



MACMILLAN TECHNOLOGY SERIES

ADSL/VDSL PRINCIPLES

*A Practical and Precise Study of
Asymmetric Digital Subscriber Lines and
Very High Speed Digital Subscriber Lines*

Dr. Dennis J. Rauschmayer

ADSL/VDSL Principles

By Dr. Dennis J. Rauschmayer

Published by:
Macmillan Technical Publishing
201 West 103rd Street
Indianapolis, IN 46290 USA

All rights reserved. No part of this book may be reproduced or transmitted in any form or by any means, electronic or mechanical, including photocopying, recording, or by any information storage and retrieval system, without written permission from the publisher, except for the inclusion of brief quotations in a review.

Copyright © 1999 by Macmillan Technical Publishing

Printed in the United States of America 1 2 3 4 5 6 7 8 9 0

02 01 00 99 4 3 2 1

Library of Congress Cataloging-in-Publication Data

Library of Congress Cataloging Number: 96-80467

ISBN: 1-57870-015-9

Universal Product Code: 619472001592

Warning and Disclaimer

This book is designed to provide information about Digital Subscriber Line technology. Every effort has been made to make this book as complete and as accurate as possible, but no warranty or fitness is implied.

The information is provided on an "as is" basis. The authors and Macmillan Technical Publishing shall have neither liability nor responsibility to any person or entity with respect to any loss or damages arising from the information contained in this book or from the use of the disks or programs that may accompany it.

Trademark Acknowledgments

All terms mentioned in this book that are known to be trademarks or service marks have been appropriately capitalized. Macmillan Technical Publishing cannot attest to the accuracy of this information. Use of a term in this book should not be regarded as affecting the validity of any trademark or service mark.

DEVELOPMENT EDITOR

Brad Miser
Christopher Cleveland

PROJECT EDITOR

Dayna Isley

COPY EDITORS

Geneil Breeze
June Waldman

TECHNICAL REVIEWER

Jacky Chow

TEAM COORDINATOR

Amy Lewis

MANUFACTURING COORDINATOR

Brook Farling

BOOK DESIGNER

Anne Jones

COVER DESIGNER

Sandra Schroeder

COVER PRODUCTION

Aren Howell

DIRECTOR OF PRODUCTION

Larry Klein

PRODUCTION MANAGER

Laurie Casey

GRAPHICS IMAGE SPECIALIST

Oliver Jackson

PRODUCTION TEAM

Louis Porter, Jr.

INDEXER

Bront Davis

PROOFREADER

Mary Ellen Stephenson

DSL Modulation Basics

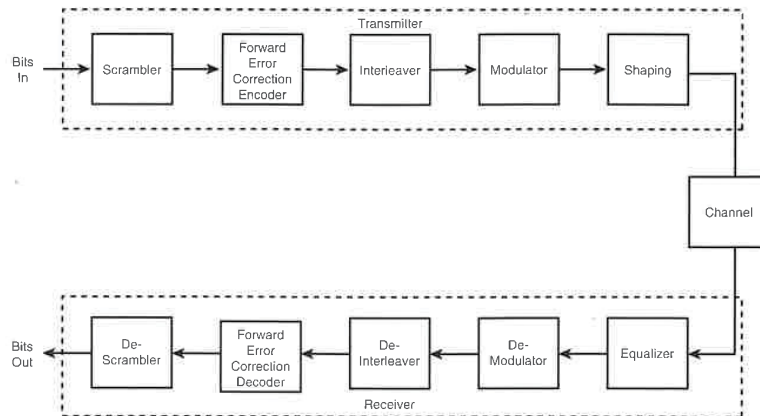
In this chapter:

- Descriptions of the different blocks used in DSL transmitters and receivers
- Basic concepts of single carrier and multicarrier modulation schemes
- Discussion of forward error correction coding and decoding
- Overview of equalization methods and the performance bounds on each

This chapter presents some of the basics of modulation systems as well as other blocks used by DSL communications systems. Subsequent chapters cover ADSL and VDSL in detail and show how the specific implementations of the basic blocks are put together to create a full DSL transmitter and receiver. Many of the topics and blocks in this chapter are also relevant to communication systems other than those used on DSLs, including voice-band modems, cable modems, and magnetic storage channels.

Figure 6.1 shows a generic block diagram of a transmitter and a receiver that is used for reference throughout this chapter. This figure is not specific to either a Carrierless Amplitude and Phase (CAP)/Quadrature Amplitude Modulation (QAM) or a discrete multitone (DMT) system, but rather it illustrates where some of the relevant blocks fit in the “big picture.” Notice that most blocks in the transmitter have corresponding sister blocks in the receiver.

Figure 6.1 Transmitter and receiver block diagram for a communications system.



Modulation/Demodulation Blocks

A modulation block typically converts input bits into waveforms to be sent over the channel. A demodulation block in the receiver maps received waveforms back to the bits that most likely generated the waveform. The main idea is to re-create the bits at the demodulator output that were present at the modulator input. The following sections describe three types of modulation common in DSLs: QAM, CAP, and multitone.

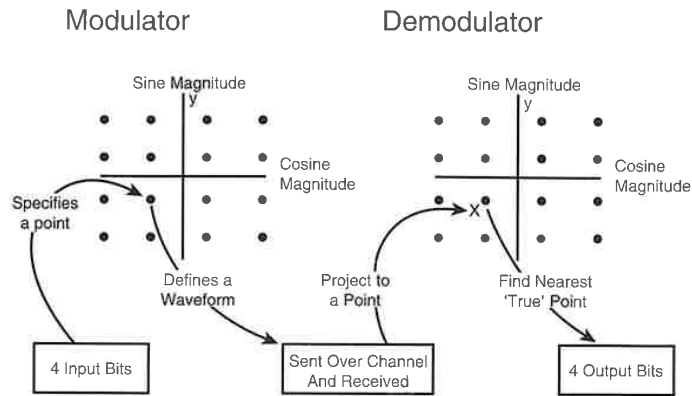
QAM Modulation

QAM utilizes a sine wave and a cosine wave with the same frequency component to convey information. The waves are sent over a single channel simultaneously, and the amplitude (including sign and magnitude) of each wave conveys the information (bits) being sent. At least one period (and sometimes more) of the waves is sent to convey a set of bits before a new set of bits is to be sent (resulting in new magnitudes for the sine and cosine waves). QAM modulation has been used for many years and is the basis for many of the voice-band modem specifications, including V.34.

A Simple QAM Example

An example of a QAM modulation system that can send four bits of information per QAM symbol appears in Figure 6.2.

Figure 6.2 An example of a QAM system sending four bits per symbol over a channel.



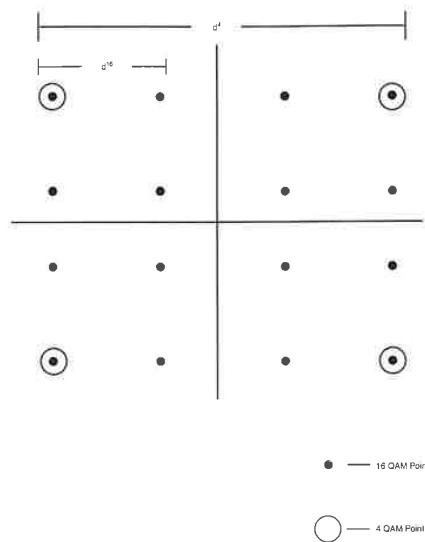
The four bits of information to be transmitted are mapped to 1 of 16 points on a QAM *constellation*. Note that for four bits of information, 16 points allow a unique point for any combination of bits. The x and y components of the point to which the bits were mapped specify the amplitude of the cosine wave and sine wave to be sent over the channel. Both the transmitter and receiver know the predetermined method of mapping between the bits and a point.

After cosine and sine waves are sent over the channel, the receiver recovers and estimates the amplitude of each wave (which normally requires significant equalization and processing as described later in the chapter). These magnitudes are projected on a constellation identical to that used at the transmitter. Again, the cosine wave amplitude is the x component, and the sine magnitude is the y component. Normally, noise and distortion in the channel and compromised electronics in the transmitter and receiver prevent the point projected onto the receiver's constellation from falling directly on top of a "true" point. The receiver, however, selects the closest "true" point to the projected point as the one the transmitter is most likely to use to generate the QAM symbol. This point is then mapped into four bits using the same mapping method employed in the transmitter (but mapping in the opposite direction). If too much noise is present at the receiver, the point projected on the constellation may be closer to the wrong known point than the accurate known point, resulting in an error estimating the recovered symbol.

This example is sometimes called 16-QAM because the constellation has 16 points. In general, other numbers of bits can be transmitted per QAM symbol. For example, if two bits per symbol were transmitted, a four-point constellation would be necessary and the modulation would be called 4-QAM. Figure 6.3 shows a 4-QAM constellation overlaid on

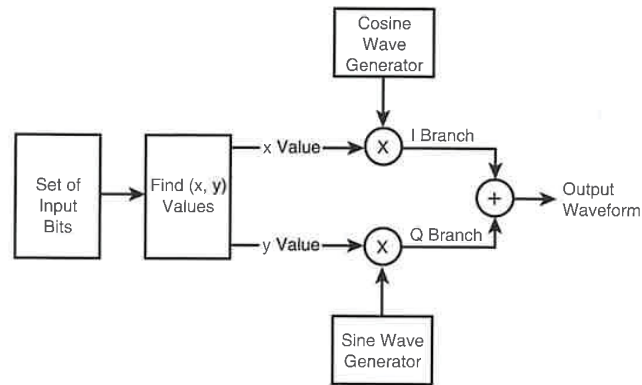
a 16-QAM constellation. The average energy of each signal set is approximately the same. Note that the distance between the points in the 4-QAM constellation (labeled d_4) is greater than the distance between the points in the 16-QAM constellation (labeled d_{16}). Consequently, on the same channel more noise is necessary to cause an error when using 4-QAM than when using 16-QAM. Said differently, 16-QAM requires a greater SNR than 4-QAM requires.

Figure 6.3 An example of a 4-QAM constellation overlaid on a 16-QAM constellation with the same average energy.



A block diagram of a QAM modulator appears in Figure 6.4. The branch of the modulator that carries the cosine wave amplitude is sometimes called the *in phase* branch. The cosine amplitude is called the in phase, or the I, component. The branch of the modulator that carries the sine wave amplitude is sometimes called the *quadrature* branch. The sine amplitude is called the quadrature, or the Q, component.

Figure 6.4 QAM modulator block diagram.



A Mathematical Treatment of QAM

Orthogonality between a sine wave and a cosine wave allows them to transmit data simultaneously over a channel. Considering a single period of each wave, the orthogonality principle can be illustrated by Eqtn. 6.1.

Eqtn. 6.1

$$\int_0^{\tau} \sin\left(\frac{2\pi}{\tau}t\right) \cos\left(\frac{2\pi}{\tau}t\right) dt = 0$$

In Eqtn. 6.1, τ is the period of the sine and cosine waves. Because of the orthogonality property, the sine and cosine functions are sometimes called *basis functions*.

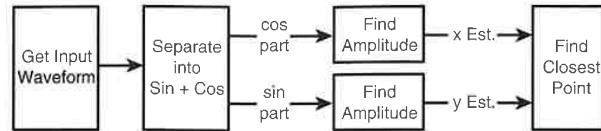
An example of a QAM demodulator appears in Figure 6.5.

The input to this block comes from the channel, and the output projects onto the receiver's constellation. If one assumes that the channel has no loss and that the receiver has perfect phase timing, then it is not difficult to write the equations for each point in Figure 6.5 for some symbol, i . Eqtn. 6.2 gives the input signal at point A.

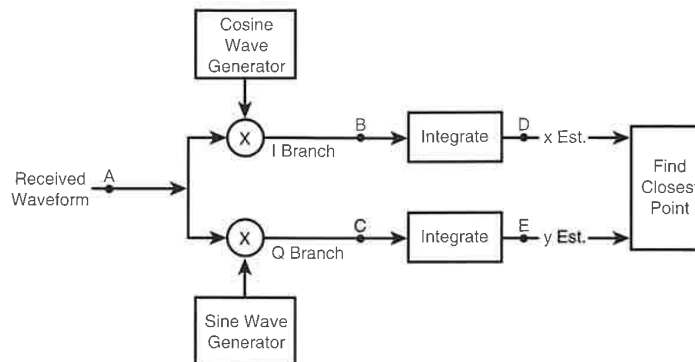
Eqtn. 6.2

$$V_A(t) = X_i \cos(\omega t) + Y_i \sin(\omega t)$$

Figure 6.5 (a) Logical view of a QAM demodulator. (b) Signal-processing view of a QAM demodulator.



(a)



(b)

In Eqtn. 6.2, X_i is the amplitude (sign and magnitude) of the cosine wave encoded at the transmitter, and Y_i is the amplitude (sign and magnitude) of the sine wave encoded at the transmitter. After passing through the multiplication blocks at points B and C, the signals are given by Eqtn. 6.3 and Eqtn. 6.4.

Eqtn. 6.3

$$V_B(t) = X_i \cos^2(\omega t) + Y_i \sin(\omega t) \cos(\omega t)$$

Eqtn. 6.4

$$V_C(t) = X_i \cos(\omega t) \sin(\omega t) + Y_i \sin^2(\omega t)$$

The signals at points B and C then independently pass through integration blocks. These blocks integrate over a single period and reset after each symbol. The switches in Figure

6.5 sample the output of the integrator at the end of each integration period. The signal at points D and E can thus be written as Eqtn. 6.5 and Eqtn. 6.6.

Eqtn. 6.5

$$\begin{aligned}
 V_D(t) &= \int_0^{\tau} V_B(t) dt \\
 &= \int_0^{\tau} X_i \cos^2(\omega t) + Y_i \sin(\omega t) \cos(\omega t) dt \\
 &= \int_0^{\tau} X_i \cos^2(\omega t) dt + \int_0^{\tau} Y_i \sin(\omega t) \cos(\omega t) dt \\
 &= \frac{\tau X_i}{2} + 0 = \frac{\tau X_i}{2}
 \end{aligned}$$

Eqtn. 6.6

$$\begin{aligned}
 V_E(t) &= \int_0^{\tau} V_C(t) dt \\
 &= \int_0^{\tau} X_i \cos(\omega t) \sin(\omega t) + Y_i \sin^2(\omega t) dt \\
 &= \int_0^{\tau} X_i \cos(\omega t) \sin(\omega t) dt + \int_0^{\tau} Y_i \sin^2(\omega t) dt \\
 &= 0 + \frac{\tau Y_i}{2} = \frac{\tau Y_i}{2}
 \end{aligned}$$

Note that the terms going to zero in Eqtn. 6.5 and Eqtn. 6.6 do so because of the orthogonality property of sines and cosines. The values at the output of the demodulator then contain a scaled estimate of the magnitudes of the received sine and cosine waves to project onto the receiver's constellation map.

Note

The scale factor can be removed before integrating by scaling the multiplication cosine and sine waves.

This analysis assumes that the baseband pulses in the modulator used to shape the symbols before multiplying by the sine and cosine waves were simple pulses with unity amplitude

during the entire symbol period. In general, a shaping filter may be used to shape the pulse before the mixing function. (The purpose of this step is to reduce the bandwidth of the transmitted signal by filtering out higher-frequency components.) If the shaping filter has a response of $p(t)$, then the transmitted signal for symbol i will be of the form shown in Eqtn. 6.7.

Eqtn. 6.7

$$V(t) = X_i p(t - i\tau) \cos(\omega t) + Y_i p(t - i\tau) \sin(\omega t)$$

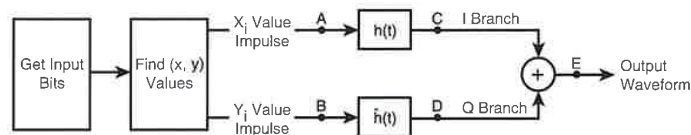
Though the shaping pulse is not very important to the basics of QAM, including the shaping term is more complete and allows comparison to CAP in the next section.

Thus I have shown mathematically how a QAM signal can be received and demodulated. Other methods of demodulation are possible as well, and most take advantage of the orthogonality property discussed in this section.

CAP Modulation

Similar to a QAM modulator, a carrierless amplitude and phase (CAP) modulator uses a constellation to encode bits at the transmitter and decode bits at the receiver. The x and y values resulting from the encoding process are then used to excite a digital filter. A CAP modulator appears in Figure 6.6.

Figure 6.6 A CAP modulator.



The modulator has two branches—one in-phase branch and one quadrature branch. The impulse responses of the digital filters are *hilbert transform pairs* or more simply, a *hilbert pair*. Two functions that form a hilbert pair are orthogonal to one another. In general, any valid hilbert pair can be used to make a CAP modulator; however, today's implementation of CAP uses a cosine wave and a sine wave shaped with a transmit pulse. Typically, CAP modulation is performed with digital filters instead of in-phase and quadrature multipliers.

When comparing a CAP modulator to a QAM modulator, consider voltages at the different points in Figure 6.6. Assume that this modulator uses the same constellation encoding method, the same size constellation, and the same symbol rate as the QAM modulator discussed earlier. The signal at each point can then be written as Eqtn. 6.8 and Eqtn. 6.9.

Eqtn. 6.8

$$V_A(t) = X_i \delta(i\tau)$$

Eqtn. 6.9

$$V_B(t) = Y_i \delta(i\tau)$$

Note that both quantities are simply discrete impulses that will excite the digital filters in the modulator.

Given the impulse responses of the filters in the modulator, the output of the two filters—found by simple convolution—due to the inputs is shown by Eqtn. 6.10 and Eqtn. 6.11.

Eqtn. 6.10

$$\begin{aligned} V_C(t) &= X_i \delta(i\tau) * h(t) \\ &= \int_0^{\infty} X_i \delta(i\alpha) h(t - \alpha) d\alpha \\ &= X_i h(t - i\tau) \end{aligned}$$

Eqtn. 6.11

$$\begin{aligned} V_D(t) &= Y_i \delta(i\tau) * \hat{h}(t) \\ &= \int_0^{\infty} Y_i \delta(i\alpha) \hat{h}(t - \alpha) d\alpha \\ &= Y_i \hat{h}(t - i\tau) \end{aligned}$$

In these equations, the sifting property was used to arrive at both results.

The output of the modulator due to symbol i is then given by Eqtn. 6.12.

Eqtn. 6.12

$$V_E(t) = X_i h(t - i\tau) + Y_i \hat{h}(t - i\tau)$$

Comparing Eqtn. 6.12 to the output of a QAM modulator with shaping given in Eqtn. 6.7, note that the two would be identical if Eqtn. 6.13 and Eqtn. 6.14 hold.

Eqtn. 6.13

$$h(t - i\tau) = p(t - i\tau) \cos(\omega t)$$

Eqtn. 6.14

$$\hat{h}(t - i\tau) = p(t - i\tau) \sin(\omega t)$$

If one further designed the system such that $\omega\tau = \pi$, then Eqtn. 6.13 and Eqtn. 6.14 can be written as Eqtn. 6.15 and Eqtn. 6.16.

Eqtn. 6.15

$$h(t - i\tau) = p(t - i\tau) \cos(\omega(t - i\tau))$$

Eqtn. 6.16

$$\hat{h}(t - i\tau) = p(t - i\tau) \sin(\omega(t - i\tau))$$

Both of these equations are valid because i is an integer. The time impulse responses of the filters with no dependence on i then become Eqtn. 6.17 and Eqtn. 6.18.

