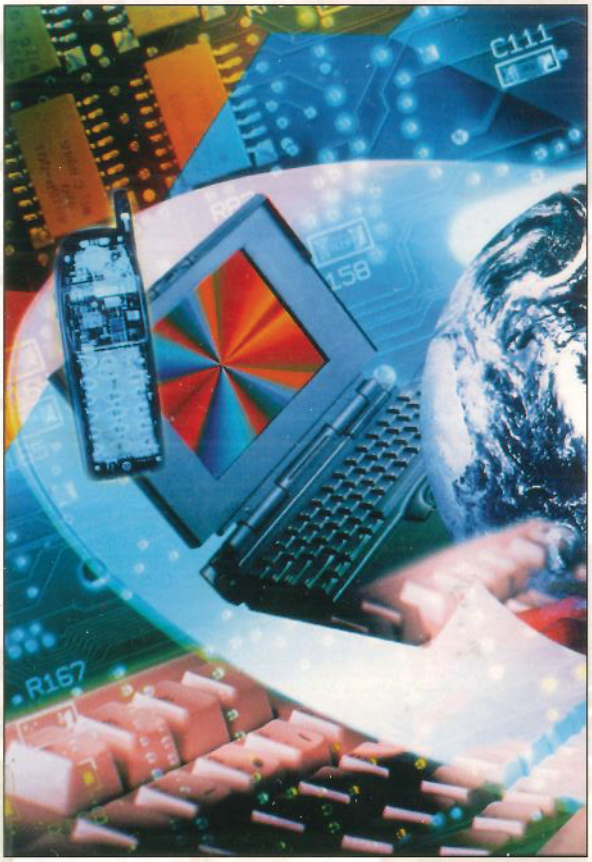
Software RADIO

A Modern Approach to Radio Engineering



The first comprehensive guide to software radio design and implementation

Multirate DSP, RF front-ends, direct digital synthesis of modulated waveforms, A/D and D/A conversion, and more

Enhancing performance through smart antennas and other adaptive array algorithms

Techniques for building more flexible, object-oriented real-time software

Jeffrey H. Reed

Prentice Hall Communications Engineering and Emerging Technologies Series
Theodore S. Rappaport, Series Editor

Reed
Software RADIO
A Modern Approac
to Radio Engineeri
TK
5103.4875
.R44
2002

A cataloging-in-publication data record for this book can be obtained from the Library of Congress.

Production Supervision: Laura Burgess
Publisher: Bernard Goodwin
Editorial Assistant: Michelle Vincente
Marketing Manager: Dan DePasquale
Manufacturing Manager: Alexis Heydt-Long
Cover Design Director: Jerry Votta
Cover Design: Nina Scuderi
Art Director: Gail Cocker-Bogusz
Production Services/Compositor: Lori Hughes

A division of Pearson Education, Inc.
Upper Saddle River, NJ 07458

Prentice Hall books are widely used by corporations
and government agencies for training, marketing, and resale.

The publisher offers discounts on this book when ordered in
bulk quantities. For more information, contact:

Corporate Sales Department
Prentice Hall PTR
One Lake Street
Upper Saddle River, New Jersey 07458

Phone: 800-382-3419
Fax: 201-236-7141
Email: corpsales@prenhall.com

Printed in the United States of America
10 9 8 7 6 5 4 3 2 1

ISBN 0-13-081158-0

Pearson Education LTD.
Pearson Education Australia PTY, Limited
Pearson Education Singapore, Pte. Ltd.
Pearson Education North Asia Ltd.
Pearson Education Canada, Ltd.
Pearson Educación de Mexico, S.A. de C.V.
Pearson Education—Japan
Pearson Education Malaysia, Pte. Ltd.

Contents

PREFACE xv

ACKNOWLEDGMENTS xix

1 INTRODUCTION TO SOFTWARE RADIO CONCEPTS 1
- 1.1 The Need for Software Radios 1
- 1.2 What Is a Software Radio? 2
- 1.3 Characteristics and Benefits of a Software Radio 3
- 1.4 Design Principles of a Software Radio 6
- 1.5 Questions 8

2 RADIO FREQUENCY IMPLEMENTATION ISSUES 11
- 2.1 The Purpose of the RF Front-End 11
- 2.2 Dynamic Range: The Principal Challenge of Receiver Design 13
- 2.3 RF Receiver Front-End Topologies 14
  - 2.3.1 Characteristics of the Topologies 14
  - 2.3.2 Topologies 15
- 2.4 Enhanced Flexibility of the RF Chain with Software Radios 21
- 2.5 Importance of the Components to Overall Performance 21
  - 2.5.1 Antennas 21
  - 2.5.2 Duplexer and Diplexer 26
  - 2.5.3 RF Filter 26
  - 2.5.4 Low Noise Amplifier 27
  - 2.5.5 Image Reject and IF Filters 27
  - 2.5.6 RF Mixer 27
  - 2.5.7 Local Oscillator 28
  - 2.5.8 Automatic Gain Control 29
  - 2.5.9 Analog to Digital Converter 32
- 2.6 Transmitter Architectures and Their Issues 33
- 2.7 Noise and Distortion in the RF Chain 35
  - 2.7.1 Noise Characterization 35
  - 2.7.2 Distortion Characterization 38
- 2.8 ADC and DAC Distortion 42

2.9 Predistortion 43
2.10 Flexible RF Systems Using Microelectromechanical Systems 49
2.11 Conclusion 52
2.12 Questions 53

3 MULTIRATE SIGNAL PROCESSING 55
3.1 Introduction 55
3.1.1 Cost 55
3.1.2 Flexibility 56
3.1.3 Overview of the Chapter 57
3.2 Sample Rate Conversion Principles 57
3.2.1 Decimation 58
3.2.2 Interpolation 62
3.2.3 Two Multirate Identities 77
3.2.4 Non-Integer-Rate Conversion 78
3.2.5 Sampling Rate Conversion by Stages 79
3.2.6 Cascaded Integrator Comb Filter 87
3.3 Polyphase Filters 96
3.3.1 Polyphase Decimation 96
3.3.2 Polyphase Interpolation 98
3.4 Digital Filter Banks 101
3.4.1 Implementation 103
3.4.2 DFT Filter Banks 103
3.4.3 Transmultiplexers 112
3.5 Timing Recovery in Digital Receivers Using Multirate Digital Filters 114
3.5.1 Timing Recovery in a Classical Analog Receiver 117
3.5.2 Timing Recovery in the Digital Domain Only 117
3.5.3 Early-Late Gate Synchronizer 120
3.5.4 Timing Offset Control Using the Early-Late Gate Principle 120
3.6 Conclusion 124
3.7 Questions 125

4 DIGITAL GENERATION OF SIGNALS 127
4.1 Introduction 127
4.2 Comparison of Direct Digital Synthesis with Analog Signal Synthesis 129
4.3 Approaches to Direct Digital Synthesis 131
4.3.1 Pulse Output Direct Digital Synthesis 131
4.3.2 ROM Look-Up Table Approach 133
4.3.3 Phase Truncation Distortion 133
4.3.4 Analysis of the Output Sequence 139
4.4 Analysis of Spurious Signals 140
4.5 Spurious Components due to Periodic Jitter 143
4.6 Bandpass Signal Generation 144
4.7 Performance of Direct Digital Synthesis Systems 146

4.7.1 Experimental Findings 146
4.7.2 Use of Hybrid Systems 147
4.8 Hybrid DDS-PLL Systems 148
4.9 Applications of Direct Digital Synthesis 148
4.10 Generation of Random Sequences 150
4.10.1 Types of Sequences and Their Properties 150
4.10.2 Randomization with the Wheatley Procedure 153
4.11 ROM Compression Techniques 157
4.11.1 Interpolation Using Taylor's Series Expansion 158
4.11.2 Interpolation Using Trigonometric Identities 160
4.12 Sine-Phase Difference Algorithm Approach 163
4.13 Modified Sine-Phase Difference Approach (Parabolic Approximation) 164
4.14 Conclusion 166
4.15 Questions 167

5 ANALOG TO DIGITAL AND DIGITAL TO ANALOG CONVERSION 169
5.1 Introduction 169
5.2 Parameters of Ideal Data Converters 171
5.2.1 Sampling Process 171
5.2.2 Quantization 184
5.3 Parameters of Practical Data Converters 195
5.3.1 Generic Data Converter Physical Models 195
5.3.2 Practical Transfer Characteristic Considerations 199
5.3.3 Dynamic Range Considerations 202
5.3.4 Practical Timing Issues 208
5.3.5 Analog Bandwidth 217
5.3.6 Power Consumption 217
5.3.7 Impact of Noise and Interference on Dynamic Range Requirements 221
5.4 Techniques to Improve Data Converter Performance 224
5.4.1 Dithering 225
5.4.2 Automatic Gain Control 228
5.5 Common ADC and DAC Architectures 232
5.5.1 Parallel Structures: Flash ADCs, String DACs, and Binary Structures 232
5.5.2 Segmented Structures: Folding and Interpolating ADC and Segmented Ladder DAC 236
5.5.3 Iterative Structures: Subranging/Pipelined/Half-Flash ADC, Successive Approximation ADC 242
5.5.4 Sigma-Delta Structures: ADC and DAC 246
5.6 Conclusion 256
5.7 Questions 259

6 SMART ANTENNAS **263**
6.1 Introduction 263
6.2 Vector Channel Modeling 264
6.2.1 Array Steering Vectors 265
6.2.2 Multipath Channel Models 270
6.2.3 Multi-User Channel Models 273
6.3 Benefits of Smart Antennas 274
6.3.1 Beamforming 275
6.3.2 Space-Time Equalization 276
6.3.3 Diversity 276
6.4 Structures for Beamforming Systems 276
6.4.1 Multiple Fixed-Beam Antenna Array 278
6.4.2 Fully Adaptive Array 278
6.4.3 Relative Benefits and Trade-Offs of Switched Beam and Adaptive Array Systems 282
6.5 Smart Antenna Algorithms 282
6.5.1 Diversity Combining Techniques 283
6.5.2 Adaptation Algorithms Using Training Sequences 289
6.5.3 Blind Algorithms 292
6.6 Diversity and Space-Time Adaptive Signal Processing 300
6.6.1 Algorithms for Receiver STAP 301
6.6.2 Overloaded Array Processing 308
6.7 Algorithms for Transmit STAP 309
6.7.1 Space-Time Pre-Filtering 309
6.7.2 Space-Time Trellis Coding 310
6.7.3 A Simple Transmit Diversity Scheme 314
6.8 Hardware Implementation of Smart Antennas 316
6.8.1 Digital Beamforming Receiver Implementation 317
6.8.2 Digital Beamforming Transmitter Implementation 317
6.8.3 Component Issues 319
6.9 Array Calibration 321
6.9.1 Remote Transmitter Approach 321
6.9.2 Test-Tone Approach 322
6.10 Virginia Tech Space-Time Adaptive Radio Case Study 323
6.10.1 Overview of the VT-STAR Architecture 324
6.10.2 RF Design of VT-STAR 325
6.10.3 Software Issues for VT-STAR 327
6.10.4 Key Design Issues of VT-STAR 333
6.11 Conclusion 333
6.12 Questions 335

7 DIGITAL HARDWARE CHOICES **339**
7.1 Introduction 339
7.2 Key Hardware Elements 340

7.3 DSP Processors 342
7.3.1 DSP Core 342
7.3.2 DSP Architectures 342
7.3.3 Numeric Representation 347
7.3.4 Addressing 350
7.3.5 Pipelining 353
7.3.6 Peripherals and Additional Features 355
7.3.7 Multi-Processing 355
7.3.8 Multi-Processing Using a Real-Time Operating System 357
7.3.9 The Software Design Cycle 358
7.3.10 Maximizing Performance 360
7.3.11 Benchmarks and Performance Evaluation 368
7.3.12 Case Study: TMS320C54x Series DSPs 370
7.4 Field Programmable Gate Arrays 371
7.4.1 Operation of an SRAM-Based FPGA Cell 371
7.4.2 Implementing DSP Functions in FPGAs 373
7.4.3 FPGA Architectures 373
7.4.4 Applications of FPGAs to Software Radios 377
7.4.5 Design Principles using FPGAs 378
7.5 Trade-Offs in Using DSPs, FPGAs, and ASICs 379
7.6 Power Management Issues 379
7.6.1 DSP Power Management 380
7.6.2 Low-Power VLSI Design 381
7.6.3 Architectural-/System-Level Approaches 383
7.7 Using a Combination of DSPs, FPGAs, and ASICs 386
7.8 Conclusion 387
7.9 Questions 388

8 OBJECT-ORIENTED REPRESENTATION OF RADIOS AND NETWORK RESOURCES 391
8.1 Introduction 391
8.2 Networks 392
8.2.1 System Layers 395
8.2.2 Switching 396
8.2.3 Quality-of-Service 396
8.2.4 Internet Protocol 397
8.2.5 Asynchronous Transfer Mode 403
8.2.6 Networks and Software Radios 404
8.3 Object-Oriented Programming 405
8.3.1 Objects 406
8.3.2 Java 411
8.3.3 Java and Software Radios 414
8.3.4 The Radio Virtual Machine 415
8.3.5 Object-Oriented Software and Software Radios 415

8.4 Object Brokers 416
8.4.1 Common Object Request Broker Architecture 419
8.4.2 Software Radio Implementation Issues 422
8.4.3 Object Brokers and Software Radios 424
8.5 Mobile Application Environments 424
8.5.1 MExE 427
8.5.2 Service Discovery 432
8.5.3 Mobile Application Environments and Software Radios 432
8.5.4 Security in Software Radio 433
8.6 Joint Tactical Radio System 434
8.6.1 Hardware Classes 434
8.6.2 SCA Structure 435
8.7 Conclusion 440
8.8 Questions 441

9 CASE STUDIES IN SOFTWARE RADIO DESIGN 443
9.1 Introduction and a Historical Perspective 443
9.1.1 Architectural Characteristics Intrinsic to a Software Radio 445
9.1.2 Architectural Characteristics Important to a Software Radio 446
9.1.3 Architectural Characteristics of Practical Software Radios 448
9.2 SPEAKeasy 450
9.2.1 SPEAKeasy Phase I 451
9.2.2 SPEAKeasy Phase II 455
9.2.3 SPEAKeasy Summary 462
9.3 JTRS 462
9.3.1 Goals of the SCA 465
9.3.2 Attributes of the SCA Developed from the PMCS Guidance Document 465
9.3.3 SCA Architectural Details 466
9.3.4 JTRS Summary 475
9.3.5 SDR Forum Architecture Details 476
9.3.6 Summary 481
9.4 Wireless Information Transfer System 481
9.4.1 Architecture Goals 483
9.4.2 Architecture Overview 483
9.4.3 Software Architecture 484
9.4.4 Hardware Architecture 484
9.4.5 Architectural Details 486
9.4.6 WITS Summary 490
9.5 SDR-3000 Digital Transceiver Subsystem 492
9.6 SpectrumWare 494
9.6.1 SpectrumWare System Description 495
9.6.2 Input/Output 495
9.6.3 Programming Environment 496

9.7.2 Layered Radio Architecture Implementation Example 507
9.7.3 CHARIOT Summary 509
9.8 Conclusion 509
9.9 Questions 512

A RF ENGINEERING BOOKS AND TRADE PUBLICATIONS 515
A.1 Electronics 515
A.2 RF Circuit Design and S-Parameters 515
A.3 Filters 516
A.4 Microwaves 516
A.5 Oscillators 516
A.6 Phase Locked Loops and Frequency Synthesizers 516
A.7 Receivers and Systems 517
A.8 PSpice 517
A.9 Trade Publications and Periodicals 517
A.10 Web-Accessible Tutorial Materials 518

B THE COORDINATE ROTATION DIGITAL COMPUTER ALGORITHM 519
B.1 Introduction 519
B.2 CORDIC Overview 520
B.3 Derivation of the CORDIC Algorithm 520
B.3.1 Translating a Point Along a Circle of Radius R 520
B.3.2 Rotation Through Iterative Subrotations 522
B.3.3 Computationally Simplifying the Iterative Rotations 523
B.3.4 Putting the Equations in Final Form 524
B.3.5 Vectoring Mode 525
B.4 CORDIC Algorithm Performance 526
B.5 Extensions to the CORDIC Algorithm 527

REFERENCES 531

GLOSSARY OF ACRONYMS 549

ABOUT THE AUTHOR 557

ABOUT THE CHAPTER CO-AUTHORS 559

INDEX 561

Chapter 3

MULTIRATE SIGNAL PROCESSING

Low complexity digital signal processing applications can be performed using a single sampling rate for the whole implementation. For high complexity applications, sometimes it is beneficial to alter the sampling rate used at different stages of the system to reduce the required computational complexity and to enable the use of low-cost digital signal processing structures. The implementation of a digital signal processing application using variable sampling rates is called *multirate digital signal processing*.

3.1 Introduction

Multirate digital signal processing techniques can improve the flexibility of a software radio. For instance, a multimode receiver may share the same ADC and sampling rate for multiple receiver implementations. Yet there is an optimal sampling rate and resolution for each of these standards. At the receiver, a signal is typically digitized at a rate significantly higher than the bandwidth so that the anti-aliasing filter specifications can be relaxed, but then the sampling rate is immediately reduced to the minimum sampling rate to avoid excessive computation. Furthermore, it is desirable that the sample rate be exactly some integer multiple of the symbol rate to ensure that the receiver is synchronized to the incoming data, otherwise, symbols may be lost or counted twice. For a transmitter, the signals are created at the minimum sampling rate to avoid excessive computation, but the sampling rate is increased before the DAC to relax the specifications of the interpolation filter.

3.1.1 Cost

In any system design, it is important to minimize the resources used by the system to perform its intended application. For a computer-based design, the most critical parameter is MIPS (millions of instructions per second). A system design that can deliver a desired application using a minimal number of MIPS is generally less expensive and consumes less power than a design that requires higher-performance hardware. Any technique that will reduce the number of instructions needed to perform a task is a welcome addition to

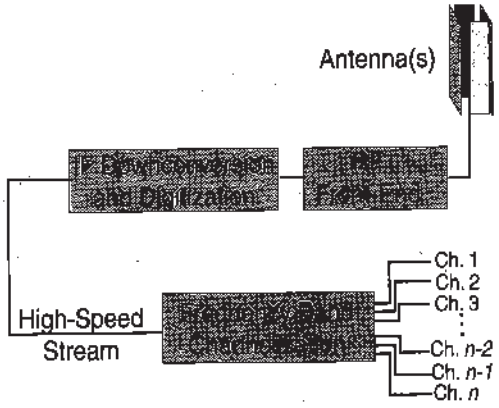


Figure 3.1: Block Diagram of a Simplified Basestation Receiver.

an efficient system design. One of the most basic means of reducing the number of MIPS used is to minimize the information content of the signal to be processed to only its essential content.

An example of a multimode system is a basestation designed to handle multiple services. The basestation can digitize the whole band of the reverse link and digitally extract individual channels. The advantage of this approach is the need for only a single RF front-end and ADC to service a whole spectrum of standards. On the forward link, signals can be combined in the digital domain so that a single, common, RF front-end and DAC can be used to service a sector. This implementation option can drastically reduce the cost of RF components. Furthermore, since the cost of state-of-the-art, high-speed digital signal processing chips rises exponentially with speed, the use of channelization (to lower sampling rates) and parallel processing allows the use of lower-speed chips, significantly reducing the implementation cost. Figure 3.1 shows an example of channelization where the task of a basestation's radio interface is to collect the signals transmitted by all mobile units and to separate each channel for individual processing.

Each new data stream stemming from the original collected data stream has a lower data rate than that of the original; if the original stream is k bps, each channelized stream will be $\frac{k}{n}$, where n is the number of channels. The lower data rate at which each channel transmits allows parallel implementation of less demanding algorithms, enabling the use of less complex and potentially more power-efficient hardware.

3.1.2 Flexibility

At times, it is important for a system to operate in several different modes. This need is most apparent in cellular phones designed to operate using different standards. For example, the design specification may require a basestation to be able to handle IS-136 (30 kHz channels), GSM (200 kHz), IS-95 (1.25 MHz channels), and WCDMA (5 MHz channels).

While each of these standards has different requirements, it is possible for a single system to be able to handle all of them.

ADCs are usually subjected to a resolution versus speed trade-off. Higher-speed data acquisition is usually achieved through the reduction of resolution; an eight-bit sample is acquired faster than a twelve-bit sample. The use of multirate digital signal processing techniques in a system design allows the designer to trade sample rate for data resolution. For more details on analog to digital conversion, refer to Chapter 5.

3.1.3 Overview of the Chapter

The application of multirate techniques to a system design allows the designer significant latitude in selecting the system's cost, modes of operation, level of parallelism, and level of quantization noise in the system. For these and other reasons, an understanding of the multirate digital signal processing algorithms is necessary.

In Section 3.2, rate conversion principles applicable to upsampling and downsampling will be presented. In Section 3.3, polyphase filters will be reviewed. Section 3.4 will present a review of digital filter banks and will include examples of existing systems employing multirate techniques. Section 3.5 applies multirate techniques to the problem of symbol timing recovery. The chapter is concluded in Section 3.6.

3.2 Sample Rate Conversion Principles

The conversion of a data stream's sample rate is an important part of digital signal processing. A data stream can be downsampled to a lower sampling rate or upsampled to a higher sampling rate. Figure 3.2 represents the process of sample rate conversion. A signal, $x(n)$, sampled at $F_{s,x}$ is passed through a sampling rate converter. The output signal is the waveform $y(m)$ sampled at $F_{s,y}$. The sampling rate converter can be thought of as the time-varying filter $h(n, m)$. Note that n and m are indices for different time scales.

The sampling rate conversion factor is expressed as the ratio of two relatively prime numbers, i.e., numbers whose greatest common divisor is one. Therefore, the filter $h(n, m)$ is a rate converter whose sampling ratio is determined by Equation 3.1.

$$\frac{F_{s,x}}{F_{s,y}} = \frac{I}{D} \tag{3.1}$$

Figure 3.2: Sampling Rate Converter.

The factors I and D in Equation 3.1 are the interpolation and decimation factors, respectively, both of which are relatively prime. In the following decimation example and analysis, I is assumed to be 1 [31,32].

3.2.1 Decimation

Decimation is the process by which high-frequency information is eliminated from a signal to reduce the sampling frequency without resulting in aliasing. As shown in Chapter 5, a sampled signal, seen in Figure 3.3, repeats its spectrum every 2π radians. If decimation without filtering were performed, aliasing would occur, an example of which is seen in Figure 3.4.

Figure 3.5 shows a block diagram of the decimation process. The operation is composed of lowpass filtering followed by downsampling. The downsampler picks a subset of the samples that are passed through the lowpass filter (LPF). The LPF used is designed to avoid aliasing and has a cutoff of $\frac{\pi}{D}$, the point that allows the non-aliased part of the signal seen in Figure 3.4 to pass.

The end result of the decimation procedure is the content of the original signal below $\frac{\pi}{D}$, but it is sampled at a lower rate. Figure 3.6 shows the expected spectrum of the decimated signal.

