

INFORMATION TO USERS

This manuscript has been reproduced from the microfilm master. UMI films the text directly from the original or copy submitted. Thus, some thesis and dissertation copies are in typewriter face, while others may be from any type of computer printer.

The quality of this reproduction is dependent upon the quality of the copy submitted. Broken or indistinct print, colored or poor quality illustrations and photographs, print bleedthrough, substandard margins, and improper alignment can adversely affect reproduction.

In the unlikely event that the author did not send UMI a complete manuscript and there are missing pages, these will be noted. Also, if unauthorized copyright material had to be removed, a note will indicate the deletion.

Oversize materials (e.g., maps, drawings, charts) are reproduced by sectioning the original, beginning at the upper left-hand corner and continuing from left to right in equal sections with small overlaps.

ProQuest Information and Learning
300 North Zeeb Road, Ann Arbor, MI 48106-1346 USA
800-521-0600

UMI[®]

A Computer Aided Design System for Transformation and Optimization of Multirate DSP Systems

Qing Ma

A Thesis
in
The Department
of
Electrical and Computer Engineering

Presented in Partial Fulfillment of the Requirements
for the Degree of Master of Applied Science at
Concordia University
Montréal, Québec, Canada

March 2003

© Qing Ma, 2003



**National Library
of Canada**

**Acquisitions and
Bibliographic Services**

**395 Wellington Street
Ottawa ON K1A 0N4
Canada**

**Bibliothèque nationale
du Canada**

**Acquisitions et
services bibliographiques**

**395, rue Wellington
Ottawa ON K1A 0N4
Canada**

Your file Votre référence

Our file Notre référence

The author has granted a non-exclusive licence allowing the National Library of Canada to reproduce, loan, distribute or sell copies of this thesis in microform, paper or electronic formats.

The author retains ownership of the copyright in this thesis. Neither the thesis nor substantial extracts from it may be printed or otherwise reproduced without the author's permission.

L'auteur a accordé une licence non exclusive permettant à la Bibliothèque nationale du Canada de reproduire, prêter, distribuer ou vendre des copies de cette thèse sous la forme de microfiche/film, de reproduction sur papier ou sur format électronique.

L'auteur conserve la propriété du droit d'auteur qui protège cette thèse. Ni la thèse ni des extraits substantiels de celle-ci ne doivent être imprimés ou autrement reproduits sans son autorisation.

0-612-77685-9

Canada

ABSTRACT

A Computer Aided Design System for Transformation and Optimization of Multirate DSP systems

Qing Ma

Multirate Digital Signal Processing (DSP) theory and technique are essential to digital communications, sonar and radar systems, speech and image processing, as well as many other applications. A computer aided design system that can significantly simplify the design and optimization of multirate signal processing structures has been presented in this thesis.

The thesis starts with the introduction of fundamentals of multirate digital signal processing. It is followed by the introduction of the principle and technique of Multirate Signal Flow Graph (MSFG). A number of MSFG identities and transformations are also given in the thesis.

The computer aided design system that has been presented in this thesis is based on the MSFG representation and transformations. With this system, instead of manual manipulation of MSFGs, the transformations and optimization of MSFGs that represent multirate signal processing systems can be performed in an automatic or interactive manner.

With this system, one can represent a multirate system in the form of a MSFG and then optimizes it by performing a series of MSFG transformations interactively. The system has the capability of calculating the output response at any CELL in the MSFG. This feature can be used to verify the correctness of the derived network.

To perform the MSFG transformation automatically or interactively, following functions have been implemented in the system.

- MSFG transformation and identities
- Simulation of MSFG response
- Verification of the transformed MSFG
- Complexity information of MSFG

To demonstrate the usefulness and the effectiveness of the system, some design examples have been given in chapter 5. In these examples, step-by-step MSFG transformations are provided, which would otherwise be very difficult or extremely tedious to derive.

Dedicated to my wife and my daughter ...

/

ACKNOWLEDGEMENTS

I would like to thank my academic advisor, Dr. Ronggang Qi and co-supervisor Dr. M. Reza Soleymani, for the invaluable support, guidance and encouragement they have provided over the past year.

I would like to thank all my friends who have provided me all kinds of help in my graduate study.

Most of all, I would like to thank my family for their love and encouragement. I could not have completed my degree without their continuous and immeasurable support.

TABLE OF CONTENTS

LIST OF FIGURES.....	x
LIST OF TABLES.....	xii
LIST OF ACRONYMS.....	xiii
<i>Chapter 1</i> Introduction	1
1.1 Background	1
1.2 Classical Signal Flow Graph and Multirate Signal Flow Graph	3
1.3 MSFG Transformation	4
1.4 A Computer Aided Design System for Transformation and Optimization of Multirate DSP systems	5
<i>Chapter 2</i> Theory of Multirate Digital Signal Processing	7
2.1 Band-limited signals.....	7
2.1.1 Lowpass signals and bandpass signals	7
2.1.2 Analytic signals and Hilbert transform	8
2.1.3 Frequency translations and bandpass signals	9
2.2 Multirate systems and signal processing.....	11
2.2.1 Discrete signal representations.....	11
2.2.2 Sampling rate alteration	12
2.2.3 Rational sampling rate conversion	14
<i>Chapter 3</i> Theory of Multirate Signal Flow Graph Transformation.....	21
3.1 MSFG	21
3.2 MSFG Operators	22
3.2.1 Fundamental MSFG operators	22
3.2.2 Basic MSFG operators	23
3.2.3 MIMO MSFG operators.....	26
3.3 MSFG transformation	27
3.4 Superposition theorem.....	28
3.5 The noble identities	28

3.6	General identities.....	29
3.6.1	Complex filter identities.....	29
3.6.2	Modulation identities.....	30
3.6.3	Identities associated with composite rate-changing operators	31
3.7	Polyphase decomposition.....	32
3.7.1	Filter Polyphase decomposition	32
3.7.2	Modulation polyphase decomposition	33
3.8	Cascade of operators	36
3.8.1	Cascade of samplers.....	36
3.8.2	Cascade of commutators	38
3.8.3	Commutator-modulator cascades.....	40
3.8.4	Commutator-filter cascades.....	42
3.8.5	Commutator-sampler commutability	43
<i>Chapter 4</i>	<i>A Computer Aided Design System for Transformation and Optimization of Multirate DSP systems.....</i>	<i>45</i>
4.1	Objectives and requirements of the computer aided design systems.....	45
4.2	Operation System and Programming Language.....	47
4.3	Data Structure.....	47
4.3.1	CELL Graph.....	49
4.3.2	CELL.....	50
4.3.3	DATA.....	51
4.3.4	Complexity Information.....	52
4.3.5	MSFG Operator.....	52
4.4	System Block Diagram.....	59
4.5	Class Model.....	60
4.5.1	Class Model for general application program with MFC	60
4.5.2	Class Model for this computer aided design system.....	62
4.6	Interface.....	67
4.6.1	User Menu	67
4.6.2	Dialog Window	70
4.6.3	File Input, Output and File Structure	72

4.7	MSFG Transformations.....	73
4.8	Simulation and Verification of MSFG Transformation	74
4.9	Complexity of MSFG.....	75
<i>Chapter 5</i>	<i>Application Examples</i>	<i>77</i>
5.1	Five-channel Frequency Demultiplexer (DEMUX).....	77
5.1.1	MSFG transformation	79
5.1.2	Simulation and Verification	83
5.2	The optimal complex BSF structure for binary tree DEMUX	84
5.2.1	MSFG transformation	84
5.2.2	Simulation and Verification	89
<i>Chapter 6</i>	<i>Conclusion and future work.....</i>	<i>92</i>
6.1	Conclusion.....	92
6.2	Future Work	93
References	94

LIST OF FIGURES

Figure 1-1 Basic MSFG Operators.....	4
Figure 2-1 Lowpass signals.....	8
Figure 2-2 Bandpass signals.....	9
Figure 2-3 Spectra relation for integer rate alteration: (a) to (e) for integer rate decimation ($M=3$); (f) to (j) for integer rate interpolation ($L=3$).....	14
Figure 2-4 Rational sampling rate conversion:	16
Figure 2-5 Polyphase analysis of signal	18
Figure 2-6 Polyphase synthesis of signal	18
Figure 2-7 Transposition of LTI polyphase network	19
Figure 2-8 Polyphase decimation filter.....	20
Figure 2-9 Polyphase interpolation filter	20
Figure 3-1 DFT.....	26
Figure 3-2 Interleaver.....	26
Figure 3-3 Superposition theorem.....	28
Figure 3-4 Noble identities.....	29
Figure 3-5 Complex filtering identities.....	30
Figure 3-6 Modulation identities.....	31
Figure 3-7 Identities associated with composite rate-changing operators	32
Figure 3-8 Polyphase decomposition transforms (PDT).....	33
Figure 3-9 Two types of MPDT structures in conventional SFG	35
Figure 3-10 MSFGs of type 1 and type 2 MPDT.....	35
Figure 3-11 MPDT for complex modulation	36
Figure 3-12 Cascade of samplers	37
Figure 3-13 Cascades of Commutators	39
Figure 3-14 Commutator-SPC and SPC-commutator identities	41
Figure 3-15 Commutator-filter identities	42
Figure 3-16 Sampler-commutator identities	44
Figure 4-1 Description of an example in MSFG and DSP block diagram.....	48

Figure 4-2 CELL Graph Description	49
Figure 4-3 Default value of Operator	53
Figure 4-4 Operators Description.....	59
Figure 4-5 System Block Diagram.....	60
Figure 4-6 Classes Model and Relationship.....	62
Figure 4-7 Classes Model.....	63
Figure 4-8 Main Menu	68
Figure 4-9 MSFG Transformation Submenu	70
Figure 4-10 MSFG Transformation Tools Bar (Figure Menu)	70
Figure 4-11 Some Dialog Windows.....	71
Figure 4-12 Serialization	72
Figure 4-13 SFG file structure	72
Figure 4-14 Simulation and Verification Menu	74
Figure 4-15 Simulation and Verification result figure	75
Figure 4-16 Complexity information of MSFG	76
Figure 5-1 Five-channel MF-TDMA signal and the demultiplexing channel filter.....	78
Figure 5-2 DEMUX function (lowpass model).....	78
Figure 5-3 Step-by-step simplification for demultiplexing the k -th channel	81
Figure 5-4 Polyphase DFT structure for the 5-channel DEMUX	82
Figure 5-5 DSP block diagram.....	82
Figure 5-6 Simulation and Verification of DEMUX	84
Figure 5-7 Direct complex BSF structure	85
Figure 5-8 Derivation for the optimal complex BSF	89
Figure 5-9 DSP block diagram.....	89
Figure 5-10 Simulation and Verification result.....	91

LIST OF TABLES

Table 3-1 Fundamental MSFG operators.....	23
Table 3-2 Basic MSFG Operators.....	25
Table 3-3 Extended MSFG Operators.....	27
Table 4-1 CELL Class.....	51
Table 4-2 Classes derived from CObject	64
Table 4-3 Classes derived from CDialog	65
Table 4-4 Functions in CSignalFlowView.....	65
Table 4-5 Functions in CSignalFlowDoc.....	66

LIST OF ACRONYMS

ADD	Adder
API	Application Program Interface
ATM	Asynchronous Transfer Mode
DDC	Digital Down-Converter
DELAY	Delay element
DEMUX	Demultiplexer
DRFU	Down-Real Filter-Up
DS	Down Sampler
DSB/SC	Double-SideBand Suppressed Carrier modulation
DSP	Digital Signal Processing
FDM	Frequency Division Multiple Access
FFT	Fast Fourier Transform
FIL	Filter
FIR	Finite Impulse Response
GCD	Great Common Divisor
GDFT	Generalized Discrete Fourier Transform
GSM	Global System for Mobile Communication
IDU	Integral & Dump
INTL	Interleave
LPTV	Linear Periodically Time-Varying
LTI	Linear Time Invariant
MAC	Multiplication and Accumulation
MF-TDMA	Multi-Frequency Time Division Multiple Access
MFC	Microsoft Foundation Class Library
MIMO	Multi-Input Multi-Output
MOD	Modulator
MPDT	Modulation Polyphase Decomposition Transform
MSDN	Microsoft Develop Network

MSFG	Multirate Signal Flow Graph
MUL	Multiplier
NULL	Null operator
OOP	Object Oriented Programming
OSPC	Overlap SPC Commutator
PSC	Parallel-to-Serial Commutator
SA	Sampler
SFG	Signal Flow Graph
SH	Sampling & Hold
SPC	Serial-to-Parallel Commutator
SSB	Single-SideBand modulation
SW	Switch
US	Up Sampler
USH	Upsampling & hold

Introduction

1.1 Background

Multirate Digital Signal Processing (DSP) technique is one of the most important techniques that have been widely used in designing and implementing digital communications systems. It is particularly useful when signal sampling rate is high or the required processing rate is close to or higher than the limits of given technology, for example, in some high speed digital receivers and digital modem applications. Multirate DSP techniques have been applied in a wide variety of areas, of signal processing including: communication systems [5], speech and audio processing systems [9], antenna systems [16], and Radar systems [12]. Its typical applications in communication system include sub-band coding [28], sampling rate conversion [3], transmultiplexing as in telephony systems [26] and frequency demultiplexing [8].

A major problem of multirate DSP is the high computational complexity when the information bandwidth is much smaller than signal sampling rate. An optimal multirate system has the processing bandwidth equal or close to the information bandwidth. In fact, one of the primary concerns in multirate DSP is how to reduce the processing bandwidth at different points of the system in order to minimize the overall computational complexity. A well-known and profound result in multirate DSP is the development of the critically decimated or interpolated polyphase filter banks [3,4], which are optimal in the sense that the computational complexity is minimized.

In practice, the frequently used approach to multirate system optimization is to directly apply such existing optimized multirate structures as polyphase filters and polyphase FFT (Fast Fourier Transform) filter banks to functional blocks in multirate

systems. Systems optimized in this way are often not globally optimal because the optimizations are performed locally.

Although multirate system optimization could be achieved by using mathematical representation and manipulations, this mathematical method is difficult to manifest structural information during the reduction process. Its another disadvantage is that it can be extremely tedious and error-prone.

Another method of multirate system optimization is use conventional Signal Flow Graph (SFG) representation and transformations [14,23] to simplify a multirate network. However, the conventional SFG, which is valid only for Linear Time Invariant (LTI) systems, provides little help to handle multirate operations, which are periodical time varying in nature. Generally speaking, this method can only be used to solve very simple problems.

With the above conventional approaches, it is very difficult to optimize even moderate complex systems such as the non-maximally decimated filter banks and filter banks with rational decimation factor, which are often encountered in frequency demultiplexer designs. Global optimization of multirate systems is very often unaddressed, largely due to lack of systematic methods for multirate system design and optimizations.

As a natural extension of the conventional SFG, MSFG has been proposed to represent and to optimize multirate systems [17,18]. Multirate system optimizations can be achieved by applying a series of MSFG identities and transformations to multirate networks to minimize the overall computations. This approach shows structural information explicitly throughout the reduction process. Using this method, a number of efficient filter bank structures have been successfully derived and some of them have been actually implemented in telecommunications industry [19].

However, MSFG approach also has its shortcomings. Manipulation of MSFGs is still a tedious and very error-prone task because manual manipulations of MSFGs are required throughout the process of the MSFG transformation. This has been the main

motivation for the development of a computer-aided design system for transformation and optimization of Multirate DSP systems. Having performed extensive search on this subject, we have found that, as far as we know, no one else has ever developed or attempted to develop such a system for optimization of multirate signal processing systems. Therefore, if such a system can be successfully developed, it could even evolve into a commercial product in the future. This has been our another motivation to develop such a system.

The work presented in this thesis is focused on the development of this computer aided design system. The system can be designed to facilitate the design and optimization of multirate signal processing systems through MSFG transformations. With this system, MSFG transformations and optimization can be performed interactively or automatically without manual manipulations of MSFGs. It can even automatically verify the transformation results. Hence, the design system can significantly simplify the process of MSFG transformations.

1.2 Classical Signal Flow Graph and Multirate Signal Flow Graph

Classical SFG is a single rate SFG. It includes three basic operators: multiplier (MUL), adder (ADD) and delay element (DELAY).

MSFG is an extension of the conventional SFG. Besides the above three basic single rate SFG operators, there are two more basic operators in MSFG, the downsampler (DS) and the upsampler (US). These five basic operators in MSFG are shown in Figure 1-1.

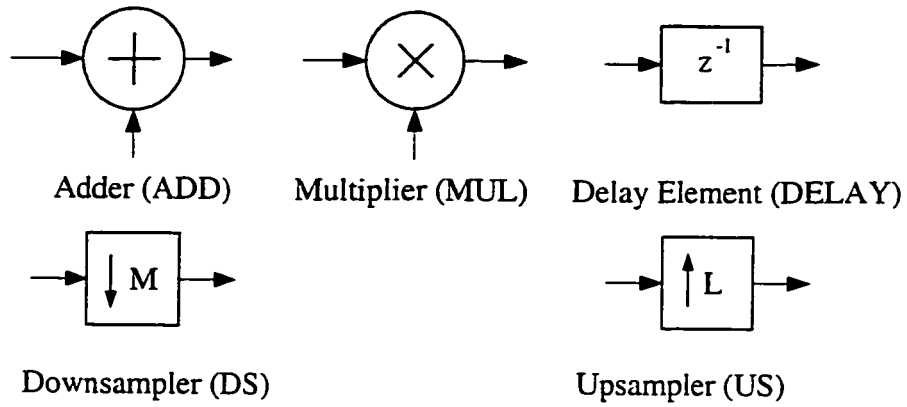


Figure 1-1 Basic MSFG Operators

From these basic operators, we further define 13 composite MSFG operators. Therefore, we have defined a total of 18 different MSFG operators for various multirate signal processing operations in MSFG. These 18 operators are Modulator (MOD), Filter (FIL), Downsampler (DS), Upsampler (US), Serial-to-Parallel Commutator (SPC), Parallel-to-Serial Commutator (PSC), Adder (ADD), Multiplier (MUL), Null Operator (NULL), Sampling & Hold (SH), Upsampling & Hold (USH), Integral & Dump (IDU), Sampler (SA), Delay element (DELAY), Discrete Fourier Transform (DFT), Interleaver (INTL), Switch (SW) and Overlap SPC Commutator (OSPC).

1.3 MSFG Transformation

A multirate DSP system can be transformed into its equivalencies with different structures via MSFG transformations. Some fundamental relationships between multirate operators have been given in the form of identity (noble identities, see section 3.5). These MSFG transformations are classified in four categories:

- *The noble identities*
- *General identities*
- *Polyphase decomposition*
- *Cascade of operators*

1.4 A Computer Aided Design System for Transformation and Optimization of Multirate DSP systems

The main objective of this study is to develop a software system, automatically or interactively, to perform MSFG transformations and get one or more optimized MSFG under given optimization constraints. It needs a friendly user interface to make the MSFG transformation and optimization an easy task.

With this design system, one can represent a multirate system in the form of a MSFG and then optimize it by performing a series of MSFG transformations interactively. The system has the capability of calculating the output response at any CELL in the MSFG. It therefore can verify the correctness of the derived network by comparing the response of the derived with that of the original MSFG.

To perform the MSFG transformation automatically or interactively, the following foundational functions have been implemented in the system.

- *MSFG transformation and identities*
- *Simulation of MSFG response*
- *Verification of the transformed MSFG*
- *Complexity information of MSFG*

C++ is an object oriented programming (OOP) language. In OOP, the structure and its functions are combined into a single entity called Class. The objects are created from Classes. OOP has the following advantages: simplicity, modularity, modifiability, extensibility, flexibility, maintainability and reusability.

MFC (Microsoft Foundation Class) Library is a large and extensive C++ class hierarchy that makes the development of Windows applications significantly easier. MFC is compatible across the entire Windows family. MFC class hierarchy encapsulates the user interface portion of the Windows Application Program Interface (API), and makes it significantly easier to create Windows applications in an object oriented way. This

hierarchy is available for and compatible with all versions of Windows. Consequently, the code created in MFC is portable.

This computer aided design system is based on MSFG representation and transformations. It is developed using Visual C++ language on the Microsoft Windows operation system. An OOP method based on MFC (Microsoft Foundation Class Library) has been used in the development of the system. The design system has a user-friendly GUI interface, and is easy to use. It allows users to perform MSFG transformations, optimization and verification of multirate signal processing systems in a visual operation mode.

Theory of Multirate Digital Signal Processing

Signals that exist in the real world are always band-limited. Digital signal processing for bandpass signals requires sampling rate alteration in order to reduce computations. A system like this is called a multirate system. To deal with multirate DSP systems efficiently, multirate DSP techniques have evolved [4,27]. Multirate DSP techniques are useful in digital telecommunication systems for handling multiple data transmission rates. This chapter will briefly introduce multirate digital signal processing fundamentals.

2.1 Band-limited signals

2.1.1 Lowpass signals and bandpass signals

Signals in transmission systems are always band-limited with finite bandwidth. A signal is **lowpass** if its significant spectral content is centered around zero frequency ($f=0$), which is shown in Figure 2-1(a) and Figure 2-1(b). For real lowpass signal, its magnitude spectrum must be even symmetric and the phase must be odd symmetric about $f=0$ as shown in Figure 2-1 (a). For complex signals (also referred as complex envelopes) shown in Figure 2-1(b) and 2.1(c), symmetry properties of spectrum are destroyed. **Bandpass** signals in the real world can be interpreted as being translated from their equivalent lowpass signals, which are often referred as **baseband** signals. Efficient signal processing tends to move the bulk of the processing load to baseband.

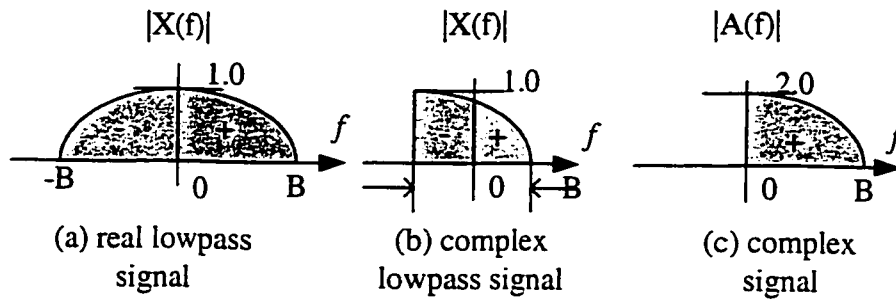


Figure 2-1 Lowpass signals

2.1.2 Analytic signals and Hilbert transform

A signal is *analytic* if, and only if its Fourier transform is zero for negative frequencies ($f < 0$), as for those illustrated in Figure 2-2(c) and Figure 2-2(d). Given a real signal $x(t)$ with Fourier transform $\mathcal{F}\{x(t)\} = X(\omega)$, its analytic signal $a_x(t)$ is defined, in frequency domain, by

$$\begin{aligned}
 A_x(\omega) &= 2X(\omega)u(\omega) \\
 &= X(\omega)[1 + \text{sgn}(\omega)] \\
 &= X(\omega) + X(\omega)\text{sgn}(\omega) \\
 &= \begin{cases} 2X^+(\omega), & \text{for } \omega \geq 0 \\ 0, & \text{for } \omega < 0 \end{cases}
 \end{aligned} \tag{2.1}$$

where $u(\omega)$ is the unit step function and $X^+(\omega)$ is the right (positive) sideband of $X(\omega)$. The non-symmetric property of $A_x(\omega)$ determines that an analytic signal must be complex.

Signals shown in Figure 2-1(c) and Figure 2-2(d) are analytic with a single sideband while the complex signal of Figure 2-2(c) can be considered as double-sideband analytic with respect to the real bandpass signal of Figure 2-2(a).

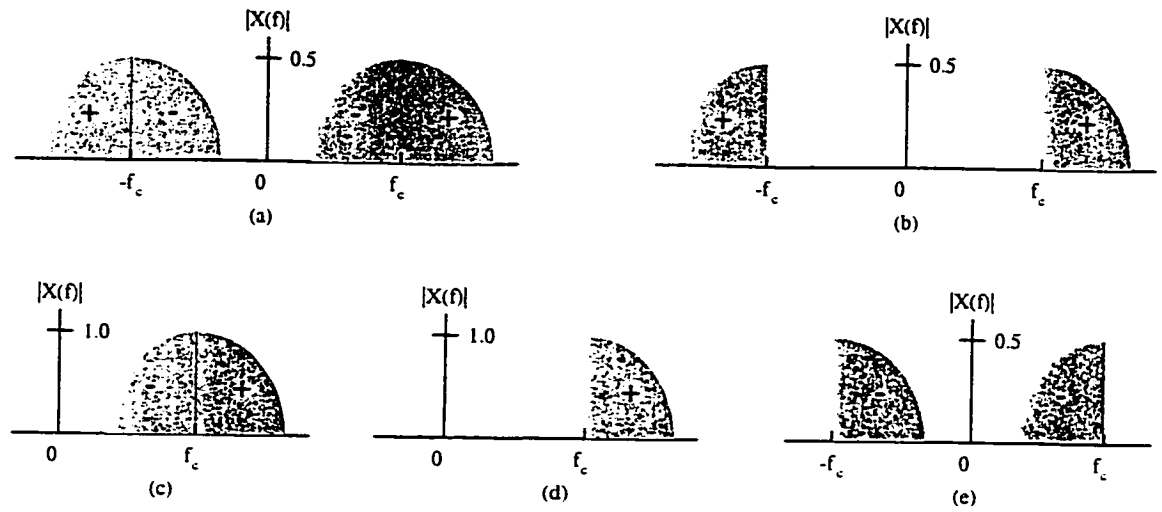


Figure 2-2 Bandpass signals

Eq. (2.1) shows that the analytic signal $A_x(\omega)$ of a real signal $X(\omega)$ is obtained by adding signal

$$\hat{X}(\omega) = X(\omega) \text{sgn}(\omega) \quad (2.2)$$

to the original real signal $X(\omega)$. Eq. (2.2) is known as the Hilbert transform of $x(t)$ and it is denoted by

$$\mathcal{H}\{x(t)\} = \hat{x}(t) \quad (2.3)$$

Hence the analytic signal $a_x(t)$ and the real signal $x(t)$ are related by Hilbert transform:

$$a_x(t) = x(t) + j\hat{x}(t) \quad (2.4)$$

We sometimes refer to the real and complex lowpass signals respectively as the real baseband and the complex baseband signals with respect to their modulated (frequency shifted) bandpass associate as shown in Figure 2-2.

2.1.3 Frequency translations and bandpass signals

Frequency translation shifts the signal spectrum from one frequency band to the other without changing the shape. Frequency shift in the continuous time domain is simply a multiplication by a continuous sinusoid either real or complex. For an arbitrary complex

signal $x(t) = x_r(t) + jx_i(t)$ with Fourier transform $X(\omega)$, the following three types of frequency translations are defined:

$$x(t)e^{j\omega_c t} \longleftrightarrow X(\omega - \omega_c) \quad (2.5)$$

$$x(t)\cos(\omega_c t) \longleftrightarrow \frac{1}{2}X(\omega + \omega_c) + \frac{1}{2}X(\omega - \omega_c) \quad (2.6)$$

$$\Re\{x(t)e^{j\omega_c t}\} \longleftrightarrow \frac{1}{2}X^*(-\omega - \omega_c) + \frac{1}{2}X(\omega - \omega_c) \quad (2.7)$$

The transform pair of Eq. (2.5) is called *one-sided frequency translation* as the signal spectrum is moved only in one direction. The complex bandpass signal of Figure 2-2(c) is the direct result of a one-sided translation of the real lowpass signal of Figure 2-1(a). Generally, signals after one-sided translations become complex.

Eq. (2.6) is a *two-sided frequency translation* since the signal spectrum is moved in both positive and negative frequency directions. The signal shown in Figure 2-2(a) is the result of a two-sided translation of the signal in Fig. 2.1(a). The two-sided frequency shift of Eq. (2.6) belongs to the class of double-sideband suppressed carrier modulation (DSB/SC).

Quadrature modulation as described by Eq. (2.7) can be considered as another kind of two-sided frequency translation. Unlike the double-sideband suppressed carrier modulation where the spectrum at the negative frequency side is exactly the same as that at the positive frequency side except the difference in the center frequencies, the negative sideband of a quadrature modulated signal is the complex conjugate of the positive sideband due to the real nature of quadrature modulation in time domain. The two two-sided frequency shifts become equivalent when dealing with real signals.

A bandpass signal can be regarded as one that is frequency translated with carrier frequency $\omega_c = 2\pi f_c$ from its lowpass equivalent, as shown in Figure 2-2 for $f_c > B$ where B is the one-sided signal bandwidth. The bandpass signal is said narrow band if $f_c \gg B$ (as often encountered in RF/IF systems).

Another kind of real bandpass signals is single-sideband (SSB) modulated signals defined by

$$x_{SSB}(t) = x(t)\cos(\omega_c t) \mp \hat{x}(t)\sin(\omega_c t) \quad (2.8)$$

where $x(t)$ is the baseband real (lowpass) signal (Figure 2-1(a)) and $\hat{x}(t)$ is the Hilbert transform of $x(t)$. When the sign in Eq. (2.8) is '-', the transmitted SSB signal is the upper sideband (Figure 2-2(b)); otherwise, it is the lower sideband (Figure 2-2(e)).