Eqtn. 6.17

$$h(t) = p(t) \cos(\omega t)$$

Eqtn. 6.18

$$\hat{h}(t) = p(t) \sin(\omega t)$$

These results are important because they show that properly relating the symbol rate and center frequency of a CAP and QAM system allows the time-domain waveforms to be identical. For other relationships of ω and τ —assuming that they are the same across a CAP and QAM system—it can be shown that the modulation systems differ by a rotation. That is, the waveforms will be identical if one of the sets of constellations used for encoding and decoding is rotated about the origin of the plane.¹

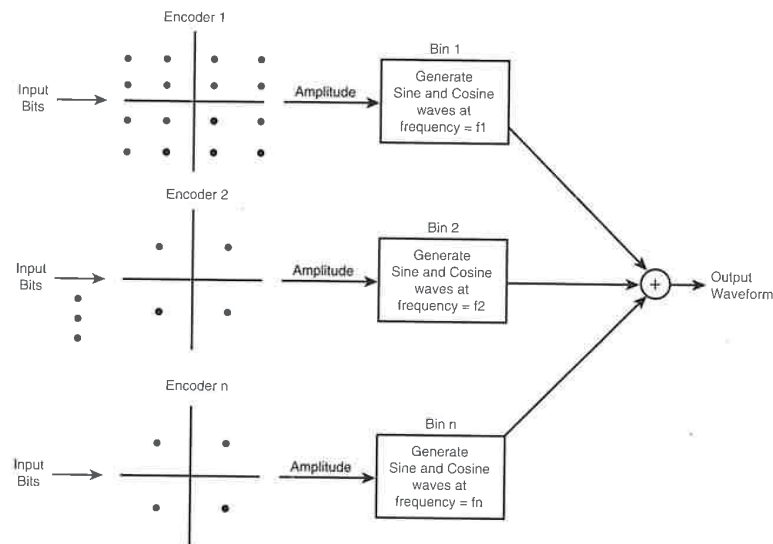
Multitone Modulation

Multitone modulation is the basis of the DMT version of ADSL as well as some multicarrier versions of VDSL. This type of modulation is sometimes called *orthogonal frequency division multiplexing* (OFDM). For simplicity, this book simply refers to multitone modulation as DMT.

DMT builds on some of the ideas of QAM. Imagine having more than one constellation encoder. Each encoder receives a set of bits that are encoded using a constellation encoder as described in the previous sections. The output values from the constellation encoder are

again the amplitudes of cosine and sine waves; however, a different sine and cosine frequency is used for each constellation encoder. All the sine and cosine waves are then summed together and sent over the channel. This waveform is a simple DMT symbol, represented by the diagram in Figure 6.7.

Figure 6.7 Conceptual view of a DMT modulator.



Assuming that one can separate the sines and cosines at different frequencies at the receiver, each set of waveforms can be independently decoded in the same way that a QAM signal can be decoded and the resultant bits output by the constellation decoders.

Note that the idea of using different frequencies to transmit information is not unique to DMT. Television and radio have readily employed such techniques. The difference here is that DMT has a “receiver” tuned to all of the channels at once, whereas the others typically tune to only one channel. Some names used for frequency channels in DMT are *frequency bins* (or *bins*), tones or DMT tones, and subchannels.

Perhaps not obvious in the previous discussion is the assumption that the waveforms in each bin are completely separable from one another. If this is not the case, then decoding for each bin is difficult because the sine and cosine waves in each bin can be corrupted by signals from other bins. One key to DMT is that the sine and cosine frequencies used in each bin should be integer multiples of a common frequency and that the symbol period, τ , be the inverse of that common frequency. (It can also be an integer multiple of the reciprocal of that frequency.)

This common frequency will be referred to as the *fundamental frequency*. From the analysis done for QAM, one can already claim that the sine and cosine waves at the fundamental frequency form basis functions. To ensure that interference does not exist between bins, one must show that sine and cosine waves from any bin are orthogonal to sine and cosine waves in any other bin. Mathematically, this orthogonality can be written as Eqtns. 6.19, 6.20, and 6.21.

Eqtn. 6.19

$$\int_0^{\tau} \cos(n\omega_f t) \cos(m\omega_f t) dt = 0$$

Eqtn. 6.20

$$\int_0^{\tau} \cos(n\omega_f t) \sin(m\omega_f t) dt = 0$$

Eqtn. 6.21

$$\int_0^{\tau} \sin(n\omega_f t) \sin(m\omega_f t) dt = 0$$

Here n and m are different integers, and ω_f is the radian fundamental frequency. Performing the integration in Eqtn. 6.19 yields Eqtn. 6.22.

Note that if $n=m$, then the first term will converge to $1/2 \tau$, which follows from the result of Eqtn. 6.5. The relationships in Eqtn. 6.20 and Eqtn. 6.21 can be likewise shown except that in Eqtn. 6.20 the orthogonality holds even for the case of $n=m$, similar to the discussion around Eqtn. 6.1. Thus the demodulation of a DMT symbol relies on the orthogonality of sine and cosine waves at different frequencies as well as between a sine and cosine wave at the same frequency.

The modulation and demodulation procedures in Figure 6.7 are brute-force methods of creating and detecting a DMT symbol. These methods would be very monotonous on an actual implementation and are generally not the way DMT systems are realized. To better understand how an implementation could be simplified, consider the summation of a sine and cosine of duration τ . Such a waveform could be written as Eqtn. 6.23.

Eqtn. 6.22

$$\begin{aligned}
& \int_0^{\tau} \cos(n\omega_f t) \cos(m\omega_f t) dt \\
&= \int_0^{\tau} \frac{1}{2} \cos((n-m)\omega_f t) + \frac{1}{2} \cos((n+m)\omega_f t) dt \\
&= \left[\frac{\sin((n-m)\omega_f t)}{2\omega_f(n-m)} + \frac{\sin((n+m)\omega_f t)}{2\omega_f(n+m)} \right]_0^{\tau} \\
&= \frac{\sin\left((n-m)\frac{2\pi}{\tau}\tau\right)}{2\omega_f(n-m)} + \frac{\sin\left((n+m)\frac{2\pi}{\tau}\tau\right)}{2\omega_f(n+m)} \\
&= \frac{\sin((n-m)2\pi)}{2\omega_f(n-m)} + \frac{\sin((n+m)2\pi)}{2\omega_f(n+m)} \\
&= 0 \text{ for } n, m \text{ integers and } n \neq m
\end{aligned}$$

Eqtn. 6.23

$$s(t) = \begin{cases} X_n \cos(n\omega_f t) + Y_n \sin(n\omega_f t) & \text{for } 0 < t \leq \tau \\ 0 & \text{otherwise} \end{cases}$$

Such a signal is representative of the contribution of a single bin, or simply bin n , to a single DMT symbol. If $s(t)$ is sampled at a rate of $2 \cdot N \cdot f_f$, the resulting nonzero values of the signal are given in Eqtn. 6.24.

Eqtn. 6.24

$$\begin{aligned}
s_k &= X_n \cos\left(n\omega_f \frac{k}{2Nf_f}\right) + Y_n \sin\left(n\omega_f \frac{k}{2Nf_f}\right) \\
&= X_n \cos\left(\frac{\pi nk}{N}\right) + Y_n \sin\left(\frac{\pi nk}{N}\right) \text{ for } 0 < k \leq 2N
\end{aligned}$$

In a DMT system, N would represent the largest bin transmitting a signal. This signal is at a frequency of Nf_f . Because the Nyquist theorem states that the sampling rate for a system must be twice the largest frequency in the system, the rate of $2Nf_f$ was chosen. If we take

the *discrete fourier transform* (DFT) of s_k using $2N$ points in the transform, the result is Eqtn. 6.25.

Eqtn. 6.25

$$S_m = \sum_{k=0}^{2N} \left(X_n \cos\left(\frac{\pi nk}{N}\right) + Y_n \sin\left(\frac{\pi nk}{N}\right) \right) e^{-j \frac{2\pi km}{2N}}$$

$$= \begin{cases} N(X_n - jY_n) & \text{for } m = n \\ N(X_n + jY_n) & \text{for } m = 2N - n \\ 0 & \text{otherwise} \end{cases}$$

Because the DFT decomposes a signal into its frequency-domain components, the result of Eqtn. 6.25 should not be surprising. Simply put, over the $2N$ points being transformed, the signal had energy at only one frequency. Nonzero values at two points in the frequency domain arise because, in the frequency domain the frequency represented after the transform represents a two-sided spectrum. For all real values in the time domain (consistent with s_k), the $2N$ points will exhibit complex conjugate symmetry about the center of the points. This is analogous to having positive and negative frequency components resulting from a fourier transform. The zero values in Eqtn. 6.25 result from no energy being present at the frequencies represented by those points. In essence, Eqtn. 6.25 illustrates orthogonality of sines and cosines at different frequencies, as well as a sine and cosine wave at the same frequency.

The result of Eqtn. 6.25 suggests another way to produce a DMT symbol. Instead of mapping the output of a constellation encoder into a cosine and sine amplitude, the output can be mapped into a complex number in a vector. The value from the X-, or cosine, axis would represent the real part of the complex number, and the Y value, or sine magnitude, would represent the imaginary part. If the outputs of all the constellation encoders are ordered in the vector, then each vector point represents one of the DMT bins. If N bins existed in the DMT system, the complex vector would have N entries. A suffix containing the complex conjugates of the vector's original entries can then be added to this vector such that the new vector has complex conjugate symmetry. An *inverse discrete fourier transform* (IDFT) on this new vector would then produce a real-valued time-domain sequence equivalent to the original DMT modulator in Figure 6.7. Figure 6.8 shows this new method of performing DMT modulation.

Figure 6.8 also shows a method of DMT demodulation. Note that it is basically the reverse of the modulator except that a DFT is used instead of an IDFT. This approach should make sense because the DFT goes from the time domain to the frequency domain. Because the time-domain values are real, the output of the DFT block has complex conjugate symmetry. Only half of the output is then necessary to drive the constellation

result is

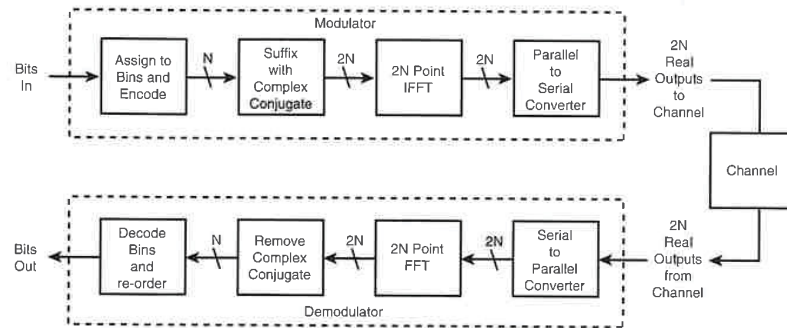
ie result of
formed,
: frequency
he trans-
nsistent
r of the
resulting
ing pre-
ates
osine wave

ad of map-
output
sine, axis
agnitude,
ers are
f N bins
ontaining
vector
rier
nain
ows this

the
roach
y domain.
lex conj-
n

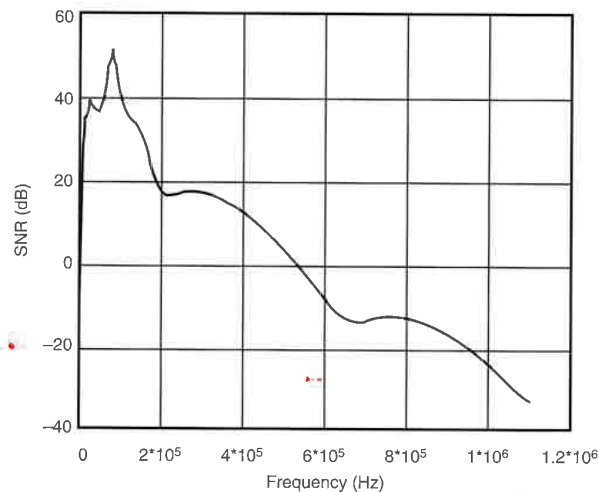
decoders. Implementations often use the fast fourier transform (FFT) and inverse fast fourier transform (IFFT) to realize the modulator and demodulator. These transform algorithms realize the DFT and IDFT with reduced computational complexity.

Figure 6.8 DMT modulation using an IDFT.



DMT allows a communications system to be very flexible and optimally utilize a channel. For example, consider the channel discussed as a case study in Chapter 5, “DSL Theoretical Capacity in Crosstalk Environments.” The SNR of the channel, assuming an ADSL downstream PSD at the transmitter, appears in Figure 6.9.

Figure 6.9 SNR with respect to frequency of a sample channel; DMT bins in the areas where the SNR is high can use more dense QAM constellations.



Compared to the bins when the SNR is low, the bins occupying parts of the channel where the SNR is high can be used to transmit more bits. The process involves increasing the number of points used in the constellations of the “good” bins. If you plot the information sent as a function of frequency for this channel using DMT, the resulting curve takes on a shape similar to that shown in Figure 5.6, thus showing the optimal utilization of the channel.

DMT also allows a simple method to increase or decrease the output PSD of a transmitter in selected frequency regions. The process involves scaling the complex valued vector for the bin where power adjustment is desired before performing the IFFT. Such adjustment can boost power in regions where the channel insertion loss is poor, or reduce power in regions where interference with other systems must be avoided. Note that power boosts and reductions can be done in a CAP/QAM system also with external shaping filters, although these tend to be complex and can be difficult to implement for arbitrary shaping requirements.

Note

A classic example is the reduction of VDSL's PSD in certain bands to avoid interference with amateur radio or emergency bands.

Forward Error Correction

A forward error correction (FEC) block adds redundancy to the data to be transmitted. Usually this redundancy is a small fraction of the actual transmitted payload. The value of adding the redundancy is that the FEC block might be able to correct bits that the demodulator decodes incorrectly. The power of a forward error correction technique is often described using the term *coding gain*. A reasonable coding gain for a DSL forward error correction technique may be 3 dB at a bit error rate (BER) of 10^{-7} . Stated another way, adding the FEC to the system makes the system perform as well as a system without coding but with twice the transmit power (3 dB more transmit power). This attribute is indeed formidable.

Two types of forward error correction are common in DSL systems. The first type is a cyclic block code known as a *Reed-Solomon coding*. The second is a *convolution code* known as *trellis-coded modulation* (TCM) and ties in tightly with the constellation encoding used in both CAP/QAM and DMT modulation. The following sections discuss the basics of these two techniques. Detailed treatment of these subjects can be found in various books, including *Digital Modulation and Coding* and *Error Control Coding: Fundamentals and Applications*.^{2,3}

Reed-Solomon Coding

Reed-Solomon coding was first described around 1960.⁴ This type of coding is a subset of cyclic coding, which in turn is a subset of block coding. Reed-Solomon codes are also a special case of nonbinary BCH codes named for Bose and Ray-Chaudhuri and Hocquenghem who independently discovered them around 1960.^{5,6} Many readers may be familiar with *cyclic redundancy check codes* (CRC codes) used in data transmission. Because CRC codes are simple and popular, they will be used as a starting point in describing Reed-Solomon codes that are typical in DSL systems.

CRC codes are typical in almost all types of layer 2 protocols including high-level data link control (HDLC)-based schemes and Ethernet. Normally, these codes operate on bits and are used for error detection but not error correction. Additionally, the number of bits being covered by the check bits is variable.

Note

Operating on bits infers that modulo 2 arithmetic is used for coding and checking. It can also be said that the arithmetic is done in GF(2).

Reed-Solomon codes used in DSL systems use *galois field arithmetic*. Specifically, they operate on GF(256), meaning that each symbol can take on one of 256 values (0 through 255). Thus each symbol is made up of eight bits—one byte long. The arithmetic rules in GF(256) define the results when two symbols are added, subtracted, multiplied, or divided. It turns out that in GF(256) carrying is not done in the same way that it is done in the base-10 decimal system.

A Reed-Solomon code produces codewords made up of a fixed number of data bytes and a fixed number of check bytes. When operating in GF(256), the total number of bytes in a codeword must be less than 255. Using different codes can vary the number of data bytes and check bytes making up the codeword. A typical DSL implementation might use a codeword size of 240 bytes, 224 of which are data bytes and 16 of which are check bytes. Figure 6.10 shows a Reed-Solomon codeword with these properties.

Figure 6.10 Example of a Reed-Solomon codeword in GF(256).

DATA PAYLOAD 224 bytes	CHECK BYTES 16 bytes
---------------------------	-------------------------

Reed-Solomon codes allow for error correction in the receiver. Note that in the receiver, the Reed-Solomon decoder block in the previous example would receive 240 bytes from the demodulator and output 224 bytes (only the data bytes) to the rest of the receiver. In

general, the number of symbols that a Reed-Solomon decoder can correct is half the number of check bytes used. Thus for the example of 16 check bytes, eight byte errors would be correctable even if these bytes all occur in a row (a burst error). This correction is an extremely powerful benefit for very little overhead (6.6% in this example).

Convolution Codes

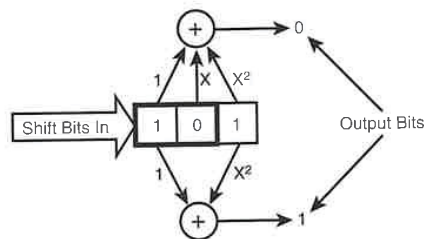
Convolutional coding differs from block coding in that the former does not have code-words made up of distinct data sections and block sections. Instead, redundancy is distributed throughout the coded data. Convolutional codes are sometimes referred to as *trellis codes*. Normally, convolutional encoding is simple, and decoding is a bit more difficult.

Note

Trellis codes can be linear or nonlinear. Convolutional codes are a subset of linear trellis codes.

Figure 6.11 shows a simple convolutional encoder.

Figure 6.11 A simple, rate 1/2 convolutional encoder; note that shifting in one bit produces two output bits.

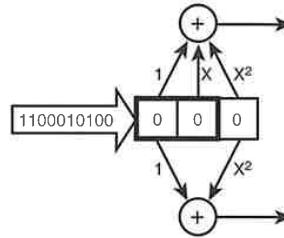


Data bits are shifted into the encoder from the left. Each data bit shifted in produces two data output bits defined by the modulo 2 additions shown. The outputs are defined by generator polynomials. This simple convolutional encoder is said to be a rate 1/2 encoder with a constraint length 2. The rate refers to the ratio of input bits to output bits, and the constraint length refers to the number of delays in the encoder. These parameters, along with the generator polynomials, determine the processing gain of the code.

Note that the first two positions in Figure 6.11 have a darkened outline. The values in these positions are called the *state of the encoder*. The convolutional encoder can be viewed as a finite state machine, changing state based on an input condition and producing an output when it changes state.

Figure 6.12 shows a simple example of using the encoder in Figure 6.11 to encode a set of bits. Note that the initial condition of the encoder is assumed to be all zeros and that the top output is taken before the bottom output.

Figure 6.12 Input and output of the sample convolutional encoder.



Input	0	0	1	0	1	0	0	0	1	1						
Output	0	0	0	0	1	1	1	0	0	0	1	0	1	1	0	1

Each node in the trellis represents a state. Each line represents a pair of received bits. Note that each state has two lines leading to the state and two lines leaving it. When two bits are received, a transition occurs between the current state and the new state. The type of line connecting the current and new state (solid or dotted) determines whether the decoder will output a zero or a one in response to the two bits received. Note that two bits in produce one bit out—just the opposite of the encoder.

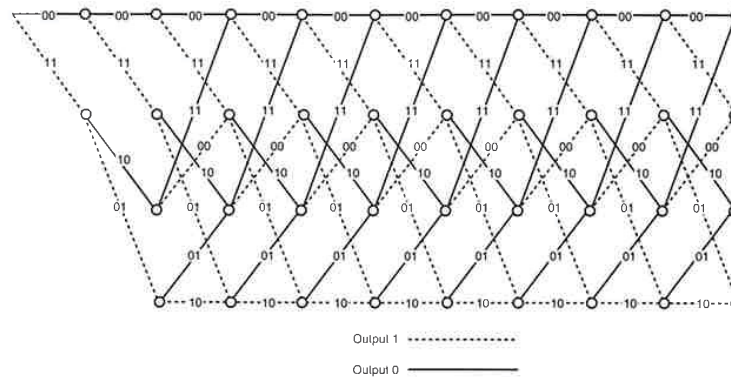
Also in Figure 6.13, note that the numbers on each line represent the input to the receiver that will cause the state transition. The type of line (solid or dotted) represents the decoder output.

