error measure in (3.8). Let

$$\hat{\Omega}_r = \{\hat{\omega}_{1,r}, \ldots, \hat{\omega}_{l,r}\}$$
$$\check{\Omega}_r = \{\check{\omega}_{1,r}, \ldots, \check{\omega}_{o,r}\}$$

and

$$\hat{\Omega}_s = \{\hat{\omega}_{1,s}, \ldots, \hat{\omega}_{u,s}\}$$
$$\check{\Omega}_s = \{\check{\omega}_{1,s}, \ldots, \check{\omega}_{v,s}\}$$

be the sets of frequencies at which the absolute errors are local maxima and local minima for the sets Ω_r and Ω_s, respectively. Also let

$$\hat{\mathbf{E}}_r = \{\hat{E}_{1,r}, \ldots, \hat{E}_{l,r}\}$$
$$\hat{\mathbf{E}}_s = \{\hat{E}_{1,s}, \ldots, \hat{E}_{u,s}\}$$

be the sets of local absolute error maxima associated with Ω_r and Ω_s, respectively. There are four cases to be considered depending on the shape of the absolute error distribution, i.e.,

Case 1 $\check{\omega}_{1,r} > \hat{\omega}_{1,r}$ and $\hat{\omega}_{l,r} > \check{\omega}_{o,r}$

Case 2 $\check{\omega}_{1,r} < \hat{\omega}_{1,r}$ and $\hat{\omega}_{l,r} > \check{\omega}_{o,r}$

Case 3 $\check{\omega}_{1,k} > \hat{\omega}_{1,k}$ and $\hat{\omega}_{\zeta,k} < \check{\omega}_{\eta,k}$

Case 4 $\check{\omega}_{1,k} < \hat{\omega}_{1,k}$ and $\hat{\omega}_{\zeta,k} < \check{\omega}_{\eta,k}$

where

$$\zeta \equiv \begin{cases} l, & k = r \\ u, & k = s \end{cases}$$

and

$$\eta \equiv \begin{cases} o, & k = r \\ v, & k = s \end{cases}$$

The value of $\gamma_k(\omega_i)$, $k = r, s$, can be calculated as

$$\gamma_r(\omega_i) = \begin{cases} \hat{E}_{1,r} & \omega_i < \check{\omega}_{1,r} \\ \hat{E}_{(\nu+1),r} & \check{\omega}_{\nu,r} \leq \omega_i < \check{\omega}_{(\nu+1),r}, \quad \nu = 1, \ldots, (o-1), r \\ \hat{E}_{l,r} & \check{\omega}_{o,r} \leq \omega_i \end{cases} \tag{3.15}$$

for Case 1,

$$\gamma_r(\omega_i) = \begin{cases} \hat{E}_{(\nu),r} & \check{\omega}_{\nu,r} \leq \omega_i < \check{\omega}_{(\nu+1),r}, \quad \nu = 1, \ldots, (o-1), r \\ \hat{E}_{l,r} & \check{\omega}_{o,r} \leq \omega_i \end{cases} \tag{3.16}$$

for case 2,

$$\gamma_k(\omega_i) = \begin{cases} \hat{E}_{1,k} & \omega_i < \check{\omega}_{1,k} \\ \hat{E}_{(\nu+1),k} & \check{\omega}_{\nu,k} \leq \omega_i < \check{\omega}_{(\nu+1),k}, \quad \nu = 1, \ldots, (\eta-1), k \\ \hat{E}_{\zeta,k} & \check{\omega}_{\eta,k} = \omega_i \end{cases} \tag{3.17}$$

for Case 3, and

$$\gamma_k(\omega_i) = \begin{cases} \hat{E}_{\nu,k} & \check{\omega}_{\nu,k} \le \omega_i < \check{\omega}_{(\nu+1),k}, \quad \nu = 1, \ldots, (\eta-1), k \\ \hat{E}_{\zeta,k} & \check{\omega}_{\eta,k} = \omega_i \end{cases} \tag{3.18}$$

for Case 4.

3.3.3 WLS Algorithm

The algorithm can be summarized as follows:

1. Set $j = 0$, choose κ to be a small positive constant, and input $\mathbf{x}_0^j = [h_0(0) \ \cdots \ h_0(\frac{N}{2}-1)]^T$ where $h_0(n)$ is the impulse response of the lowpass filter H_0.

2. Initialize $W^0(\omega)$ as

$$W^0(\omega_i) = \begin{cases} 1.0 & \omega_i \in \Omega_r \\ \alpha & \omega_i \in \Omega_s \end{cases} \tag{3.19}$$

3. Minimize (3.8) to obtain $\mathbf{x}_{opt}^j$ using the iterative method in [17].

4. Check the termination condition as follows:

   (a) For the design of QMF banks with equiripple reconstruction error with analysis/synthesis filters having equiripple stopband errors, if

$$\max(\varepsilon_r, \varepsilon_s) \le \kappa$$

   then output $\mathbf{x}_{opt}^j$ and stop. Otherwise, update the weighting value according to (3.11)-(3.12), set $j = j+1$, $\mathbf{x}_0^j = \mathbf{x}_{opt}^{j-1}$, and go to step 3, where ε_r

and ε_s are the normalized maximum deviations of absolute reconstruction error and absolute stopband error, respectively, defined as

$$\varepsilon_r = \frac{|\max(\hat{\mathbf{E}}_r^j) - \min(\hat{\mathbf{E}}_r^j)|}{\max(\hat{\mathbf{E}}_r^j)}$$
$$\varepsilon_s = \frac{|\max(\hat{\mathbf{E}}_s^j) - \min(\hat{\mathbf{E}}_s^j)|}{\max(\hat{\mathbf{E}}_s^j)}$$

(b) For the design of QMF banks with equiripple reconstruction error with analysis/synthesis filters having least-squares stopband errors, if

$$\varepsilon_r \leq \kappa$$

then output $\mathbf{x}_{opt}^j$ and stop. Otherwise, update the weighting value according to (3.11), set $j = j + 1$, $\mathbf{x}_0^j = \mathbf{x}_{opt}^{j-1}$, and go to step 3.

3.3.4 Design Examples and Comparisons

Two examples are presented using the various WLS methods. The designs were performed using MATLAB. A comparison is made concerning the number of iterations, MFLOPS, CPU time, SNR_r, A_s, peak reconstruction error (PRE), ε_r, and ε_s for equiripple stopband error. In the following, WLS methods I, II, and III refer to the weight updating techniques in [86], [85], and [83], respectively, in accordance with (3.11)-(3.14). In dealing with Lawson's method, the absolute error is prevented from assuming a value less than a certain lower bound, e.g., $0.1 \max\{|\, e_k(\omega_i)\,|\}$, $k = r, s$, for all ω_i belonging to Ω_r or Ω_s, with $1.4 \leq \theta \leq 1.7$. This modification prevents a zero weight from occurring.

Table 3.2: Computation comparisons for Example 3.3 using the initial filter in (3.20).

WLS	NOI	MFLOPS	CPU Time (s)	SNR$_r$ (dB)	A_s (dB)	PRE (dB)	ε_r	ε_s
I	23	63.3781	38.899	56.11	44.32	0.0136	0.0082	0.0020
II	23	63.4079	40.099	56.08	44.32	0.0136	0.0097	0.0040
III	81	223.0548	142.506	56.12	44.31	0.0136	0.0062	0.0099

Example 3.3:

A QMF bank with $N = 32$ and $\omega_s = 0.6\pi$ was designed using the iterative method in [17] and the modified WLS algorithm (method I). The QMF bank was designed to have equiripple reconstruction error and equiripple stopband errors in the analysis/synthesis filters. The relative weight α was set to 0.5. The initial filter used was as in [17], i.e.,

$$h(n) = \begin{cases} 0.5 & \text{for } n = N/2 - 1,\ N/2 \\ 0.0 & \text{otherwise} \end{cases} \tag{3.20}$$

for starting the design process. The value of κ for terminating the design process was set to 0.01. The value of θ used by the WLS algorithm was set to 1.5. The design process converged within 23 iterations. The amplitude responses of the analysis filters are shown in Fig. 3.7. Also shown is the scaled channel amplitude response $2 \mid T(e^{j\omega}) \mid$. The design was also carried out using the WLS methods in [83] and [85] and a comparison is made in Table 3.2. The designs were also carried out using an initial filter with the same length and edge frequencies designed by the Remez algorithm [39] and the results are listed in Table 3.3.

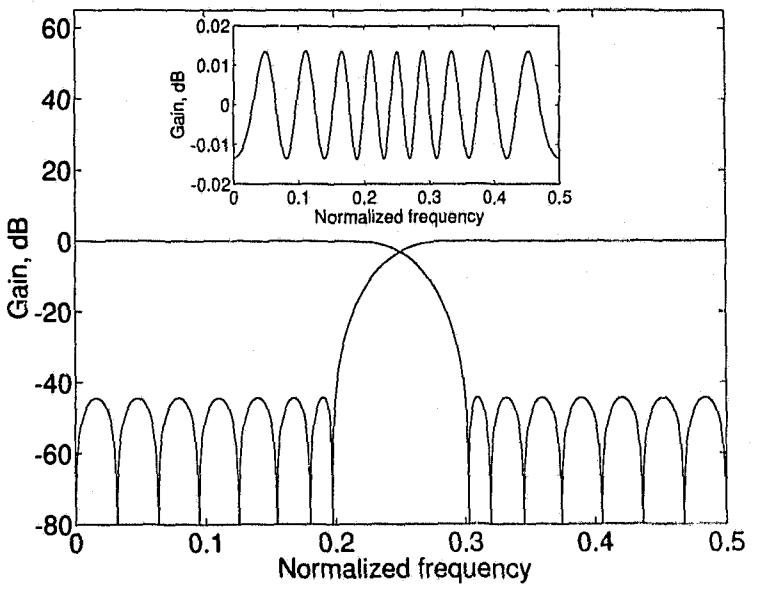


Figure 3.7: Amplitude responses of the analysis filters of the QMF bank of Example 3.3. The inset shows the scaled channel amplitude response.

Table 3.3: Computation comparisons for Example 3.3 using an initial filter designed by the Remez algorithm.

WLS	NOI	MFLOPS	CPU Time (s)	SNR_r (dB)	A_s (dB)	PRE (dB)	ε_r	ε_q
I	18	49.5911	30.866	56.11	44.32	0.0136	0.0082	0.0020
II	18	49.6193	31.583	56.08	44.32	0.0136	0.0098	0.0040
III	76	209.2673	132.510	56.12	44.31	0.0136	0.0062	0.0099

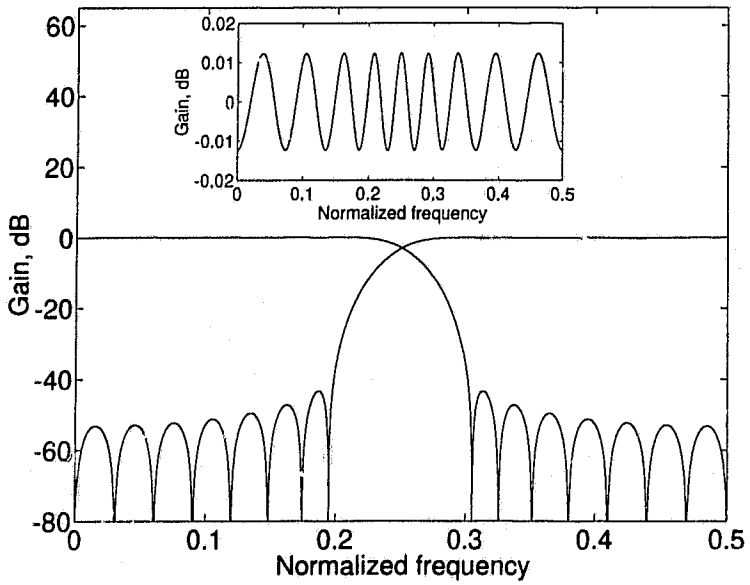


Figure 3.8: Amplitude responses of the analysis filters of the QMF bank of Example 3.4. The inset shows the scaled channel amplitude response.

Example 3.4:

A QMF bank was designed to have equiripple reconstruction error and least-squares stopband errors in the analysis/synthesis filters. The design parameters followed those of Example 3.3 but with $\kappa = 0.02$. The design process converged within 21 iterations using the initial filter in (3.20). The amplitude responses of the analysis filters are shown in Fig. 3.8. Also shown is the scaled channel amplitude response $2 \mid T(e^{j\omega}) \mid$. The design was carried out using the WLS methods in [83] and [85]. A comparison is made in Table 3.4 in which the data available from reference [17] is provided in parentheses. The designs were also carried out using an initial filter with the same length and edge frequencies designed by the Remez algorithm [39] and the results are listed in Table 3.5.

From Tables 3.2-3.5, it is noted that the design of QMF banks using the mod-

Table 3.4: Computation comparisons for Example 3.4 using the initial filter in (3.20).

WLS	NOI	MFLOPS	CPU Time (s)	SNR_r (dB)	A_s (dB)	PRE (dB)	ε_r
I	21	50.1439	33.000	56.97	43.44	0.0123	0.0051
II	22(22)	52.5487	35.400	56.92	43.41	0.0123	0.0107
III	28	66.8946	44.230	57.03	43.07	0.0123	0.0171

Table 3.5: Computation compari ns for Example 3.4 using an initial filter designed by the Remez algorithm.

WLS	NOI	MFLOPS	CPU Time (s)	SNR_r (dB)	A_s (dB)	PRE (dB)	ε_r
I	16	38.1495	25.133	56.97	43.43	0.0123	0.0050
II	15(15)	35.7769	24.983	56.87	43.41	0.0123	0.0173
III	23	54.8995	36.433	57.03	43.07	0.0123	0.0171

ified version of the method in [86] is comparable to the method in [85] regarding computational efficiency and design time.

Both methods are more efficient than the method in [83]. The quality of the filters is essentially the same in all methods. For the design of Example 3.3, the equiripple quality for the channel amplitude response, reflected by the quantity ε_r, of method III is the best obtained out of the three methods. Furthermore, the equiripple quality for the stopband amplitude response, reflected by the quantity ε_s, of method I is considered the best obtained out of the three methods. Also, for the design of Example 3.4, the equiripple quality is best obtained in method I.

3.3.5 Prescribed Specifications

The prescribed specifications for two-channel linear-phase FIR QMF banks are ω_s, E_r, the maximum allowable absolute reconstruction error (or PRE in dB), and E_s, the maximum allowable absolute stopband error. Prescribed specifications can be achieved for a given α by using the following design procedure [39]:

1. Estimate N; if N is odd, set $N = N + 1$.

2. Design a two-channel linear-phase FIR QMF bank with analysis/synthesis filters of lengths N using the WLS algorithm of Section 3.3.3 and determine $e_r = \max(\hat{\mathbf{E}}_r)$ and $e_s = \max(\hat{\mathbf{E}}_s)$.

   (A) If $e_r > E_r$ OR $e_s > E_s$, then do:

   (a) Set $N = N+2$, design a QMF bank with analysis/synthesis filters of lengths N using the WLS algorithm of Section 3.3.3, and determine e_r and e_s;

(b) If $e_r \leq E_r$ AND $e_s \leq E_s$, then go to step 3; else, go to step 2(A)(a).

(B) If $e_r < E_r$ AND $e_s < E_s$, then do:

(a) Set $N = N - 2$, design a QMF bank with analysis/synthesis filters with lengths N using the WLS algorithm of Section 3.3.3, and determine e_r and e_s;

(b) If $e_r > E_r$ OR $e_s > E_s$, then go to step 4; else go to step 2(B)(a).

3. Use the last set of filter coefficients of the corresponding N; and stop.

4. Use the last but one set of filter coefficients of the corresponding N; and stop.

Example 3.5:

A QMF bank with equiripple reconstruction error and equiripple stopband errors for the analysis/synthesis filters was designed using $\omega_s = 0.6\pi$, $\alpha = 0.5$ and $\kappa = 0.01$. The prescribed specifications were $E_r = 0.0005$ and $E_s = 0.005$. The design started using a lowpass filter of length $N = 30$ and repeated for an estimated length of $N = 58$. It was found that a lowpass analysis filter of length $N = 54$ satisfied the specifications with $e_r = 0.000399$ and $e_s = 0.000248$. The amplitude responses of the analysis filters are shown in Fig. 3.9. Also shown is the scaled channel amplitude response $2 \mid T(e^{j\omega}) \mid$.

3.4 Conclusions

Two design approaches for filter banks have been presented. In the first approach, the iterative algorithm reported in [18] has been extended to the design of cosine-modulated pseudo-QMF banks. The method is quite efficient in producing good

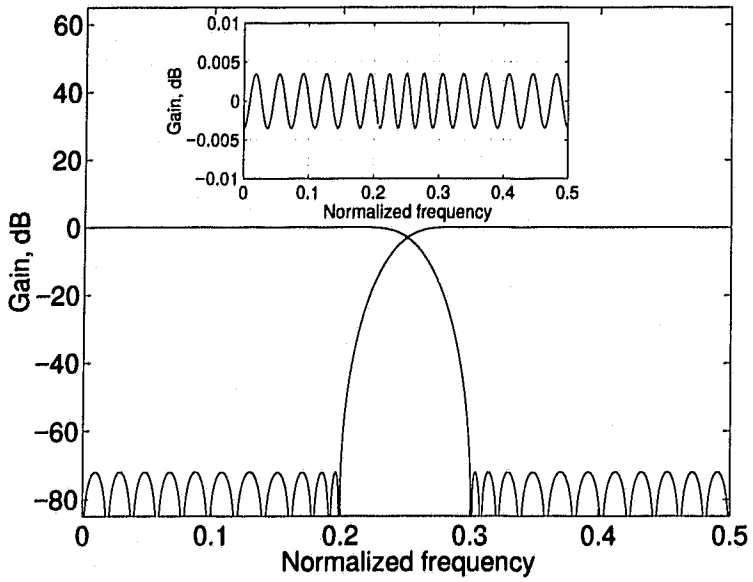


Figure 3.9: Amplitude responses of the analysis filters of the QMF bank of Example 3.5. The inset shows the scaled channel amplitude response.

designs while reducing the amount of computation and design time. In the second approach, the weight updating approach developed in [86] has been incorporated in the iterative method reported in [17] for the design of linear-phase FIR QMF banks. The approach was compared to others available in the literature and is considered more efficient than that in [83] and comparable to that in [85] for the design of linear-phase QMF banks. Design of QMF banks satisfying prescribed specifications has also been considered.

Chapter 4

VLSI Array Processors for Digital Filters

4.1 Introduction

Digital filters are integral parts of the polyphase implementations of decimators, interpolators, and filter banks that are presented in Chapter 6. Systolic and semi-systolic implementations of digital filters are reported in [54]-[56] using an algebraic approach in the z domain.

In this chapter, attention will be focused on the implementation of FIR filters as these filters are the most commonly used in PR filter banks. An array-processor scheme for implementing FIR filter algorithm is obtained using the SFG technique [52]. In this scheme, the outputs are localized in separate PEs. This scheme is an improved version of that obtained in [53]. The application of the SFG approach to obtain various array-processor implementations for linear-phase FIR filters is also explored. Two schemes are obtained by systematic application of the approach. The first scheme, in which the input and output are pipelined, is similar to one obtained in [88] using a methodology that relies on the z domain. In the second scheme, the

inputs are pipelined and broadcast and the partial results are pipelined. This scheme can be obtained easily by applying the method in [52]. A novel implementation is obtained in which each partial output stays in a separate PE to accumulate its terms. The implementations can be easily modified to implement linear-phase FIR differentiators, Nyquist filters, and Hilbert transformers.

4.2 FIR Filter Algorithms

An FIR filter algorithm is characterized by the relation [39]

$$y(nT) = \sum_{k=0}^{N-1} h(kT)x(nT - kT) \tag{4.1}$$

where N is the filter length.

Linear phase is obtained in FIR filters if the impulse response has even or odd symmetry about its midpoint, i.e.

$$h(nT) = \pm h[(N-1-n)T], \quad 0 \le n < N$$

The positive and negative signs represent the cases of even and odd symmetries, respectively. Using this condition, we can write

$$\begin{aligned} y(nT) &= \sum_{k=0}^{N-1} h(kT)x(nT - kT), \quad n \ge 0 \\ &= \sum_{k=0}^{N/2-1} h(kT)[\mathcal{E}^{-k}x(nT) \pm \mathcal{E}^{-(N-1-k)}x(nT)] \end{aligned} \tag{4.2}$$

for N even, and

$$y(nT) = \sum_{k=0}^{(N-1)/2} a(kT)[\mathcal{E}^{-k}x(nT) \pm \mathcal{E}^{-(N-1-k)}x(nT)] \tag{4.3}$$

for N odd, where $a(nT) = h(nT)$, for $n = 0, 1, \ldots, (N-3)/2$, and $a[(N-1)T/2] = 0.5h[(N-1)T/2]$. The symbol $\mathcal{E}$ is the shift operator defined by

$$\mathcal{E}^{-k}x(nT) = x(nT - kT)$$

Note that for the antisymmetrical case with N odd, $h[(N-1)T/2] = 0$.

4.3 The SFG Methodology

The design of an array structure for a given algorithm can be decomposed into three steps [52]: (a) mapping the algorithm to the dependence graph (DG) describing the data dependencies, (b) converting the DG to the signal flow graph (SFG) (which can be carried out by choosing a projection vector **d** and a schedule vector **s**), and (c) mapping the SFG onto an array processor. In the proposed implementations, the SFG array is the array processor and steps (b) and (c) are regarded as one step.

4.4 VLSI Array Processors for FIR Filters

The DG of the FIR filter algorithm is similar to that of the convolution algorithm and is given in [52].

Choosing a schedule vector $\mathbf{s} = [1 \quad -1]^T$, the DG in Fig. 4.1(a) is produced for a filter of length $N = 3$. Projecting the DG along the projection vector $\mathbf{d} = [0 \quad -1]^T$ results in an array processor that has an infinite number of PEs, which is not practical. However, this impracticality can be eliminated by recognizing the fact that the first and the second PEs are idle after one and two sampling periods, respectively, and every other PE is idle after 3 sampling periods (N sampling periods in general).

Table 4.1: Data flow/timing for the LP scheme (assuming $T = 1$ s).

T	PE(0)		PE(1)		PE(2)	
	$x(n)$	$h(k)$	$x(n)$	$h(k)$	$x(n)$	$h(k)$
T_0	$x(0)$	$h(0)$	$x(0)$	$h(1)$	$x(0)$	$h(2)$
T_1	$x(1)$	$h(2)$	$x(1)$	$h(0)$	$x(1)$	$h(1)$
T_2	$x(2)$	$h(1)$	$x(2)$	$h(2)$	$x(2)$	$h(0)$
T_3	$x(3)$	$h(0)$	$x(3)$	$h(1)$	$x(3)$	$h(2)$
T_4	$x(4)$	$h(2)$	$x(4)$	$h(0)$	$x(4)$	$h(1)$
T_5	$x(5)$	$h(1)$	$x(5)$	$h(2)$	$x(5)$	$h(0)$
T_6	$x(6)$	$h(0)$	$x(6)$	$h(1)$	$x(6)$	$h(2)$
T_7	$x(7)$	$h(2)$	$x(7)$	$h(0)$	$x(7)$	$h(1)$

Thus, the processing element PE(i) that computes $y(iT)$ can be used again to compute $y[(i + nN)T]$ for $n = 1, 2, \ldots$, where N is the number of PEs in the array. This scheme is called local processing (LP) scheme.

To obtain the array structure a data flow/timing information table is constructed as in Table 4.1 based on the data flow and timing for the DG of Fig. 4.1(a). Mapping the data flow/timing information available in the data flow/timing table onto an array structure, results in the array-processor scheme in Fig. 4.1(b) where all the data shown correspond to the sampling instant T_0 in Table 4.1. A ring memory is implemented in each PE for circulating the filter coefficients. Using ring memories is preferred to transmitting the filter coefficients across different PEs as reported in [53] to reduce the communication overhead and increase system speed. The output is provided through an output bus which is driven by a set of three-state drivers. One driver is enabled at a time in a round-robin fashion, thus each PE provides an output once every N sampling periods.