To fully explore the process of signal decimation, an analysis of the algorithm is necessary. Two processes are used in the decimation procedure: filtering and downsampling. Equation 3.2 shows filtering performed with filter $h_D(n)$ on signal $x(n)$ in the time domain. Subsampling is the selection of every Dth sample, shown in Equation 3.3.

$$v(n) = \sum_{k=-\infty}^{\infty} h_D(k)x(n-k) \tag{3.2}$$

$$y(m) = v(mD) \tag{3.3}$$

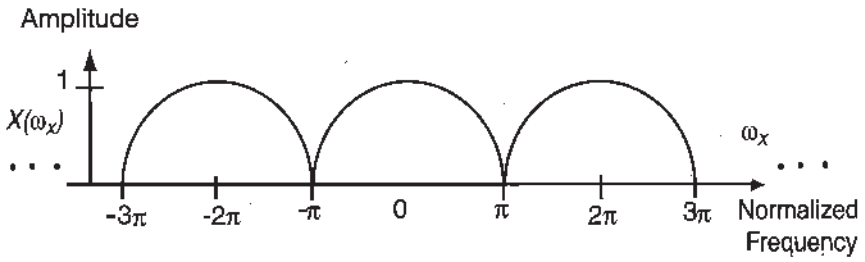


Figure 3.3: Sampled Signal Spectrum.

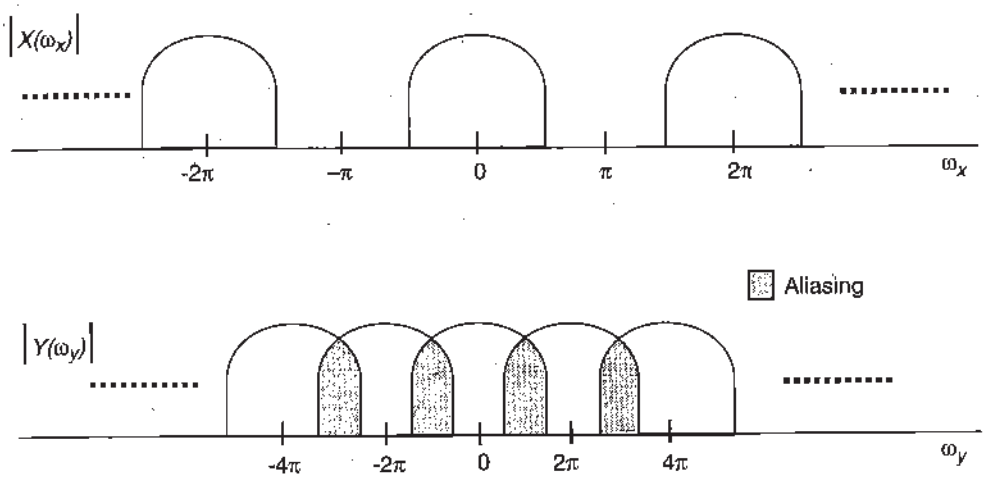


Figure 3.4: Aliased Signal Spectrum ($D = 3$).

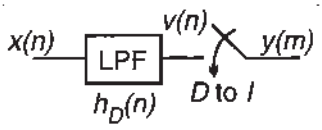


Figure 3.5: Decimation Circuit Block Diagram.

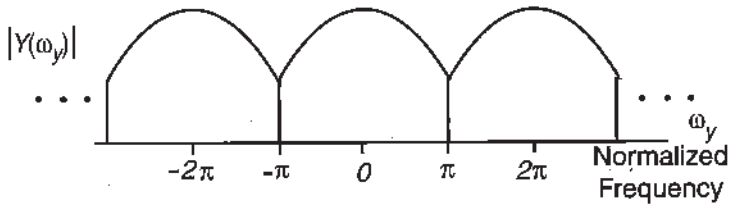


Figure 3.6: Decimated Signal Magnitude Spectrum. (Note in this case 2π corresponds to $F_{s,x}/D$ in actual frequency.)

The combination of Equations 3.2 and 3.3 yields

$$y(m) = \sum_{k=-\infty}^{\infty} h_D(k)x(mD-k). \tag{3.4}$$

Define $\widetilde{v}(n)$ as

$$\widetilde{v}(n) \equiv \begin{cases} v(n) & n = 0, \pm D, \pm 2D, \ldots \\ 0 & \text{otherwise} \end{cases} \tag{3.5}$$

and model as $v(n)$ sampled with impulse train $p(n)$, where $p(n)$ is defined in Equation 3.6 using the Fourier series for a sampling impulse of period D. $\widetilde{v}(n)$ is shown in Equation 3.7.

$$p(n) \equiv \frac{1}{D}\sum_{k=0}^{D-1} e^{j2\pi\frac{kn}{D}} \tag{3.6}$$

$$\widetilde{v}(n) = v(n)p(n) \tag{3.7}$$

Using Equations 3.3–3.7, the following relationship can be derived:

$$y(m) = v(mD) = v(mD)p(mD) = \widetilde{v}(m). \tag{3.8}$$

The discrete time Fourier transform (DTFT) of $y(m)$ can be calculated as

$$Y(\omega_y) = \sum_{m'=-\infty}^{\infty} y(m')e^{-j\omega_y m'}$$

where

$$\omega_y = 2\pi\frac{f}{F_{s,y}}$$

represents the normalized sample rate with respect to signal y.

$$\begin{aligned} Y(\omega_y) &= \sum_{m'=-\infty}^{\infty} v(m'D)e^{-j\omega_y m'} \\ &= \sum_{m'=-\infty}^{\infty} \widetilde{v}(m')e^{-j\omega_y m'} \end{aligned}$$

Let $m = m'D$.

$$\begin{aligned} Y(\omega_y) &= \sum_{m=-\infty}^{\infty} \tilde{v}(m) e^{-j\frac{\omega_y}{D}m} \\ &= \sum_{m=-\infty}^{\infty} v(m)p(m) e^{-j\frac{\omega_y}{D}m} \\ &= \sum_{m=-\infty}^{\infty} v(m) \left[\frac{1}{D}\sum_{k=0}^{D-1} e^{j2\pi\frac{m}{D}k}\right] e^{-j\frac{\omega_y}{D}m} \\ &= \frac{1}{D}\sum_{k=0}^{D-1}\sum_{m=-\infty}^{\infty} v(m)\left[e^{-j2\pi\frac{k}{D}} e^{j\frac{\omega_y}{D}}\right]^{-m} \\ &= \frac{1}{D}\sum_{k=0}^{D-1}\sum_{m=-\infty}^{\infty} v(m) e^{-j\left(\frac{\omega_y}{D}-\frac{2\pi k}{D}\right)m} \\ &= \frac{1}{D}\sum_{k=0}^{D-1} V\left(\frac{\omega_y}{D}-\frac{2\pi k}{D}\right) \end{aligned} \tag{3.9}$$

Note the sampled $v(n)$ frequency domain representation consists of replicated images at $\frac{2\pi k}{D}$ with a normalized frequency scale of ω_y/D.

Since $V(\omega_y) = H_D(\omega_y)X(\omega_y)$, the final form of Equation 3.9 can be rewritten as

$$Y(\omega_y) = \frac{1}{D}\sum_{k=0}^{D-1} H_D\left(\frac{\omega_y - 2\pi k}{D}\right) X\left(\frac{\omega_y - 2\pi k}{D}\right). \tag{3.10}$$

From Equation 3.10, it can be seen that an image of Y is replicated every $\frac{2\pi}{D}$ radians (relative to the original signal's normalized sample rate).

The image of interest, the primary image, is centered at zero; Equation 3.11 shows the form of the primary image ($k = 0$).

$$Y(\omega_y) = \frac{1}{D} H_D\left(\frac{\omega_y}{D}\right) X\left(\frac{\omega_y}{D}\right) \tag{3.11}$$

Assuming that H_D is a perfect LPF, [1] i.e.,

$$H_D(\frac{\omega_y}{D}) = \begin{cases} 1 & |\omega_y| \le \pi \\ 0 & \text{otherwise,} \end{cases} \tag{3.12}$$

[1] Note: with respect to the original sample rate

$$H_D(\omega_x) = \begin{cases} 1 & |\omega_x| \le \pi/D \\ 0 & \text{otherwise} \end{cases}$$

then Equation 3.11 is simplified to

$$Y(\omega_y) = \frac{1}{D} X\left(\frac{\omega_y}{D}\right), |\omega_y| \leq \pi. \tag{3.13}$$

Decimation filters out the information in the original signal above $\frac{\pi}{D}$ (with respect to the original sample rate). A lowpass direct mapping is possible from ω_y to ω_x and vice versa; this relationship is best described as the spectrum spanned by $X(\omega_x)$, $0 \leq \omega_x \leq \frac{\pi}{D}$, is also spanned by $Y(\omega_y)$, $0 \leq \omega_y \leq \pi$.

3.2.2 Interpolation

Upsampling is the process to increase the number of points per unit time used to describe a signal. The spectral content of the signal does not change; what *does* change is the spectral separation between images of the original spectrum. *When upsampling is employed, no new information is added to the signal.* The process of upsampling decreases the time between samples of a signal. This process can be used for matching sampling rates between two systems or as the last step before the DAC to help relax the requirements for the reconstruction filter. If the signal shown in Figure 3.6, $X(\omega_x)$, were sampled at twice the rate shown in the figure, its spectrum would appear as shown in Figure 3.7. Several methods of interpolation are presented in this section: zero-insertion, zero-order-hold (ZOH), zero-insertion and raised-cosine filtering, and fast Fourier transform (FFT) expansion.

Zero-Insertion Interpolation

In this method, zeros are inserted between samples of a signal, generating a new signal. This new signal is then lowpass filtered, yielding an upsampled version of the original (see Figure 3.8).

For the purpose of analysis, assume that the original signal is $x(n)$ and that the goal is to upsample it by a factor of I. $I-1$ zeros are inserted between each pair of consecutive samples of $x(n)$, yielding $v(m)$, which can then be defined as

$$v(m) = \left\{ \begin{array}{ll} x\left(\frac{m}{I}\right) & m = 0, \pm I, \pm 2I, \ldots \\ 0 & \text{otherwise} \end{array} \right\}. \tag{3.14}$$

Taking the Z-transform of $v(m)$ and defining $m' = mI$ yields $V(z)$:

$$\begin{aligned} V(z) &= \sum_{m'=-\infty}^{\infty} v(m')z^{-m'} = \sum_{m=-\infty}^{\infty} v(mI)z^{-mI} \\ &= \sum_{m=-\infty}^{\infty} x(m)z^{-mI} = \sum_{m=-\infty}^{\infty} x(m)\left(z^I\right)^{-m}. \end{aligned} \tag{3.15}$$

From Equations 3.15 and 3.14,

$$V(z) = X(z^I). \tag{3.16}$$

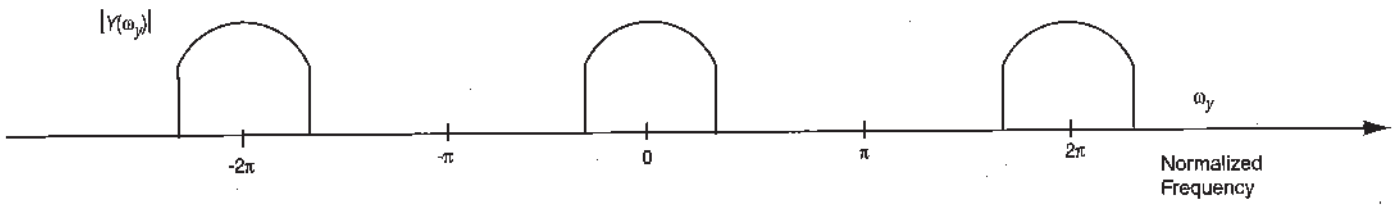


Figure 3.7: Upsampled Signal Magnitude Spectrum ($I = 2$).

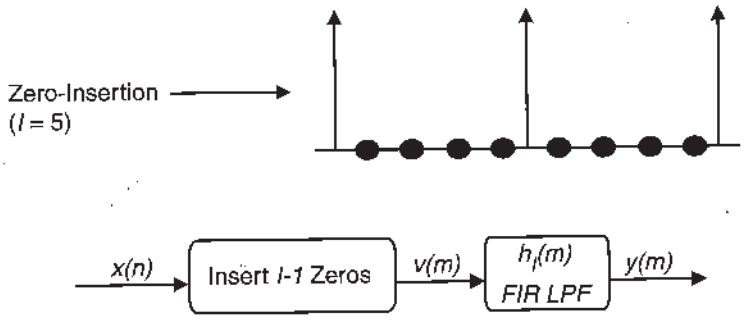


Figure 3.8: Direct Implementation of an Interpolator.

To evaluate the DTFT of $V(z)$, evaluate Equation 3.16 with $z = e^{j\omega_y}$, yielding

$$V(\omega_y) = \frac{1}{I} X(\omega_y I). \tag{3.17}$$

Note that the $\frac{1}{I}$ factor is included to model the reduction in power (in the normalized scale) resulting from inserting $I - 1$ zeros.

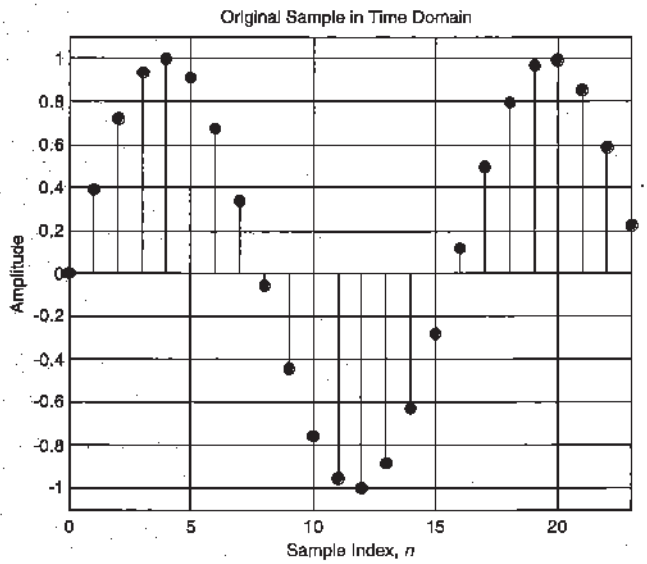
Equation 3.17 shows a contraction of the spectrum; a copy of the spectrum of the original signal is generated every $\frac{2\pi}{I}$ radians instead of every 2π. This means that a direct mapping between ω_y and ω_x is possible, as seen in Section 3.2.1.

Since $\omega_y = 2\pi \frac{f}{F_{s,y}}$ and $\omega_x = 2\pi \frac{f}{F_{s,x}}$, $\omega_y = \frac{\omega_x}{I}$, and therefore,

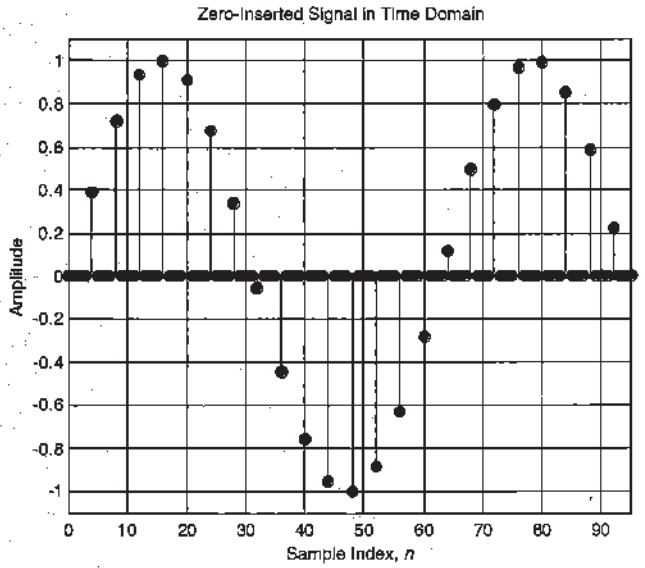
$$0 \leq \omega_x \leq \pi \text{ maps to } 0 \leq \omega_y \leq \frac{\pi}{I}. \tag{3.18}$$

Using Equation 3.18, the range for ω_y becomes $0 \leq |\omega_y| \leq \pi/I$. Equation 3.18 implies that the upsampled signal can be recovered by filtering the zero-inserted signal with an LPF $h_I(m)$ whose cutoff frequency is less than $\frac{\pi}{I}$.

For illustrative purposes, a MATLAB example of this process is considered. The upsampling factor selected for this example is four. Figure 3.9 shows the original signal and its zero-insertion version. Figure 3.10 shows the time signals' respective spectrum. As expected, the zero-insertion version contains an additional $I - 1 = 3$ copies of the original spectrum, or a copy every $\frac{2\pi}{I}$ radians. This signal was filtered with a 200-order LPF with a cutoff frequency of $\frac{\pi}{I}$ radians, resulting in an interpolated version of the signal seen in Figure 3.11. Note on the normalized scale, the peak has moved closer to zero radians per second for the upsampled signal compared to that of Figure 3.10a. The true frequency is still the same but is lower when normalized with respect to the sample rate.

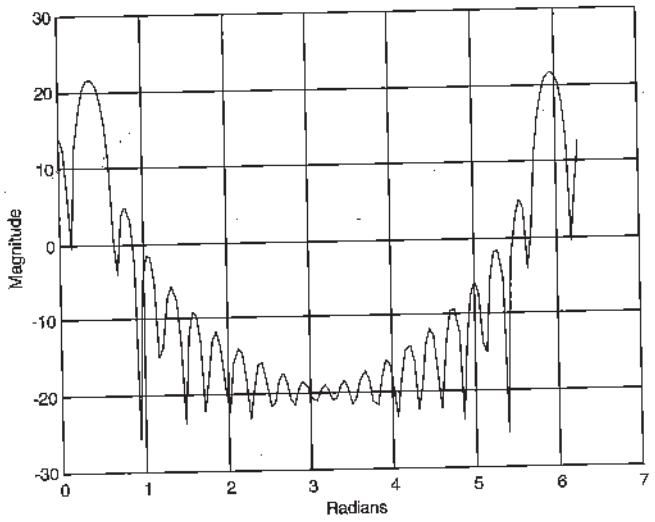


(a)

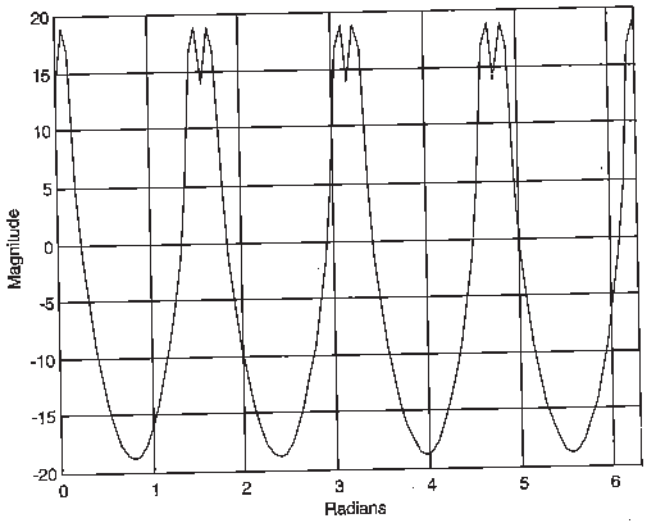


(b)

Figure 3.9: (a) Sinusoid and (b) Zero-Insertion Equivalent ($I = 4$).

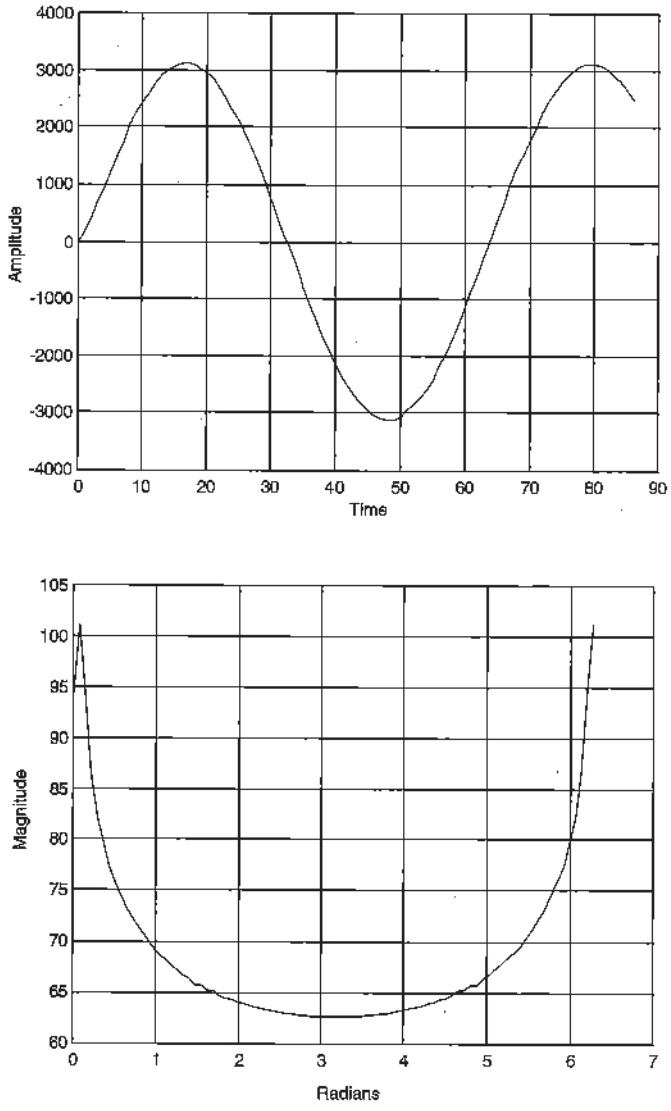


(a)



(b)

Figure 3.10: (a) Spectrum of Sinusoid and (b) Zero-Insertion Equivalent ($I = 4$; note both scales are with respect to input sample rate).



(b)

Figure 3.11: (a) Time and (b) Frequency Domain Representation of the Interpolated Signal ($I = 4$; note both scales are with respect to the output sample rate).

Zero-Order-Hold Interpolation

In this method, space between two samples is filled with a sample generated using some function, in this case a ZOH or zero-order interpolation. This method for upsampling is similar to the zero-insertion method described in Section 3.2.2, but this method is computationally more expensive and generates less distortion at the higher frequencies for a given interpolation filter. In zero-order interpolation, a signal is upsampled by extending a sample's value over several samples, as seen in Figure 3.12, before lowpass filtering.

Figure 3.12 presents an example of ZOH upsampling at four times the original sampling rate. Sample n from the original sequence $x[n]$ is placed in samples $v[4n]$, $v[4n+1]$, $v[4n+2]$, and $v[4n+3]$ of the new sequence $v[m]$ of size $4n$. This addition of extra points creates a significant amount of high-frequency distortion in the form of replicated, attenuated images in the spectrum.