Decoding of a convolutional code can be done with a trellis and the *Viterbi algorithm*. The trellis in Figure 6.13 can be used for this encoder example.

If no errors occur, the bits received while in a state correspond to one of the paths leaving the state. When errors occur on the line, the received bits may correspond to neither of the lines leaving the state and knowing which line to follow is unclear. The decoding algorithm allows for multiple paths through the trellis to be kept active. By tallying the number of mismatches (errors) that have occurred along each path, the decoding algorithm terminates those with a high error count. The error count is called a *metric*. As time passes

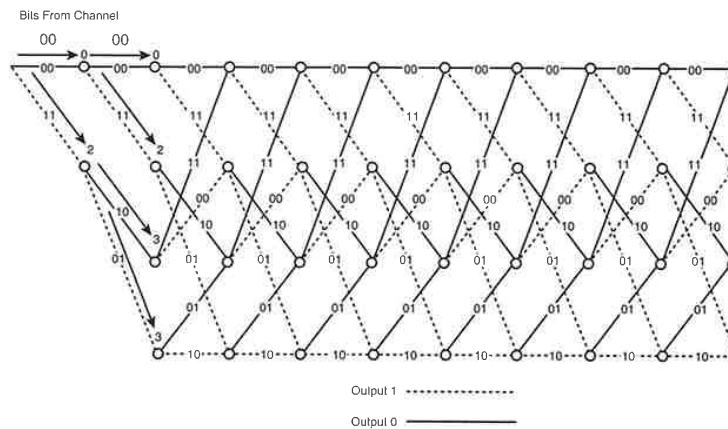
and the decoding process continues, a path with a low metric develops, and all others are terminated. This path is then used to produce the output of the decoder.

Figure 6.13 A trellis that can be used to decode the sample convolutional encoder: each node represents one of the four states.



An example of decoding is the best way to understand the decoding algorithm. Consider the coded output from Figure 6.12. The decoding algorithm for the first four bits is shown in Figure 6.14.

Figure 6.14 Trellis decoding example; the first four received bits are zeros and begin the decoding process.



The received bits are shown above the trellis (00, 00). For these four bits, all paths in the trellis are kept alive (this step is just to get the algorithm started). At each node, a metric is computed that counts the mismatches between the received bits and the path bits. (Note that the metric is a running sum for each path.) At this point, the paths are extended using the following rules:

1. Two lines will lead to each node in the next set of states.
2. For each node, follow only one line to each node, specifically, the line having the lowest metric.

Note

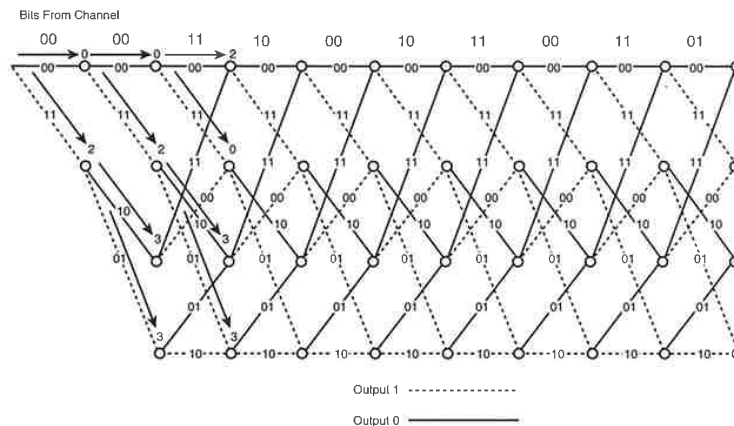
The example being discussed uses hard-decision decoding. Another method, using soft decisions, does not completely decode the received bits before the trellis, but instead assigns a metric to each received bit, indicating how close to a one or zero it is. Soft decoding can outperform hard decoding by several dB.

3. Terminate the other path that would arrive at that node.

Note that four paths always remain active. Ideally, one main path develops through the trellis, and all others terminate. In actual practice, some delay is necessary to identify the correct path.

Turning back to the example, consider the next two bits being received. This situation is shown in Figure 6.15.

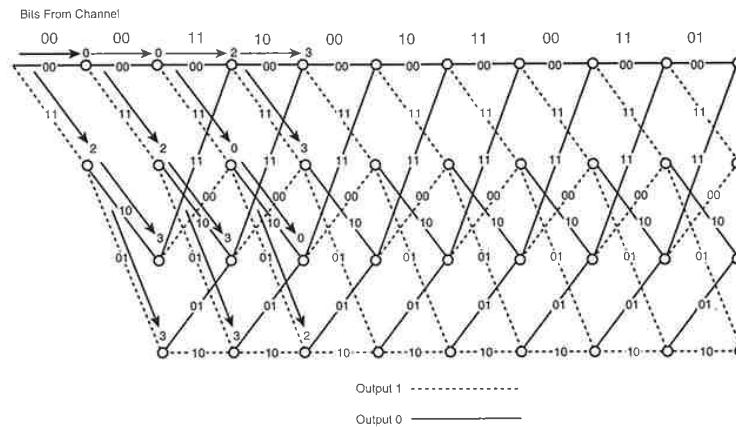
Figure 6.15 Example of moving along the trellis—only one of the two lines leading each state is followed.



The previous rules are now used to decide which paths will go forward. For example, for the top node, the path through the line marked 00 was followed instead of the path through the line marked 11. The former produced a metric of two, whereas the latter would have produced a metric of three. Thus the first is followed, and the second terminated.

Figure 6.16 shows the next step in the decoding process.

Figure 6.16 An example of moving along the trellis; note that one path (which we will eventually find to be the correct path) still has a metric of zero.



Notice that the new metrics are simply the previous metrics with a number of new mismatches added. The best metric still has a value of zero, indicating that it is a good candidate for the correct path. Figure 6.17 shows the result of several more iterations.

Note that many paths along the way have been terminated and that only one main path gives rise to all the existing paths. This main path is highlighted in Figure 6.18, and the decoded output for each set of inputs appears at the bottom of the trellis.

These outputs match the first few bits at the input to the decoder. As time passes and decoding continues, the main path will extend farther into the trellis and more outputs are decoded.

In Figure 6.18, the path can be decoded only up to the point where the main path diverges. When a surviving path is chosen up to a certain point, the “bad” paths up to that point are all discarded. Future input bits will be necessary to decode beyond this point.

Figure 6.17 Trellis appearance after several iterations from the example.

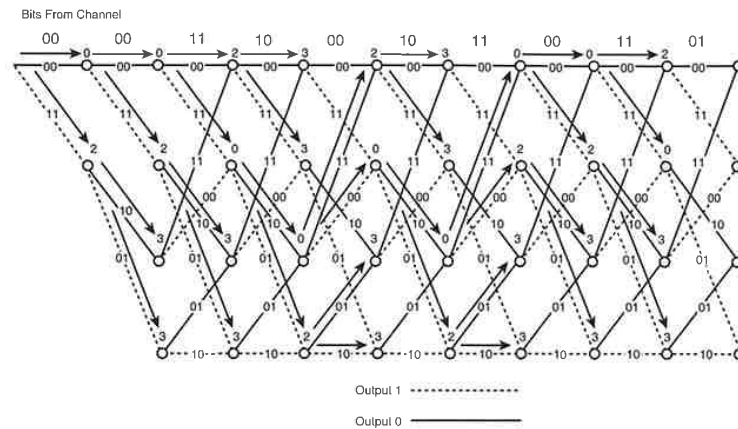
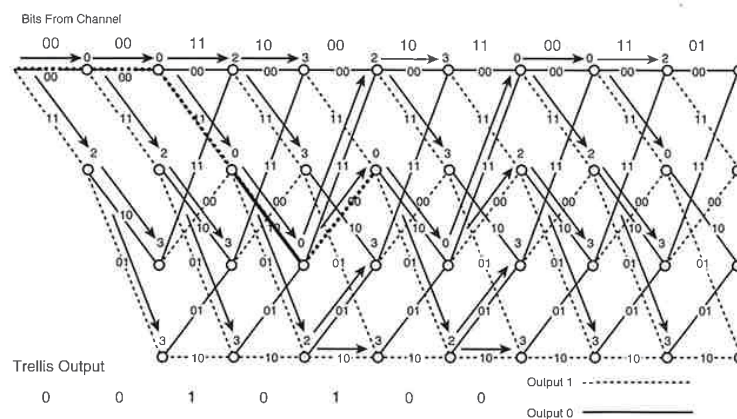


Figure 6.18 The decoded output of the trellis.



The preceding example considered decoding when no errors were made on the channel; thus error correction was not necessary. Consider now the case in which an error is made on the channel and one of the bits is incorrectly received. Figure 6.19 shows the first three steps of this decoding process.

Although none of the metrics is zero, the same rules are used to move through the trellis. The example is continued for several more iterations in Figure 6.20.

Figure 6.19 First few iterations for the sample channel with an error in the received bits; note that none of the metrics is zero in this case.

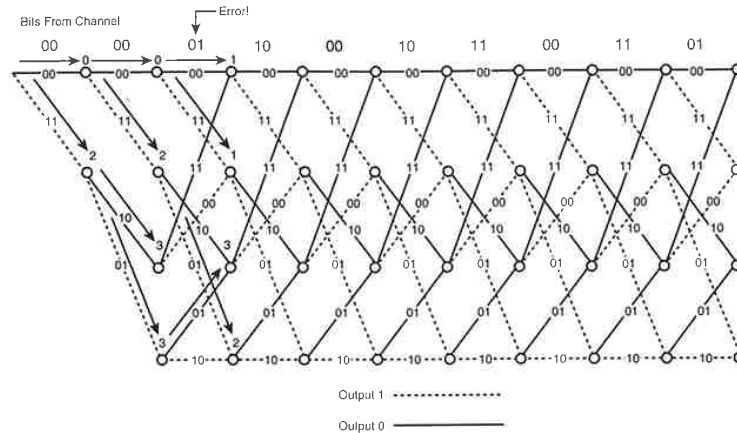
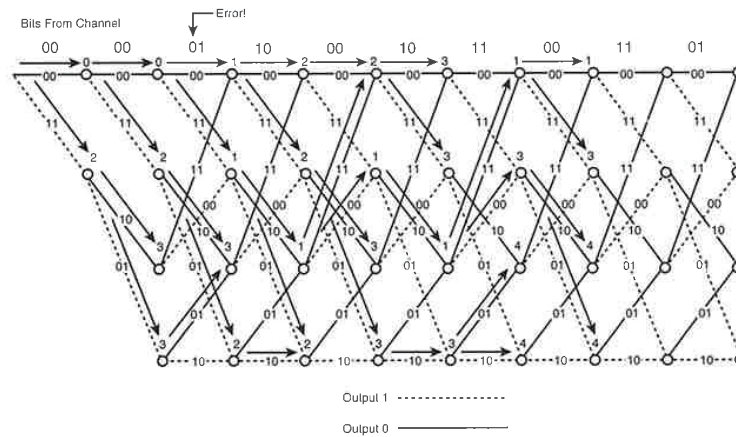


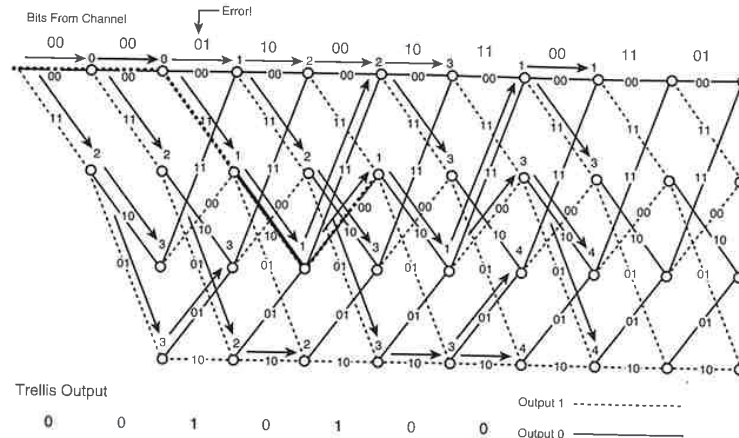
Figure 6.20 More iterations for a channel with received bit errors; note the paths that have terminated due to high metrics.



In Figure 6.20, many of the metrics have grown, but one is still small. Also, note that up to a certain point in the trellis all paths have terminated except one (the one with the lowest metric). This path is darkened in Figure 6.21, and the output shown at the bottom of the trellis.

Figure 6.21

Decoding of the trellis; note that even with the error, only one main path survives and decodes with the proper values.



Even with the bit error that occurred on the line, the output of this example produced the correct sequence (identical to the preceding example), illustrating the forward error correcting mechanism of convolutional codes. In all, it is fair to say that bad paths are eventually terminated and a surviving path, even when some errors occur, is eventually apparent.

Note

Trellis decoding can also be compared to following directions to someone's house. If you make a **wrong turn at some point**, the rest of the directions most likely do not make sense; however, it is **possible to retrace your route** to the point at which the directions stopped making sense.

Convolution coding may also be combined with modulation. When combined with QAM (as well as other types of modulation, such as pulse amplitude modulation [PAM]), the result is *trellis-coded modulation*. Normally, a subset of the bits (one or two) making up a QAM symbol is passed through an encoder. The uncoded bits usually determine the quadrant or quadrant portion a point will be in. The coded bits are the least significant bits distinguishing a point from neighboring points. Coding the least significant bits adds to the likelihood that a constellation point will not be incorrectly decoded as a neighboring point, the most common type of error occurrence in QAM. TCM is analyzed in many publications, most notably G. Ungerboeck's 1987 article in *IEEE Communications Magazine*, vol. 25, no. 2, entitled "Trellis Coded Modulation with Redundant Signal Sets—Part I: Introduction."⁷

Interleaving

Many FEC blocks have trouble when it comes to correcting long strings of errors. Such errors are common on channels and can occur on DSLs (for example, when a transient spike is incident upon the twisted pair). The purpose of interleaving is to spread out a codeword such that channel burst errors are also spread out. Interleaving often occurs between a block forward error correction module and a modulation module in a transmitter. The corresponding block in the receiver is a “de-interleaving” block. The following sections describe two types of interleavers: a block interleaver and a convolutional interleaver. Block interleavers are simple and easy to understand; however, convolutional interleavers are more common in DSLs, as they have delay and memory requirement advantages.

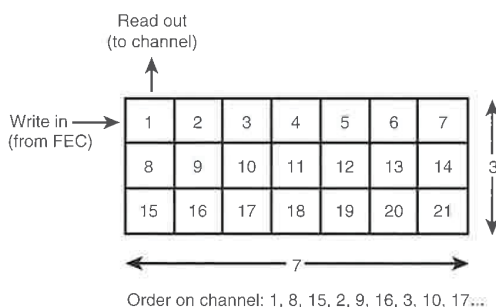
Note

A poorly filtered ringing burst or ground bounce due to lightning or other disturbances can produce such a transient spike.

Block Interleaving

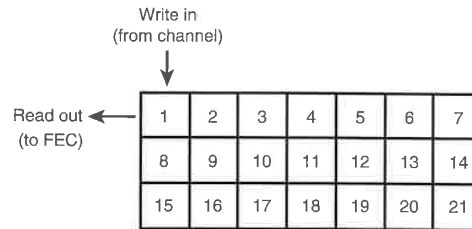
Figure 6.22 shows an example of a block interleaver—specifically, a block interleaver with depth $D=3$ and span $N=7$.

Figure 6.22 A block interleaver with depth 3 and span 7.



The numbers in the blocks represent the order in which bits would enter the interleaver. In general, bits are written in rows and read out in columns. Normally, a single row contains a complete FEC codeword. Thus the codeword length for this example would be 7. Figure 6.23 shows a de-interleaver with $D=3$ and $N=7$.

Figure 6.23 A de-interleaver with depth 3 and span 7; the output of the de-interleaver is the proper order of the symbols.



Here the bits coming from the channel are written into the de-interleaver in columns and read out in rows. The rows are then properly ordered bits and are normally destined for an FEC block.

Table 6.1 shows the value of interleaving.

Table 6.1 Comparison on burst errors with and without interleaving

Bit Order (No Interleaving)	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
Bit Order (With Interleaving)	1	8	15	2	9	16	3	10	17	4	11	18	5	12	19	6
Error Burst																
Received Bits (No Interleaving)	1	2				6	7	8	9	10	11	12	13	14	15	16
Received Bits (De-interleaved)	1		3	4	5	6	7	8		10	11	12	13	14		16

The first two rows of Table 6.1 illustrate the order in which these bits will be sent over the channel both without interleaving and with the interleaver from Figure 6.22. If a burst error occurred in the channel as shown in the third row of Table 6.1, note the bit numbers being corrupted in each case. The last two rows of Table 6.1 show the bits that would be sent to the FEC block in the receiver on the sample channel. Note that when interleaving is used, the bit errors are broken up, giving the FEC block a better chance to correct the errors. This example might be used on a channel where burst errors are not expected to be longer than three bit periods. A more practical interleaver would normally have larger D and N parameters.

Interleaving adds delay to the end-to-end transport of data and also requires memory buffers at both the transmitter and receiver. In general, both the transmitter and receiver need about $D \times N$ bits of memory space to support block interleaving and a delay of about $2DN$ bits is inherent. For a trivial example like the one in Table 6.1, such negative aspects

may seem insignificant, but for larger interleaver depths and spans, they are very significant.

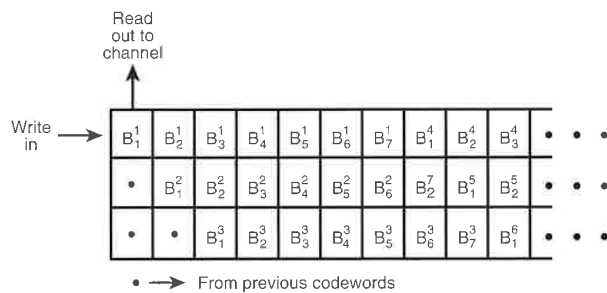
Note

For higher-layer protocols that use acknowledgments, such as TCP, the added delay can cause the protocol to stall, drastically reducing the data throughput.

Convolutional Interleaving

Convolutional interleaving is more efficient than block interleaving with respect to memory requirements and end-to-end delay. Note that convolutional interleaving does not imply anything about the FEC type, and it can function with either block or convolutional FEC techniques. A convolutional interleaver with a codeword size $N=7$ and a depth $D=3$ appears in Figure 6.24.

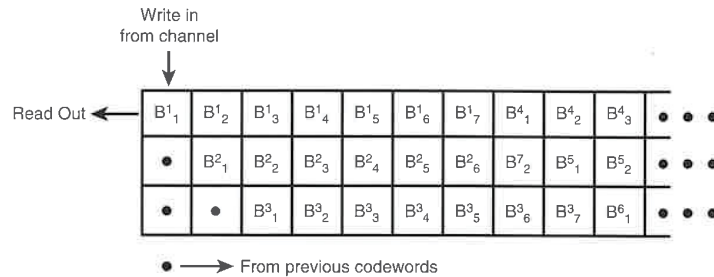
Figure 6.24 A convolutional interleaver with depth $D=3$ and codeword size $N=7$.



In this example, the superscripts of B represent the codeword to which B belongs. The subscripts represent the bit number within that codeword. The codewords are again written into the interleaver in rows and read out in columns. The difference between this scheme and the block interleaver scheme is that codewords do not always start in the same column in the convolutional interleaver as they do in the block interleaver. In addition, the rows do not end. The depth and length of the interleaver decide whether the next codeword should be written into the next row or in the top row immediately following the codeword already written there.

Figure 6.25 shows a convolutional de-interleaver for use in a receiver when a transmitter uses a convolutional interleaver.

Figure 6.25 A convolutional de-interleaver with depth $D=3$ and length $N=7$; reading values out of the de-interleaver is not as straightforward as for the block de-interleaver.