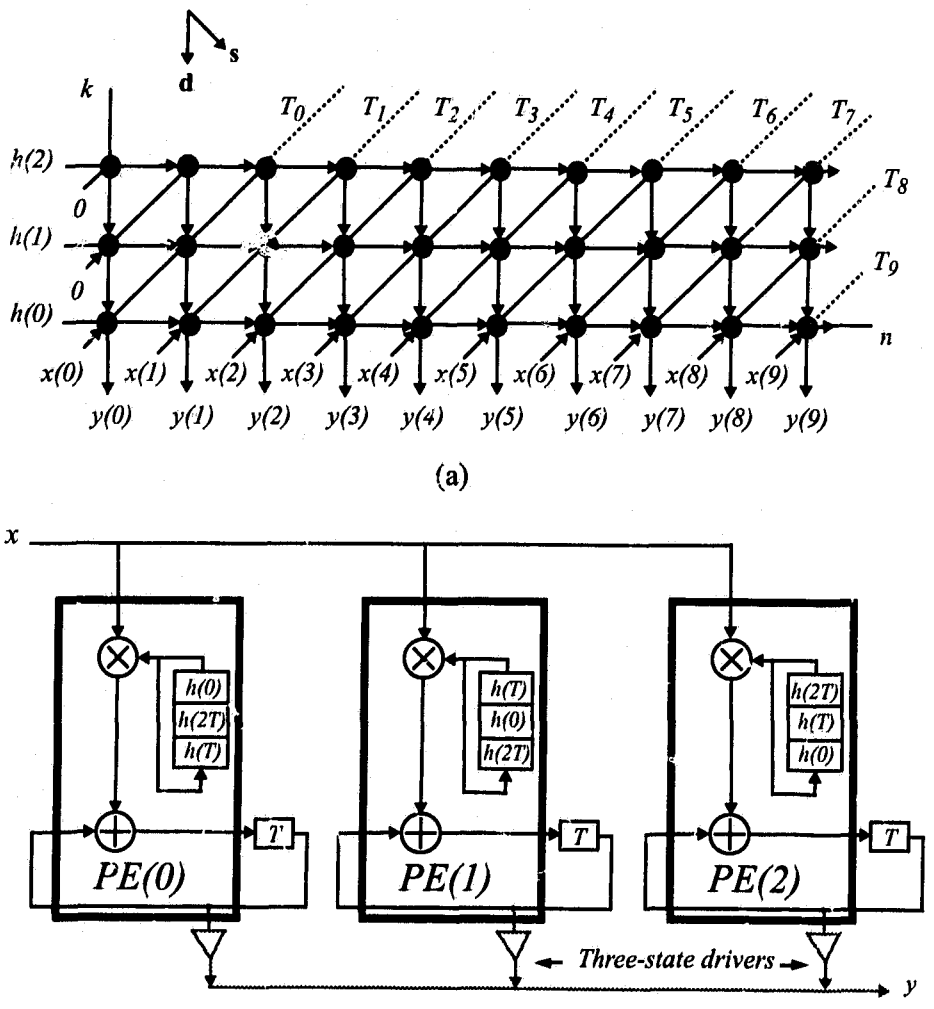


Figure 4.1: FIR local processing (LP) scheme. (a) DG for the FIR filter scheme for $N = 3$ (assuming $T = 1$ s) (b) The array processor.

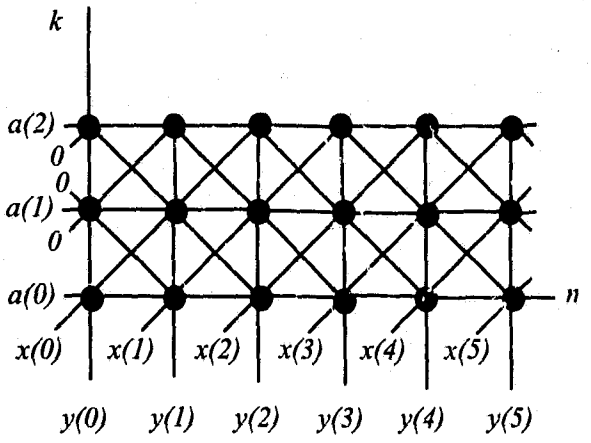


Figure 4.2: The DG for odd-length linear-phase filters ($N = 5$).

4.5 VLSI Array Processors for Linear-Phase FIR Filters

The array processors presented will be for symmetrical, odd-length filters. Structures for antisymmetrical and even-length filters can be deduced in a similar way.

The DG of the linear-phase FIR-filter algorithms can be derived by using a DG for a general FIR filter algorithm and folding it about its midline related to the midpoint in the coefficient space axis (k axis in Fig. 4.1). Based on the above and referring to (4.3), the DG for linear-phase odd-length filters is obtained as shown in Fig. 4.2.

Scheme I

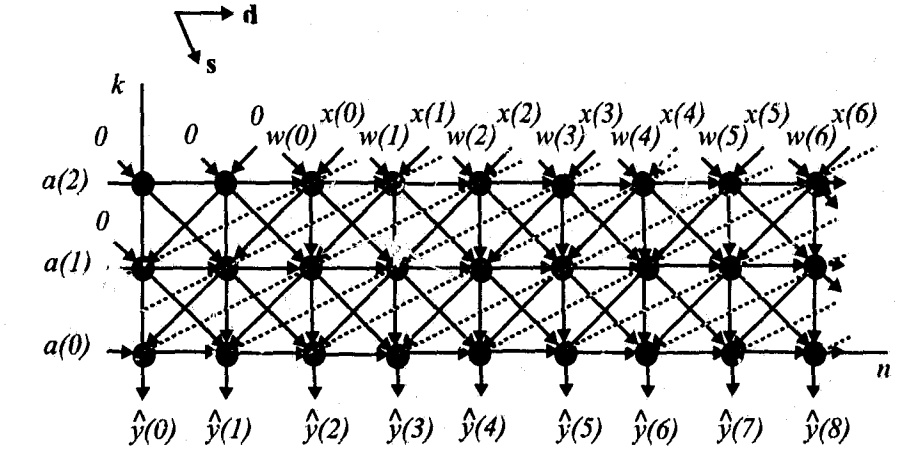
An array processor in which the inputs and the outputs are pipelined can be obtained if a schedule vector $\mathbf{s} = [1 \quad -2]^T$ is employed in the DG of Fig. 4.2. The modified DG is shown in Fig. 4.3(a) where the inputs $x(nT)$ and $w(nT) = x(nT)$ and the output $\hat{y}(nT) = \mathcal{E}^{-(N-1)/2}y(nT)$ are pipelined through the index space (n, k). Projecting the DG along the projection vector $\mathbf{d} = [1 \quad 0]^T$ results in the array processor of Fig. 4.3(b). The details of the PE involved are shown in Fig. 4.3(c). A similar scheme has been reported in [88].

Scheme II

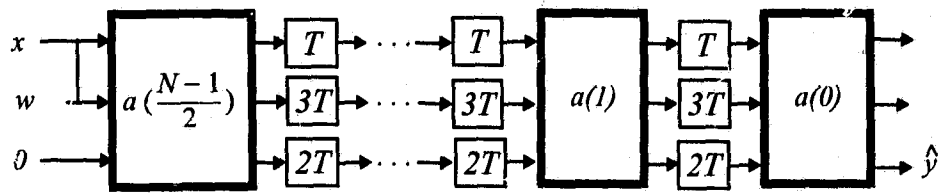
It is possible to obtain an array structure in which the inputs are broadcast/pipelined and the outputs are pipelined [89, 90, 91]. Choosing a schedule vector $\mathbf{s} = [1 \quad -1]^T$ in the DG of Fig. 4.2, the modified DG in Fig. 4.4(a) is produced, where the output $y(nT)$ and input $w(nT)$ are pipelined and the input $x(nT) = w(nT)$, $n = 0, 1, \ldots,$ is broadcast. Projecting the DG along the projection vector $\mathbf{d} = [1 \quad 0]^T$, the linear array processor in Fig. 4.4(b) is produced. Figure 4.4(c) shows the details of the PE involved.

Scheme III

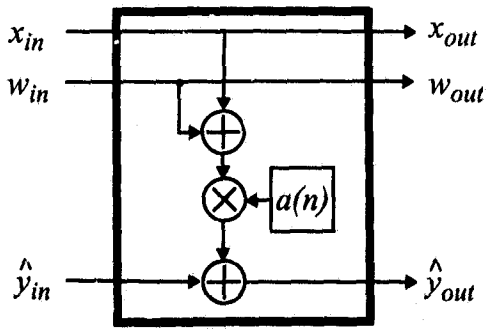
Figure 4.5(a) repeats the DG of Fig. 4.4(a). Projecting the DG along the projection vector $\mathbf{d} = [0 \quad -1]^T$ results in an array processor that has an infinite number of PEs, which is not practical. However, this impracticality can be eliminated by recognizing the fact that the first and second PEs are idle after one and two sampling periods,



(a)



(b)



(c)

Figure 4.3: Linear-phase FIR: scheme I. (a) Modified DG for scheme I (assuming $T = 1$ s). (b) Scheme I array processor. (c) Details of the PE involved.

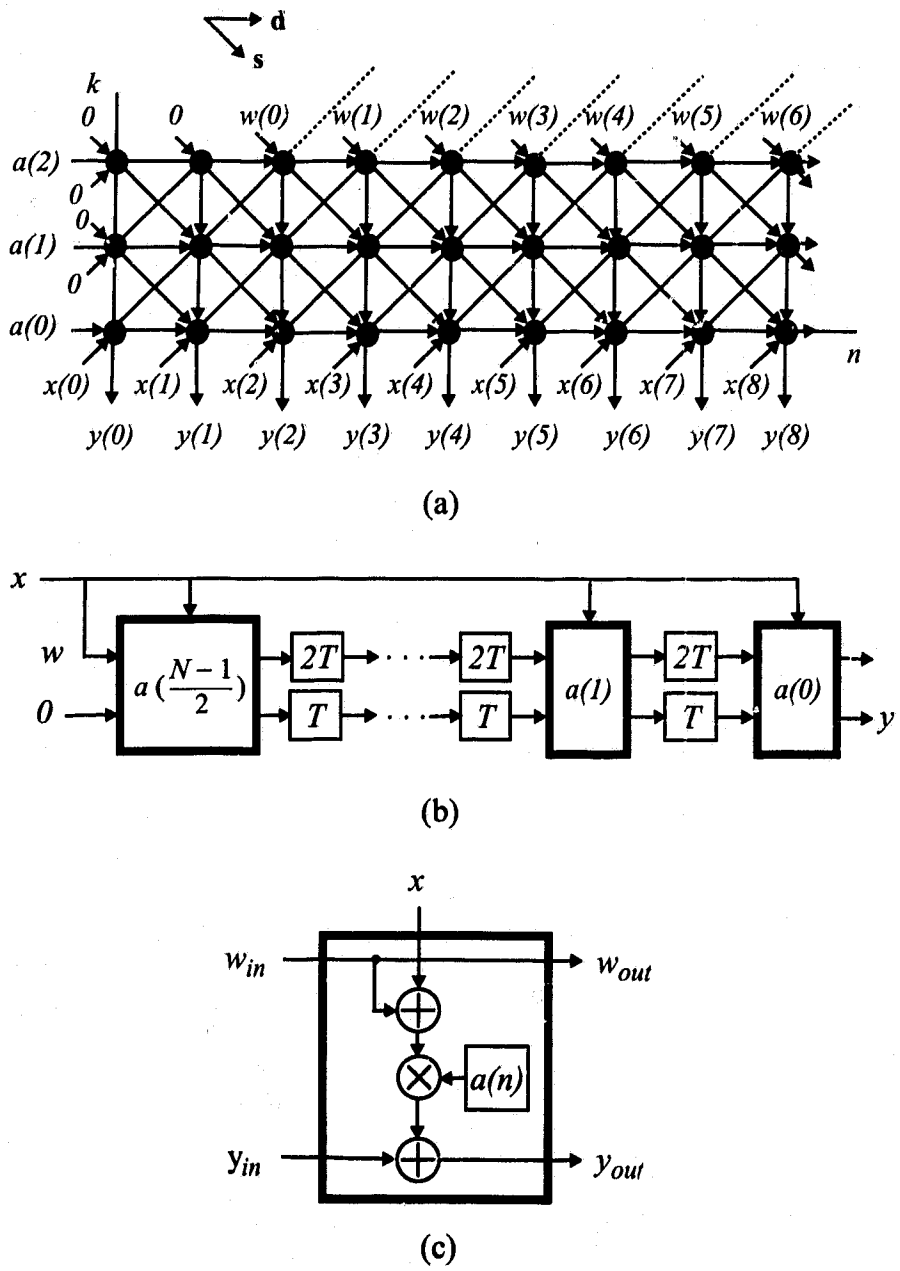
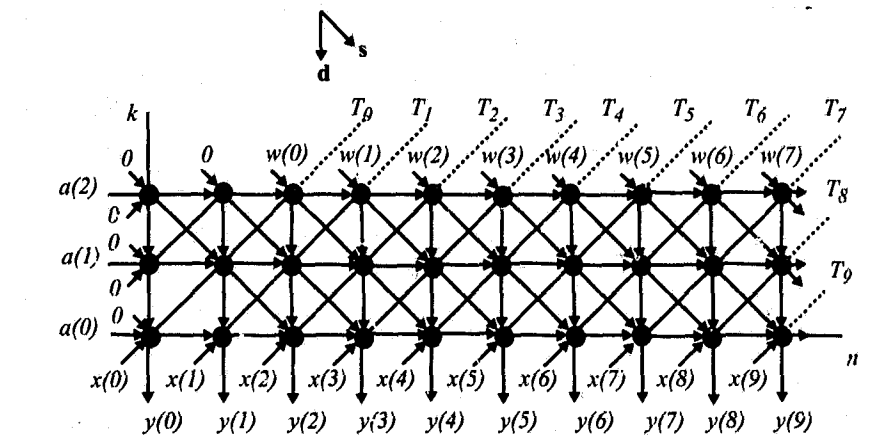
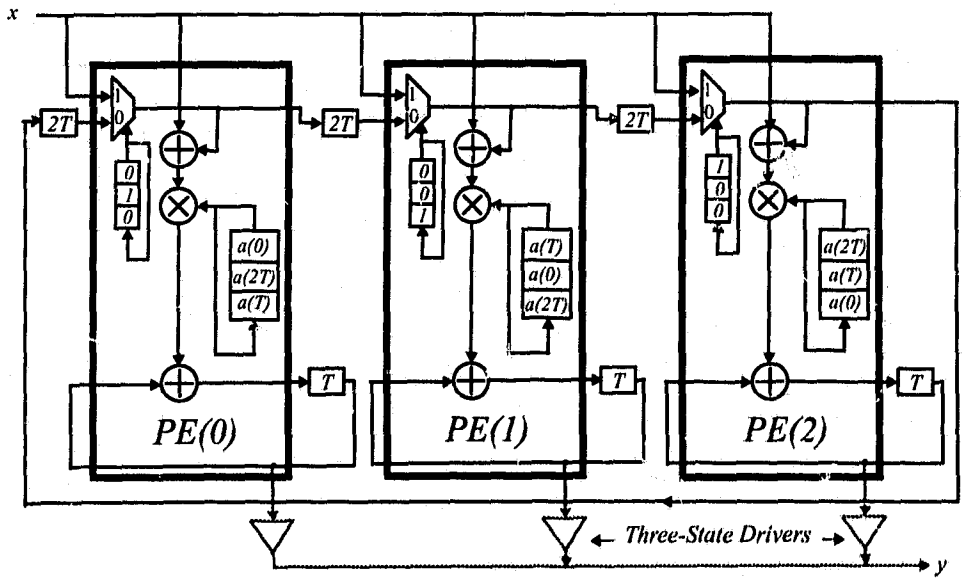


Figure 4.4: Linear-phase FIR: scheme II. (a) Modified DG for scheme II (assuming $T = 1$ s). (b) Scheme II array processor. (c) Details of the PE involved.



(a)



(b)

Figure 4.5: Linear-phase FIR: scheme III. (a) Modified DG for scheme III (assuming $T = 1$ s). (b) The array processor.

Table 4.2: Data flow/timing for scheme III (assuming $T = 1$ s).

T	PE(0)			PE(1)			PE(2)		
	$x(n)$	$a(k)$	$w(n)$	$x(n)$	$a(k)$	$w(n)$	$x(n)$	$a(k)$	$w(n)$
T_0	$x(0)$	$a(0)$	0	$x(0)$	$a(1)$	0	$x(0)$	$a(2)$	$w(0)$
T_1	$x(1)$	$a(2)$	$w(1)$	$x(1)$	$a(0)$	0	$x(1)$	$a(1)$	0
T_2	$x(2)$	$a(1)$	$w(0)$	$x(2)$	$a(2)$	$w(2)$	$x(2)$	$a(0)$	0
T_3	$x(3)$	$a(0)$	0	$x(3)$	$a(1)$	$w(1)$	$x(3)$	$a(2)$	$w(3)$
T_4	$x(4)$	$a(2)$	$w(4)$	$x(4)$	$a(0)$	$w(0)$	$x(4)$	$a(1)$	$w(2)$
T_5	$x(5)$	$a(1)$	$w(3)$	$x(5)$	$a(2)$	$w(5)$	$x(5)$	$a(0)$	$w(1)$
T_6	$x(6)$	$a(0)$	$w(2)$	$x(6)$	$a(1)$	$w(4)$	$x(6)$	$a(2)$	$w(6)$
T_7	$x(7)$	$a(2)$	$w(7)$	$x(7)$	$a(0)$	$w(3)$	$x(7)$	$a(1)$	$w(5)$
T_8	$x(8)$	$a(1)$	$w(6)$	$x(8)$	$a(2)$	$w(8)$	$x(8)$	$a(0)$	$w(4)$
T_9	$x(9)$	$a(0)$	$w(5)$	$x(9)$	$a(1)$	$w(7)$	$x(9)$	$a(2)$	$w(9)$

respectively, and every other PE is idle after 3 sampling periods ($(N+1)/2$ sampling periods in general). Thus, the processing element PE(i) that computes $y(iT)$ can be used again to compute $y[(i+nl)T]$ for $n = 1, 2, \ldots$, where $l = (N+1)/2$; hence the number of PEs in the array, N_p, is equal to $(N+1)/2$.

To obtain the array structure, a data flow/timing table is constructed as in Table 4.2 based on the data flow and timing for the DG of Fig. 4.5(a). Mapping the data flow/timing information available in Table 4.2 onto an array structure results in the array-processor scheme in Fig. 4.5(b) where all the data shown correspond to the sampling instant T_0 in Table 4.2. The control select signal for each multiplexer block is implemented by means of a ring memory. The multiplexers are used to update the signal $w(nT)$, $n = 0, 1, \ldots$, according to Table 4.2. Also, another ring memory is implemented in each PE for circulating the filter coefficients for each sampling period according to Table 4.2. Using ring memories is preferred over transmitting the control select signal and the filter coefficients across different PEs to reduce

the communication overhead and increase system speed. The output is provided through an output bus which is driven by a set of three-state drivers. One driver is enabled at a time in a round-robin fashion, thus each PE provides an output once every $(N+1)/2$ sampling periods.

4.6 Performance

The filter maximum processing rate depends on the hardware details. For the conventional direct-form FIR scheme [92] in which the inputs are pipelined and the outputs are added simultaneously, the minimum sampling period is determined by the time required to produce the filter output. Assigning a single-precision (sp) word for the input data and a double-precision (dp) word for the output data, we have

$$T > NT_{add-dp} + T_{mult} + T_r \tag{4.4}$$

where T_{add-dp}, and T_{mult} are the delays for a $2b$-bit adder, and $b \times b$ 2's-complement multiplier, respectively, N is the number of PEs in the array, b is the number of bits per word, and T_r is the delay figure of the register expressed as

$$T_r = T_s + T_d$$

where T_s is the register setup time and T_d is the register delay.

For the LP scheme , the minimum sampling period is determined by

$$T > \max[T_B + T_r,\ T_{add-dp} + T_{mult} + T_r] \tag{4.5}$$

where T_B is the delay of either the input or the output data bus which is proportional to the square of the number of PEs (i.e., N^2) [93]. As N increases, T_B increases

quadratically and may impose an upper bound on the processing rate for high-order filters.

It is clear that the dominant factor in the above equation is the time $T_{add-dp} + T_{mult}$. As this time gets shorter, the processing rate would increase until the bus delays would dominate the above formula for high-order filters.

For the conventional direct-form linear-phase FIR scheme [92] in which the inputs are pipelined and the outputs are added simultaneously, the minimum sampling period is determined by the time required to produce the filter output, i.e.,

$$T > T_{add-sp} + N_p T_{add-dp} + T_{mult} + T_r \tag{4.6}$$

where T_{add-sp} is the delay for a b-bit adder, and N_p is the number of PEs in the array. Clearly, such an implementation is practical only for low-order filters where the asynchronous adder does not dominate the response time of the system.

For scheme I, the minimum sampling period is determined by the delay in one pipeline stage. Thus

$$T > T_P + T_r \tag{4.7}$$

where T_P is the processing time expressed as

$$T_P = T_{add-sp} + T_{add-dp} + T_{mult}$$

It is noted that the above sampling time does not depend on the number of PEs involved which makes this scheme practical for high-order filters. Moreover, the full local communication feature makes the scheme attractive for high-speed signal processing applications.

For scheme II, the minimum sampling period is determined by the delay in one

pipeline stage and the data bus delay. Thus

$$T > \max[T_B + T_r, T_P + T_r] \tag{4.8}$$

As N_p increases, T_B increases quadratically and may impose an upper bound on the processing rate for high-order filters.

For scheme III, the minimum sampling period is determined by

$$T > \max[T_B + T_r + T_{MUX}, T_P + T_r] \tag{4.9}$$

where T_B is the delay of either the input or the output data bus, respectively, and T_{MUX} is the multiplexer delay. It is clear that the dominant factor in the above equation is the time T_P. As this time gets shorter, the processing rate would increase until the bus delays would dominate the above formula for high-order filters.

System latency is defined as the time elapsed between the application of a sample to the input and the appearance of the corresponding output sample. The latency for the conventional direct form realizations [92] is one sampling period T. The latency for the LP scheme is one sampling period T. The latency for scheme I is N_pT while that for schemes II and III is one sampling period T.

4.7 Comparisons and Discussion

In the previous sections, array-processor implementations have been obtained for linear-phase FIR filters. From the perspective of speed, higher data rates can be achieved with scheme I than with schemes II and III especially for high-order filters. In scheme III, the output word can be truncated (or rounded) locally in each PE. This would decrease the communication overhead compared to that in schemes I and

II in which the output word is to be truncated in the final PE if noise performance is to be improved. From the perspective of latency, schemes II and III have the minimum latency of one sampling period (assuming the same processing rate for all schemes).

The cascade form is less sensitive to coefficient quantization than the equivalent direct form realization in situations where quantization is very coarse, or for high-order systems with closely spaced zeros [92]. If $H(z)$ is the z transform of $h(nT)$, then the symmetry condition causes the zeros of the linear-phase transfer function $H(z)$ to occur in mirror-image pairs [39]. Zeros at $z = \pm 1$ are their own reciprocal and real zeros not on the unit circle occur in reciprocal pairs. Complex zeros on the unit circle are conveniently grouped in pairs. Complex zeros not on the unit circle occur in groups of four. Therefore, the filter can be implemented as a cascade of first-, second-, and fourth-order filters using the array-processor schemes obtained in Section 4.5.

4.8 Conclusions

Array processors for FIR filters and linear-phase FIR filters have been obtained using the signal flow graph approach. The method was employed to obtain an FIR array-processor scheme in which the outputs are localized in separate processing elements. The method was also employed to obtain array structures for odd-length linear-phase FIR filters. Even-length linear-phase filter structures can be obtained in a similar way using the same PEs as for the odd-length filter arrays. All the implementations obtained are modular and regular.

Chapter 5

Design of Fixed-Point Processors for DSP Applications

5.1 Introduction

Multiplier and adder delays are the dominant factors that determine the speed of the array processor structure. Minimizing these delays results in high-speed array processors suited for high-speed digital signal processing applications. In this chapter, two special fixed-point processor modules based on the parallel multiplier design [71, 72] are presented. The first processor module performs the inner-product operation. The processor enhances the speed of operation with a slight increase in area. It can be incorporated in digital filter implementations including the LP scheme of Section 4.4 [94, 95]. The second processor module performs an add-multiply-accumulate operation in the same time as a simple multiplier. The module enhances the speed of operation of linear-phase FIR filter implementations without incurring extra silicon area or introducing extra latency to the system [91]. Both designs would improve the noise performance of the system since double-precision word is assigned for the output without incurring extra communication overhead.