2.2 Multirate systems and signal processing

Multirate digital signal processing technique provides a very efficient means for bandpass signal processing and multirate system optimizations. It minimizes computations in a multirate system by using the most appropriate sampling rate that is commensurate with the signal bandwidth [3]. This technique is particularly relevant to narrow band signal processing, e.g., in transmultiplexing/demultiplexing problems.

2.2.1 Discrete signal representations

2.2.1.1 Polyphase representation

Given an integer M , exactly M discrete signals can be obtained from a signal $x(n)$, each of which is down sampled from $x(n)$ with a different phase offset and is given as

$$x_\lambda^{(p)}(n) = x(nM + \lambda), \lambda = 0, 1, \dots, M-1, \quad (2.9)$$

where $x_\lambda^{(p)}(n)$ s are known as polyphase components of $x(n)$. Consequently, the signal can be represented by the polyphase components in the following form:

$$\begin{aligned} x(n) &= \sum_{\lambda=0}^{M-1} x_\lambda^{(p)}\left(\frac{n-\lambda}{M}\right) w_M(n-\lambda) \\ &= \sum_{\lambda=0}^{M-1} x(n) w_M(n-\lambda) \end{aligned} \quad (2.10)$$

where $w_M(n)$ is the discrete sampling function, which is defined as

$$w_M(n) = \frac{1}{M} \sum_{k=0}^{M-1} W_M^{kn} = \begin{cases} 1 & \text{for } n = mM, m \text{ integer} \\ 0 & \text{otherwise} \end{cases} \quad (2.11)$$

It is easy to show that the polyphase representation of discrete signal in the z-domain is:

$$X(z) = \sum_{\lambda=0}^{M-1} z^{-\lambda} X_{\lambda}^{(p)}(z^M) \quad (2.12)$$

where the z -transform of polyphase components is given as

$$X_{\lambda}^{(p)}(z) = \sum_{n=-\infty}^{\infty} x_{\lambda}^{(p)}(n) z^{-n} \quad (2.13)$$

2.2.1.2 Modulation representation

A set of complex exponential modulated signals $x_{k/M}^{(m)}(n)$, $k = 0, 1, \dots, M-1$ is defined as

$$x_{k/M}^{(m)}(n) = x(n) e^{j \frac{2\pi}{M} kn} = x(n) W_M^{-kn} \quad (2.14)$$

Their z -transforms are

$$X_{k/M}^{(m)}(z) = X(z W_M^k) \quad (2.15)$$

It is also simple to show, by considering Equations (2.10) and (2.11), that the z -transform of $x(n)$ can be represented by its modulated signals, i.e.,

$$X(z) = \sum_{\lambda=0}^{M-1} \left(\frac{1}{M} \sum_{k=0}^{M-1} X_{k/M}^{(m)}(z) W_M^{\lambda k} \right) \quad (2.16)$$

Comparing with Eq. (2.12), we have the following relationship between the polyphase components and the modulation components:

$$z^{-\lambda} X_{\lambda}^{(p)}(z^M) = \frac{1}{M} \sum_{k=0}^{M-1} X_{k/M}^{(m)}(z) W_M^{\lambda k} \quad (2.17)$$

That is, the polyphase components can be obtained by performing an IDFT (Inverse DFT) to the modulation components.

2.2.2 Sampling rate alteration

2.2.2.1 Downsampling and decimation

Wherever sampling rates are considerably higher than the signal's Nyquist rate, there might be possibilities to reduce the sampling rate hence reducing the computation load. The process of downsampling is also called *sampling rate decimation*. In most cases, it consists of two stages: anti-aliasing filter followed by a downsampler, as shown in Figure 2-3(a). The purpose of the anti-aliasing filter ($h(n)$) is to remove spectral fold-over

(aliasing) into the band of interest after downsampling. For decimation by M , the downsampler simply takes every M -th samples and discards the rest from the input sequence. Hence, for the decimated signal depicted in Figure 2-3 (a),

$$y(m) = \sum_{k=-\infty}^{\infty} h(k) x(mM - k) \quad (2.18)$$

A prominent property of downsampling is that the spectrum of the downsampled signal is the sum of equally spaced shifted copy of the original signal spectrum. This property can be seen in the z-transform of the downsampled signal,

$$Y(z^M) = \frac{1}{M} \sum_{k=0}^{M-1} V(z W_M^k) \quad (2.19)$$

With the substitutions of $z = e^{j\omega}$ and $V(z) = H(z)X(z)$, the Fourier relation between the decimated signal and the original signal is ,

$$Y(M\omega) = \frac{1}{M} \sum_{k=0}^{M-1} X\left(\omega - \frac{2\pi k}{M}\right) \cdot H\left(\omega - \frac{2\pi k}{M}\right) \quad (2.20a)$$

or, in terms of new frequency $\omega' = M\omega$

$$Y(\omega') = \frac{1}{M} \sum_{k=0}^{M-1} X\left(\frac{\omega' - 2\pi k}{M}\right) \cdot H\left(\frac{\omega' - 2\pi k}{M}\right) \quad (2.20b)$$

The spectral relation (ignoring scaling factors for the amplitudes) is shown in Figure 2-3(b) to Figure 2-3(e)

2.2.2.2 Upsampling and interpolation

If several narrow-band signals are to be combined to form a wide-band signal (as in FDM signal generation), their sampling rates must be increased before combining them together. The process of sampling rate increase is called *interpolation*. The sampling rate interpolation performs the upsampling followed by an anti-imaging filtering as shown in Figure 2-3(f). The latter is necessary because the up sampled sequence $v(m)$ contains $L-1$ images of the baseband spectrum at harmonics of the original sampling frequency $2\pi k/L$, $k=1, 2, \dots, L-1$.

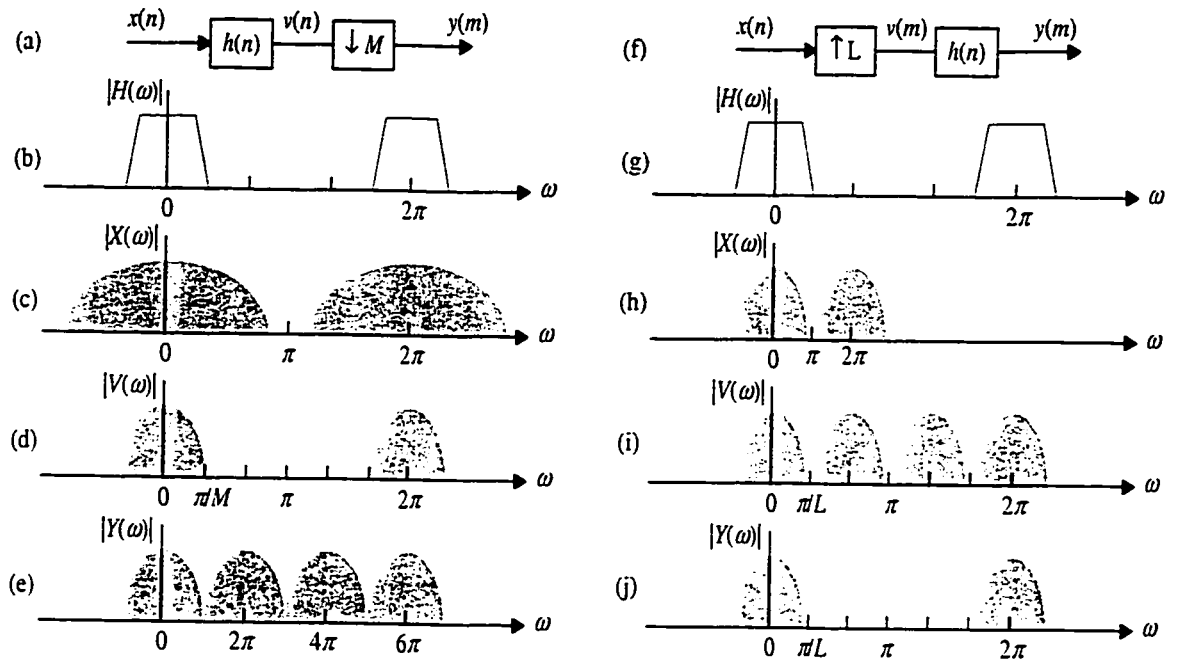


Figure 2-3 Spectra relation for integer rate alteration: (a) to (e) for integer rate decimation ($M=3$); (f) to (j) for integer rate interpolation ($L=3$)

The interpolated signal of Figure 2-3(f) can be represented as

$$y(m) = \sum_{k=-\infty}^{\infty} h(k) v(m-k) \quad (2.21)$$

where

$$v(m) = \begin{cases} x(m/L) & \text{for } m = nL, n \text{ integer,} \\ 0 & \text{otherwise.} \end{cases} \quad (2.22)$$

The z-transform of the up sampled signal is

$$V(z) = X(z^L) \quad (2.23)$$

Therefore, its Fourier transform is

$$V(\omega) = X(L\omega) = X(\omega') \quad (2.24)$$

These relations are illustrated in Figure 2-3(g) through Figure 2-3(j).

2.2.3 Rational sampling rate conversion

Sampling rate decimation or interpolation is the increase or decrease of sampling rate by an integer factor. In many cases, rational sampling rate conversions are required as in

frequency demultiplexing filter banks. In this section, it will be shown that rational sampling conversion can be realized by a linear periodically time-varying (LPTV) filter. Alternatively, the LPTV filter can be replaced by a polyphase network in which all the subfilters are linear time invariant (LTI) and the LPTV operations are decoupled from the filtering by the use of SPC and PSC commutators.

2.2.3.1 Rational sampling rate conversion

It has been shown that rational sampling rate conversion of a LTI discrete system leads to a LPTV system with input-to-output relation given as [19],

$$y(m) = \sum_{n=-\infty}^{\infty} g_m(n) x\left(\left\lfloor \frac{mM}{L} \right\rfloor - n\right) \quad (2.25)$$

where the time-varying impulse response $g_m(n)$ is defined as

$$g_m(n) = h\left(\left(nL + \left\langle \frac{mM}{L} \right\rangle\right)T^*\right) \quad (2.26)$$

where $h(t)$ is the impulse response of the corresponding continuous LTI system and $1/T^*$ is the sampling frequency for $h(t)$.

The time-varying nature of Eq. (2.25) is apparent. Since $g_m(n)$ is periodic in L , the system is linear periodically time-varying. Therefore, sampling rate alteration of a LTI system leads to a LPTV system if the rate change is rational. As the multirate filter bank theory is under the assumption of rational rate conversion, multirate filter banks thus belong to the class of LPTV systems.

For non-rational sampling rate alteration, the rate ratio could be expressed by a ratio of two very large integers which approach infinity, then the periodic nature of $g_m(n)$ no longer exists due to infinite L . Therefore, for non-rational sampling alteration, the LTI system becomes in general linear time-varying. Figure 2-4 shows the rational sampling rate conversion using LPTV filter.

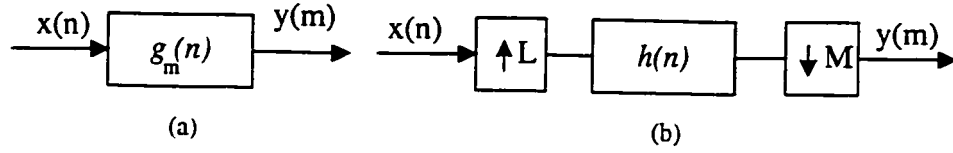


Figure 2-4 Rational sampling rate conversion:
(a) using a LPTV filter; (b) using a LTI filter

An alternative structure to Figure 2-4(a) is to use interpolation and decimation filters in tandem. As a result, the impulse responses of the anti-aliasing filter in the decimator and the anti-imaging filter in the interpolator can be combined into a single LTI filter $h(n)$. The resultant rational sampling converter is shown in Figure 2-4(b). Ideally, for lowpass signals, the combined filter should have the frequency response as

$$H(\omega) = \begin{cases} L & \text{for } |\omega| \leq \min\left\{\frac{\pi}{L}, \frac{\pi}{M}\right\} \\ 0 & \text{otherwise} \end{cases} \quad (2.27)$$

Compared to the LPTV filter approach, the symmetry property of the LTI filter can be exploited in this method, saving complexity by approximately a factor of two. Despite the advantages of avoiding time-varying filtering and being able to exploit symmetry property, the structure suffers poor computational efficiency since the filtering is performed at the highest sampling rate of the system. As will be shown later, a polyphase decimator and interpolator can greatly improve the computational efficiency by using a set of LTI sub-filters operating at a low sampling rate. They, however, lose symmetry due to the decomposition of the filter.

2.2.3.2 Polyphase analysis and polyphase synthesis of signals

Before introducing the concept of polyphase decomposition of decimation/interpolation filters, let us first look at two basic multirate operations, the polyphase analysis and polyphase synthesis of signals.

Polyphase components of signal:

For a discrete signal $x(n)$ (real or complex), its N polyphase components are defined as

$$x_i(m) = x(mN + i), \quad i = 0, 1, \dots, N-1 \text{ and } m \in \mathbb{Z} \quad (2.28)$$

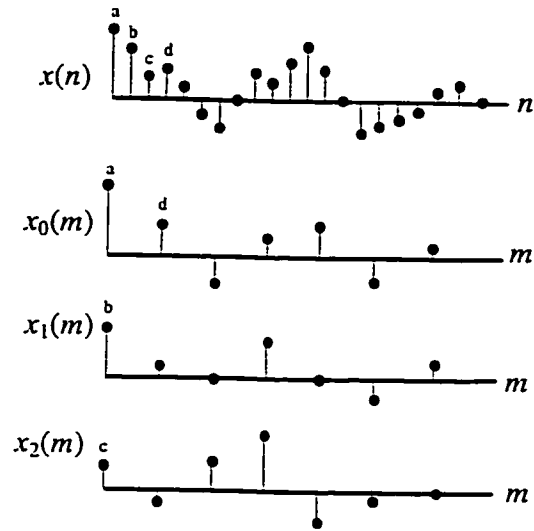
where N is an arbitrary integer. Clearly, a polyphase signal is a decimated signal with a time shift as illustrated in Figure 2-5(a). Though polyphase signals can also be defined as $x_i(m) = x(mN - i)$, as appeared in some literature, Eq. (2.28) will be solely used as the definition for polyphase signals hereafter to avoid confusion.

Polyphase analysis:

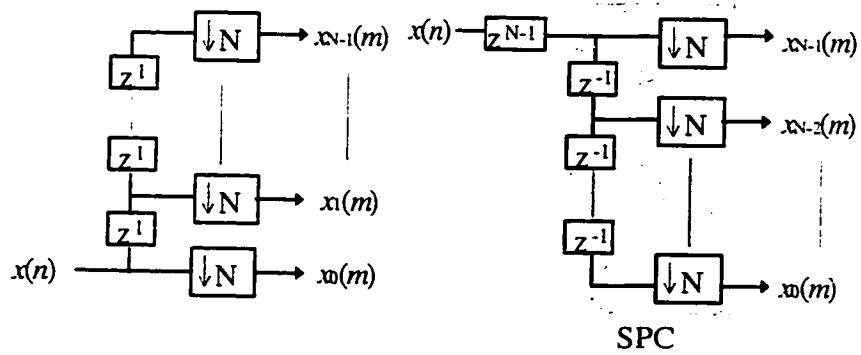
From the polyphase definition of Eq. (2.28), the polyphase components can be obtained through the structure shown in Figure 2-5(b). The commutation process in the structure is known as *polyphase analysis* and the structure is called *serial-to-parallel commutator* (converter) (SPC). Because time advances are used in the structure, it is non-causal and not realizable in practice. The time advances can be abstracted from the structure transforming the non-causal SPC structure into an equivalent structure consisting of a time advance and a causal SPC (shaded area) as shown in Figure 2-5(c).

Polyphase synthesis:

The dual process of polyphase analysis is called *polyphase synthesis* of a signal, which interleaves (combines) the polyphase components back into the original signal. The process is also governed by the relations of Eq. (2.28). The structure that synthesizes the original signal from the polyphase components should be the dual of Figure 2-5(b), which can be obtained by taking the generalized transposition to the figure-giving rise to Figure 2-6. The structure is called *parallel-to-serial commutator* (converter) (PSC).



(a) down-commutation ($N=3$)



(b) non-causal SPC

(c) causal SPC

Figure 2-5 Polyphase analysis of signal

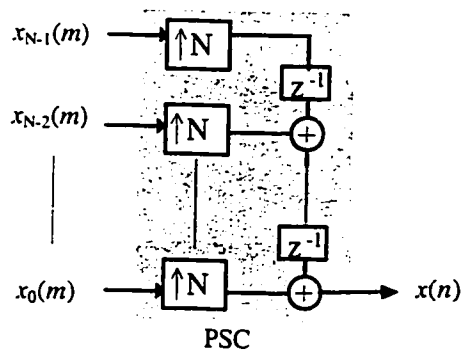


Figure 2-6 Polyphase synthesis of signal

2.2.3.3 Polyphase decomposition of decimation and interpolation filters

Polyphase structures are extremely important in multirate systems and filter banks. They provide opportunities for moving filtering (computations) from high sampling rates to lower ones and enables hardware sharing amongst different channels in filter banks, leading to both computationally efficient and low-complexity structures.

Consider the direct structure of the sampling rate decimator in Figure 2-3(a). Since the impulse response of the LTI anti-aliasing filter can be regarded as a signal (it is indeed from system point of view), it can be polyphase decomposed according to Eq. (2.12). Hence, we have in z -domain,

$$H(z) = \sum_{\lambda=0}^{M-1} z^{-\lambda} H_{\lambda}^{(p)}(z^M) \quad (2.29)$$

which represents a LTI polyphase network as shown in Figure 2-7(a). Replacing the anti-aliasing filter $h(n)$ in Figure 2-3(a) with the polyphase network results in an equivalent structure shown in Figure 2-8(a). It follows that the downsampler can be moved across the polyphase network by applying noble identities (see section 3.5), giving rise to the computational efficient structure of Figure 2-8(b) in which LTI sub-filters are at the lower sampling rate.

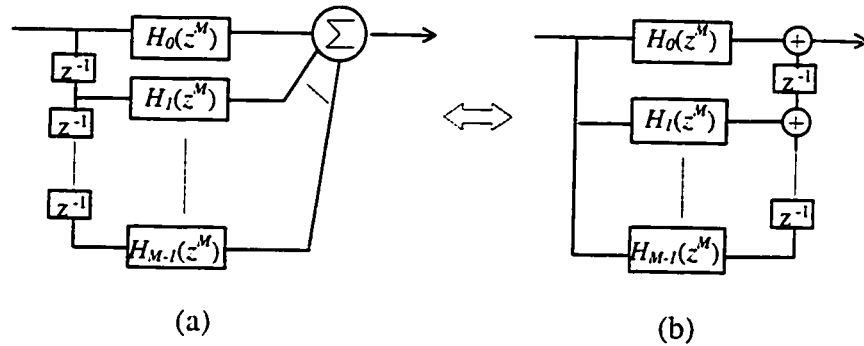


Figure 2-7 Transposition of LTI polyphase network

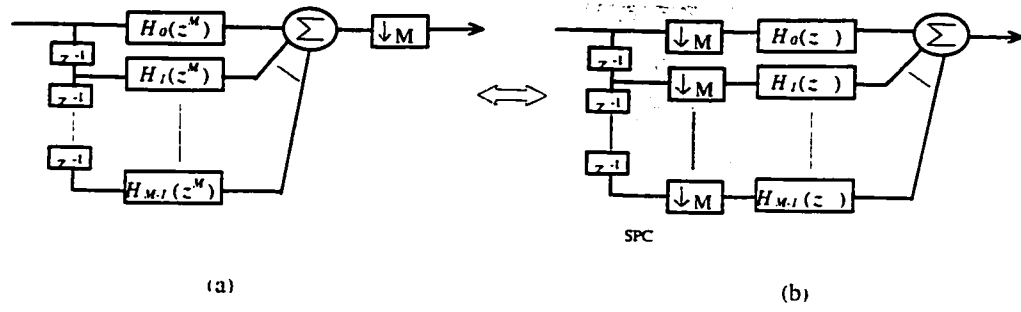


Figure 2-8 Polyphase decimation filter

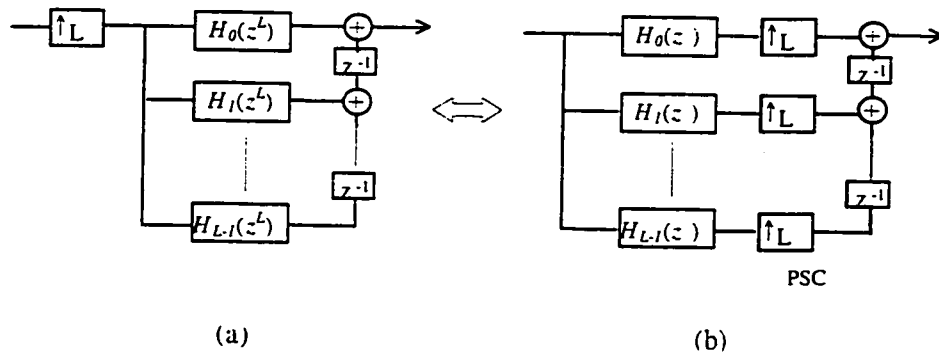


Figure 2-9 Polyphase interpolation filter

For the polyphase interpolator structure, the concept of transposition of LTI network can be used transforming the LTI polyphase network of Figure 2-7(a) into an equivalent transposed polyphase structure of Figure 2-7(b) (for $M=L$). Again, replacing the anti-imaging filter in Figure 2-3(e) with the transposed polyphase filter of Figure 2-7(b), results in Figure 2-9(a). Applying noble identities to the structure, the upsamplers are moved across the polyphase network leaving the polyphase filters at the lower sampling rate. The resulting computationally efficient polyphase interpolator structure is shown in Figure 2-9(b).

Theory of Multirate Signal Flow Graph Transformation

In this chapter we will briefly introduce the foundation of Multirate SFG (MSFG) and MSFG transformation. In addition, 18 MSFG operators as well as a number of MSFG transformations and identities in four categories will also be introduced.

3.1 MSFG

The signal graph representation renders explicit structural information making it suitable for hardware structure mapping. Nevertheless, the conventional signal flow graph (SFG) representation, which suits linear time invariant systems only [14,15], proves inadequate in multirate environment due to the linear periodically time varying nature of linear multirate systems [3].

For multirate systems, we introduce the multirate SFG representation as a complement to conventional design approaches and an extension to the SFG. A MSFG provides more direct and clearer link to the hardware structure and the parallelism of filter banks than does the pure mathematical representation. Being extended from the conventional SFG, the MSFG preserves most of the properties of the former. In multirate environment, identities and MSFG transformations are defined. With the introduction of a set of shorthand notation for MSFG, the flow graph manipulation and transformation problems can be considerably simplified.

Signal Flow Graph is traditionally defined by a set of branches and nodes in which the former define the signal operations and the latter define the connection points of these

branches in the structure [3,14]. Similarly, a MSFG also consists of branches and nodes. However, branches in MSFG are restricted to be LTI (Linear Time Invariant), just as those in the conventional SFG, and all the non-linear and time-varying (modulation, down-sampling, up-sampling, for examples) operations are defined by node functions. The reason that we attribute the non-linear and multirate operations to node functions is because all the “irregularities” of the MSFG, with respect to the conventional SFG, can be approached by identities and transforms associated with nonlinear/time-varying nodes. Thus, special care is only needed when those nonlinear/time-varying nodes are involved.

From the point of view of signal processing, branches or node functions perform specified processing for input signal, so these branches and node functions can be represented as MSFG operators in my study.

3.2 MSFG Operators

3.2.1 Elementary MSFG operators

In multirate signal flow graph approach, five elementary operators, namely, adder (ADD), multiplier (MUL), delay element (DELAY), downsampler (DS) and upsampler (US) are defined.

Conceptually, the fundamental difference between classical DSP and multirate DSP lies in the sampling rate alteration, which is not allowed in the former.

The most fundamental single-rate operators are *adder (ADD)*, which adds a group of input signals, *multiplier (MUL)*, which multiplies the input signal with a real or complex constant, and *delay element (DELAY)*, which delays the input signal by an integer number k of samples.

The most fundamental multirate operators are *downsampler (DS)* and *upsampler (US)*, which respectively decrease or increase the sampling rate in multirate DSP.

These elementary MSFG operators are described detailedly in Table 3-1.

Table 3-1 Elementary MSFG operators

No	Operator Name	Block diagram Notation	Short-hand notation	Output waveform
1	ADD (adder)			
2	MUL (multiplier)			
3	DELAY (delay element)			
4	DS (down-sampler)			
5	US (upsampler)			

3.2.2 Basic MSFG operators

From these fundamental single-rate and multirate operators, we can define some composite MSFG operators, they are Modulator (MOD), Filter (FIL), S/P commutator (SPC), Overlapped S/P commutator (OSPC), P/S commutator (PSC), Null Operator (NULL), Sampling & Hold (SH), Upsampling & Hold (USH), Integral & Dump (IDU), and Sampler (SA).

From DS and US, three important rate-changing operators are defined. They are *serial-to-parallel commutator (SPC)*, which decreases the sampling rate by decomposing the incoming signal into a group of sub-signals, *parallel-to-serial commutator (PSC)*, which increases the sampling rate by combining a group of signals and *overlapped serial-to-parallel commutator (OSPC)*, which is similar to the SPC except that the number of parallel branches N is larger than the decimation factor M .

There are some other multirate operators frequently encountered in digital networks. A **sampling & hold (SH)** does not actually change the sampling rate of a signal. It retains (holds) the signal value at the sampling point N (SH factor) times. An **upsampling & hold (USH)**, on the other hand, does increase the sampling rate of a signal by repeating (holding) every signal element of the input signal L (USH factor) times. Similar to the upsampler that has the dual operation of downsampler, the USH also has a dual operation, which is the **integral & dump (IDU)**. The IDU operator is defined by the transposition of USH. An IDU with decimation factor M integrates (accumulates) every M signal elements and dumps the integral to output reducing the sampling rate by a factor of M . Unlike the SH operator, a **sampling (SA)** operator samples the incoming sequence without holding the sampled signal elements, where k_n denotes the sampling instance n that satisfied $\langle n - k \rangle_N = 0$, where $\langle \bullet \rangle_N$ performs modulo N operation.

Apart from the above eight multirate operators, i.e., US, DS, SH, USH, SPC, PSC, IDU, and SA, three single-rate operators are also defined in MSFG. The output of a **null operator (NULL)** is exactly the same as its input. A **modulation operator (MOD)** performs signal modulation (multiply the input signal with modulation signal $\phi(n)$ or W_N^n). A **filter (FIL)** performs linear filtering whose impulse response is defined by

$$H(z) = \frac{Y(z)}{X(z)} = \frac{b_0 + b_1 z^{-1} + \dots + b_{M-1} z^{-(M-1)}}{a_0 + a_1 z^{-1} + \dots + a_{N-1} z^{-(N-1)}} = \frac{\sum_{k=0}^{M-1} b_k z^{-k}}{\sum_{k=0}^{N-1} a_k z^{-k}}. \quad (3.1)$$

These basic MSFG operators are described detailedly in Table 3-2.

Table 3-2 Basic MSFG Operators

No	Operator Name	Block diagram Notation	Short-hand notation	Output waveform
6	MOD (modulator)			
7	FIL (filter)			
8	SPC (S/P commutator)			
9	OSPC (overlap S/P commutator)			
10	PSC (P/S commutator)			
11	NULL (null operator)			
12	SH (sampling &hold)			
13	USH (upsampling &hold)			
14	IDU (integral &dump)			
15	SA (sampler)			

3.2.3 MIMO MSFG operators

In addition, we define some useful MIMO (Multi-Input Multi-Output) MSFG operators, i.e., Discrete Fourier Transform (DFT), Interleaver (INTL), and Switch (SW).

A **Digital Fourier Transform (DFT)**, shown in Figure 3-1, performs

$$X[k] = \sum_{n=0}^{N-1} x[n] \cdot e^{-j\frac{2\pi}{N}kn} = \sum_{n=0}^{N-1} x[n] \cdot W_N^{kn}. \quad (3.2)$$

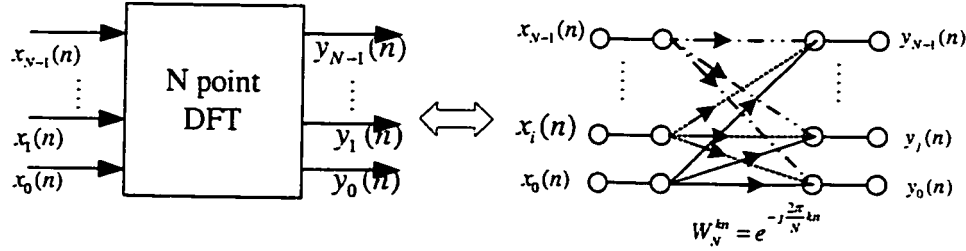


Figure 3-1 DFT

A **switch (SW)** interconnects an array of parallel input signals to an array of output ports. An **interleaver (INTL)**, as shown Figure 3-2, writes an array of parallel signals into a memory in an order and reads them out in another order.