The zero-insertion signal, $v(m)$, is generated using Equation 3.14. In the ZOH version of this signal, $\widehat{v}(m)$ can be generated by Equations 3.19 and 3.20. A ZOH can be considered as the sum of the sampled signal plus $I-1$ delayed copies of $v(m)$, shown in Equation 3.20.

$$\widehat{v}(m) = \left\{ \begin{array}{ll} x\left(\frac{m}{I}\right) & m = 0, \pm I, \pm 2I, \ldots \\ 0 & \text{otherwise} \end{array} \right\} \tag{3.19}$$

$$v(n) = \sum_{k=0}^{I-1} \widehat{v}(n-k) \tag{3.20}$$

With the Z-transform for $\widehat{v}(m)$ defined as

$$\widehat{V}(z) = \sum_{m=-0}^{\infty} \widehat{v}(m) z^{-m}, \tag{3.21}$$

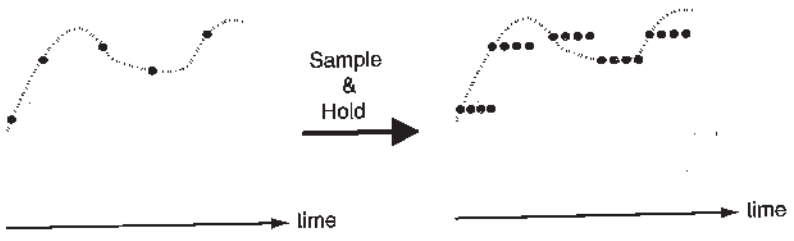


Figure 3.12: Sample and Hold. (Note: Black dots are the sampled signal and gray dots represent the held value.)

$V(z)$ can be calculated as

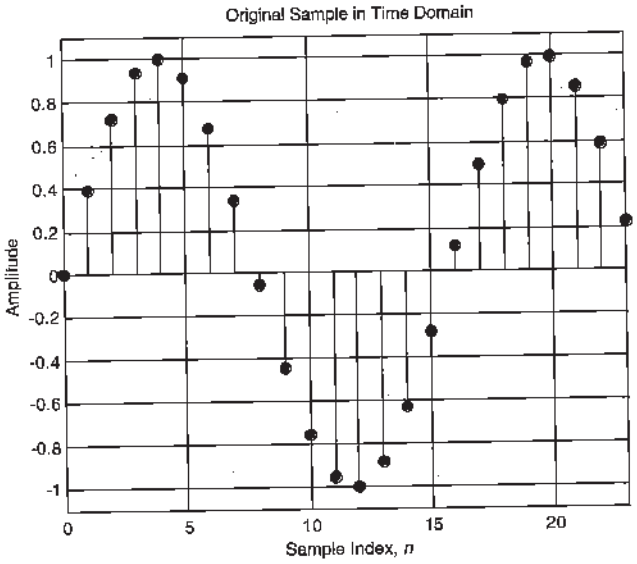
$$\begin{aligned} V(z) &= \sum_{k=0}^{I} \widehat{V}(z) z^{-k} \\ &= \sum_{k=0}^{I} \sum_{m=-\infty}^{\infty} \widehat{v}(mI) z^{-mI} z^{-k} \\ &= \sum_{m=-\infty}^{\infty} \widehat{v}(mI) z^{-mI} \sum_{k=0}^{I} z^{-k} \\ &= \sum_{m=-\infty}^{\infty} x(m) z^{-mI} \sum_{k=0}^{\infty} \left(u(k) - u(k-I)\right) z^{-k} \\ &= X(z^I) \left(\frac{1-z^{-I}}{1-z^{-1}}\right) \end{aligned} \tag{3.22}$$

Substituting $e^{j\omega_y}$ for z in Equation 3.22 yields Equation 3.24, or a lowpass filtered version of the spectrum seen in Equation 3.17. The attenuated images resulting from this process can be filtered out using an LPF with a cutoff frequency less than $\frac{\pi}{I}$, where I is the upsampling ratio.

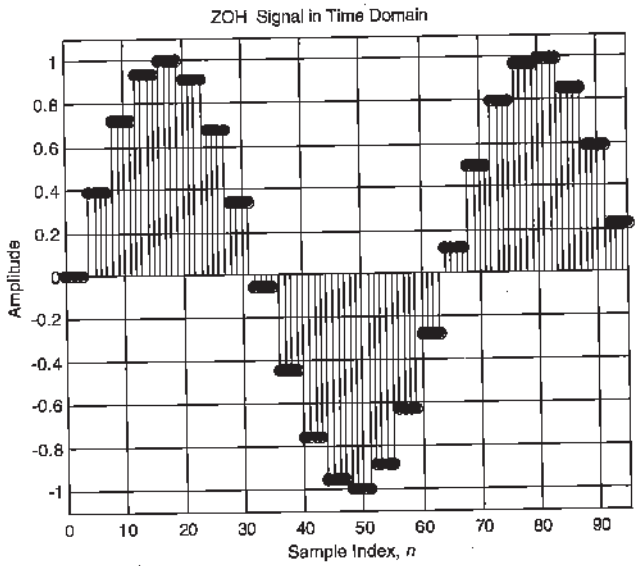
$$\begin{aligned} V(\omega_y) &= X(\omega_y I) \left(\frac{1-e^{-j\omega_y I}}{1-e^{-j\omega_y}}\right) \\ &= X(\omega_y I) \frac{e^{-j\frac{\omega_y}{2}I}}{e^{-j\frac{\omega_y}{2}}} \left(\frac{\sin\left(\frac{\omega_y I}{2}\right)}{\sin\left(\frac{\omega_y}{2}\right)}\right) \\ &= X(\omega_y I) e^{-j\frac{\omega_y}{2}(I-1)} \frac{\sin(\omega_y I/2)}{\sin(\omega_y/2)} \end{aligned} \tag{3.23}$$

Figure 3.13 shows a sinusoid and its zero-order upsampled version before filtering. The high-frequency components generated by the ZOH operation are seen in Figure 3.14. A significant amount of distortion is apparently generated by this procedure. However, given a well oversampled signal, this distortion is well beyond the spectrum of interest, so it can be filtered out using an LPF. Figure 3.15 shows the waveform resulting from the filtering operation and its spectrum. Note that the spurious signal content has been considerably reduced by the holding process compared to the zero-insertion interpolated signal for the same filter order. The gain of the LPF can be adjusted to ensure equal power for the input and output signals.

Note that the use of a ZOH for interpolation introduces distortion on the interpolated signal beyond that introduced through the zero-insertion method. Compared to the zero-insertion spectrum, the ZOH spectrum has an extra sinc term that helps to attenuate the high frequency distortion although it slightly distorts the desired portion of the interpolated signal spectrum. This in-band distortion can be cancelled by using the proper passband characteristics in the interpolation filter design.



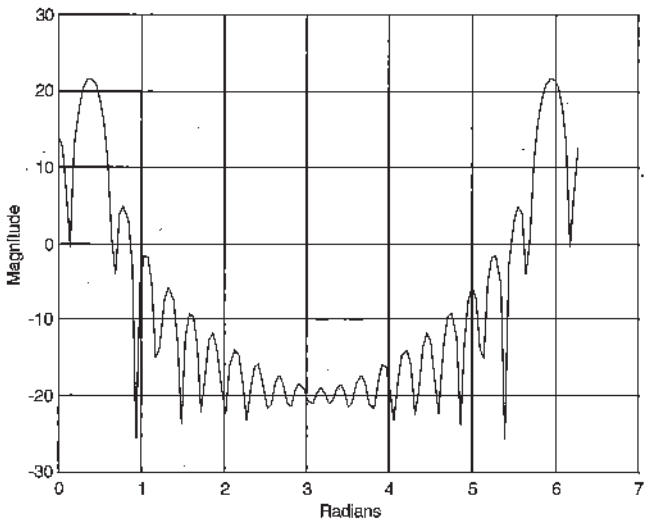
(a)



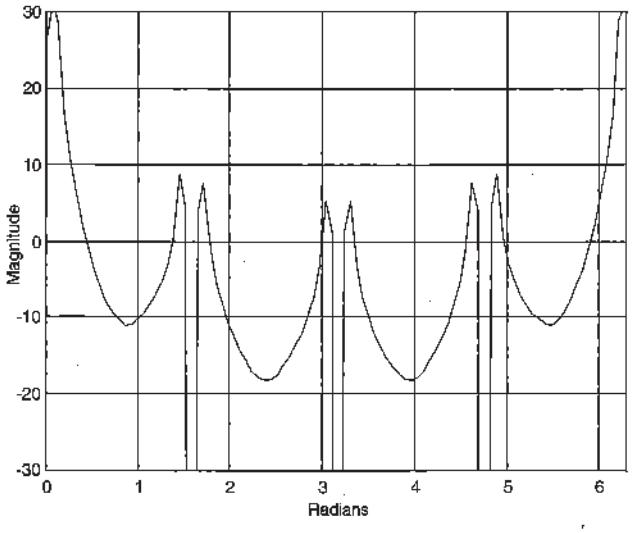
(b)

Figure 3.13: (a) Sinusoid and (b) Zero-Order-Hold Equivalent.

70

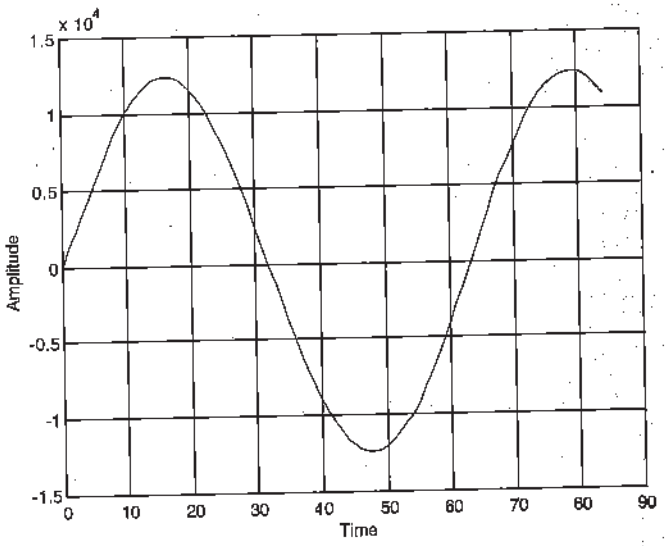


(a)

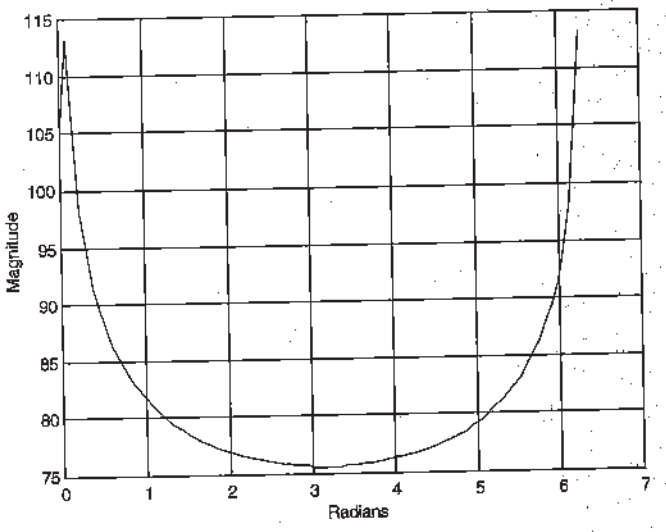


(b)

Figure 3.14: (a) Frequency Domain Representation (Magnitude) of the Original Sinusoid, (b) Zero-Order-Hold Interpolated Signal (scaled to output sample rate).



(a)



(b)

Figure 3.15: (a) Time and (b) Frequency Domain Representation (Magnitude) of Filtered ZOH Interpolated Signal (scaled to output sample rate).

72

Zero-Insertion and Raised-Cosine

To minimize inter-symbol interference (ISI), the use of raised-cosine pulse-shaping is important. If the upsampling of a pulse code modulation (PCM) signal is to be performed at the transmitter, the upsampling and raised-cosine filtering can be combined to simplify the overall design.

This combination is performed by using the zero-insertion interpolation method described earlier but with a raised-cosine filter instead of a lowpass interpolation filter. In practice, the LPF can be ignored since the raised-cosine filter has a smaller bandwidth than the interpolating LPF, as indicated in Figure 3.16. One could view the process as a pulse train that excites a filter that performs both pulse-shaping and interpolation.

Figure 3.17 shows an example of this application. A three-bit PCM sequence is sampled. Each of these samples is mapped onto a one-dimensional constellation and upsampled before modulation. Though not shown in Figure 3.17, the pulse-shaped waveform is now ready to be modulated and transmitted.

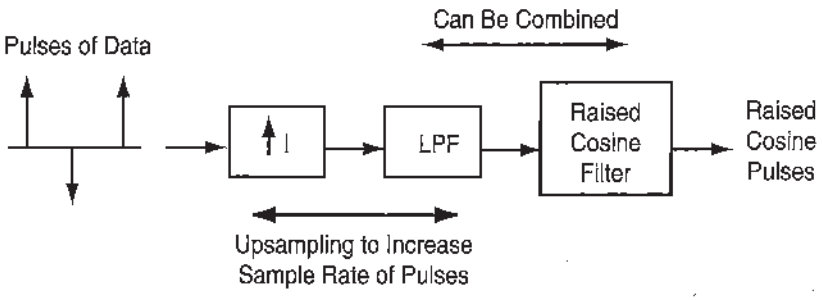
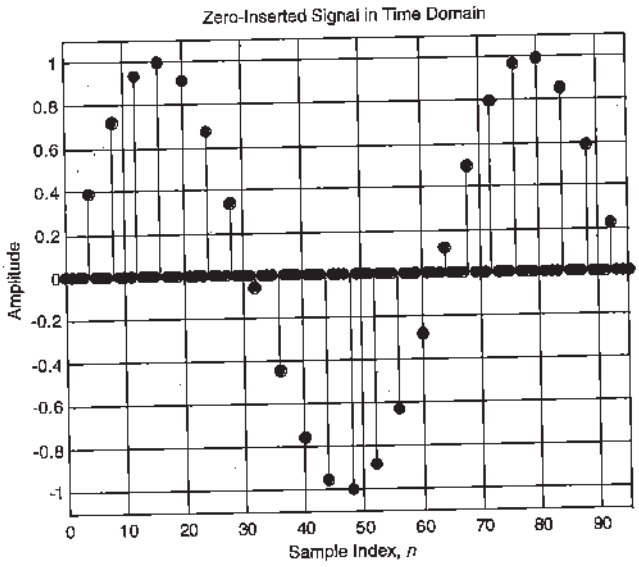


Figure 3.16: Combined Upsampling and Raised-Cosine Filtering

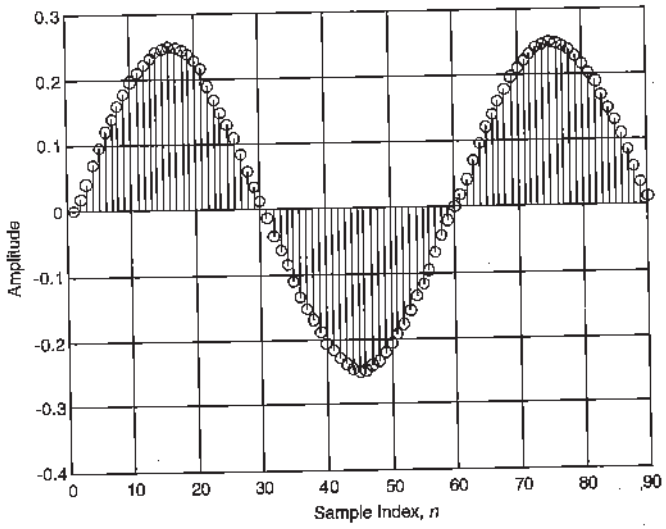
Fast Fourier Transform Expansion

The FFT lends itself well for upsampling a signal. The basic idea behind the FFT approach is to construct what the FFT of an oversampled signal should be by using the FFT of the original signal. Each sample at the output of an FFT is separated by $\frac{F_s}{N}$, where F_s is the sampling frequency and N is the number of points used in each block. Upsampling using an FFT is performed by taking N frequency samples, zero-padding $N(I-1)$ consecutive zeros between the positive and negative images, and performing an inverse FFT of the augmented signal. The principle behind this process is to create the Fourier transform of the desired signal by inserting zeros in the frequency domain representation of the signal. The resulting signal contains NI samples but spans the same time interval as the original signal.

Figure 3.18 shows the original signal and its upsampled version, the signal used in this example is a Rayleigh fading envelope. Figure 3.19 shows the magnitude and phase of the original spectrum and the padded magnitude and phase of the upsampled signal. Although



(a)



(b)

Figure 3.17: (a) Upsampled Time Domain Signal, (b) Interpolated Signal with Raised-Cosine Pulse-Shaping.

74

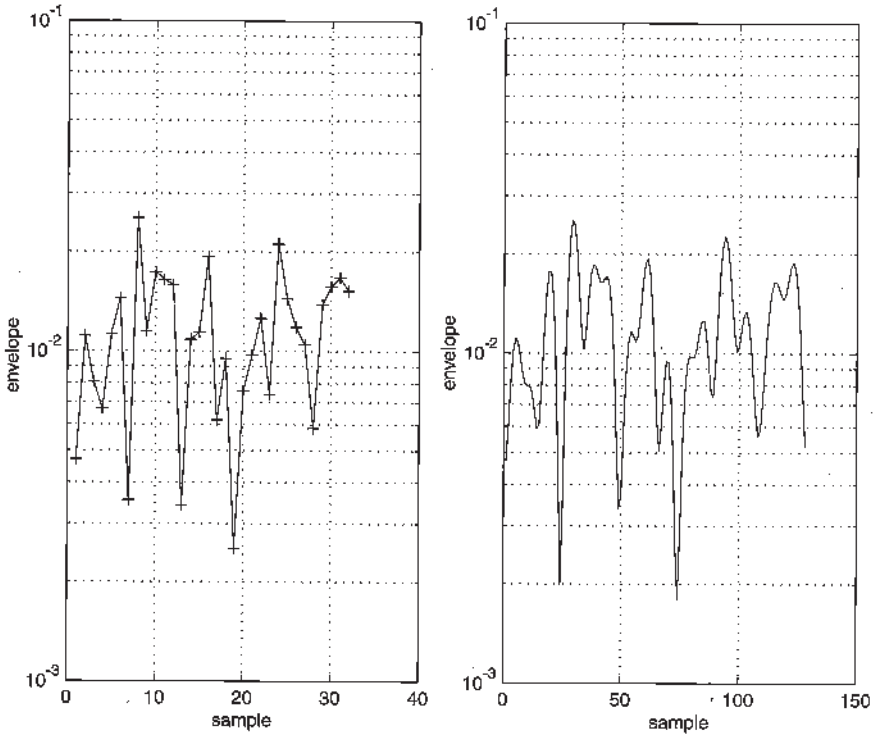
this example is not directly applicable to the implementation of a software radio, it is useful for the simulation of fading channels, where fades could span over several samples [33].

Assuming that a signal $x(n)$ was periodic with a period of N samples, then the discrete Fourier transform (DFT) of that signal is defined as

$$X(k) = \sum_{n=0}^{N-1} x(n) e^{-\frac{j2\pi kn}{N}}. \tag{3.24}$$

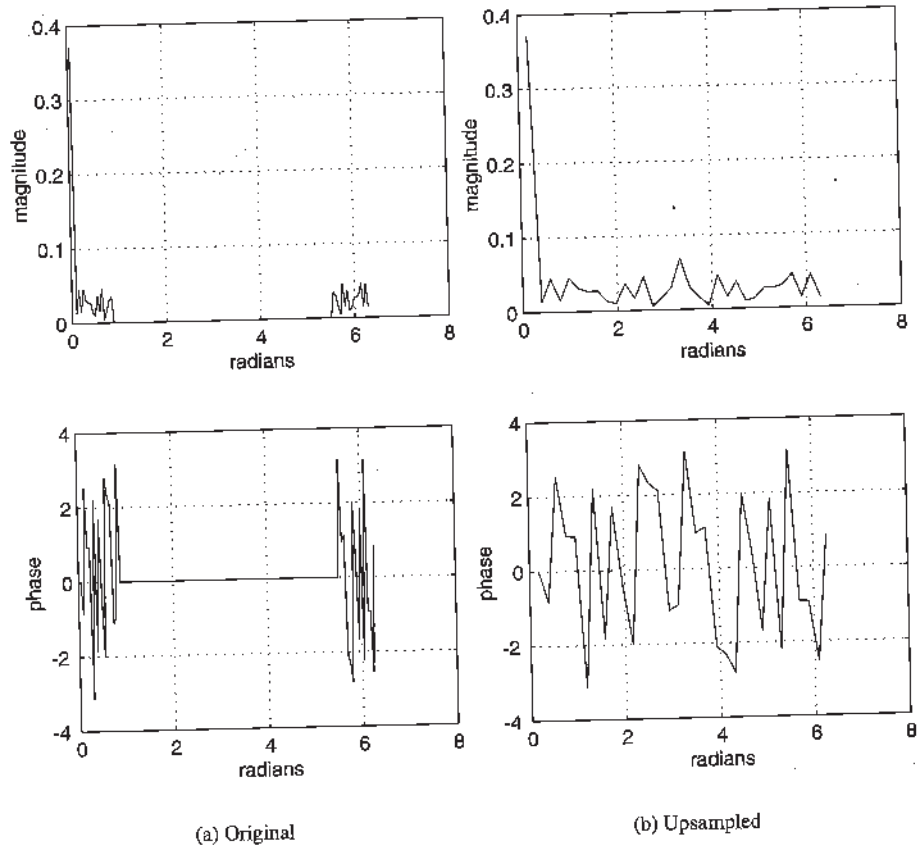
The inverse DFT (IDFT) of this signal would be

$$x(n) = \frac{1}{N} \sum_{k=0}^{N-1} X(k) e^{\frac{j2\pi kn}{N}}. \tag{3.25}$$



(a) Original (b) Upsampled

Figure 3.18: Rayleigh Fading Envelope



(a) Original (b) Upsampled

Figure 3.19: Rayleigh Fading Envelope, Original and Upsampled Spectrum (Magnitude and Phase).

If $X(k)$ were padded with $N(I-1)$ zeros over $\frac{\pi}{I} < \omega < 2\pi - \pi/I$, then the IDFT of that signal would be

$$x(m) = \frac{1}{NI} \sum_{k=0}^{NI-1} X(k) e^{\frac{j2\pi km}{NI}}. \tag{3.26}$$

The result seen in Equation 3.26 is a description of the same spectral content as seen in Equation 3.25, but the signal in Equation 3.26 is cyclical over NI samples; it is an oversampled version of the original signal $x(n)$.

3.2.3 Two Multirate Identities

As we have seen so far, the basic multirate system[2] consists of a sampling rate alteration device and a digital filter. In many applications, these components are cascaded. Changes in the position of these components will result in computationally efficient realization, keeping the input/output (I/O) relation unaltered. For example, placing the filter $H_D(z^D)$, the impulse response of which is interpolated by D, in front of the downsampler is equivalent to downsampling a signal by D first and then passing it through an LPF $H_D(z)$ created by expanding the original filter impulse response by a factor of D. This concept is shown in Figure 3.20.