5.2 Design of Inner-Product Processors

The conventional way to obtain an inner-product is to use a multiplier followed by an accumulator where the result of each multiplication is added to the previous result stored in the accumulator. Further increase in speed can be achieved by eliminating the need for the separate adder which also leads to a reduction in the required area [73]-[74]. By using the array-multiplier as a multiplier and adder in the same time, the multiplication and the addition operations required for each inner-product step can be executed during the same clock cycle. In this case, the modified scheme of the array-multiplier is called an accumulator-multiplier (AM). The schematic diagram of the AM module proposed in [73, 74] is shown in Fig. 5.1(a). An inner product of length N can be executed in only N AM cycles. The overall inner-product operation speed is governed by the AM speed. In the AM module, the most significant b-bits of the $2b$-bit product are formed by the carry-propagate adder (carry look-ahead can be also used) at the bottom of the AM module. The adder delay is a significant part of the multiplier delay. Therefore, the adder delay influences the overall inner-product operation speed.

5.2.1 Proposed Carry-Save Inner-Product Processor

Figure 5.1(b) shows the schematic diagram of the proposed inner-product processor. The details of the new inner-product processor are shown in Fig. 5.2 which is based on a modification of the AM module. The inputs to the carry-propagate adder at the bottom $\{q_0^i, \ldots, q_{b-1}^i\}$ and $\{c_0^i, \ldots, c_{b-1}^i\}$, and the least significant byte (LSB) $\{y_0^i, \ldots, y_{b-1}^i\}$ of the previous output product are fed back to the corresponding binary weights. In order to achieve that feedback, one column of full

adders has been added to the AM. In this new scheme, the addition operation is performed concurrently with the multiplication operation without using the adder. This carry-propagate (or carry look-ahead) adder will be used only after all inner product steps are performed in order to get the most significant byte (MSB) of the final result $\{y_b^N, \ldots, y_{2b-1}^N\}$. To initiate the computation of the inner-product, the output registers are reset, the first multiplication is performed, and the results $\{y_0^1, \ldots, y_{b-1}^1\}$, $\{q_0^1, \ldots, q_{b-1}^1\}$, and $\{c_0^1, \ldots, c_{b-1}^1\}$ are stored in the output registers. At the next clock cycle, the bits of the new two operands are input to the multiplier along with the bits stored in the output registers. This process is repeated N times (N is the length of the inner-product operation). An extra cycle is needed to get the MSB of the final result $\{y_b^N, \ldots, y_{2b-1}^N\}$ by adding the words $\{q_0^N, \ldots, q_{b-1}^N\}$ and $\{c_0^N, \ldots, c_{b-1}^N\}$ through the carry-propagate (or carry look-ahead) adder. The two operands of the new inner-product can be loaded simultaneously to the processor during this extra cycle. The scheme is still highly regular and thus very amenable for VLSI implementation. The new scheme of Fig. 5.2 is called carry-save inner-product processor (CSIPP). The overflow in the CSIPP module can be detected as in other available 2's-complement inner-product processors [96].

5.2.2 Performance Analysis

In this section, delay and area comparisons of the AM module and the CSIPP module are presented.

The delay for the AM module to perform an inner product of length N is given

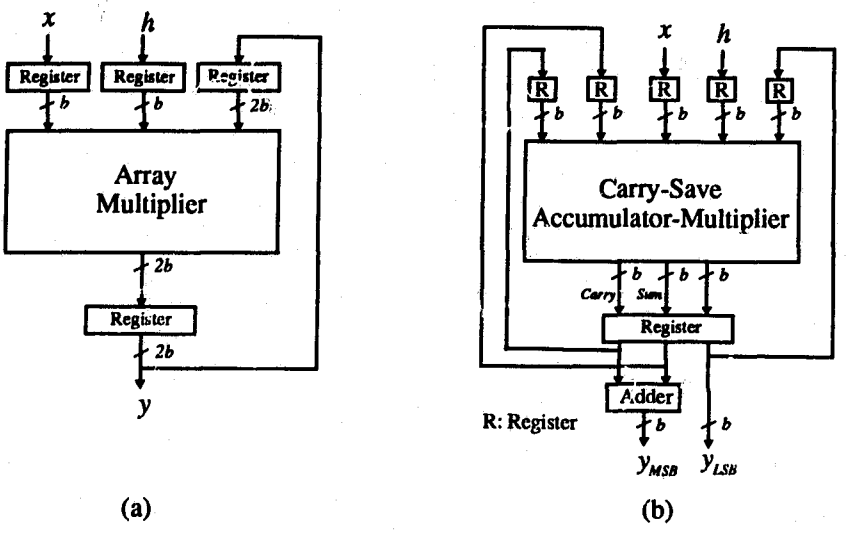


Figure 5.1: (a) Accumulator-multiplier module. (b) Carry-save inner-product processor.

by

$$T_{AM} \approx N[T_N + T_{HA} + (2b-1)T_{FA} + T_{inv} + T_r] \tag{5.1}$$

where T_{FA}, T_{HA}, T_N, and T_{inv}, are the propagation delays for a 1-bit full-adder, a 1-bit half-adder, a 2-input NAND gate, and a 1-bit inverter, respectively. T_r is the setup plus delay times of a 1-bit register. The delay for the CSIPP is given by

$$T_{CSIPP} \approx (N+1)(bT_{FA} + T_r) \tag{5.2}$$

The area required by the AM module can be estimated as

$$\begin{aligned} A_{AM} \approx\ & b^2 A_{FA} + bA_{HA} + [(b-1)^2 + 1]A_A \\ & +2(b-1)A_N + A_{inv} + 2bA_r \end{aligned} \tag{5.3}$$

where A_{FA}, A_{HA}, A_A, A_N, A_{inv}, and A_r are the layout areas for a 1-bit full-adder, a 1-bit half-adder, a 2-input AND gate, a 2-input NAND gate, a 1-bit inverter, and

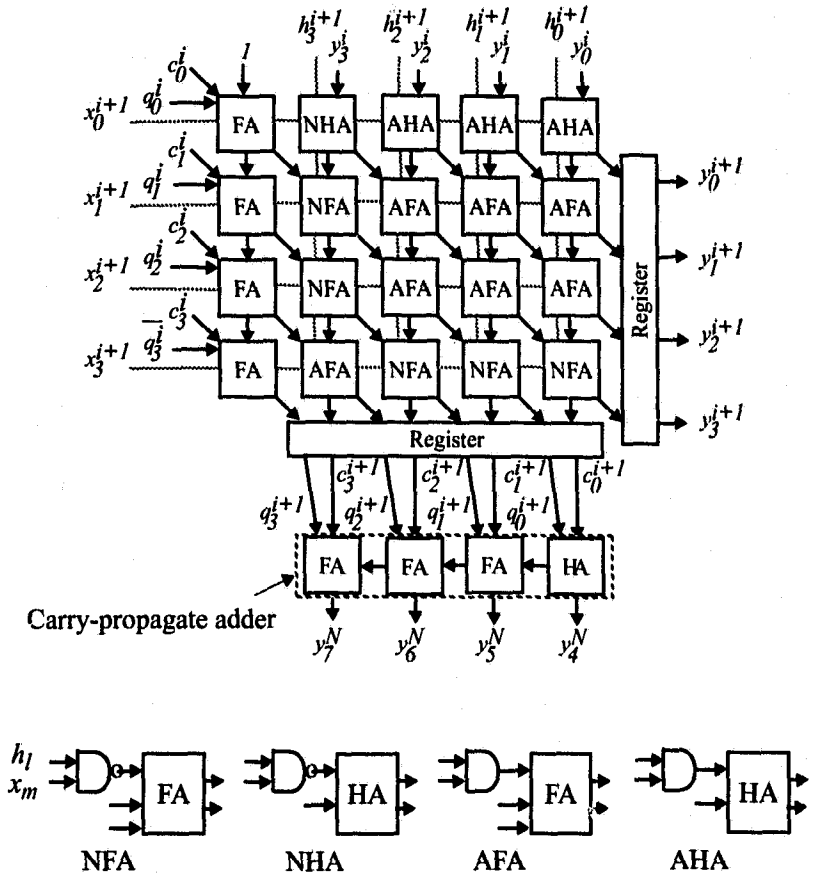


Figure 5.2: Details of a 4×4 2's-complement CSIPP module.

a 1-bit register, respectively. Also, the area required by the CSIPP module can be estimated as

$$\begin{aligned} A_{CSIPP} \approx\ & (b^2+b-1)A_{FA} + (b+1)A_{HA} \\ & +[(b-1)^2+1]A_A + 2(b-1)A_N \\ & +A_{inv} + 3bA_r \end{aligned} \tag{5.4}$$

The areas required for routing are assumed to be equal in the two processors and

hence are omitted from the area calculations for simplification.

It can be seen that the above computations depend on several factors such as the implementation technology and the layout style. In order to compare the delay and the required area for the above two designs, some normalization is carried out. All delays and areas are normalized relative to the inverter delay and area based on the following assumptions [93, 96]:

1. $T_N \approx T_{inv}$, $A_N \approx 2A_{inv}$
2. $T_A \approx 2T_{inv}$, $A_A \approx 3A_{inv}$
3. $T_{HA} \approx 5T_{inv}$, $A_{HA} \approx 10A_{inv}$
4. $T_{FA} \approx 6T_{inv}$, $A_{FA} \approx 15A_{inv}$
5. $T_r \approx 4T_{inv}$, $A_r \approx 12A_{inv}$

The normalized delay and area are listed in Table 5.1. As can be seen, the increase in speed of the CSIPP module over the AM module can be expressed in terms of speedup gain Sg defined as

$$S_g = \frac{T_{AM}}{T_{CSIPP}} = \frac{N(12b+5)}{(N+1)(6b+4)} \tag{5.5}$$

Figure 5.3 shows the speedup gain versus the data wordlength for different inner-product lengths. Figure 5.4(a) and (b) shows normalized areas and area $\times$ time performances for the two designs. From Figs. 5.3-5.4, it is evident that the proposed design enhances the speed of operation without a significant increase in the required area.

Table 5.1: Normalized delay and area computations.

Module	T/T_{inv}	A/A_{inv}
AM	$N(12b+5)$	$18b^2+32b+3$
CSIPP	$(N+1)(6b+4)$	$18b^2+57b-2$

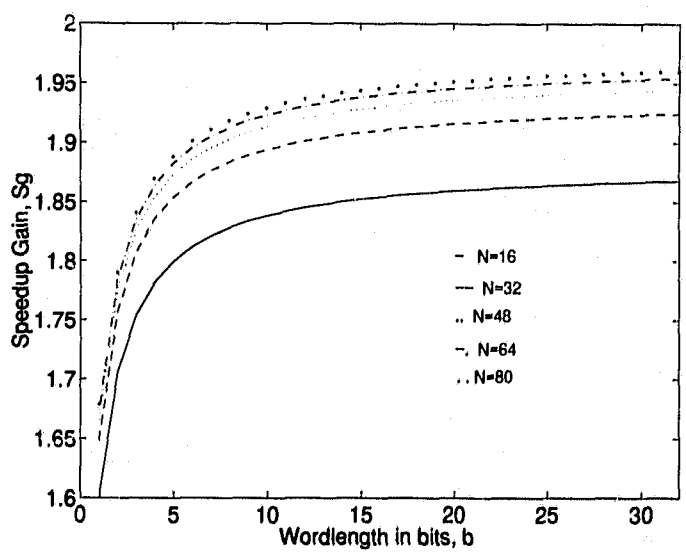
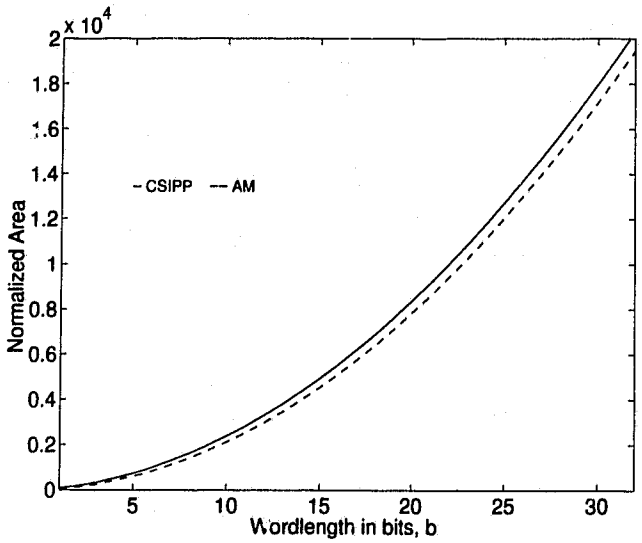
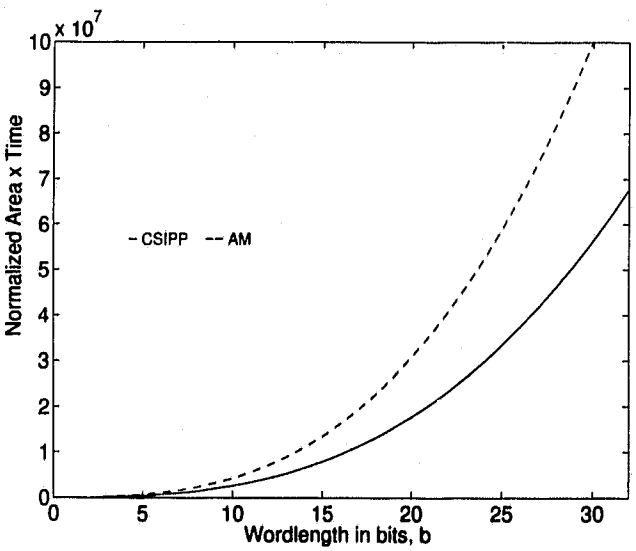


Figure 5.3: Speedup gain versus wordlength.



(a)



(b)

Figure 5.4: Area required and area $\times$ time performances versus wordlength for $N = 16$. (a) Normalized area. (b) Normalized area $\times$ time performance.

5.3 Design of Adder-Multiplier-Accumulator Processors

In this section, a hardware improvement to increase the processing rate in linear-phase FIR filter implementations is presented. The improved processor that will be discussed is based on the AM processor reported in [54, 73].

5.3.1 Proposed Merged-Operand Module

Figure 5.5(a) shows the hardware details of each PE for the array implementation schemes discussed in Section 4.5. We refer to this conventional design as an adder-multiplier-accumulator (AMA) module. The delay in each PE is the sum of two adder delays and one multiplier delay. One attempt to reduce the PE delay is to use an adder-accumulator-multiplier (AAM) module as in Fig. 5.5(b) in which an adder is followed by the AM processor module reported in [54, 73]. It is possible, however, to reduce the PE delay to just the multiplier delay without incurring extra hardware penalties by performing the addition and multiplication operations in the same time. Figure 5.5(c) shows the new merged-operand (MO) module that achieves this performance improvement. The details of a 4×4 MO module are illustrated in Fig. 5.6. It is noted that the addition of the two operands x and w is performed simultaneously with the multiplication operation.

5.3.2 Performance Analysis

In the following, the delay and the area of the MO module will be compared with those of the AMA and AAM modules. The propagation delay of the standard design

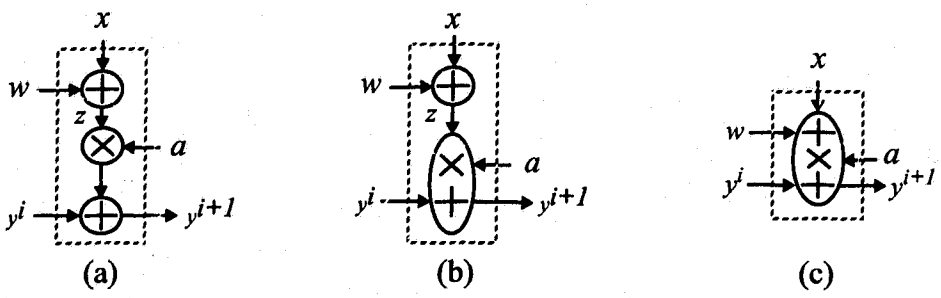


Figure 5.5: (a) Adder-multiplier-accumulator (AMA) module. (b) Adder-accumulator-multiplier (AAM) module. (c) Merged-operand (MO) module.

of Fig. 5.5(a) is given by

$$T_{AMA} \approx T_{add-sp} + T_{mult} + T_{add-dp} + T_r \tag{5.6}$$

Alternatively, the above formula can be expressed as

$$T_{AMA} \approx 5(b-1)T_{FA} + 3T_{HA} + T_A + T_{inv} + T_r \tag{5.7}$$

Similarly, the propagation delay of the design of Fig. 5.5(b) is given by

$$T_{AAM} \approx (3b-2)T_{FA} + 2T_{HA} + T_N + T_{inv} + T_r \tag{5.8}$$

The propagation delay of the MO module is determined by the time required for the longest path indicated by the dotted line in Fig. 5.6. It is given by

$$T_{MO} \approx (2b-1)T_{FA} + 2T_{HA} + T_N + T_{inv} + T_r \tag{5.9}$$

The area required by the design of Fig. 5.5(a) can be expressed as

$$\begin{aligned} A_{AMA} \approx\ & [b^2+b-1]A_{FA} + (b+1)A_{HA} \\ & +[(b-1)^2+1]A_A + 2(b-1)A_N \\ & +A_{inv} + 2bA_r \end{aligned} \tag{5.10}$$

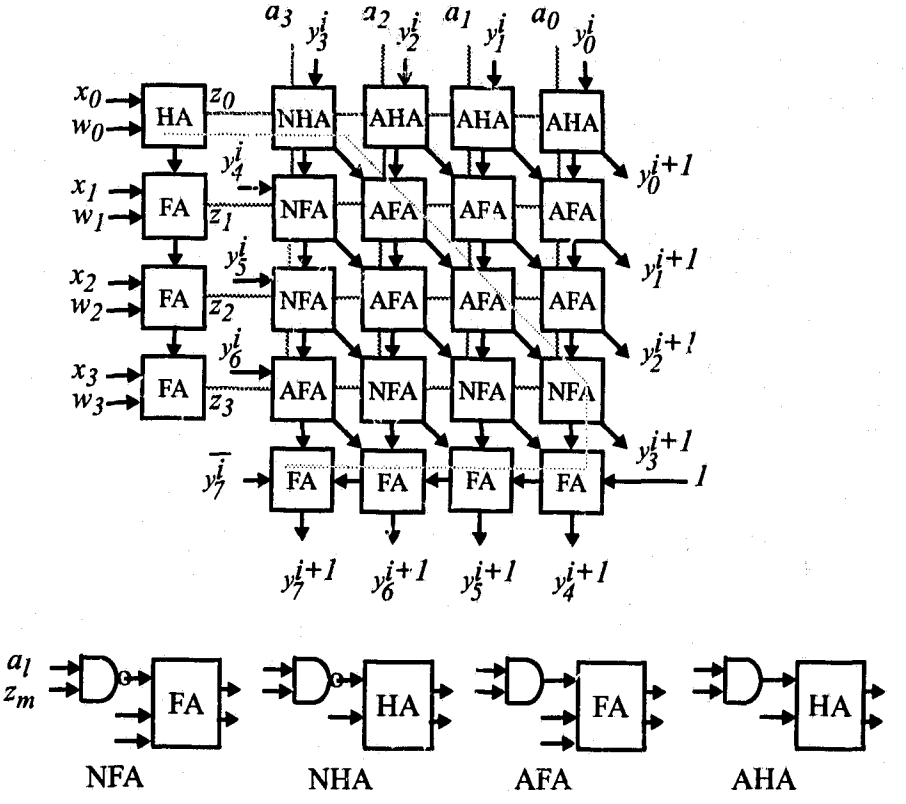


Figure 5.6: Details of a 4×4 2's-complement Merged-operand (MO) module.

It is easy to verify that the areas of the AAM and MO modules are approximately equal to that of the AMA module; hence they can be represented by the above formula. It is noted that in deriving the above relations, we assumed the routing areas to be the same for the three modules.

It can be seen that the above computations depend on several factors such as the implementation technology and the layout style. In order to compare the delay performance of the above designs, the normalization in Section 5.2.2 is carried out.

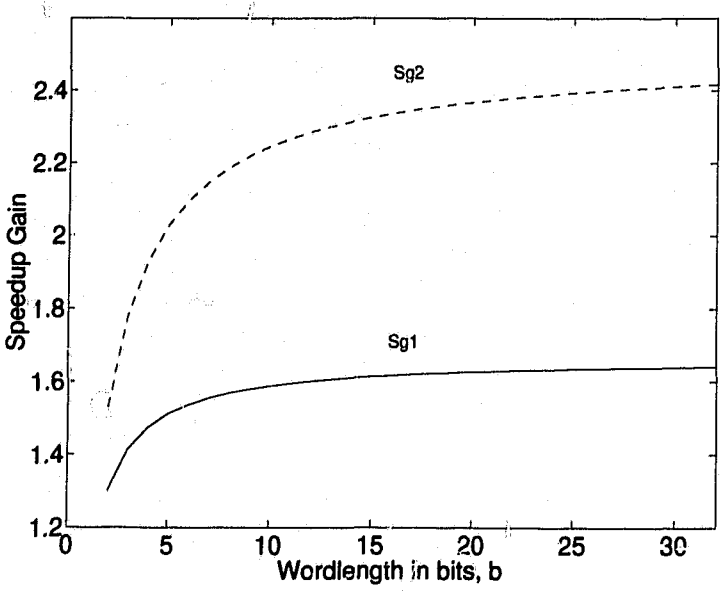


Figure 5.7: Speedup gain versus wordlength.

Referring to (5.7)-(5.9), the increase in speed of the AAM and the MO modules over the AMA module can be expressed in terms of speedup gains S_{g1} and S_{g2} defined as

$$S_{g1} = \frac{T_{AMA}}{T_{AAM}} = \frac{30b-8}{18b+4}, \quad S_{g2} = \frac{T_{AMA}}{T_{MO}} = \frac{30b-8}{12b+10} \tag{5.11}$$

Figure 5.7 shows the two speedup gains versus the data wordlength. It is evident that the proposed design offers the most improvement.

Generally, the MO module can be incorporated in the linear-phase filter implementations of Chapter 4 referred to as schemes I and II to perform an add-multiply-accumulate operation as well as in scheme III to perform an inner-product operation.

5.4 Conclusions

Two fixed-point processors for DSP applications have been presented. A new inner-product processor has been presented which enhances the speed of operation of FIR filter implementations with a slight increase in the area. A new processor module to perform an add-multiply-accumulate operation has been presented which enhances the speed of operation of the linear-phase FIR implementations of Chapter 4 without increasing the silicon area or introducing extra latency in the system.

Chapter 6

VLSI Array Processors for Filter Banks

6.1 Introduction

In this chapter, new efficient array-processor structures for filter banks are obtained. This is achieved by mapping decimator and interpolator algorithms onto hardware structures using an algebraic mapping methodology. Two approaches are employed to obtain several array-processor structures for decimators and interpolators. In the first approach, FIR/IIR decimator and interpolator array structures are obtained based on their polyphase structures. In the second approach FIR decimator and interpolator array structures are obtained based on their direct-form structures. The structures obtained are modular, regular, and some of them are fully pipelined. Upper bounds on the input/output processing rates are deduced in terms of system parameters and hardware delays.

6.2 Multirate Systems, Descriptions and Implementations

6.2.1 Algebraic representations and control signals

The sampling rate of a signal $x(nT)$, where T is the sampling period, can be reduced by an integer factor M by defining the new signal

$$x_d(nT') = x(nMT) \tag{6.1}$$

where $T' = MT$. This process is referred to as downsampling [39]. A downsampler or a compressor is often represented by the symbol depicted in Fig. 6.1(a). Although such a representation is attractive for its simplicity, it proves inadequate for algebraically describing multirate systems. This inadequacy results from the implicit assumption that some control signal is available to maintain the correct operation of the downsampler. Figure 6.1(b) illustrates the representation of the compressor as a switch explicitly controlled by a signal $c(nT')$. The control signal $c(nT')$ is periodic with a period $T' = MT$. When the control signal becomes high, the input is sampled. Figure 6.1(c) illustrates the representation of the downsampling operation in terms of the relation

$$\mathcal{D}[x(nT)] = x_d(nT') \tag{6.2}$$

where $\mathcal{D}$ is an operator. The sampling rate of a signal $x(nT)$ can be increased by an integer factor L by using the upsampler or expander of Fig. 6.2(a) and its response to an excitation $x(nT)$ can be expressed as

$$x_u(nT'') = \begin{cases} x(\frac{nT}{L}) & \text{if } n \text{ is a multiple of } L \\ 0 & \text{otherwise} \end{cases} \tag{6.3}$$

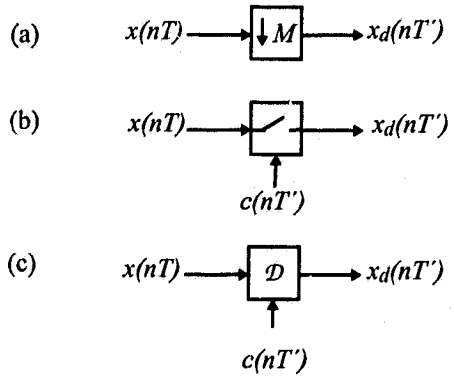


Figure 6.1: Representation of compressor.