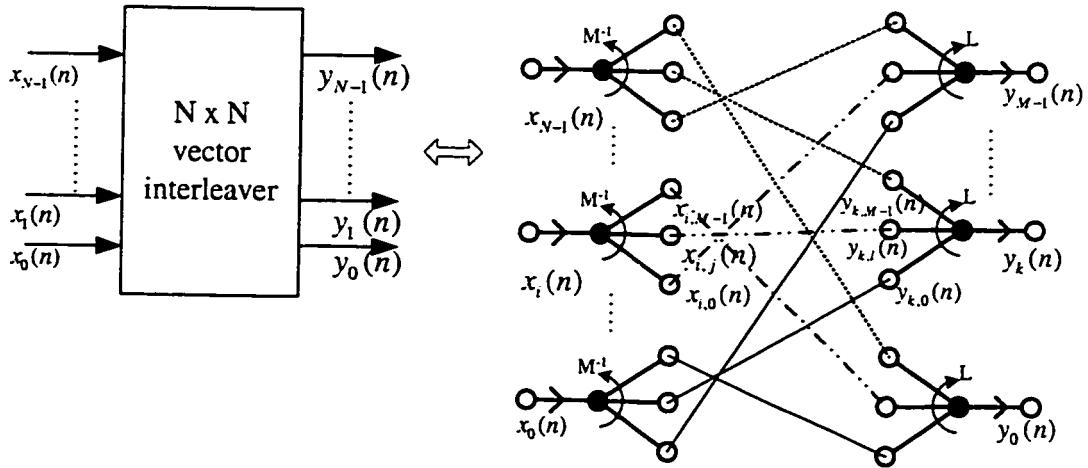
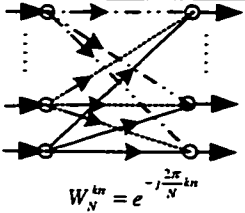
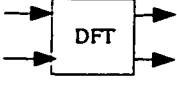
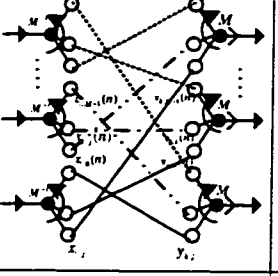

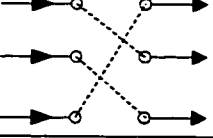
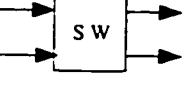


Figure 3-2 Interleaver

These MIMO MSFG operators are described in Table 3-3.

Table 3-3 MIMO MSFG Operators

No	Operator Name	Block diagram Notation	Short-hand notation	Output waveform
16	DFT (Digital Fourier Transform)	 $W_N^{kn} = e^{-j\frac{2\pi}{N}kn}$		
17	INTL (inter-leaver)			
18	SW (switch)			

3.3 MSFG transformation

A multirate DSP system can be transformed into its equivalencies with different structures via flow graph transformations to be given in the follow sections. The fundamental relationships between multirate operators have been given in the form of identity (noble identities shown in Figure 3-4).

The most fundamental and important result in multirate DSP theory is, perhaps, the concept of polyphase decomposition of signals and networks. Polyphase filter transform can lead to computationally efficient filter bank structures. The polyphase decomposition concept can be extended to modulation leading to the Modulation Polyphase Decomposition Transform (MPDT) and its variations.

Since the upsampling and downsampling processes are often realized via PSC and SPC (as in polyphase decomposition of filters and modulators) in multirate DSP networks, one will inevitably deal with various combinations of commutators and other signal processing components in multirate networks. A set of identities associated with

the cascades of commutators with filters, modulators, and single up/down-samplers have been identified. They are very useful in multirate DSP network transformations.

3.4 Superposition theorem

For a linear multirate system $H(z)$, if it has more than one independent signal sources S_i , $i=1, \dots, N$, and the transfer functions from $S_i(n)$ to the output are $H_i(z)$, $i=1, \dots, N$, then the response of the system is the sum of the partial responses $D_i(z)=S_i(z) \bullet H_i(z)$, i.e.,

$$D(z) = \sum_{i=1}^N S_i(z) \bullet H_i(z) \quad (3.3)$$

This theorem can be expressed graphically by Figure 3-3.

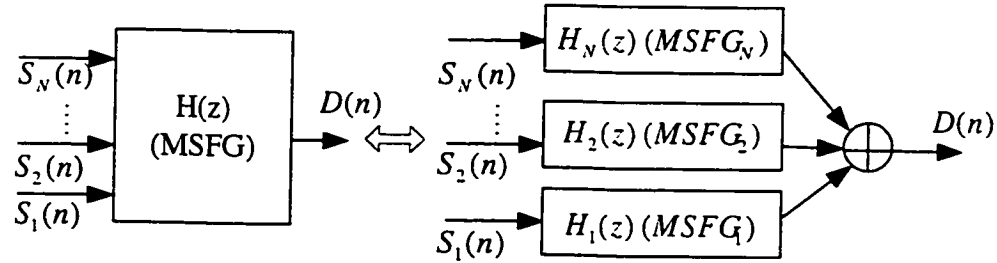


Figure 3-3 Superposition theorem

In the following sections, we will summarize various MSFG transformations in different categories.

3.5 The noble identities

The noble identities are considered the most fundamental characteristics of multirate systems, with which most MSFG identities and transforms can be derived. Some noble identities are shown in Figure 3-4.

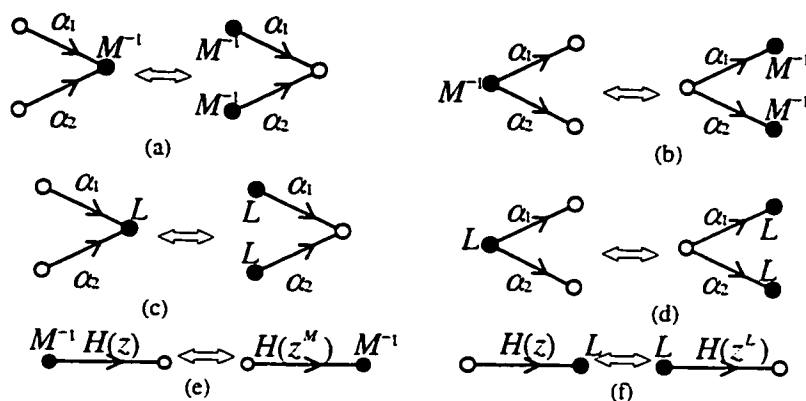


Figure 3-4 Noble identities

The identities (a) to (d) are obvious because of the linear assumption of MSFG. The noble identity (e) shows that the downsampler (with factor M) can be moved forward across any LTI system and it can be moved backward across a LTI system if, and only if, the system is divisible by z^{-M} to avoid non-realizable fractional delays. Similarly, the noble identity (f) suggests that an upsampler (with factor L) can move backward freely across any LTI system and move forward across a LTI system only when it is divisible by z^{-L} .

3.6 General identities

3.6.1 Complex filter identities

Figure 3-5 show that complex filter identities. Identity Figure 3-5(a) shows that a complex filter can be realized using the DRFU (Down-Real Filter-Up) structure. Identity Figure 3-5(b) shows that if a signal is firstly modulated by a complex exponential and then filtered by a real filter, the process is equivalent to firstly filtering with an inverse modulated filter (complex) and then modulating with the complex exponential.

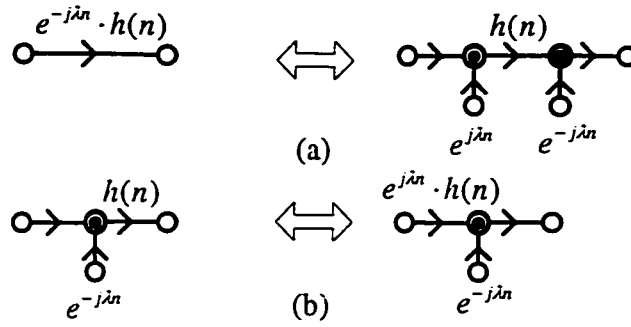


Figure 3-5 Complex filtering identities

3.6.2 Modulation identities

Some important equivalent structures associated with modulation operators, which are frequently used in multirate filter bank design, are summarized into identities shown in Figure 3-6. The identities shown in Figure 3-6(a) to Figure 3-6(d) holds for arbitrary modulating sequence $\phi(n)$, periodic or non-periodic. The modulating sequences $\phi(\lfloor n/M \rfloor)$ in Figure 3-6(b) are up-sampled (hold or without hold) versions for $\phi(m)$; and $\phi(\lfloor m/L \rfloor)$ in Figure 3-6(d) are for $\phi(n)$.

Since most often the modulating sequences are complex exponentials, the identities shown in Figure 3-6(a) to Figure 3-6(d) are equivalent to those of Figure 3-6 (e) to Figure 3-6 (h).

We often have to deal with cascades of modulator and delay/advance in MSFG networks. Assuming that the modulation sequence extends infinitives in both time directions, it is easy to show that the delay can be moved across the modulator if the origin of the modulating sequence changes accordingly as shown in Figure 3-6 (i) and Figure 3-6 (j). The identities shown in Figure 3-6 (k) and Figure 3-6 (l) are the equivalent to those of Figure 3-6(i) to Figure 3-6(j) when complex exponentials are used.

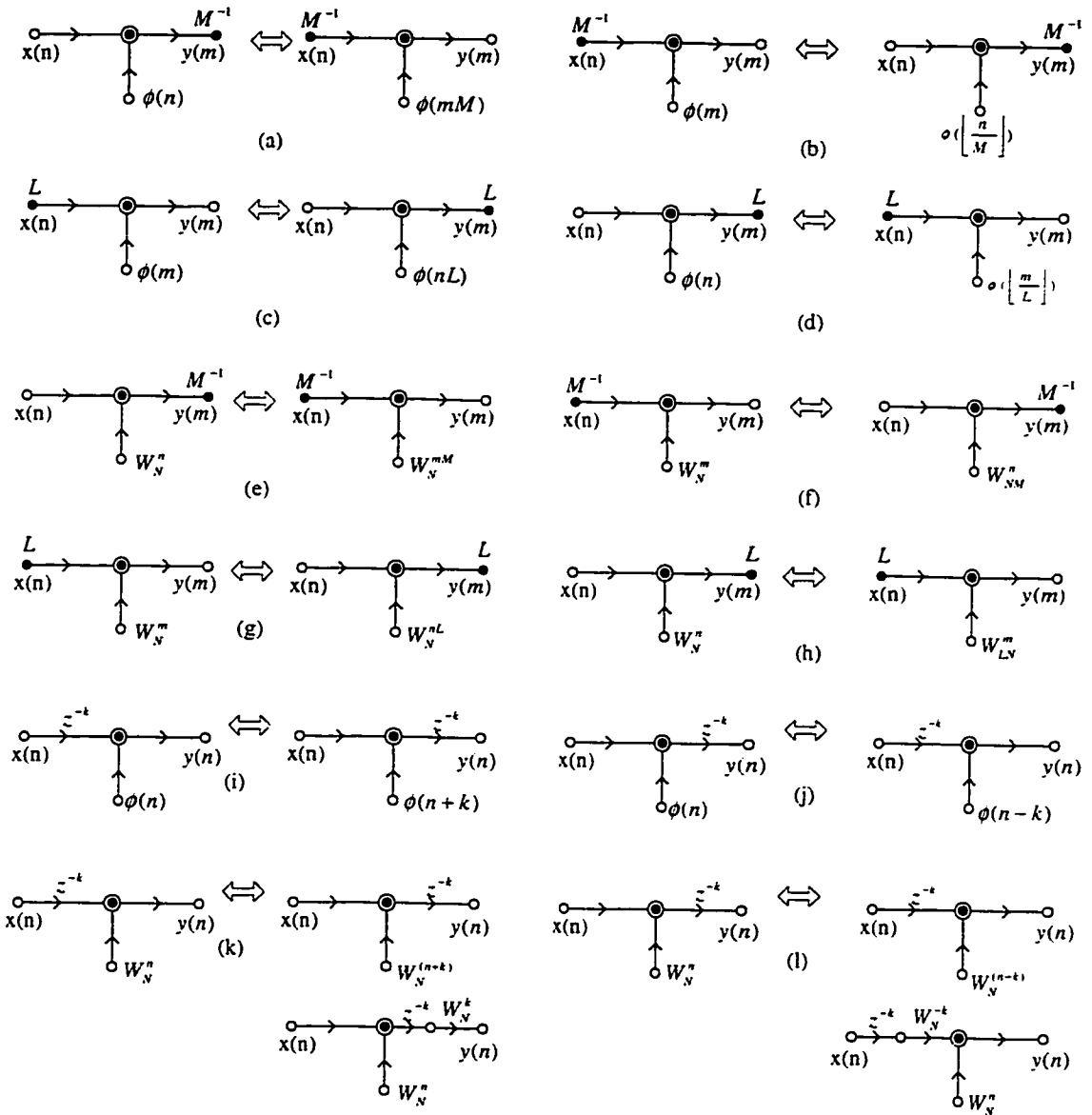


Figure 3-6 Modulation identities

3.6.3 Identities associated with composite rate-changing operators

According to the definition of USH operator, it is equivalent to a PSC that has all its parallel inputs connecting to an ordinary operator as shown in Figure 3-7(a). The IDU function, as the dual to USH, is equivalent to the transposition of USH and is shown in Figure 3-7 (b). Another way of realizing the IDU function is to perform an M tap comb filtering first and then down sample the sequence by M as shown in the Figure 3-7(b). Having defined the USH function, the SH function can be realized by the simple cascade

of a down sampler and an USH as shown in Figure 3-7(c). Figure 3-7(d) shows this relation, which includes the identity shown in Figure 3-12(g), as special case with $k=0$.

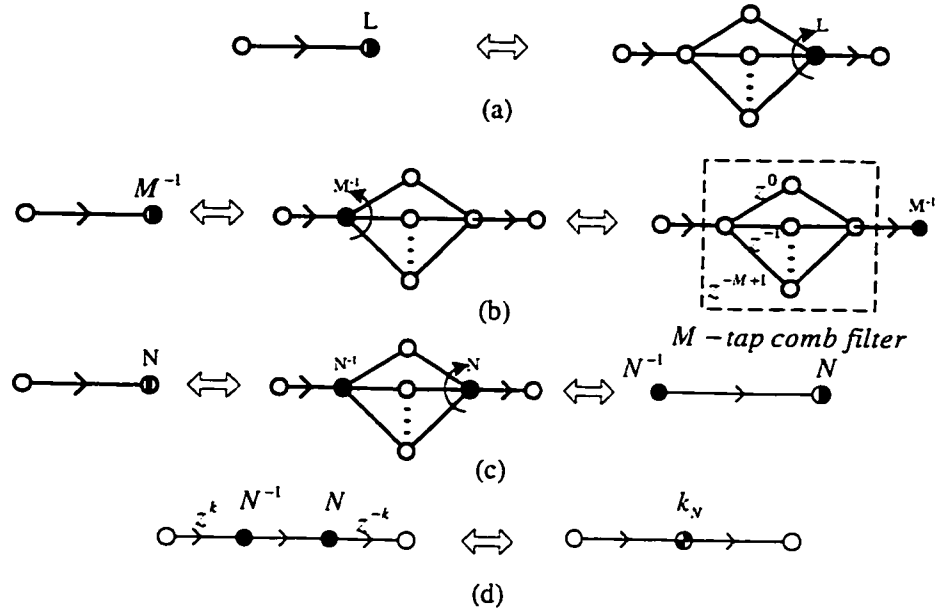


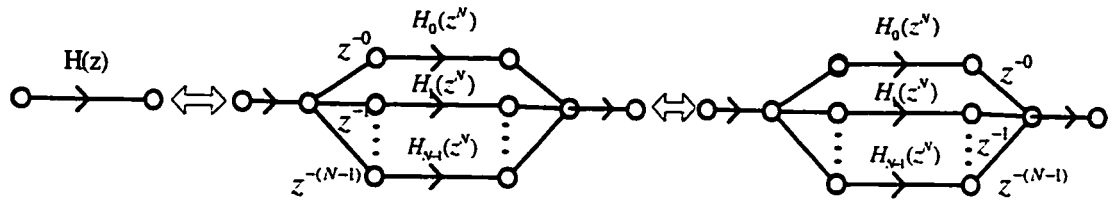
Figure 3-7 Identities associated with composite rate-changing operators

3.7 Polyphase decomposition

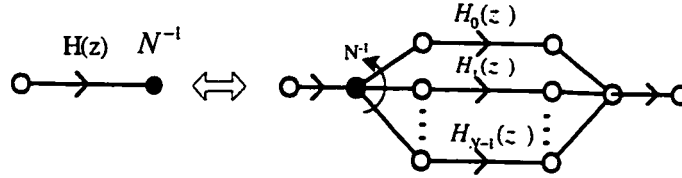
3.7.1 Filter Polyphase decomposition

Polyphase structures are extremely important in multirate systems and filter banks, which are shown as Figure 3-8. They provide opportunities for moving filtering from high sampling rates to lower sampling rates and enables hardware sharing amongst different channels in filter banks, leading to both computationally efficient and low-complexity structures. $H_i(z)$'s are the z -transforms of polyphase filters $h_i(m)$'s which are defined as

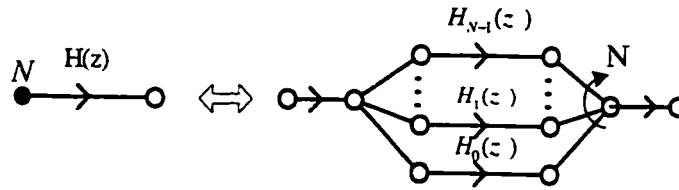
$$h_i(m) = h(mN + i), i = 0, 1, \dots, N - 1 \quad (3.4)$$



(a) polyphase decomposition for LTI filter



(b) polyphase decomposition for decimation filter



(c) polyphase decomposition for interpolation filter

Figure 3-8 Polyphase decomposition transforms (PDT)

3.7.2 Modulation polyphase decomposition

Consider the modulation of signal $x(n)$ with a sequence $\phi(n)$, the result signal $y(n)$ is: $y(n)=x(n)\phi(n)$. Similar to the FIR filter polyphase decomposition, the signal $y(n)$ can be represented by its N polyphase components:

$$y_i(m) = y(mN + i) = x_i(m)\phi_i(m) \quad (3.5)$$

where

$$\begin{aligned} x_i(m) &= x(mN + i) \\ \phi_i(m) &= \phi(mN + i) \end{aligned} \quad (3.6)$$

$i=0,1,\dots,N-1$, m is integer. $x_i(m)$ and $\phi_i(m)$ are polyphase components of $x(n)$ and $\phi(n)$ respectively. With the SPC and PSC structures of Figure 2-5(b) and Figure 2-6, the Eq. 3.7 can be described as a polyphase network shown in Figure 3-9(a). It is called **type 1 MPDT** (Modulation Polyphase Decomposition Transform). Alternatively, the modulation can be expressed with a different set of polyphase components by:

$$\begin{aligned}
y_i(m) &= y_{-i}(m) = y(mN - i) \\
&= x(mN - i)\phi(mN - i) \\
&= x_{-i}(m)\phi_{-i}(m)
\end{aligned} \tag{3.7}$$

In this case, the SPC will be causal but the PSC will not be causal. The embedded poly-phase network is shown in Figure 3-9(b), which is equivalent to Figure 3-9(a) and is called **type 2 MPDT**. In the above polyphase networks, the serial-to-parallel commutator (SPC) in Figure 3-9(a) and the parallel-to-serial commutator (PSC) in Figure 3-9(b) are non-causal hence unrealizable. In section 2.2.3.2, it has been shown that non-causal SPC can be replaced by a time advance of z^{N-1} followed by a causal SPC. Similarly, a non-causal PSC can be substituted by a causal PSC followed by a time advance z^{N-1} . Therefore, the type 1 and type 2 MPDT have the equivalent structures shown in Figure 3-9(c) and Figure 3-9(d) respectively. The corresponding MSFGs are shown in Figure 3-10(a) and Figure 3-10(b) respectively.

If the modulating sequence is a complex exponential with period N,

$$\phi(n) = e^{-j\frac{2\pi}{N}n} = W_N^n \tag{3.8}$$

then its components become complex constants $\phi_i(m) = \phi(mN + i) = W_N^i$, which leads to simple MPDT structure in which the modulation is replaced by simple scalar, as shown as Figure 3-11(a) and Figure 3-11 (b).

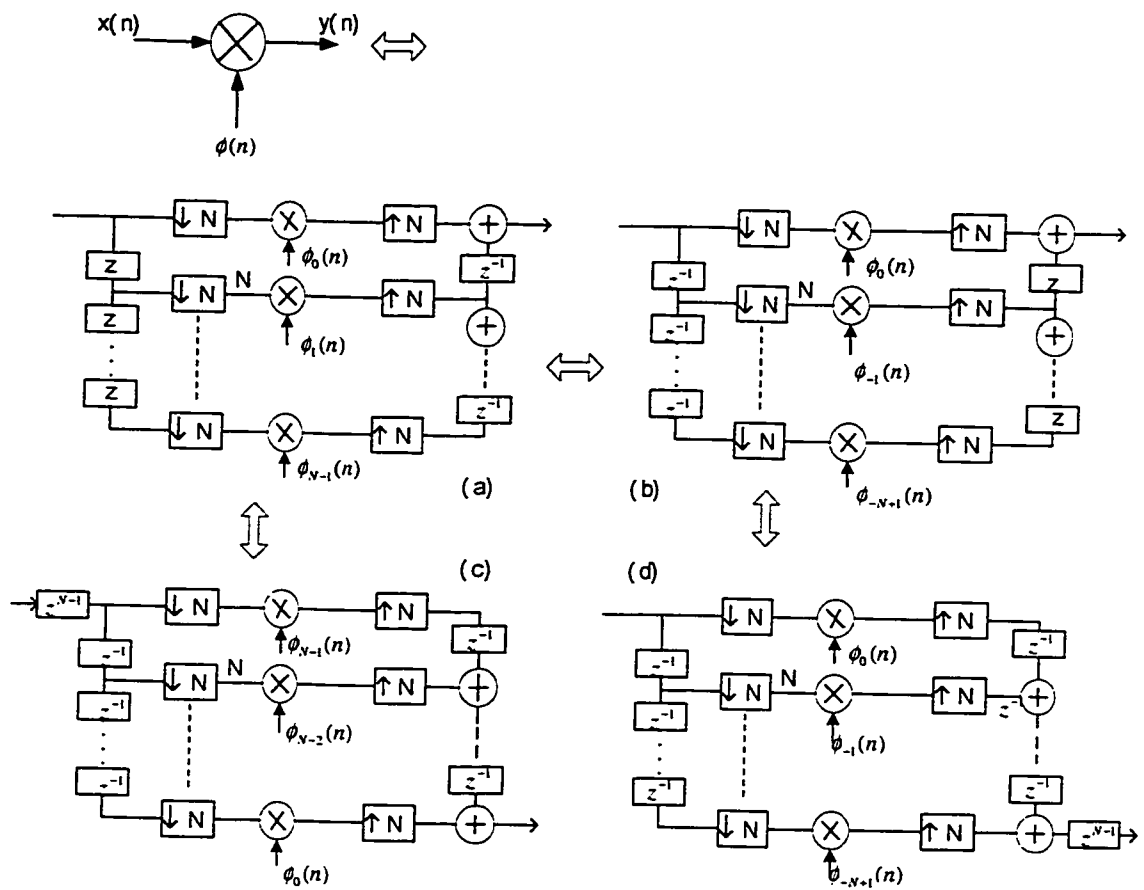


Figure 3-9 Two types of MPDT structures in conventional SFG

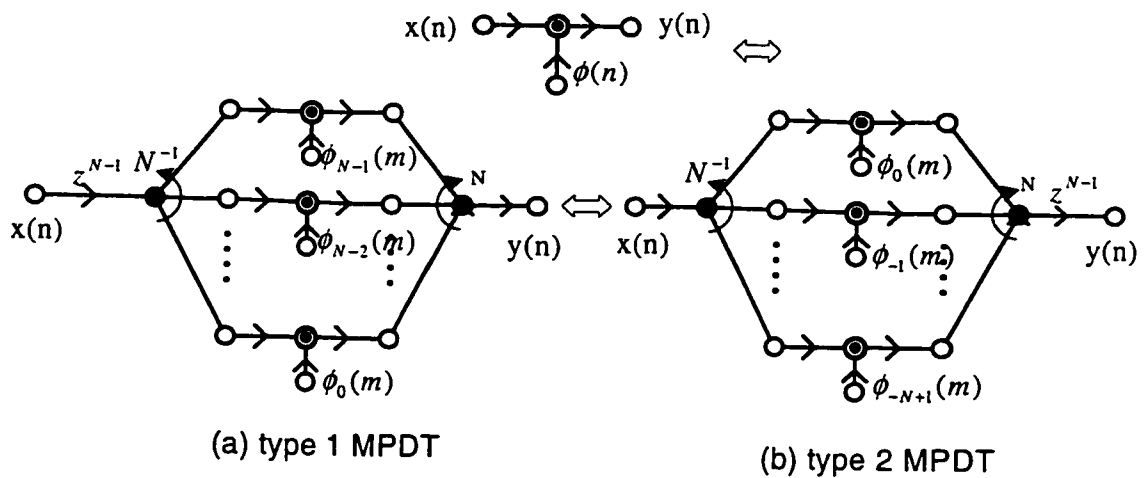


Figure 3-10 MSFGs of type 1 and type 2 MPDT

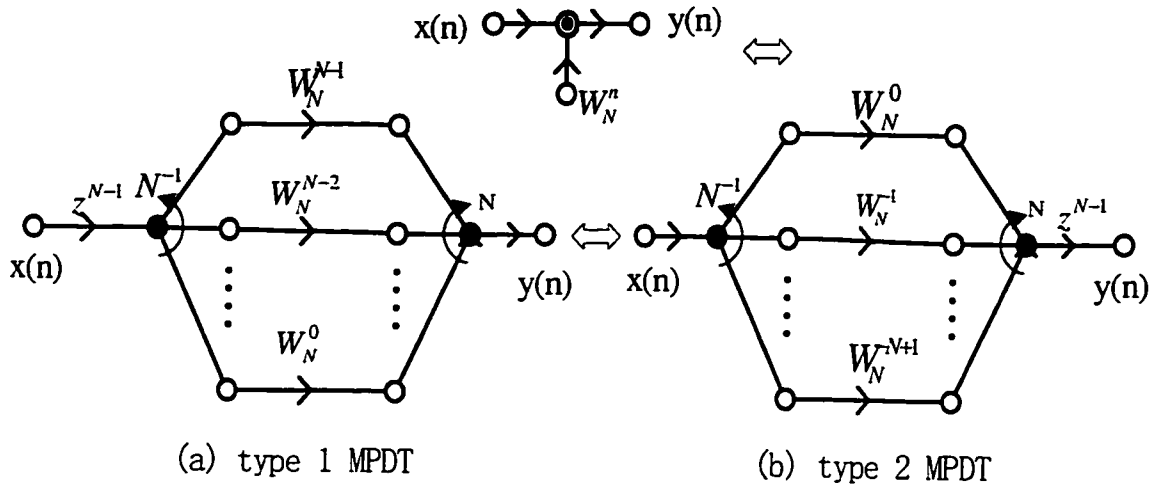


Figure 3-11 MPDT for complex modulation

3.8 Cascade of operators

3.8.1 Cascade of samplers

Cascade of samplers are shown in Figure 3-12. The identities of Figure 3-12(a) and (b) show that down-sampling and up-sampling can be performed in stages if the factors are composite. It's evident from the definitions of down-sampling and up-sampling. Identity Figure 3-12(c) shows that for a upsampler-downsampler, or downsampler-up-sampler, cascade to be commutable the necessary and sufficient condition is that the two rate converting factors L and M must be co-prime, i.e., $\text{GCD}(L, M) = 1$. When the upsampler and the downsampler have the same factor M , the order cannot be changed. Identity Figure 3-12(d) show that if up-sampler is preceding down-sampler and two rate factors are all M , the inserted $M-1$ zeros to each signal sample by up-sampler will be discarded by down-sampler, result the output are exactly the same as input. Figure 3-12(e) and (f) shown that if $k=qM$, noble identity (e) or (f) can be applied, the delay can be moved across the samplers and become to z^{-q} from z^{-qM} ; otherwise the output samples will be constant zero because at the sampling points of the downsampler the signal elements are all zeros. In contrast, if down-sampler is performed first shown in Figure 3-12(g), the following upsampler will insert $M-1$ zeros after each sample retained by the downsampler, hence the process is equivalent to the modulation of the signal with a sampling sequence as shown in Figure 3-12(g) and is also equivalent to the sampler (SA)

with offset equal to zero. In identity Figure 3-12(h), it is very clear that when this sampling process is cascaded with a delay which $k=qM$ on any side of the network, the delay can move across the downsampler-up sampler cascade freely. This property can be proved by simply applying the noble identity (e) or (f). Figure 3-12(i) shows that if M and L are co-prime, we can find i and j which satisfy $iL-jM = -k$, then z^{iL} and z^{-jM} can exchange with upsampler and downsampler, applying noble identities (e) and (f).