A proof of the identity is given below. We know from Equation 3.9 that the subsampled signal in Figure 3.20b can be written as

$$V(\omega_y) = \frac{1}{D}\sum_{k=0}^{D-1} X\left(\frac{\omega_y}{D} - \frac{2\pi k}{D}\right), \tag{3.27}$$

and the filtered signal in Figure 3.20b becomes

$$Y(\omega_y) = \frac{1}{D}\sum_{k=0}^{D-1} X\left(\frac{\omega_y}{D} - \frac{2\pi k}{D}\right) H_D(\omega_y). \tag{3.28}$$

Likewise, Figure 3.20a can be represented as

$$Y(\omega_y) = \frac{1}{D}\sum_{k=0}^{D-1} X\left(\frac{\omega_y}{D} - \frac{2\pi k}{D}\right) H_D\left(D\frac{\omega_y - 2\pi k}{D}\right). \tag{3.29}$$

This equation can be realized by Figure 3.20. For notational simplicity, we will express Equation 3.28 as

$$Y(\omega_y) = [X(\omega_x)]_{D\downarrow} H_D(\omega_y) \tag{3.30}$$

where $D\downarrow$ corresponds to subsampling by a factor D. Equation 3.30 can be realized as Figure 3.20b, and thus Figures 3.20a and 3.20b represent equivalent systems.

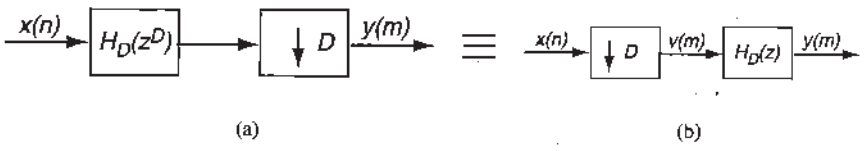


Figure 3.20: The Decimation Identity.

[2]Material for this section is adapted from [31, 34–36].

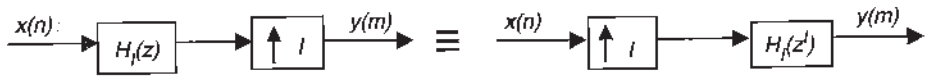


Figure 3.21: The Interpolation Identity.

Similar equivalence can be realized in the case of interpolation. There is a corresponding interpolator identity that states that upsampling using a filter $H_I(z^I)$ is equivalent to filtering with $H_I(z)$ and then zero-filling $I-1$ zeros between samples of the filter output. This identity is shown in Figure 3.21. The identities expressed by Figures 3.20 and 3.21 are called the *Noble Identities*.

3.2.4 Non-Integer-Rate Conversion

In Sections 3.2.1 and 3.2.2, integer downsampling by a factor of D and upsampling by a factor of I were discussed. In both cases, only integer-rate conversions were discussed. However, integer-rate conversions[3] are not always the desired operation. Non-integer-rate conversions are achievable through the use of cascaded interpolations/decimations such that a total rate change of $\frac{I}{D}$ is achieved.

Figure 3.22 shows a block diagram of the implementation of a non-integer-rate conversion. Since the input vector $x(n)$ is interpolated by a factor of I, then

$$v(p) = \begin{cases} x\left(\frac{p}{I}\right) & p = 0, \pm I, \pm 2I, \ldots \\ 0 & \text{otherwise.} \end{cases} \tag{3.31}$$

Since the filter $H(z)$ acts both as a decimation and an interpolation filter, then its frequency response should be

$$H(\omega) = \begin{cases} I & |\omega| \leq min\left(\frac{\pi}{I}, \frac{\pi}{D}\right) \\ 0 & \text{otherwise} \end{cases} \tag{3.32}$$

where the gain of I is used to compensate for the upsampling process to achieve the same output power.

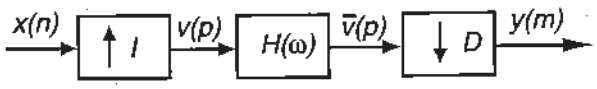


Figure 3.22: Non-Integer-Rate Conversion Block Diagram.

[3]Material for this section follows the development in [34].

Using the results of Equation 3.17, the signal $\overline{V}(\omega_p)$ is

$$\begin{aligned} \overline{V}(\omega_p) &= H(\omega_p)V(\omega_p) \\ &= H(\omega_p)X(\omega_p I) \end{aligned} \tag{3.33}$$

where $\omega_p = 2\pi f_p$ is the spectrum for the normalized sample rate corresponding to the time index p. Since $H(\omega_p)$ acts as the ideal interpolation filter, $\overline{V}(\omega_p)$ can be simplified to

$$\overline{V}(\omega_p) = \begin{cases} IX(\omega_p I) & |\omega_p| \leq min\left(\frac{\pi}{I}, \frac{\pi}{D}\right) \\ 0 & \text{otherwise.} \end{cases} \tag{3.34}$$

Decimating $\overline{V}(\omega_p)$ yields

$$Y(\omega_y) = \frac{1}{D}\sum_{k=0}^{D-1} \overline{V}\left(\frac{\omega_y - 2\pi k}{D}\right). \tag{3.35}$$

Combining Equations 3.34 and 3.35 and applying the conversion $\omega_y = \omega_p/D$ yields

$$Y(\omega_y) = \begin{cases} \frac{I}{D}X\left(\frac{I}{D}\omega_y\right) & |\omega_y| \leq min\left(\frac{\pi}{I}, \frac{\pi}{D}\right) \\ 0 & \text{otherwise.} \end{cases} \tag{3.36}$$

Equation 3.36 clearly shows that the sampling rate has been changed by a factor of $\frac{I}{D}$. Although this method allows non-integer-rate conversions, it is limited to changing rates by rational numbers.

3.2.5 Sampling Rate Conversion by Stages

The decimator and interpolator discussed so far are of a single-stage structure. When large changes in sampling rate are required, multiple stages of sample rate conversion are found to be more computationally efficient. Most practical systems employ a multi-stage structure, resulting in a considerable relaxation in the specifications of anti-aliasing (decimation) or anti-imaging (interpolation) filters in each stage compared to a single stage realization.

The decimation in Figure 3.23 can be realized in two stages if the decimation factor D can be expressed as a product of two integers, D_1 and D_2. Referring to Figure 3.24, in the first stage, the signal $x(n)$ is decimated by a factor of D_1. The output, $v(p)$ is further decimated by D_2 in the second stage resulting in an overall decimation of $x(n)$ by

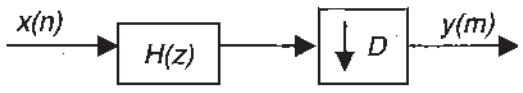


Figure 3.23: Decimation in a Single Stage.

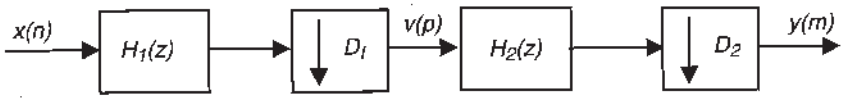


Figure 3.24: Decimation in Two Stages.

$D = (D_1 D_2)$. The filters $H_1(z)$ and $H_2(z)$ are so designed that the aliasing in the band of interest is below a prescribed level and that the overall passband and stopband tolerances are met. This multi-stage sampling rate conversion system offers less computation and more flexibility in filter design. An example is given below to illustrate the idea of multi-stage sampling rate conversion.

Example: Multi-Stage Sampling Rate Conversion

We have a discrete time signal with a sampling rate of 90 kHz. The signal has the desired information in the frequency band from 0 to 450 Hz (passband), and the band from 450 to 500 Hz is the transition band. The signal is to be decimated by a factor of ninety. The required tolerances are a passband ripple of 0.002 and a stopband ripple of 0.001.

Decimation in a Single Stage

First we consider a single-stage design as shown in Figure 3.25(a). The specifications of the required LPF are shown in Figure 3.25(b).

According to the formula by Kaiser, the approximate length of an FIR filter is given by [34]

$$N = \frac{-20log_{10}\sqrt{\delta_p \delta_s} - 13}{14.6\Delta f} \tag{3.37}$$

where peak passband ripple (linear) $\delta_p = 0.002$, peak stopband ripple (linear) $\delta_s = 0.001$, normalized transition bandwidth $\Delta f = \frac{f_s - f_p}{F_s}$, passband edge frequency $f_p = 450$ Hz, stopband edge frequency $f_s = 500$ Hz, and sampling frequency $F_s = 90$ kHz.

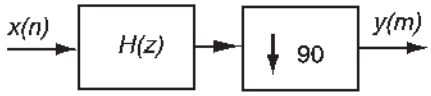
From Equation 3.37, the lowpass FIR filter $H(z)$ has a length of $N \approx 5424$. Therefore, the number of multiplications per second, M_{sec}, needed for this single-stage decimator is

$$M_{sec} = 5424 * \frac{90000}{90} = 5,424,000. \tag{3.38}$$

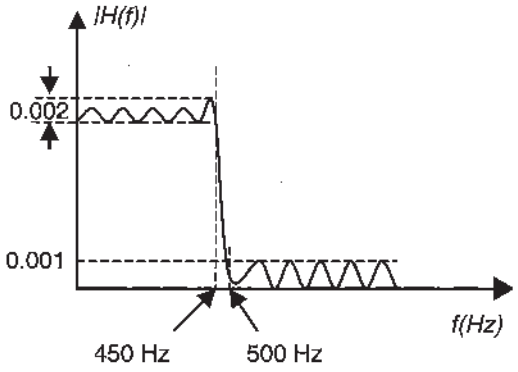
Since only one out of ninety samples is actually used, the computation rate is based on the decimated signal rate.

Decimation in Two Stages

Let us now consider the two-stage implementation of the decimation process as shown in Figure 3.26.



(a)



(b)

Figure 3.25: (a) Block Diagram for Single-Stage Decimation, (b) The Filter Specification.

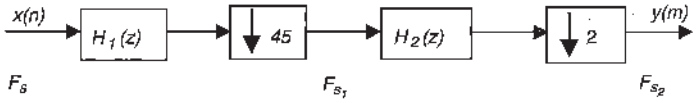


Figure 3.26: Block Diagram for Multi-Stage Decimation.

Due to the cascade decomposition, each of the two filters, $H_1(z)$ and $H_2(z)$, must have a linear passband ripple specification half of that specified for the single-stage filter, $H(z)$. The stopband ripple specifications for these two filters can be the same as that of $H(z)$ since the cascade connection will only reduce the stopband ripple.

Stage One

The first stage will decimate the input signal $x(n)$ by a factor of forty-five. The filter specifications for the first-stage LPF $H_1(z)$ are

$$\begin{aligned}
\delta_{p_1} &= 0.001 = (0.002/2), \\
\delta_{s_1} &= 0.001, \\
f_{p_1} &= 450 \text{ Hz}, \\
f_{s_1} &= 2000 - 500 = 1500 \text{ Hz},
\end{aligned}$$

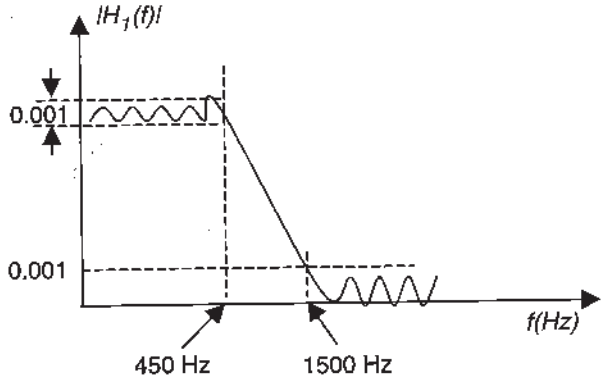


Figure 3.27: Decimation Filter Design for Stage One.

$$F_s = 90 \text{ kHz, and}$$
$$F_{s_1} = \frac{90{,}000}{45} = 2{,}000 \text{ Hz.}$$

These specifications are shown in Figure 3.27.

The reason for choosing this value of the stopband edge is that, after decimation by a factor of forty-five, the residual energy of the signal in the band from 1000 to 2000 Hz will be aliased back to the band from 0 to 1000 Hz. Due to the attenuation in the stopband, the energy of the signal in the band from 1500 to 2000 Hz is very small compared to that in 1000 to 1500 Hz. So the amount of aliasing in the desired band of interest (0 to 450 Hz) will also be small, resulting in very little signal distortion.

According to Equation 3.37, the approximate length of the FIR filter, $H_1(z)$ is $N_1 = 276$. The number of multiplications per second for the first stage is

$$M_{1,sec} = N_1 \frac{F_s}{D_1} = 276 * \frac{90000}{45} = 552{,}000. \tag{3.39}$$

Stage Two

The specifications for the second-stage filter, $H_2(z)$, are

$$\delta_{p_1} = 0.001 = (0.002/2),$$
$$\delta_{s_1} = 0.001,$$
$$f_{p_2} = 450 \text{ Hz},$$
$$f_{s_2} = 500 \text{ Hz, and}$$
$$F_{s_1} = 2000 \text{ Hz.}$$

Figure 3.28 shows the characteristics of $H_2(z)$. This stage will perform a decimation of factor two on the output signal of the first stage. So, the total decimation of $x(n)$ is by a factor of ninety as required.

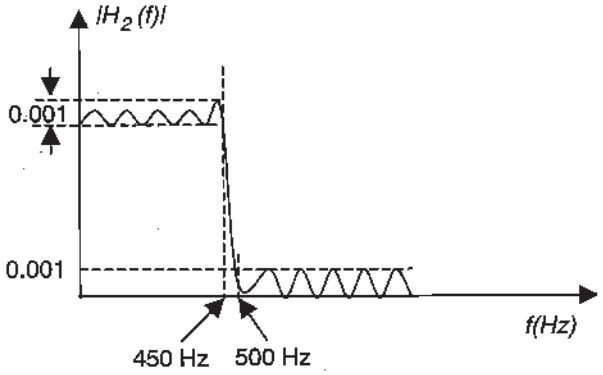


Figure 3.28: Decimation Filter Design for Stage Two.

For the second stage, the length of the filter, as calculated from Equation 3.37, is $N_2 = 129$. The number of multiplications required for this stage is

$$M_{2,sec} = 129 * \frac{2000}{2} = 129,000. \tag{3.40}$$

The total number of multiplications per second required for the two-stage implementation of the decimator is

$$M_{sec} = M_{1,sec} + M_{2,sec} = 552,000 + 129,000 = 681,000 \tag{3.41}$$

So, the two-stage implementation requires only $\frac{681,000}{5,424,000} = \frac{1}{8}$ of the operation required of the single-stage implementation.

Decimation in Three Stages

To further illustrate the concept of multi-stage implementation of decimator and interpolator, we will now consider the three-stage implementation as shown in Figure 3.29.

Stage One

In this stage, decimation by fifteen is performed on the input signal $x(n)$. The characteristics of the LPF, $H_1(z)$, are shown in Figure 3.30. The filter specifications are

$$\begin{aligned}
&\delta_{p_1} = 0.00067 = (0.002/3),\\
&\delta_{s_1} = 0.001,\\
&f_{p_1} = 450 \text{ Hz},\\
&f_{s_1} = 6000 - 500 = 5500 \text{ Hz},\\
&F_s = 90 \text{ kHz, and}\\
&F_{s_1} = \tfrac{90,000}{15} = 6,000 \text{ Hz}.
\end{aligned}$$

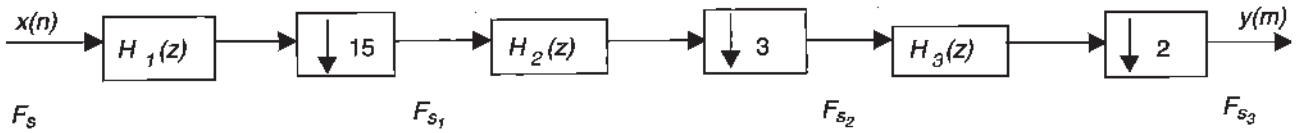


Figure 3.29: Block Diagram for Decimation in Three Stages.

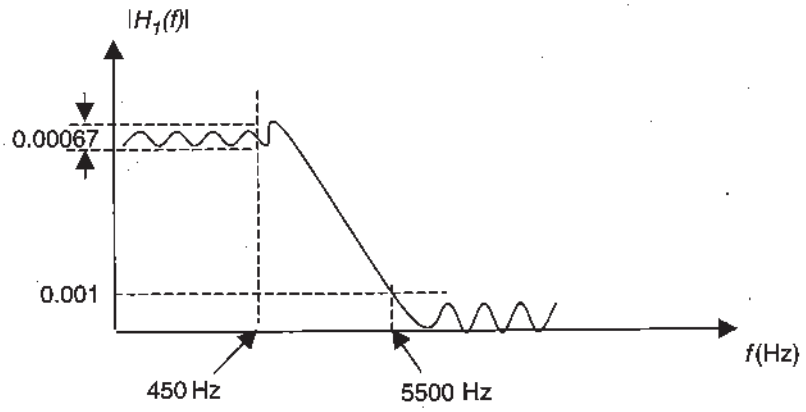


Figure 3.30: Decimation Filter Design for Stage One.

As in the two-stage case, the choice of stopband edge frequency can be extended to the point for which negligible aliasing occurs in the passband (band of interest).

The approximate length of the filter as given by Equation 3.37 is $N_1 = 60$. The number of multiplications per second for this stage is calculated as

$$M_{1,sec} = N_1 \frac{F_s}{D_1} = 60 * \frac{90000}{15} = 36,000. \tag{3.42}$$

Stage Two

In this stage, a decimation by a factor of three is done. The specifications of the LPF in this stage, $H_2(z)$, are

$$\begin{aligned}
&\delta_{p_2} = 0.00067 = (0.002/3),\\
&\delta_{s_2} = 0.001,\\
&f_{p_2} = 450 \text{ Hz},\\
&f_{s_2} = 2000 - 500 = 1500 \text{ Hz},\\
&F_{s_1} = 6000 \text{ Hz, and}\\
&F_{s_2} = \tfrac{6000}{3} = 2000 \text{ Hz}.
\end{aligned}$$

As before, the stopband edge frequency can be stretched out to 1500 Hz. The filter characteristics are shown in Figure 3.31.

The length of filter required for this stage is $N_2 = 20$ and the number of multiplications per second is

$$M_{2,sec} = N_2 \frac{F_{s1}}{D_2} = 20 * \frac{6000}{3} = 40,000. \tag{3.43}$$

Stage Three

The third stage performs a decimation of factor two on the output of the second stage. The specifications of the LPF, $H_3(z)$, in this stage are

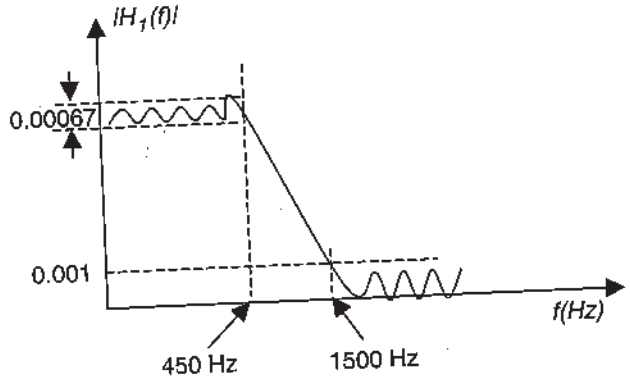


Figure 3.31: Decimation Filter Design for Stage Two.

$$\delta_{p_3} = 0.00067 = (0.002/3),$$
$$\delta_{s_3} = 0.001,$$
$$f_{p_3} = 450 \text{ Hz},$$
$$f_{s_3} = 500 \text{ Hz},$$
$$F_{s_2} = 2000 \text{ Hz, and}$$
$$F_{s_3} = N_3 \frac{F_{s_2}}{D_3}.$$

Figure 3.32 shows the specifications for $H_3(z)$. As before, the approximate length of the filter, as calculated from Equation 3.37, is $N_3 = 134$. The number of multiplications required per second in the third stage is

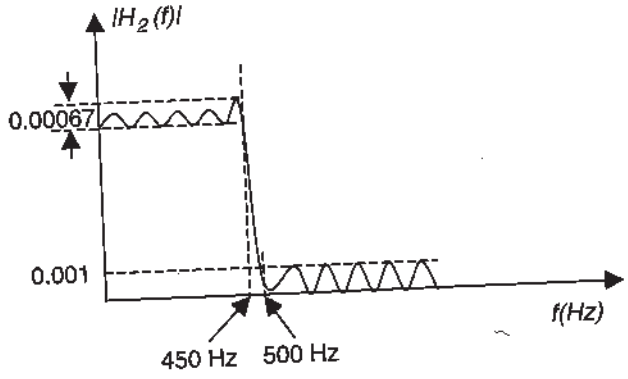


Figure 3.32: Decimation Filter Design for Stage Three.

$$M_{3,sec} = 134 * \frac{2000}{2} = 134,000. \tag{3.44}$$

The total number of multiplications in the three stages of implementation is

$$M_{sec} = M_{1,sec} + M_{2,sec} + M_{3,sec} = 360,000 + 40,000 + 134,000 = 534,000. \tag{3.45}$$

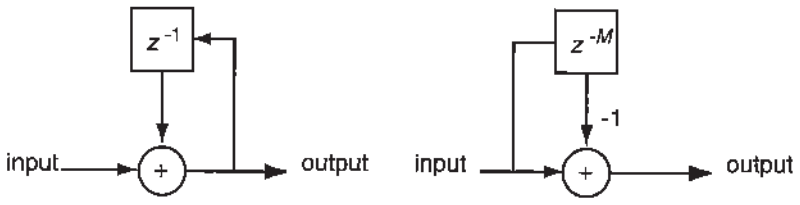
Compared to the single-stage implementation, the number of multiplications per second are reduced by a factor of $\frac{5,424,000}{534,000} = 10$ by using three stages.

From this example, we can see that a significant saving in computation as well as in storage can be achieved by a multi-stage decimator and interpolator design. These savings depend on the optimum design of the number of stages and the choice of decimation factor for the individual stages. ■

The examples illustrate the many different combinations and ordering possible. One approach is to determine the sets of I and D factors that satisfy the filtering requirements and then estimate the storage and computational costs for each set. The lowest cost solution is then selected [37].

3.2.6 Cascaded Integrator Comb Filter

In software radio systems, sample rate changes can be very large, with changes from many tens of MHz to around 100 kHz being common. Of course, such a requirement leads to large order and high-rate digital filters, which can easily become a bottleneck in the overall system design. A cascaded integrator comb (CIC) filter[4] can be used to reduce the computational demands. A CIC filter is what the name indicates: a cascade of simple integrators (accumulators) and a cascade of comb filters (delay and subtract from current sample). These basic building blocks are shown in Figure 3.33. The CIC filter can implement an interpolation or decimation filter (see Figure 3.34) that uses only delay and add operations and thus is well-suited for FPGA and ASIC implementation. Furthermore, the same basic filter structure can be used to handle variable sample rate conversion.



(a) Simple Integrator Stage. (b) Simple Comb Filter Stage.

Figure 3.33: Building Blocks of a CIC Filter.

[4]This section is based on material from [38–40].