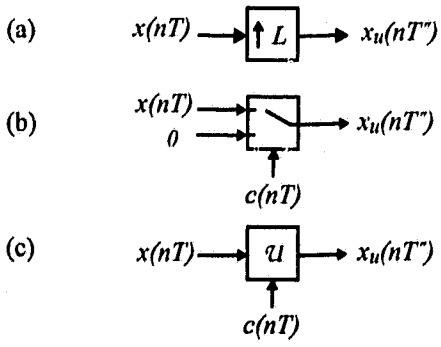


Figure 6.2: Representation of expander.

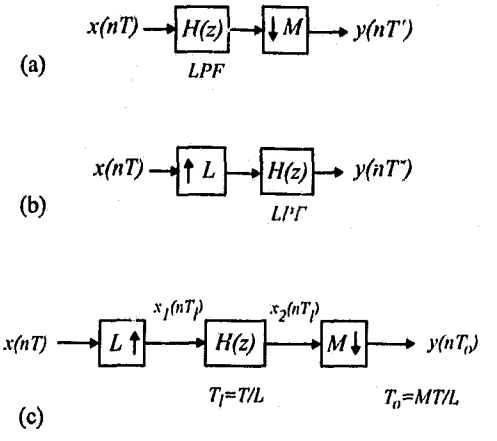


Figure 6.3: General representation of (a) the decimator, (b) the interpolator, and (c) rational sampling rate conversion system.

where $T'' = T/L$. Figure 6.2(b) shows the representation of the expander as a single-pole, double-throw switch explicitly controlled by the signal $c(nT)$. The control signal $c(nT)$ is periodic with a period $T = LT''$. The signal is passed if $c(nT)$ equals 1 and a zero output is produced otherwise. Figure 6.2(c) illustrates the representation of the upsampling operation in terms of the relation

$$\mathcal{U}[x(nT)] = x_u(nT'') \tag{6.4}$$

where $\mathcal{U}$ is an operator.

To eliminate aliasing for decimation and images for interpolation, lowpass filters are incorporated as shown in Fig. 6.3(a) and (b) [39].

By combining decimation and interpolation, it is possible to change the sampling rate by a non-integer factor. The system in Fig. 6.3(c) converts the input signal by: (a) increasing its sampling rate by a factor L, (b) filtering $x_1(nT_I)$ to eliminate the signal images by a standard linear time-invariant lowpass filter of transfer function

$H(z)$ to give $x_2(nT_l)$, and (c) compressing the sampling rate of $x_2(nT_l)$ by a factor M. If $M > L$ there is a decrease in the sampling rate and if $M < L$, the opposite is true.

6.2.2 Polyphase structures

The polyphase structure of a type 1 M-to-1 decimator is shown in Fig. 6.4(a) [27, 63]. The subfilters represented by the transfer functions $P_i(z)$ in the parallel branches are referred to as polyphase filters and their impulse responses are related to that of a prototype filter by the relations

$$p_i(nT') = h[(nM + i)T] \quad \text{for } 0 \leq i \leq M - 1 \text{ and all } n \tag{6.5}$$

where $h(nT)$ is the impulse response of either an FIR or an IIR filter. The transfer function of the prototype filter can be written as

$$H(z) = \sum_{i=0}^{M-1} z^{-i} P_i(z^M) \tag{6.6}$$

where

$$P_i(z) = \sum_{n=-\infty}^{\infty} p_i(nT')z^{-n}$$

For IIR filters, the polyphase transfer functions share the same denominator polynomial [63], i.e.,

$$P_i(z) = \frac{N_i(z)}{\hat{D}(z)} \quad \text{for } i = 0, 1, \ldots, M - 1 \tag{6.7}$$

Let

$$H(z) = \frac{N(z)}{D(z)} = \frac{\sum_{k=0}^{N-1} b_k z^{-k}}{1 - \sum_{r=1}^{R} a_{rM} z^{-rM}} \tag{6.8}$$

and

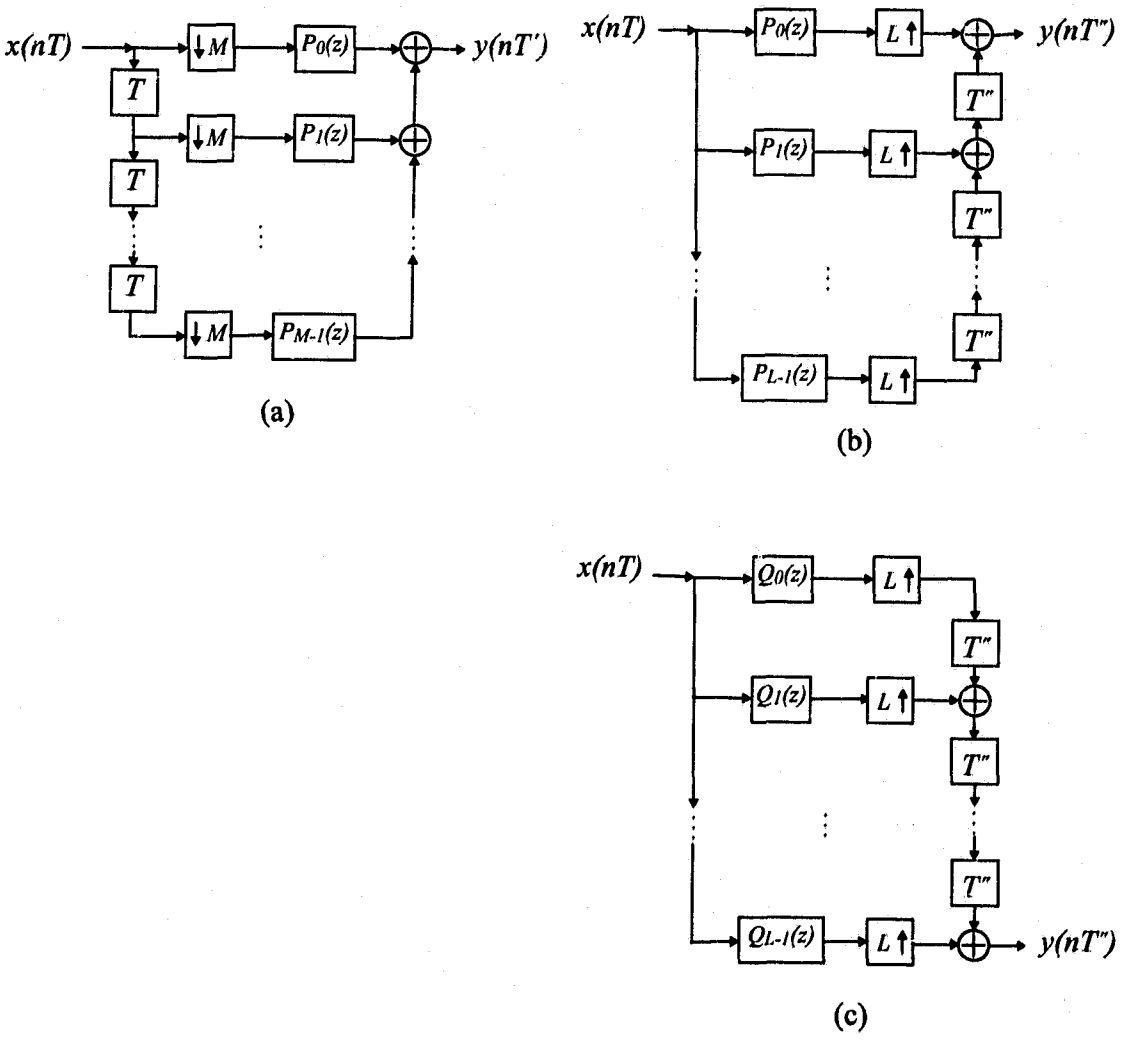


Figure 6.4: Polyphase representations of decimtors and interpolators. (a) Type 1 polyphase structure for an M-to-1 decimator. (b) Type 1 polyphase structure for a 1-to-L interpolator. (c) Type 2 Polyphase structure for a 1-to-L interpolator.

$$P_i(z) = \frac{\sum_{k=0}^{n_i-1} d_{k,i} z^{-k}}{1 - \sum_{r=1}^{R} c_r z^{-r}}, \quad \text{for } i = 0, 1, \ldots, M-1 \tag{6.9}$$

where $n_i - 1$ is the order of the numerator and R is the order of the denominator. The relations between the coefficients of $P_i(z)$ and those of $H(z)$ are [63]

$$c_r = a_{rM} \quad \text{for } r = 1, 2, \ldots, R$$

$$d_{k,i} = b_{kM+i} \quad \text{for } k = 0, 1, \ldots, n_i - 1;\ i = 0, 1, \ldots, M-1$$

By transposing the structure of Fig. 6.4(a), the polyphase 1-to-L interpolator structure of Fig. 6.4(b) is produced [63]. A type 2 polyphase representation for a 1-to-L interpolator is shown in Fig. 6.4(c) where $Q_i(z) = P_{L-1-i}(z)$ [27].

6.3 Algebraic Mapping Methodology

The algebraic approach for mapping decimator/interpolator algorithms onto array processing hardware can be summarized in the following rules:

- The input and the control signals are considered as inputs to the decimator/interpolator system. As a consequence, introducing a certain time shift to one of them implies introducing the same time shift to the other signal to preserve their timing interrelationship.
- The decimator/interpolator algorithm is converted into a polynomial involving the input samples and delay operators. By using certain polynomial evaluation techniques, the decimator or interpolator polynomial is converted into a set of recursive expressions.
- Each iteration in the set of recursive expressions is assigned a PE.

- The right-hand side of each expression defines the operations to be performed in terms of the input variables.
- The left-hand side of each expression defines the corresponding processor output.
- The number of delay operators attached to the recursive variables dictates the size of the buffers between or within the processors.
- The number of PEs is determined by the number of iterations required to produce the final result.

The characteristics of the array-processor structures obtained can be deduced from the iterative equations according to the following observations:

- Processor computational load is dictated by the nature of the operations on the right-hand side of each iterative formula.
- Chip area is determined by the number and complexity of each of the PEs.
- Communication requirements are determined by studying the source of the data used on the right-hand side of each iterative formula.

By following the above observations, different array structures can be derived as described below [97]-[100]. Time domain is employed which is more efficient in describing multirate systems than the z domain [39].

6.4 VLSI Array Processors for Polyphase Decimators and Interpolators

In this section, array-processor implementations of polyphase decimators and interpolators with integer compression/expansion factors are derived using the algebraic mapping methodology where the prefixes PD and PI are used to denote a polyphase decimator and a polyphase interpolator, respectively. The work is then extended to the mapping of fractional decimator/interpolator algorithms onto systolic hardware which is required for nonuniform filter banks with fractional compression/expansion factors [51].

6.4.1 Array-Processor Implementation of Decimators

By replacing the compressor blocks in Fig. 6.4(a) by their equivalent representation of Fig. 6.1(c), the algebraic description of the decimator can be expressed by the relation

$$y(nT') = \sum_{i=0}^{M-1} \mathcal{D}[x(nT - iT)] * p_i(nT') \tag{6.10}$$

where $x(nT - iT)$ is the input signal to the ith polyphase filter represented by the transfer function $P_i(z)$ (or by the impulse response $p_i(nT')$), and $*$ is the convolution operation. Using the shift operator $\mathcal{E}$ for the input sampling period T, (6.10) can be written as

$$y(nT') = \sum_{i=0}^{M-1} \mathcal{D}\{\mathcal{E}^{-i}[x(nT)]\} * p_i(nT') \tag{6.11}$$

If a processor array is to evaluate (6.11), then different PEs could be assigned to each term at the right-hand side. This hardware implementation results in a processor

array in which all inputs and control signals are broadcast and the input buffers are not equal.

Scheme PD-I

The input buffers can be made of equal size if the input signal can be propagated between processors using the intermediate variable $x_i(nT)$ given by

$$\begin{aligned} x_i(nT) &= \mathcal{E}^{-1}x_{i-1}(nT) \qquad \text{for } i = 1, 2, \cdots, M-1 \\ x_0(nT) &= x(nT) \end{aligned} \tag{6.12}$$

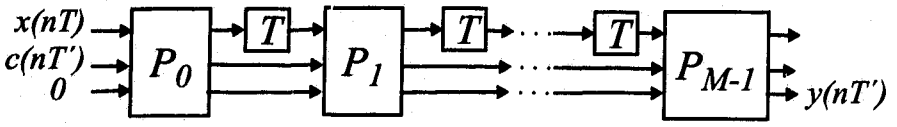
Thus (6.11) becomes

$$\begin{aligned} y(nT') = \mathcal{D}[x_0(nT)] * p_0(nT') + \mathcal{D}[x_1(nT)] * p_1(nT') + \cdots \\ + \mathcal{D}[x_{M-1}(nT)] * p_{M-1}(nT') \end{aligned} \tag{6.13}$$

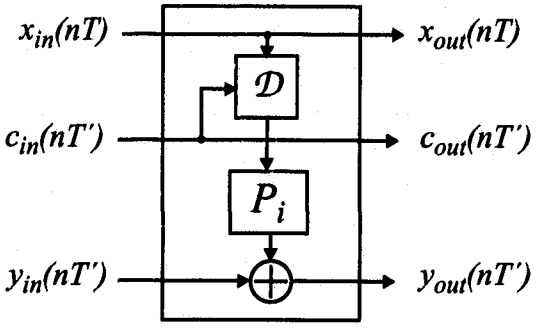
If each term is assigned to a processing element, then from (6.13) it can be seen that the input is propagated from one PE to the next. Furthermore, the input buffers of all PEs are of equal length. The filter output is produced after all the partial products have been added. The above equation can be written in the recursive form

$$\begin{aligned} y_i(nT') &= \{\mathcal{D}[x_i(nT)] * p_i(nT')\} + y_{i-1}(nT') \qquad \text{for } 0 \le i \le M-1 \qquad (6.14) \\ y_{-1}(nT') &= 0 \\ y(nT') &= y_{M-1}(nT') \end{aligned}$$

Figure 6.5(a) shows the mapping of (6.14) while Fig. 6.5(b) shows the details of the PE involved. The polyphase filters represented by the transfer functions $P_i(z)$ can be implemented using the array structures reported in Chapter 4 and Refs. [55, 56, 57, 89, 91]. It is noted that the resulting structure is identical to the



(a)



(b)

Figure 6.5: Scheme PD-I decimator structure. (a) Mapping of (6.14) onto an array-processor structure. (b) Details of PE involved.

basic decimator structure of Fig. 6.4(a). However, Fig. 6.5(a) explicitly shows the distribution of the control signals required for the correct operations of the system.

Scheme PD-II

Polyphase decimators are linear periodically time-variant systems. Thus, it is possible to delay the outputs and the inputs by integer multiples of times T' and MT, respectively, without affecting the system response. One can achieve a systolic struc-

ture in which the input, the control and output signals are pipelined by delaying both inputs and output by time $(M-1)T'$. Thus, (6.11) can be written in the form

$$\mathcal{E}_1^{-(M-1)} y(nT') = \mathcal{E}_1^{-(M-1)} \sum_{i=0}^{M-1} \mathcal{D}\{\mathcal{E}^{-i}[x(nT)]\} * p_i(nT') \tag{6.15}$$

where $\mathcal{E}_1$ is the shift operator for the output sampling period $T' = MT$. The above equation can be written in the form

$$\widehat{y}(nT') = \sum_{i=0}^{M-1} \mathcal{E}_1^{-(M-i-1)} \{\mathcal{D}[\mathcal{E}^{-i(M+1)} x(nT)] * p_i(nT')\} \tag{6.16}$$

where $\widehat{y}(nT') = \mathcal{E}_1^{-(M-1)} y(nT')$ represents the delayed system output. It is noted that the filter input as well as the associated control signal $c(nT')$ are both to be delayed by time iT'. Introducing the intermediate variable $x_i(nT)$ given by

$$\begin{aligned} x_i(nT) &= \mathcal{E}^{-(M+1)} x_{i-1}(nT) \quad \text{for } 1 \le i \le M-1 \\ x_0(nT) &= x(nT) \end{aligned} \tag{6.17}$$

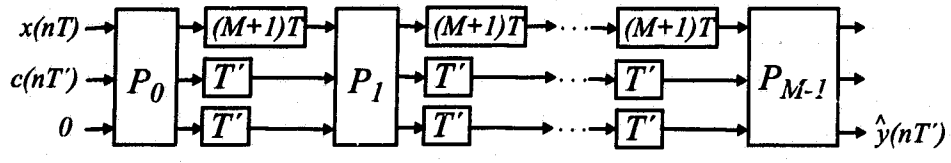
Equation (6.16) can be written in a recursive form as

$$\begin{aligned} \widehat{y}_i(nT') &= \mathcal{E}_1^{-1} \widehat{y}_{i-1}(nT') + \mathcal{D}[x_i(nT)] * p_i(nT') \quad \text{for } 0 < i \le M-1 \\ \widehat{y}_0(nT') &= \widehat{y}_{-1}(nT') + \mathcal{D}[x_0(nT)] * p_0(nT') \\ \widehat{y}_{-1}(nT') &= 0 \\ \widehat{y}(nT') &= \widehat{y}_{M-1}(nT') \end{aligned} \tag{6.18}$$

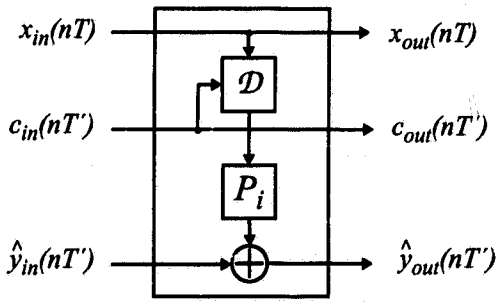
and the control signal recursions can be written in the form

$$\begin{aligned} c_i(nT') &= \mathcal{E}_1^{-1} c_{i-1}(nT') \quad \text{for } 1 \le i \le M-1 \\ c_0(nT') &= c(nT') \end{aligned} \tag{6.19}$$

The mapping of (6.18) onto a systolic structure is shown in Fig. 6.6(a) while Fig. 6.6(b) shows the details of the PE involved.



(a)



(b)

Figure 6.6: Scheme PD-II decimator structure. (a) Mapping of (6.18) onto a systolic structure. (b) Details of PE involved.

Scheme PD-III

A systolic scheme in which the input signal is broadcast, the control signal, and the output signal are pipelined can be derived. Exchanging the downsampling and delay operators, equation (6.11) can be written in the form

$$y(nT') = \sum_{i=0}^{M-1} \mathcal{E}^{-i}\{\mathcal{D}[x(nT)] * p_i(nT')\} \tag{6.20}$$

Applying Horner's rule gives

$$\begin{aligned} y(nT') = {} & \mathcal{D}[x(nT)] * p_0(nT') \\ & + \mathcal{E}^{-1}\{[\mathcal{D}[x(nT)] * p_1(nT')] + \cdots \\ & + \mathcal{E}^{-1}\{[\mathcal{D}[x(nT)] * p_{M-1}(nT')]\}\cdots\} \end{aligned} \tag{6.21}$$

The above equation can be put in the recursive form

$$\begin{aligned} y_i(nT') &= \mathcal{D}[x(nT)] * p_i(nT') + \mathcal{E}^{-1} y_{i+1}(nT') \quad \text{for } 0 \le i < M-1 \quad (6.22) \\ y_{M-1}(nT') &= \mathcal{D}[x(nT)] * p_{M-1}(nT') + y_M(nT') \\ y_M(nT') &= 0 \\ y(nT') &= y_0(nT') \end{aligned}$$

The mapping of (6.22) onto an array structure is shown in Fig. 6.7(a) while Fig. 6.7(b) shows the details of the PE involved.

It is noted that the control signal is governed by the recursion

$$\begin{aligned} c_i(nT') &= \mathcal{E}^{-1} c_{i+1}(nT') \quad \text{for } 0 \le i < M-1 \quad (6.23) \\ c_{M-1}(nT') &= c(nT') \end{aligned}$$

to preserve the timing interrelationship with the input signal. Such an implementation requires special and elaborate techniques for control and signal timing as the output signal has to be delayed by a fraction of the output sampling period.

6.4.2 Array-Processor Implementation of Interpolators

By replacing the expander blocks in Fig. 6.4(b) by their equivalent representation of Fig. 6.2(c), the description of the interpolator can be expressed by the relation

$$y(nT'') = \sum_{i=0}^{L-1} \mathcal{E}_2^{-i} \mathcal{U}[x(nT) * p_i(nT)] \quad (6.24)$$

where the operator $\mathcal{E}_2$ is the shift operator for the output sampling period $T'' = T/L$.

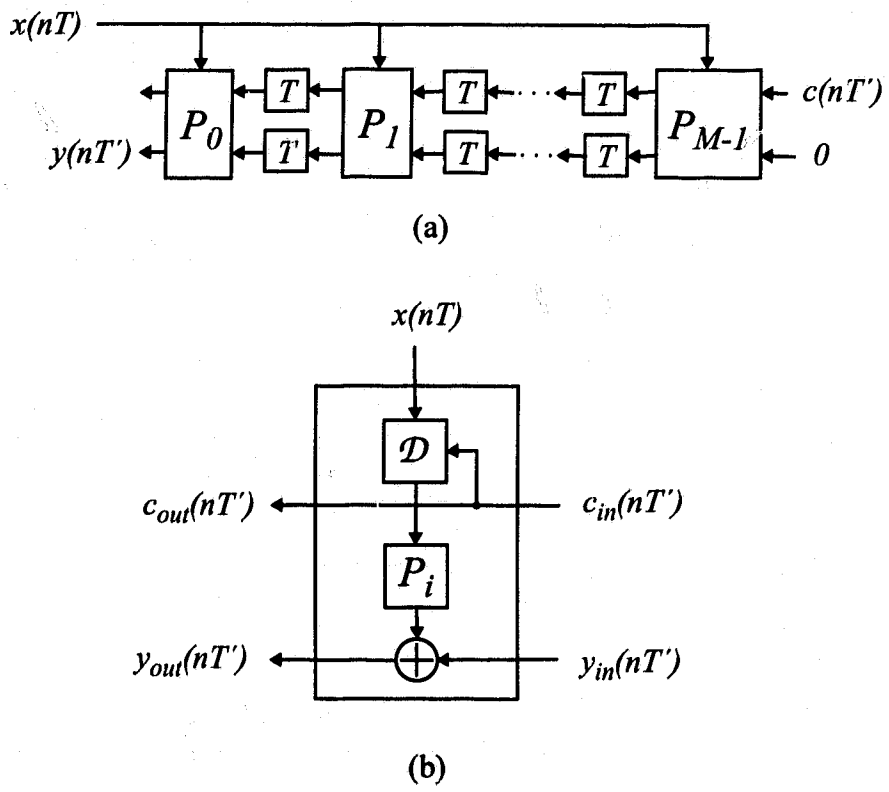


Figure 6.7: Scheme PD-III decimator structure. (a) Mapping of (6.22) onto an array-processor structure. (b) Details of PE involved.

Scheme PI-I

Applying Horner's rule, equation (6.24) can be evaluated in the following sequence [54, 56]

$$y(nT'') = \mathcal{U}[x(nT) * p_0(nT)] + \mathcal{E}_2^{-1}\{\mathcal{U}[x(nT) * p_1(nT)] + \cdots$$
$$+\mathcal{E}_2^{-1}\{\mathcal{U}[x(nT) * p_{L-1}(nT)]\}\cdots\} \qquad (6.25)$$

Equation (6.25) can be written as

$$y_i(nT'') = \mathcal{U}[x(nT) * p_i(nT)] + \mathcal{E}_2^{-1} y_{i+1}(nT'') \quad \text{for } 0 \le i < L-1 \tag{6.26}$$

$$y_{L-1}(nT'') = \mathcal{U}[x(nT) * p_{L-1}(nT)] + y_L(nT'')$$

$$y_L(nT'') = 0$$

$$y(nT'') = y_0(nT'')$$

Each iteration step in (6.26) can now be assigned to a PE. Figure 6.8(a) shows the mapping of (6.26) onto an array structure while Fig. 6.8(b) depicts the details of the PE involved. It is noted that the input and the control signals are broadcast while the output signal is pipelined.