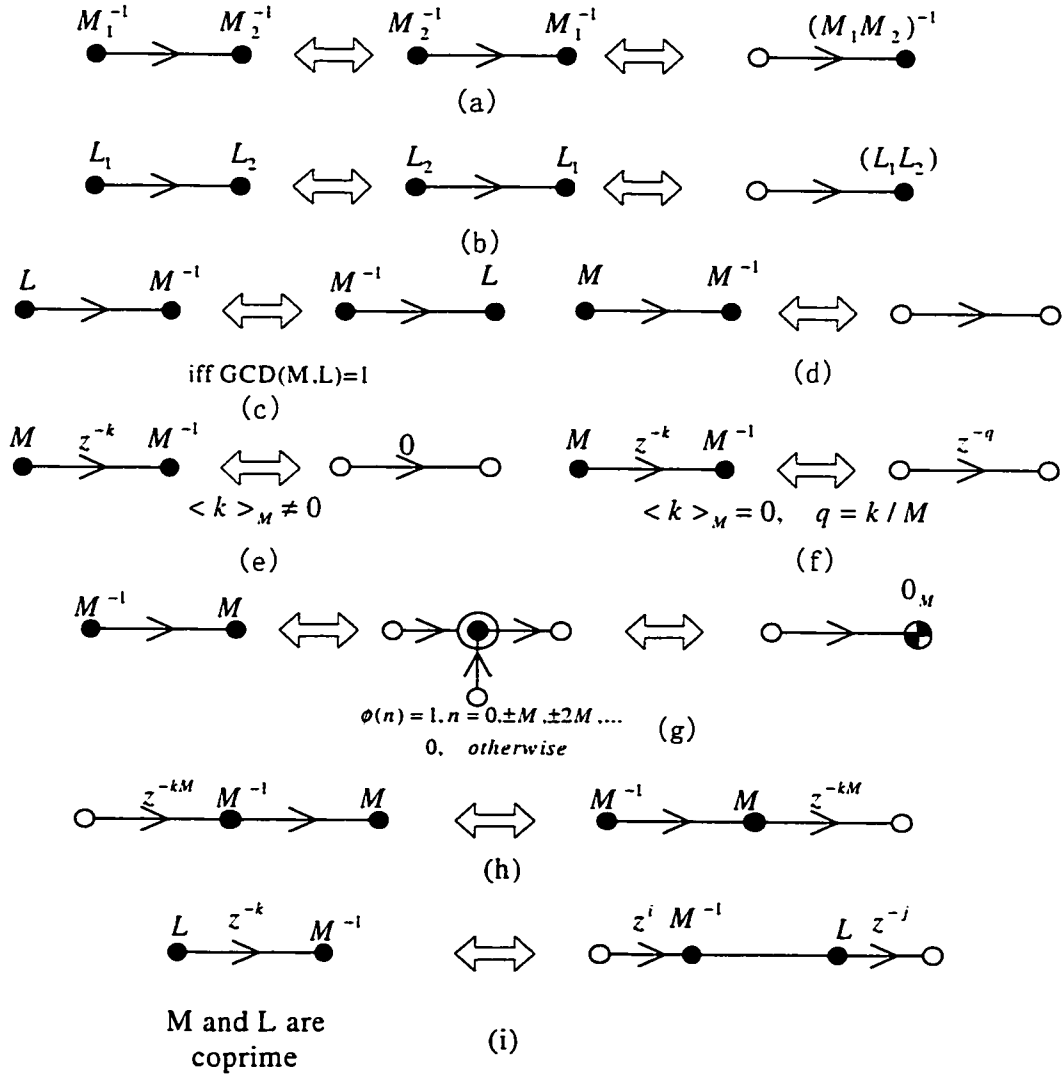


Figure 3-12 Cascade of samplers

3.8.2 Cascade of commutators

Serial-to-parallel (SPC) and parallel-to-serial (PSC) commutations can be performed in multi-stage if the commutation factors are composite. That is, for N-fold commutation, if $N = \prod_{i=0}^{m-1} N_i$, an M-stage commutation tree can be constructed, Figure 3-13 (a) and (b) demonstrates how a SPC and a PSC (N=6 for both cases) are decomposed into commutation trees. The numbers in the figure are the sample indices of signal elements. This property is frequently used in MSFG simplification.

From time to time, we have to deal with SPC-PSC and PSC-SPC cascades in MSFG manipulation. Since SPC and PSC are dual and perform operations complementary to each other, it is intuitive to foresee that cascading of the two would cancel the effects of the two types of commutations. Therefore, it can be expected that the cascades should be equivalent to simple networks connecting the input(s) and the output(s) with, perhaps, some pure delays as a consequence of the use of causal PSC. It is simple to verify (e.g., graphically) that an SPC-PSC cascade with commutation factor of N is equivalent to a pure delay of N-1 (shown in Figure 3-13(c)). Similarly, a PSC-SPC cascade is equivalent to a part delayed connection (except a_0 and b_0 no delay, others delayed with a unit delay) as shown in Figure 3-13 (d). In particular, when a unit delay is inserted between the PSC and SPC, the network can be transformed into a full delayed connection (with a unit delay) between corresponding ports of PSC and SPC (Figure 3-13 (e)). The simplest case is the direct connections between the corresponding PSC and SPC ports, which occurs when a time advance of z^{N-1} is in sandwich between the commutators, as shown in Figure 3-13(f).

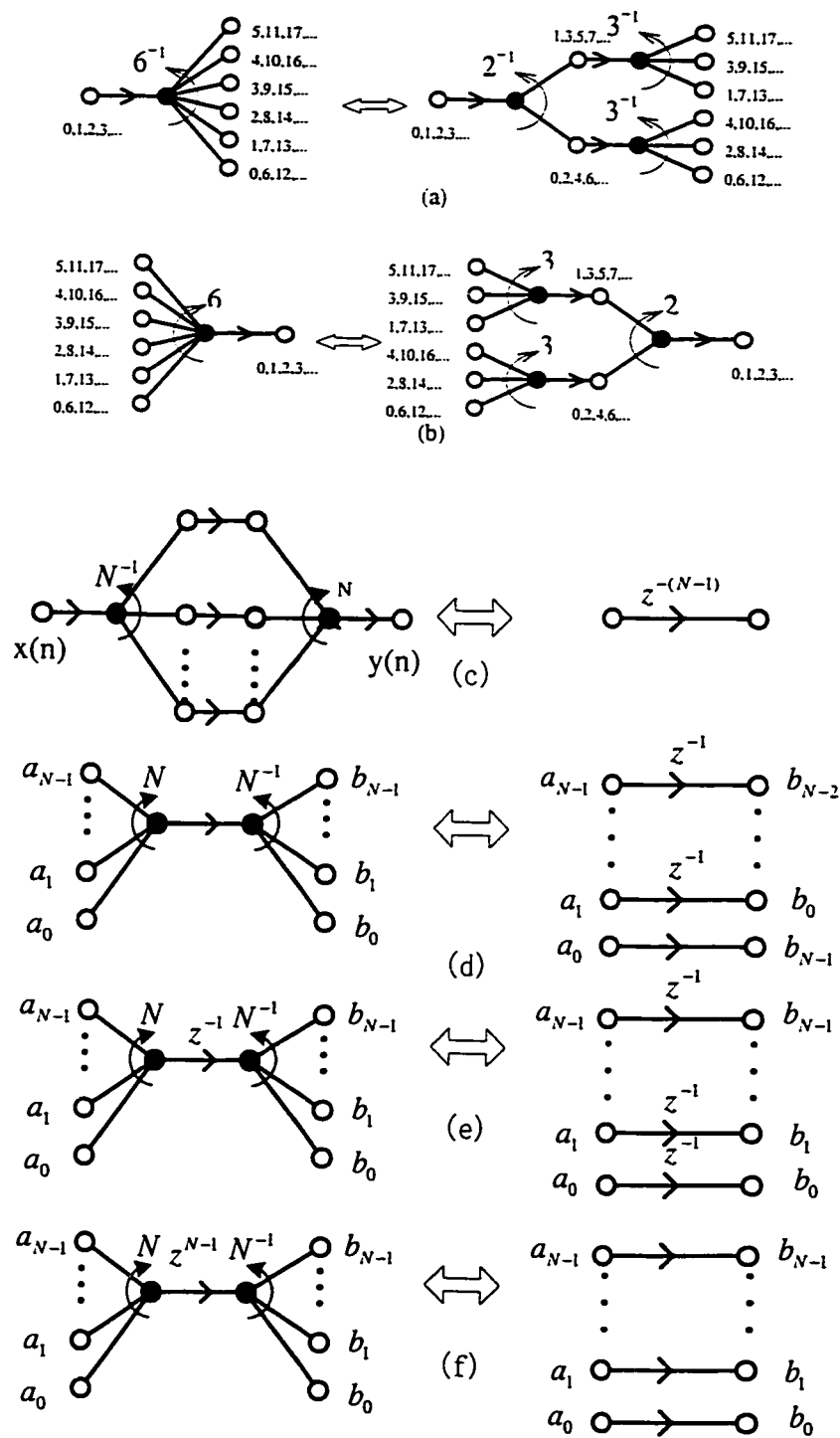


Figure 3-13 Cascades of Commutators

3.8.3 Commutator-modulator cascades

When modulators are cascaded with commutators, MPDTs introduced in section 3.7.2 can be used to move polyphase components of modulating sequences into polyphase branches of commutators. For a PSC-modulator cascade, by using the type 1 MPDT to the modulator and then applying the identity of Figure 3-13(f), the modulator is polyphase decomposed and moved into the PSC branches as shown in Figure 3-14(a). Similarly, for a modulator-SPC cascade, it can be transformed into an equivalent structure shown in Figure 3-14 (b) by applying the type 2 MPDT and the identity of Figure 3-13(f). If the modulation sequence in the above cascades is periodic with period N of $\phi(n) = W_N^n$ being equal to the commutation factor N , then the modulators in the commutator branches become scalar multipliers as shown in Figure 3-14(c) and 3.14(d).

If branches of a SPC are modulated with the same periodic sequence, say a complex exponential of $\phi(n) = W_N^n$, then a common modulator can be used for all the branches and each branch is phase shifted by a constant phase as shown in Figure 3-14(e). This identity can be proved by applying the identities of Figure 3-6(f) and Figure 3-6(i). Similarly, when branches of a PSC are modulated with the same complex exponential, a common modulator can be used and each branch has to be phase shifted accordingly as shown in Figure 3-14(f).

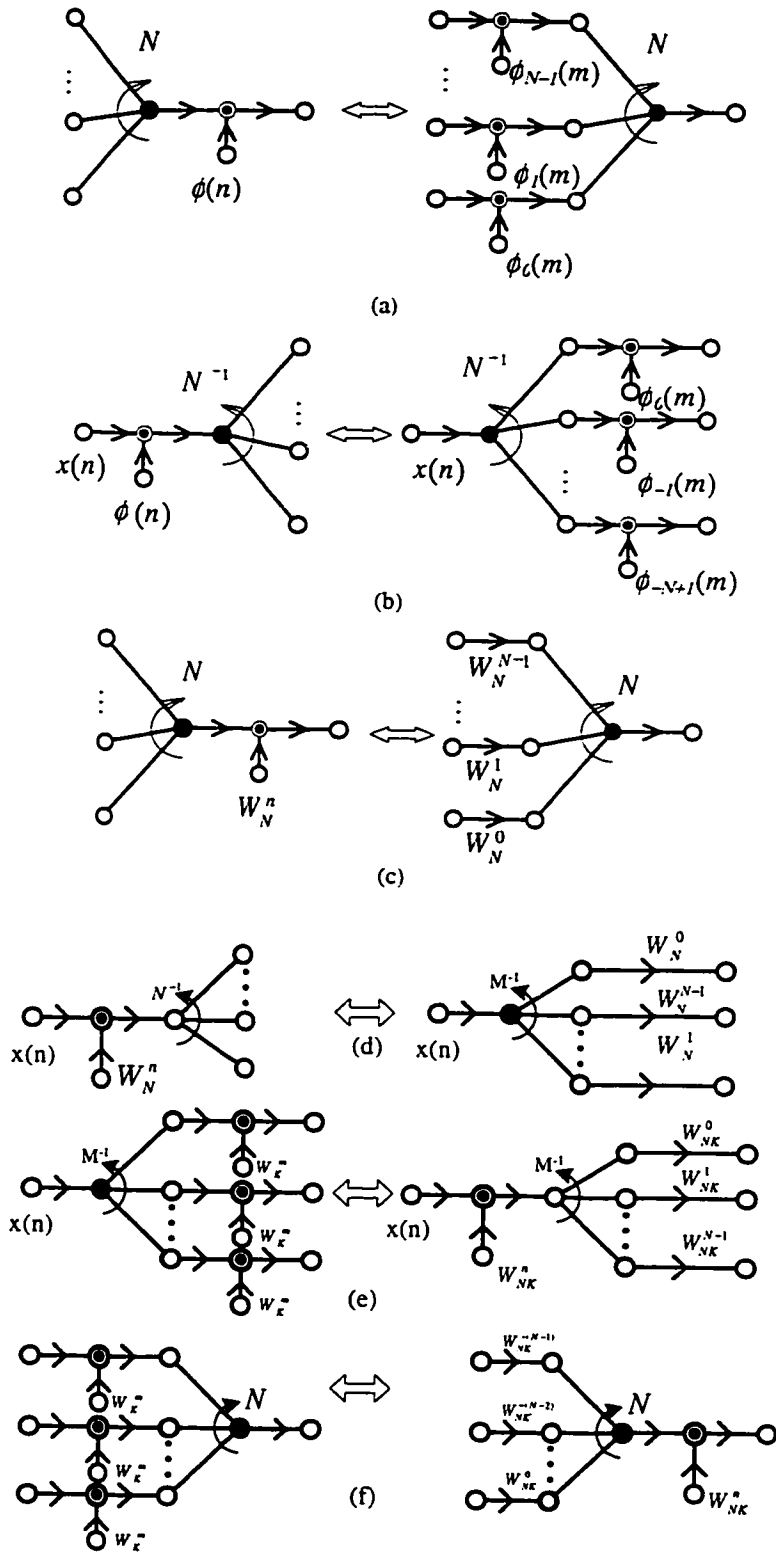


Figure 3-14 Commutator-SPC and SPC-commutator identities

3.8.4 Commutator-filter cascades

Some times, it is necessary to identify equivalent structures for commutators whose branches connect to identical filters as shown in Figure 3-15 (a) and (b). In these cases, applying with noble identities, we can move the filters to the other side of the commutators resulting in a cascade of commutator with an interpolated filter. If filter $H(z)$ is replaced with a pure delay z^{-k} , the situation will change, the delay can across the SPC-commutator, but the delay for each branches will be z^{l_i} , and the output sequence will change also to y_{j_i} , where $l_i = -\left\lfloor \frac{i-k}{N} \right\rfloor$ and $j_i = (i-k) \bmod N$. It is shown at Figure 3-15(c).

The notation $\lfloor x \rfloor$ indicates the floor function, which is the greatest integer less than or equal to x .

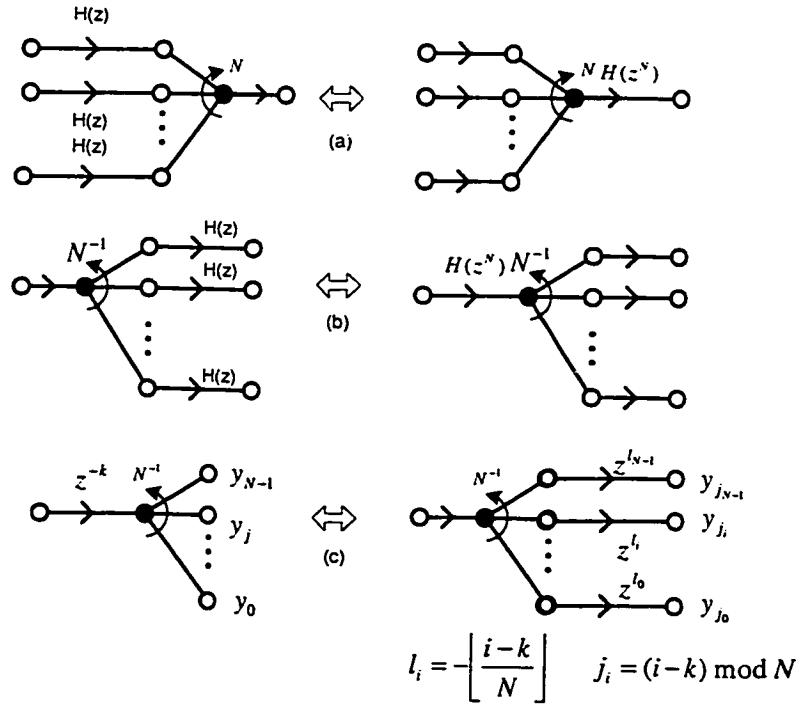


Figure 3-15 Commutator-filter identities

3.8.5 Commutator-sampler commutability

In MSFG transformation, it is often needed to move samplers across commutators, e.g., in upsampler-SPC and PSC-downsampler cascades. It can be shown that the commutation is allowed only when the sampler factor L of the sampler and the commutation factor M are co-prime (i.e., $\text{GCD}(L, M)=1$). Here, GCD--Great Common Divisor, is the greatest common divisor of two positive integers L and M , for example, $\text{GCD}(3,5)=1$. Also, the move of the sampler will lead to reordering of the commutator branches and time advance to the branch signals shown as Figure 3-16.

For an upsampler-SPC cascade (Figure 3-16(a)), if L and M are co-prime, the upsampler can be moved to the output of the SPC. The time advance introduced to the i -th branch of the SPC will be

$$l_i = \left\lfloor \frac{(M - 1 - i)L}{m} \right\rfloor \quad (3.9)$$

where l and m are any solution of

$$mL - lM = 1 \quad (3.10)$$

The original output poly phase signals will be reordered as a result of the transform by delivering k -th polyphase component $y_k(m)$, instead $y_i(m)$ at the i -th branch of the transformed structure. Where k is given by

$$k = M - 1 - \langle (M - 1 - i)L \rangle_M \quad (3.11)$$

Similarly, when the downsampler moves across the PSC in a PSC-downsampler cascade as shown in Figure 3-16 (b), the time advance introduced to the i -th branch of the PSC is

$$m_i = \left\lfloor \frac{im}{l} \right\rfloor \quad (3.12)$$

where l and m are any solution of

$$mL - lM = -1 \quad (3.13)$$

Again, as shown in Figure 3-16(b), the input poly-phase signal are reordered in the transformed network with $x_k(m)$ instead of $x_i(n)$, as the input signal to i -th channel, where k is determined by

$$k = \langle iM \rangle_L \quad (3.14)$$

The notation $\langle \bullet \rangle$ indicates modulo operation, that is, $\langle n \rangle_N \equiv n \bmod N$.

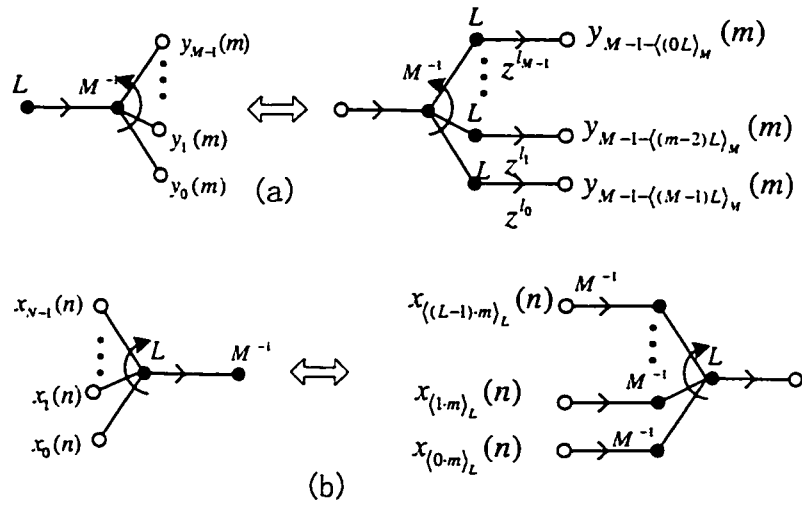


Figure 3-16 Sampler-commutator identities

A Computer Aided Design System for Transformation and Optimization of Multirate DSP systems

Instead of manipulating the MSFG manually, this computer aided design system can perform the MSFG transformations and optimization of multirate networks in an automatic or interactive manner.

In this chapter, we will introduce in detail the data structure, system block diagram, MSFG transformation, simulation, verification and optimization of this computer aided design system [11].

4.1 Objectives and requirements of the computer aided design systems

The main objective of this study is to develop a novel software system, in an automatic or interactive manner, to perform MSFG transformations and get one or more optimized MSFG under given optimization constraints.

This computer aided design system can be used for any areas where multirate DSP is encountered. It needs a friendly user interface to make it easy to perform MSFG transformations and optimization.

With this design system, one can represent a multirate system in the form of a MSFG and then optimize it by performing a series of MSFG transformations interactively. The system has the capability of calculating the output response at any CELL in the MSFG.

It therefore can verify the correctness of the derived network by comparing the response of the derived MSFG with that of the original MSFG.

To perform the MSFG transformation automatically or interactively, the following functions have been implemented in the system.

1. ***MSFG transformation and identities:*** includes 18 multirate operators and a number of MSFG transformations and identities in four categories.
2. ***Simulation of MSFG response:*** the response of any CELL to a deterministic or a random excitation applied at the MSFG input can be calculated, displayed and stored in a MSFG (.sfg) file. The length of random excitation can be changed.
3. ***Verification of the transformed MSFG:*** the simulator of the system allows the user to verify the correctness of the derived MSFG network without resorting to other tools for verification.
4. ***Complexity of MSFG:*** the system can calculate such useful statistics of a MSFG as the number of adders, multipliers, memories, etc. The system also calculates the computational complexity as a measure of effectiveness of a MSFG.

4.2 Operation System and Programming Language

Microsoft Windows Operation System is the most popular PC operating system nowadays. We choose a PC running Microsoft windows as the platform for our computer aided design system.

We use the Microsoft Visual C++ [24,31] to develop the design system because, in addition to its powerful object-oriented techniques [2,29,32], it also provides an integrated visual development environment for building Windows applications rapidly. Reusable MFC library [7,25] allows us to use standard GUI (Graphic User Interface) components and other foundation classes to rapidly develop a windows application system.

4.3 Data Structure

To show how the data structure is defined, let us take a look at the example shown in Figure 4-1. Figure 4-1(a) shows the MSFG of a two channel filter branches and Figure 4-1(b) shows the DSP block diagram of the system. In a MSFG, the graph is formed by interconnected MSFG operators and each of them has its own in/out signals (or links). In our system, each of the operators and together with its in/out links are represented using a unified data structure [1,6], called a “CELL”. Therefore, as shown in Figure 4-1(c), a MSFG can be represented by a number of interconnected CELLS (or a graph of CELLS), called a CELL Graph.

A commonly used data type in programming is array [6,13] because it is very easy to handle and program. Though simple, array can only be used when the structure of the array element and the array length are fixed. Another frequently used data type is pointer. It is very flexible because its length is changeable and hence can be used in situations where the structure of element is unfixed or varying. The drawback of using pointers is error-prone in programming and the system often crashes when error occurs.

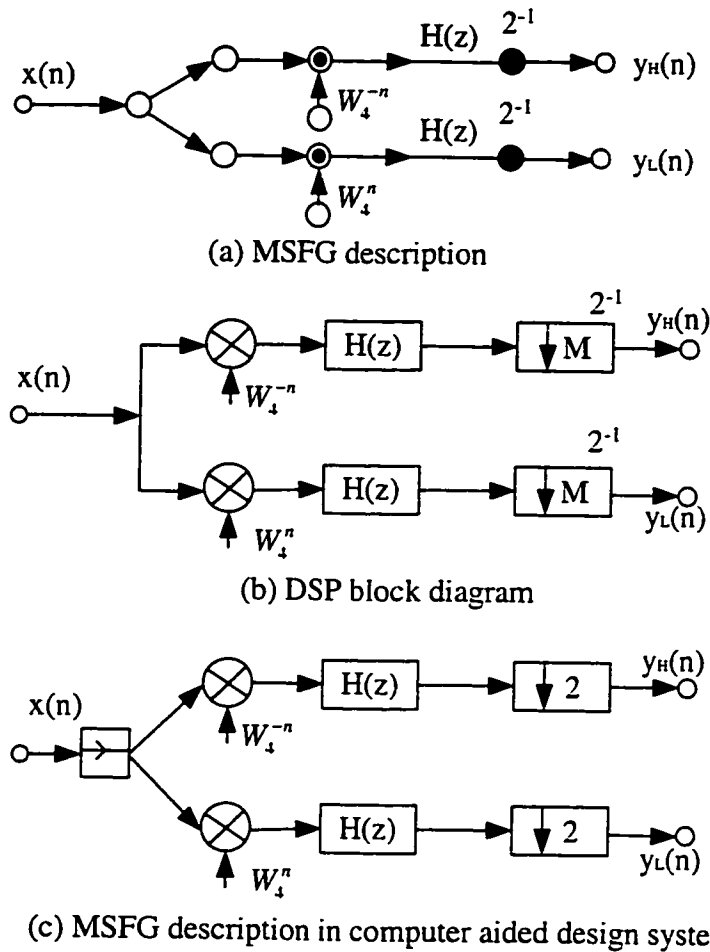


Figure 4-1 Description of an example in MSFG and DSP block diagram

Visual C++ provides a new class – CArray [31,33], which is defined as:

Template <class TYPE, class ARG_TYPE > Class CArray: public CObject

Parameters:

TYPE: Template parameter that specifies the type of objects stored in the array. TYPE is a parameter that is returned by CArray.

ARG_TYPE: Template parameter that specifies the argument type used to access objects stored in the array. Often a reference to TYPE. ARG_TYPE is a parameter that is passed to CArray.

The CArray class supports arrays that are similar to C arrays, but can dynamically shrink and grow as necessary. Hence, CArray has the advantage of both Array and

Pointer. With the use of links, it can describe the CELL graph in our application, so we choose the CArray to construct the data structure.

4.3.1 CELL Graph

A CELL Graph is defined by CArray <CELL, CELL> in this system, which is a group of interconnected CELLS. It begins with a Head CELL and ends with a (or more) Tail CELL(s). In our design system, MSFGs are represented using CELL Graphs. Figure 4-2(a) shows the system object logic relationship and Figure 4-2 (b) shows the connection relationship of an example. A CELL Graph contains many CELLS linked together. A CELL contains DATA, OPTYPE, Complexity and IntCArray. A DATA contains object Complex.

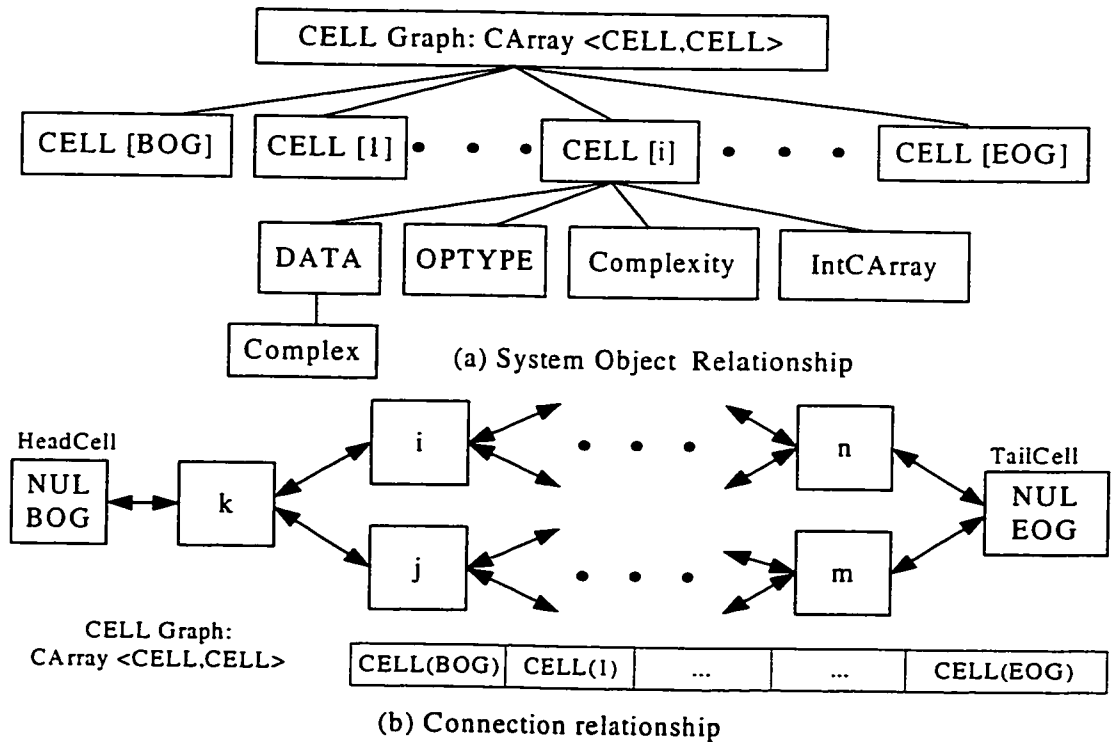


Figure 4-2 CELL Graph Description

In the example of Figure 4-2(b), the connection relationship among CELL[i], CELL[j] and CELL[k] can be described by:

CELL[k].NextCELLID[0]=j; CELL[k].NextCELLID[1]=i.
CELL[i].PrevCELLID[0]=k; CELL[j].PrevCELLID[0]=k.