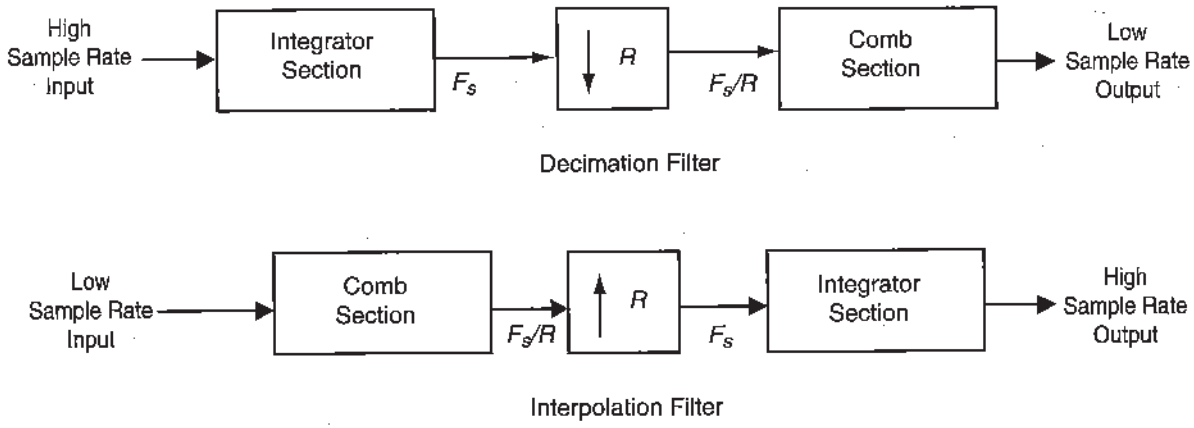


Figure 3.34: Decimation and Interpolation Using Comb and Integrator Filters.

Basic Structures

CIC Decimation Filter

The CIC implementation of a decimation filter is the cascade of an integrator stage, a decimation procedure, and a comb stage as shown in Figure 3.35. To analyze the CIC filter's response, combining the integrator and comb stages into a single transfer function are important to reduce the complexity of the analysis. However, to reduce the computational expense of the operation, the implementation of the filter is performed in two separate sections, before and after the decimation.

The integrator section is the cascade of N ideal digital integrator stages operating at the high sampling rate of F_s, where the transfer function is defined as

$$H_I(z) \equiv \left(\frac{1}{1-z^{-1}}\right)^N. \tag{3.46}$$

The comb sections consist of N comb stages operating at the low sampling rate of $\frac{F_s}{R}$, where R is the integer-rate change factor. The comb sections have a differential delay of M samples per stage, where the system function is defined as

$$H_C(z) \equiv \left(1-z^{-M}\right)^N \tag{3.47}$$

and M is typically chosen as 1 or 2.

The cascade of the integrator and comb sections of the CIC filter requires a decimation process to be implemented between these two filters. It is possible to apply the Noble Identities to switch the positions of the comb stages and the decimation procedure (see Section 3.2.3) to simplify the analysis. Figure 3.36 shows a block diagram of this change. The order of the comb filter and the decimation procedure can be switched without causing any change in the end results of the filtering operation.

The integrator and comb stages of the filter can be combined into a single transfer function, thus simplifying analysis. The transfer function of the cascaded integrator comb filter before decimation is then

$$H(z) = \left(\frac{1-z^{-RM}}{1-z^{-1}}\right)^N = \left(\sum_{k=0}^{RM-1} z^{-k}\right)^N \tag{3.48}$$

and the frequency response (with respect to the higher input sample rate) is

$$|H(\omega)| = \left[\frac{\sin\left(\frac{\omega RM}{2}\right)}{\sin\left(\frac{\omega}{2}\right)}\right]^N. \tag{3.49}$$

This filter is a cascade of N copies of an RMth-length FIR filter whose coefficients specify a rectangular time-domain window, so it is indeed an LPF. Furthermore, because of the symmetry, this is a linear phase filter. Therefore, the CIC decimation filter is actually an alternative implementation of a general decimation filter. The key is that, in this CIC

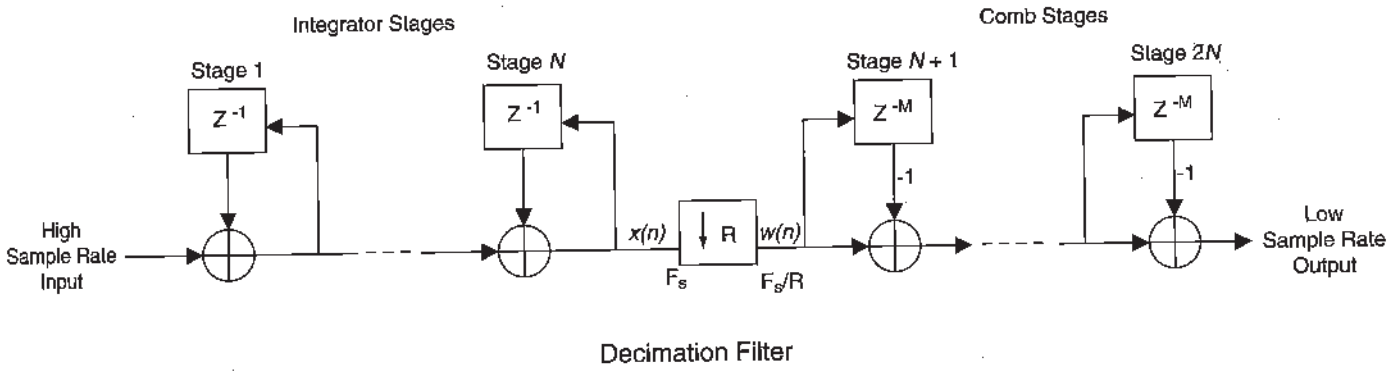


Figure 3.35: CIC Decimation Filter.

SOURCE: E. B. Hogenauer, "An Economical Class of Digital Filters for Decimation and Interpolation," [38]. © IEEE, 1981. Used by Permission.

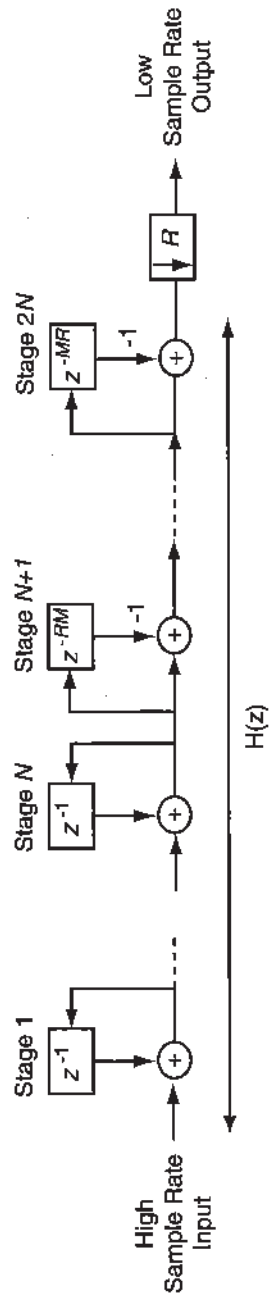


Figure 3.36: Equivalent CIC Structure for Analysis Purposes.

structure, the comb stage operates at the low sampling rate, resulting in a more efficient implementation of the decimation filters.

Note that the frequency response $H(\omega)$ is with respect to the original sample rate. By letting $\omega' = \omega R$, the frequency response with respect to the decimated sample rate is found.

$$|H(\omega')| = \left| \frac{\sin \pi M \omega'/2}{\sin \frac{\pi \omega'}{2R}} \right|^N \tag{3.50}$$

Note that a null in the frequency response occurs at $1/M$ for both the decimated and non-decimated filters. Furthermore, if R is large, then $H(\omega')$ can be approximated as

$$|H(\omega')| \approx \left| RM \frac{\sin \pi M \omega'/2}{M \pi \omega'/2} \right|^N . \tag{3.51}$$

The gain of the filter is $(RM)^N$, which can become very large, necessitating the need for large registers.

As an example, the frequency response of Equation 3.48 is plotted in Figure 3.37 for $N = 4, M = 1, R = 7$ for a cutoff frequency $f_c = 1/8$. The input sample rate is 7 and

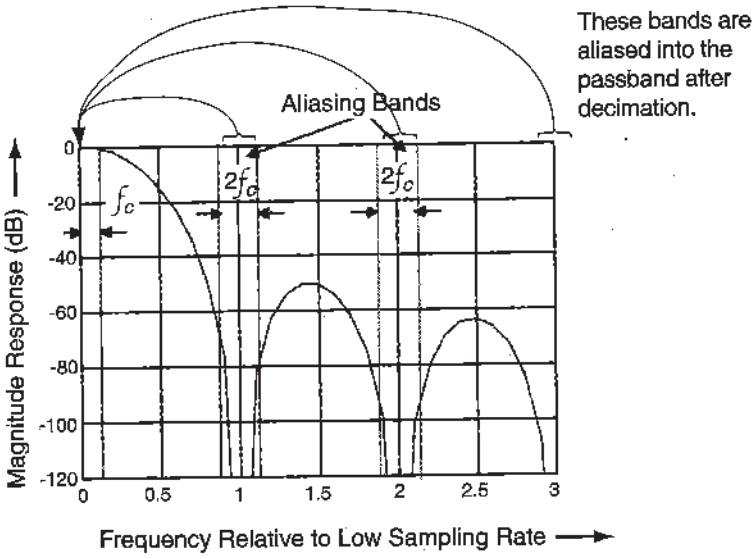


Figure 3.37: Frequency Response of the CIC Filter.

SOURCE: E. B. Hogenauer, "An Economical Class of Digital Filters for Decimation and Interpolation," [38]. © IEEE, 1981. Used by Permission.

the output sample rate is 1. Note the bands at multiples of 1 Hz will be aliased into the passband region about 0 Hz. At multiples of 1 Hz, there are nulls in the frequency response and thus aliasing will be minimal in the passband. The filter is designed to attentuate the signal so that there are acceptable levels of aliasing at the passband edge, f_c. There is, of course, a droop in the passband, but this droop can be compensated for by a short FIR filter at the output of the CIC filter operating at the reduced sample rate or using the sharpening technique proposed by Kwentus [40].

CIC Interpolation Filter

The structure of the interpolation CIC filter, as seen Figure 3.38, uses a comb stage followed by an upsampler and an integrator stage. Using the Noble Identities, the CIC interpolator can be analyzed using the equivalent structure shown in Figure 3.39.

Obviously Figure 3.38 has a more attractive implementation since comb filter stages operate at a lower rate. The analysis of the interpolator CIC filter follows that of the decimator CIC filter. One difference here is that the purpose of the CIC filter for the interpolator is to suppress replicated images that occur due to zero-insertion, while the CIC filter for the decimator is used to suppress aliasing.

Economics of CIC Filters

The CIC filter presents several advantages over more basic implementations of decimation and interpolation filters:

- no multipliers are required,
- no storage is required for filter coefficients,
- the structure of CIC filters is very regular, consisting of two simple building blocks,
- very little external or complicated control is needed.

Two primary problems are encountered in the implementation of CIC filters. First, the register widths can become large, especially for large values of R. Secondly, the frequency response is fully determined by only three integer parameters, M, N, and R, resulting in a limited range of filter configurations [38–40]. While a detailed description of this register growth problem is beyond the scope of this discussion, Hogenauer [38] provides details that show the number of output bits B_{out} is given by

$$B_{out} = B_{in} + N \log_2 RM \tag{3.52}$$

where B_{in} is the number of input bits. Strategies exist for truncating and rounding the number of bits throughout the filtering process to reduce the number of bits [38].

Given the advantages and disadvantages of using CIC filters, they are ideal for applications in which high sampling rates make the use of multipliers a computationally expensive option. This technique is especially useful for FPGA design where multipliers are avoided because of the large silicon area required. The use of CIC filters is also important in applications in which large rate change factors require large amounts of coefficient storage or fast impulse response generation and the memory is either unavailable or too slow to perform the desired application.

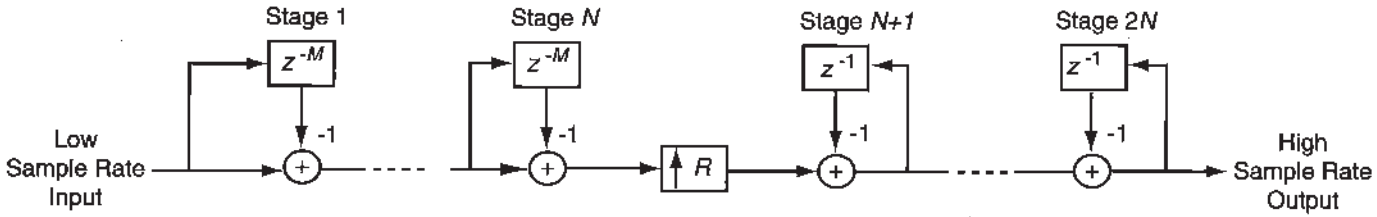


Figure 3.38: CIC Implementation of an Interpolation Filter.

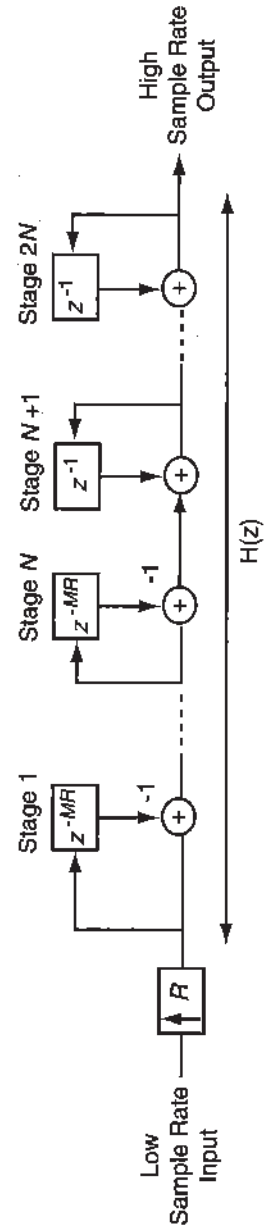


Figure 3.39: Equivalent CIC Implementation for Purposes of Analysis.

3.3 Polyphase Filters

One of the principal benefits of the implementation of multirate systems is the large reduction in computational cost [35]. This cost reduction may be achieved by using inexpensive, low-speed processing elements in place of costly, high-speed elements. Polyphase filters allow this trade-off. These polyphase filter structures are derived in the following subsections.

3.3.1 Polyphase Decimation

From Section 3.2.1, the time domain relationship of the decimation process is expressed as

$$y(m) = \sum_{l=-\infty}^{\infty} h_D(l)x(mD-l). \tag{3.53}$$

Let

$$l = iD + k.$$

Then

$$y(m) = \sum_{k=0}^{D-1} \sum_{i=0}^{\infty} h_D(iD+k)\, x\left((m-i)\,D-k\right).$$

Define

$$p_k(i) = h_D(iD+k) \tag{3.54}$$

$$x_k(i) = x(iD-k). \tag{3.55}$$

Note that $p_k(i)$ comes from decimated versions of $h_D(m)$ that are time-shifted by k. $x_k(i)$ has a similar interpretation. Thus,

$$y(m) = \sum_{k=0}^{D-1} \sum_{i=-\infty}^{\infty} p_k(i)x_k(m-i)$$

$$y(m) = \sum_{k=0}^{D-1} p_k(m) * x_k(m). \tag{3.56}$$

Hence, the decimation operation can be decomposed into a sum of D parallel filtering stages where the filters $p_k(m)$ are formed by decimating $h_D(m+k)$ by a factor D. Furthermore, $x_k(m)$ has a similar interpretation. A realization of Equation 3.56 is shown in Figure 3.40, and, with some thought, Figure 3.40 can be transformed into Figure 3.41.

Figure 3.41 does have an intuitive interpretation. Examining Figure 3.5, it is apparent that as a sample progresses through $h(n)$, only a subset of the filter coefficients (one out of D) will engage the sample to lead to an output value. The set of coefficients that engage that sample represents a single branch in Figure 3.41.

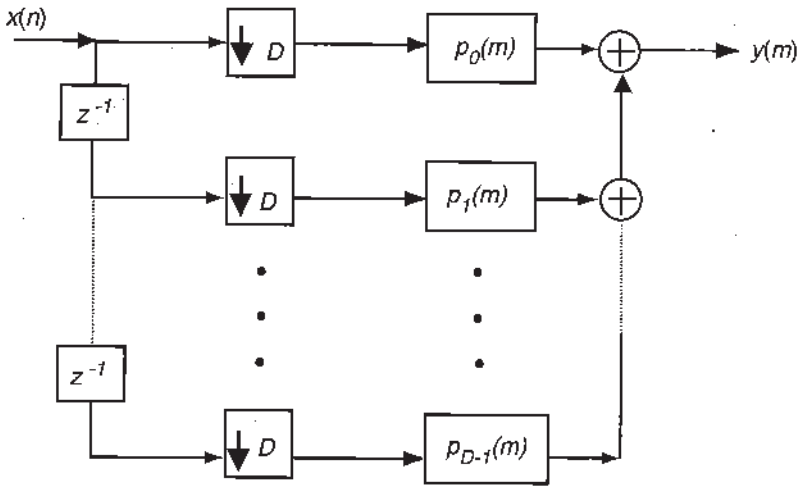


Figure 3.40: General Structure of a Polyphase Decimator.

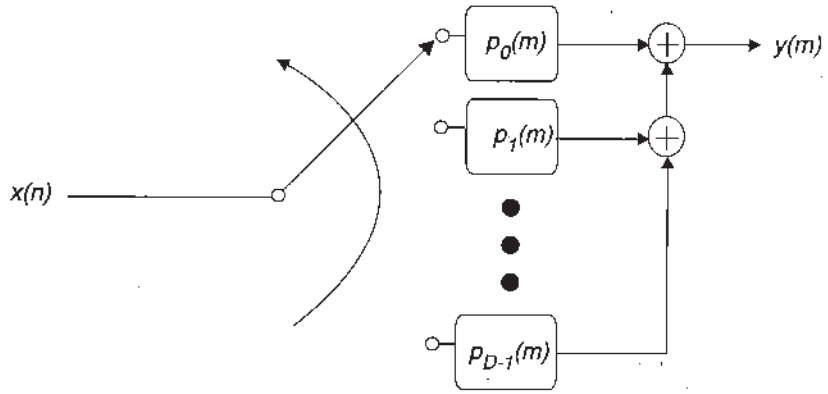


Figure 3.41: Equivalent Commutator Model of the Polyphase Decimator.

97

3.3.2 Polyphase Interpolation

The derivation of the polyphase interpolator follows closely the derivation of the polyphase decimator [35]. We know from Figure 3.8 that the upsampling operation can be expressed as

$$y(m) = v(m) * h_I(m) \tag{3.57}$$

where $v(m)$ is given by Equation 3.14. We will assume for this analysis that $h_I(m)$ is a finite impulse response (FIR) filter with length N and that $N = P \cdot I$. Thus,

$$y(m) = \sum_{l=0}^{N-1} v(m-l)h_I(l). \tag{3.58}$$

Let

$$l = iI + k. \tag{3.59}$$

Then

$$y(m) = \sum_{k=0}^{I-1} \sum_{i=0}^{P-1} v(m - iI - k)h_I(iI + k). \tag{3.60}$$

From Equation 3.14,

$$v(m - iI - k) = \begin{cases} x(n-i) & m = nI + k \\ 0 & \text{otherwise.} \end{cases} \tag{3.61}$$

Thus, only one index of k results in a non-zero term.

$$y(m) = \sum_{k=0}^{I-1} \sum_{i=0}^{P-1} x(n-i)p_k(i) \tag{3.62}$$

where again

$$p_k(i) = h_I(iI + k) \tag{3.63}$$

or

$$y(m) = \sum_{k=0}^{I-1} x(n) * p_k(n) \tag{3.64}$$

or

$$y(nI + k) = x(n) * p_k(n). \tag{3.65}$$

Equation 3.64 can be realized using the structure shown in Figure 3.42, which can easily be shown to be equivalent to Figure 3.43.

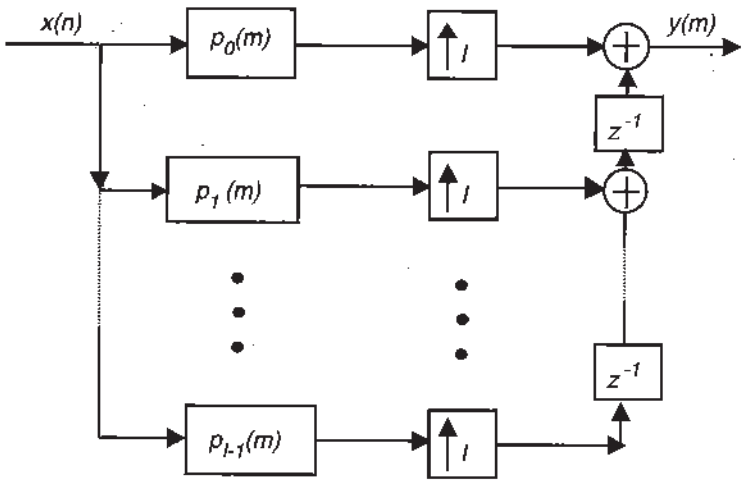


Figure 3.42: General Structure of a Polyphase Interpolator.

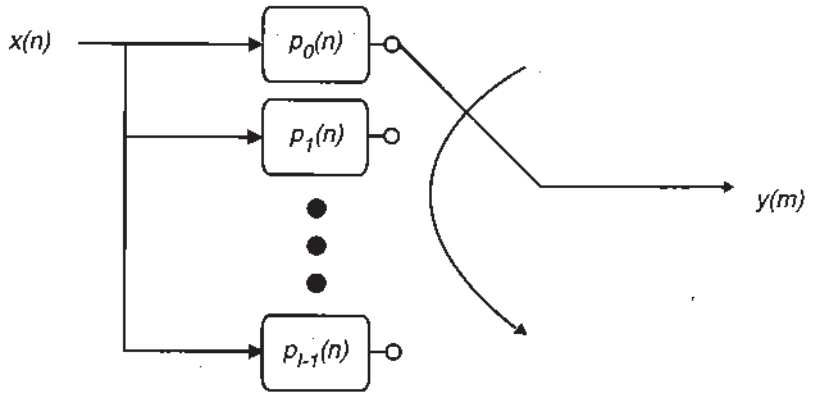


Figure 3.43: Equivalent Commutator Model of the Polyphase Interpolator.

In Figure 3.42, for each new sample $x(n)$, the interpolator gives I output samples of $y(m)$. The upper path ($k = 0$) contributes the samples of $y(m)$ for $m = 0, \pm I, \pm 2I....$ The next path ($k = 1$) contributes for $m = 1, \pm(I+1), \pm(2I+1)....$ The last path ($k = I-1$) contributes to $y(m)$ for $m = (I-1), \pm(2I-1), \pm(3I-1)....$ So for each input sample, $x(n)$, each branch contributes one of the I outputs of the interpolation. If the length of the original interpolation filter is N, then each of the I filters polyphase representation has an integer length of $\frac{N}{I}$.