Here bits are written in columns and read out in rows for further processing by an FEC block. The de-interleaver must read only a single codeword from each row and then proceed to the next row until the last row is read. After reading the last row, the de-interleaver returns to the first row and reads from the next unread position.

Convolutional interleaving can distribute a codeword of length N over an interval of ND , incurring an end-to-end delay of ND bits.

Note that although the interleaving examples given are in terms of bits (the depth and codeword size are in terms of bits), many coding schemes work at the byte level (for example, Reed-Solomon codes operating in $GF(256)$), or even more generally, at the symbol level. Interleaving schemes employed along with FEC systems operating at the byte or symbol level normally operate at the byte or symbol level as well. For example, assuming a codeword size of seven bytes, a payload stream is interleaved at the byte level, and the end-to-end delay of a convolutional interleaver and de-interleaver is ND bytes.

Note that other types of interleaving are also possible. For example, helical interleaving is a different interleaving scheme.⁸

Scrambling

Most DSLs include a scrambler in the transmitter and a descrambler in the receiver. A scrambler decreases the probability that a long string of ones or zeros are passed to the modulator. Such strings may occur in either a packet-based system or an ATM system when no packets or cells exist to transmit and the input to the transmitter is held either high or low. Often it is said that a scrambler randomizes data and a descrambler removes the randomness.

Note

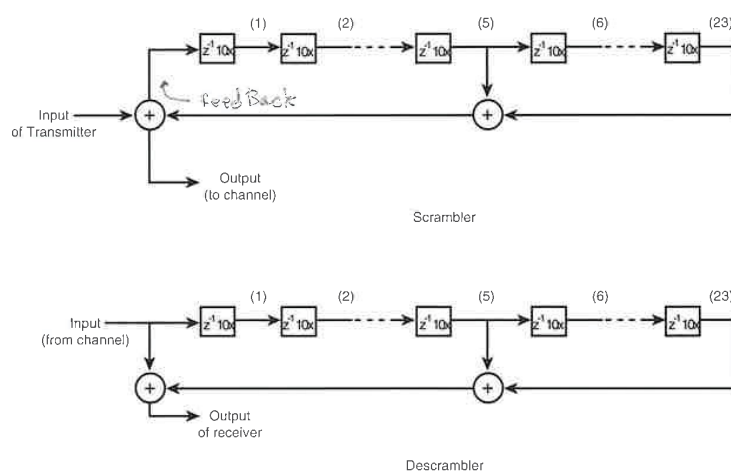
T1 lines do not include scramblers; however, ADSL, VDSL, HDSL, and ISDN all use scrambling in the transmitter and descrambling in the receiver.

Scrambling has two positive benefits in a communications system. First, it helps ensure that the transmitted power spectral density observed at the output of the transmitter is predictable. When deriving equations for PSD in Chapter 4, "Power Spectral Densities and Crosstalk Models," an assumption was made about the statistics of the transmitter (for example, for ISDN it was assumed that all quats were equally probable). Scrambling helps to make such assumptions valid.

Second, many types of circuits or algorithms depend on a mix of ones and zeros to operate properly. Timing recovery phase locked loops and some adaptive equalization algorithms are good examples of these. Scrambling helps ensure that such blocks function properly. Note that scrambling in a transmitter is typically not used for data encryption or security purposes. The means of descrambling received data at the receiver are much too simple to be effective for security applications.

Figure 6.26 illustrates a scrambler and a descrambler.

Figure 6.26 An example of a scrambler and descrambler; the z^{-1} blocks represent a one-bit delay, and all addition is modulo 2.



Note that the scrambler is a recursive device, whereas the descrambler does not include feedback. The pair shown are characterized by a generating polynomial $g(x)$, which in the case of Figure 6.26 is given by Eqtn. 6.26.

Eqtn. 6.26

$$g(x) = 1 + x^{-5} + x^{-23}$$

This popular scrambler is used in several communications standards. The coefficients in Eqtn. 6.26 indicate the tap locations at which the scrambler performs an exclusive OR function. The generating polynomial of a scrambler is usually chosen to produce a *maximum length sequence* when the input to the scrambler is constant ones or constant zeros. Assuming the proper coefficients are chosen, longer scramblers normally produce longer maximum length sequences.

Note

Assuming the initial taps of a scrambler are not all zero, the maximum length sequence of a scrambler is given by $2^n - 1$, where n is the order of the scrambler's generating polynomial.

The scrambler shown in Figure 6.26 is called a *self-synchronizing scrambler*. This name reflects the fact that the descrambler can begin in an arbitrary state (the initial values inside the scrambler) and, after a finite amount of time, begin to output the correctly unscrambled data. In addition, a bit error on the channel will not catastrophically affect the descrambler or cause it to significantly multiply the number of errors produced. The lack of feedback in the descrambler is directly related to these properties, as feedback can cause a bit error to infinitely affect the operation of the block.

A bit error on the channel is somewhat enhanced by a descrambler. In general, a single bit error is increased by a factor equal to the number of nonzero taps in the descrambler. In the example shown in Figure 6.26, a single bit error at the input produces three bit errors at the output. (More precisely, a bit error at the input produces an output error as it propagates to all of the nonzero tap positions.) This event is often termed the *error propagation factor*. Such increases in errors are often deemed acceptable when traded off against the many benefits of scrambling.

Equalization

Most modern communications systems that operate near theoretical limits employ equalization in the transmitter, receiver, or both to optimize or nearly optimize transmission. Often the equalization is done digitally by adaptive digital filters. This approach provides a very flexible way to accommodate different types of channels and different types of noise environments. In most DSLs (as well as in voice-band modems), the adaptive filters converge to optimal initial settings during a training period and then can be updated during

normal runtime operation of the system. Training can also be done without a predetermined training phase in what is called *blind equalization*.⁹

Note

Updating is necessary as many channels and noise conditions change over the lifetime of operation. A good example is the change in twisted pair characteristics due to the difference of temperature during the day and night.

Any time a channel's frequency response is not flat over the range of frequencies being transmitted, *intersymbol interference* (ISI) can occur. A channel with a frequency response that is not flat is sometimes called a *nonflat channel*, a *dispersive channel*, a channel with memory, or simply a channel with ISI. Additionally, a nonflat channel has an impulse response that is nonzero at more than one point.

Note

A flat channel will have an impulse response that is simply an impulse. Such a channel is often called a *memoryless* channel.

Figure 6.27 shows an impulse response of a channel with ISI. Note that an impulse in at time 0 produces an output with components over a range of times.

Figure 6.27 Response of a channel with ISI to an impulse.

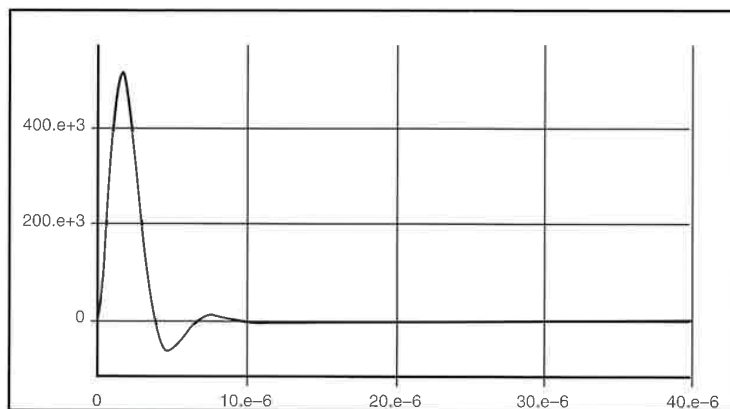
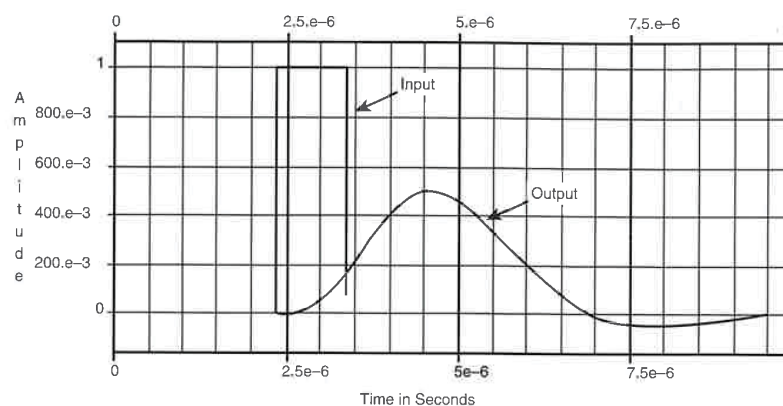


Figure 6.28 shows a single pulse transmitted through this channel.

Figure 6.28 Output of a channel with ISI when excited by a pulse at the input.

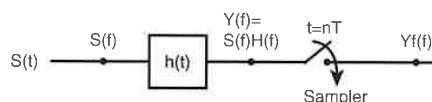


Note that the pulse is spread out and distorted at the output of the channel. Such spreading would interfere with the next pulse transmitted. Also, a previously transmitted pulse would interfere with the one shown. A channel can have even more memory than the one shown, causing pulses or symbols to be spread out and interfere not only with immediately adjacent symbols but also with several or many of them. The different types of equalization discussed in the next few sections are mainly intended to remove some or all of this ISI so that optimal decisions can be made on incoming symbols in the receiver. The following analysis is done in various domains, depending on which domain is more intuitive for the explanation. Keep in mind that transforms can be used to find equivalent duals in different domains.

Folded Spectrums

Before analyzing different types of equalizers, a quick discussion of signal spectrums and the effect of sampling on signal spectrums is in order. First, consider a signal $s(t)$ with a frequency spectrum of $S(f)$ passing through an LTI system $h(t)$ with spectrum $H(f)$. This is shown in Figure 6.29.

Figure 6.29 A channel with sampling at the output, illustrating spectrum folding.



Working in the frequency domain, the output of the system $Y(f)$ is simply given by the multiplication of the input spectrum and transfer function section as shown by Eqtn. 6.27.

Eqtn. 6.27

$$Y(f) = S(f)H(f)$$

When the output is sampled at some frequency f_s , the output samples are denoted by y_i and the output frequency spectrum is denoted as $Y_f(f)$. This spectrum is a *folded spectrum*. When working with two-sided PSDs, the equation defining the spectrum $Y_f(f)$ appears as Eqtn. 6.28.

Eqtn. 6.28

$$Y_f(f) = \sum_{n=-\infty}^{\infty} Y(f + nf_s) \quad \text{for } |f| < \frac{f_s}{2}$$

Note that if $Y(f)$ does not have frequency components at frequencies higher than $f_s/2$, then the spectrum of $Y_f(f)$ will be the same as that for $Y(f)$. Normally, when a folded spectrum is computed, it is computed only for small n values ($-2, -1 \dots 2$) because these components dominate in the calculation.

Note

Put another way, in this case aliasing does not occur because the signal was sampled at a rate at least two times higher than the highest frequency component it contains.

Folded spectrums are sometimes normalized with respect to frequency in discussions when the frequencies being discussed are not known or are generalized. This step is done by recognizing that in the frequency domain, the radian-sampling frequency $2\pi f_s$ is equal to a complete period. The frequency variable f is then replaced by the quantity $\theta f_s/2\pi$ where θ is the normalized frequency variable. The normalization of a folded frequency spectrum appears in Eqtn. 6.29.

Because f was defined on $(-f_s/2, f_s/2)$, θ is defined on $(-\pi/2, \pi/2)$. Folded spectrums are revisited in later sections on equalization.

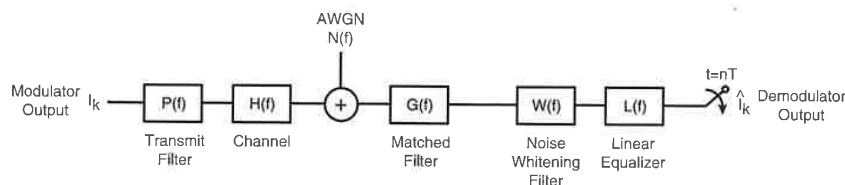
Eqtn. 6.29

$$\begin{aligned}
 Y_f(f) &= \sum_{n=-\infty}^{\infty} Y(f + nf_s) \\
 &= \sum_{n=-\infty}^{\infty} Y\left(\frac{\theta f_s}{2\pi} + nf_s\right) \\
 &= \sum_{n=-\infty}^{\infty} Y\left(\frac{\theta}{2\pi\tau} + \frac{n}{\tau}\right) \\
 &= \sum_{n=-\infty}^{\infty} Y\left(\frac{\theta + 2\pi n}{2\pi\tau}\right) \quad |\theta| < \pi
 \end{aligned}$$

Linear Equalization

A linear equalizer (LE) is typically placed in a receiver in front of the demodulator. To analyze an LE, consider the system shown in Figure 6.30.

Figure 6.30 Simple block diagram for LE analysis.



Note: $P(f) H(f) M(f) = G(f) G^*(f)$

In the description that follows, some of the filtering functions may be combined, resulting in less complex implementations than actually implementing several filters in cascade. However, for explanatory purposes, it is convenient to discuss the filters as separate entities. In the following discussion, assume that all filters may be implemented digitally and that the sample frequencies are large enough to prevent folding (the appreciable frequencies of all signals are much less than $f_s/2$). However, folding may take place when sampling the output of the system.

It is well known that the front end of any optimal receiver is made up of a *matched filter* followed by some type of processing. The matched filter $M(f)$ is typically matched to the convolution of the channel response and the excitation pulse response (the shaping filter). Eqtn. 6.30 gives the response of the matched filter in Figure 6.30.

Eqtn. 6.30

$$M(f) = [H(f)P(f)]^*$$

The filter following the matched filter in Figure 6.30 designated by $W(f)$ is called a *noise-whitening filter*. The purpose of this filter is to remove any memory in the noise output of the matched filter before further processing. Eqtn. 6.31 gives the typical frequency response of this filter.

Eqtn. 6.31

$$W(f) = \frac{1}{G^*(f)}$$

$G(f)$ and $G^*(f)$ are factors of the transfer function between the modulator and noise-whitening filter input including the channel, the shaping filter, and the matched filter. This is reflected in Figure 6.30. $G(f)$ and $G^*(f)$ are chosen such that $G^*(f)$ has minimum phase so that $W(f)$ is realizable.

The symbol $*$ in Eqtn. 6.30 and Eqtn. 6.31 denotes the complex conjugate operator. A linear equalizer would then follow the noise-whitening filter.

You can use two techniques to optimize the response of a linear equalizer.* The first optimization technique results in the *zero-forcing* (ZF) linear equalizer. The second optimization technique results in the *mean squared error* (MSE) linear equalizer.

The zero-forcing LE is optimized with respect to the *peak distortion criterion*. This criterion attempts to force the output of the equalizer to have no residual ISI. For a sufficiently long equalizer, the result of the peak distortion criterion is an equalizer with a response given by Eqtn. 6.32.

Eqtn. 6.32

$$L(f) = \frac{1}{G(f)}$$

Note that the net result of the noise-whitening filter and the ZF equalizer is the inverse of the channel up to the input of the noise-whitening filter (an added processing delay is not discussed but would also be present). Assuming the baud rate of the system is τ , the SNR resulting from a zero-forcing linear equalizer is found by Eqtn. 6.33.

Eqtn. 6.33

$$\begin{aligned}
 SNR_{ZFLE} &= \frac{Signal_{ZFLE}}{Noise_{ZFLE}} \\
 &= \frac{\int_{-\frac{1}{2\tau}}^{\frac{1}{2\tau}} S_f(f) df}{\int_{-\frac{1}{2\tau}}^{\frac{1}{2\tau}} \frac{N_o}{X_f(f)} df}
 \end{aligned}$$

In this equation N_o is the PSD of the noise process $N(f)$, and $S(f)$ is the PSD of the signal power at the output of the modulator. Note that the spectrums shown are folded spectrums sampled at the baud rate of the system (denoted by the subscript f). The folded spectrum $X_f(f)$ is given by Eqtn. 6.34.

Eqtn. 6.34

$$X_f(f) = \sum_{n=-\infty}^{\infty} \left| P\left(f + \frac{n}{\tau}\right) H\left(f + \frac{n}{\tau}\right) \right|^2$$

Note from Eqtn. 6.33 that when the folded spectrum $X_f(f)$ contains a null or some value close to zero, the SNR of the zero-forcing linear equalizer becomes very small. This artifact of the ZF equalizer occurs because, at frequencies with significant channel loss, the equalizer tries to compensate by introducing high gain. This gain enhances the background noise, causing the SNR to approach zero.

The MSE linear equalizer minimizes the mean square error energy at its output. This type of equalization can trade off the amount of error caused by ISI with the amount of error caused by background noise. For a sufficiently long equalizer, the optimal response of the MSE case is shown in Eqtn. 6.35.

Eqtn. 6.35

$$\begin{aligned}
 L(f) &= \frac{G^*(f)}{G(f)G^*(f) + N_o} \\
 &= \frac{G^*(f)}{P(f)H(f)M(f) + N_o}
 \end{aligned}$$

In this equation, N_o is the variance of the white noise process $N(f)$. Note that when the noise is negligible, the MSE linear equalizer becomes identical to the ZF linear equalizer

(given by Eqtn. 6.32). The residual mean squared error resulting from the MSE linear equalizer is shown in Eqtn. 6.36.

Eqtn. 6.36

$$J_{MSE} = \int_{-\frac{1}{T}}^{\frac{1}{T}} \frac{N_o}{X_f(f) + N_o} df$$

The resulting SNR at the output of the MSE equalizer is shown in Eqtn. 6.37.

Eqtn. 6.37

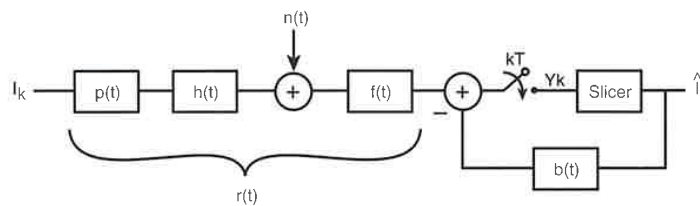
$$SNR_{MSE} = \frac{1 - J_{MSE}}{J_{MSE}}$$

It should be evident from Eqtn. 6.36 and Eqtn. 6.37 that the MSE linear equalizer does not perform well in environments where the folded spectrum of the channel up to the receiver has nulls, because the error increases as the SNR decreases. Such nulls are common in many channels, including twisted pairs. The performance problems of linear equalizers on these channels provide motivation to discuss other types of equalization in subsequent sections. However, linear equalizers do play a role in these other types of equalization and are used in many different receivers.

Decision Feedback Equalization

Performance problems of the linear equalizer have motivated the search for some type of nonlinear equalization technique. The *decision feedback equalizer* (DFE) is one such technique. A DFE is made up of a *feed-forward filter* and a *feedback filter*. Note that each of these filters is normally a linear filter, but they are used together in a nonlinear fashion. The DFE is illustrated in Figure 6.31.

Figure 6.31 A decision feedback equalizer on a communications channel.



The slicer shown in Figure 6.31 is the decision-making device. Thus the input to the slicer is an estimate of the symbol to be decoded, and the output is the actual signal nearest to the estimate. The slicer is the nonlinear component of a DFE. Note that the output of the slicer goes into the feedback filter. The output of the feedback filter is subtracted from the received signal. In effect, the feedback section of the DFE removes ISI from a symbol due to previously transmitted symbols. This procedure is sometimes termed postcursor ISI. Assuming that the slicer makes the correct decisions, the feedback filter provides a way to remove the postcursor ISI without significantly amplifying noise (a major shortcoming of linear equalizers).

Note

Of course, making the assumption that the decisions are made by slicer is not always correct. However, when the probability of error is low (less than 10^{-7}), this assumption allows for a simpler analysis of the performance of a DFE. The result stemming from this ideal assumption is called the *ideal DFE bound*.

To mathematically illustrate the operation of the feedback filter, consider an optimization technique minimizing the mean squared error at the output of the slicer. The expected value of the error can be written as in Eqtn. 6.38.