4.3.2 CELL

A CELL class contains the following variables (properties):

1. **CELL ID:** CELL ID is a positive integer from BOG (Begin Of Graph) to EOG (End Of Graph), which indicates a unique CELL in the graph.
2. **Operation type:** it indicates the type of operation that the CELL performs. It is defined by an enumerated type OPTYPE (enum), for details see section 4.3.5.
3. **Operation parameter:** it specifies the parameters required by the operation. Operation parameters are DATA class, which will be discussed in section 4.3.3.
4. **Number of Input (NumOfInput):** the number of signal links that connect the preceding CELLS to the current CELL.
5. **Previous CELL ID (PrevCELLID):** CELL IDs of preceding CELLS that has direct link to the current CELL.
6. **Input Signal:** the signals link from the Previous CELLS to the current CELL.
7. **Number of Output (NumOfOutput):** the number of signal links that connect current CELL to Next CELLS.
8. **Next CELL ID (NextCELLID):** CELL IDs of Next CELLS that has direct link to the current CELL.
9. **Output Signal:** the signals link from the current CELL to Next CELLS.
10. **Complexity Information:** parameters on adder, multiplier, and memory counts as well as weighted (by sample rates) sum of the component counts of MSFG.

11. **Coordinate (X,Y)**: the coordinate of the current CELL in the graphic window.

The CELL class can be summarized by the table given below.

Table 4-1 CELL Class

	Field Name	Data Type
ID	CELL ID	Positive integer
Operator info.	Operation Type	OPTYPE
	Operation Parameters	DATA
	Num. Of Inputs	Positive integer
	Num. Of Outputs	Positive integer
Connection info.	Previous CELL ID	CArray <integer, integer>
	Next CELL ID	CArray <integer, integer>
In/out signals	Input Signal	CArray <DATA, DATA>
	Output Signal	CArray <DATA, DATA>
others	Complexity info.	Complexity
	Coordinate (X,Y)	Integer

4.3.3 DATA

As shown in Table 4-1, Operation Parameter, Input Signal and Output Signal in CELL Class, are all “DATA class” that is defined as following seven variables (properties):

1. **Field_1** : Integer;
2. **Field_2** : Integer;
3. **Field_3** : CString;
4. **Field_4** : CString;
5. **Sequential Parameter_1 (SeqPara_1)**: CArray <Complex, Complex>;
6. **Sequential Parameter_2 (SeqPara_2)**: CArray <Complex, Complex>;
7. **Response**: CArray <Complex, Complex>;

Properties 1-4 are mainly used to define some general parameters, for example, $(H_0(z^2))$ denotes $H_0(z^2)$ for Filter (OPFIL).

Properties 5-6 are used to store sequential parameters, for example, SeqPara_1=(1, 2+3i) denotes $H(z) = 1 + (2 + i3)z^{-1}$ for Filter (OPFIL).

Property 7 is used to store the response of the operator (CELL).

4.3.4 Complexity Information

As shown in CELL description, Complexity is a “ComplexityInfo” class that is defined as following five variables (properties):

1. The number of Adders (NumOfADD);
2. The number of Multipliers (NumOfMUL);
3. The number of Memories (NumOfMEM);
4. The number of Modulators (NumOfMOD);
5. The rate of Samples (SampleRate).

4.3.5 MSFG Operator

MSFG Operator information is only parts of the CELL information. It includes:

- *Number of Input (NumOfInput);*
- *Operation type (OperationType);*
- *Operation parameter (Parameter): Include Field_1 to Field_4 and SeqPara_1 to SeqPara_2.*

The MSFG Operator information (shown in Figure 4-3) is, by default,

- NumOfInput : =1;
- Field_1 and Field_2 : =EOG;
- Field_3 and Field_4 : “”;
- SeqPara_1 and SeqPara_2 : “”;

NumOfInput		Operation Type	
Field_1	Field_2	Field_3	Field_4
SeqPara_1			
SeqPara_2			

1		OPNUL	
EOG	EOG		

Figure 4-3 Default value of Operator

In Chapter 3 (see section 3.2), we have defined 18 operators. In our system, these operators are described as follows (If the value is the same as default value, it will be skipped):

1. **Modulator (OPMOD)**: It performs the multiplication of two sequences $x(n)$ and $\phi(n)$. See Figure 4-4(1).

The modulating signal $\phi(n)$ can be:

(1). An arbitrary sequence.

(2). A periodic sequence with period of N, for instance, $\phi(n) = W_N^n = e^{-j\frac{2\pi}{N}n}$.

- Field_1 : N (positive integer); EOG;
- Field_2 : k (positive integer); EOG;
- Field_3 : "W" or "phi";
- SeqPara_1 : "" $\phi[0,1,2\dots]$
- Response : $y(n) = x(n) \bullet \phi(n)$.

2. **Filter (OPFIL)**: It filters the input signal $x(n)$ with system transfer function $H(z)$ or Impulse Response $h(n)$. See Figure 4-4(2).

- Field_1 and Field_2 : integer;
- Field_3 and Field_4 : "H(z)" or "h(n)";
- SeqPara_1 : $(b_0, b_1, \dots, b_{M-1})$
- SeqPara_2 : $(a_0, a_1, \dots, a_{N-1})$ or "";

$$H(z) = \frac{\sum_{i=0}^{M-1} b_i z^{-i}}{\sum_{i=0}^{N-1} a_i z^{-i}} \quad \text{or} \quad \sum_{k=0}^{N-1} a_k y[n-k] = \sum_{m=0}^{M-1} b_m x[n-m]$$

- Response:

$$Y(z) = H(z) \bullet X(z) = \frac{\sum_{i=0}^{M-1} b_i z^{-i}}{\sum_{i=0}^{N-1} a_i z^{-i}} \bullet X(z) \quad \text{or} \quad y[n] = \frac{1}{a_0} \left(\sum_{m=0}^{M-1} b_m x[n-m] - \sum_{m=1}^{N-1} a_m y[n-m] \right)$$

3. **Down Sampler (OPDS):** It decreases the sampling rate of input signal $x(n)$ by a factor of M . See Figure 4-4(3).

- Field_1 : M (decimation factor, positive integer);
- Response : $y(n) = x(Mn), \quad n = 0, 1, 2, \dots$

4. **Up Sampler (OPUS):** It increases the sampling rate of input signal $x(n)$ by a factor of L . See Figure 4-4(4).

- Field_1 : N (interpolation factor, positive integer);
- Response : $y(n) = \begin{cases} x(n/L) & , \text{ for } n = 0, L, 2L, \dots \\ 0 & , \text{ otherwise} \end{cases}$

5. **S/P commutator (OPSPC):** It parallelizes the input to M sequences. As a result, the output sample rate is only one M -th of the input rate. See Figure 4-4(5).

- Field_1 : M (decimation factor, positive integer);
- Response : $y_{M-1}(m) = x[0], x[M], x[2*M], \dots x[i*M] \dots$

...

$$y_1(m) = 0, \quad x[2], x[M+2], \dots x[i*M+2] \dots$$

$$y_0(m) = 0, \quad x[1], x[M+1], \dots x[i*M+1] \dots$$

6. **P/S commutator (OPPSC):** It serializes L simultaneous sequences into one output sequence. Therefore, the output rate is L times higher than that of the input. See Figure 4-4(6).

- NumOfInput : =L;
- Field_1 : =L (interpolation factor, positive integer);
- Response : $y(n) = x_0[0], x_1[0], \dots, x_{N-1}[0], x_0[1], x_1[1], \dots, x_{N-1}[1], \dots$

7. **Adder (OPADD):** It sums a group of input signals x_0 to x_{N-1} . See Figure 4-4(7).

- NumOfInput : N (positive integer);
- Response : $y[m] = \sum_{i=0}^{N-1} x_i[m]$

8. **Multiplier (OPMUL):** It multiplies the input signal $x(n)$ with a real or complex factor α . See Figure 4-4(8).

- NumOfInput : 1 N (positive integer);
- Field_1 : integer (α is integer factor); EOG
- SeqPara_1 : real or complex number ; -----
- Response : $y[n] = x[n] \cdot \alpha$ $y[n] = \sum_{i=0}^{N-1} x_i[n]$

9. **Null Operator (OPNUL):** It doesn't perform any operation. As a result, the output $y(n)$ is exactly the same as input $x(n)$. See Figure 4-4(9).

- Response : $y[n] = x[n]$

10. **Sampling and hold (OPSH):** It does not actually change the sampling rate of input signal $x(n)$. It just retains (holds) the signal value at the sampling point a number of times N (SH factor). See Figure 4-4(10).

- Field_1 : N (positive integer, SH factor);
- Response : $y[n] = x\left[\left\lfloor \frac{n}{N} \right\rfloor \cdot N\right]$

11. **Upsampling and hold (OPUSH):** It does increase the sampling rate of input signal $x(n)$ by repeating (holding) every signal element of the input signal L times where L is the USH factor. See Figure 4-4(11).

- Field_1 : L (positive integer, USH factor);
- Response : $y[n] = x\left[\left\lfloor \frac{n}{N} \right\rfloor\right]$

12. **Integral & dump and hold (OPIDU):** An IDU with decimation factor M integrates (accumulates) every M signal elements and dumps the integral to output reducing the sampling rate by a factor of M . See Figure 4-4(12).

- Field_1 : M (positive integer, decimation factor);
- Response : $y[n] = \sum_{i=0}^{M-1} x[(n-1) \cdot M + i + 1]$

13. **Sampler (OPSA):** It samples the incoming sequence $x(n)$ without holding the sampled signal elements where the CELL value k_N denotes the sampling instance n that satisfied $\langle n - k \rangle_N = 0$. See Figure 4-4(13).

- Field_1 : N (positive integer);
- Field_2 : k (positive integer);
- Response : $y(n) = \begin{cases} x(n) & , \text{ for } n = k, N + k, 2N + k, \dots \\ 0 & , \text{ otherwise} \end{cases}$

14. **Delay element (OPDELAY):** It delays the input by an integer number of samples. See Figure 4-4(14).

- Field_1 : k (delay factor);
- Response : $y[n] = x[n - k]$

15. **DFT (OPDFT):** This is a MIMO (Multi-Input Multi-Output) operator. It performs the Discrete Fourier Transform to the input. See Figure 4-4(15).

- NumOfInput, NumOfOutput : $=N$;

- Field_1 : N (positive integer);
- Response : $X[k] = \sum_{n=0}^{N-1} x[n] \bullet W_N^{kn}$, for $k=0,1,\dots,N-1$.

16. **Interleaver (OPINTL)**: Also an MIMO operator. It reshuffles the input data blocks according to a specific order. See Figure 4-4(16).

- NumOfInput, NumOfOutput : N (positive integer);
- Field_1 : N (positive integer);
- Field_2 : k_0 (positive integer);
- Field_3 : k (positive integer);
- Field_4 : step (positive integer);

17. **Switch (OPSW)**: Also an MIMO operator. It acts as a space switch between N- input ports and N output ports. See Figure 4-4(17).

- NumOfInput, NumOfOutput : N (positive integer);
- SeqPara_1 : a_0, a_1, \dots, a_{N-1} ;
- SeqPara_2 : b_0, b_1, \dots, b_{N-1} ;
- Response : $y_{b_i}(n) = x_{a_i}(n)$ for $i=0$ to $N-1$.

18. **Overlapped S/P commutator (OPOSPC)**: Similar to SPC, but instead of parallelizes the input into M sequences, where M is the decimation factor. There are overlaps between adjacent output sequences. See Figure 4-4(18).

- NumOfOutput : $>M$;
- Field_1 : M (decimation factor, positive integer);
- Response : $y_{N-1}(m) = x[-M+N], x[N], x[M+N], \dots, x[i*M+N] \dots$
 $y_M(m) = x[1], x[M+1], x[M+M+1], \dots, x[i*M+M+1] \dots$
 $y_{M-1}(m) = x[0], x[M], x[M+M], \dots, x[i*M+M] \dots$
 \dots
 $y_1(m) = 0, x[2], x[M+2], \dots, x[i*M+2] \dots$
 $y_0(m) = 0, x[1], x[M+1], \dots, x[i*M+1] \dots$

<table border="1" style="width: 100%; border-collapse: collapse;"> <tr> <td colspan="2" style="text-align: center;">I</td> <td colspan="2" style="text-align: center;">OPMOD</td> </tr> <tr> <td style="text-align: center;">N</td> <td style="text-align: center;">k</td> <td style="text-align: center;">W</td> <td style="text-align: center;"></td> </tr> <tr> <td colspan="4" style="text-align: center;">= W_N^k</td> </tr> <tr> <td colspan="4" style="height: 20px;"></td> </tr> <tr> <td colspan="4" style="height: 20px;"></td> </tr> </table>	I		OPMOD		N	k	W		= W_N^k												<table border="1" style="width: 100%; border-collapse: collapse;"> <tr> <td colspan="2" style="text-align: center;">I</td> <td colspan="2" style="text-align: center;">OPMOD</td> </tr> <tr> <td style="text-align: center;">EOG</td> <td style="text-align: center;">EOG</td> <td style="text-align: center;">phi</td> <td style="text-align: center;"></td> </tr> <tr> <td colspan="4" style="text-align: center;">[0.1.2....]</td> </tr> <tr> <td colspan="4" style="text-align: center;">= $\phi(n)$</td> </tr> <tr> <td colspan="4" style="height: 20px;"></td> </tr> </table>	I		OPMOD		EOG	EOG	phi		[0.1.2....]				= $\phi(n)$																																
I		OPMOD																																																																
N	k	W																																																																
= W_N^k																																																																		
I		OPMOD																																																																
EOG	EOG	phi																																																																
[0.1.2....]																																																																		
= $\phi(n)$																																																																		
(1) Modulator (OPMOD=1)																																																																		
<table border="1" style="width: 100%; border-collapse: collapse;"> <tr> <td colspan="2" style="text-align: center;">I</td> <td colspan="2" style="text-align: center;">OPFIL</td> </tr> <tr> <td style="text-align: center;">i</td> <td style="text-align: center;">k</td> <td style="text-align: center;">H</td> <td style="text-align: center;">z</td> </tr> <tr> <td colspan="4" style="text-align: center;">= $H_i(z^k)$</td> </tr> <tr> <td colspan="4" style="text-align: center;">Transfer Function</td> </tr> <tr> <td colspan="4" style="text-align: center;">b_0, b_1, \dots, b_M</td> </tr> <tr> <td colspan="4" style="text-align: center;">a_0, a_1, \dots, a_N</td> </tr> </table>	I		OPFIL		i	k	H	z	= $H_i(z^k)$				Transfer Function				b_0, b_1, \dots, b_M				a_0, a_1, \dots, a_N				<table border="1" style="width: 100%; border-collapse: collapse;"> <tr> <td colspan="2" style="text-align: center;">I</td> <td colspan="2" style="text-align: center;">OPFIL</td> </tr> <tr> <td style="text-align: center;">i</td> <td style="text-align: center;">EOG</td> <td style="text-align: center;">h</td> <td style="text-align: center;">n</td> </tr> <tr> <td colspan="4" style="text-align: center;">= $h_i(n)$</td> </tr> <tr> <td colspan="4" style="text-align: center;">Unit Impulse Response</td> </tr> <tr> <td colspan="4" style="text-align: center;">b_0, b_1, \dots, b_M</td> </tr> <tr> <td colspan="4" style="text-align: center;">a_0, a_1, \dots, a_N</td> </tr> </table>	I		OPFIL		i	EOG	h	n	= $h_i(n)$				Unit Impulse Response				b_0, b_1, \dots, b_M				a_0, a_1, \dots, a_N				<table border="1" style="width: 100%; border-collapse: collapse;"> <tr> <td colspan="2" style="text-align: center;">I</td> <td colspan="2" style="text-align: center;">OPDS</td> </tr> <tr> <td style="text-align: center;">M</td> <td style="text-align: center;"></td> <td style="text-align: center;"></td> <td style="text-align: center;"></td> </tr> <tr> <td colspan="4" style="text-align: center;">M: decimation factor</td> </tr> <tr> <td colspan="4" style="height: 20px;"></td> </tr> </table>	I		OPDS		M				M: decimation factor							
I		OPFIL																																																																
i	k	H	z																																																															
= $H_i(z^k)$																																																																		
Transfer Function																																																																		
b_0, b_1, \dots, b_M																																																																		
a_0, a_1, \dots, a_N																																																																		
I		OPFIL																																																																
i	EOG	h	n																																																															
= $h_i(n)$																																																																		
Unit Impulse Response																																																																		
b_0, b_1, \dots, b_M																																																																		
a_0, a_1, \dots, a_N																																																																		
I		OPDS																																																																
M																																																																		
M: decimation factor																																																																		
$H(z) = \frac{b_0 + b_1 z^{-1} + \dots + b_{M-1} z^{-(M-1)}}{a_0 + a_1 z^{-1} + \dots + a_{N-1} z^{-(N-1)}}$		(2) Filter (OPFIL=2)																																																																
(4) Up Sampler (OPUS=4)		(5) S/P commutator (OPSPC=5)																																																																
<table border="1" style="width: 100%; border-collapse: collapse;"> <tr> <td colspan="2" style="text-align: center;">I</td> <td colspan="2" style="text-align: center;">OPUS</td> </tr> <tr> <td style="text-align: center;">L</td> <td style="text-align: center;"></td> <td style="text-align: center;"></td> <td style="text-align: center;"></td> </tr> <tr> <td colspan="4" style="text-align: center;">L: interpolation factor</td> </tr> <tr> <td colspan="4" style="height: 20px;"></td> </tr> </table>	I		OPUS		L				L: interpolation factor								<table border="1" style="width: 100%; border-collapse: collapse;"> <tr> <td colspan="2" style="text-align: center;">I</td> <td colspan="2" style="text-align: center;">OPSPC</td> </tr> <tr> <td style="text-align: center;">M</td> <td style="text-align: center;"></td> <td style="text-align: center;"></td> <td style="text-align: center;"></td> </tr> <tr> <td colspan="4" style="text-align: center;">M: decimation factor</td> </tr> <tr> <td colspan="4" style="height: 20px;"></td> </tr> </table>	I		OPSPC		M				M: decimation factor								<table border="1" style="width: 100%; border-collapse: collapse;"> <tr> <td colspan="2" style="text-align: center;">NumOfInput</td> <td colspan="2" style="text-align: center;">OPPSC</td> </tr> <tr> <td style="text-align: center;">L</td> <td style="text-align: center;"></td> <td style="text-align: center;"></td> <td style="text-align: center;"></td> </tr> <tr> <td colspan="4" style="text-align: center;">L: interpolate factor</td> </tr> <tr> <td colspan="4" style="height: 20px;"></td> </tr> </table>	NumOfInput		OPPSC		L				L: interpolate factor																							
I		OPUS																																																																
L																																																																		
L: interpolation factor																																																																		
I		OPSPC																																																																
M																																																																		
M: decimation factor																																																																		
NumOfInput		OPPSC																																																																
L																																																																		
L: interpolate factor																																																																		
(7) Adder (OPADD=7)		(8) Multiplier (OPMUL=8)																																																																
<table border="1" style="width: 100%; border-collapse: collapse;"> <tr> <td colspan="2" style="text-align: center;">NumOfInput</td> <td colspan="2" style="text-align: center;">OPADD</td> </tr> <tr> <td style="text-align: center;"></td> <td style="text-align: center;"></td> <td style="text-align: center;"></td> <td style="text-align: center;"></td> </tr> <tr> <td colspan="4" style="height: 20px;"></td> </tr> </table>	NumOfInput		OPADD										<table border="1" style="width: 100%; border-collapse: collapse;"> <tr> <td colspan="2" style="text-align: center;">I</td> <td colspan="2" style="text-align: center;">OPMUL</td> </tr> <tr> <td style="text-align: center;">alpha</td> <td style="text-align: center;"></td> <td style="text-align: center;"></td> <td style="text-align: center;"></td> </tr> <tr> <td colspan="4" style="text-align: center;">alpha</td> </tr> <tr> <td colspan="4" style="height: 20px;"></td> </tr> </table>	I		OPMUL		alpha				alpha								<table border="1" style="width: 100%; border-collapse: collapse;"> <tr> <td colspan="2" style="text-align: center;">NumOfInput</td> <td colspan="2" style="text-align: center;">OPMUL</td> </tr> <tr> <td style="text-align: center;"></td> <td style="text-align: center;"></td> <td style="text-align: center;"></td> <td style="text-align: center;"></td> </tr> <tr> <td colspan="4" style="height: 20px;"></td> </tr> </table>	NumOfInput		OPMUL																																	
NumOfInput		OPADD																																																																
I		OPMUL																																																																
alpha																																																																		
alpha																																																																		
NumOfInput		OPMUL																																																																
(9) NULL operator (OPNUL=9)		(10) Sampling & hold (OPSH=10)																																																																
<table border="1" style="width: 100%; border-collapse: collapse;"> <tr> <td colspan="2" style="text-align: center;">I</td> <td colspan="2" style="text-align: center;">OPNUL</td> </tr> <tr> <td style="text-align: center;"></td> <td style="text-align: center;"></td> <td style="text-align: center;"></td> <td style="text-align: center;"></td> </tr> <tr> <td colspan="4" style="height: 20px;"></td> </tr> </table>	I		OPNUL										<table border="1" style="width: 100%; border-collapse: collapse;"> <tr> <td colspan="2" style="text-align: center;">I</td> <td colspan="2" style="text-align: center;">OPSH</td> </tr> <tr> <td style="text-align: center;">N</td> <td style="text-align: center;"></td> <td style="text-align: center;"></td> <td style="text-align: center;"></td> </tr> <tr> <td colspan="4" style="text-align: center;">N: SH factor</td> </tr> <tr> <td colspan="4" style="height: 20px;"></td> </tr> </table>	I		OPSH		N				N: SH factor								<table border="1" style="width: 100%; border-collapse: collapse;"> <tr> <td colspan="2" style="text-align: center;">I</td> <td colspan="2" style="text-align: center;">OPUSH</td> </tr> <tr> <td style="text-align: center;">L</td> <td style="text-align: center;"></td> <td style="text-align: center;"></td> <td style="text-align: center;"></td> </tr> <tr> <td colspan="4" style="text-align: center;">L: interpolate factor</td> </tr> <tr> <td colspan="4" style="height: 20px;"></td> </tr> </table>	I		OPUSH		L				L: interpolate factor																											
I		OPNUL																																																																
I		OPSH																																																																
N																																																																		
N: SH factor																																																																		
I		OPUSH																																																																
L																																																																		
L: interpolate factor																																																																		
(11) Upsampling & hold (OPUSH=11)																																																																		

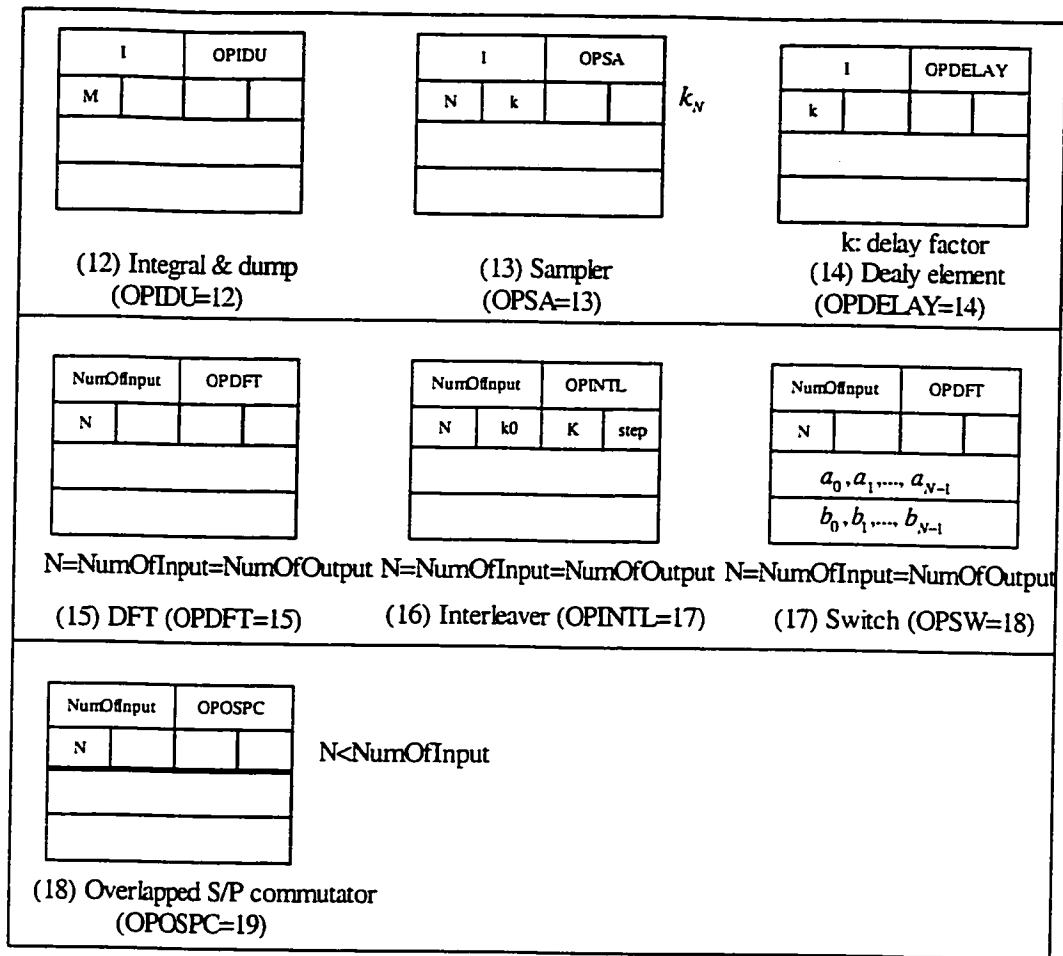


Figure 4-4 Operators Description

4.4 System Block Diagram

The system architecture of our computer-aided design system is given in Figure 4-5. It can be seen that signal processing and data handling operations (i.e., CELL operation, File I/O, MSFG transformations, simulation, verification, optimization etc.) are performed by the corresponding functional modules.

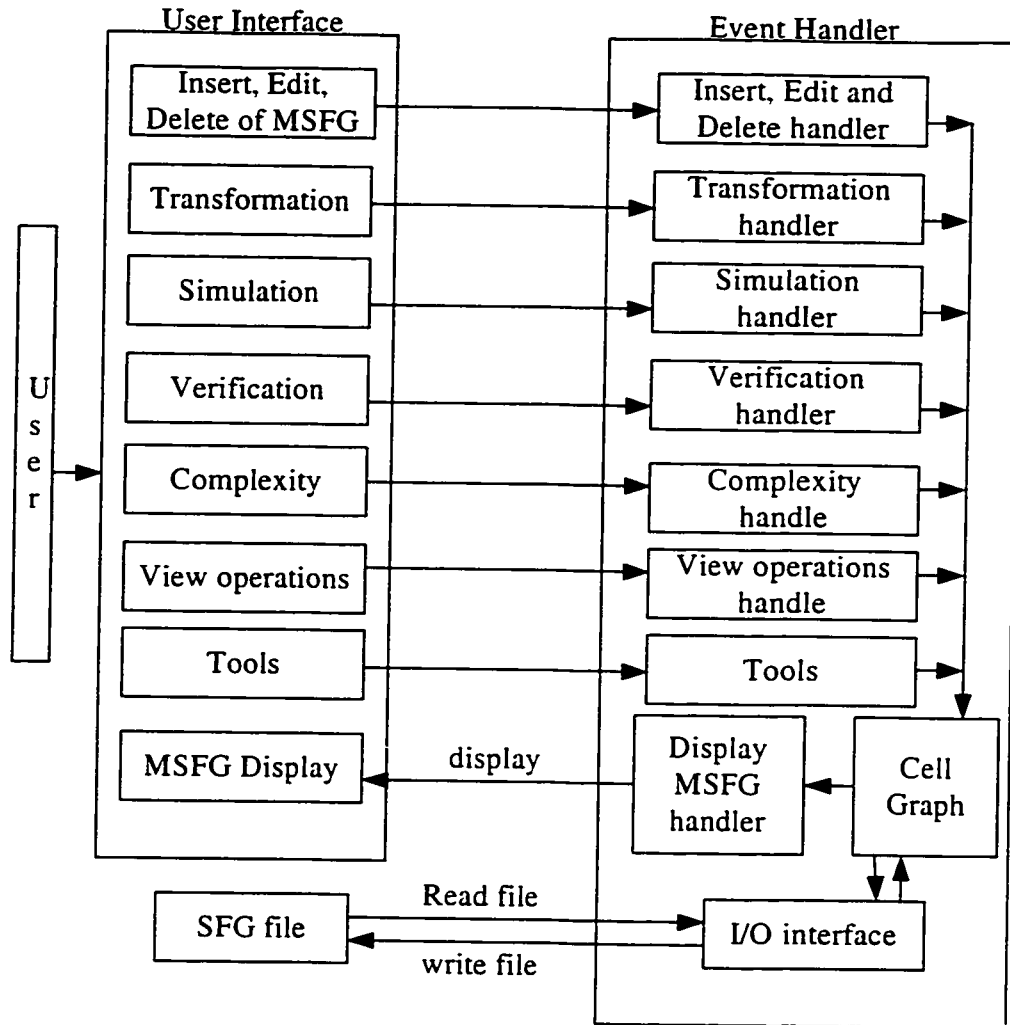


Figure 4-5 System Block Diagram

4.5 Class Model

4.5.1 Class Model for general application program with MFC

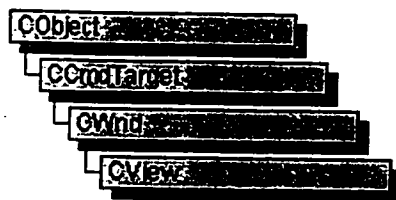
For general application program with MFC [7,24], there are following important basic classes:

1. **CObject**: CObject is the principal base class for the Microsoft Foundation Class Library. It serves as the root not only for library classes such as CFile and COBList, but also for the classes that users write. CObject provides basic services, including

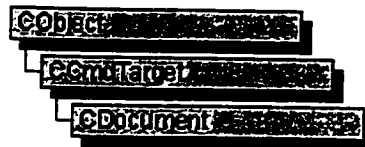
- Serialization support
 - Run-time class information
 - Object diagnostic output
 - Compatibility with collection classes
2. **CView:** The CView class provides the basic functionality for user-defined view classes. A view is attached to a document and acts as an intermediary between the document and the user: the view renders an image of the document on the screen or printer and interprets user input as operations upon the document.
 3. **CDocument:** The CDocument class provides the basic functionality for user-defined document classes. A document represents the unit of data that the user typically opens with the File Open command and saves with the File Save command.