Scheme PI-II

Polyphase interpolators are linear periodically time-variant systems. Thus, it is possible to delay the outputs and the inputs by integer multiples of times T and LT'', respectively, without affecting the system response. One can achieve a systolic structure in which the input and output are pipelined by delaying both inputs and output by time $L(L-1)T''$. Thus, (6.24) can be written in the form

$$\mathcal{E}_2^{-L(L-1)} y(nT'') = \mathcal{E}_2^{-L(L-1)} \sum_{i=0}^{L-1} \mathcal{E}_2^{-i} \mathcal{U}[x(nT) * p_i(nT)] \tag{6.27}$$

which can be written as

$$\hat{y}(nT'') = \sum_{i=0}^{L-1} \mathcal{E}_2^{-i(L+1)} \mathcal{U}\{[\mathcal{E}^{-(L-i-1)} x(nT)] * p_i(nT)\} \tag{6.28}$$

where $\hat{y}(nT'') = \mathcal{E}_2^{-L(L-1)} y(nT'')$ represents the delayed system output. Introducing the intermediate variable $x_i(nT)$ given by

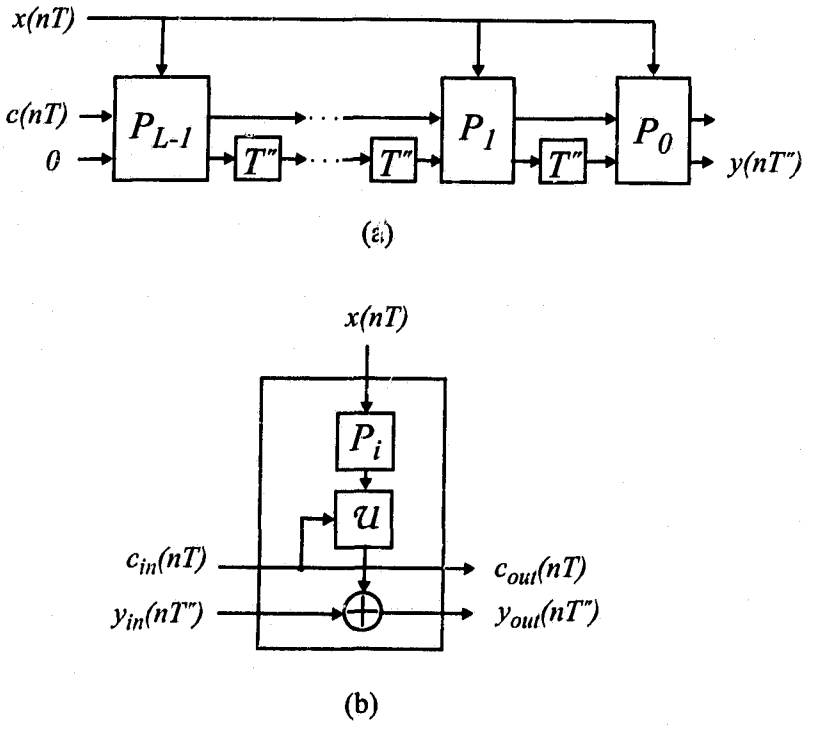


Figure 6.8: Scheme PI-I interpolator structure. (a) Mapping of (6.26) onto an array-processor structure. (b) Details of PE involved.

$$x_i(nT) = \mathcal{E}^{-1}x_{i+1}(nT) \quad \text{for } 0 \le i \le L-2 \tag{6.29}$$

$$x_{L-1}(nT) = x(nT)$$

equation (6.28) can be written in the recursive form

$$\hat{y}_i(nT'') = \mathcal{E}_2^{-(L+1)}\hat{y}_{i+1}(nT'') + \mathcal{U}[x_i(nT) * p_i(nT)] \quad \text{for } 0 \le i < L-1 \tag{6.30}$$

$$\hat{y}_{L-1}(nT'') = \hat{y}_L(nT'') + \mathcal{U}[x_{L-1}(nT) * p_{L-1}(nT)]$$

$$\hat{y}(nT'') = \hat{y}_0(nT'')$$

$$\hat{y}_L(nT'') = 0$$

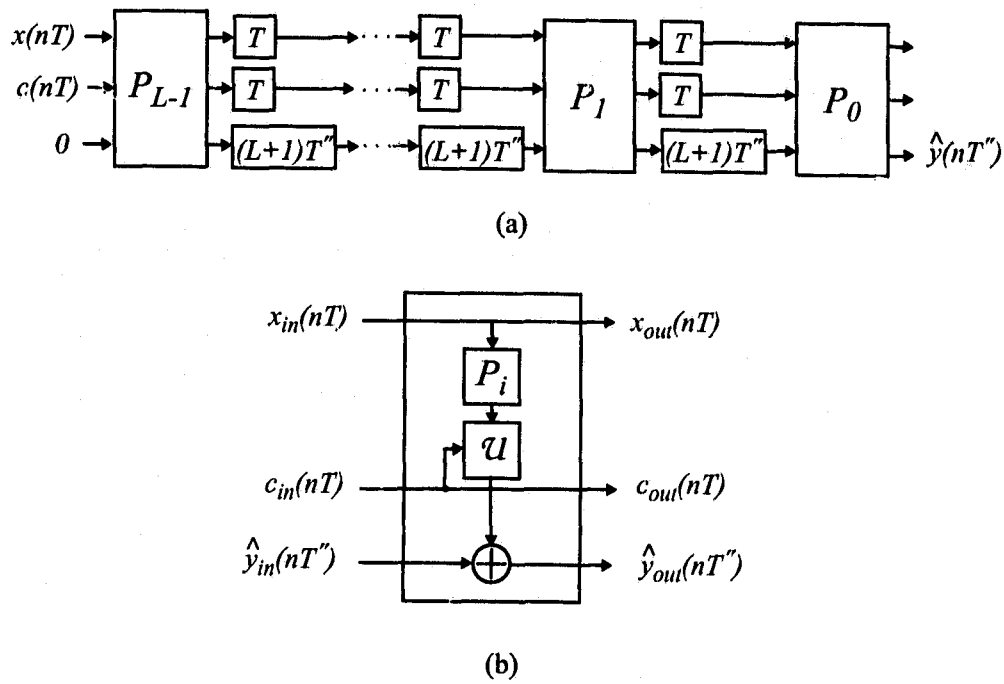


Figure 6.9: Scheme PI-II interpolator structure. (a) Mapping of (6.30) onto a systolic structure. (b) Details of PE involved.

while the recursion for the control signal can be written as

$$c_i(nT) = \mathcal{E}^{-1}c_{i+1}(nT) \quad \text{for } 0 \leq i \leq L-2$$
$$c_{L-1}(nT) = c(nT) \tag{6.31}$$

Relation (6.30) leads to the systolic implementation of Fig. 6.9(a) while Fig. 6.9(b) shows the details of the PE involved.

6.4.3 Alternative Structures for IIR Decimators and Interpolators

Since the polyphase transfer functions of IIR decimators share the same denominator polynomial, the structures of decimators can be made more efficient if the

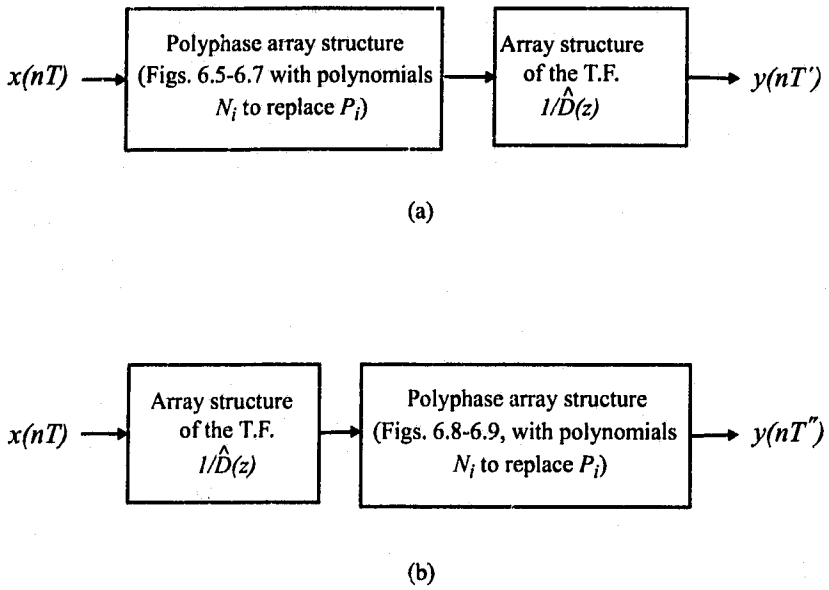


Figure 6.10: Alternative IIR decimator and interpolator structures. (a) Efficient IIR decimator structure. (b) Efficient IIR interpolator structure.

denominator polynomial is realized after the summation . In this case, the transfer functions $N_i(z)$, $i = 0, 1, \ldots, M-1$, which represent FIR filters should replace their counterparts $P_i(z)$ in Figs. 6.5-6.7. The denominator part can be implemented in cascade as shown in Fig. 6.10(a). The array-processor implementation of the cascaded block can be implemented using the implementation in [54, 55] with all the coefficients $a(iT)$, $i \neq 0$, (T is related to sampling rate in [54, 55]) set to zero.

Interpolator array processors can be obtained as in Fig. 6.10(b) where the structure is composed of two parts. The first part is the implementation of the denominator polynomial. The second part is the same as in Figs. 6.8-6.9 with the polynomials $N_i(z)$ replacing $P_i(z)$.

6.4.4 Efficient Descriptions of Fractional Decimators and Interpolators

In the following we will assume that L and M are mutually prime numbers and that $M > L$ although the method is suitable also for the case $L > M$. The system of Fig. 6.3(c) can be described by the relation

$$y(nT_o) = \mathcal{D}\left[\mathcal{U}[x(nT)] * h(nT_l)\right] \tag{6.32}$$

where $h(nT_l)$ is the impulse response of the prototype filter. To obtain an efficient implementation for the system of Fig. 6.3(c), the polyphase representation is applied. The term in the outer brackets of the above relation can be expressed using type 2 polyphase representation of the interpolator and (6.32) can be written as

$$y(nT_o) = \mathcal{D}\{\sum_{j=0}^{L-1} \mathcal{E}_l^{-(L-1-j)} \mathcal{U}[x(nT) * q_j(nT)]\} \tag{6.33}$$

where $q_j(nT)$ for $j = 0, 1, \ldots, L-1$, are the impulse responses of the polyphase filters of type 2 and $\mathcal{E}_l$ is a shift operator related to the sampling period T_l.

Since M and L are mutually prime, the following properties are valid [27, 64, 65]:

1. There exist integers r_0 and r_1 satisfying the relation $-r_0 L + r_1 M = 1$.

2. The compressor/expander blocks can be interchanged.

Applying the first property to (6.33), yields

$$y(nT_o) = \mathcal{D}\{\sum_{j=0}^{L-1} \mathcal{E}_l^{-(L-1-j)r_1 M} \mathcal{E}_l^{(L-1-j)r_0 L} \mathcal{U}[x(nT) * q_j(nT)]\} \tag{6.34}$$

In the following, we will apply the noble identities described in [63] to translate the shifts to the input and output of the system of Fig. 6.3(c). Thus, (6.34) can be written as

$$y(nT_o) = \sum_{j=0}^{L-1} \mathcal{E}_o^{-(L-1-j)r_1} \mathcal{D}\{\mathcal{U}[\mathcal{E}^{(L-1-j)r_0} x(nT) * q_j(nT)]\} \tag{6.35}$$

where $\mathcal{E}_o$ and $\mathcal{E}$ are shift operators related to the output and input sampling periods, respectively. Applying the second property, (6.35) can be written as

$$y(nT_o) = \sum_{j=0}^{L-1} \mathcal{E}_o^{-(L-1-j)r_1} \mathcal{U}\{\mathcal{D}[\mathcal{E}^{(L-1)r_0} \mathcal{E}^{-jr_0} x(nT) * q_j(nT)]\} \tag{6.36}$$

The positive delay in (6.36) can be ignored since, in practice, this positive delay can be compensated by adding an extra delay to the transfer function of the prototype filter. Accordingly, (6.36) is modified as

$$\begin{aligned} y(nT_o) &= \sum_{j=0}^{L-1} \mathcal{E}_o^{-(L-1-j)r_1} \mathcal{U}\{\mathcal{D}[\mathcal{E}^{-jr_0} x(nT) * q_j(nT)]\} \\ &= \sum_{j=0}^{L-1} \mathcal{E}_o^{-(L-1-j)r_1} \mathcal{U}\{\mathcal{D}[x_j(nT) * q_j(nT)]\} \end{aligned} \tag{6.37}$$

where $x_j(nT) = \mathcal{E}^{-jr_0} x(nT)$. Expressing the decimation operation in terms of the polyphase representation, (6.37) can be written as

$$y(nT_o) = \sum_{j=0}^{L-1} \mathcal{E}_o^{-(L-1-j)r_1} \mathcal{U}\{\sum_{i=0}^{M-1} \mathcal{D}[\mathcal{E}^{-i} x_j(nT)] * q_{j,i}(nT')\} \tag{6.38}$$

where $T' = MT$ and $q_{j,i}(nT')$ are the impulse responses of the type 1 polyphase components for the filters of transfer functions $Q_j(z)$ for $i = 0, 1, \ldots, M-1$, and $j = 0, 1, \ldots, L-1$ [27]. Equation (6.38) can be written as

$$y(nT_o) = \sum_{j=0}^{L-1} \mathcal{E}_o^{-(L-1-j)r_1} \mathcal{U}[v_j(nT')] \tag{6.39}$$

where

$$v_j(nT') = \sum_{i=0}^{M-1} \mathcal{D}[\mathcal{E}^{-i} x_j(nT)] * q_{j,i}(nT') \tag{6.40}$$

Equations (6.39) and (6.40) describe an efficient and hierarchical realization of fractional decimators.

6.4.5 Systolic Implementation of Fractional Decimators

In this section, systolic structures for fractional decimators are obtained. The algebraic methodology will be hierarchically applied by first obtaining a systolic integer decimator structure to implement (6.40) and then it will be applied to implement (6.39).

It is easy to see that (6.40) is the same as (6.11) but with the output $y(nT')$, the input $x(nT)$, and the impulse response $p_i(nT')$ replaced by $v_j(nT')$, $x_j(nT)$, and $q_{j,i}(nT')$, respectively. On this basis, Figs. 6.5-6.7, and 6.10(a) represent possible array structures to evaluate (6.40).

Equation (6.39) can be put in the recursive form

$$\begin{aligned} y_j(nT_o) &= \mathcal{U}[v_j(nT')] + \mathcal{E}_o^{-r_1} y_{j-1}(nT_o) \quad \text{for } 0 < j \leq L-1 \\ y_0(nT_o) &= \mathcal{U}[v_0(nT')] + y_{-1}(nT_o) \\ y_{-1}(nT_o) &= 0 \\ y(nT_o) &= y_{L-1}(nT_o) \end{aligned} \tag{6.41}$$

and the input can be expressed as

$$\begin{aligned} x_j(nT) &= \mathcal{E}^{-r_0} x_{j-1}(nT) \quad \text{for } 1 \leq j \leq L-1 \\ x_0(nT) &= x(nT) \end{aligned} \tag{6.42}$$

Figure 6.11(a) shows the mapping of (6.41) and (6.42) onto a systolic structure while Fig. 6.11(b) describes the details of the PE involved. The symbol H_j represents the transfer function $H_j(z)$ of each polyphase integer decimator which has been implemented in Figs. 6.5-6.7, and 6.10(a). It can be seen that the structure is modular, pipelined, and hierarchical.

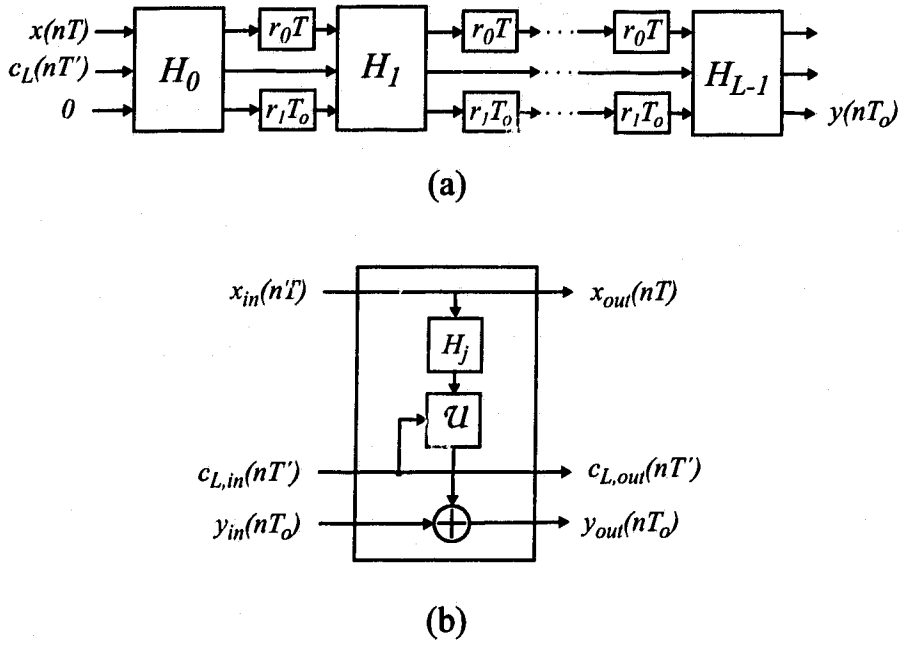


Figure 6.11: Fractional decimator scheme. (a) Mapping of (6.41) and (6.42) onto a systolic structure. (b) Details of PE involved.

6.4.6 Performance and Discussion

Processing rate is affected by the scheme used for decimator/interpolator structures. To illustrate this fact consider the decimator schemes. In scheme PD-I, the input data is pipelined, the control signal is broadcast, and the output results are added simultaneously according to Fig. 6.5. The minimum output sampling period can be deduced from the relation

$$T' > \max[M(T_r), T_{Pi} + MT_{add} + T_r] \tag{6.43}$$

where T_{Pi} is the processing time of a polyphase filter represented by the transfer function $P_i(z)$ and T_{add} is the adder delay.

For scheme PD-II, the inputs and the outputs are pipelined according to Fig. 6.6.

The minimum sampling period can be deduced from the delays in one pipeline stage using the relation

$$T' > \max[M(T_r),\ T_{Pi} + T_{add} + T_r] \tag{6.44}$$

Comparing relations (6.43) and (6.44), it is clear that scheme PD-II can handle higher processing rates than scheme PD-I especially for higher values of M.

For scheme PD-III, the input is broadcast while the outputs are pipelined according to Fig. 6.7. This scheme requires complex timing of input and control signals which yields inter processor buffers each introducing a delay equal to a fraction of one sampling period. The minimum sampling period can be deduced from the relation

$$T' > \max[M(T_B + T_r),\ T_{Pi} + T_{add} + T_r] \tag{6.45}$$

where T_B is the delay of the input data bus. T_B is proportional to the square of the number of PEs (i.e., M^2) [93]). As M increases, T_B could be the dominant term in the above inequality and puts an upper bound on the input processing rate because of bus loading and clock skewing problems. Scheme PD-III is considered less practical for VLSI implementation although it is attractive from the perspective of speed compared to scheme PD-I. Processing rate limitations can be deduced in the same way for interpolator structures.

It is noted that the latency for the array structures obtained in Sections 6.4.1-6.4.2 depends on the latency of the array structures of the polyphase filters. For decimator structures, the latency of scheme PD-I structure equals the latency of the polyphase subfilter structure T_p. For scheme PD-II, the latency is equal to $T_p + (M-1)T'$ and T_p for scheme PD-III. For interpolator structures, the latency

is T_p for scheme PI-I and $T_p + (L-1)LT''$ for scheme PI-II. It is noted that T_p is measured in terms of the low-rate sampling period for all structures. The latency for the efficient IIR decimator/interpolator structures and the fractional decimator/interpolator structures can be deduced in a similar way.

6.5 VLSI Array Processors for FIR Decimators and Interpolators

In this section, efficient array-processor implementations of FIR decimators and interpolators with integer compression/expansion factors are obtained by mapping their difference equations using the algebraic mapping methodology. The implementations are based on decimators/interpolators direct-form structures, where the prefixes FD and FI are used to denote an FIR decimator and an FIR interpolator, respectively.

6.5.1 Decimator Structures

The model of an M-to-1 decimator is shown in Fig. 6.12(a) [63]. According to this model, the filter with impulse response $h(nT)$, where T is the sampling period, operates at the high sampling rate F. $M-1$ out of every M output samples are discarded by the M-to-1 sampling rate compressor. For an N-tap FIR filter, a more efficient realization is obtained as shown in Fig. 6.12(b) in which the multiplications and additions associated with coefficients $h(0)$ to $h[(N-1)T]$ are performed at the low sampling rate F/M. Therefore, the computation rate in the system is reduced by a factor M. The structure in Fig. 6.12(b) is a direct-form realization of the

following relation [63]

$$y(nT') = \sum_{k=0}^{N-1} h(kT)\, x(nMT - kT) \tag{6.46}$$

where T and $T' = MT$ are the high-rate and low-rate sampling periods, respectively. The above equation can be expressed as

$$y(nT') = \sum_{i=0}^{m-1} \sum_{j=0}^{M-1} h[(iM+j)T]\, x(nMT - iMT - jT)$$

where $m = \lceil N/M \rceil$, $\lceil \cdot \rceil$ is the ceiling function, and $h(lT) = 0$ for $l > N-1$. Using the shift operators $\mathcal{E}$ and $\mathcal{E}_1$ for the input and output sampling periods, respectively, and the downsampling operator $\mathcal{D}$, the above relation can be written as

$$y(nT') = \sum_{i=0}^{m-1} \sum_{j=0}^{M-1} h[(iM+j)T]\, \mathcal{D}[\mathcal{E}_1^{-i}\mathcal{E}^{-j}x(nT)] \tag{6.47}$$

Alternatively,

$$y = \sum_{i=0}^{m-1} \mathbf{U}_i^T\, \mathcal{D}[\mathcal{E}_1^{-i}\mathbf{X}] \tag{6.48}$$

where $y \equiv y(nT')$, and vectors $\mathbf{U}_i$ and $\mathbf{X}$ are defined as

$$\mathbf{U}_i = [\ h(iMT) \quad h[(iM+1)T] \quad \cdots \quad h[(iM+M-1)T]\ ]^T$$
$$\mathbf{X} = \left[\ x(nT) \quad \mathcal{E}^{-1}x(nT) \quad \cdots \quad \mathcal{E}^{-(M-1)}x(nT)\ \right]^T$$

Based on (6.48), the decimator structure of Fig. 6.13(a) results where the ith PE performs the inner product operation $\mathbf{U}_i^T\mathcal{D}[\mathbf{X}_i]$, $\mathbf{X}_i = \mathcal{E}_1^{-i}\mathbf{X}$. Better utilization of each PE can be achieved if the inner product operation is performed on incoming data immediately and the partial results are stored as shown in Fig. 6.13(b). It is easy to see that the PE in Fig. 6.13(b) performs the downsampling and filtering operations.

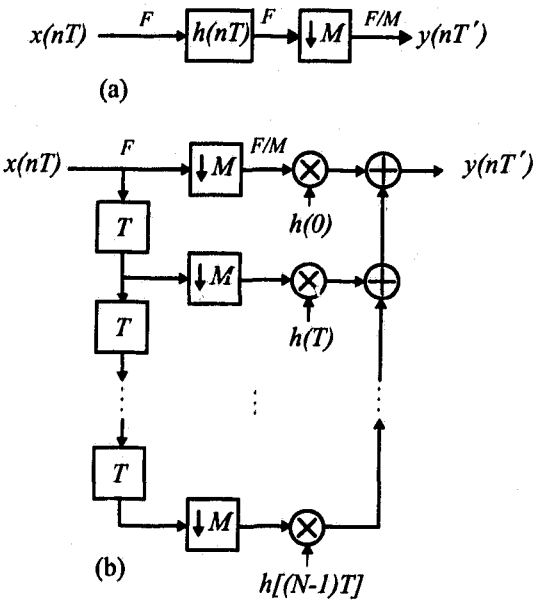


Figure 6.12: (a) General M-to-1 decimator system. (b) An efficient direct-form structure of an M-to-1 decimator.