Eqtn. 6.38

$$J_{DFE} = E|Y_k - I_k|^2$$

If we let $r(t)$ represent the combined response of the pulse-shaping filter, the channel, and the feed-forward filter, then the input to the slicer is made up of three terms. These terms are the signal component exiting the feed-forward filter, the noise component exiting the feed-forward filter, and the signal from the feedback filter. Thus we can write Y_k as in Eqtn. 6.39.

Eqtn. 6.39

$$\begin{aligned} Y_k &= I_k * r(t)|_{t=kT} + n(t) * f(t)|_{t=kT} - \sum_{n=1}^{\infty} b_n I_{k-n} \\ &= \sum_{n=-\infty}^{\infty} r_n I_{k-n} - \sum_{n=1}^{\infty} b_n I_{k-n} + n(t) * f(t)|_{t=kT} \\ &= \sum_{n=-\infty}^{-1} r_n I_{k-n} + \sum_{n=1}^{\infty} (r_n - b_n) I_{k-n} + r_0 I_k + n(t) * f(t)|_{t=kT} \end{aligned}$$

In Eqtn. 6.39, the third line simply results from distributing the first summation into the second and third terms. If we then substitute Eqtn. 6.39 into the mean squared error equation, we get the result shown in Eqtn. 6.40.

Eqtn. 6.40

$$\begin{aligned}
 J_{MSE} &= E \left| \sum_{n=-\infty}^{-1} r_n I_{k-n} + \sum_{n=1}^{\infty} (r_n - b_n) I_{k-n} + r_0 I_k + n(t) * f(t) \Big|_{t=kT} - I_k \right|^2 \\
 &= E \left| \sum_{n=-\infty}^{-1} r_n I_{k-n} + \sum_{n=1}^{\infty} (r_n - b_n) I_{k-n} + I_k (r_0 - 1) + n(t) * f(t) \Big|_{t=kT} \right|^2 \\
 &= E \left| \sum_{n=-\infty}^{-1} r_n I_{k-n} \right|^2 + E \left| \sum_{n=1}^{\infty} (r_n - b_n) I_{k-n} \right|^2 + E |I_k (r_0 - 1)|^2 + E |n(t) * f(t) \Big|_{t=kT}|^2
 \end{aligned}$$

Note in Eqtn. 6.40 that the expected value of all multiplication cross-terms goes to zero due to independence, and thus, the multiplication cross-terms are not shown on the last line. Assuming that σ_I^2 is the variance of I_n and $N_o/2$ is the variance of the noise term, Eqtn. 6.40 can be simplified as shown in Eqtn. 6.41.

Eqtn. 6.41

$$J_{MSE} = \sigma_I^2 \sum_{n=-\infty}^{-1} r_n^2 + \sigma_I^2 \sum_{n=1}^{\infty} (r_n - b_n)^2 + \sigma_I^2 (r_0 - 1)^2 + \frac{N_o}{2} \int_{-\infty}^{\infty} f^2(t) dt$$

Note that all terms in Eqtn. 6.41 are positive and that only the second term involves coefficients from the feedback filter. You should easily see that $b_n = r_n$ for $n > 0$ are the optimal feedback filter coefficients. This result would force the second term of Eqtn. 6.41 to zero and minimize J_{MSE} with respect to the feedback filter. Logically, this value follows because r_n for $n > 0$ defines the interference from past symbols into the current slicer input. Mimicking this operation in the feedback filter and subtracting its output cancels this postcursor ISI.

You could continue to minimize the mean squared error, J_{MSE} , by standard optimization techniques and solve for the optimal feed-forward filter.^{10,11} In addition, optimization can be done for the transmitter pulse-shaping filter. The result is that the feed-forward filter is a matched filter followed by a tapped delay line.

Note

If the matched filter is implemented in front of the feed-forward filter, then it is simply just the tapped delay line.

Eqtn. 6.42 gives the MSE result after minimization.

Eqtn. 6.42

$$J_{MSE} = \exp \left(-2\tau \int_0^{\frac{1}{2\tau}} \ln(1 + SNR_f(f)df) \right)$$

SNR_f in this equation represents the folded SNR at the input to the receiver given by Eqtn. 6.43.

Eqtn. 6.43

$$SNR_f(f) = \sum_{n=-\infty}^{\infty} \frac{S\left(f + \frac{n}{\tau}\right) C\left(f + \frac{n}{\tau}\right)}{N\left(f + \frac{n}{\tau}\right)} \quad 0 < f < \frac{1}{2\tau}$$

In this equation, $C(f)$ is the combined response of the pulse shaping filter and the channel. The resulting SNR at the input to the slicer is then shown in Eqtn. 6.44.

Eqtn. 6.44

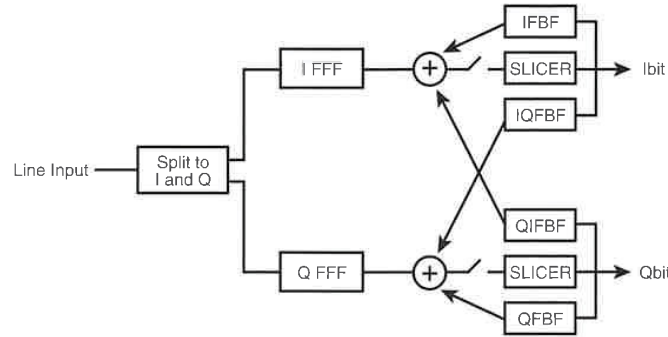
$$\begin{aligned} SNR_s &= \frac{1 - J_{MSE}}{J_{MSE}} \\ &\approx \frac{1}{J_{MSE}} \\ &= \frac{10}{\ln(10)} \left(2\tau \int_0^{\frac{1}{2\tau}} \ln(1 + SNR_f(f)df) \right) \end{aligned}$$

The approximation in the second line of Eqtn. 6.44 holds for all $J_{MSE} < 1$.

Note the difference between SNR_f and SNR_s . The first is a function of frequency and represents the SNR of signal power to noise power at the front end of the receiver at a given frequency. The signal power in this case may be made up of signal energy from more than one symbol due to ISI. The second term, SNR_s , is the SNR at the input to the decision device. The signal power in this case is only due to energy of the symbol entering the slicer. The noise power is due to the filtered background noise and any residual ISI.

The analysis of DFEs up to this point has assumed a PAM-based system. A DFE can be applied to a CAP/QAM-based system to decode received symbols as well. Figure 6.32 shows the basic structure of a DFE in this environment.

Figure 6.32 A decision feedback equalizer for a CAP/QAM system.



Note that both the I and Q paths have a feed-forward filter. In addition, four feedback filters exist, allowing interference to be eliminated on the two branches due to both previous I components and previous Q components. Eqtn. 6.45 gives the SNR at the input to either slicer for the CAP/QAM DFE.

Eqtn. 6.45

$$\begin{aligned}
 SNR_s &= \frac{1 - J_{MSE}}{J_{MSE}} \\
 &\approx \frac{1}{J_{MSE}} \\
 &= \frac{10}{\ln(10)} \left(\tau \int_{f_c - \frac{1}{2r}}^{f_c + \frac{1}{2r}} \ln(1 + SNR_f(f)) df \right)
 \end{aligned}$$

The folded SNR, SNR_p , is given by Eqtn. 6.46.

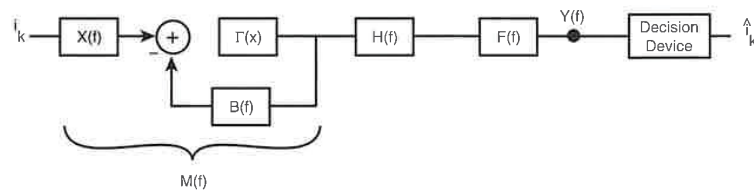
Eqtn. 6.46

$$SNR_f(f) = \sum_{n=0}^{\infty} \frac{S\left(f + \frac{n}{\tau}\right) C\left(f + \frac{n}{\tau}\right)}{N\left(f + \frac{n}{\tau}\right)} \text{ for } f_c - \frac{1}{2\tau} < f < f_c + \frac{1}{2\tau}$$

Tomlinson-Harashima Precoding

Although not obvious at first, Tomlinson precoding, sometimes called *Tomlinson-Harashima precoding* (THP) is related to decision feedback equalization. A main difference between the two is that for THP the feedback filter is not present in the receiver, but rather a precoding feedback filter is present in the transmitter. Figure 6.33 shows a simple THP system.

Figure 6.33 A Tomlinson-Harashima precoder in a transmitter along with a channel and a receiver.



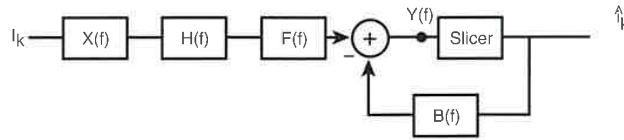
The operator $\Gamma(x)$ in Figure 6.33 is a modulo operator. Ignoring this operator, the output of the receiver (input to the decision device) is given by Eqtn. 6.47.

Eqtn. 6.47

$$\begin{aligned} Y(f) &= M(f)H(f)F(f) \\ &= \left(\frac{X(f)}{1-B(f)} \right) H(f)F(f) \\ &= \frac{X(f)H(f)F(f)}{1-B(f)} \end{aligned}$$

Figure 6.34 shows a DFE with similar notation.

Figure 6.34 Block diagram of a decision feedback equalizer.



Assuming ideal decisions, Eqtn. 6.48 gives the input of the slicer.

Eqtn. 6.48

$$\begin{aligned}
 Y(f) &= X(f)H(f)F(f) - \hat{X}(f)B(f) \\
 &= X(f)H(f)F(f) - X(f)B(f) \\
 &= X(f)(H(f)F(f) - B(f))
 \end{aligned}$$

In Eqtn. 6.48, the second line reflects the ideal decision assumption.

For the zero-forcing case, no ISI should exist at the input to the decision device. For the THP, Eqtn. 6.49 must hold.

Eqtn. 6.49

$$\frac{H(f)F(f)}{1 + B(f)} = 1$$

For the DFE, the zero-forcing condition requires that Eqtn. 6.50 be true.

Eqtn. 6.50

$$H(f)F(f) - B(f) = 1$$

Note that Eqtn. 6.50 can be rewritten as Eqtn. 6.51.

Eqtn. 6.51

$$H(f)F(f) = 1 + B(f)$$

or

$$\frac{H(f)F(f)}{1 + B(f)} = 1$$

Because this result matches Eqtn. 6.49, it is clear that the ZF constraint placed on the THP filters results in the same requirements being placed on the DFE filters. Thus the optimal $F(f)$ and $B(f)$ are the same for both systems.

The MSE condition applied to the THP does not result in a similar intuitively convenient comparison with the DFE. However, it can be shown that the optimum $F(f)$ and $B(f)$ for the MSE THP are identical to those for the MSE DFE.^{12,13}

Frequency-Domain Equalization

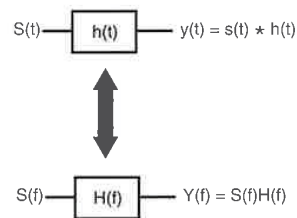
Equalization can be done in the frequency domain instead of in the time domain. Given a signal $s(t)$ and a linear, time-invariant filter with impulse response $h(t)$, the basic relationship between filtering (convolution) in the time domain and filtering in the frequency domain is given by Eqtn. 6.52.

Eqtn. 6.52

$$s(t) * h(t) \Leftrightarrow S(f)H(f)$$

The arrow in this equation represents the fourier transform. In other words, convolution in the time domain is equivalent to multiplication in the frequency domain, as shown in Figure 6.35.

Figure 6.35 The duality of time-domain and frequency-domain representations of filtering.



It may be apparent that a modulation technique such as DMT, which involves a transformation between the time domain and frequency domain at the receiver using a discrete fourier transform, may lend itself nicely to frequency-domain equalization. In this case, a single complex multiplication to the received value in each DMT bin would sufficiently equalize the channel. However, a small caveat must be considered before this technique will work.

Note

Such equalization is sometimes much more flexible than equalization in the time domain. Additionally, long filters can realize significant savings in the number of multiplications needed.

Assume that two signals $s(t)$ and $h(t)$ are sampled and an N -point discrete fourier transform taken. Multiplication of the resulting frequency-domain components is related to the sampled time-domain values as shown in Eqtn. 6.53.

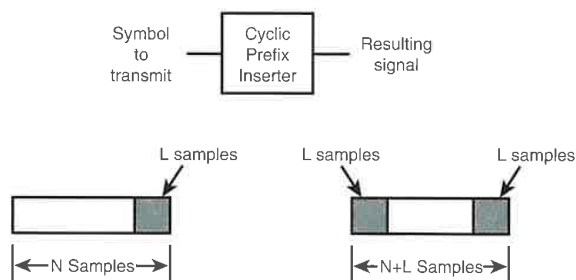
Eqtn. 6.53

$$S^{(N)}(kf_o)H^{(N)}(kf_o) = s^{(N)} \otimes h^{(N)}$$

The superscripts in Eqtn. 6.53 represent the block or window size of the samples. The fundamental difference between Eqtn. 6.52 and Eqtn. 6.53 is the circular convolution in the latter as opposed to a linear convolution in Eqtn. 6.52. With discrete domains, multiplication in the frequency domain is equivalent to circular convolution in the time domain. The obvious problem with directly implementing frequency-domain equalization is that the output of a twisted pair channel looks like a linear convolution of the input and the channel response, not a circular convolution.

To make the output look like a circular convolution, a *cyclic prefix* is added to each symbol to be transmitted over the channel. The process involves appending the last L samples in the symbol to the beginning of the symbol before sending it over the channel. If the original symbol is N samples long, the resulting symbol will be $N + L$ samples long, as illustrated in Figure 6.36.

Figure 6.36 An example of a cyclic prefix insertion block.



At the receiver, only N samples are processed, and the L samples added are dropped. However, the N -length block appears to have been circularly convolved with the channel. In other words, the channel has been made a *circular channel*.

Note that adding a cyclic prefix does add overhead and reduces the amount of information that can be sent over the channel. For long blocks of data (large N), this overhead is usually a very small fraction of the payload and can be ignored. Also, if the impulse response of the channel is larger than the length of added cyclic prefix, energy from the previous symbol will leak into the N saved samples and the channel will no longer look circular. This

occurrence is common. The solution is to combine some front-end time-domain equalization before removing the N samples at the receiver. The front-end equalization can effectively shorten the channel response to less than L samples, thus restoring the circular property desired.

Shaping

Shaping filters are often used at the output of a modulator. Analog or digital filters or both are used. Most often, shaping is used to ensure that the output waveforms of the transmitter adhere to pertinent spectral requirement. For DSLs, some of these requirements are discussed in Chapter 4. Shaping is often challenging, as the out-of-band energy must be properly attenuated while the in-band energy should experience minimal distortion.

Summary

This chapter presented an overview of DSL modulation basics and discussed many of the transmitter and receiver blocks used in a DSL system. CAP and QAM modulation and demodulation were discussed and the similarities and differences between them explained. DMT modulation and demodulation was also discussed, along with the method of coupling a frequency-domain equalizer into the system.

The benefits and methods of forward error correction were discussed. The differences between block and convolutional coding and decoding were explained. Block and convolutional interleaving were also discussed. The method of each was explained, as well as the delay and memory requirements of the two schemes.

Equalization in general and various forms of equalization, including a linear equalizer, a decision feedback equalizer, and a Tomlinson-Harashima precoder, were discussed. Performance equations for the equalizers were given, and the pros and cons of each discussed.

This chapter introduced the various modulation and demodulation blocks used in DSL systems. The next two chapters discuss how DSL systems actually implement some of these blocks to produce a realizable, high-performance transmitter and receiver.

Exercises

1. For a given background noise power, derive and plot the probability of error as a function of average signal power for a 4-QAM, 16-QAM, and 64-QAM system.
2. For a 4-QAM system, assume that the background noise power is $1/10$ of the average signal power. Through simulation, plot the location of a point projected at the receiver onto the constellation decoder if the constellation point from the first

quadrant is sent. Repeat this procedure on the same constellation plot 1,000 times. Do the same assuming the other three constellation points were sent.

3. Repeat exercise 2 assuming that the background noise power is $1/5$ of the average signal power. Repeat again assuming $1/3$. Discuss the locus of received points and how they relate to error probability.
4. Simulate a DMT transmitter with eight tones by using uniform iid random variables on $[-A, A]$ as real and imaginary inputs for the IFFT. Over many symbols, record the peak time-domain power value (output sample squared) and keep a running average of the square of the samples. Repeat and compare for 32 and 64 tone DMT symbols and for a QAM symbol with the same uniform input. Comment on the peak-to-average ratio (PAR) of the squared time-domain samples.
5. Find the output sequence for a rate $1/2$ convolutional encoder with generator polynomials $G1(x)=1+x+x^2$ and $G2(x)=x$. Assume the same input sequence given in the text.
6. Using the convolutional decoder example given in the text, decode the received signal assuming that the seventh bit was received in error.
7. Write the input and output sequence of a block interleaver with a depth of 5 and a span of 9. Repeat for a convolutional interleaver with the same properties. Calculate the time that it would take for a single codeword to reach the receiver with both schemes as well as if no interleaver were used.
8. Find the folded spectrum of an ISDN signal sent over the mid-CSA Loop and sampled at the receiver at 80 kHz. Compare this spectrum to the spectrum received assuming no folding.
9. Find the SNR of an HDSL signal at the input to the decision device using CSA Loop #4 if a ZF linear equalizer is used. Repeat assuming that a MSE linear equalizer is used. Assume that additive white Gaussian noise (AWGN) is present with a PSD of -140 dBm/Hz along with 24 self-NEXT disturbers.
10. Repeat exercise 9 assuming that a DFE is used.
11. Create two arbitrary vectors of length 8. Verify that the FFT of the circular convolution of the vectors is equal to the frequency-domain multiplication of the FFT of each vector.

Endnotes

1. Ken Kerpez, CAP and QAM System Interoperability, T1E1.4 Contribution 96-242, Nashua, NH, September 5, 1996.

2. Stephen G. Wilson, *Digital Modulation and Coding*. NJ: Prentice Hall, 1996.
3. Shu Lin and Daniel J. Costello Jr., *Error Control Coding: Fundamentals and Applications*. Englewood Cliffs, NJ: Prentice Hall, 1983.
4. I. S. Reed and G. Solomon, "Polynomial Codes over Certain Finite Fields," *J. Soc. Ind. Appl. Math*, 9, June 1960, pp. 300–304.
5. R. C. Bose and D. K. Ray-Cyaudhuri, "On a Class of Error Correcting Binary Group Codes," *Inf. Control*, 3, March 1960, pp. 68–79.
6. A. Hocquenghem, "Codes Corecteurs d'Erreurs," *Chiffres*, 2, 1959, pp. 147–156.
7. G. Ungerboeck, "Trellis Coded Modulation with Redundant Signal Sets—Part I: Introduction," *IEEE Communications Magazine*, vol. 25, no. 2, February 1987.
8. S. Benedetto, M. Mondin, and G. Montorsi, "Performance Evaluation of Trellis Codes," *IEEE Transactions on Information Theory*, vol. IT-32, March 1987.
9. J.J. Werner contribution on blind equalization.
10. J. Salz, "Optimum Mean Square Decision Feedback Equalization," *Bell System Technical Journal*, vol. 52, no. 8, October 1973.
11. Carlos A. Belfiore and John H. Park, "Decision Feedback Equalization," *Proceedings of the IEEE*, vol. 67, no. 8, August 1979.
12. Richard D. Wesel and John Cioffi, "Achievable Rates for Tomlinson-Harashima Precoding," submitted to *IEEE Transactions on Information Theory*, 1996.
13. Richard D. Wesel and John Cioffi, "Precoding and the MMSE-DFE," *Proceedings of the 28th Asilomar Conference on Signals, Systems, & Computers*, 1995, pp. 1144–1149.