CDocument supports standard operations such as creating a document, loading it, and saving it. The framework manipulates documents using the interface defined by CDocument.
 4. **CDialog:** The CDialog class encapsulates the functionality of a Windows dialog box.
 5. **CWnd:** The CWnd class provides the base functionality of all window classes in the Microsoft Foundation Class Library.

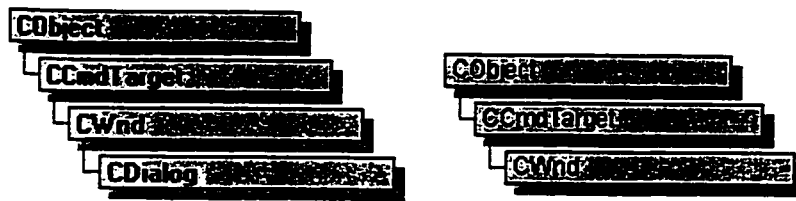
The hierarchy chart and relation of these classes are shown in Figure 4-6(a) to (e).



(a). CView Class

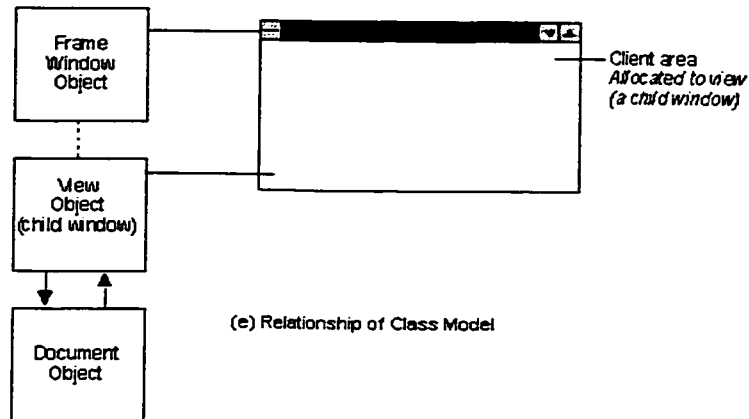


(b). CDocument Class



(c). CDIALOG Class

(d). CWnd Class



(e) Relationship of Class Model

Figure 4-6 Classes Model and Relationship

4.5.2 Class Model for this computer aided design system

In our design system, there are two important classes derived from CView and CDocument:

1. **CSignalFlowView class** : mainly used to display, input and output data.
2. **CSignalFlowDoc class**: it supports standard operations such as creating a document, loading it, and saving it. It also is used to process data, check the condition of MSFG transformation and perform MSFG transformations.

Moreover, the computer aided design system includes the following other classes (the relationship among them are clearly shown in Figure 4-7):

- **11 classes derived from CObject class**: they will be used to process data, check the condition of MSFG transformation and perform MSFG transformations.

- **7 classes derived from CDialog class:** they will be used to input / output data and option selection of MSFG transformations with a popup dialog window.
- **2 classes derived from CWnd class:** in our system, two Active-X controls are used to display the table and chart of verification result.

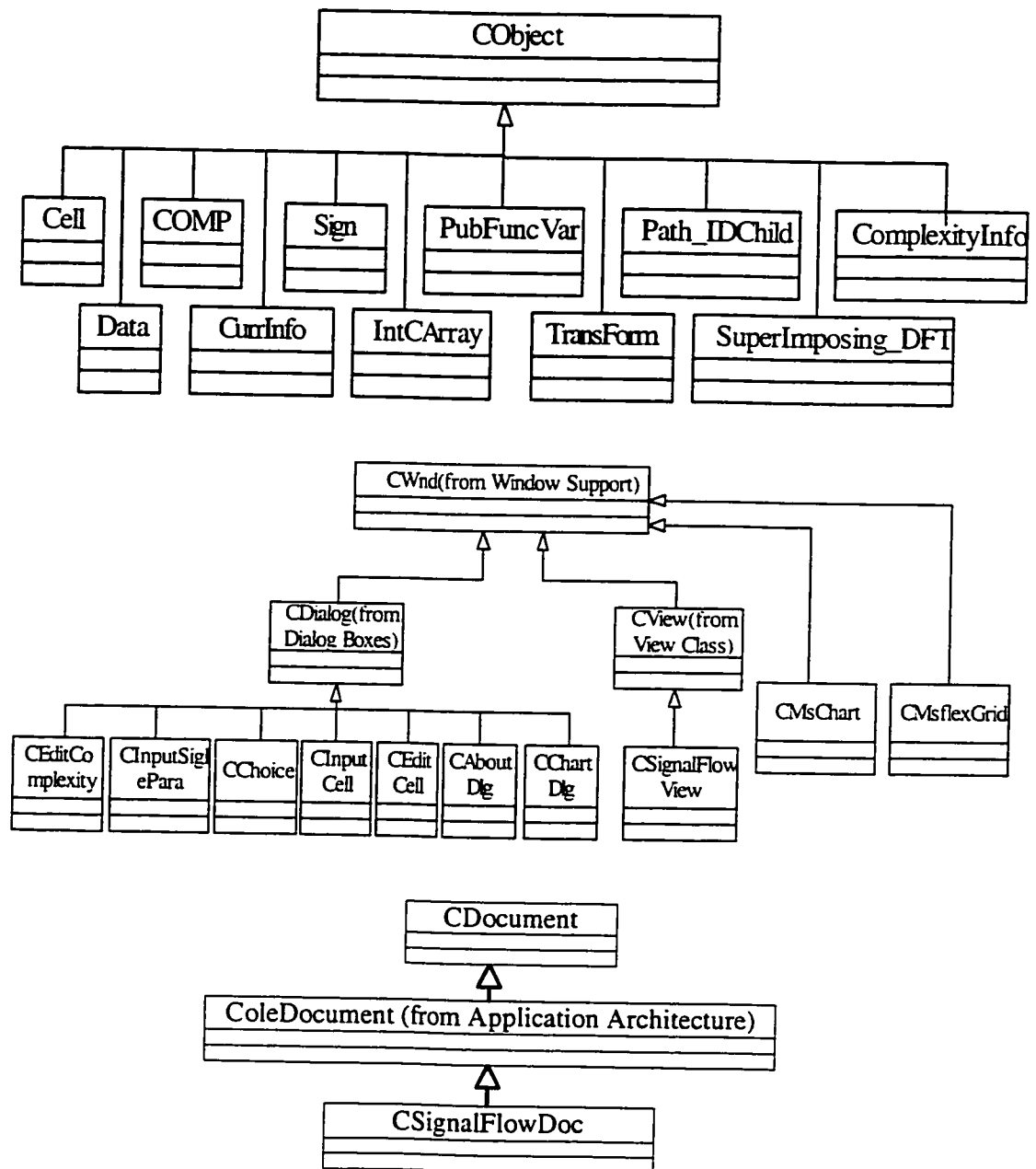


Figure 4-7 Classes Model

4.5.2.1 Classes derived from CObject:

These 11 classes that derived from CObject are mainly used to process data (including operations of CELL, MSFG transformations etc.), which are detailed described in following table.

Table 4-2 Classes derived from CObject

No	Class Name	Function
1.	ComplexityInfo	Information and operations about the analysis and statistic of a MSFG
2.	TransForm	Some MSFG transformations (most MSFG transformations are defined in class CSignalFlowDoc)
3.	PubFunVar	Most public functions and variables
4.	IntCArray	A CArray of Integer
5.	Superposition _DFT	Definition of operator DFT, transformation of Super Imposing and some related functions
6.	Sign	All Signs (marks) used in this system
7.	CELL	Definition of CELL
8.	CurrInfo	Current Information
9.	Data	Definition of Data
10	COMP	Definition of Complex number
11	Path_IDChild	Definition of Path

4.5.2.2 Classes Derived from CDialog:

These 7 classes that derived from CDialog are mainly used to input / output data and option selection of MSFG transformation with a popup dialog, which are described in Table 4-3.

Table 4-3 Classes derived from CDialog

No	Class Name	Function description
1.	CEditComplexity	Edit the parameters of analysis and statistic information
2.	CInputSiglePara	Input one parameter
3.	CChoice	Select the choice of MSFG transformation
4.	CInputCELL	Input a new CELL
5.	CEditCELL	Edit an existed CELL
6.	CAboutDlg	Display version information of system
7.	CChartDlg	Display the table and chart of comparison and verification for MSFG transformations

4.5.2.3 Classes Derived from CView:

In this system, only one class (CSignalFlowView) is derived from class CView. It is mainly used to display the data (CELL Graph) information, draw MSFG figure, and accept the command of users. Its main functions are shown in Table 4-4.

Table 4-4 Functions in CSignalFlowView

No	Function Name	Function description
1.	AddTransToolsBar	Add corresponding Tools Bar of MSFG transformation
2.	Coordinate2ID	Change Coordinate (X,Y) to CELL ID
3.	GetXY	Get the Coordinate (X,Y) from specified CELL
4.	OnHotkey	Check hotkey and execute corresponding functions
5.	OnDraw	Draw the MSFG figure
6.	OnFlowdspAutodraw	Automatic calculate the (X,Y) and draw MSFG figure
7.	OnLButtonDbkClk	If double click L button in a CELL, edit this CELL
8.	OnLButtonDown	If L button of mouse is down, select current CELL or current signal (link)
9.	OnLButtonUp	Move a CELL, connect two CELL, or select all CELLS in current area
10	OnRButtonDown	If R button is down, add a corresponding tools bar of MSFG transformation

4.5.2.4 Classes Derived from CDocument:

In this system, only one class (CSignalFlowDoc) is derived from CDocument class. It is mainly used to process the data (CELL Graph) information, check the condition of MSFG transformation and perform the MSFG transformations when the condition is satisfied. Its mainly functions are shown in Table 4-5.

Table 4-5 Functions in CSignalFlowDoc

No	Function Name	Function description
1.	ChangeRelation	Change the connection after a Switch vector is defined
2.	ChangeID	Change the ID of a CELL
3.	ComparePara	Compare two parameters
4.	ConditionXXXADDY	Check the condition of 3-stages ($X+Adder+Y$)
5.	ConditionXY	Check the condition of 2-stages ($X+Y$)
6.	ConditionXY_Para	Check the condition of 2-stages ($X+Y$) and parameter
7.	ConditionXYZ_Para	Check the condition of 3-stages ($X+Y+Z$) and parameter
8.	EditCELL	Edit a CELL
9.	ExchangeCELL	exchange two CELLS
10	FileSave	Save a MSFG File
11	FlowdspEnlarge	Zoom in or Zoom out of a MSFG graph
12	InOutPaste	Paste a CELL or a group of CELLS
13	InsertCELL	Insert a CELL after current CELL
14	NewCELL	Build a new CELL
15	OnBitVerifyCal	Simulation of MSFG
16	OnBitVerifyCompare	Comparison of simulation result in two signals (links)
17	OnBitVerifySetSR	Set the length of excitation signal
18	OnInoutCopy	Copy a CELL or a group of CELLS
19	OnInoutDeletepath	Delete a signal (link)
20	OnInoutDelCELL	Delete a CELL

21	OnInoutInputCELL	Input a new CELL
22	OnOperationRedo	Redo of MSFG transformation
23	OnOperationUndo	Undo of MSFG transformation
24	OnPFCal	Calculate the complexity information of a MSFG
25	OnPlot	Plot the figure of comparison
26	OnT_R...	MSFG transformation. Refer to section 4.7
27	OnU_R...	Check the condition of MSFG transformation. Refer to section 4.7

4.5.2.5 Classes Derived from CWnd:

1. CMSChart: an Active X control is used to plot the curve of verification result.
2. CMSFlexGrid: an Active X control is used to display the data of verification result.

4.6 Interface

4.6.1 User Menu

4.6.1.1 Main Menu

With MFC, we can easily design a menu for the computer aided design system. According to the objective and requirements of this system, the main menu is designed with following 10 items (See Figure 4-8):

- 1). **File:** operations for MSFG file (.sfg). It includes the following functions:
 - (a). Open and Close MSFG file (.sfg);
 - (b). Save and Save as MSFG file (.sfg);
 - (c). Build new MSFG file (.sfg).
- 2). **Edit:** operations for existed CELL or Branch (Signal). It includes the following functions:
 - (a). Cut, Copy and Paste a CELL or a group of CELLS.
 - (b). Input a new CELL, Edit an existed CELL and Delete/Remove a CELL.

- (c). Edit a Signal, Delete/Remove a Signal.
 - (d). Redo or Undo of MSFG transformations.
- 3). **View:** operations for displaying MSFG figure. It includes:
- (a). Movement (a CELL, a group of CELLS or whole figure): left, right, up or down.
 - (b). Zoom In/Zoom Out/Reset.
 - (c). Automatic drawing.
 - (d). Enable / Disable the display of Toolbar and Status bar.
- 4). **Window:** Windows arrangement.
- 5). **Transformation:** operations for MSFG transformation. For details, see section 4.6.1.2.
- 6). **Complexity:** analysis and statistic information of a MSFG. For details, refer to section 4.9.
- 7). **Verification:** Simulation and verification of MSFG transformation. The details see section 4.8.
- 8). **Tools:** Include some useful tools for MSFG transformation:
- (a). Remove redundant CELLS;
 - (b). Check zero CELLS;
 - (c). Display the information of a specified CELL. Some other often-used tools are displayed in the tools bar in Figure 4-10.
- 9). **Help:** display information about this computer aided design system.

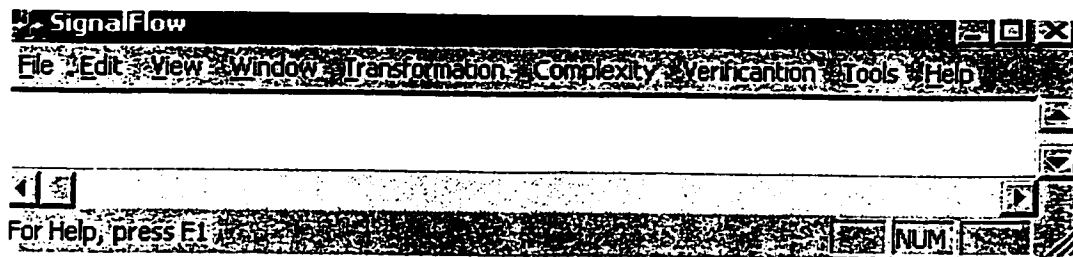


Figure 4-8 Main Menu

4.6.1.2 MSFG Transformation Submenu

All the MSFG transformations are put in MSFG transformation submenu, which are shown in Figure 4-9. It includes:

- (a). 18 categories of MSFG transformations corresponding to 18 different operators. For the details, refer to section 3.2.
- (b). Superposition Theorem.
- (c). Display corresponding tools bar.

The corresponding MSFG transformation submenu will become “real (highlight)” if the condition of MSFG transformation is satisfied; otherwise, the corresponding submenu will become “virtual (lowlight)”.

To perform MSFG transformation,

- 1. Choose a CELL that you want to transform;
- 2. According to current CELL (operator) type, choose the corresponding MSFG transformation submenu.
- 3. Among many MSFG transformations, choose one MSFG transformation that you want to transform.

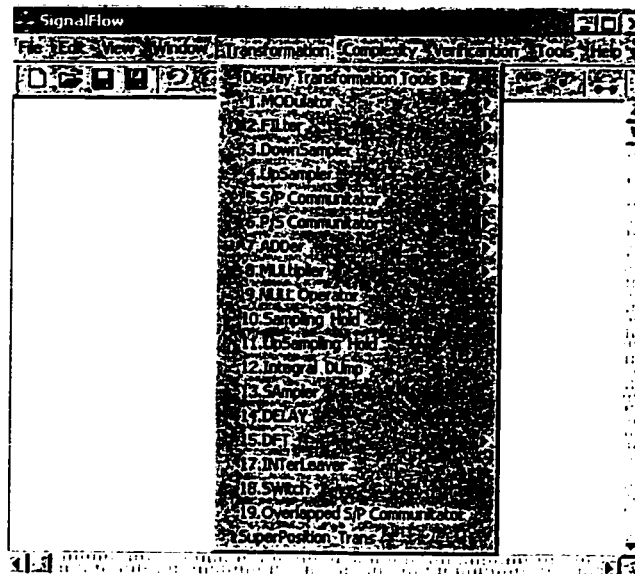


Figure 4-9 MSFG transformation submenu

4.6.1.3 MSFG Transformation Tools Bar (Figure Menu)

The operation of MSFG transformation, which is described in section 4.6.1.2 is not very easy to operate. It is because:

1. User need select a transformation among many MSFG transformations;
2. The description about MSFG transformation is only in text, so it is not easy for the system to understand.

Based on the above reasons, we analyze some professional software and then design the following MSFG Transformation Tools Bar (Figure Menu). This tools bar is shown in Figure 4-10.

When we select a CELL and click the right button of the mouse or select Menu "Transformation--Display Transformation Tools Bar", the corresponding MSFG transformation tool bar will be displayed on the bottom of screen.

The corresponding MSFG transformation figures will become enable (highlight) if the condition is satisfied; otherwise, the figures will become disable (dim).

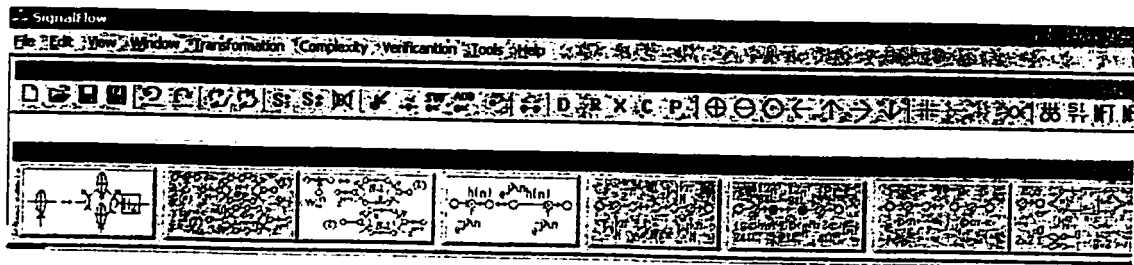
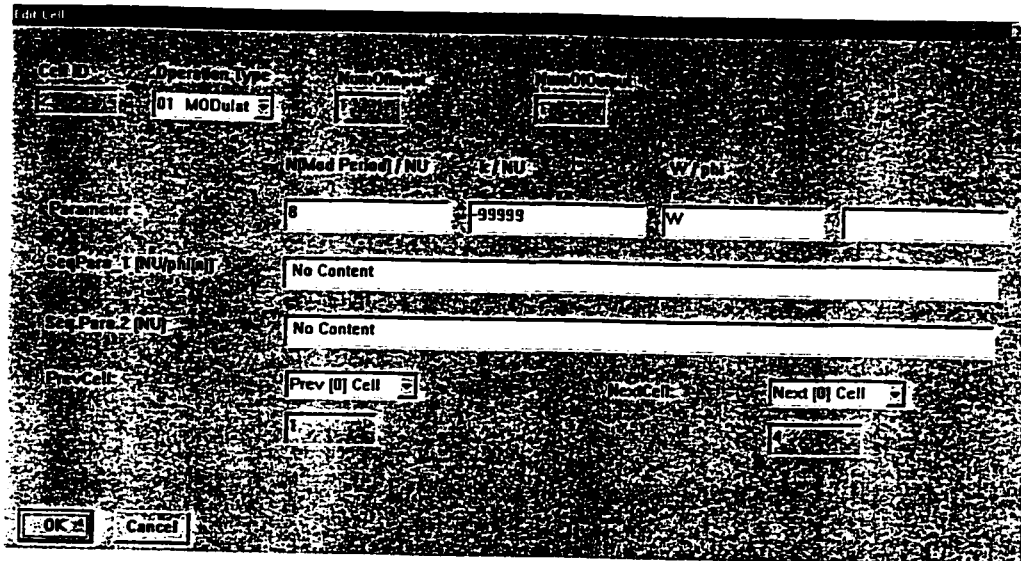


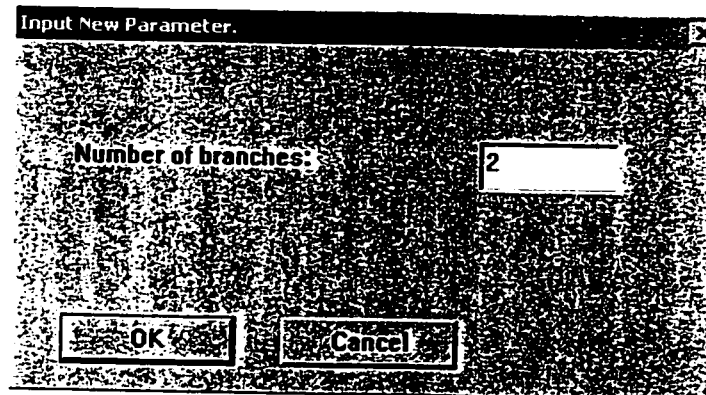
Figure 4-10 MSFG Transformation Tools Bar (Figure Menu)

4.6.2 Dialog Window

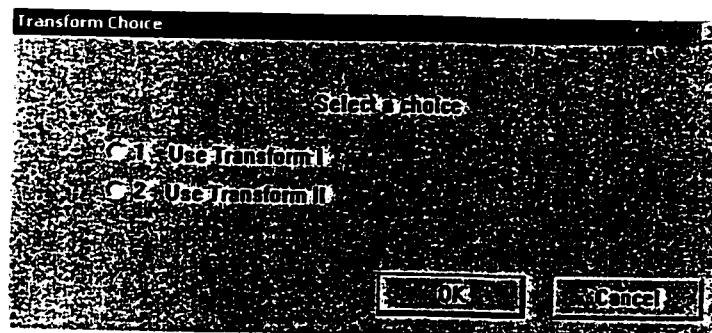
More conditions and parameters are needed for a MSFG transformation. Therefore, the user needs to adjust the process or to select some options when he performs a MSFG transformation. There are seven dialog windows in this system, which have been introduced in section 4.5.2.2. Figure 4-11(a) shows a CEditCELL dialog window, which is used to edit an existing CELL. Figure 4-11(b) shows a CInputSinglePara dialog window. It is used to specify a number that the MSFG transformation needed (in this figure, it is used to specify the number of signal branches). Figure 4-11(c) shows a choice dialog window, which is used to select the MSFG transformation.



(a) CEditCELL Dialog Window



(b) CInputSinglePara Dialog Window



(c) CChoice dialog window

Figure 4-11 Some Dialog Windows

4.6.3 File Input, Output and File Structure

Because most classes are derived from CObject class, they will inherit many advantages and functions from CObject. We can use these functions directly.

For file input and output, we can use serialization recursion. For saving or loading a MSFG file (.sfg), we just need to call the serialization in the top class and we do not need to care about the bottom class structure. It is shown in Figure 4-12. If we want to save the whole CELL Graph information to a MSFG file (.sfg), we just need to call save or load function in “CELL Graph”, and it will call the “CELL serialization—Data serialization—Complex serialization—Performance serialization” automatically.

The MSFG file structure needs record its version information. According to this version information, this computer-aided design system can read or write all the data information correctly. Figure 4-13 displays the file structure of version 1.1.

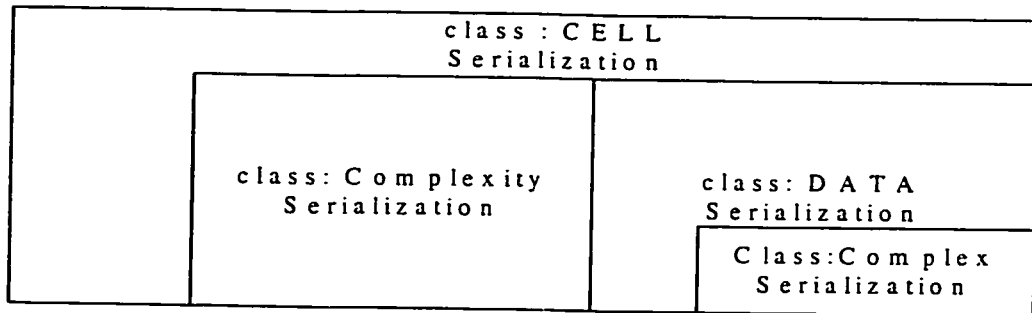


Figure 4-12 Serialization

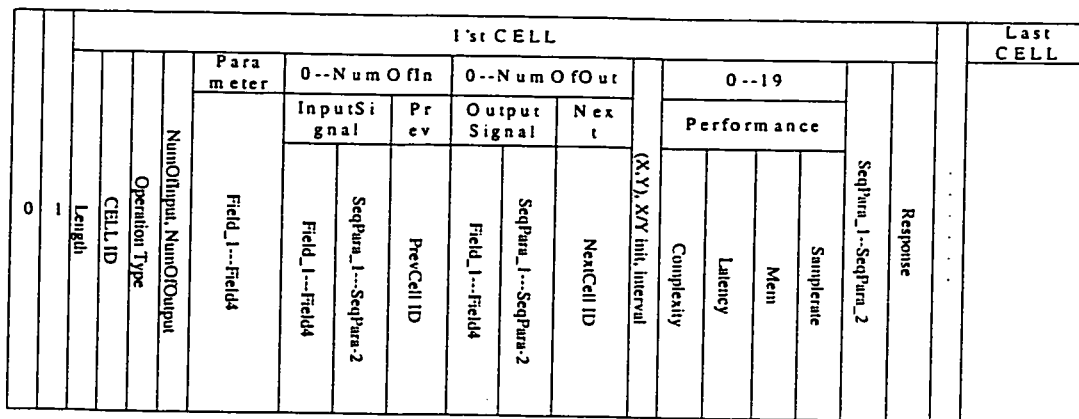


Figure 4-13 SFG file structure

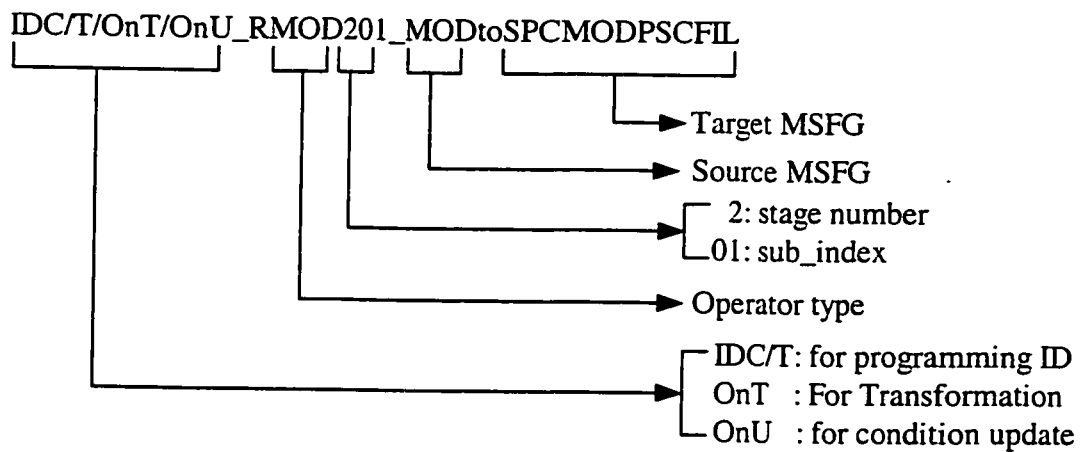
Additionally, for any CELL, we need to save its CELL length, CELL ID, operation type and corresponding parameters, number of input and corresponding signal, number of output and corresponding signal, (X,Y) coordinate, complexity information, sequential parameters and responses.

4.7 MSFG Transformations

To perform a MSFG transformation, choose a CELL that you want to transform and then you can choose one of the following three methods:

1. Select the corresponding MSFG transformation submenu;
2. Select "Display Transformation Tools Bar" and then select corresponding MSFG transformation in the tools bar;
3. Click right button of mouse and then select the corresponding MSFG transformation in the tools bar.

As described in section 3.5 to 3.8, MSFG transformations are classified into four categories (a total of about 80 MSFG transform pairs) in this computer aided design system. For convenience, we use the following symbols to represent a MSFG transformation:



4.8 Simulation and Verification of MSFG Transformation

For verify the correctness of the transformed MSFG, we implemented a simulation tool in this system to simulate and verify the MSFG transformations.