The I branches of the network in Figure 3.42 contribute I sample values for I different time slots so that the upsampler and the delays can be replaced by a commutator as shown in Figure 3.43. Starting from the branch $k = 0$ at time $m = 0$, the commutator rotates counterclockwise, sweeping through the I branches of the polyphase network for each input sample $x(n)$. The computational load would then decrease significantly by avoiding multiplications and additions with zero compared to a non-polyphase structure. The filter implementation seen in Figure 3.43 performs the same function as a standard implementation of an FIR filter, but no operations are performed on the zero coefficients of the input vector.

Example: Pulse-Shaping

Pulse-shaping is used to minimize both ISI and bandwidth. The Nyquist criterion places conditions on the pulse-shaping filter to ensure that the ISI at the decision sampling instance is zero. Special filters, such as raised-cosine filters and Gaussian filters, are used to satisfy the Nyquist criterion. Furthermore, the pulse-shaping filter should have a narrow bandwidth and a sharp roll-off to maximize spectral efficiency of the communication system.

The ISI can be eliminated by using certain special pulse shapes. The magnitude of the impulse response of the pulse-shaping filter should be zero at multiples of the symbol interval, T_{symbol} as shown in Figure 3.44. The pulse-shaping filter can have any non-zero value between the symbol centers. Thus even if the pulse shapes are infinitely long, there will be no interference from other symbols if they satisfy the Nyquist criteria at the instant at which the symbol is sampled. DECT and GSM standards use Gaussian pulse-shaping, but raised-cosine is the most popular pulse-shaping. ■

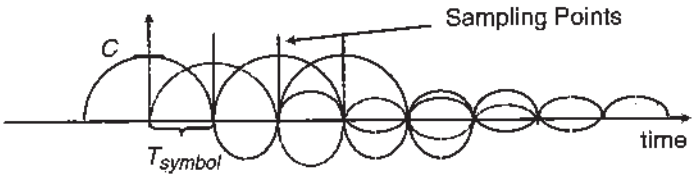


Figure 3.44: Pulse Shapes Satisfying the Nyquist Criteria

Example: Raised-Cosine Filter

The continuous time raised-cosine filter satisfies the Nyquist criteria for eliminating ISI. The impulse response and frequency response of the raised-cosine pulse-shaping filter is given by

$$h_e(t) = 2f_0 \left(\frac{\sin(2\pi f_0 t)}{2\pi f_0 t} \right) \left[\frac{\cos(2\pi f_\Delta t)}{1-(4f_\Delta t)^2} \right] \tag{3.66}$$

and

$$\begin{aligned} H_e(f) &= 1, && |f| < f_1 \\ &= 1/2 \left(1 + \cos\left(\tfrac{\pi(|f|-f_1)}{2f_\Delta}\right)\right), && f_1 \le |f| \le B \\ &= 0, && |f| > B \end{aligned}$$

where B is the absolute bandwidth, f_0 is the 6 dB bandwidth, $f_\Delta = B - f_0$, $f_1 = f_0 - f_\Delta$, $r = f_\Delta / f_0$. The value r is defined as the roll-off factor and determines the width of the transition band in the frequency spectrum. If $r = 0$, the pulse becomes rectangular in the frequency domain. The ideal raised-cosine pulse is of infinite duration in the time domain. However, in practical applications, it is truncated to a desired length, which results in side lobes in the frequency spectrum. A thirty-two-tap raised-cosine filter for $r = 0.33$ is created as shown in Figure 3.45 by sampling the continuous time impulse response.

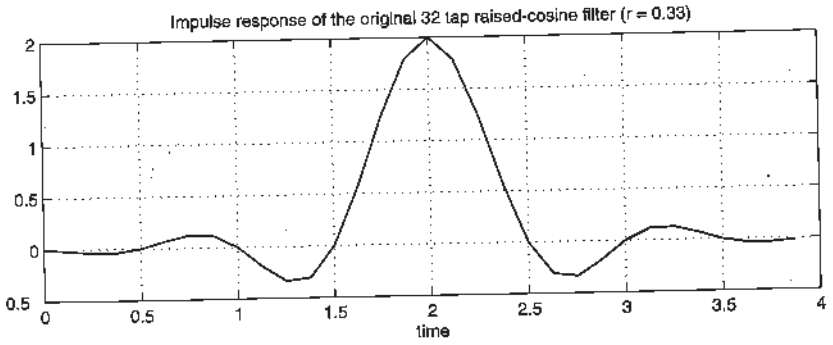
This filter has to be implemented in the polyphase form for decimating by a factor of four. The coefficients of each of the filters $p_0(i)$, $p_1(i)$, $p_2(i)$, and $p_3(i)$ are given by Equation 3.63 through sampling of 3.66 and are tabulated in Table 3.1. ■

3.4 Digital Filter Banks

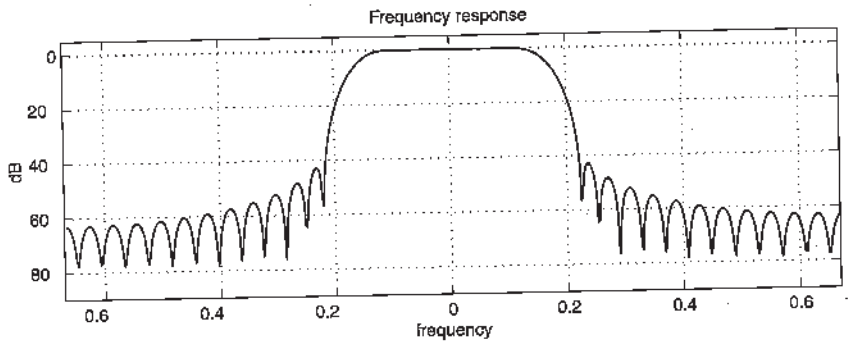
Polyphase decomposition of decimation and interpolation filters, as explained in Section 3.3, suggests a way to reduce the computational complexity in the implementation of multirate techniques. A more intuitive approach, however, is the application of digital filter banks[5] in breaking a wideband signal into multiple bands (channels). Each of these bands can be processed individually, thereby allowing lower-speed computational techniques to be applied to each of the channels. This partitioning of a signal using a digital filter bank can be applied to other applications apart from the reduction of computational complexity, such as the conversion of a signal from frequency division multiplexing (FDM) to time division multiplexing (TDM) or vice versa.

In the following discussion, we assume a uniform filter bank, i.e., all the channels have the same bandwidth and sampling rates. The channel outputs are also assumed to be critically sampled, meaning that the total number of samples per unit time from all the channel outputs together is the same as the number of samples per unit time of the input signal. We consider the DFT as a filter bank, and, although it is not very good in terms of stopband responses, it can be improved considerably as shown in Section 3.4.2.

[5]Material for this section is based on [31,34,41].



(a)



(b)

Figure 3.45: (a) Impulse Response and (b) Frequency Response of a Raised-Cosine Filter.

i	$p_0(i)$	$p_1(i)$	$p_2(i)$	$p_3(i)$
0	-0.0001	-0.0174	-0.0372	-0.0373
1	0.0005	0.0689	0.1263	0.1141
2	-0.0013	-0.1872	-0.3357	-0.304
3	0.0036	0.5718	1.2464	1.7924
4	2.0000	1.7859	1.2359	0.5613
5	-0.0036	-0.3073	-0.3343	-0.1841
6	0.0013	0.1151	0.1258	0.0678
7	-0.0005	-0.0376	-0.0370	-0.0170

Table 3.1: Filter Coefficients for Polyphase Raised-Cosine Filter.

102

3.4.1 Implementation

The implementation of a digital filter bank follows two basic architectures: an analyzer for decomposing the input signal into several channels and a synthesizer for reconstructing the input by combining the channel signals. Figure 3.46 shows a basic block diagram of both of these architectures.

In a filter bank analyzer, the input signal is divided into a set of N channels using complex bandpass filters. The center frequencies of these bandpass filters $H_k(\omega - \omega_k)$ are equally spaced with center frequencies $\omega_k = \frac{2\pi k}{N}, k = 0, 1, \ldots, N-1$.[6] The outputs of these filters can now be decimated by a factor of N. An equivalent procedure is to downconvert (frequency shift) by ω_k, lowpass filter, and decimate by N. Understanding the equivalence of these channelization approaches is an important concept.

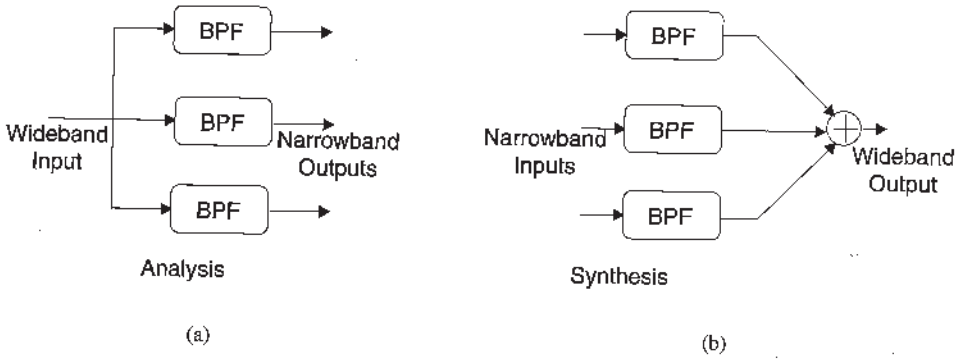


Figure 3.46: (a) Digital Filter Bank Analysis, (b) Synthesis Architecture.

3.4.2 DFT Filter Banks[7]

Basic Forms

The DFT is a well-known approach for splitting a signal into several bands. Figure 3.47 shows a basic N-channel DFT filter bank analyzer. The input signal, $x(n)$, is first multiplied by the complex sinusoids $e^{-j\omega_k n}$, $k = 0, 1, 2, \ldots, N-1$, resulting in N complex signals. These modulated signals are then lowpass filtered by $h(n)$ and downsampled by a factor of N (critically sampled), and thereby effectively converting them to baseband signals. The channel outputs, $X_k(m)$, can be expressed as

$$X_k(m) = \sum_{l=-\infty}^{\infty} h_0(mD - l)x(l)e^{-j2\pi\frac{k}{N}l}, \quad k = 0, 1, 2, \ldots, N-1, \tag{3.67}$$

[6] $H_0(\omega)$ is defined as an LPF.

[7] Material in this section follows the development in [31,34].

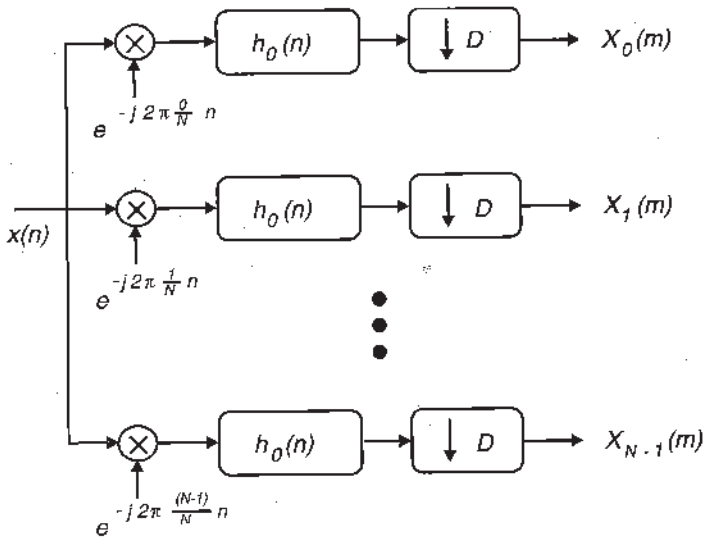


Figure 3.47: Basic Form of a DFT Filter Bank Analyzer (when $D = N$ system is critcally sampled).

and when critically sampled ($D = N$), $X_k(m)$ is

$$X_k(m) = \sum_{l=-\infty}^{\infty} h_0(mN - l)x(l)e^{-j2\pi\frac{k}{N}l}. \tag{3.68}$$

Recalling that the N-point DFT of a signal $x(n)$ has the form

$$X_k = \sum_{n=0}^{N-1} x(n)e^{-j2\pi\frac{k}{N}n} \quad , k = 0, 1, 2, ..., N-1, \tag{3.69}$$

for practical purposes, we can assume that the filter $h(n)$ has a finite impulse response of length N. With this, we can conclude that Equation 3.67 expresses the N-point DFT of the sequence $h(mN - l)x(l)$ for each value of m.

In the DFT filter bank synthesizer, as shown in Figure 3.48, all the N channel signals are interpolated (upsampled and lowpass filtered) first by a factor of N that restores them to their original sampling rate. Finally, all the channel signals are upconverted and summed to get the synthesizer output.

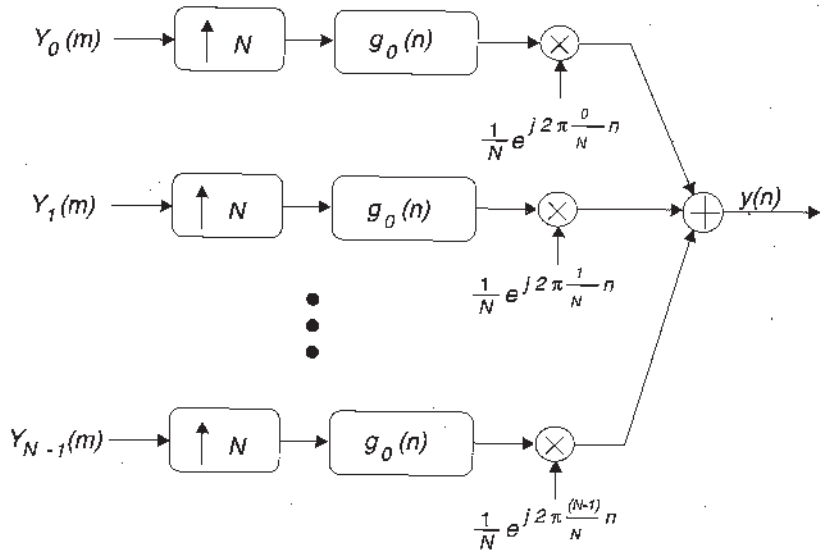


Figure 3.48: Basic Form of a DFT Filter Bank Synthesizer.

Mathematically, each of the channel signals can be expressed as

$$\begin{aligned} y_k(n) &= \frac{1}{N}\left[[Y_k(m)]_{N\uparrow} * g_0(n)\right] e^{j2\pi\frac{n}{N}k} \\ &= \frac{1}{N}\sum_{l=-\infty}^{\infty} Y_k(l)g_0(n-lN)e^{j2\pi\frac{n}{N}k}\ , k = 0, 1, 2, ..., N-1. \end{aligned} \tag{3.70}$$

The operation $[Y_k(m)]_{N\uparrow}$ represents placing $N-1$ zeros between samples in $Y_k(m)$.
From the definition of the IDFT,

$$y_k(n) = \frac{1}{N}\sum_{k=0}^{N-1} Y_k(n)e^{j2\pi\frac{k}{N}n}\ , n = 0, 1, 2, ..., N-1. \tag{3.71}$$

The output of the synthesizer is the sum of all the channel signals as given below.

$$\begin{aligned} y(n) &= \sum_{k=0}^{N-1} y_k(n) \\ &= \sum_{l=-\infty}^{\infty} g_0(n-lN)\left[\frac{1}{N}\sum_{k=0}^{N-1} Y_k(l)e^{j2\pi\frac{n}{N}k}\right]. \end{aligned} \tag{3.72}$$

Equation 3.72 says that synthesis is achievable by taking the IDFT across channels and interpolating each resulting time sample with the LPF $g_0(n)$.

Alternative Architectures

An alternative way of implementing the DFT filter bank can be developed as follows.[8] From Section 3.4.2, the basic DFT filter bank analyzer output is given by

$$X_k(m) = \sum_{l=-\infty}^{\infty} h_0(mD-l)x(l)e^{-j2\pi\frac{k}{N}l} \quad , k = 0, 1, 2, ..., N-1. \tag{3.73}$$

This time it is assumed that the signal is not critically sampled, i.e., $D \neq N$. Equation 3.73 can then be written as

$$\begin{aligned} X_k(m) &= \left[\sum_{l=-\infty}^{\infty} \left(h_0(mD-l)e^{j2\pi\frac{k}{N}(mD-l)}\right) x(l)\right] e^{-j2\pi\frac{k}{N}mD} \\ &= \left[\sum_{l=-\infty}^{\infty} h'_k(mD-l)x(l)\right] e^{-j2\pi\frac{k}{N}mD} \\ &= \left[h'_k(m) * x(m)\right]_{D\downarrow} e^{-j2\pi\frac{k}{N}mD} \quad , k = 0, 1, 2, ..., N-1 \end{aligned} \tag{3.74}$$

where

$$h'_k(n) = h_0(n)e^{j2\pi\frac{k}{N}n} \quad , k = 0, 1, 2, ..., N-1 \tag{3.75}$$

is a complex BPF centered at the frequency, $\omega_k = \frac{2\pi k}{N}$, $k = 0, 1, 2, ..., N-1$ and, as noted before, $[\ \ ]_{D\downarrow}$ is the operation of taking 1 of every D samples. Figure 3.49 is the alternate structure for the analyzer. In each channel, the signal is bandpass filtered by $h'_k(n)$, downsampled by a factor of D, and then downconverted by the complex sinusoid $e^{-j2\pi\frac{k}{N}mD}$, $k = 0, 1, 2, ..., N-1$. In the case of critical sampling, i.e., $D = N$, Equation 3.74 becomes

$$X_k(m) = \sum_{l=-\infty}^{\infty} h'_k(mN-l)x(l) \tag{3.76}$$

$$= \left[h'_k(m) * x(m)\right]_{N\downarrow}. \tag{3.77}$$

Similarly, an alternative structure can be created for the synthesizer. In the basic DFT filter bank synthesizer, each channel signal as given by Equation 3.70 is (assuming no critical sampling)

$$y_k(n) = \frac{1}{N}\sum_{l=-\infty}^{\infty} Y_k(l)g_0(n-lD)e^{j2\pi\frac{n}{N}k} \quad , k = 0, 1, 2, ..., N-1. \tag{3.78}$$

[8] Material for this section follows the development in [31, 34].

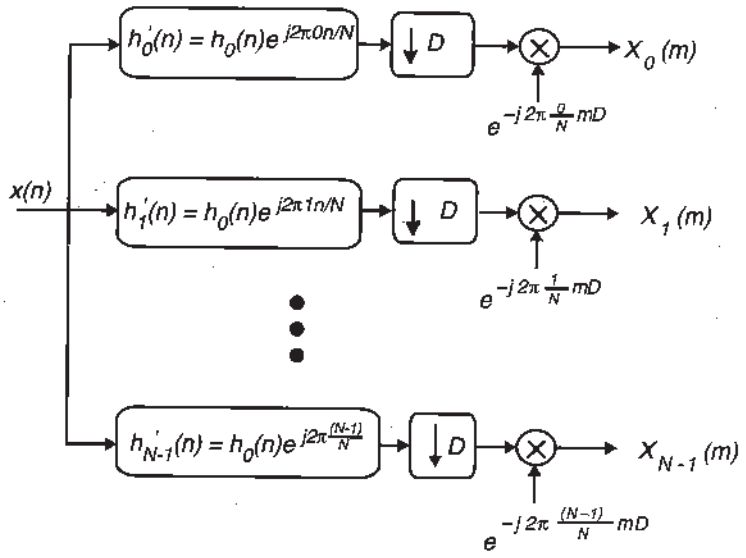


Figure 3.49: Alternate Implementation of a DFT Filter Bank Analyzer.

Another way of expressing Equation 3.78 is

$$\begin{aligned} y_k(n) &= \frac{1}{N}\left[\sum_{l=-\infty}^{\infty} Y_k(l)g_0(n-lD)e^{j2\pi\frac{n-lD}{N}k}\right]e^{j2\pi\frac{lD}{N}k} \\ &= \frac{1}{N}\sum_{l=-\infty}^{\infty}\left[Y_k(l)e^{j2\pi\frac{l}{N}kD}\right]\left[g_0o(n-lD)e^{j2\pi\frac{n-lD}{N}k}\right] \end{aligned}$$

Again, let

$$g_k'(n) = g_0(n)e^{j2\pi\frac{k}{N}n}.$$

Then,

$$y_k(n) = \frac{1}{N}\left[\sum_{l=-\infty}^{\infty} Y_k(l)e^{j2\pi\frac{l}{N}kD}g_k(n-lD)\right].$$

Note that this equation resembles that of the interpolation process shown in Figure 3.8 where effectively only a subset of coefficients engage the data, and thus, it can be written as

$$y_k^{(n)} = \frac{1}{N}\left[Y_k(l)e^{j2\pi\frac{k}{N}lD}\right]_{D\uparrow} * g_k'(n-l)\ , k = 0, 1, 2, ..., N-1. \tag{3.79}$$

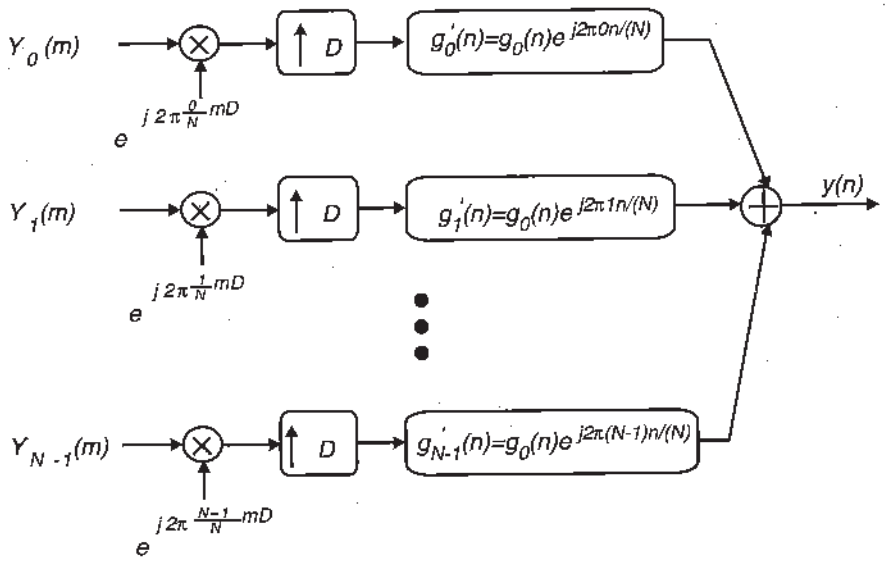


Figure 3.50: Alternate Implementation of a DFT Filter Bank Synthesizer.

Note that $g_k'(n) = g_0(n)e^{j2\pi\frac{k}{N}n}$ is the complex BPF representation of the LPF $g_0(n)$ with the center frequency at ω_k. This alternative implementation of the basic synthesizer is shown in Figure 3.50.