ADSL Modulation Specifics

In this chapter:

- Details of a DMT ADSL transmitter and receiver implementation
- Discussion of data and overhead channels for DMT ADSL
- Explanation of training and initialization procedures for DMT ADSL
- Discussion of a CAP/QAM ADSL transmitter and receiver
- Discussion of special ADSL functions to accommodate ATM

The preceding chapter discussed many of the common blocks used in DSL transmitters and receivers. This chapter describes how these blocks are implemented to create an ADSL transceiver. Specifically, blocks will be discussed for ADSL end modems: the ADSL transceiver unit-remote (ATU-R) normally located at a customer premise and the ADSL transceiver unit-central office (ATU-C) normally located at the central office aggregation point. Many of the ADSL options are described along with the more common implementations of them. The first part of the chapter deals with a discrete multitone (DMT) ADSL implementation, and the second part deals with a Carrier Amplitude and Phase/Quadrature Amplitude Modulation (CAP/QAM) implementation. The DMT implementation is according to the specification in the ANSI document T1E1.4/98-007R4,¹ and the CAP/QAM implementation is according to the specification in ANSI document T1E1.4/97-104R2a.²

DMT ADSL Systems

Figures 7.1a and 7.1b show a block diagram of a DMT ADSL transmitter. Figure 7.2 shows a block diagram of an upstream and downstream DMT ADSL receiver.

Figure 7.1a Block diagram for an upstream ADSL transmitter.

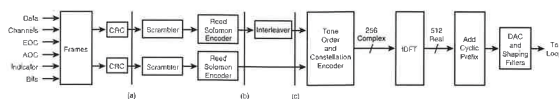


Figure 7.1b Block diagram for a downstream ADSL transmitter.

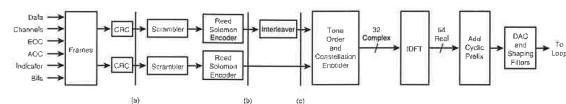
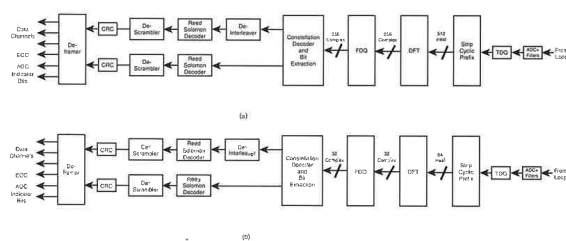


Figure 7.2 shows a DMT ADSL receiver.

Figure 7.2 Block diagram for an upstream (a) and downstream (b) ADSL receiver.



Both upstream and downstream transmitters and receivers are shown in these figures. As discussed previously, DMT ADSL is the standard for ADSL set forth by ANSI and ETSI^{1,3}. The specifications allow for high-speed transmission greater than 6 Mbps in the downstream direction and lower-speed transmission up to about 640 kbps in the upstream direction. The actual rates depend on line and noise conditions as discussed in Chapter 5, "DSL Theoretical Capacity in Crosstalk Environments." Also, the company providing the ADSL service (often called the access provider) may limit the actual rate. In such cases, a line that could potentially support higher rates is configured to support a lower rate.

Note

This situation is typical in many service offerings. An access provider may offer customers different service tier options with some tiers providing higher rates (and presumably costing more) than others.

Transmitter and Receiver Blocks

The following sections discuss the various blocks shown in Figure 7.1a, 7.1b, and 7.2 for DMT ADSL. Note that many of the blocks were discussed in general terms in Chapter 6, "DSL Modulation Basics." Because of the asymmetric nature of ADSL, some of the blocks may have a different implementation at the ATU-C and the ATU-R.

It is worth mentioning up front that the transceivers may carry more than one data channel and also carry an embedded operations channel (EOC), an ADSL overhead channel (AOC), and also various other bits to be described later in this chapter. All but the data is often simply termed *overhead*. It is said that a single *physical channel*, the physical ADSL link or ADSL channel, carries more than one *logical channel*, the logical channels being all data and overhead channels. Combining and separating the logical channels at the transmitter and receiver is done using a frame structure that is described later in this chapter.

The logical data channels on an ADSL link consist of four possible simplex downstream channels and three possible full-duplex channels. The logical channels are sometimes called *bearer channels*. The simplex channels are named *AS0*, *AS1*, *AS2*, and *AS3*. The duplex channels are named *LS0*, *LS1*, and *LS2*. The duplex channels may have different rates upstream and downstream including a rate of zero in one or both directions. When a duplex channel rate is zero in one direction and nonzero in the other direction, it is analogous to a simplex channel in the nonzero direction. The allowable rates of the different bearer channels are listed in Table 7.1. For all logical data channels, the actual rates must be multiples of 32 kbps (the reason is explained later). In many implementations, only one of the simplex data channels is used in the downstream direction, and a duplex channel is used in simplex mode for the upstream direction. Typically, the channels used would be *AS0* and *LS0*.

Table 7.1 ADSL Logical Data Channels and Associated Rates

Channel	Type	Allowable Rates	Comments
AS0	Downstream Simplex	0-8.192 Mbps	This channel is most commonly used as the lone downstream channel.
AS1	Downstream Simplex	0-4.608 Mbps	
AS2	Downstream Simplex	0-3.072 Mbps	
AS3	Downstream Simplex	0-1.536 Mbps	
LS0	Duplex	0-640 kbps	Can have different rates in each direction. This channel is most commonly configured as an upstream simplex channel.
LS1	Duplex	0-640 kbps	Can have different rates in each direction.
LS2	Duplex	0-640 kbps	Can have different rates in each direction.

The different logical channels allow ADSL to be flexible enough to support many different applications. For example, one of the first drivers of ADSL was video on demand over a telephone line. In this case, a downstream simplex channel, AS0, for example, can be used to send streaming digital video and audio while a duplex channel, LS0, for example, can be used to send and receive control information, such as pause or rewind commands. In any case, the total bandwidth available on the physical channel must be greater than the sum of the bandwidths used by each logical channel. (One may argue that these should be equal; however, some of the physical channel bandwidth is used for other purposes such as synchronization. This topic is discussed later.)

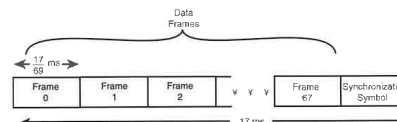
Note also that the ADSL transmitter and receiver each have two associated paths. One path is called the *fast path*, and one path is called the *interleaved path*. Cyclic redundancy check (CRC) generation, scrambling, and forward error correction are independent on each path. The main difference between the two is that the interleaved path contains an interleaving function in the transmitter (and de-interleaving function in the receiver), whereas the fast path does not. The name fast path comes from the fact that interleaving, and thus the delay caused by interleaving, is not present so that data gets through the transmitter and receiver faster than in the other path. Simply put, bits transmitted on the fast path get through the transmitter and receiver with less delay than bits transmitted

through the interleaved path. You will see later that a logical data channel is assigned to either the fast or interleaved channel, but never to both. However, both paths can be active simultaneously (each with one or more logical data channels assigned to it). Many implementations, however, use only one of the paths.

Framing

The framing structure of ADSL is key to understanding how the different logical channels are combined or separated when transmitted on the physical ADSL channel. ADSL has a framing structure as well as a *superframe structure*. The superframe structure is 17 ms long and is made up of 68 frames, which are numbered 0 through 67, and a synchronization symbol. Figure 7.3 shows this structure.

Figure 7.3 The structure of an ADSL superframe.



The basic superframe structure is the same for both the upstream and downstream directions. The synchronization symbol has the same duration as a data frame, and it can be thought of as a frame that carries no user data. For short, I refer to the *synchronization symbol* as simply the *sync symbol* or *sync frame*.

Each ADSL frame, including the sync frame, corresponds to a single ADSL symbol. Each symbol has a duration of just over 246 μ s (specifically $250 \times 68 / 69$ μ s). Adjusting for the time taken by the sync frame, the data symbols have a rate of 4 kHz. This relationship can be seen in Eqtn. 7.1.

Eqtn. 7.1

$$f_{\text{data symbol}} = \frac{68 \text{ data frames}}{17 \text{ ms}} = 4000 \frac{\text{data frames}}{\text{s}} = 4 \text{ kHz}$$

Each data frame within a superframe has a similar structure. Each data frame contains a specified number of data bytes for each of the active logical data channels in both the fast and interleaved paths. Overhead bytes are also contained in each data frame; however, the

overhead bytes contained are different for different frame numbers. Figure 7.4 shows a general data frame for an ADSL link.

Figure 7.4 The structure of a data frame including both the fast path and interleaved path components.

Fast Overhead	Fast Data	Fast Overhead	Interleaved Overhead	Interleaved Data	Interleaved Overhead
---------------	-----------	---------------	----------------------	------------------	----------------------

Each data frame is divided into a fast buffer part and an interleaved buffer part. As one might expect, the fast part contains the data to be sent through the fast path while the interleaved part contains data to be sent through the interleaved path. Figure 7.5 shows a more detailed description for both the upstream and downstream directions of the fast part of point A in Figure 7.1.

Figure 7.5 Detailed structure of the fast path portion of an ADSL data frame for (top) the downstream direction and (bottom) the upstream direction.

Fast Byte	AS0 Bytes	AS1 Bytes	AS2 Bytes	AS3 Bytes	LS0 Bytes	LS1 Bytes	LS2 Bytes	AEX Byte	LEX Byte
-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	----------	----------

Fast Byte	LS0 Bytes	LS1 Bytes	LS2 Bytes	LEX Byte
-----------	-----------	-----------	-----------	----------

In Figure 7.5, you can see that each logical channel is allotted a certain number of bytes. Note that the downstream data frame has room for bytes from all seven bearer channels, whereas the upstream frame has space only for the LS channels because the AS channels are not transmitted in the upstream direction. If a logical channel is not using the fast path, the number of bytes allotted to that channel is zero (thus it would take up no space in the frame). Note that the allocation of bytes (eight bits), rather than fractions of bytes, to a data channel in a data frame combined with the 4 kHz data frame rate gives rise to the 32 kbps granularity of a channel's data rate. In fact, the data rate of a data channel can be found by Eqtn. 7.2:

Eqtn. 7.2

$$R_{\text{channel}} = \# \text{ data bytes per frame} \times 8 \frac{\text{bits}}{\text{byte}} \times 4000 \frac{\text{data frames}}{\text{s}}$$

Present in the data frame for the fast buffer are several overhead bytes, the *fast byte*, the *AEX byte*, and the *LEX byte* in the downstream direction and the fast byte and LEX byte

in the upstream direction. The description of these bytes is given in the following paragraphs.

The fast byte is short for the *fast path synchronization byte*. The fast byte is always present even if no bearer channels are using the fast path. It has four different uses depending on the frame number (0 through 67). Its four uses are

- Carrying the CRC check for a superframe
- Carrying the indicator bits
- Carrying the embedded operations channel (EOC)
- Carrying synchronization control information to allow stuffing and robbing of bytes for synchronization

In frame 0, the fast byte carries superframe CRC information. This information is the CRC computed on the entire preceding superframe. In frames 1, 34, and 35, the fast byte carries *indicator bits (ibs)*. An indicator bit communicates some type of condition or primitive indication to the peer transceiver. There are 24 ibs in all; each is explained later in this chapter. In data frames 2 through 33 and 36 through 67, the fast byte carries either EOC data or synchronization control information. Either type of payload is two bytes long and must occupy an even-numbered-frame fast byte followed by an odd-numbered-frame fast byte.

An EOC message is distinguished from a synchronization control message because all bytes in an EOC message have a Least Significant Bit (LSB) of one, whereas all bytes in a synch control message have an LSB of zero. EOC messages and synch control messages are discussed later in this chapter. In summary, Figure 7.6 gives the makeup of the fast byte for both the upstream and downstream directions.

Figure 7.6 Makeup of the fast byte for all frames within the ADSL superframe.

MSB								LSB							
Frame 0								Frame 1							
CRC7	CRC8	DFC7	DFC8	CRC7	CRC8	DFC7	DFC8	IB1	IB2	IB3	IB4	IB5	IB6	IB7	IB8
Frame 34								Frame 35							
IB1	IB4	IB3	IB2	IB5	IB6	IB7	IB8	IB9	IB10	IB11	IB12	IB13	IB14	IB15	IB16
Even frame								Odd frame							
IB1	IB4	IB3	IB2	IB5	IB6	IB7	IB8	IB9	IB10	IB11	IB12	IB13	IB14	IB15	IB16
Even frame								Odd frame							
IB1	IB4	IB3	IB2	IB5	IB6	IB7	IB8	IB9	IB10	IB11	IB12	IB13	IB14	IB15	IB16

The AEX and LEX bytes are used to stuff an extra byte into an AS and LS channel. The AEX byte is present in the fast path only if at least one of the AS bearer channels uses the fast path. Likewise, LEX is present on the path only if an LS channel is present. Whether or not a byte is added to a bearer channel during a frame is controlled by synchronization control information in the fast byte. As stated previously, the synchronization control information can be present in a fast byte during frames 2 through 33 and 36 through 67. Table 7.2 gives the format of an eight-bit synchronization message.

Table 7.2 Format of a Fast Channel Synchronization Message (Fast Byte)

Bits	Function	Code
7,6	Designates the AS channel to stuff or rob	00—channel AS0. 01—channel AS1. 10—channel AS2. 11—channel AS3.
5,4	Controls the action to be taken on the designated AS channel	00—do nothing. 01—add AEX byte. 10—add AEX and LEX bytes. 11—delete last byte.
3,2	Designates the LS channel to stuff or rob	00—channel LS0. 01—channel LS1. 10—channel LS2. 11—no synchronization action.
1	Controls the action to be taken on the designated LS channel	0—add LEX byte. 1—delete last byte.
0	Sync/EOC indicator	0—indicates byte is a fast control byte. 1—In downstream direction, indicates that LEX byte is an EOC byte, but synchronization on AS channels may be performed as indicated. In upstream direction, indicates that the Fast byte is an EOC byte.

Quite simply, the synchronization messages define what action is to be taken on a specified AS channel and a specified LS channel for that frame. Note that for an AS channel, up to two bytes can be stuffed (AEX and LEX) or one byte deleted. A deleted byte is sometimes called a *robbed byte*. For the LS channels, only one byte can be stuffed or robbed. Note

that for the upstream, the upper four bits of the fast byte are unused because the upstream channel does not use AS bearers.

Byte stuffing and robbing is used to equalize the rate of a logical channel over the ADSL link with the rate needed to transport the channel. A good example of this is if a logical channel is transparently transporting a T1 line (1.544 Mbps). If the logical channel is running at a rate slightly lower than 1.544 Mbps, every so often the channel must stuff a byte or its input buffers will overflow. Likewise, if the logical channel were running slightly faster than 1.544 Mbps, every so often it must rob a byte so that its input buffers do not underflow. In each case, stuffing and robbing allow the logical channel, on average, to run at 1.544 Mbps.

Figures 7.7a and 7.7b show the fast path data buffer at points B and C in Figure 7.1a and Figure 7.1b for both the upstream and downstream directions, respectively. The only difference at this point is the addition of the Forward Error Correction (FEC) bytes from the encoder. Note that at these points, the data payload is scrambled.

Figure 7.7a The structure of the fast path portion of an ADSL data frame after the FEC block for the upstream direction.

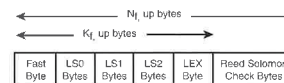


Figure 7.7b The structure of the fast path portion of an ADSL data frame after the FEC block for the downstream direction.

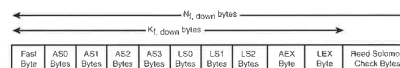


Figure 7.8a and Figure 7.8b show the data buffer at point A in Figure 7.1a and Figure 7.1b for the interleaved path.

Figure 7.8a The structure of an interleaved mux data frame before the FEC block for the upstream direction.

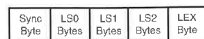


Figure 7.8b The structure of an interleaved mux data frame before the FEC block for the downstream direction.



At this point, the data frame on the interleaved path is called a *mux data frame*. Note that at point A, interleaving has not yet been done. Similar to the fast path, the interleaved buffer also contains an AEX and a LEX byte (only LEX in the upstream direction). Again, the AEX byte is present only if an AS bearer channel uses the interleaved path. The LEX byte is present unless no channels are using the interleaved path. Analogous to and in lieu of the fast byte, however, the interleaved path contains a *sync byte*.

The sync byte has several functions. Some of these functions are analogous to fast bytes' functions, but at least one is different. In general, the sync byte performs the following actions:

- Carries CRC information for the interleaved channel in the preceding superframe
- Carries synchronization control information for stuffing and robbing bytes from bearer channels using the interleaved path
- Signals when the LEX byte of the interleaved path is used to carry a byte for the ADSL overhead channel (AOC) channel
- Carries the AOC channel when no bearer channels are using the interleaved path

Similar to the fast byte, in frame zero, the sync byte carries eight bits of CRC information. These cover the bearer channels assigned to the interleaved path over the preceding superframe. In all frames 1 through 67, when at least one bearer channel is using the interleaved path, the sync byte carries eight bits decoded as shown in Table 7.3.

Table 7.3 Format of an Interleaved Channel Synchronization Message (Sync Byte)

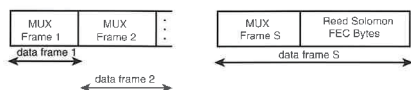
Bits	Function	Code
7,6	Designates the AS channel to stuff or rob	00—channel AS0. 01—channel AS1. 10—channel AS2. 11—channel AS3.
5,4	Controls the action to be taken on the designated AS channel	00—do nothing. 01—add AEX byte. 11—add AEX and LEX bytes. 10—delete last byte.
3,2	Designates the LS channel to stuff or rob	00—channel LS0. 01—channel LS1. 10—channel LS2. 11—no synchronization action.
1	Controls the action to be taken on the designated LS channel	1—add LEX byte. 0—delete last byte.
0	LEX Byte Sync/AOC indicator	0—indicates the LEX byte is to be used for synchronization as specified in bits 1–7. 1—In downstream direction, indicates that the sync byte contains AOC data and limited synchronization can be done. In upstream direction, indicates that the LEX contains AOC data.

As shown in Table 7.3, a mechanism to add or delete a byte from any channel, almost identical to that for the fast byte, is provided by the sync byte. Note that for the upstream, no mechanism is present to add or stuff bytes in the AS channels because no AS channels run in the upstream direction. In the downstream direction, either one or two bytes can be appended to an AS channel, whereas only one byte can be appended to an LS channel. Note that the LSB of the sync byte indicates the contents of the LEX byte. If the LSB is one, the LEX byte contains AOC information. In this case, bits 4 through 7 of the sync byte still define synchronization action to perform to the AS channel (although only a single byte may be added). Additionally, a delete may still be performed to the LS channels. However, because LEX is already carrying AOC information, neither a one-byte stuff can be performed on the LS channels nor can a two-byte stuff be performed on the AS channels.

When no channels are using the interleaved path, no interleaved path bearer channel synchronization is necessary. In this case, both the AEX and LEX bytes are absent, and the sync byte is used to carry the AOC channel in frames 1–67.

On the interleaved path, multiple mux data frames can make up an FEC code word. This situation contrasts with the fast path, where a one-to-one correspondence between symbols and FEC codewords occurs. The actual number of mux data frames in an FEC codeword is decided during initialization. If this number is labeled as S , then at point B in Figure 7.1a and Figure 7.1b, the interleaved path buffer is shown in Figure 7.9. In this case, note that the FEC codeword spans multiple frames but is always an integer number of frames.

Figure 7.9 An FEC codeword on the interleaved path spanning S data frames.



At point C in Figure 7.1a and Figure 7.1b, the FEC codewords are interleaved. Thus, the data to be modulated is made up of bytes from several FEC codewords. If a codeword is made up of S mux data frames for a total of N bytes and has K check bytes, then each interleaved path frame will consist of $(N+K)$ bytes.