The verification submenu is shown in Figure 4-14. It includes the following main functions:

1. **Set Signal Generator:** The system can generate impulse $\delta(n)$, sine wave $\sin(n\omega)$, complex tone $e^{jn\omega}$ and random sequence (real or complex). It can take any user-defined signals for simulation and verification of MSFG transformation. Its default length is 1024 samples and it can be changed.
2. **Simulation:** Apply an input signal from the Signal Generator to the Source CELL (SC), and run simulation, then we can get the simulation result at any specified CELL.
3. **Signal Comparison:** We can compare the simulation result before and after MSFG transformation or any two signals in a MSFG. It will display the comparison result (difference) in tabular and graphic forms.
4. **Table and Chart:** It will plot a chart and draw a table to display the detailed comparison result in the CChartdlg dialog window, as shown in Figure 4-15.

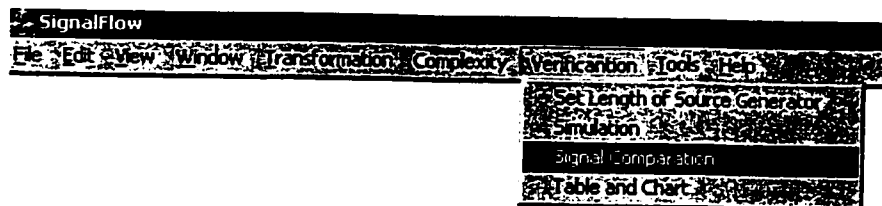


Figure 4-14 Simulation and Verification Menu

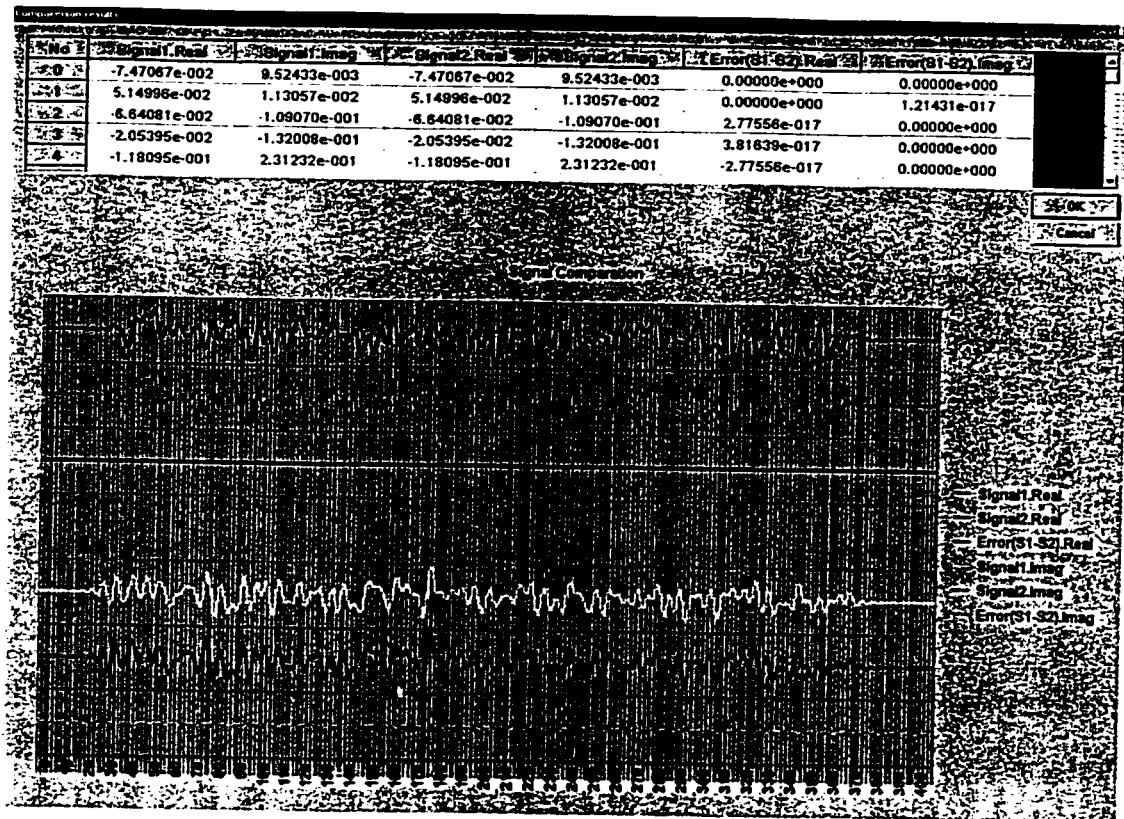


Figure 4-15 Simulation and Verification result figure

4.9 Complexity of MSFG

For a multirate system represented in MSFG, we can apply the tool provided by the system to calculate the complexity of the system in terms of component counts (number of multipliers, adders, memories, etc.). The complexity tool also calculates the computational complexity of the system, which is weighted (by the sample rate) of the arithmetic operations and hence can be used as a measure of merit or efficiency of the MSFG.

Figure 4-16 shows an example of the report result of the complexity tool.

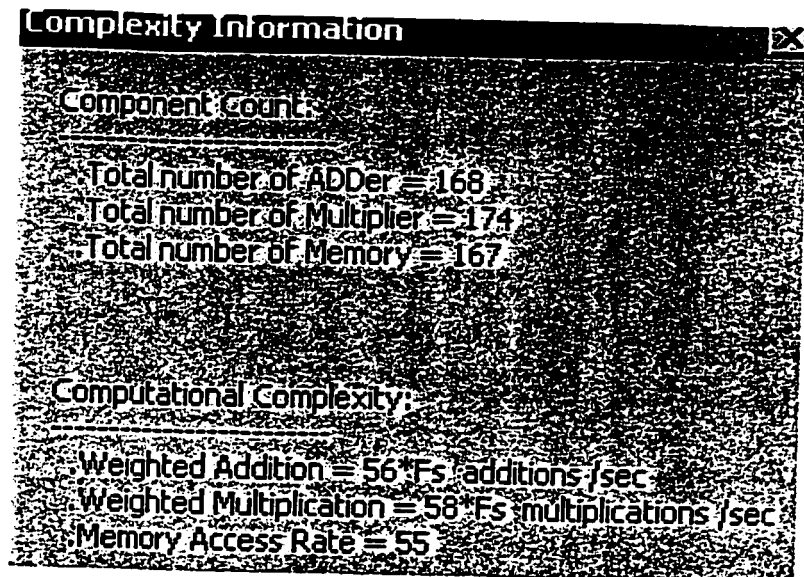


Figure 4-16 Complexity information of MSFG

Application Examples

In this chapter, the usefulness and the effectiveness of this computer aided design system will be demonstrated through some design examples. In these examples, step-by-step MSFG transformations will be provided, which would otherwise be very difficult to obtain without the aid of this system. In addition, we have also successfully applied this computer aided design system to derive a novel programmable Digital Down-Converter (DDC) structure [10].

5.1 Five-channel Frequency Demultiplexer (DEMUX)

The task of this example is to channelize MF-TDMA traffic consisting of five equally spaced carriers. Each of the MF-TDMA channels has the symbol rate of 1 M baud. The roll-off factor is assumed 0.5. The spectrum of the sampled input signal is illustrated in Figure 5-1(a). It is obviously an odd-stacking real FDM.

To be able to use real channel filters in the lowpass DEMUX model of Figure 2-1 and to avoid using GDFT (Generalized Discrete Fourier Transform), the sampled input signal needs to be in even channel stacking. The odd-to-even channel stacking conversion can be achieved by using a trivial $\pi/2$ -frequency-shift, which requires no arithmetic operations. Figure 5-1(b) shows the spectrum after the conversion. The key characteristic of the channel filter is illustrated in Figure 5-1 (c).

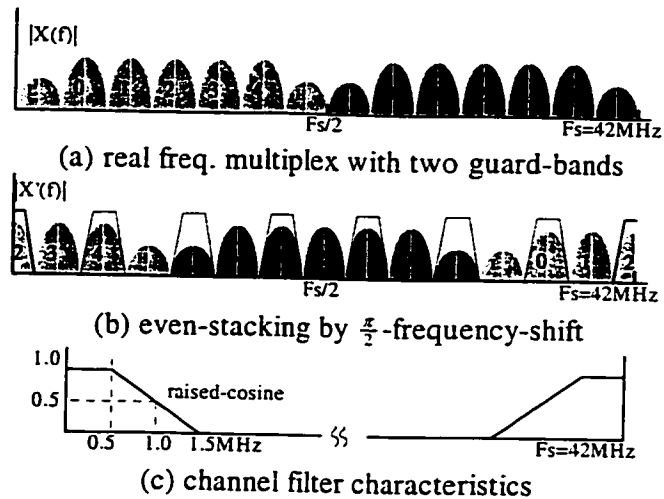


Figure 5-1 Five-channel MF-TDMA signal and the demultiplexing channel filter

The DEMUX function can be described by the MSFG shown in Figure 5-2. A direct implementation of this structure would require five (not considering the guard-channels) identical lowpass channel filters and a bank of frequency shifts. In the multirate filter bank theory, an uniform modulated filter bank like this can be efficiently realized with only one shared polyphase decomposed channel filter plus a DFT (hence polyphase-DFT), a significant saving in computation over the direct structure.

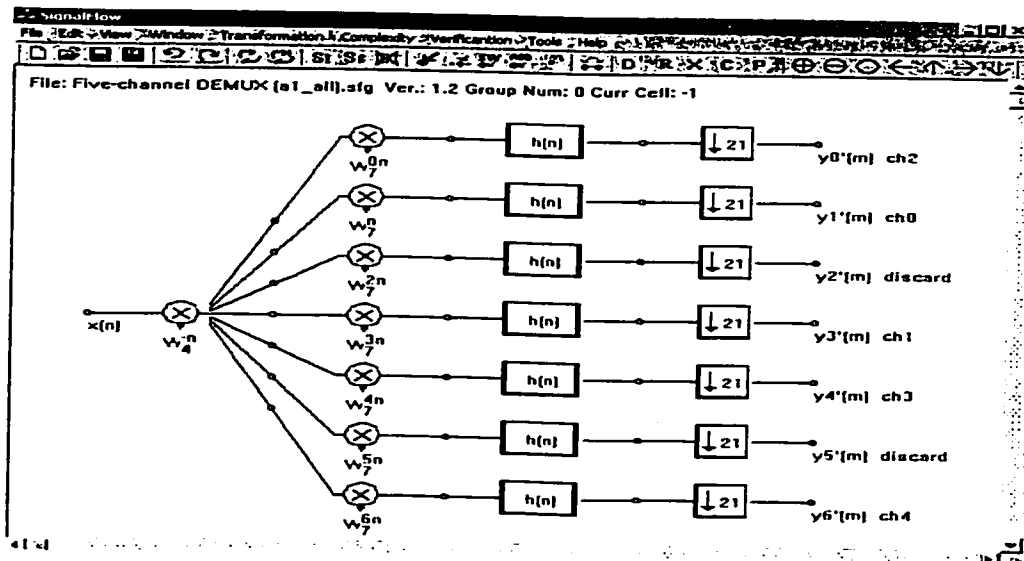


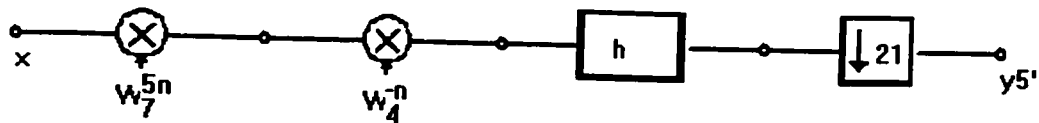
Figure 5-2 DEMUX function (lowpass model)

5.1.1 MSFG transformation

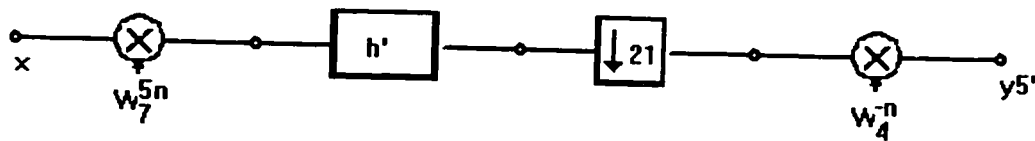
To derive the optimized structure for the filter bank of Figure 5-2, we redraw the MSFG for the 6th channel in Figure 5-3(a). The basic idea of the MSFG simplification is that we should move arithmetic operations, multiplications in particular, to places where the sampling rate is as low as possible in order to reduce the computation rate.

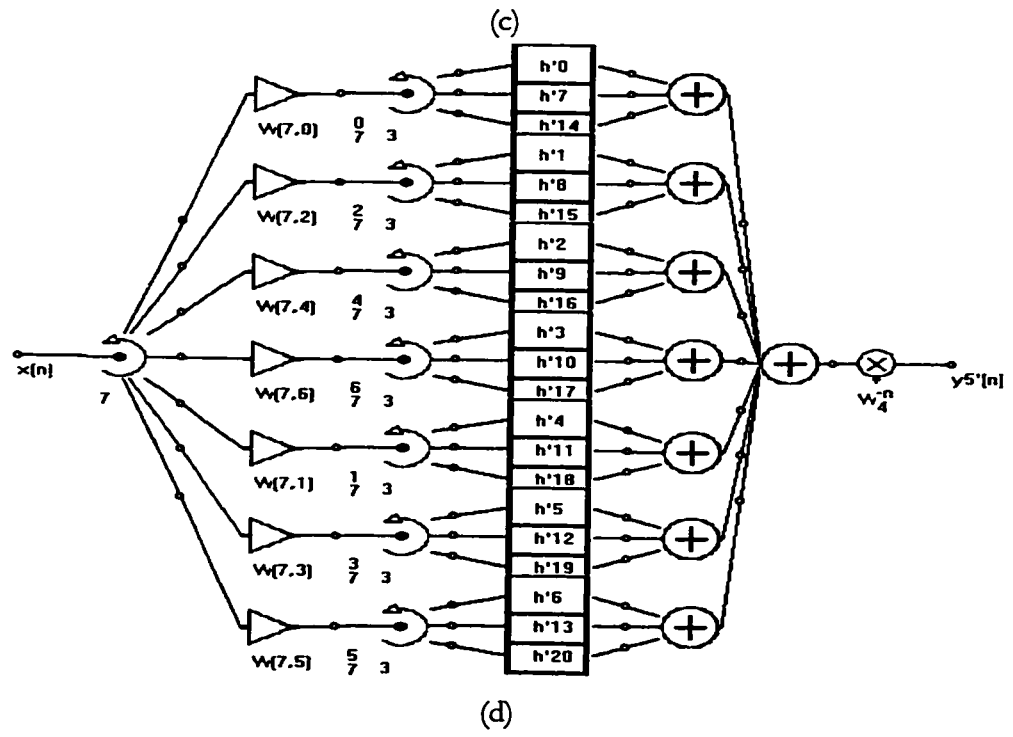
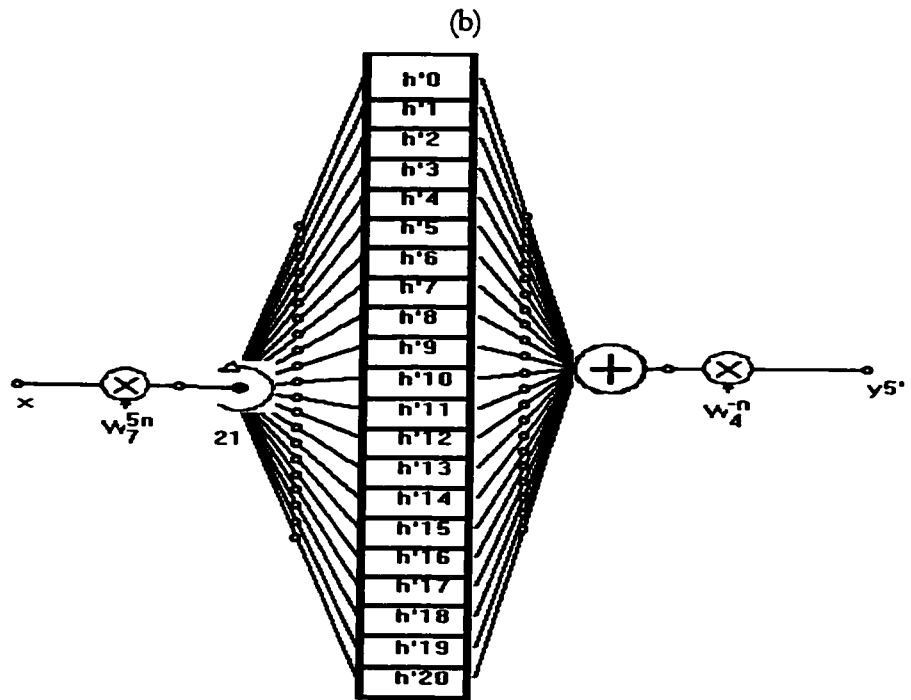
To this end, we firstly convert the real filter $h(n)$ into a trivial complex one $\tilde{h}(n) = W_4^n h(n) = (j)^n h(n)$ using the complex filtering identity (Figure 3-5(b)) as shown in Figure 5-3 (b) and then polyphase decompose the complex decimation filter leading to Figure 5-3 (c). Use the type 2 MPDT of Figure 3-11(b) to the first frequency shift and decompose the 1-to-21 SPC into a tree consisting of a 1-to-7 SPC and seven 1-to-3 SPCs (referring to Figure 3-13(a)). The 7-to-1 PSC of the decomposed frequency shift will cancel with the 1-to-7 SPC of the SPC tree (Figure 3-13(d)) resulting in Figure 5-3 (d). The complex multipliers in the figure can be moved after the sub-filters as shown in Figure 5-3 (e). Reorganize the structure we get Figure 5-3 (f).

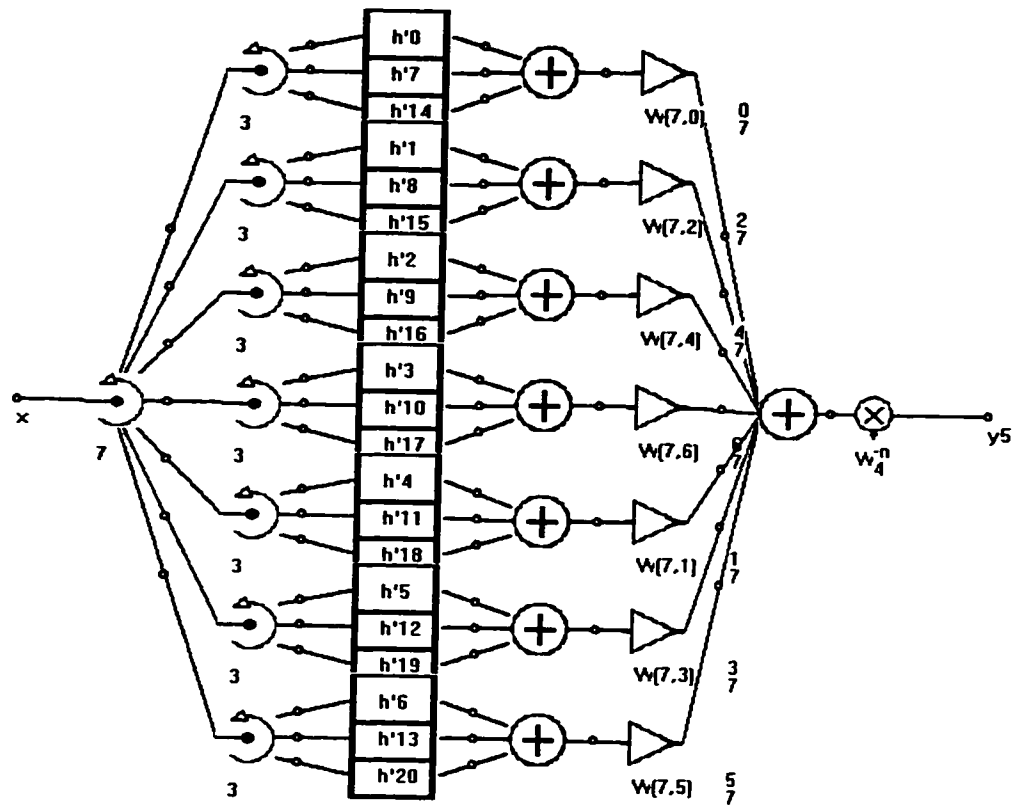
The structure of Figure 5-3 (f) shows that the polyphase network is independent of the channel index k . It can be therefore shared by all the channels. Since the frequency-shift network in the figure simply performs the k -th component of a 7-point DFT, the whole DEMUX network can be constructed by a polyphase filter bank followed by a 7-point DFT as shown in Figure 5-4 and it's relative DSP block diagram is shown in Figure 5-5.



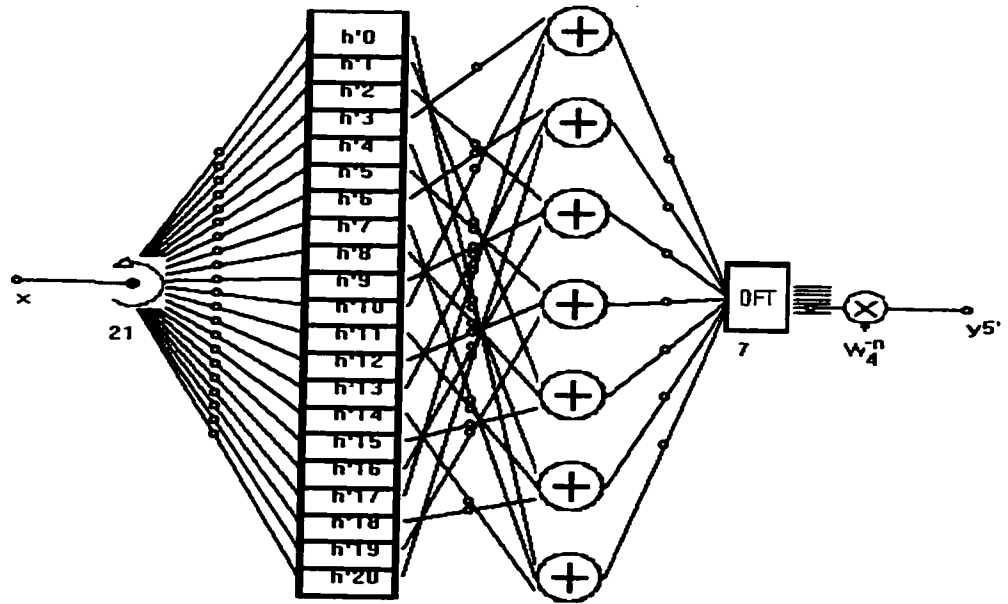
(a) Original SFG







(e)



(f)

Figure 5-3 Step-by-step simplification for demultiplexing the k -th channel

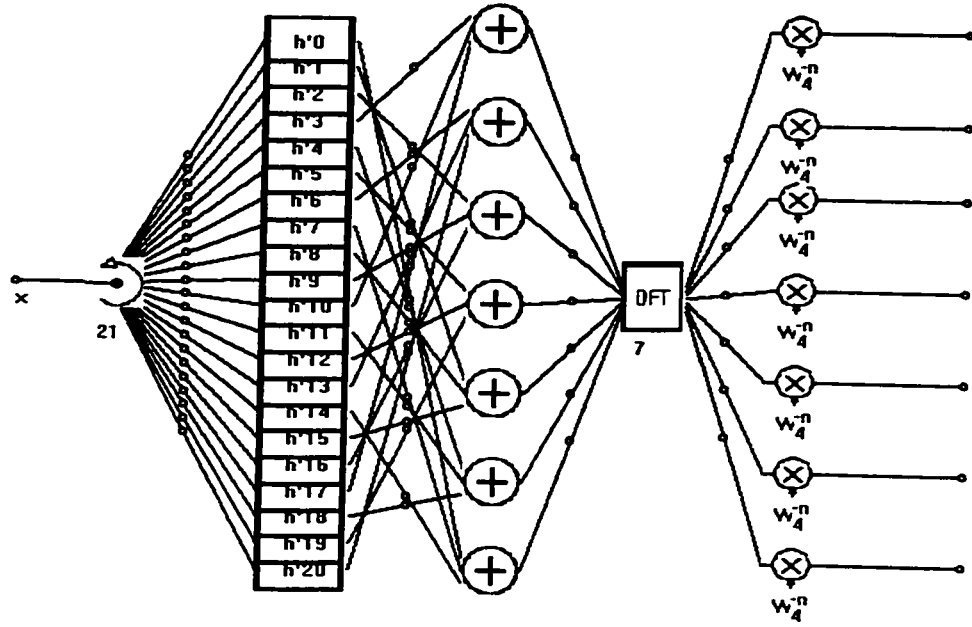


Figure 5-4 Polyphase DFT structure for the 5-channel DEMUX

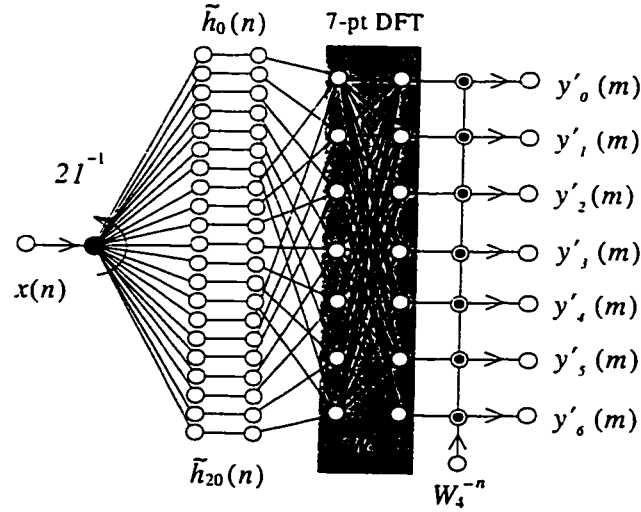
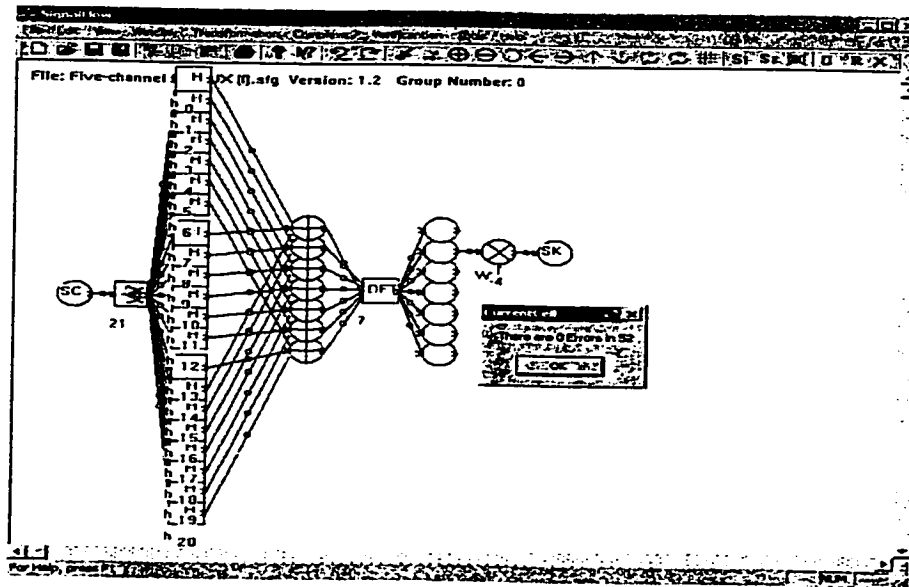


Figure 5-5 DSP block diagram

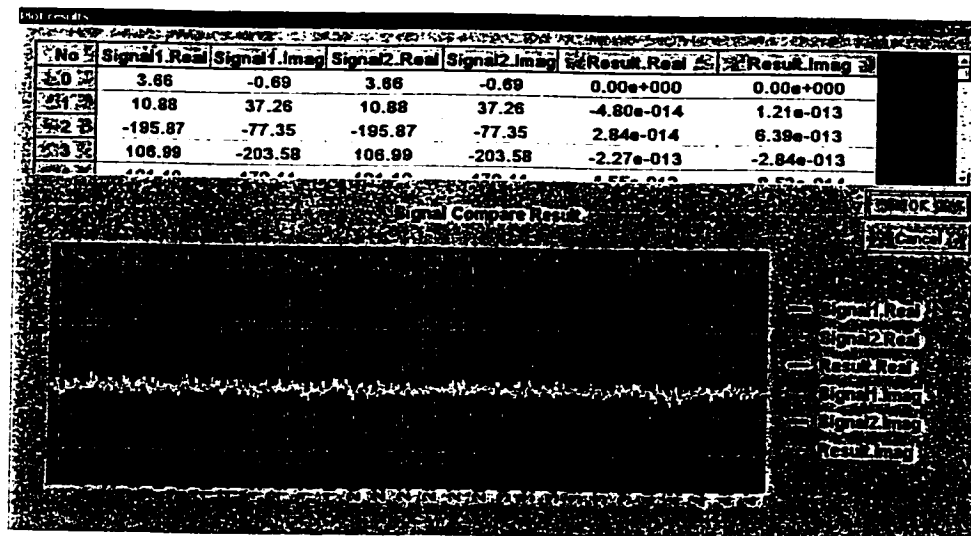
Considering that the input signal is real and that the complex filter $\tilde{h}(n) = (j)^n h(n)$ has coefficients either pure real or pure imaginary, the actual computation load of the complex polyphase filter bank is the same as that of the real prototype filter $h(n)$. Furthermore, all the computations are carried out at the low sampling rate of 2 MHz and the $\pi/2$ -frequency-shift bank requires no arithmetic effort. Hence, we conclude that the derived polyphase-DFT structure is computationally optimal.