Scheme FD-I

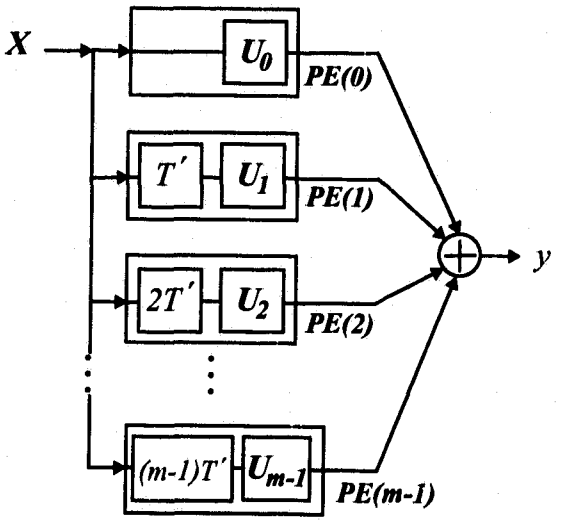
To obtain array-processor structures in which all PEs are identical, we introduce the intermediate vector

$$\mathbf{X}_i = \mathcal{E}_1^{-1}\mathbf{X}_{i-1}$$

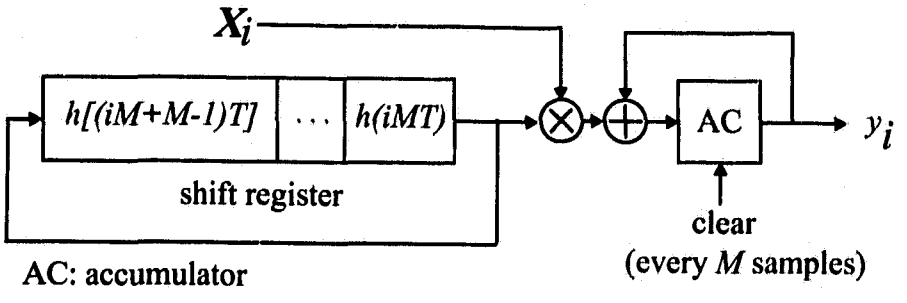
for $i = 1, 2, \ldots, m-1$ and $\mathbf{X}_0 = \mathbf{X}$, then (6.48) becomes

$$y = \{\mathbf{U}_0^T \mathcal{D}[\mathbf{X}_0]\} + \{\mathbf{U}_1^T \mathcal{D}[\mathbf{X}_1]\} + \cdots + \{\mathbf{U}_{m-1}^T \mathcal{D}[\mathbf{X}_{m-1}]\} \tag{6.49}$$

If each PE performs the operations inside each pair of brackets, then from (6.49) it can be seen that the input is propagated from one PE to the next. The filter output is produced after all the partial products have been added. Figure 6.14(a) shows the mapping of (6.49) while Fig. 6.14(b) shows the details of the PE involved. It can be seen that the structure is modular and that the input signal is pipelined between



(a)



(b)

Figure 6.13: (a) The proposed structure for M-to-1 decimator. (b) Efficient implementation of $\mathbf{U}_i^T \mathcal{D}[\mathbf{X}_i]$.

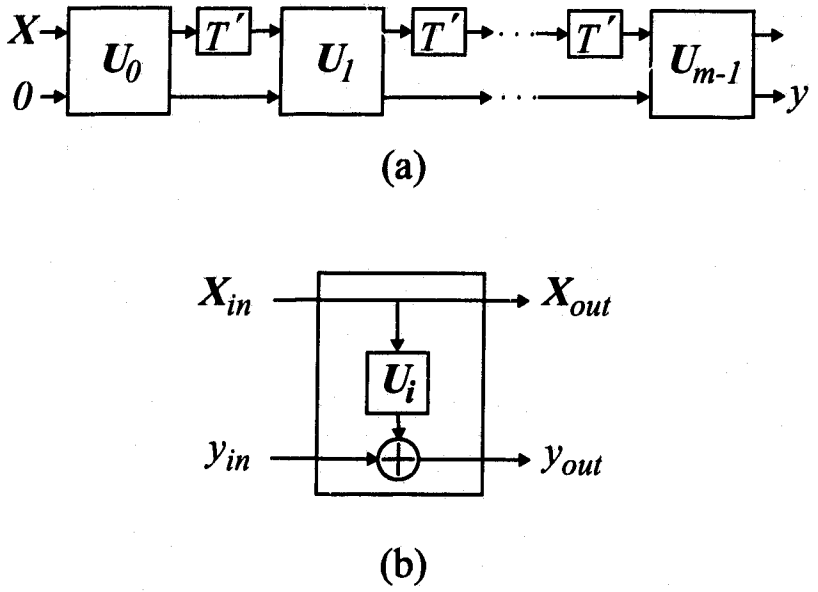


Figure 6.14: Scheme FD-I. (a) Mapping of (6.49) onto an array-processor structure. (b) Details of PE involved.

the PEs while the outputs are added simultaneously.

Scheme FD-II

Applying Horner's rule as in [56], (6.48) can be put in the form

$$y = \mathbf{U}_0^T \mathcal{D}[\mathbf{X}] + \mathcal{E}_1^{-1}\{\mathbf{U}_1^T \mathcal{D}[\mathbf{X}] + \cdots + \mathcal{E}_1^{-1}\{\mathbf{U}_{m-1}^T \mathcal{D}[\mathbf{X}]\}\cdots\} \tag{6.50}$$

Alternatively, in terms of recursive equations,

$$y_i = \mathbf{U}_i^T \mathcal{D}[\mathbf{X}] + \mathcal{E}_1^{-1} y_{i+1} \qquad \text{for } 0 \le i < m-1 \tag{6.51}$$

$$y_{m-1} = \mathbf{U}_{m-1}^T \mathcal{D}[\mathbf{X}] + y_m$$

$$y_m = 0$$

$$y = y_0$$

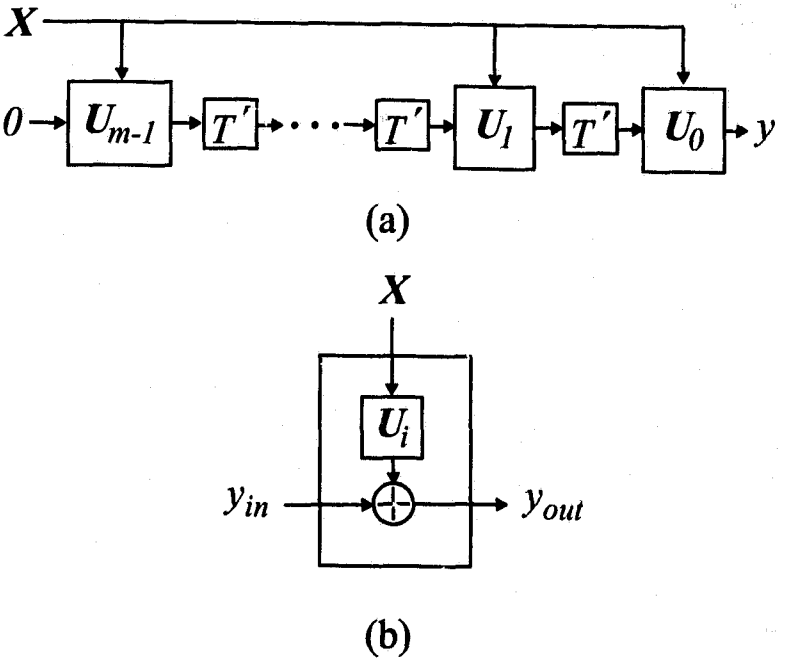


Figure 6.15: Scheme FD-II. (a) Mapping of (6.51) onto an array-processor structure. (b) Details of PE involved.

The ith PE computes y_i in (6.51) using one inner product and one addition. Figure 6.15(a) shows the mapping of (6.51) onto an array-processor structure while Fig. 6.15(b) describes the details of the PE involved. It can be seen that the structure is modular and that the input signal is broadcast to all the PEs while the outputs are pipelined.

Scheme FD-III

Implementations in which both the input and the output signals are pipelined can be obtained [100]. Delaying the output by $m-1$ sampling periods, (6.48) can be written as

$$\widehat{y} = \sum_{i=0}^{m-1} \mathcal{E}_1^{-(m-1-i)} \mathbf{U}_i^T \mathcal{D}[\mathcal{E}_1^{-2i}\mathbf{X}] \tag{6.52}$$

where $\widehat{y}$ represents the delayed system output. The above relation can be written as

$$\widehat{y} = \mathbf{U}_{m-1}^T \mathcal{D}[\mathbf{X}_{m-1}] + \mathcal{E}_1^{-1}\{\mathbf{U}_{m-2}^T \mathcal{D}[\mathbf{X}_{m-2}] + \cdots + \mathcal{E}_1^{-1}\{\mathbf{U}_0^T \mathcal{D}[\mathbf{X}_0]\}\cdots\} \tag{6.53}$$

where

$$\mathbf{X}_i = \mathcal{E}_1^{-2}\mathbf{X}_{i-1}$$

for $i = 1, 2, \ldots, m-1$ and $\mathbf{X}_0 = \mathbf{X}$. Equation (6.53) can be written in the recursive form

$$\begin{aligned} \widehat{y}_i &= \mathbf{U}_i^T \mathcal{D}[\mathbf{X}_i] + \mathcal{E}_1^{-1}\widehat{y}_{i-1} \qquad \text{for } 0 < i \leq m-1 \\ \widehat{y}_0 &= \mathbf{U}_0^T \mathcal{D}[\mathbf{X}_0] + \widehat{y}_{-1} \\ \widehat{y}_{-1} &= 0 \\ \widehat{y} &= \widehat{y}_{m-1} \end{aligned} \tag{6.54}$$

The ith PE is assigned to compute y_i in (6.54). The mapping of (6.54) onto a systolic structure is shown in Fig. 6.16(a) while Fig. 6.16(b) depicts the details of the PE involved.

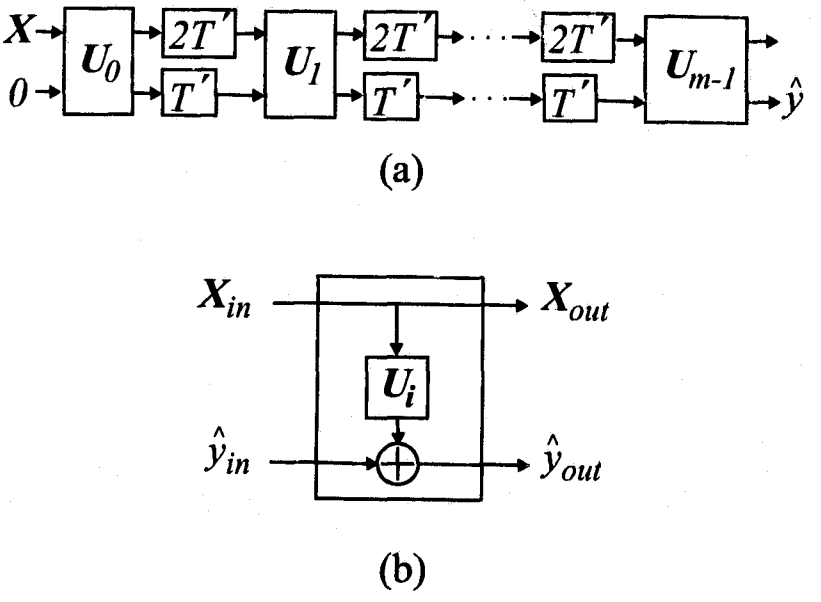


Figure 6.16: Scheme FD-III. (a) Mapping of (6.54) onto a systolic structure. (b) Details of PE involved.

Scheme FD-IV

Implementations in which both the input and the output signals are pipelined while maintaining a latency of one low-rate sampling period can be obtained. This scheme involves the use of more complex PEs. Delaying the output by one low-rate sampling period, (6.48) can be written as

$$\hat{y} = \sum_{i=0}^{m-1} \mathbf{U}_i^T \mathcal{D}\left[\mathcal{E}_1^{-(i+1)}\mathbf{X}\right] \tag{6.55}$$

For the case when m is even, (6.55) can be written as

$$\begin{aligned}\hat{y} = {} & \mathcal{E}_1^{-1}\{[\mathbf{U}_0^T\mathcal{D}[\mathbf{X}_0] + \mathbf{U}_1^T\mathcal{D}[\mathbf{X}_1]] + \mathcal{E}_1^{-1}\{[\mathbf{U}_2^T\mathcal{D}[\mathbf{X}_1] + \mathbf{U}_3^T\mathcal{D}[\mathbf{X}_2]] + \cdots \\ & + \mathcal{E}_1^{-1}\{[\mathbf{U}_{m-2}^T\mathcal{D}[\mathbf{X}_p] + \mathbf{U}_{m-1}^T\mathcal{D}[\mathbf{X}_q]]\}\cdots\}\}\end{aligned} \tag{6.56}$$

where

$$\mathbf{X}_i = \mathcal{E}_1^{-1}\, \mathbf{X}_{i-1}$$

for $i = 1, 2, \ldots, p, q$, where $p = (m-2)/2$ and $q = m/2$, and $\mathbf{X}_0 = \mathbf{X}$. Equation (6.56) can be written as

$$\widehat{y}_i = \mathcal{E}_1^{-1}\{\mathbf{U}_{2i}^T \mathcal{D}[\mathbf{X}_i] + \mathbf{U}_{2i+1}^T \mathcal{D}[\mathbf{X}_{i+1}] + \widehat{y}_{i+1}\} \quad \text{for } 0 \le i \le m/2-1 \qquad (6.57)$$

$$\widehat{y}_q = 0$$

$$\widehat{y} = \widehat{y}_0$$

The ith PE is assigned to compute y_i in (6.57). Each PE is required to perform two inner product evaluations and two additions, and has two inputs and two outputs. The two inputs are the signal input and the partial sum from the adjacent PE. The outputs are the delayed input signal and the partial sum. The mapping of (6.57) onto a systolic structure is shown in Fig. 6.17(a) while Fig. 6.17(b) depicts the details of the PE involved.

A systolic structure for the case when m is odd can be derived using the same PE of Fig. 6.17(b). The resulting systolic implementation is shown in Fig. 6.17(c).

6.5.2 Interpolator Structures

The general structure for the 1-to-L integer interpolator is shown in Fig. 6.18(a). The output $y(nT'')$ can be written as [63]

$$y(nT'') = \sum_{i=-\infty}^{+\infty} h[(iL + n \oplus L)T'']\; x[(\lfloor \frac{n}{L} \rfloor - i)T] \qquad (6.58)$$

where $T'' = T/L$ is the high-rate output sampling period, $n \oplus L$ denotes the value of n modulo L, and $\lfloor \cdot \rfloor$ is the floor function. The output is obtained by passing the

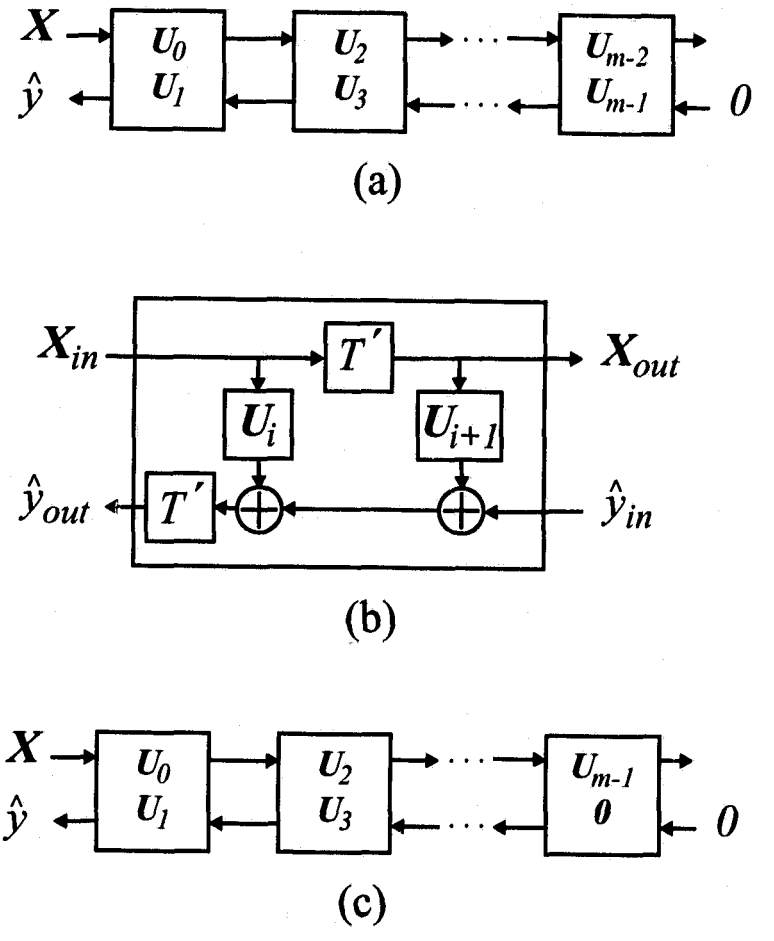


Figure 6.17: Scheme FD-IV. (a) Mapping of (6.57) onto a systolic structure for m even. (b) Details of PE involved for m even. (c) The systolic array for m odd.

input $x(nT)$ through a time-varying filter with impulse response

$$g[(i,n)T''] = h[(iL+n \oplus L)T''], \quad -\infty < i,n < +\infty \tag{6.59}$$

It is noted that $g[(i,n)T'']$ is periodic in the variable n with period L. Let N be the filter length, $m = \lceil \frac{N}{L} \rceil$, and $h(lT'') = 0$ for $l > N-1$. Consequently, the set of coefficients $g[(i,n)T'']$ for each $n = 0, 1, \ldots, L-1$, contains m elements. It follows that

$$y(nT'') = \sum_{i=0}^{m-1} g[(i, n - \lfloor \frac{n}{L} \rfloor L)T''] \; x[(\lfloor \frac{n}{L} \rfloor - i)T] \tag{6.60}$$

The output can be obtained by processing blocks of data of length m by a set of m filter coefficients $g[(i, n - \lfloor \frac{n}{L} \rfloor L)T'']$, $i = 0, 1, \ldots, m-1$. There are L such sets of coefficients, one for each block of L output samples of $y(nT'')$. For each block of L output samples, there is a corresponding input sample of $x(nT)$ that enters the computation. A structure to compute $y(nT'')$ is shown in Fig. 6.18(b). Equation (6.60) can be written as

$$\mathbf{Y} = \sum_{i=0}^{m-1} \mathbf{V}_i \mathcal{H}[\mathcal{E}^{-i} x] \tag{6.61}$$

where $x \equiv x(nT)$ and $\mathcal{H}$ is an operator performing digital hold and sample. The vectors $\mathbf{Y}$ and $\mathbf{V}_i$ are defined as

$$\mathbf{V}_i = [\; h(iLT'') \quad h[(iL+1)T''] \quad \cdots \quad h[(iL+L-1)T''] \;]^T$$

$$\mathbf{Y} = \left[\; y(nT'') \quad \mathcal{E}_2^{-1} y(nT'') \quad \cdots \quad \mathcal{E}_2^{-(L-1)} y(nT'') \;\right]^T$$

where $\mathcal{E}_2$ is a shift operator for the output sampling period. Since $\mathcal{E}^{-1} = \mathcal{E}_2^{-L}$, (6.61) can be put in the form

$$\mathbf{Y} = \sum_{i=0}^{m-1} \mathcal{E}_2^{-iL} \mathbf{V}_i \mathcal{H}[x] \tag{6.62}$$

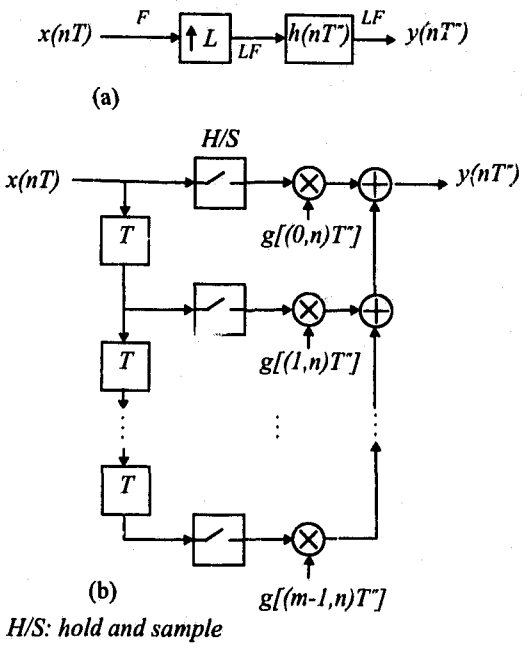
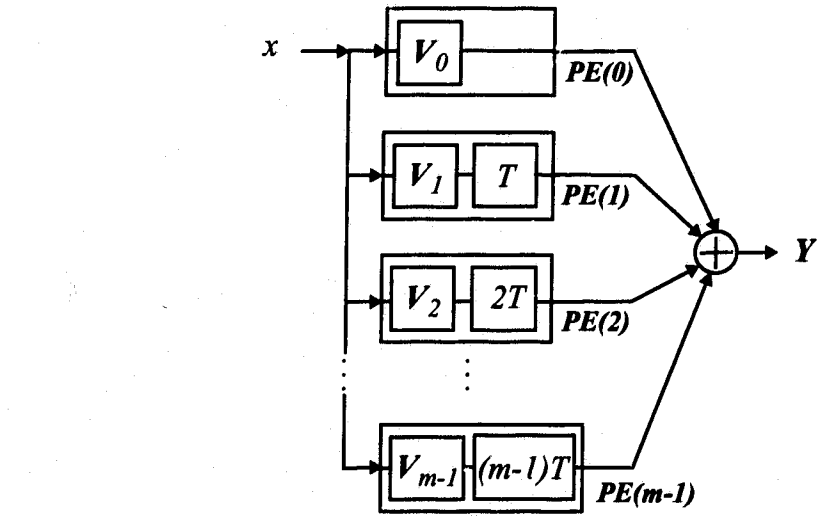


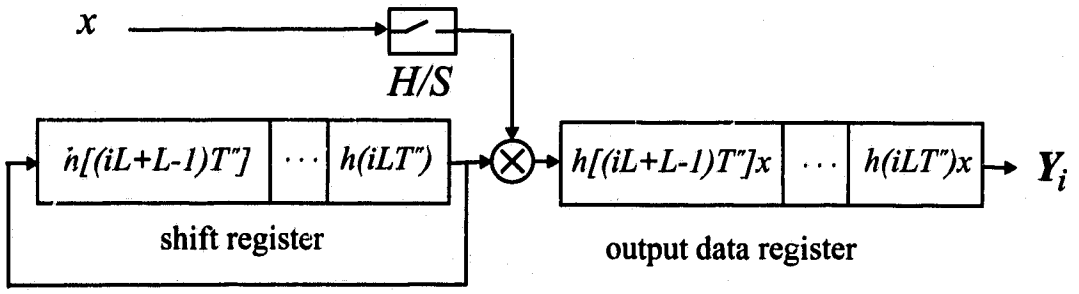
Figure 6.18: (a) General 1-to-L interpolator system. (b) An efficient structure of a 1-to-L interpolator.

Based on (6.61)-(6.62), the interpolator structure can be redrawn as in Fig. 6.19(a), where each polynomial evaluation $\mathbf{V}_i\mathcal{H}[x]$ can be implemented using the structure of Fig. 6.19(b).

It is easy to show that the array-processor schemes for the interpolator can be obtained by following the same procedure for the decimator schemes provided the following modifications are adopted. Subsystem $\mathbf{U}_i$ is replaced by subfilter $\mathbf{V}_i$, the input $\mathbf{X}$ is replaced by x, the output y is replaced by $\mathbf{Y}$, and T' is replaced by T. Furthermore, the input data is processed at the low rate while the output is processed at the high rate. On the basis of the aforementioned modifications, Figs. 6.14-6.17. yield interpolator array structures.



(a)



(b)

Figure 6.19: (a) The proposed structure for 1-to-L interpolator. (b) Efficient implementation of $\mathbf{V}_i\mathcal{H}[x]$.

6.5.3 Performance and Discussion

For scheme FD-I, sampling period limitations can be deduced from the inequality

$$T' > mT_{add} + M(T_{add} + T_{mult}) + T_r \tag{6.63}$$

For scheme FD-II, sampling period limitations can be deduced from the relation

$$T' > \max[T_B + T_r,\ T_{add} + T_r + M(T_{add} + T_{mult})] \tag{6.64}$$

where T_B is the delay of the input data bus which is proportional to the square of the number of PEs (i.e., m^2) [93]. As m increases, T_B dominates the above inequality and puts an upper bound on the input processing rate. For schemes FD-III and FD-IV, processing rates can be deduced by the following relations

$$T' > T_{add} + T_r + M(T_{add} + T_{mult}) \tag{6.65}$$

From perspective of speed, higher data rates can be achieved with schemes FD-III and FD-IV than schemes FD-I and FD-II.