CRC Generator/Detector Blocks

For both the fast and interleaved paths, an eight-bit CRC is generated for each superframe and sent during the first frame of the next superframe. On the fast path, the CRC is sent via the fast byte. On the interleaved path, the CRC is set via the sync byte. The CRC for a path covers all bits sent through that path during a superframe except for the FEC bytes, the previous CRC, and the synchronization symbol (frame 68). The CRC is meant to help keep track of how many superframes were received that contained an error uncorrectable by the FEC blocks. Both CRC generators and receivers are based on the generator polynomial shown in Eqtn. 7.3.

Eqtn. 7.3

$$G(x) = 1 + x^2 + x^3 + x^4 + x^8$$

Scrambler/Descrambler

For both the ATU-R and the ATU-C, the generator polynomial for the scramblers and descramblers is given in Eqtn. 7.4.

Eqtn. 7.4

$$g(x) = 1 + x^{18} + x^{23}$$

As mentioned previously, the fast and interleaved paths are scrambled and descrambled independently. All bits from all logical channels are scrambled and descrambled. FEC bytes are not scrambled because they are added after the scrambling block. At the receiver, they are removed before the descrambling block. The only frame that does not pass through the scrambler is the sync frame.

Forward Error Correction Block

After being scrambled, the data in both the fast and interleaved paths enter independent FEC blocks. Both paths utilize the Reed-Solomon technique in Galois field (256). The number of check bytes K and the number of payload bytes N that make up a codeword are decided during initialization of the ADSL link. For the fast path, a single codeword is sent for each data frame. For the interleaved path, a codeword can consist of an integer number of frames S , as long as S is a power of two less than or equal to 16 ($S=1, 2, 4, 8, \text{ or } 16$). For both paths, the number of check bytes may be any even number between 0 and 16, inclusive. As a rule of thumb, the ratio of the number of check bytes to the number of data payload bytes should be around 0:1. This ratio gives a reasonable coding gain without a significant amount of overhead.

Interleaver/De-interleaver

After passing through the forward error correction block, data in the interleaved path enters the interleaver in the transmitter, in both the ATU-R and ATU-C. DMT ADSL uses convolutional interleaving. The interleaver depth is expressed in Reed-Solomon codewords and may be any power of two up to and including 64. The interleaver works at the byte level because the Reed-Solomon code works in GF(256).

At the receiver, data enters the de-interleaver after constellation decoding and bit extraction (explained later in the chapter) are performed. The de-interleaver essentially reorders the bytes back into Reed-Solomon codewords for processing by the FEC decoder. The fast and interleaved paths have independent de-interleavers. Each de-interleaver uses the same parameters as its respective interleaver in the transmitter agreed upon during initialization.

Tone Ordering and Constellation Encoder

Before entering the DMT modulation block, the payload to be transmitted consists of a buffer of data from the fast channel and a buffer of data from the interleaved channel. To modulate this data, you must first extract the proper number of bits for each DMT bin and code these bits into a complex Z_i value (one complex value per bin). The process of extracting the bits and assigning them to tones is called *tone ordering*. The process of encoding the extracted bits into complex values for each bin is called *constellation encoding*. Constellation encoding for a DMT modulator is discussed in Chapter 6.

For tone ordering, the number of bits assigned to each bin must be known. This information is decided during startup training when the bins are analyzed. After analysis, the ATU-C and ATU-R calculate and agree to the number of bits to be handled by each bin.

Commonly, the value that represents calculated bits per bin is called the bin's *bi* value. The number of bits per bin for ADSL must be between 2 and 15, inclusive, or be 0. The sum of the number bits carried by all subcarriers directly correlates to the number of bits in each fast and interleaved path data frame. If the number of bits per bin is known, tone ordering follows this process:

1. Extract the N bits to be sent from the fast and interleaved paths.
2. Order these bits from 1 to N , putting the fast buffer bits first and the interleaved buffer bits second.
3. Execute the following algorithm on the ordered bits:


```

      For k=2 to 15
      {
      While an unused tone remains that has  $b_i=k$  bits
      {
      Find the lowest numbered tone that has  $b_i=k$  bits
      Assign the next  $k$  bits from the data buffer to that tone
      }
      }
      
```

Essentially, the preceding algorithm assigns bits from the fast path to the tones carrying fewer bits. Bits from the interleaved buffer are, in general, assigned to tones carrying more bits. The reason is that the algorithm begins filling lightly loaded bins first and the fast bits are ordered before the interleaved bits (step 2). Note that one of the tones may carry bits from both the fast and interleaved paths.

Note

Another option defined for DMT ADSL uses trellis-coded modulation on each bin. In this case, bit ordering is done slightly differently.¹

The tone-ordering algorithm was designed to minimize error occurrences. Impulses during a symbol (whether from an outside influence or from a clip) will cause an almost even amount of noise in each DMT bin. In this case, tones carrying a larger number of bits are more likely to make a decoding error than are tones carrying a smaller number of bits. Thus, in general, an impulse will cause more bit errors on the interleaved path and less on the fast path. In the receiver, the bit errors on the fast path will likely be corrected by the forward error correction. The bit errors from the interleaved path, although more numerous than the fast path, will be split up and, because they will most probably belong to different FEC codewords (due to interleaving), will also be corrected. Note that the main assumption here is that impulses do not occur regularly.

After bits are properly assigned to each subchannel, constellation encoding takes place. The encoding technique is independent for each subchannel. The number of points in each bin's constellations depends on the number bits assigned to the bin. In the range of 2 to 15 bits per bin, the constellation size range is 4 points (2^2) to 32,768 points (2^{15}).

The output of each constellation encoder is a complex value. As discussed in the preceding chapter, this complex value represents one of the input points to the DMT modulator's Inverse Fast Fourier Transform (IFFT). For the downstream direction, a total of 256 bins exists. Thus the output of the tone order and constellation encoder at the ATU-C is 256 complex values. In the upstream direction, a total of 32 bins exists, and so the output of the block at the ATU-R is 32 complex values.

The constellation encoder block also includes a fine-gain-adjustment function. Apart from the b_i value, each tone has assigned to it a fine-gain-adjustment value called g_i . The constellation output values are scaled by this g_i value. Fine-gain adjustment equalizes the probability of error on subchannels by adjusting the output signal power (and thus the received signal power) by a factor in the range between -1.25 and +2 (approximately 0.75 to 1.33). For subchannel i , the constellation encoder's output complex value is referred to as Z_i .

At first the reason for reducing the output signal power and increasing the error probability on a subchannel may not seem obvious. The reason lies in the *peak-to-average ratio* (PAR) of a DMT symbol. PAR is simply the ratio of the peak possible power of a DMT sample (a time domain's value after the IFFT) to the average power of a DMT sample. A large PAR is undesirable because the analog front end of a modem with a large PAR must have a very large dynamic range. This condition tends to make the front-end designs more complex and also have higher power consumption. It turns out that DMT has a very high PAR (between 10 dB and 20 dB). This high PAR means that the dynamic range of the receiver must

all the individual bins add constructively to a high value. In many cases, this value actually goes outside the linear region of the analog front end and clipping results. Reducing power on tones that do not need the extra power to maintain a reasonable probability of error reduces not only the overall output power of the modem but also the probability that a clip will occur.

Note

Several algorithms are under development to reduce the PAR value for DMT.^{4,5,6}

In both the upstream and downstream direction, it is common for some of the bins to carry zero bits of information. The constellation encoder will output a value of zero for these bins. The g_i values for these bins are normally set to zero.

Modulation Block

The heart of the transmitter is the DMT modulation block. The ATU-C and ATU-R, although both employing DMT, differ in the number of points being used in the IFFT. In the ATU-C, the modulator has 256 bins and uses 255 complex frequency points (and their complex conjugates) along with two special points to generate 512 real time-domain points. Thus the downstream signal is made up of 256 bins or tones. The spacing between DMT bin centers is 4.3125 kHz with the first center at 4.3125 kHz and the highest center at 1.104 MHz. One of the special points described previously is the tone at 1.104 MHz. This tone is called the *Nyquist tone*, is not used to carry data, and normally is set to zero. The other special point is a DC term that is also set to zero. Eqtn. 7.5 gives the basic equation governing the IFFT.

Eqtn. 7.5

$$x_k = \sum_{i=0}^{511} Z_i e^{j \frac{2\pi k i}{512}} \text{ for } k = 0..511$$

The Z_i value $i=0$ (in Eqtn. 7.4) is the DC term set to zero. The next 256 Z_i values are the complex values coded from the QAM constellations for each subcarrier (including the Nyquist value of zero). The final 255 Z_i values are the complex conjugates of the values from $i=1$ through $i=255$ with symmetry about the Nyquist tone. An equation for these values is given in Eqtn. 7.6.

Eqtn. 7.6

$$Z_m = \text{conj}(Z_{512-m}) \text{ for } m = 257..511$$

For the special case of the synchronization symbol, all the Z_i are set to real values with nominal amplitude.

The DMT modulator in the ATU-R uses 32 bins. The 32 complex values (including a zero for tone 32) representing the constellation encoding outputs for each subchannel, a DC term, and the complex conjugates of tones 1 through 31 produce 64 real time-domain values after the IFFT operation. Again, the tone spacing is 4.3125 kHz, which gives a tone range from 4.3125 kHz to 138 kHz. The basic equation for the ATU-R IFFT is shown in Eqtn. 7.7.

Eqtn. 7.7

$$x_k = \sum_{i=0}^{63} Z_i e^{j \frac{2\pi k i}{64}} \text{ for } k = 0..63$$

Here the first value of Z_i is the DC term, and the next 31 are the complex QAM encoder outputs for each subchannel (including the value of zero at the Nyquist tone). The values of Z_i for $i=33$ to $i=63$ are complex conjugates of tones 1 through 31 with complex conjugate symmetry about the Nyquist tone. An equation for this is shown in Eqtn. 7.8.

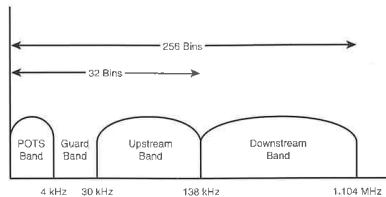
Eqtn. 7.8

$$Z_m = \text{conj}(Z_{64-m}) \text{ for } m = 33..63$$

For the case of the synchronization symbol, all Z_i are set to real values with nominal amplitude.

The flexibility of DMT modulation can be seen by considering the three areas in the frequency spectrum shown in Figure 7.10.

Figure 7.10 Spectral placement of the downstream, upstream, and POTS regions for frequency division multiplex (FDM) mode ADSL.



The lower-frequency area is the POTS band used for traditional voice transmission. This area is left open by the ADSL upstream and downstream signals by simply not using the first seven bins in both directions. Reserving the first seven bins in both directions is accomplished by assigning the bins in this region a $Z_k=0$ as well as using analog filtering on the front end of the transmitter to filter these frequencies. Setting the complex constellation encoder output to zero "turns off" the bin or tone. The middle frequency range in Figure 7.10 is used by the upstream signal. The downstream signal can leave this area free if it turns off tones 8 through 32. The upper-frequency region in Figure 7.10 is used exclusively by the downstream channel. The upstream channel does not need to do anything special to leave this region open because that channel transmits only on bins up to 138 kHz.

This type of DMT operation is called *frequency division multiplex (FDM) DMT* or *frequency division duplex (FDD) DMT*. The names stem from the fact that the duplexing function of the line is realized by using separate upstream and downstream frequencies. Note in this case, for multiple ADSL lines in a binder group, self-FEXT exists. However, the more damaging self-NEXT does not exist.

Another type of operating mode exists and is called *echo-canceled (EC) DMT*. For EC mode, the downstream also transmits on all tones above 30 kHz, causing it to overlap the ATU-R output in the 30 kHz to 138 kHz region. EC operation allows the downstream channel to achieve higher rates than does FDM mode.

EC operation has a couple of downsides. First, self-NEXT exists at the ATU-C and the ATU-R because local transmitters are using the 30 kHz to 138 kHz region. Note that at

the ATU-R, NEXT exists only in the 30 kHz to 138 kHz; the upper part of the downstream band is self-NEXT free. Second, both the analog and digital portions of the modem design become more complex. The analog portion must handle a more dynamic range, and the digital portion must implement a very precise echo canceler. Though some ADSL implementations do employ an EC mode, many network implementers prefer FDM mode because of the absence of self-NEXT.

Pilot Tones

In both the upstream and downstream directions, one of the DMT tones is reserved for a pilot signal. In the downstream direction, the pilot tone is at 276 kHz, or equivalently tone 64. In the upstream direction, the pilot tone is at 69 kHz, or equivalently tone 16. The pilot tones can be used to resolve sample timing at the receiver. The pilot tones will never be used for data, but will always send bits (0,0) from a 4QAM constellation (the constellation point in the first quadrant).

Cyclic Prefix Block

As discussed in Chapter 6, a cyclic prefix can help to make a channel look circular so that equalization can occur more easily in the frequency domain. In the downstream direction, a 32-sample cyclic prefix is appended to the 512 time-domain samples. Basically, the prefix consists of samples 480 through 511. The output sequence of the DMT transmitter consists of 544 samples given by $x_{480}, x_{481}, \dots, x_{511}, x_0, x_1, \dots, x_{511}$. The cyclic prefix for the upstream channel is four samples long and contains samples x_{60} through x_{63} from the output of the upstream IFFT. Thus the output sequence in the upstream direction consists of 68 samples given by $x_{60}, x_{61}, \dots, x_{63}, x_0, x_1, \dots, x_{63}$.

It is important to understand and distinguish the data frame rate and the frame rate of a DMT symbol as well as to understand why the bin spacing is 4.3125 kHz. The cyclic prefix as well as the synchronization frame play a role in these rates and frequencies. The data frame rate, as explained earlier, is exactly 4 kHz (or 4 kbaud). Because every 69th frame is a synchronization frame, the frame rate must be slightly higher than the data frame rate and is given by Eqn. 7.9.

Eqn. 7.9

$$\begin{aligned} R_{\text{frame}} &= \frac{69}{68} R_{\text{data}} \\ &= \frac{69}{68} 4000 \\ &\approx 4.0588 \text{ kHz} \end{aligned}$$

Given the number of samples in the downstream direction (544) per frame, Eqtn. 7.10 gives the sample rate leaving the ATU-C (and arriving at the ATU-R).

Eqtn. 7.10

$$R_{\text{Sample, DS}} = \frac{69}{68} 4000 \frac{\text{frames}}{\text{s}} \left(544 \frac{\text{samples}}{\text{frame}} \right) = 2208 \frac{\text{Msample}}{\text{s}}$$

In the upstream direction, the sample rate leaving the ATU-R (and arriving at the ATU-C) is similarly given by Eqtn. 7.11.

Eqtn. 7.11

$$R_{\text{Sample, US}} = \frac{69}{68} 4000 \frac{\text{frames}}{\text{s}} \left(68 \frac{\text{samples}}{\text{frame}} \right) = 276 \frac{\text{ksample}}{\text{s}}$$

The frequency-domain resolution of an FFT (the spacing of points after the transform) with $2N$ real time-domain samples is given by Eqtn. 7.12.

Eqtn. 7.12

$$\text{Res}_{\text{FFT}} = \frac{R_{\text{Sample}}}{2N}$$

For the downstream direction, a simple calculation with $2N=512$ and a sample frequency of 2.208 MHz give a bin spacing in the frequency domain of 4.3125 kHz. Likewise, for the upstream direction $2N=64$ and a sample rate of 276 kHz, the resulting bin spacing is also 4.3125 kHz.

Time-Domain Equalization Block

Though a DMT system can do equalization most efficiently in the frequency domain, a time-domain equalizer is generally present at the front end of the receiver. This equalizer generally serves two purposes. First, the equalizer removes any intersymbol interference from the channel that is longer than the cyclic prefix. Such interference would otherwise cause a symbol to interfere with the next symbol in time, and the channel would not look circular. Second, the time-domain equalizer can be used to partially bandpass the incoming signal and filter out-of-band energy. For example, the time-domain equalizer at the input

to an ATU-C might contain a low pass section to reduce energy that is below tone number 33 and a high pass section to reduce energy that is above 138 kHz. In general, the implementor decides on the length and type of the time-domain equalizer used in an ADSL receiver. The filters are generally adaptive in nature and are trained to initial optimal values during initialization.

Demodulation Block with FDQ

The demodulator in an ADSL receiver very much follows from the modulator. At the ATU-R, the heart of the demodulator is basically an FFT operating on 512 real-valued points. Note that before the FFT is performed, the 32-sample cyclic prefix is removed. After the FFT is performed, only the lower 256 complex FFT outputs are kept, and the other 256, which are complex conjugates, are dropped. Eqtn. 7.13 describes the demodulation mathematically.

Eqtn. 7.13

$$Z'_k = \sum_{i=0}^{511} x_i e^{-j \frac{2\pi k i}{256}} \text{ for } k = 0..255$$

Here the Z'_k are the unequalized values to be decoded by each bin's QAM constellation decoder.

At the ATU-C, the heart of the demodulator is an FFT using 64 real time-domain samples. Again, the four-sample cyclic prefix is discarded before the FFT block. Only the first 32 FFT outputs are kept, and their complex conjugates are dropped. Eqtn. 7.14 describes the FFT.

Eqtn. 7.14

$$Z'_k = \sum_{i=0}^{63} x_i e^{-j \frac{2\pi k i}{32}} \text{ for } k = 0..31$$

At both the ATU-R and ATU-C, frequency-domain equalization is done after the FFT operation. As discussed in Chapter 6, this process is accomplished by simply using one complex multiply for each bin. The multiplier values are determined during initialization of the ADSL link and are normally updated either during the synchronization frame or on a frame-by-frame basis.

Constellation Decoding and Bit Extraction

After the demodulation and equalization, the values for each bin are individually decoded using a QAM constellation decoder. Decoding involves the same constellation mapping on each bin as used in the transmitter. Bins carrying zero bits do not have to be decoded.

The decoded bits from each bin are sent to either the fast or interleaved buffer, using the reverse procedure of how they were distributed. This procedure is known as *bit extraction* and is essentially the reverse of tone ordering.

The rest of the receiver process is the reverse of the transmitter process. Specifically, deinterleaving is done on the interleaved path, and Reed-Solomon decoding and descrambling are done independently on each path. The resulting bit streams are passed through the CRC check block. Note that this block does not change any of the payload bits. Deframing is then performed, at which point the fast and sync bytes are decoded. If necessary, the AEX or LEX bytes are stuffed onto the appropriate channels, or a byte is removed from the end of the bytes of an indicated AS or LS channel. At this point, all the incoming data are passed out onto the appropriate interface (including all LS and AS channels as well as AOC and EOC information).

Timing Recovery

For proper operation of most systems, the sampling clock at the transmitter and receiver must be exactly the same frequency. Put another way, the transmitter and receiver clocks must be locked. Typically, locking is done by one end being the master clock and the other end recovering the clock. In an ADSL system, either end can be the master, and the decision is made during link initialization. Most often the ATU-C is the master, and the ATU-R does clock recovery. In this case, we say that the ATU-R is *loop timed*. Loop timing can be done in many ways but most often involves a *phase-locked loop* (PLL). The ATU-R basically needs to recover the 2.208 MHz sampling clock. All other clocks and timing can be found from that clock, including the frame rate, the superframe rate, and the upstream transmit clock. In addition, the upstream sampling clock of 276 kHz can be derived from the 2.208 MHz sampling clock. A modem may have to use the synchronization frame to find superframe boundaries.

DMT ADSL Overhead

As discussed in the framing sections, ADSL has several overhead channels. The overhead includes the EOC, AOC, and the ibs. The uses of the overhead are discussed in the next few sections.

EOC Channel

As discussed previously, the EOC is carried in the fast byte of the fast channel. The ATU-C uses the EOC to retrieve information from and write information to the ATU-R during runtime. An EOC message contains five fields spanning 13 bits, as shown in Table 7.4. An EOC message always starts with a fast byte in an even frame number and continues into the fast byte from the following odd frame.