5.1.2 Simulation and Verification

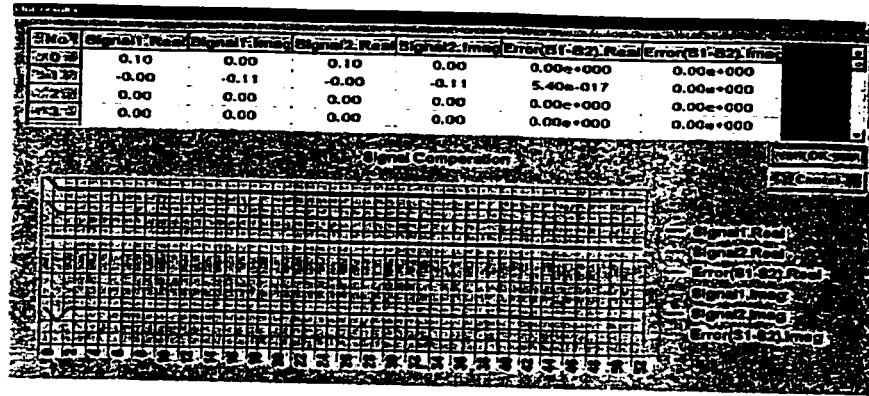
The verification results of DEMUX are shown in Figure 5-6 (a), (b) and (c).



(a) Comparison result



(b) Comparison with random signal source



(c) Comparison with delta function signal source ($\delta(n)$)

Figure 5-6 Simulation and Verification of DEMUX

5.2 The optimal complex BSF structure for binary tree DEMUX

In this example, we derive a novel complex band-splitting filter structure, which is believed optimal in the sense that it has the minimum computation rate amongst the known existing BSF structures using MSFG transforms. The direct complex BSF structure is shown in Figure 5-7 (a). Applying the transform of Figure 3-5 (a) to the complex filters and the identity of Figure 3-6(a) to the figure, the direct structure can be transformed into Figure 5-7 (b) consisting of a BSF core CELL and trivial frequency shifts $((j)^n$ or $(-j)^n$) at both input and output of the BSF. These frequency shifts will be cancelled at intermediate stages of the tree.

5.2.1 MSFG transformation

Now let us derive the optimal complex BSF via MSFG transforms of the core CELL. The core CELL has the MSFG shown in Figure 5-8(a). Using type 2 MPDT (Figure 3-10(b)) and polyphase decomposition of decimation filter (Figure 3-8 (b)) transforms the direct structure of Figure 5-8(a) into that of Figure 5-8(b). Decomposing the 1-to-4 SPCs and 4-to-1 PSCs into 1-to-2 SPCs and 2-to-1 PSCs (Figure 3-13) and applying the PSC-SPC cascade identity (Figure 3-13(c)), Figure 5-8(b) can be transformed to Figure 5-8(c) in which the lowpass and the highpass branches share common SPCs and π -frequency shifts.

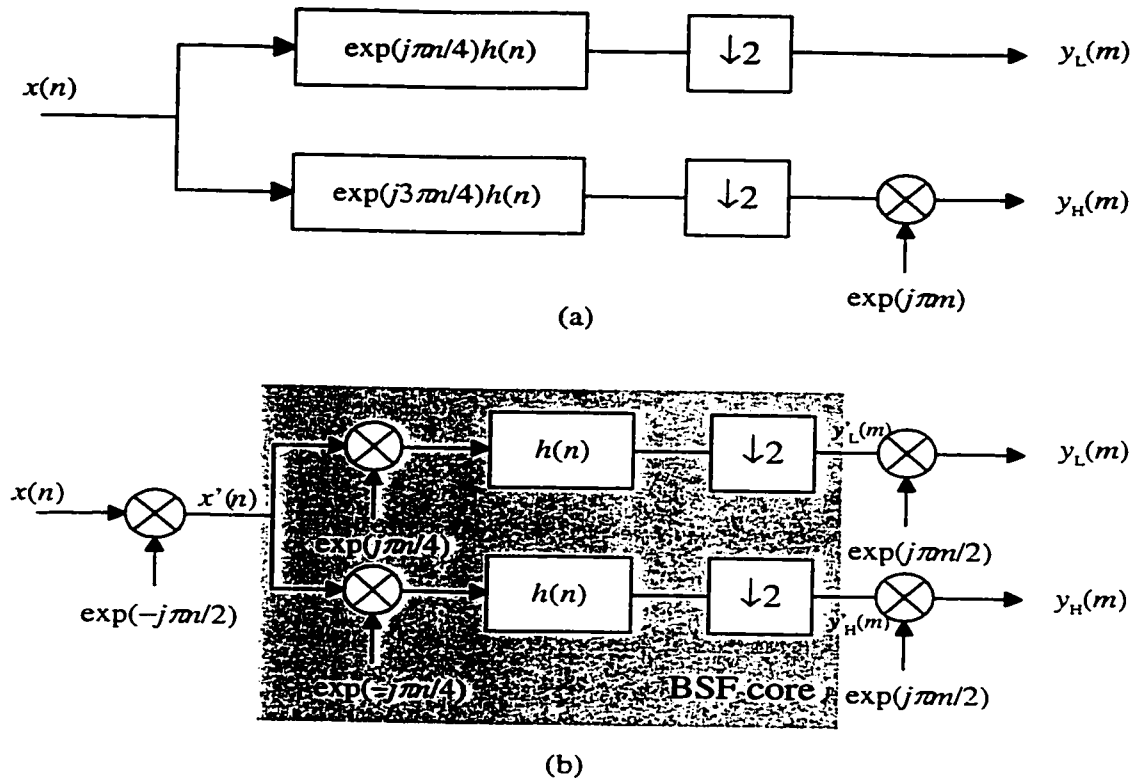


Figure 5-7 Direct complex BSF structure

Till now, the computational complexity remains the same as the original BSF model: four real full lengths FIR filters (remember that the signal paths are complex) operating at the input (high) sampling rate. To reduce the complexity we need to share the filters as much as possible and to move the filtering to the lower sampling side, that is, immediately after the front SPCs. For this purpose, we firstly polyphase decompose the interpolation filters in Figure 5-8(c) and then move the τ -shifts before the 1-to-2 SPCs with the identity of Figure 3-14(e). That gives Figure 5-8(d). Reorganizing the middle part of the figure by applying the superposition theorem for linear networks and noticing that the combinations of the rate changing operators in the reorganized network can be equalized to pure delays or a τ -shifts according to identities of Figure 3-13(a) and Figure 3-11, we have Figure 5-8(e).

Basically, the structure in Figure 5-8(e) is already computationally optimal for an arbitrary real prototype filter $h(n)$ because that the computational complexity of the four real sub-filters accounts for that of a single real prototype filter and that the computations are in the low rate side. If $h(n)$ is linear phase, and furthermore, lowpass half-band type,

which is usually the case for binary tree, structures, then the complexity of Figure 5-8(e) can be further reduced. For a half-band lowpass filter with length $N=4n-1$ where n is the number of distinct non-zero coefficients except the centre tap which is 0.5, let us assume that n is even (which gives $N=7, 15, 23, \dots$). Then the polyphase filters

$$H_{10}(z)=0, \quad (5.1)$$

$$H_{11}(z)=0.5z^{-(n/2-1)} \quad (5.2)$$

$H_{00}(z)$ and $H_{01}(z)$ are non-symmetric and are related by

$$H_{00}(z)=z^{-n}H_{01}(z^{-1}) \quad (5.3)$$

(or, $h_{00}(i)=h_{01}(n-1-i)$, $i=0, 1, \dots, n-1$). Hence, the signal paths corresponding to $H_{10}(z)$ disappear. If we define

$$P(z)=[H_{01}(z)+H_{00}(z)]/2 \quad (5.4)$$

$$Q(z)=[H_{01}(z)-H_{00}(z)]/2, \quad (5.5)$$

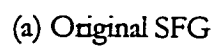
such that

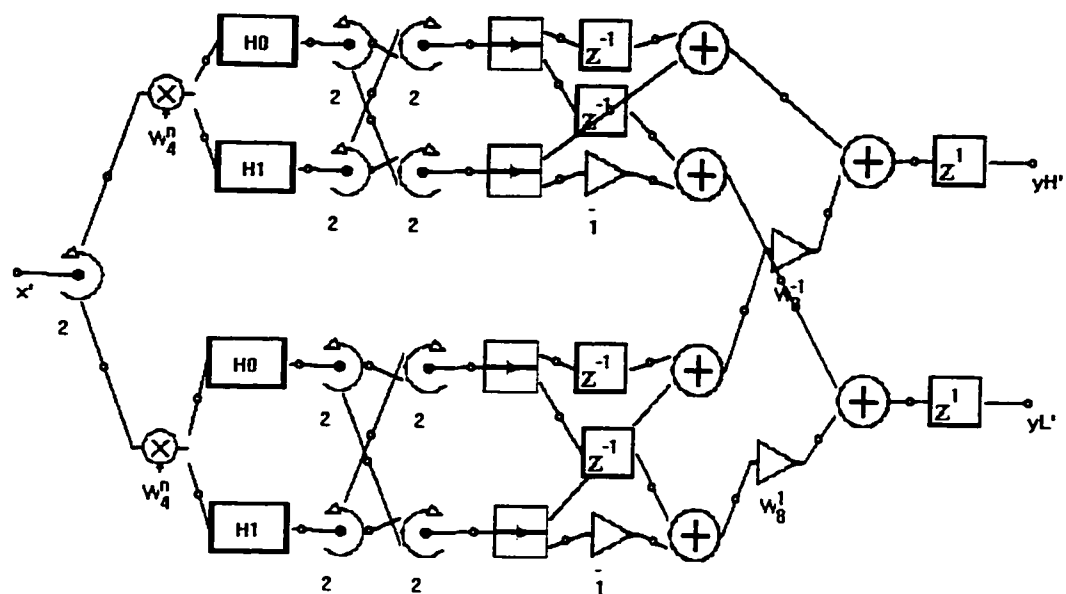
$$H_{01}(z)=P(z)+Q(z) \quad (5.6)$$

$$H_{00}(z)=P(z)-Q(z) \quad (5.7)$$

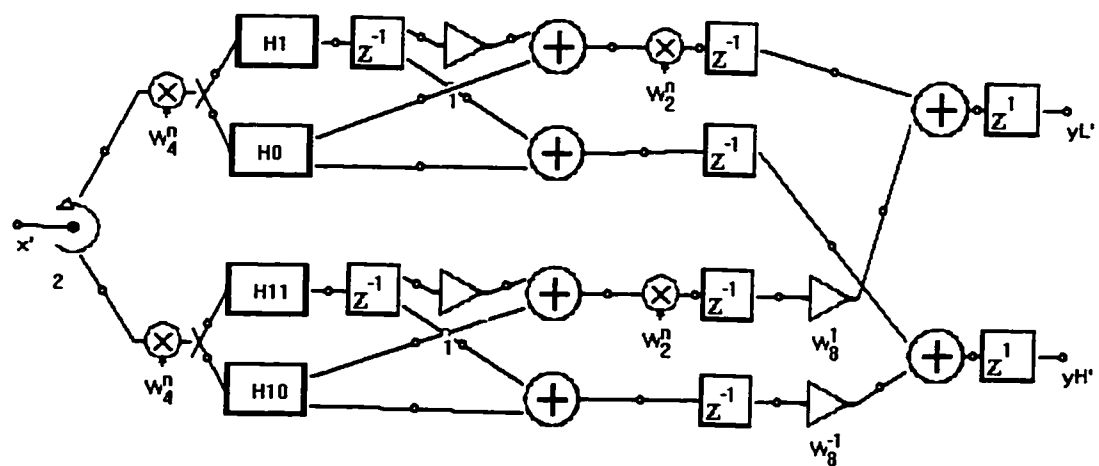
Then, according to Eq. (5.3), $P(z)$ and $Q(z)$ are symmetric and anti-symmetric respectively. Obviously, the number of multiplications can be halved in implementation of Eq.(5.6) and (5.7) using a symmetrical transversal FIR structure for $P(z)$ and $Q(z)$ compared to direct implementations of $H_{01}(z)$ and $H_{00}(z)$. Thus, the structure of Figure 5-8(e) can be transformed into that shown in Figure 5-8(f).

Finally, by moving the complex multiplier $W_8^{-1} = e^{j\frac{\pi}{4}} = \frac{1}{\sqrt{2}}(1+j)$ in Figure 5-8(f) backwards and combining the centre tap (0.5) with the above real constant factor, we have the optimal BSF structure shown in Figure 5-8(g). It's relative DSP block diagram is shown in Figure 5-9.

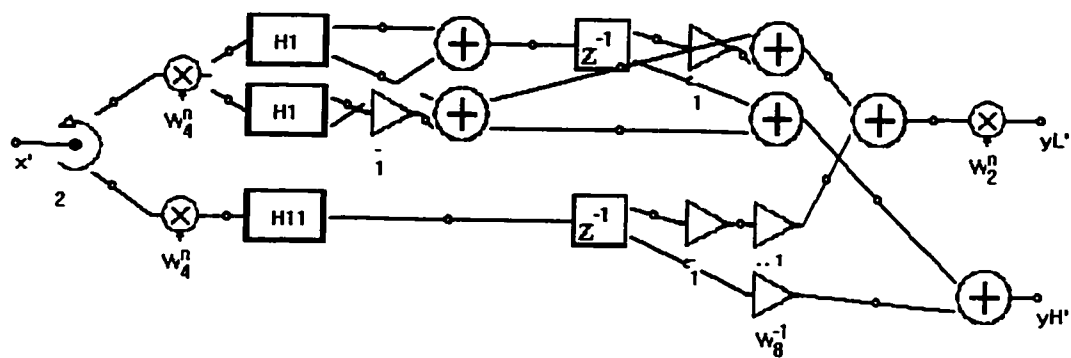




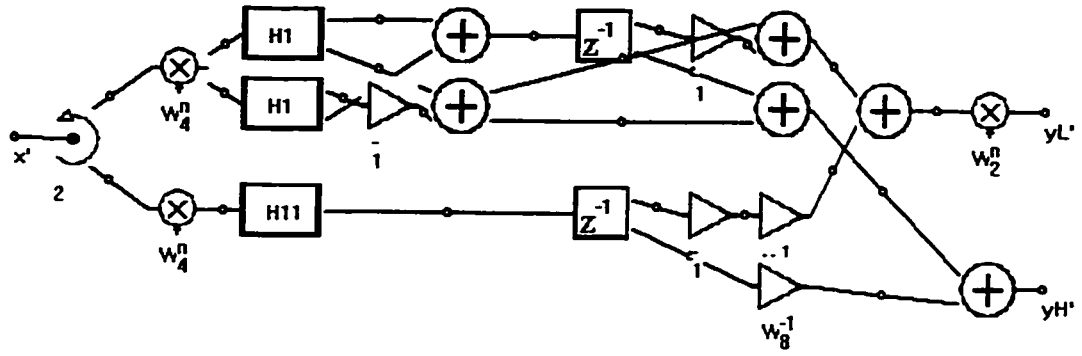
(d)



(e)



(f)



(g)

Figure 5-8 Derivation for the optimal complex BSF

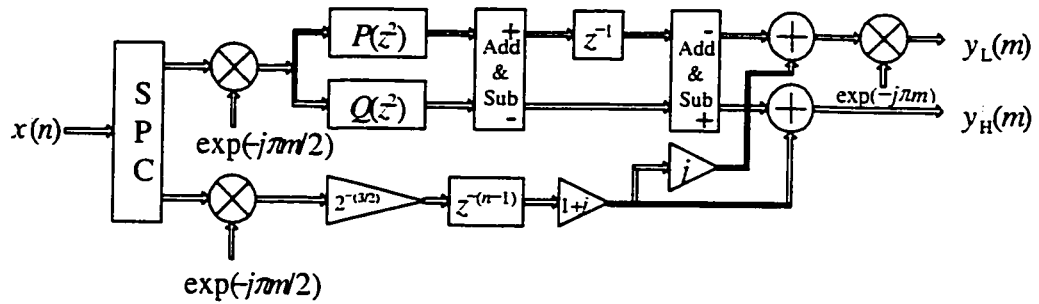
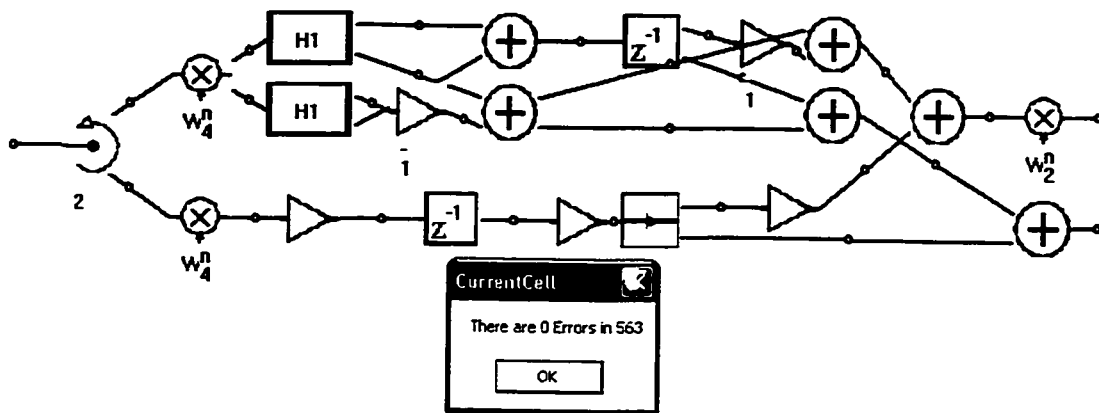


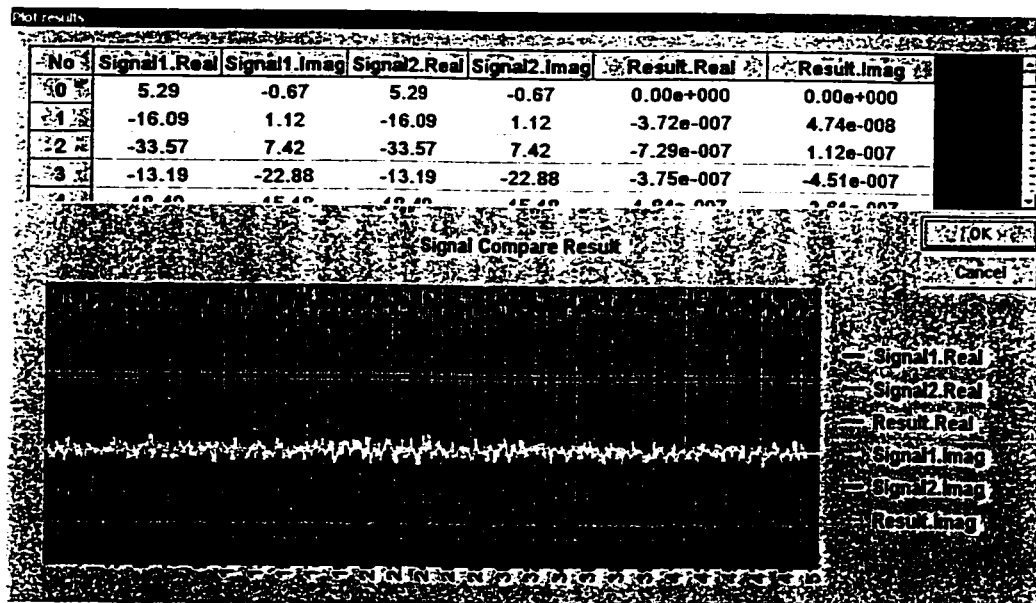
Figure 5-9 DSP block diagram

5.2.2 Simulation and Verification

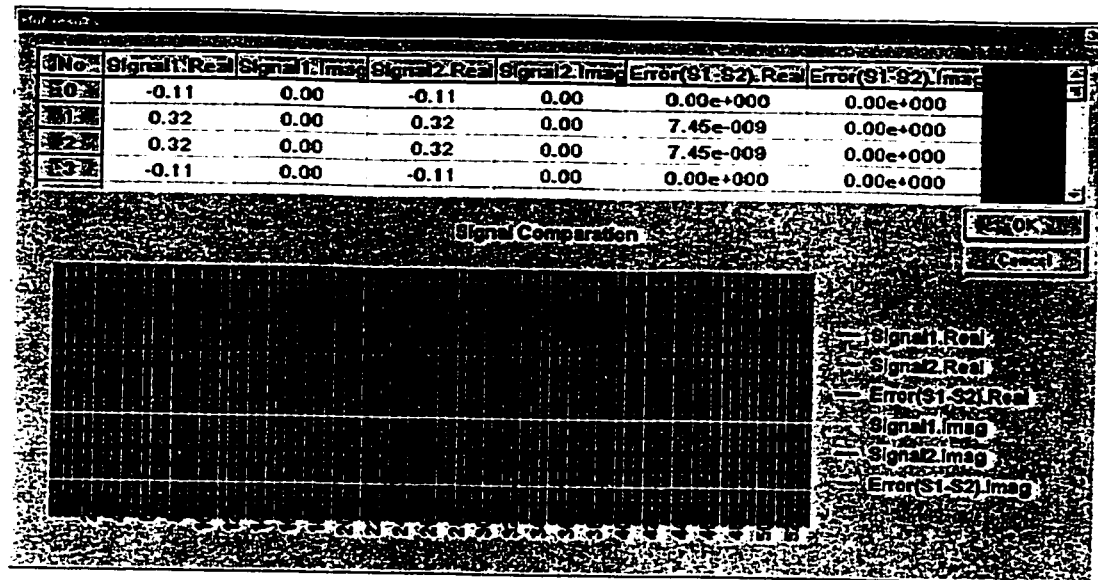
Assume $h(n)$ is a half-band lowpass filter with length $N=7$ and $h(n)=[-0.1061, 0, 0.3183, 0.5, 0.3183, 0, -0.1061]$. The verification results are shown in Figure 5-10(a), (b) and (c).



(a) Comparison result



(b) Comparison with random signal source



(c) Comparison with delta function signal source ($\delta(n)$)

Figure 5-10 Simulation and Verification result

Conclusion and future work

6.1 Conclusions and discussions

In this thesis, we have presented a computer aided design system for structural simplification and optimization of multirate signal processing systems. The usefulness and effectiveness of the system have been demonstrated through design examples. Following conclusions can be drawn based on our study on MSFG and the work presented in this thesis.

- MSFG representation and transformation provide a systematic approach to multirate system optimization.
- The computer aided design system presented in this thesis has been proved effective and easy to use for multirate DSP structure design and optimization.
- The proposed MSFG approach and the computer aided design system are useful and effective and they can find applications in both academia and industry.

The computer-aided design system that has been presented in this thesis is the first version of the system. It is therefore far from perfect and mature. Just like any other product before its maturity, this system has inevitably shortcomings and limitations. For instance, the present version of the system is lack of the intelligence which is required to guide the simplification/optimization process. Consequently, it can not perform MSFG optimization in a fully automatic manner. Secondly, the present version still has difficulties to handle feed back and recursive structures because there are few MSFG

transformations available that can be applied to this type of structures. Another limitation of the system is that it does not support hierarchical modelling and design (i.e., MSFGs represented by the system are all flattened). This may cause problems for complex designs where a large number of MSFG operators are involved.

6.2 Future Work

Based on the discussion given above, the following work is suggested to improve the quality and to overcome the limitations of the computer aided design system reported in this thesis in future.

- To make the computer aided system a truly intelligent expert system that can search for and derive the optimal multirate DSP structure automatically, Artificial Intelligence (AI) technique [30] will be incorporated into the system to in future.
- The quality of the Graphic User Interface (GUI) of the present system is not quite up to the standard that most commercial products have. More effort is needed to improve the quality of the GUI.
- Further study and investigation on MSFG properties and transformations for feedback and recursive structures.
- Link the complexity estimate (number of multipliers, adders, memories) provided in the Complexity Report Tool to a VLSI technology (e.g., ASIC or FPGA) so that the Complexity Report Tool can translate the estimated component counts into the gate count (or number of slices) and report the actual hardware complexity directly.
- Modify the computer aided design system to support hierarchical design and to make the MSFG transformation library expandable by allowing user to add user-defined MSFG transforms.

References

1. Leendert Ammeraal, "Algorithms and Data Structures in C++", 1996.
2. G. Booch, "Object-Oriented Analysis and Design with Applications", Second Edition, Benjamin Cummings, 1994.
3. R. E. Crochiere and L. R. Rabiner, "Multi-rate Digital Signal Processing", Prentice-Hall, Inc., Englewood Cliffs, 1983.
4. N. J. Fliege, "Multirate Digital Signal Processing: Multirate Systems, Filter Banks, Wavelets", John Wiley & Sons, 1994.
5. S. L. Freeny, R. B. Kiebartz, K. V. Mina, and S. K. Tewksbury, "Some Applications of Digital Signal Processing in Telecommunications," in Applications of Digital Signal Processing (A. V. Oppenheim, Ed.). Englewood Cliffs, N.J.: Prentice-Hall, 1978, pp. 1-28.
6. G. Gonnet and R. Baeza-Yates, "Handbook of Algorithms and Data Structures -- in Pascal and C," Addison-Wesley, 1991.
7. Eugène Kain, "The MFC answer book: Solutions for effective visual C++ applications," Addison-Wesley, 1998.
8. S. Kato, et al., "Onboard digital signal processing technologies for present and future TDMA and SCPC systems", IEEE Journal on Selected Area in Communications, Vol. SAC5, No. 4, May 1987, pp. 685-700.
9. R. Lagadec, D. Pelloni, and D. Weiss, "A 2-Channel, 16-Bit Sampling Frequency Converter for Professional Digital Audio," Proc. Of the 1982 IEEE Int. Conf. On Acoust. Speech Signal Process, Paris, May 1982, pp. 93-96.
10. Zifeng Li, Qing Ma, Ronggang Qi, "Design of a Programmable Digital Down-Converter Structure", CCECE2003-CCGEI2003, Montreal, May 2003.
11. Qing Ma, Zifeng Li, Ronggang Qi, "A Computer Aided Design System for Transformation and Optimization of Multirate DSP Systems", International Signal Processing Conference '03, April 1-3, 2003, Dallas, Texas.
12. J.H. McClellan and R. J. Purdy, "Applications of Digital Signal Processing to Radar," in Applications of Digital Signal Processing (A. V. Oppenheim, Ed.). Englewood Cliffs, N.J.: Prentice-Hall, 1978, pp. 239-330.

13. Larty Myhoff, "C++:an introduction to data structures," Upper Saddle River, N.J.: Prentice Hall, 1999.
14. A.V. Oppenheim and R.W. Schafer, "Digital Signal Processing", Prentice-Hall, Inc., 1975.
15. A.V. Oppenheim, R.W. Shafer and Prentice Hall, "Discrete-Time Signal Processing," 1989.
16. R. G. Pridhman and R. A. Mucci, "Digital Interpolation Beam Forming for Lowpass and Bandpass Signals," Proc. IEEE Vol. 67, June 1979, pp. 904-919.
17. R. Qi, "Multicarrier Demultiplexing and VLSI Implementation for Satellite Communications Systems", Ph. D. thesis, university of Surrey, England, 1996.
18. R. Qi, "Low Complexity and Low Power Consumption Design for OBP Multicarrier DEMUX VLSI", the 16th International Communications Satellite Systems Conference, Washington, DC, USA, February 25-29, 1996.
19. R. Qi, "Optimization of Complex Binary Tree Filter Bank Structure Using Multirate Signal Flow Graph Transformation," the IEE International Conference on Communication Systems, Singapore, November 25-29, 1996.
20. R. Qi, "A Gate Array Design of a Multi-channel Tree Filter Bank Demultiplexer," the Third International Workshop on Digital Signal Processing Techniques Applied to Space Communications, ESTEC, Noorwijk, The Netherlands, Sept. 1992.
21. Bindu Rama Rao, "C++ and the OOP Paradigm," New York: McGraw-Hill, 1993.
22. A. Reilly, G. Frazer, B. Boashash, "Analytic signal generation - Tips and traps," IEEE Trans. on Signal Processing, 42, No. 11,1994, pp3241-3245.
23. L.P.A. Robichaud, et al., "Signal Flow Graphs and Applications", Prentice-Hall International, 1962.
24. Jeff Salvage, "C++ Coach: Essentials for Introductory Programming", 2001.
25. Herbert Schildt, "MFC programming from the ground up," 2nd ed, Osborne/McGraw-Hill, 1998.
26. T. Tsuda, S. Morita and Y.Fujii, "Digital TDM-FDM translator with multistage structure," IEEE Trans. Communications, Vol. 26, No. 5, May 1978, pp.734-741.

27. P.P. Vaidyanathan, "Multirate Systems and filter Banks," Prentice-Hall International, 1992.
28. P.P Vaidyanathan, "Quadrature mirror filter banks, M-band extensions and perfect-reconstruction techniques", IEEE ASSP Magazine, July 1987, pp.4-20.
29. Paul S. Wang, "Standard C++ with object-oriented programming," 2nd ed., Pacific Grove, CA: Brooks/Cole, 2001.
30. Patrick H. Winston, "Artificial Intelligence", 3rd ed., Addison-Wesley Publishing Company, 1992.
31. Michael J. Young, "Mastering Microsoft Visual C++ Programming".
32. Edward Yourdon, "Object-oriented systems design: an integrated approach," Prentice-Hall PTR; 1st edition, 1994.
33. Beck Zaratian, "Microsoft Visual C++ 6.0 programmer's guide", Microsoft Press, c1998.