In this alternative implementation, each of the channel input signals is separately multiplied by the complex sinusoid, translating the frequency bands to their original spectral location. The upconverted signals are then interpolated by a factor of D, converting the sample rate to the original value. The interpolation filter is the complex BPF, the impulse response of which is given by $g_k'(n)$, $k = 0, 1, 2, ..., N-1$, which selects the desired bandpass image. For the case of critical sampling, i.e., $D = N$, Equation 3.79 reduces to

$$y_k(n) = \frac{1}{N}\sum_{l=-\infty}^{\infty} Y_k(l)g_k'(n-lN) \tag{3.80}$$

$$= \frac{1}{N}\left[Y_k(m)_{N\uparrow}\right] * g_k'(m). \tag{3.81}$$

Polyphase Implementation of the Digital Filter Bank

Polyphase filters[9] take advantage of the similarities in the digital filter banks with the DFT and IDFT implementation structures. The polyphase structures for the decimator and interpolator, as discussed in Section 3.3, can be applied to the DFT filter banks, resulting in a highly efficient filter bank realization.

[9]Material for this section is based on [31,34].

The basic DFT filter bank analyzer, discussed earlier, implements Equation 3.67:

$$X_k(m) = \sum_{l=-\infty}^{\infty} h_0(l)x(mN-l)e^{j2\pi\frac{k}{N}l} \quad , k = 0, 1, 2, ..., N-1. \tag{3.82}$$

With the assumption of critical sampling, the impulse response $h_0(n)$ of the LPF in each channel can be expressed in polyphase form as

$$p_i(n) = h_0(nN-i) \quad , i = 0, 1, 2, ..., N-1 \tag{3.83}$$

where i refers to the ith branch in the N-branch polyphase decomposition of $h_0(n)$ in any of the k channels, $k = 0, 1, 2, ..., N-1$. From Equation 3.82, replacing l by $rN-i$ and using the definition in Equation 3.83 yields

$$\begin{aligned} X_k(m) &= \sum_{i=0}^{N-1}\sum_{r=-\infty}^{\infty} h_0(rN-i)x(mN-rN+i)e^{j2\pi\frac{k}{N}(rN-i)} \\ &= \sum_{i=0}^{N-1}\sum_{r=-\infty}^{\infty} p_i(r)x\left((m-r)N+i\right)e^{-j2\pi\frac{k}{N}i} \\ &= \sum_{i=0}^{N-1}\sum_{r=-\infty}^{\infty} p_i(r)x_i(m-r)e^{-j2\pi\frac{k}{N}i} \end{aligned} \tag{3.84}$$

where the input signal to each branch of the polyphase network for a single channel, i, in the DFT filter bank structure is expressed as

$$x_i(n) = x(nN+i) \quad , i = 0, 1, 2, ..., N-1. \tag{3.85}$$

Equation 3.85 can be interpreted as advancing the original input $x(N)$ by i samples and then downsampling it by a factor of N. A closer inspection of the result in Equation 3.84 reveals that it can be expressed as the DFT of a convolution operation, i.e.,

$$X_k(m) = \sum_{i=0}^{N-1} \left[p_i(m) * x_i(m)\right] e^{-j2\pi\frac{k}{N}i} \tag{3.86}$$

$$= DFT\left[p_i(m) * x_i(m)\right]. \tag{3.87}$$

The polyphase implementation of the DFT filter bank analyzer is shown in Figure 3.51 for a single filter bank channel along with the equivalent clockwise commutator model (see Figure 3.41).

In a similar fashion, the output of the DFT filter bank synthesizer as expressed by Equation 3.72 with the assumption of critical sampling is

$$y(n) = \sum_{l=-\infty}^{\infty} g_0(n-lN)\left[\frac{1}{N}\sum_{k=0}^{N-1} Y_k(l)e^{j2\pi\frac{n}{N}k}\right] \quad , k = 0, 1, 2, ..., N-1. \tag{3.88}$$

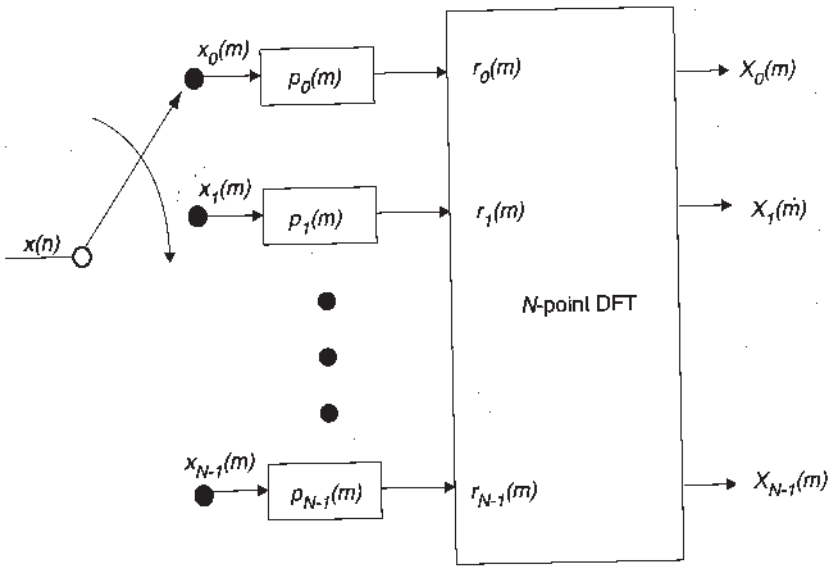


Figure 3.51: Polyphase/Convolution Implementation of a DFT Filter Bank Analyzer.

Let $y(n)$ be divided into N sets of polyphase branch output signals of the form

$$y_i(n) = y(nN+i) \ \ , i = 0, 1, 2, ..., N-1. \tag{3.89}$$

This actually implies that the output $y(n)$ is computed by sequentially taking the polyphase branch outputs, $y_i(n)$, $i = 0, 1, 2, ..., N-1$. Using Equation 3.88, we can reformulate Equation 3.89 as

$$\begin{aligned} y_i(n) &= \sum_{l=-\infty}^{\infty} g_0(nN - lN + i) \left[\frac{1}{N} \sum_{k=0}^{N-1} Y_k(l) e^{j2\pi \frac{nN+i}{N} k} \right] \\ &= \sum_{l=-\infty}^{\infty} g_0\left((n-l)N + i\right) \left[\frac{1}{N} \sum_{k=0}^{N-1} Y_k(l) e^{j2\pi \frac{k}{N} i} \right] \end{aligned} \tag{3.90}$$

By polyphase decomposition of the LPF $g_0(n)$,

$$q_i(n) = g_0(nN+i) \ \ , i = 0, 1, 2, ..., N-1. \tag{3.91}$$

Applying this definition, Equation 3.90 takes the form

$$\begin{aligned} y_i(n) &= \sum_{l=-\infty}^{\infty} q_i(n-l)\left[\frac{1}{N}\sum_{k=0}^{N-1} Y_k(l)e^{j2\pi\frac{k}{N}i}\right] \\ &= \sum_{l=-\infty}^{\infty} q_i(n-l)IDFT\left[Y_k(l)\right] \\ &= q_i(n) * IDFT[Y_k(l)]\,, i = 0,1,2,...,N-1. \end{aligned} \tag{3.92}$$

According to this equation, first the IDFT is calculated using the N-channel DFT filter bank synthesizer input signals, $Y_k(m)$, $k = 0, 1, 2, ..., N-1$ for each sample time, m. The ith IDFT output $V_i(m)$ is the input of the ith branch of the polyphase decomposition of the LPF $g_0(n)$. Then for that instant of time, m, the output of the ith branch filter, $y_i(m)$, is the convolution of the input, $V_i(m)$ and filter coefficients for that branch, $q_i(m)$. The final output $y(n)$ is the interleaved version of these polyphase branch outputs. The implementations of Equations 3.89 and 3.92 are shown in Figure 3.52.

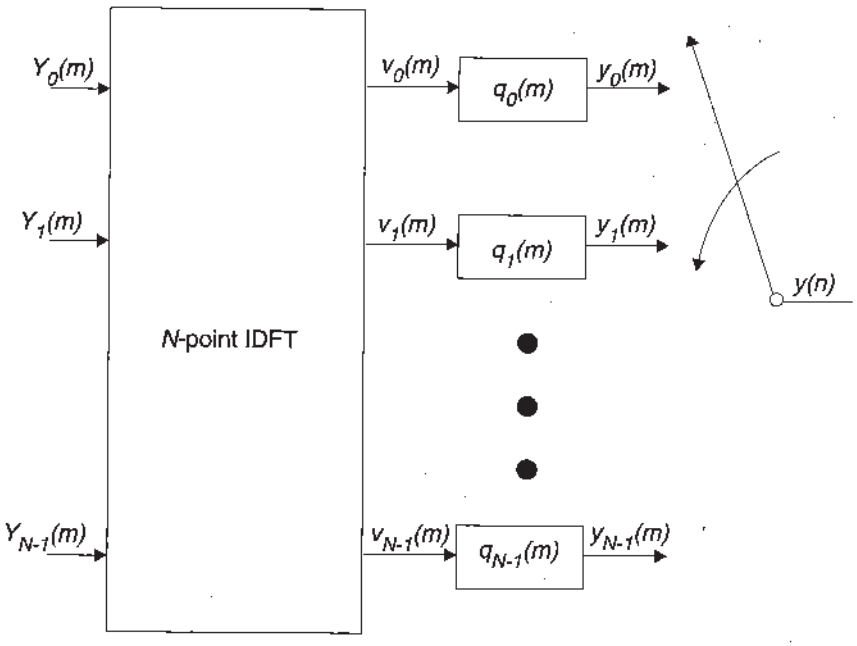


Figure 3.52: Polyphase/Convolution Implementation of a DFT Filter Bank Synthesizer.

3.4.3 Transmultiplexers

An application of multirate digital signal processing in communications is in the channelization of data streams. The goal of these techniques is to convert a TDM signal, seen in Figure 3.53, into a FDM signal, seen in Figure 3.54; conversely, an FDM signal can be converted in a TDM signal.

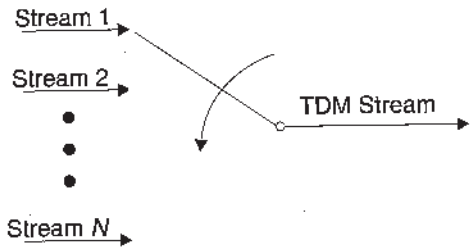


Figure 3.53: TDM Example.

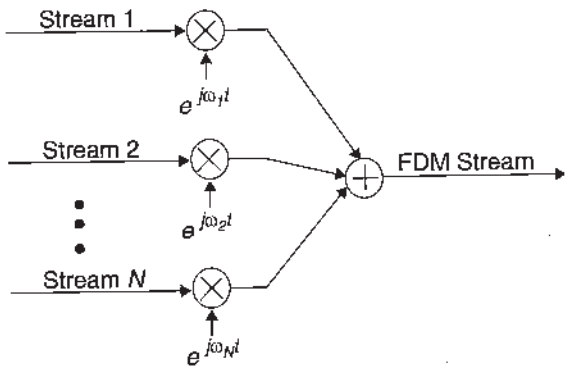


Figure 3.54: FDM Example.

TDM to FDM Conversion

Figure 3.55 shows a block diagram of a system that performs a TDM to FDM conversion. Note the similarity in system layout between that seen in Figure 3.55 and that seen in Figure 3.48, the basic form of a DFT filter bank synthesizer. The underlying architecture is remarkably similar even though a specific type of interpolator structure and modulator can be used. The TDM to FDM conversion converts a single data stream into a channelized version of the original signal. The IDFT process collects several narrowband signals and generates a single wideband signal that contains all the information received.

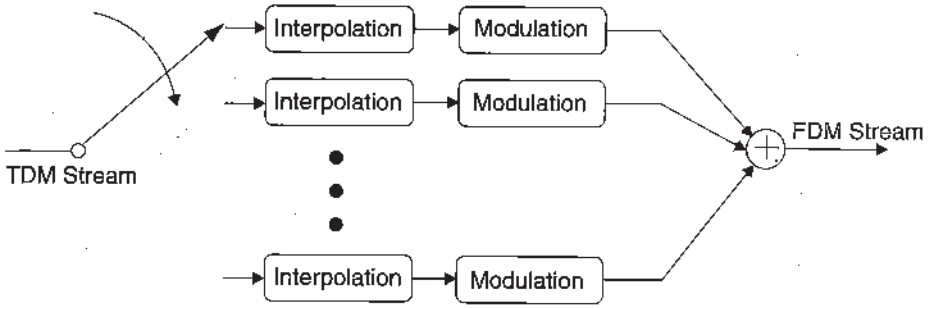


Figure 3.55: TDM to FDM.

FDM to TDM Conversion

Figure 3.56 shows a block diagram of an FDM to TDM conversion system. The similarities between this system layout and the basic form of a DFT filter bank analyzer as shown in Figure 3.47 can be easily seen. Figure 3.47 may be considered as a specific case of the more generalized system structure seen previously.

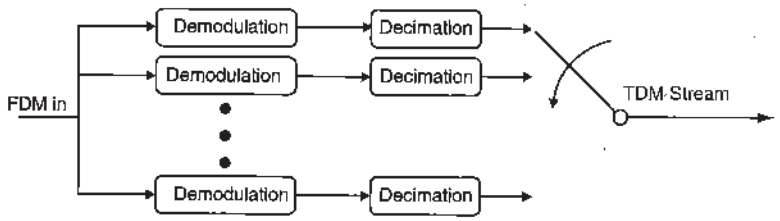


Figure 3.56: FDM to TDM.

Example: Watkins-Johnson's Cellular Basestation Architecture

Wideband transceivers employing software radios can significantly reduce the cost and complexity of basestations. Currently most of the deployed basestations utilize narrowband transceiver technology in which each transceiver individually processes a separate RF channel. In this implementation, the complexity of the system increases linearly with the number of channels resulting in a fairly large and costly realization of the basestation. On the contrary, a wideband transceiver processes the entire frequency band of interest and uses low-cost digital signal processing instead of costly analog components to perform individual transceiver functions on the signal at various stages.

A generic wideband receiver structure used in a basestation developed by Watkins-Johnson, Inc.,[10] is shown in Figure 3.57 [42]. In this implementation, the entire cellular band of interest (e.g., North American cellular band A and B with extensions) is first downconverted to baseband by a single RF tuner. The baseband signal with a bandwidth of approximately 14 MHz is then digitized by a single ADC, which is sampling at 30.72 MHz, slightly above the Nyquist sampling rate. The output (real) of the ADC, which contains all the channels in the cellular band, is digitally downconverted to individual I&Q complex baseband channel signals by an oscillator tuned at the individual channel carriers (complex modulation). The sampling rate of these individual complex signals is still that of the ADC, i.e., 30.72 MHz. A decimation operation (lowpass filtering and downsampling) by a factor of 384 is performed on these high-rate complex channel signals resulting in signals with a sample rate of 80 kHz. The decimated channel signals are now ready for baseband processing (demodulation). The baseband processing block requires an input sampling rate that is an integer multiple of the baud (symbol) rate to facilitate symbol synchronization. Usually the decimated signals' sample rate is not an exact integer multiple of the baud rate. So a further sample rate conversion is performed on the decimated signals before baseband processing.

The part of the receiver that performs extraction of the individual radio channels from the output of the ADC is called the *channelizer*. Reviewing Figure 3.57, it is clearly seen that the structure of the channelizer is similar to the basic DFT filter bank analyzer explained in Section 3.4.2. Alternately, according to the alternate implementation of the DFT filter bank analyzer as in Section 3.4.2, the channelizer can be considered a bank of complex-valued digital BPFs followed by mixers, downsamplers, and sample rate converters. Figure 3.58 shows this representation of the channelizer where each complex BPF $H_k(\omega)$ has a center frequency of $\omega_k = \frac{2\pi k}{K}$, $k = 0, 1, ..., K-1$, which corresponds to a particular RF channel.

Theoretically, a filter bank channelizer can extract any channel in the band $(-F_s/2, F_s/2)$ where F_s is the sampling rate of the channelizer input (output of ADC). This implies that the complexity of a filter bank channelizer remains constant, independent of the number of channels. The impulse responses of the bandpass filters are defined by $h_k(n) = h_0(n)e^{j2\pi\frac{k}{N}n}$ where $h_0(n)$ is a real causal LPF. It follows then that the frequency response of the BPF $H_k(\omega)$ can be expressed as the modulated version of $H_0(\omega)$, i.e.,

$$H_k(\omega) = H_0(\omega - \frac{2\pi k}{K}) \quad , i = 0, 1, 2, ..., N-1. \tag{3.93}$$

■

3.5 Timing Recovery in Digital Receivers Using Multirate Digital Filters

Multirate signal processing is very useful for creating flexible timing recovery techniques for radios supporting multiple waveforms. Multirate digital signal processing timing recovery techniques may also reduce complexity compared to conventional timing recovery

[10] Some divisions of Watkins-Johnson, Inc., are now with BAE Systems.

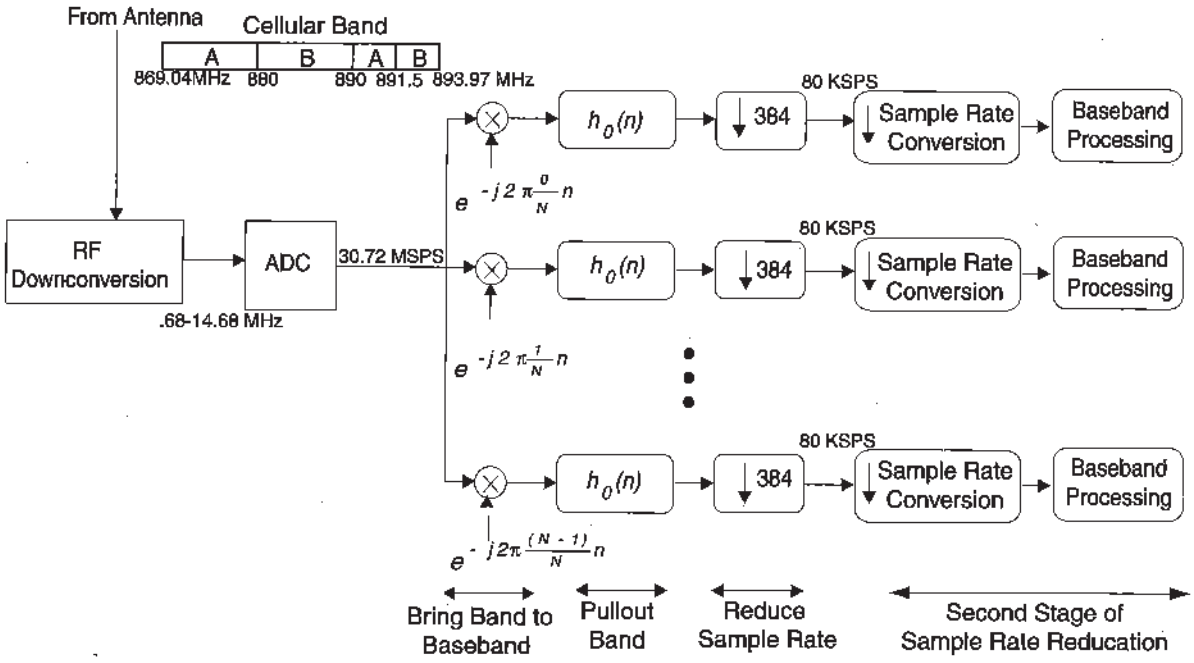


Figure 3.57: Wideband Receiver Block Diagram.

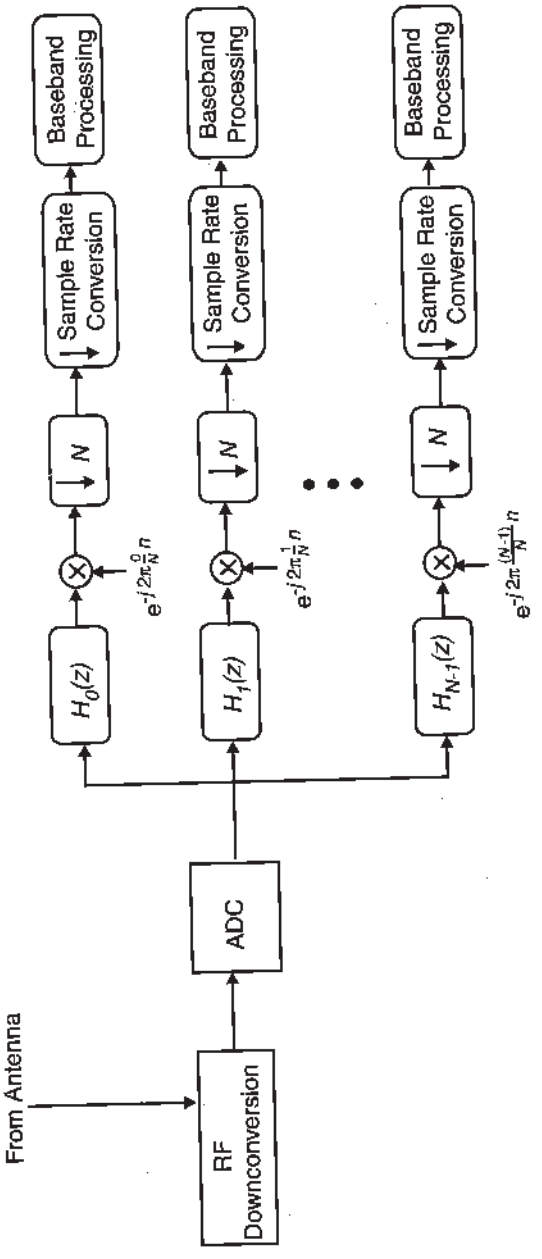


Figure 3.58: Filter Bank View of a Wideband Channelizer.

techniques that rely on altering the sample rate of the ADC. A feedback loop is used in conventional techniques to adjust ADC timing, which spans across both analog and digital domains, greatly complicating the overall design. Multirate digital filters can be used in digital signal processing based receivers to avoid crossing between domains and to allow for simplified timing recovery, providing a more flexible approach to support multiple waveforms.

3.5.1 Timing Recovery in a Classical Analog Receiver

In the classic I&Q receiver, shown in Figure 3.59, a PLL is used to control the phase of a VCO. By altering the phase, the output of the matched filter is sampled at the optimal instance. Note that this structure is somewhat inflexible since software control of the analog matched filter is difficult to accomplish.

A more modern approach is shown in Figure 3.60. The signal is sampled prior to the matched filtering operation. However, the timing recovery requires feedback between the analog and digital domains, which complicates the overall design.

3.5.2 Timing Recovery in the Digital Domain Only

A multirate digital signal processing approach can be used to avoid the need to create a feedback loop with the ADC. The idea is simple: the digitized signal is interpolated using a polyphase filter and the sample point with the maximum energy/power is chosen, as illustrated in Figure 3.61. Two possible approaches for this implementation exist. In the first, the sampling process is performed asynchronously at two samples/symbol and the subsequent polyphase filter interpolates the digitized signal to a higher sampling rate. The correct sample is chosen from the interpolated signal as indicated in Figure 3.61 and the data stream is then downsampled to the symbol rate by selecting the optimal point in the symbol. Since only a limited number of interpolated samples are actually needed to determine the actual sample, a full interpolation filter can be avoided.