The minimum latency for schemes FD-I, FD-II, and FD-IV is one low-rate sampling period. For scheme FD-III, the latency is m low-rate sampling periods.

6.6 Comparisons

In Section 6.4, array-processor implementations for decimators and interpolators were obtained based on their polyphase structures. Here, we compare those implementations with their counterparts in Section 6.5 for the case of decimators.

From the performance of Sections 6.4.6 and 6.5.3, it is easy to verify that the processing rates of the implementations obtained in Section 6.4 are higher than their

counterparts reported in Section 6.5. However, they require more area. This can be verified by observing that the implementations of Section 6.5 require fewer adders and multipliers than those in Section 6.4. As an example, consider an FIR decimator with a prototype filter of length $N = 16$ with $M = 2$, hence $m = 8$ implemented in terms of the fully pipelined schemes, i.e., scheme PD-II of Section 6.4, and scheme FD-III of Section 6.5. The processing time of scheme PD-II is approximately equal to $2T_{add} + T_{mult}$ assuming fully pipelined implementations for the individual polyphase filters, while that of scheme FD-III is equal to $3T_{add} + 2T_{mult}$. On the other hand, scheme PD-II requires 18 adders and 16 multipliers, while scheme FD-III requires 16 adders and 8 multipliers.

6.7 Filter Bank Implementation

Array-processor implementations for PR filter banks can be constructed using decimator and interpolator array structures presented in Sections 6.4-6.5. Any of the analysis filters followed by a compressor can be constructed using any of the array structures in Figs. 6.5-6.7, 6.10(a), and Figs. 6.14-6.17. Similarly, any of the synthesis filters preceded by an expander can be constructed using any of the interpolator array structures. It is observed that the input delay chains and compressors can be common for all channels for the analysis bank. Conversely, the output delay chains and the expanders can be common for the synthesis bank.

6.7.1 Array Processors for Polyphase Filter Banks

It is preferable to incorporate polyphase filter structures having the same latency throughout the system. This avoids having to add extra delays to the polyphase filter structures with small latencies.

One example to illustrate how to obtain filter bank array structures is to implement the two-channel PR FIR QMF bank using schemes PD-II and PI-II. For the PR FIR QMF banks, two independent filters are designed to be used as analysis and synthesis filters. Since $M = L = 2$, the analysis and synthesis structures each comprises two PEs as shown in Fig. 6.20 where T is the high-rate sampling period and T_{sb} is the low-rate sampling period for the the subband signals. The polyphase filters are related to their particular prototype filter as indicated in the figure. The structure in Fig. 6.20 is suitable to implement PR filter banks. The processing rate depends on the polyphase filter structure. The latency of the structure is $T_{sb} + T_p$. Following the same procedure, different structures can be obtained using various decimator and interpolator schemes obtained in Section 6.4.

Systolic implementations of M-channel FIR PR filter banks can be obtained following a similar procedure. In this case, the number of PEs is M and the number of polyphase filter structures in each PE is also M.

The example in Fig. 6.21 illustrates the implementation of the conventional two-channel QMF bank using schemes PD-I and PI-I. From the VLSI implementation perspective, this category has the advantage of using the unique relation between $H_0(z)$ and $H_1(z)$. This means that the analysis/synthesis filters can be combined in a single structure using polyphase components of $H_0(z)$. This results in 50% savings in the number of multipliers used compared with the two-channel PR FIR QMF banks. Following a similar procedure, implementations for M-channel filter banks can be obtained. In this case, the number of PEs is M with an M-point discrete-Fourier transform (DFT) and a scaled inverse DFT (IDFT) to replace the 2-point DFT and scaled IDFT processors. It is noted that the scaling multiplicative factor is equal to M.

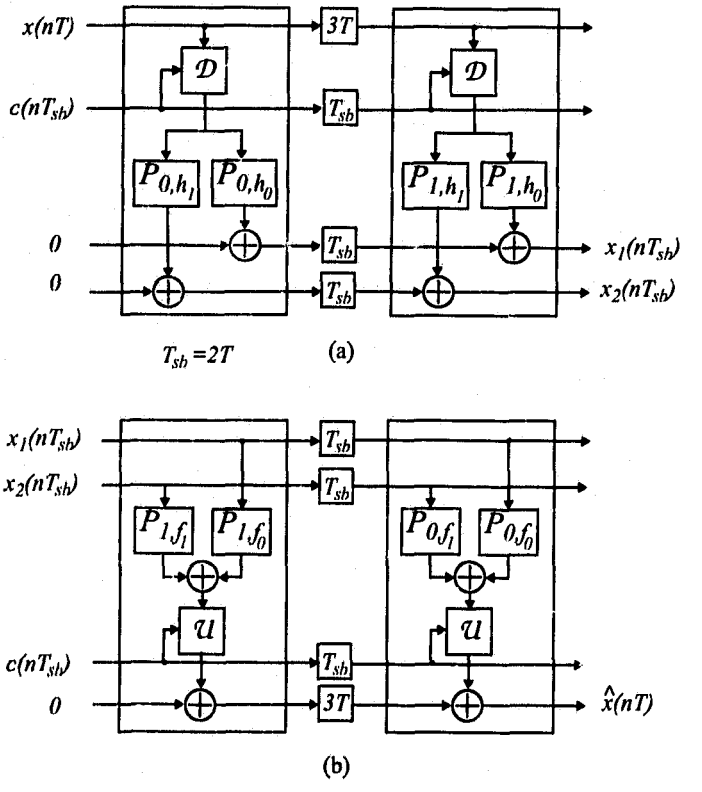


Figure 6.20: Systolic implementation of a two-channel PR FIR QMF bank. (a) Systolic structure of the analysis bank. (b) Systolic structure of the synthesis bank.

6.7.2 Array Processors for FIR Filter Banks

One example to illustrate how to obtain filter bank array structures is to implement the two-channel PR FIR QMF bank using scheme FD-III and its interpolator counterpart, i.e., scheme FI-III. Figure 6.22 illustrates the resulting systolic implementation of the analysis and synthesis banks. Since $M = L = 2$, the structures of Figs. 6.13(b) and 6.19(b) have to be modified as in Fig. 6.23 where the CSIPP module presented in Chapter 5 can be used in Fig. 6.23(a) to increase the process-

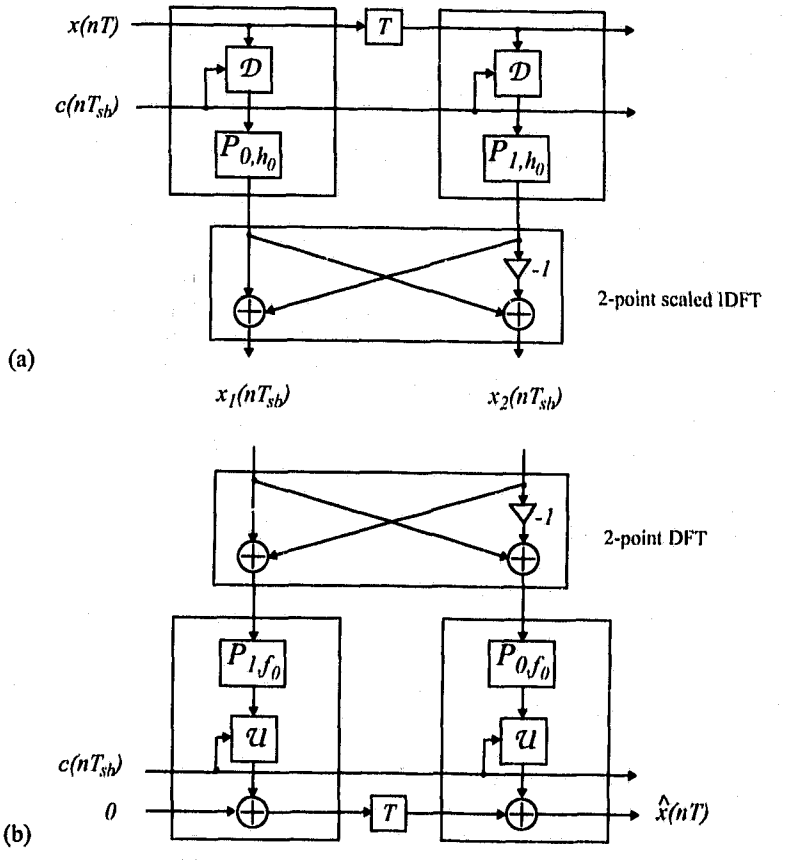


Figure 6.21: Array-processor implementation of a conventional two-channel QMF bank. (a) Array-processor structure of the analysis bank.
(b) Array-processor structure of the synthesis bank.

ing rate. In Figs. 6.22 and 6.23, $\mathbf{X} = [x(nT) \;\; \mathcal{E}^{-1}x(nT)]^T$, $\hat{\mathbf{X}} = \sum_{i=0}^{m-1} \mathcal{E}_1^{-i}\hat{\mathbf{X}}_i$, and $\hat{\mathbf{X}}_i = [\hat{x}(nT) \;\; \mathcal{E}^{-1}\hat{x}(nT)]^T$, where $\mathcal{E}$ and $\mathcal{E}_1$ denote shift operators for the high-rate and low-rate sampling periods, respectively. In this case, $m = \max(\lceil N_0/2 \rceil, \lceil N_1/2 \rceil)$ where N_0 and N_1 are the lengths of the analysis filters H_0 and H_1, respectively, or the lengths of the synthesis filters F_1 and F_0. The latency of the structure is mT_{sb}. Structures with latency of one low-rate sampling period (T_{sb}) can be obtained using the decimator and interpolator structures in schemes I, II, and IV. It is noted

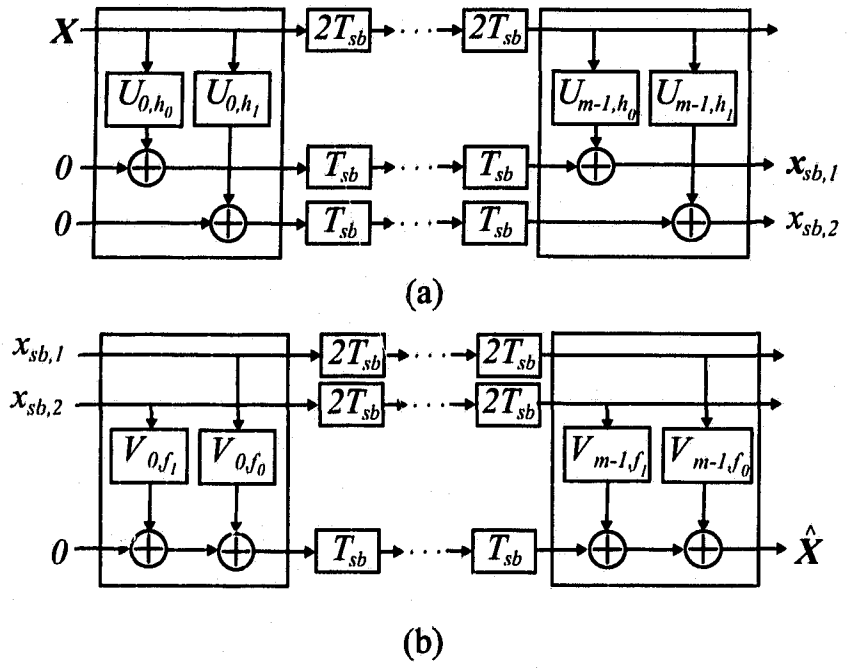


Figure 6.22: Systolic implementation of a two-channel PR FIR QMF bank. (a) Systolic structure of the analysis bank. (b) Systolic structure of the synthesis bank.

that this implementation is not efficient for implementing classical QMF banks and linear-phase FIR PR QMF banks.

6.8 Conclusions

New array-processor implementations for multirate filter banks have been obtained by mapping decimator and interpolator algorithms onto hardware structures us-

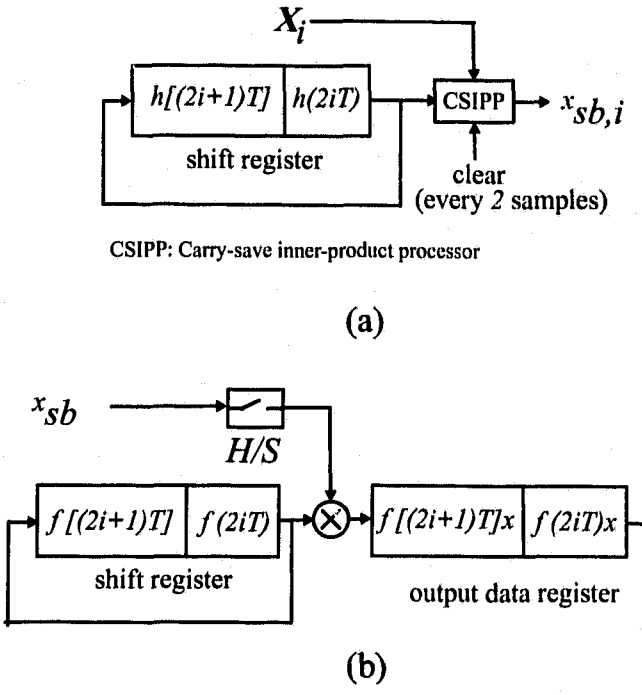


Figure 6.23: (a) Efficient implementation of $\mathbf{U}_i^T \mathcal{D}[\mathbf{X}_i]$. (b) Efficient implementation of $\mathbf{V}_i \mathcal{H}[x_{sb}]$.

ing an algebraic mapping methodology. Two approaches have been employed. In the first approach, array-processor implementations for polyphase decimators/interpolators with integer/fraction compression/expansion factors have been presented. These structures are useful for high-speed multirate signal processing applications and, eventually, are of particular importance to filter bank implementations. Most of the proposed structures do not suffer from signal timing problems and are amenable to hardware implementation in which one master clock is incorporated for system timing. The schemes obtained are efficient and flexible in realizing FIR or IIR decimator, interpolator, and filter bank systems. The implementations are regular, modular, hierarchical, and some of them entail local communications among PEs.

In the second approach, new array-processor implementations for FIR decimators and interpolators have been obtained based on their direct-form structures. The structures obtained entail savings in the number of multipliers and adders for each PE. One other advantage achieved is the absence of commutators in all structures. A comparison among the different proposed schemes has been presented. Upper bounds on the input/output processing rates have been provided in terms of system parameters and hardware delays. The implementations are regular, modular, and some of them entail local communications among PEs and/or minimum latency of one low-rate sampling period.

Chapter 7

Conclusions

The objectives of this thesis have been to develop new approaches for the design of two-channel QMF banks as well as M-channel filter banks, to obtain efficient VLSI array processors for filter banks based on the polyphase and direct-form structures of decimators and interpolators, to obtain new VLSI array processors for digital filters, and to obtain new fixed-point processors that can be incorporated in the above implementations.

7.1 Contributions

In Chapter 2, two constrained optimization approaches have been applied for the design of two-channel PR FIR QMF banks with low delays. The first approach is based on the Lagrange-multiplier method and can be used to design banks with filters of unequal lengths. The approach is simple, efficient, flexible, and leads to a closed-form exact solution. The second approach can be used to design filters with equal as well as unequal lengths. In this approach, the design is formulated as a quadratic constrained least-squares minimization problem which can be solved using standard minimization algorithms. The two approaches have been compared

with other known methods for the design of FIR QMF banks and found to yield reduced reconstruction error and it is sometimes possible to obtain increased minimum stopband attenuation and decreased passband ripple.

In Chapter 3, two design approaches for filter banks have been presented. In the first approach, a computationally efficient method has been applied to the design of cosine-modulated pseudo-QMF banks. The method is quite efficient in producing good designs while reducing the amount of computation and design time. In the second approach, a modified WLS method has been employed to obtain a weighted minimax design of linear-phase FIR QMF banks. The approach is considered easy to implement and efficient in terms of the amount of computations and design time compared to other WLS design approaches. Design of QMF banks satisfying prescribed specifications has also been considered.

In Chapter 4, array processors for FIR filters and linear-phase FIR filters have been obtained using the SFG approach. Those implementations are considered integral parts of the polyphase structures of decimators, interpolators, and filter banks presented in Chapter 6. The SFG method was employed to obtain an FIR array processor scheme in which the outputs are localized in separate processing elements. The method was also employed to obtain new and novel array structures for linear-phase FIR filters. All the implementations obtained are modular and regular.

In Chapter 5, a new inner-product processor has been presented. The new processor enhances the speed of operation with a slight increase in the area. This processor can be used in the LP FIR scheme presented in Chapter 4 and in the array-processor implementations of FIR decimators and filter banks presented in Chapter 6. Also, a new processor module to perform an add-multiply-accumulate

operation has been presented which enhances the speed of operation of the linear-phase FIR filter implementations of Chapter 4 without increasing the silicon area or introducing extra latency in the system.

In Chapter 6, new array-processor implementations for multirate filter banks have been obtained by mapping decimator and interpolator algorithms onto hardware structures using an algebraic mapping methodology. Two approaches have been employed. In the first approach, array-processor implementations for polyphase decimators/interpolators with integer/fraction compression/expansion factors have been presented. These structures are useful for high-speed multirate signal processing applications and, eventually, are of particular importance to filter bank implementations. Most of the proposed structures do not suffer from signal timing problems and are amenable to hardware implementation in which one master clock is incorporated for system timing. The schemes obtained are efficient and flexible in realizing FIR or IIR decimator, interpolator, and filter bank systems. The implementations are regular, modular, hierarchical, and some of them entail local communications among PEs. In the second approach, new array-processor implementations for FIR decimators and interpolators have been obtained based on their direct-form structures. The structures obtained entail savings in the number of multipliers and adders for each PE. One other advantage achieved is the absence of commutators in all structures. A comparison among the different proposed schemes has been presented. Upper bounds on the input/output processing rates have been provided in terms of system parameters and hardware delays. The implementations are regular, modular, and some of them entail local communications among PEs and/or minimum latency of one low-rate sampling period.

7.2 Suggestions for Further Research

As there is a close relation between the discrete wavelet transform and filter banks, the design of wavelet filter banks can be considered using the proposed design procedures. One-dimensional QMF banks can be used to construct two-dimensional separable filter banks. A problem arising with separable sampling structures is that some of the subbands obtained contain mixed orientations. A possible remedy would be to consider the hexagonal sampling lattice which produces subbands with pure directional orientations. Hence, extension of the design methods incorporated in this thesis to two-dimensional case is recommended.

Multirate filter banks have shown promises to increase the rate of convergence and reduce the amount of computation in adaptive systems [101]. However, perfect reconstruction systems exhibit problems related to the overall delay of the filter bank system. Eventually, using low-delay filter banks in the adaptive multirate system would alleviate these problems.

The implementations presented for the one-dimensional filter banks can be extended to two-dimensional filter banks based on the polyphase representations of two-dimensional decimators and interpolators.

Digital filter algorithms are characterized by an arithmetic sum of products. Distributed arithmetic (DA) [102] is a parallel serial implementation of the sum of products. The serial nature of the DA circuit precludes the highest-speed application. However, with the concurrent processing of all inputs, the performance is adequate for many real-time DSP applications. Furthermore, the regularity of the memory array and its peripheral circuits permit highly efficient VLSI implementation. DA techniques permits the sum of products to map well into field-programmable gate ar-

rays. Xilinx devices [103] are ideal for this application and can be used to implement digital filters, decimators, interpolators, and filter banks.

In Chapter 5, new fixed-point processors were presented based on parallel multiplier design. Fixed-point processors that deal with complex data input can be similarly designed. An important research would be to extend the existing designs to floating-point arithmetic.

Bibliography

[1] A. Croisier, D. Esteban, and C. Galand, "Perfect-channel splitting by use of interpolation/decimation tree decomposition techniques," *Proc. IEEE Int. Conf. Inform. Sci. Syst.*, Patras, Greece, 1976.

[2] D. Esteban, and C. Galand, "Application of quadrature mirror filters to split band voice coding schemes," *Proc. IEEE Int. Conf. Acoust., Speech, Signal Processing*, Hartford, CT, pp. 191-195, May 1977.

[3] J. W. Woods and S. D. O'Neil, "Subband coding of images," *IEEE Trans. Acoust., Speech, Signal Processing*, vol. ASSP-34, pp. 1278-1288, Oct. 1986.

[4] H. Gharavi and A. Tabatabai, "Sub-band coding of monochrome and color images," *IEEE Trans. Circuits Syst.*, vol. CAS-35, pp. 207-214, Feb. 1988.

[5] R. D. Koilpillai, T. Q. Nguyen, and P. P. Vaidyanathan, "Some results in the theory of cross-talk free transmultiplers," *IEEE Trans. Signal Processing*, vol. 39, pp. 2174-2183, Oct. 91.

[6] H. O. Özturk and W. E. Alexander, "Multiplexing of 2-D signals in the frequency domain using QMF structures," *Proc. 32nd Midwest Symp. Circuits*

Syst., pp. 1107-1110, Aug. 1989.

[7] M. Vetterli, "Perfect Transmultiplexers," *Proc. IEEE Int. Conf. Acoust., Spreech, Signal Processing*, Tokyo, Japan, pp. 2567-2570, Apr. 1986.

[8] M. Vetterli and C. Herley, "Wavelets and filter banks: relationships and new results," *Proc. IEEE Int. Conf. Acoust., Speech, Signal Processing*, Albuquerque, New Mexico, pp. 1723-1726, Apr. 1990.

[9] R. V. Cox, W. E. Bush, K. B. Bauer, J. D. Johnston, and J. H. Snyder, "The analog voice privacy system," *Proc. IEEE Int Conf Acoust., Speech, Signal Processing*, pp. 341-344, Apr. 1986.

[10] P. Vary and U. Heute, "A short-time spectrum analyzer with polyphase network and DFT," *Signal Processing*, vol. 2, pp. 55-65, 1980.

[11] P. Vary and U. Heute, "A digital filter bank with polyphase network and FFT hardware: measurements and applications," *Signal Processing*, vol. 3, pp. 307-319, 1981.

[12] J. D. Johnston, "A filter family designed for use in quadrature mirror filter banks," *Proc. IEEE Int. Conf. Acoust., Speech, Signal Processing*, Denver, CO, pp. 291-294, Apr. 1980.

[13] G. Pirani and V. Zingarelli, "An analytical formula for the design of quadrature mirror filters," *IEEE Trans. Acoust. Speech, Signal Processing*, vol. ASSP-32, pp. 645-648, June 1984.

[14] V. K. Jain and R. E. Crochiere, "Quadrature mirror filter design in the time domain," *IEEE Trans. Acoust., Speech, Signal Processing*, vol. ASSP-32, pp. 353-

361, Apr. 1984.

[15] K. Swaminathan and P. P. Vaidyanathan, "Theory and design of uniform DFT, parallel, quadrature mirror filter banks," *IEEE Trans. Circuits Syst.*, vol. CAS-33, pp. 1170-1191, Dec. 1986.

[16] F. Grenez, "Chebyshev design of filters for subband coders," *IEEE Trans. Acoust., Speech, Signal Processing*, vol. ASSP-36, pp. 182-185, Feb. 1988.

[17] C.-K. Chen and J.-H. Lee, "Design of quadrature mirror filters with linear-phase in the frequency domain," *IEEE Trans. Circuits Syst.-II: Analog and Digital Signal Processing*, vol. 39, pp. 593-605, Sept. 1992.

[18] Y. C. Lim, R. H. Yang, and S. N. Koh, "The design of weighted minimax quadrature mirror filters," *IEEE Trans. Signal Processing*, vol. 41, pp. 1780-1789, May 1993.

[19] F. Mintzer, "Filters for distortion-free two-band multirate filter banks," *IEEE Trans. Acoust., Speech, Signal Processing*, vol. ASSP-33, pp. 626-630, June 1985.

[20] M. J. T. Smith and T. P. Barnwell, III, "Exact reconstruction techniques for tree-structured subband coders," *IEEE Trans. Acoust., Speech, Signal Processing*, vol. ASSP-34, pp. 434-441, June 1986.

[21] S. K. Mitra, H. Babic, and V. S. Somayazulu, "A modified perfect-reconstruction QMF bank with an auxiliary channel," *Proc. IEEE Int. Conf. Acoust., Speech, Signal Processing*, Glasgow, Scotland, pp. 2132-2135, May 1989.

[22] M. Vetterli, "Filter banks allowing perfect reconstruction," *Signal Processing*, vol. 10, pp. 219-244, 1986.

[23] T. Q. Nguyen and P. P. Vaidyanathan, "Two-channel perfect-reconstruction FIR QMF structures which yield linear-phase analysis and synthesis filters," *IEEE Trans. Acoust., Speech, Signal Processing*, vol. ASSP-37, pp. 676-690, May 1989.

[24] B.-R. Horng and A. N. Willson, Jr., "Lagrange multiplier approaches to the design of two-channel perfect-reconstruction linear-phase filter banks," *IEEE Trans. Signal Processing*, vol. 40, pp. 364-374, Feb. 1992.

[25] T. Q. Nguyen, "A quadratic constrained least-squares approach to the design of digital filter banks," *Proc. IEEE Int. Symp. Circuits Syst.*, San Diego, CA, pp. 1344-1347, May 1992.

[26] P. P. Vaidyanathan, "Quadrature mirror filter banks, M-band extensions and perfect-reconstruction techniques," *IEEE Signal Processing Magazine*, pp. 4-20, July 1987.

[27] P. P. Vaidyanathan, *Multirate Systems and Filter Banks*, Prentice-Hall, Englewood Cliffs, New Jersey, 1993.

[28] P. P. Vaidyanathan, "Multirate digital filters, filter banks, polyphase networks, and applications: A tutorial," *Proc. IEEE*, vol. 78, pp. 56-93, Jan. 1990.

[29] T. P. Barnwell, III, "Subband coder design incorporating recursive quadrature filters and optimum ADPCM coders," *IEEE Trans. Acoust., Speech, Signal Processing*, vol. ASSP-30, pp. 751-765, Oct. 1982.

[30] P. C. Millar, "Recursive quadrature mirror filters-criteria specification and design method," *IEEE Trans. Acoust., Speech, Signal Processing*, vol. ASSP-33, pp. 413-420, Apr. 1985.

[31] P. Regalia, S. K. Mitra, P. P. Vaidyanathan, M. K. Renfors, and Y. Neuvo, "Tree-structured complementary filter banks using all-pass sections," *IEEE Trans. Circuits Syst.*, vol. CAS-34, pp. 1470-1484, Dec. 1987.

[32] S.-C. Pei and S.-B. Jaw, "A class of efficient recursive quadrature mirror filters for subband coding," *IEEE Trans. Signal Processing*, vol. 39, pp. 2721-2725, Dec. 1991.

[33] T. A. Ramstad, "IIR filterbank for subband coding of images," *Proc. IEEE Int. Symp. Circuits Syst.*, Espoo, Finland, pp. 827-830, June 1988.

[34] M. J. T. Smith and S. L. Eddins, "Analysis/synthesis techniques for subband image coding," *IEEE Trans. Acoust., Speech, Signal Processing*, vol. ASSP-38, pp. 1446-1456, Aug. 1990.

[35] T. Kronander, "A new approach to recursive mirror filters with a special application in subband coding of images," *IEEE Trans. Acoust., Speech, Signal Processing*, vol. ASSP-36, pp. 1496-1500, Sept. 1988.

[36] A. Fettweis, T. Leickel, M. Bolle, and U. Sauvagerd, "Realization of filter banks by means of wave digital filters," *Proc. IEEE Int. Symp. Circuits Syst.*, New Orleans, LA, pp. 2013-2016, May 1990.

[37] K. Nayebi, T. P. Barnwell, and M. J. T. Smith, "Time-domain filter bank analysis: A new design theory," *IEEE Trans. Signal Processing*, vol. 40, pp. 1412-

1429, June 1992.

[38] K. Nayebi, T. P. Barnwell, and M. J. T. Smith, "Low delay FIR filter banks: Design and evaluation," *IEEE Trans. Signal Processing*, vol. 42, pp. 24-31, Jan. 1994.

[39] A. Antoniou, *Digital Filters: Analysis, Design, and Applications*, 2nd Ed., McGraw-Hill, New York, 1993.

[40] C. Galand and H. J. Nussbaumer, "New quadrature mirror filter structures," *IEEE Trans. Acoust. Speech, Signal Processing*, vol. ASSP-32, pp. 522-529, June 1984.

[41] L. Ghazi, "Explicit formulae for lattice wave digital filters," *IEEE Trans. Circuits Syst.*, vol. 32, pp. 68-88, Jan. 1985.

[42] H. Nussbaumer, "Pseudo QMF filter bank," *IBM Tech. Disclosure Bull.*, vol. 24, pp. 3081-3087, Nov. 1981.

[43] J. Rothweiler, "Polyphase quadrature filters-A new subband coding technique," *Proc. IEEE Int. Conf. Acoust., Speech, Signal Processing*, Boston, MA, pp. 1280-1283, 1983.

[44] R. V. Cox, "The design of uniformly and nonuniformly spaced pseudo quadrature mirror filters," *IEEE Trans. Acoust., Speech, Signal Processing*, vol. ASSP-34, pp. 1090-1096, Oct. 1986.

[45] P. L. Chu, "Quadrature mirror filter design for an arbitrary number of equal bandwidth channels," *IEEE Trans. Acoust., Speech, Signal Processing*, vol. ASSP-33, pp. 203-218, Feb. 1985.

[46] P. P. Vaidyanathan, "Theory and design of M-channel maximally decimated quadrature mirror filters with arbitrary M, having the perfect-reconstruction property," *IEEE Trans. Acoust., Speech, Signal Processing*, vol. ASSP-35, pp. 476-492, Apr. 1987.

[47] M. J. T. Smith and T. P. Barnwell, III, "A new filter bank theory for time-frequency representation," *IEEE Trans. Acoust., Speech, Signal Processing*, vol. ASSP-35, pp. 314-326, Mar. 1987.

[48] R. D. Koilpillai and P. P. Vaidyanathan, "A spectral factorization approach to pseudo-QMF design," *Proc. IEEE Int. Symp. Circuits Syst.*, Singapore, pp. 160-163, May 1991.

[49] H. S. Malvar, "Modulated QMF filter banks with perfect reconstruction," *IEE Electron. Lett.*, vol. 26, pp. 906-907, June 1990.

[50] T. Q. Nguyen, "Near-perfect-reconstruction pseudo-QMF banks," *IEEE Trans. Signal Processing*, vol 42, pp. 65-76, Jan. 1994.

[51] P.-Q. Hoang and P. P. Vaidyanathan, "Non-uniform multirate filter banks: theory and design," *Proc. IEEE Int. Symp. Circuits Syst.*, Portland, OR, pp. 371-374, May 1989.

[52] S. Y. Kung, *VLSI Array Processors,* Prentice-Hall, Englewood Cliffs, New Jersey, 1988.

[53] H. T. Kung, "Why Systolic Architectures?," *IEEE Computer*, vol. 15, pp. 37-46, Jan. 1982.

[54] S. Sunder, *VLSI Implementation of Digital Filters*, Ph.D. Dissertation, Department of Electrical and Comp. Eng., University of Victoria, B.C., Canada, 1992.

[55] F. El-Guibaly, S. Sunder, and A. Antoniou,"Systolic implementation of FIR filters," *Signal Processing V: Theories and Applications*, L. Torres, E. Masgrau, and M. A. Lagunas (eds.), Elsevier, New York, pp. 1423-1426, 1990.

[56] S. Sunder, F. El-Guibaly, and A. Antoniou, "Systolic implementation of digital filters," *Multidimensional Systems and Signal Processing*, vol. 3, pp. 63-78, Mar. 1992.

[57] W. Luk and G. Jones, "Systolic recursive filters," *IEEE Trans. Circuits Syst.*, vol. CAS-35, pp. 1067-1068, Aug. 1988.

[58] S. K. Rao and T. Kailath, "Regular iterative algorithms and their implementation on processor arrays," *Proc. IEEE*, vol. 76, pp. 259-269, Mar. 1988.

[59] D. I. Moldovan, "On the analysis and synthesis of VLSI algorithms," *IEEE Trans. Comput.*, vol. C-31, pp. 1121-1126, Nov. 1982.

[60] P. Quinton, "The systematic design of systolic arrays," *IRISA Research Report*, No. 193, Apr. 1983.

[61] S. V. Rajobadhye, "Synthesizing systolic arrays with control signals from recurrence equations," *Distributed Computing*, vol. 3, pp. 88-105, 1989.

[62] M. G. Bellanger, G. Bonnerot, and M. Coudreuse, "Digital filtering by polyphase network: application to sample rate alternation and filter banks," *IEEE Trans. Acoust., Speech, Signal Processing*, vol. ASSP-24, pp. 109-114, Apr. 1976.

[63] R. Crochiere and L. R. Rabiner, *Multirate Digital Signal Processing*, Prentice Hall, Englewood Cliffs, New Jersey, 1983.

[64] G. Bi, F. P. Coakley, and B. G. Evans, "Rational sampling rate conversion structures with minimum delay requirements," *IEE Proc.*, Pt. E, vol. 139, pp. 477-485, Nov. 1992.

[65] C. -C Hsiao, "Polyphase filter matrix for rational sampling rate conversions," *Proc. IEEE Int. Conf. Acoust., Speech, Signal Processing*, Dallas, Texas, pp. 2173-2176, Apr. 1987.

[66] H. K. Kwan and T. S. Okullo-Oballa, "Systolic implementation of linear-phase FIR decimators and interpolators," *Proc. 31st Midwest Symp. Circuits Syst.*, St. Louis, Missouri, pp. 59-62, Aug. 1988.

[67] H. K. Kwan and T. S. Okullo-Oballa, "Systolic array implementation of a decimator and an interpolator," *IEE Proc.*, Pt. F, vol. 135, pp. 70-72, Jan. 1988.

[68] N. Petkov, "Mapping systolic FIR filter banks onto fixed-size linear processor arrays," *Proc. IEEE Int. Symp. Circuits Syst.*, NewOrleans, LA, pp. 1244-1247, May 1990.

[69] U. Pestel and K. Gröger, "Design of HDTV subband filterbanks considering VLSI constraints," *IEEE Trans. Circuits, Syst. for Video Tech.*, vol. 1, pp. 14-21, Mar. 1991.

[70] G. Privat and A. Wittmann, "Pipelined recursive filter architectures for subband image coding," *Integration*, vol. 14, pp. 361-379, 1993.

[71] M. Hatamian and G. L. Cash, "A 70-MHz 8-bit × 8-bit parallel pipelined multiplier in 2.5-μm CMOS," *IEEE J. Solid-State Circuits*, vol. SC-21, pp. 505-513, Aug. 1986.

[72] C. P. Lerouge, P. Girard, and J. S. Colardelle, "A fast 16 bit NMOS parallel multiplier," *IEEE J. Solid-State Circuits*, vol. SC-19, pp. 338-342, June 1984.

[73] S. Sunder, F. El-Guibaly, and A. Antoniou, "VLSI implementation of a second-order digital filter," *Can. J. Elec. & Comp. Eng.*, vol. 19, pp. 143-147, Mar. 1994.

[74] M. O. Ahmed and D. V. Poomalah, "Design of an efficient VLSI inner-product processor for real-time DSP applications," *IEEE Trans. Circuits Syst.*, vol. CAS-36, pp. 324-329, Feb. 1989.

[75] R. Fletcher, *Practical Methods of Optimization*, 2nd ed., Wiley, New York, 1987.

[76] The IMSL library: A set of FORTRAN subroutines for mathematics and statistics.

[77] S. Sunder and R. P. Ramachandran, "A least-squares design of nonrecursive filters satisfying prescribed magnitude and phase specifications," *Proc. IEEE Int. Symp. Circuits Syst.*, Chicago, Michigan, May 1993, pp. 335-338.

[78] T. Nguyen, "The design of arbitrary FIR digital filters using the eigenfilter method," *IEEE Trans. Signal Processing*, vol. 41, Mar. 1993, pp. 1128-1139.

[79] G. W. Stewart, *Introduction To Matrix Computation*, Academic Press, New York, 1973.

[80] E. Abdel-Raheem, F. El-Guibaly and A. Antoniou, "Design of low-delay FIR QMF banks using the Lagrange-multiplier approach," *IEE Electron. Lett.*, vol. 30, no. 12, pp. 924-926, June 1994.

[81] E. Abdel-Raheem, F. El-Guibaly and A. Antoniou, "Design of low-delay FIR QMF banks using the Lagrange-multiplier approach," *Proc. 37th Midwest Symp. Circuits Syst.*, Lafayette, Louisiana, pp. 1057-1060, Aug. 1994.

[82] E. Abdel-Raheem, F. El-Guibaly, and A. Antoniou, "Design of low-delay perfect-reconstruction FIR filter banks for tree-structured subband coders," *Proc. IEEE Int. Symp. Circuits Syst.*, Seattle, WA, pp. 969-972, Apr. 1995.

[83] C. L. Lawson, *Contributions to the Theory of Linear Least Maximum Approximations*, Ph.D. Dissertation, U.C.L.A., 1961.

[84] V. R. Algazi, M. Suk, and C.-S. Rim, "Design of almost minimax FIR filters in one and two dimensions by WLS techniques," *IEEE Trans. Circuits Syst.*, vol. CAS-33, pp. 590-596, June 1986.

[85] Y. C. Lim, C. K. Chen, and R.-H. Yang, "A weighted least squares algorithm for quasi equiripple FIR and IIR digital filter design," *IEEE Trans. Signal Processing*, vol. 40, pp. 551-558, Mar. 1992.

[86] S. Sunder and V. Ramachandran, "Design of equiripple nonrecursive differentiators and Hilbert transformers using a weighted least-squares technique," *IEEE Trans. Signal Proc.*, vol. 42, pp. 2504-2509, Sept. 1994.

[87] MATLAB, *The Math Works, Inc.*, South Natick, MA.

[88] H. K. Kwan and T. S. Okullo-Oballa, "Systolic realization of linear-phase FIR digital filters," *IEEE Trans. Circuits Syst.*, vol. CAS-34, pp. 1604-1605, Dec. 1987.

[89] E. Abdel-Raheem, F. El-Guibaly, and A. Tawfik, "Systolic implementation of linear-phase FIR filters," *Proc. Canadian Conf. Elec., Comp. Eng.*, Vancouver, B.C., pp. 680-683, Sept. 1993.

[90] E. Abdel-Raheem, F. El-Guibaly, and A. Antoniou, "VLSI array processors for linear-phase FIR filters," *Proc. 28th Asilomar Conf. Signals, Syst., Comp.*, Pacific Grove, California, pp. 1036-1040, Oct. 1994.

[91] E. Abdel-Raheem, F. El-Guibaly, and A. Antoniou, "VLSI array processors for linear-phase FIR filters," *Canadian J. Elec. & Comp. Eng.*, vol. 20, pp. 73-77, Apr. 1995.

[92] A. Oppenheim and R. Schafer, *Discrete-Time Signal Processing*, Prentice-Hall, Englewood Cliffs, New Jersey, 1989.

[93] N. H. E. Weste and K. Eshraghian, *Principles of CMOS VLSI Design*, 2nd Ed., Addison-Wesley Reading, MA, 1993.

[94] A. Tawfik, F. El-Guibaly, M. Fahmi, E. Abdel-Raheem, and P. Agathoklis, "High-Speed Area-Efficient Inner-Product Processor," *Canadian J. Elec. & Comp. Eng.*, vol. 19, pp. 187-191, Oct. 1994.

[95] E. Abdel-Raheem, A. Tawfik, M. Fahmi, and F. El-Guibaly, "A new inner-product processor for FIR filter implementation," *Proc. IEEE Pac. Rim*

Conf. Comm., Comp., Signal Processing, Victoria, B.C., pp. 395-398, May 1995.

[96] K. Hwang, *Computer Arithmetic: Principles, Architecture. and Design*, Wiley, New York, 1979.

[97] E. Abdel-Raheem, F. El-Guibaly, and A. Antoniou, "Systolic implementation of polyphase decimators and interpolators," *Proc. 37th Midwest Symp. Circuits Syst.*, Lafayette, Louisiana, pp. 749-752, Aug. 1994.

[98] E. Abdel-Raheem, F. El-Guibaly, and A. Antoniou, "Efficient decimator and interpolator array structures," *J. Circuits Syst. Comp.*, vol. 5, No. 2, (in press).

[99] E. Abdel-Raheem, F. El-Guibaly, and A. Antoniou, "Systolic implementation of FIR decimators and interpolators," *Proc. 36th Midwest Symp. Circuits Syst.*, Detroit, Michigan, pp. 938-941, Aug. 1993.

[100] E. Abdel-Raheem, F. El-Guibaly, and A. Antoniou, "Systolic implementation of FIR decimators and interpolators," *IEE Proc.-Circuits Devices Syst.*, vol. 141, pp. 489-492, Dec. 1994.

[101] J. Shynk, "Frequency-domain multirate adaptive filtering," *IEEE Signal Processing Magazine*, vol. 9, pp. 14-35, Jan. 1992.

[102] S. A. White, "Applications of distributed arithmetic to digital signal processing: a tutorial review," *IEEE Signal Processing Magazine*, pp. 4-19, July 1989.

[103] Xilinx Inc., *The Programmable Gate Array Data Book*, 1992.

VITA

Surname: Abdel-Raheem

Given Name: Esam

Place of Birth: Alexandria, Egypt

Date of Birth: November 26, 1961

Educational Institutions Attended:

University of Victoria, Canada	1991 to 1995
University of Ottawa, Canada	1990 to 1990
Ain Shams University, Egypt	1985 to 1989
Ain Shams University, Egypt	1979 to 1984

Degrees Awarded:

M.Sc.	Ain Shams University	1989
B.Sc.(Hon.)	Ain Shams University	1984

Honours and Awards:

Entrance Scholarship	University of Ottawa	1990
B.Sc. (Hon.)	Ain Shams University	1984
Egyptian Government Excellence Award	Ain Shams University	1980-84

Publications:

Refereed journal publications

1. E. Abdel-Raheem, F. El-Guibaly, and A. Antoniou, "Design of low-delay two-channel FIR filter banks using constrained optimization", submitted to *Signal Processing*, May 1994.

2. E. Abdel-Raheem, F. El-Guibaly and A. Antoniou, "Efficient decimator and interpolator array structures", *Journal of Circuits, Syst. and Comp.*, vol. 5, No. 2, (in press).

3. E. Abdel-Raheem, F. El-Guibaly, and A. Antoniou, "VLSI array processors for linear-phase FIR filters", *Canadian J. Elec. & Comp. Eng.*, vol. 20, pp. 73-77, Apr. 1995.

4. E. Abdel-Raheem, F. El-Guibaly, and A. Antoniou, "Systolic implementation of FIR decimators and interpolators", *IEE Proc.-Circuits Devices Syst.*, vol. 141, pp. 489-492, Dec. 1994.

5. A. Tawfik, F. El-Guibaly, M. Fahmi, E. Abdel-Raheem, and P. Agathoklis, "High-speed area-efficient inner-product processor", *Canadian J. Elec. & Comp. Eng.*, vol. 19, pp. 187-191, Oct. 1994.

6. E. Abdel-Raheem, F. El-Guibaly and A. Antoniou, "Design of low-delay FIR QMF banks using the Lagrange-multiplier approach", *IEE Electron. Lett.*, vol. 30, no. 12, pp. 924-926, June 1994.

Refereed conference publications

1. E. Abdel-Raheem, F. El-Guibaly, and A. Antoniou, "Weighted minimax design of QMF banks: results and comparisons", submitted to *Proc. 38th Symp. Circuits Syst.*, Rio de Janeiro, Brazil, Aug. 1995.

2. E. Abdel-Raheem, A. Tawfik, M. Fahmi, and F. El-Guibaly, "A new inner-product processor for FIR filter implementation", *Proc. IEEE Pac. Rim Conf. Comm., Comp., Signal Processing*, Victoria, B.C., pp. 395-398, May 1995.

3. E. Abdel-Raheem, F. El-Guibaly, and A. Antoniou, "Efficient design of cosine-modulated filter banks", *Proc. IEEE Pac. Rim Conf. Comm., Comp., Signal Processing*, Victoria, B.C., pp. 387-390, May 1995.

4. E. Abdel-Raheem, F. El-Guibaly, and A. Antoniou, "Design of low-delay perfect-reconstruction FIR filter banks for tree-structured subband coders", *Proc. IEEE Int. Symp. Circuits, Syst.*, Seattle, WA, pp. 969-972, April 1995.

5. E. Abdel-Raheem, F. El-Guibaly, and A. Antoniou, "VLSI array processors for linear-phase FIR filters", *Proc. 28th Asilomar Conf. Signals, Syst., Comp.*, Pacific Grove, California, USA, pp. 1036-1040, Oct. 1994.

6. E. Abdel-Raheem, F. El-Guibaly, and M. Fahmi, "A new array processor implementation for high-speed linear-phase FIR filters", *Proc. Int. Conf. Microelectronics*, Istanbul, Turkey, pp. 51-54, Sept. 1994.

7. E. Abdel-Raheem, F. El-Guibaly and A. Antoniou, "Design of low-delay FIR QMF banks using the Lagrange-multiplier approach", *Proc. 37th Midwest Symp. Circuits, Syst.*, Lafayette, Louisiana, pp. 1057-1060, Aug. 1994.

8. E. Abdel-Raheem, F. El-Guibaly, and A. Antoniou, "Systolic implementation of polyphase decimators and interpolators", *Proc. 37th Midwest Symp. Circuits, Syst.*, Lafayette, Louisiana, pp. 749-752, Aug. 1994.

9. F. El-Guibaly, A. Tawfik, E. Abdel-Raheem, and M. Fahmi, "Systolic arrays for 2-D FIR filters using a computational geometric approach", *Proc. Int. Conf. Microelectronics*, Dhahran, Saudi Arabia, pp. 185-188, Dec. 1993.

10. A. Tawfik, F. El-Guibaly, E. Abdel-Raheem, and P. Agathoklis, "A novel systolic implementation of fixed-point state-space digital filter", *Proc. Int. Conf. Microelectronics*, Dhahran, Saudi Arabia, pp. 177-180, Dec. 1993.

11. M. Fahmi, F. El-Guibaly, E. Abdel-Raheem, A. Tawfik, and D. Shpak, "Area-time efficient fixed point multiplier-accumulators for inner-product computation", *Proc. Int. Conf. Microelectronics*, Dhahran, Saudi Arabia, pp. 189-189, Dec. 1993.

12. F. El-Guibaly, A. Tawfik, E. Abdel-Raheem, M. Fahmi, and W.-S. Lu, " 2-D IIR filter architectures using computational geometry concepts ", *Proc. Canadian Conf. VLSI'93*, Banff, Alta., pp. 7-22-7-27, Nov. 1993.

13. E. Abdel-Raheem, F. El-Guibaly, and A. Tawfik, "Systolic implementation of linear-phase FIR filters", *Proc. Canadian Conf. Elec., Comp. Eng.*, Vancouver, B.C., pp. 680-683, Sept. 1993.

14. E. Abdel-Raheem, F. El-Guibaly, and A. Antoniou, "Systolic implementation of fractional decimators and interpolators", *Proc. Canadian Conf. Elec., Comp. Eng.*, Vancouver, B.C., pp. 692-695, Sept. 1993.

15. E. Abdel-Raheem, F. El-Guibaly, and A. Antoniou, "Systolic implementation of FIR decimators and interpolators", *Proc. 36th Midwest Symp. Circuits, Syst.*, Detroit, Michigan, pp. 938-941, Aug. 1993.

16. E. Abdel-Raheem, F. El-Guibaly, and A. Antoniou, "QMF banks: design and implementation perspectives", *Proc. IEEE Pac. Rim Conf. Comm., Comp., Signal Processing*, Victoria, B.C., pp. 354-357, May 1993.

17. E. Abdel-Raheem, and M. M. Ibrahim, "Applications of efficient scene matching algorithms for satellite images: a comparison," *Proc. Canadian Conf. Elec., Comp. Eng.*, Toronto, Ont., pp. MM3.4.1-3.4.4., Sept. 1992.

18. E. Abdel-Raheem, A. Shahin and S. Mahrous, "Analogy of different matching methods for satellite imagery," *Proc. Aeronautical Sci. Aviation Tech. (ASAT) Conf.*, MTC, Cairo, Egypt, pp. 91-99, Apr. 1989.

Other publications

1. F. El-Guibaly, A. Tawfik, W.-S. Lu, and E. Abdel-Raheem, "Synthesising systolic arrays: basic tools", *Technical Report VMC-92-2*, ECE Dept., University of Victoria, B.C., 1992.

2. E. Abdel-Raheem, *Microcomputer Applications in Scene Matching*, M.Sc. Thesis, Electronics and Computer Engineering Dept., Ain Shams University, Cairo, Jan. 1989.

PARTIAL COPYRIGHT LICENSE

Title of Thesis :

Design and VLSI Implementation of Multirate Filter Banks

Author: _____

Esam Mostafa M. Abdel-Raheem

July 7, 1995