A second approach is to sample the input at a rate higher than the Nyquist rate and then decimate it, using a polyphase filter, to the desired output data rate. Timing information is recovered by controlling the starting index of the input samples presented to the decimating polyphase filter's commutator. An advantage of using a polyphase filter approach is that, along with the downsampling operation, it performs a spectral translation by multiples of the output data rate since the signal is being resampled.

For a given signal modulation and roll-off factor α of a raised-cosine pulse, the interpolation factor and the number of stages of interpolation required depends on the number of samples per symbol. The interpolation factor should be large enough to keep the implementation loss due to missampling within a specified limit. As the roll-off factor decreases for a given modulation type, timing becomes a more critical issue and the number of stages of interpolation must increase to prevent significant sampling timing error. For example, for 256-QAM modulation with $\alpha = 0.5$, an interpolation factor of $I = 271$ is required, compared to $I = 500$ for the same modulation with $\alpha = 0.1$, to keep the loss below 0.12 dB [43]. The direct method of implementation of the latter case is to operate at one thousand samples per symbol by using five hundred polyphase stages to increase the sample

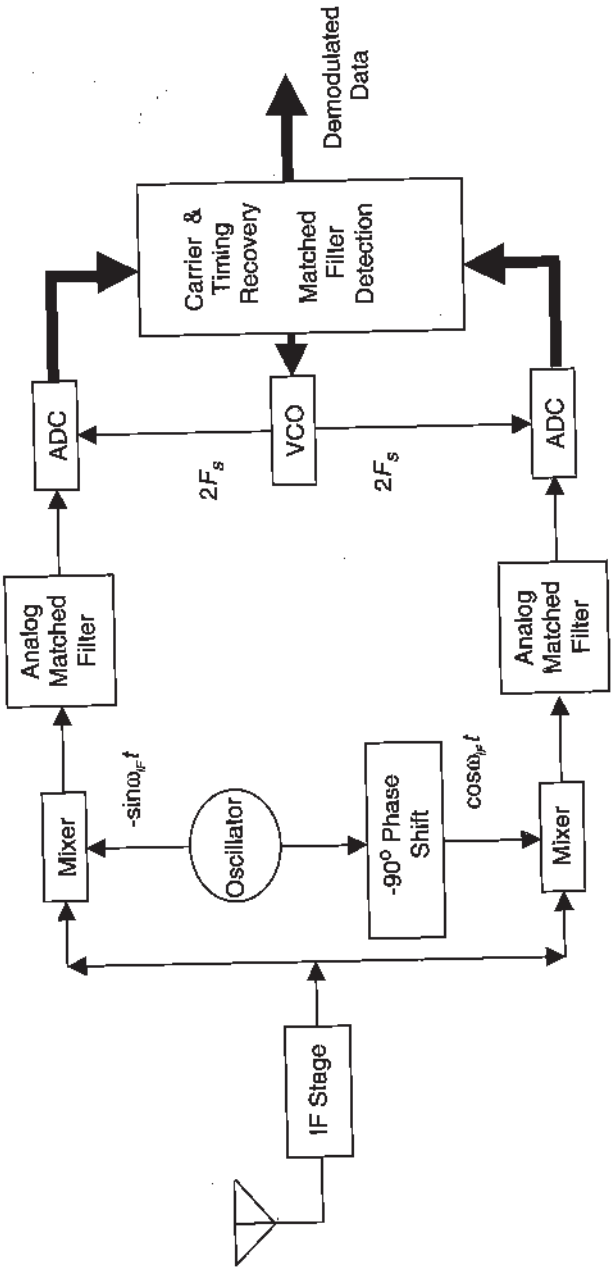


Figure 3.59: Timing Recovery in an Analog Receiver.

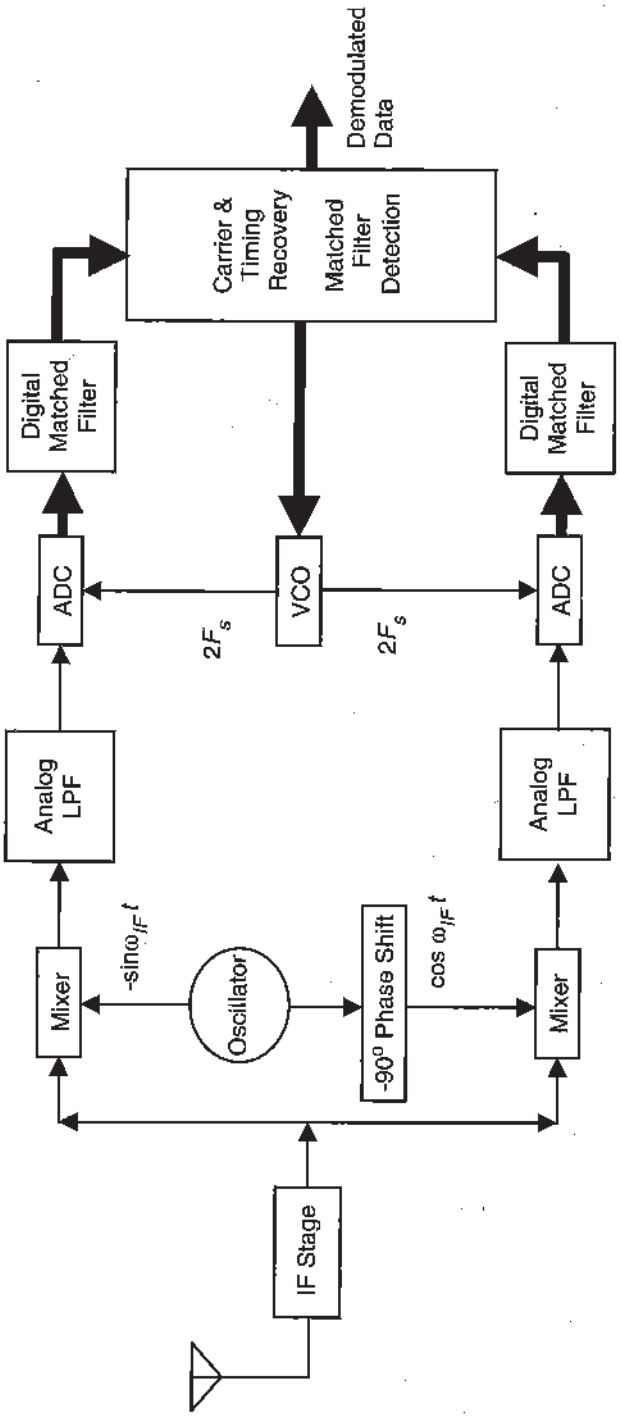


Figure 3.60: Timing Recovery in a First-Generation Digital Receiver.

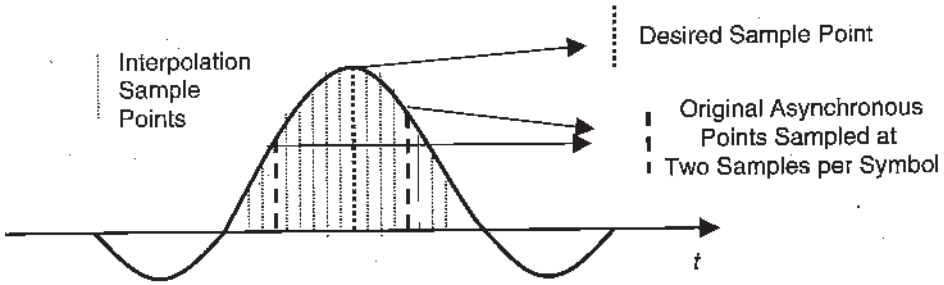


Figure 3.61: Illustration of Interpolated Nyquist Pulse.

rate by a factor of five hundred. However, the unnecessary computation involved can be avoided by using only the specific taps in the polyphase filter that output samples close to the optimum sample point. Such an implementation can be done using the early-late gate synchronizing technique.

3.5.3 Early-Late Gate Synchronizer

For illustrative purposes, consider the simpler problem of detecting the optimal sampling time of a rectangular pulse that is match filtered. The early-late gate synchronizing technique exploits the symmetry of the signal or the symmetry of the matched-filtered signal. The matched-filtered signal output $R(t)$ typically has a symmetric shape (neglecting distortion and noise). The optimal timing is obtained when $R(t)$ is sampled at $t = T$. If we take two measurements $R(T_0 + d)$ and $R(T_0 - d)$ and if $T_0 = T$, the two measurements are equal and the optimal sample would be located halfway between the two samples, as illustrated in Figure 3.62. Using this property, an error signal to a timing recovery loop can be created:

$$\Delta = R(T_0 + d) - R(T_0 - d).$$

The variable Δ is called the *early/late decision statistic* and is an approximation of the scaled derivative of the matched-filter output. If $\Delta > \Delta_{thresh\ early}$, the sampling clock should be decreased by some amount. If $\Delta < \Delta_{thresh\ late}$, then the sampling clock should be increased. This simple principle, which was illustrated in continuous time, can be extended to discrete time.

3.5.4 Timing Offset Control Using the Early-Late Gate Principle

The fundamental goal of symbol synchronization is to sample the pulse at its peak value. This peak can be determined by estimating the derivative of the sampled signal. The first derivative is calculated as demonstrated—taking the difference of the early and late measurements that encompasses the estimated location of the peak. In Figure 3.63, the early-late gate derivative measurement $y'(n - k/N)$ for the kth polyphase output at time n can be computed from the outputs of $k - 1$ and $k + 1$ filter stages [43,44].

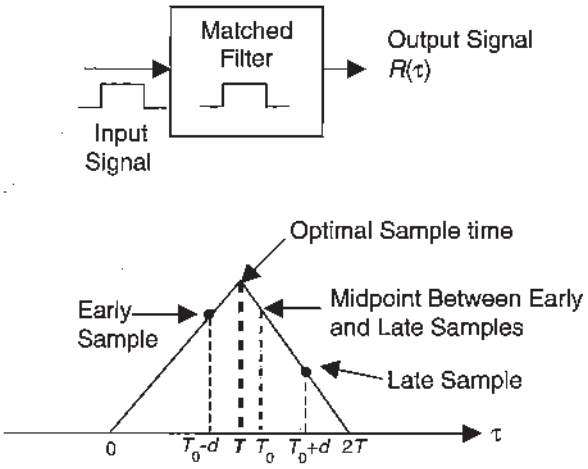


Figure 3.62: The Matched Filtering Process Assuming a Positive Rectangular-Shaped Pulse.

The decision statistic here is the product of the kth polyphase stage and its derivative. This product is necessary to invert the derivative when the output is negative. Thus, the decision statistic will be independent of the desired symbol's value. Driving this product to zero by means of a feedback loop leads to maximum likelihood timing recovery. Notice that the product goes to zero if either the derivative is zero (at the peak) or the kth stage output (signal) is zero.

The process of forming the derivative can be implemented by means of a polyphase derivative matched filter with weights defined as the difference in weights of the $k-1$ and $k+1$ stages shown in Figure 3.63. That is, the derivative can be implemented as a polyphase filter and thus only one filter actually needs be computed to supply information to the feedback loop. Such a filter in conjunction with a polyphase-matched filter can implement the system of Figure 3.63. A generic DSP-based receiver that uses the polyphase filter approach for timing recovery is shown in Figure 3.64. Note that there is no feedback between the digital and analog domains. Furthermore, this approach is well-suited for multiple channel systems, such as FDMA systems, since each channel can have its own interpolation filter. In this case, the polyphase filters can also aid in the receiver channelization as well as in timing recovery.

This approach to synchronization has the maximum flexibility. The analog section becomes more straightforward since the analog processing can be generic, and matched filters for specific pulse-shaping and timing considerations are handled within the more flexible digital domain. Note that the resampling process translates the signal by multiples of the data rate. This reduces the carrier offset and subsequently reduces the complexity of the digital signal processing based carrier recovery technique since the residual frequency offset is lower after resampling.

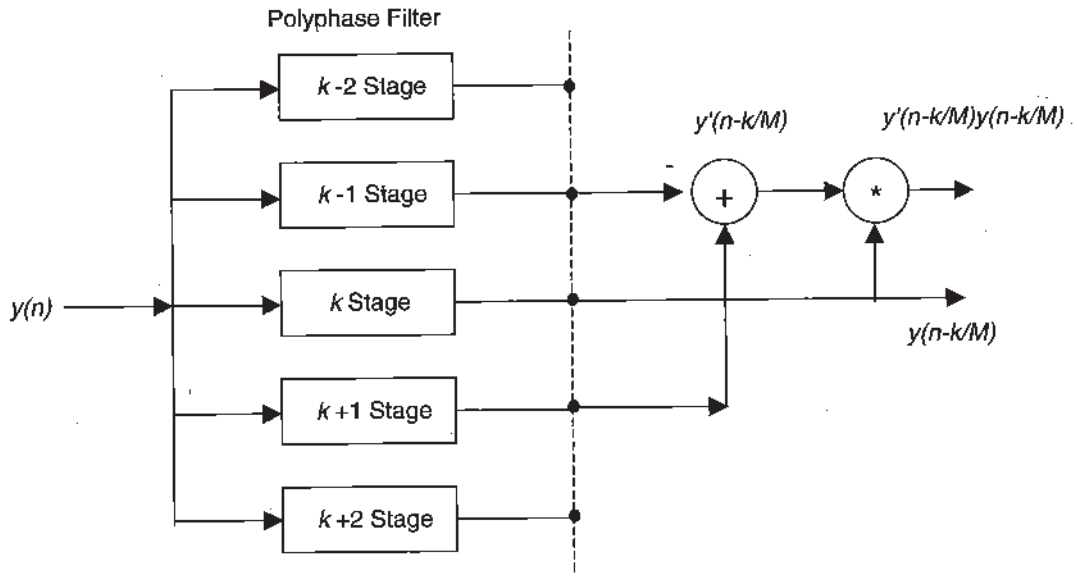


Figure 3.63: Forming the Decision Statistic with a Polyphase Filter Bank.

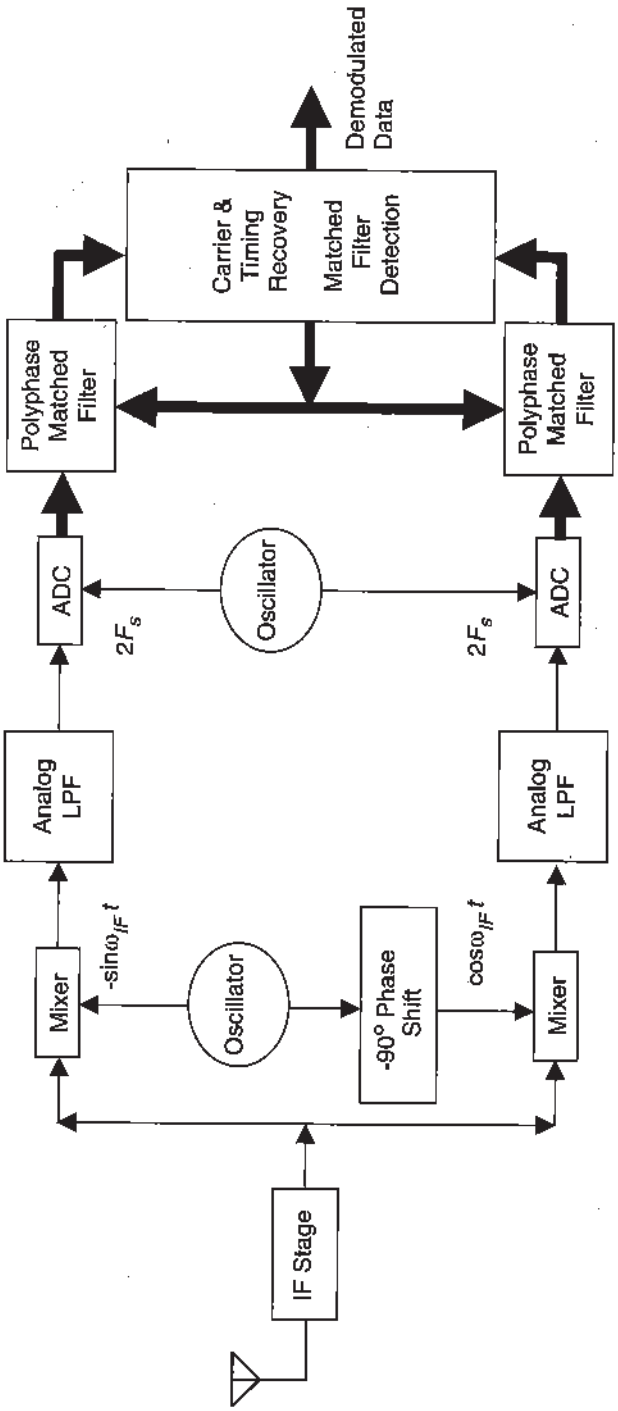


Figure 3.64: A Multirate Filtering Approach to Synchronization.

123

3.6 Conclusion

The advantages of multirate signal processing are manyfold and its application is common in many systems, such as digital filtering, spectral analysis, speech coding, and analog to digital conversion. Multirate techniques are essential in applications of high-speed data acquisition and storage. They reduce the need for expensive anti-aliasing analog filters and enable processing of different types of signals with different sampling rates. The cost of high-speed components goes up exponentially with sample rate or clock rate. Multirate signaling allows partitioning of the high-speed processing into parallel multiple lower-speed processing tasks, thus allowing implementation with low-speed components at a reduced cost. In the design of narrowband digital FIR filters, multirate techniques are especially appealing. They facilitate the implementation of the filter at a lower sampling rate, thereby greatly reducing the filter order and in turn the computational complexity. Wideband receivers take advantage of multirate signal processing for efficient channelization, i.e., partitioning a broadband signal into smaller individual radio channels. Since the channelizer is the most computationally intensive part of the software radio, judicious design using the multirate concept can lead to a significant saving in computational power. Finally, multirate filtering techniques offer flexibility for symbol synchronization and downconversion for software radios. Multirate approaches to timing recovery lead to straight-forward implementations.

3.7 Questions

1. What is the benefit of using the multi-stage (rather than single-stage) structures of a decimator or interpolator when large changes of sampling rate are required?

2. Suppose we are building a cellular basestation and we have a cellular signal of width 25 MHz, sampled at a rate of 50 Msamples/sec. We wish to channelize this signal into channels of width 100 kHz. We want an adjacent channel rejection of 60 dB. This should be implemented using the polyphase filter bank approach using a Kaiser window filter.

   (a) Show the prototype filter (sub-sampled filter). Show a block diagram of the designed filter and a plot of the filter response.

   (b) Show a frequency response plot of the filter bank with a white noise input to prove the adjacent channel interference rejection characteristics.

3. (a) Implement in MATLAB a CIC filter with four stages of integrate and four stages of comb, to reduce the sampling rate by a factor of four. Filter a sinusoidal signal with a frequency of 0.2 Hz (assume a normalized sample rate of 1 Hz) to illustrate the effect of aliasing.

   (b) Repeat this exercise for six stages of integrate and six stages of comb.

   (c) Compare the frequency response of the filter from MATLAB with theoretical results for frequency response. Does the aliasing and attenuation observed match predicted results?

4. Prove that decimating a signal by a factor of D frequency shifts the signals by multiples of D/F_s.

5. Prove that insertion of $I - 1$ zeros between samples creates replicated images at multiples of F_s/I.

6. Design a two-stage LPF and decimator that will take a signal sampled at 96 MHz and convert it to a sample rate of 1 MHz. The passband of the signal after decimation should be 450 kHz with a passband ripple of 0.01 and a stopband ripple of 0.001. Repeat for a single-stage decimator. Determine the filter order(s) and show the frequency response for each. What is the memory requirement and computational rate required for each implementation?

7. Redo Question 6 using three stages of decimation.

8. An interpolator is to be created for implementing an FDM system. Each channel of the FDM system must be interpolated to a sample rate to accommodate the composite signal. Assume that each channel is 20 MHz wide and the sample rate is 44 MHz. There are eight channels in the composite signal sampled at a rate of 8×44 MHz. Assume that at 22 MHz the stopband starts and the attenuation must be 50 dB. The passband ripple is 0.5 dB. Show the overall frequency response, computation rate, and memory of a single-stage implementation and a two-stage implementation.

9. In the example represented by Figure 3.29, redesign the filter assuming that decimation of two is done before decimation by three. Compare the computational requirements for each implementation and demonstrate the frequency response for each.

10. Show the frequency response of the filter given by Equation 3.46 for $N = 1, 2, ...5$. Comment on the trend in the frequency response as N increases.

11. Show the frequency response of the filter given by Equation 3.47 for $N = 1, 2, ...5$ and $M = 1, 2, ..5$ and comment on the trend in the frequency response as N and M increase.

12. Show mathematically the difference in distortion between a reconstructed signal that uses a ZOH and one that doesn't, and comment on the significance of this distortion.

13. Consider a multiband system where the desired system must be able to handle GSM, CDMA, and UMTS (Universal Mobile Telecommunication System), although not all at the same time. We desire a constant sample rate for the ADC, which provides exact multiples of the highest chip rate at 16x, 32x, etc. We want to use a digital downconverter with a polyphase resampler to obtain exact multiples of the GSM symbol rate, 270.833 kHz. The polyphase resampler will be bypassed for the CDMA signals where the wider bandwidths are required. What are the required decimation rates for UMTS, CDMA, and GSM? Note that the chip rate for CDMA is 1.2288 Mc/s and for UMTS (3GPP) is 3.84 Mc/s.

14. Determine the transfer function of the CIC interpolator filter structure shown in Figure 3.39 and plot its frequency response.

Wireless Communications

Software RADIO

A Modern Approach to Radio Engineering

The *definitive* engineer's guide to designing and building software-based radios

Radios, once implemented purely in hardware, are increasingly built using programmable digital signal processing (DSP) devices that enhance device flexibility, simplify manufacturing, and reduce costs. However, many engineers are unfamiliar with the latest techniques for building software radios for wireless systems and devices. This book fills the gap, introduces the key concepts of software radio design, and covers every issue and technique engineers must understand to successfully utilize DSP in their radio systems and subsystems. Coverage includes:

- Central role of multirate DSP in software radio design
- Constructing RF front-ends: utilizing digital processing to overcome key problems in RF design
- Direct digital synthesis of modulated waveforms
- A/D and D/A converters and conversion processes: key tradeoffs among resolution, sampling rate, and dynamic range
- Enhancing performance through smart antennas and other adaptive array algorithms
- Practical techniques for choosing among DSP microprocessors, FPGAs, and ASICs
- A systematic, object-oriented approach to creating flexible and reusable real-time software

The book concludes with case studies drawn from classic historical software radios, current commercially available products, and the advanced work of the SDR Forum, the leading consortium of companies, universities, and research organizations promoting software radio development.

About the Author

JEFFREY H. REED, recent Director of the Mobile and Portable Radio Research Group (MPRG) and Professor of Electrical Engineering at Virginia Tech, specializes in software radios, smart antennas, interference rejection, modem design, and position location. Dr. Reed is co-author/co-editor of eight books and has written chapters for many others.

PRENTICE HALL
Upper Saddle River, NJ 07458

www.phptr.com

ISBN 0-13-081158-0

90000

9 780130 811585