Table 7.4 Format of an EOC Message

Bits	Description	Notes
1,2	Address Field	00—ATU-R address 01—unused 10—unused 11—ATU-C
3	Data/Opcode	0—Data (used for reads and writes) 1—Opcode
4	Odd/Even Byte	Used for multibyte transfers to delineate successive bytes
5	Autonomous Message Field	Set to 0 by ATU-R to send 'dying gasp' message
6–13	Information Field	Data or op codes

Information that can be read from the ATU-R includes the ATU-R vendor ID, the version number, serial number, results of a self-test, line attenuation, SNR margin, and configuration information. This information is held at the ATU-R in registers that the ATU-C can read via the EOC. Additionally, the ATU-C can use the EOC to command the ATU-R to perform a self-test, send corrupt CRC values, and prepare to receive corrupt CRC values.

Note

Often corrupt CRCs are generated to make sure that other performance information (the indicator bits) are properly logging erroneous frame information.

The EOC protocol is very simple. The ATU-C is the master of the channel, and the ATU-R is the slave, responding to commands from the master. The protocol uses a message/echo-response scheme. To send a message to the ATU-R, the ATU-C sends the message three consecutive times. These three messages represent three outstanding messages and one outstanding command. Upon receiving three identical EOC messages, the ATU-R echoes the messages by sending them back to the ATU-C. Upon receiving three

VDSL Overview

In this chapter:

- Basic deployment locations of VDSL
- Requirements for VDSL systems
- Proposed modulation methods for VDSL
- VDSL in WAN networks

ADSL satisfies the need to send up to 8 Mbps or so over twisted pairs. However, there is still a need to send more data at even higher speeds. VDSL fills this void by delivering up to 50 Mbps. Unlike ADSL, VDSL offers both symmetric and asymmetric configurations. Typically, VDSL uses far more bandwidth than ADSL uses and has a much shorter reach due to the higher bit rates supported. This chapter covers the basic goals of this relatively new technology and presents some candidate implementations. As with ADSL, some implementations promote DMT, and others use CAP/QAM techniques. The chapter also includes a brief discussion of how WAN networks might implement VDSL.

VDSL Deployment Locations

Like ADSL, VDSL can provide frequency separation from baseband, leaving room for Plain Old Telephone Service (POTS). (As you will see later, in addition to POTS, other services such as ISDN can also be run on the same twisted pair as VDSL.) Thus each end of a twisted pair carrying VDSL and POTS uses a splitter to separate the two signals. At the subscriber's location, the modem terminating VDSL is called the *VTU-R*. The very high speeds of VDSL suggest that some sort of high-speed optical network will feed a bank of centralized VDSL modems. This central point might be a CO of some remote optical

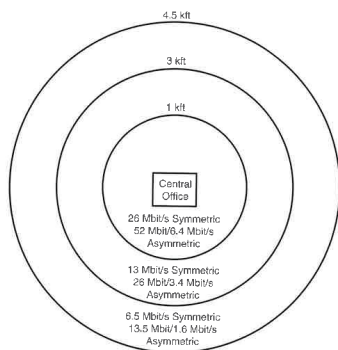
node. For this reason, the peer modem to the VTU-R modem is called the VTU-O (as opposed to VTU-C, which seems to follow from the ADSL notation), denoting an optical feeder.

Note

However, other methods of feeding VDSL banks are certainly possible. For example, VDSL can be used as a means to multiplex and carry multiple T1 streams over a short distance.

Figure 9.1 shows an example of VDSL range and reach from a central deployment location. If the location is a CO, the actual area to which VDSL could be deployed, or the coverage area, would be much smaller than the area served by the CO (typically extending well beyond 12 kft). Limiting deployment to this percentage of the overall serving area severely limits the business potential of VDSL.

Figure 9.1 Range and reach of VDSL from a central location.

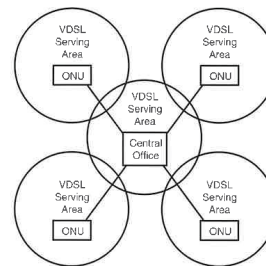


Instead, VDSL deployment will likely take place from the CO as well as from *optical network units* (ONUs). These nodes typically are deployed in, and serve, neighborhoods, industrial parks, and other areas with high telecommunications traffic patterns. ONUs are sometimes used in context when discussing technologies such as fiber to the curb (FTTC),

fiber in the loop (FTTL), and fiber to the neighborhood (FTTN). The common feature of these technologies is that high-capacity fiber is deployed to an ONU that serves a small area. The media connecting subscribers to the ONU can be coaxial cable, some sort of wireless link, or more probably, a twisted pair. The combination of a high-capacity link to a serving node, the small area served by each node, and the popularity of twisted pairs connecting subscribers to the serving node make a network utilizing ONUs a perfect fit for a technology like VDSL.

The example in Figure 9.2 shows an area with VDSL deployment using ONUs to serve more remote areas and VDSL deployment from the CO (CO) to serve less remote areas. In the figure, simple optical links serve each ONU. In reality, optical rings or some other type of optical distribution might be more practical.

Figure 9.2 Coverage area of VDSL when deployed from remote ONUs.



The total optical bandwidth available to an ONU is usually not greater than the sum of potential bandwidths of all ONU users. For example, if an ONU serves 20 subscribers and each has a 50 Mbps VDSL link, the total potential bandwidth for that ONU is 1 Gbps, much more than the ONU would normally be fed. The ratio of the optical bandwidth available to the ONU to the potential aggregate bandwidth of all users is called the *oversubscription ratio*. Oversubscription ratios are engineered so that performance remains reasonable for all users.

Crosstalk Issues

As with FDM ADSL, VDSL does not have self-NEXT. The trend toward technologies with no self-NEXT, or at least with limited self-NEXT, has evolved because of the increased performance achievable and, in some cases, the design simplifications possible with such implementations. Modulation schemes for VDSL and how they avoid self-NEXT is discussed later in this chapter.

Many of the crosstalk analyses in previous chapters are applicable to VDSL. In addition, the short deployment range of VDSL poses several new crosstalk possibilities. Consider the two deployment configurations in Figure 9.3 and Figure 9.4.

Figure 9.3 Crosstalk scenario with VDSL and another DSL technology in the CO mix.

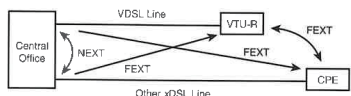


Figure 9.4 Crosstalk scenario with VDSL and another DSL technology in the customer premise (CP) mix.

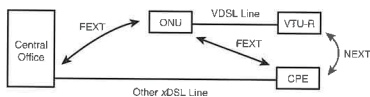


Figure 9.3 is sometimes called CO mix.¹ In Figure 9.3, VDSL and another xDSL technology are both provided from the CO and share a binder group between the CO and some point at which the VDSL is removed. (The VDSL might be terminated at this point, or it might be routed into a different binder group.) The various types of NEXT and FEXT that can exist are also shown in Figure 9.3. The FEXT between the two endpoints is normally not a factor because the distance is usually large enough to attenuate the disturbing signal. However, other FEXT paths may exist as well. Whether these FEXT signals will be significant or not depends on the lengths of the two channels as well as the frequencies used by the upstream and downstream channels.

In the configuration shown in Figure 9.4, the VDSL signal is terminated at a remote ONU and the other xDSL signal at the CO. At one point, the twisted pair segments carrying the signals share a common binder. This configuration is sometimes called the CP mix.² Note that it is not necessary for the VTU-R and the other xDSL CPE modem to be located in the same building, but the twisted pairs carrying them must share a binder group for some length near the respective endpoints. Figure 9.4 also shows the possible significant NEXT and FEXT paths for the CP mix configuration. The significance of the FEXT paths depends on the location of the ONU as well as the frequencies used by the channels. Note that the endpoints of the VDSL circuit can contribute either NEXT or FEXT into the endpoints of the other xDSL technology, depending on the deployment configuration. This flexibility adds a new degree of freedom in analyzing crosstalk.

Of particular interest is the coexistence of ADSL and VDSL. In the CO mix configuration, ADSL and VDSL are both provided from the CO and share a binder group between the CO and the point at which the VDSL is removed. (The VDSL might be terminated at this point, or it might be routed into a different binder group.) For no NEXT to exist between these services at the CO, the following two conditions must exist:

- The downstream VDSL signal must not overlap the upstream ADSL signal (the VDSL downstream signal must start higher than 138 kHz).
- The downstream ADSL signal must not overlap the upstream VDSL signal (the VDSL upstream signal must not overlap 138 kHz to 1.1 MHz).

In addition, the downstream ADSL signal may cause a reasonable level of FEXT into the downstream VDSL signal since the distance between the ADSL transmitter at the CO and the VTU-R receiver is small. Likewise, FEXT from the VDSL signal may pose a problem to the upstream ADSL signal.

For no NEXT to exist between the ADSL and VDSL in the CP mix configuration, the upstream and downstream frequency spectrums of the respective technologies must not overlap. Additionally, VDSL FEXT into the downstream ADSL signal might occur, and ADSL FEXT can occur into VDSL depending on the ONU location. Owing to the short loop lengths supported by VDSL, the FEXT, though less severe than NEXT, might have a detrimental impact on the ADSL downstream.

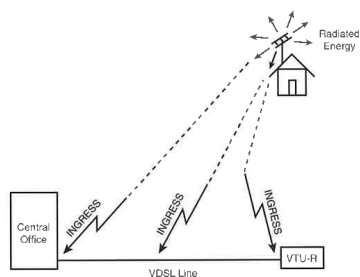
Because of the need to eliminate NEXT between ADSL and VDSL systems and the possibility that FEXT may indeed cause performance degradation, the spectrum allotted to VDSL falls above that used by either the upstream or downstream ADSL channels when ADSL and VDSL are both present. In addition, when dealing with other technologies, the deployment configuration of VDSL must be taken into account to decide which points

contribute NEXT or FEXT to other points. The specific frequency allocations for VDSL are discussed later in this chapter (after other influencing factors are discussed).

RFI Issues

VDSL receivers must deal with the issue of radio frequency interference (RFI). Included in RFI issues are *ingress* and *egress*. The cause of RFI ingress is in-band radio waves from nearby antennae incident upon a twisted pair carrying the VDSL. An amateur radio antenna is a good example of an RFI ingress disturber. Figure 9.5 illustrates this basic situation.

Figure 9.5 RFI ingress into VDSL due to a local transmitter.



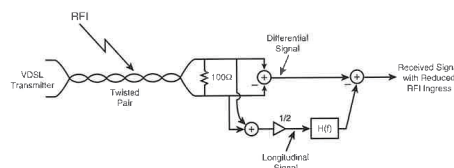
The factors influencing the amount of ingress include the power output of the antenna, the distance between the antenna and the twisted pair, the relative orientation and shielding properties of the binder group, and the balance of the twisted pair itself. Normally, the RFI ingress excites each wire in the twisted pair, thus creating a longitudinal ingress signal on the pair. Because the balance of the twisted pair is not ideal (normally it is 30 dB to 35 dB for high frequencies), some of the ingress leaks into the differential signal.

RFI ingress signals are often very narrow in bandwidth when compared to the VDSL signal. This characteristic is helpful, as the ingress will affect only a small portion of the usable bandwidth.

Another issue is that the ingress signal may be very large compared to the received VDSL signal. In this case, the analog front end of the receiver must be carefully designed so as not to saturate. In addition, some action must be taken so that the analog-to-digital converter (ADC) performs with the proper precision. The issue here is that a large ingress signal will swamp the converter, causing it to waste bits of precision on this unusable signal instead of on the received VDSL signal. Because the VDSL signal is quantized with less precision, the quantization noise, or imprecision due to the conversion process, increases, effectively decreasing the achievable bit rate on the channel. The ADC must have enough range to handle the ingress signal as well as enough precision to adequately quantize the VDSL received signal. Even after quantization, further RFI cancellation may be necessary in the digital domain as documented in Jeong and Yoo's article, "An Efficient Digital RFI Canceller for DMT-Based VDSL," and Wiese and Bingham's article, "Digital Radio Frequency Cancellation for DMT VDSL."^{3,4}

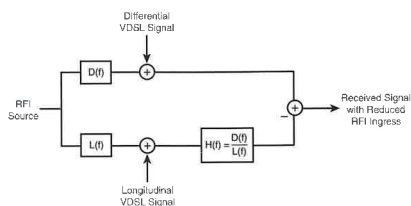
To facilitate the design of the ADC, the analog front end of a VDSL receiver might employ circuitry to reduce RFI ingress. Recall that ingress is normally present both longitudinally and differentially on a twisted pair. Figure 9.6 shows a method to use the longitudinal ingress signal to reduce the metallic ingress signal.

Figure 9.6 A method to reduce ingress that uses a longitudinal signal.



The filter tries to match the unbalance of the twisted pair that is essentially the mechanism by which the ingress reaches the differential signal. If the filter matched the unbalance perfectly, the signal at the output of the summation would consist only of the received VDSL signal (as well as crosstalk and background noise). Figure 9.7 illustrates this concept via a block diagram.

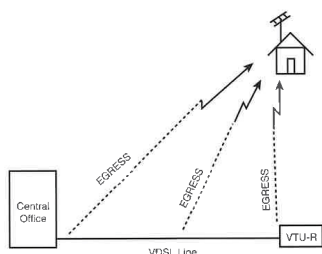
Figure 9.7 A specific implementation of an RFI ingress cancellation circuit.



A specific implementation is described in Cioffi, et al.'s article, "Analog RF Cancellation with SDMT."⁵ Ingress reduction on the order of 35 dB can be achieved by this method. Note that this would be equivalent to gaining more than five bits of precision in the ADC.

RFI egress is also a concern with VDSL. Egress is illustrated in Figure 9.8.

Figure 9.8 An example of RFI egress.



Here the VDSL signal is radiated from the twisted pair and can disturb the signal received by local antennae if these received signals overlap the VDSL spectrum. To combat this problem, the VDSL transmit power in frequency regions reserved for wireless or radio

services must be lowered (these regions were discussed in Chapter 4, "Power Spectral Densities and Crosstalk Models"). Normally, 20 dB reductions in these areas will be adequate to mitigate problems from VDSL RFI egress.

VDSL Requirements

This section discusses the more fundamental requirements put forth for VDSL systems. These requirements are captured in the technical reports "Very High Speed Digital Subscriber Line System Requirements"⁶ and "Transmission and Multiplexing: Access Transmission Systems on Metallic Access Cables; Very High Speed Digital Subscriber Line (VDSL); Part I: Functional Requirements."⁷ It should be noted that at the time of publication, the requirements were not completely static. Also, some aspects of the candidate implementations in this chapter may not completely satisfy all requirements in the aforementioned reports.

Rate and Reach Goals

In contrast to ADSL, VDSL technology provides either symmetric or asymmetric data rates to a subscriber. The reason that both need to be supported is that some VDSL applications are best served by an asymmetric connection, whereas others are best served by a symmetric connection. Video on demand (VoD), for example, is an application that benefits from an asymmetric connection. Another application, ATM LAN extension, might better be served by a symmetric connection. In addition, applications that do not even exist might turn out to be application drivers for VDSL. Defining symmetric and asymmetric modes of operation up front does not limit the types of application that might one day use VDSL.

Note

This situation developed with ADSL. The initial application driver was VoD; however, Internet/LAN access later became an even stronger application driver.

Table 9.1 shows asymmetric data rate goals for VDSL along with the approximate reach goals for each configuration. How the rates are provisioned (hard configuration as compared to automatically adapting to the loop) is yet to be determined. At the very low end of the asymmetric rates (6.5 Mbps/0.8 Mbps), VDSL approaches the performance limits of ADSL. Though it might seem logical to define even lower rates for VDSL and not bother with ADSL, this approach would most probably cause modems to be more complex than necessary and would impact cost effectiveness.

Table 9.1 Asymmetric Data Rate Goals for VDSL

Length Classification	Downstream Rate (Mbps)	Upstream Rate (Mbps)	Reach Goal (kft)
Short	52	6.4	1
Medium	26	3.2	3
Long	13	1.6	4.5
Long	6.5	0.8	6

Table 9.2 shows rate and reach goals for symmetric VDSL operation. Many of these rates are primarily intended to support ATM access systems (for example, 26 Mbps would support 25.6 Mbps ATM adapters along with some overhead). It is expected that some of the symmetric as well as asymmetric rate and reach goals will churn and evolve as VDSL goes through the standardization process. However, Tables 9.1 and 9.2 provide good insight into the probable operating ranges of VDSL.

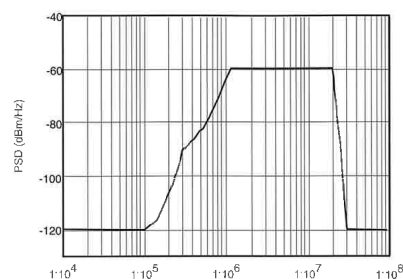
Table 9.2 Rate and Reach Goals for Symmetric VDSL Operation

Length Classification	Downstream Rate (Mbps)	Upstream Rate (Mbps)	Reach Goal (kft)
Short	26	26	1
Medium	13	13	3
Long	6.5	6.5	4.5

Frequency Allocation

The frequency ranges available for VDSL to use are limited by the need for compatibility with other DSL technologies as well as RFI concerns discussed previously. The base power spectral density (PSD) mask for VDSL appears in Figure 9.9.

Figure 9.9 The basic PSD requirements of a VDSL transmitter.



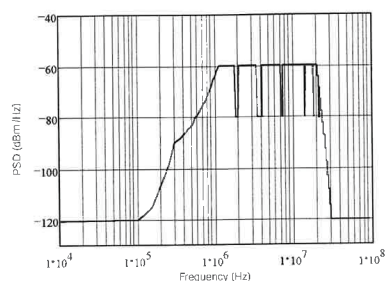
This basic template has a PSD of -60 dBm/Hz from 1.1 MHz to 20 MHz. This range would normally be considered the usable part of the spectrum, and the rest would be considered sidebands. In the ADSL band, the PSD rolls off so as to minimize the crosstalk degradation into adjacent ADSL lines. A POTS or ISDN link could be run on the same line as a VDSL system using this mask, given that a splitter was used to separate the signals.

The mask shown in Figure 9.9 may cause excess egress and interfere with local antennas. In this case, a notching option can be applied resulting in the mask shown in Figure 9.10. Here bands known to be prone to causing egress into existing radio/wireless services are notched down to a level of -80 dBm/Hz (essentially reducing the egress power in the notched areas by a factor of 10). The notched frequency regions can still transmit information if the received SNRs across the regions are favorable.

Note

Of course, if egress is a problem that causes a band to be notched, ingress may likely exist in the same band as well (the local antennas may transmit as well as receive), further degrading the SNR and causing the notched bands to be useless for transmitting data.

Figure 9.10 A VDSL PSD with notches to reduce RFI egress.



A power boost option has also been proposed for VDSL. When ADSL compatibility is not necessary (perhaps because a service provider did not deploy ADSL), this option would not only boost the VDSL level in the 1.1 MHz to 20 MHz region but also allow the usable end to stretch down to about 300 kHz. Below 300 kHz, the PSD would roll off and eventually intersect with that shown in Figure 9.9 so that ISDN and POTS could be supported.

It may take trial deployments of VDSL to determine which spectral mask properties are more desirable. Real-life deployment may also flag new problems needing to be overcome. In addition, VDSL spectrum requirements may depend on territory. In any case, Figure 9.9 provides at least a starting point to understanding VDSL PSD development.

One final note on the VDSL PSDs discussed is that the masks do not assume anything about symmetric versus asymmetric operation nor upstream and downstream channels. As I discussed previously, implementations are free to use the frequencies within the defined mask in any way they want, although self-NEXT avoidance will likely be necessary for performance reasons. The following sections discuss different proposed implementations of VDSL. You will see that several methods are employed to abide by the PSD requirements and, at the same time, avoid self-NEXT.

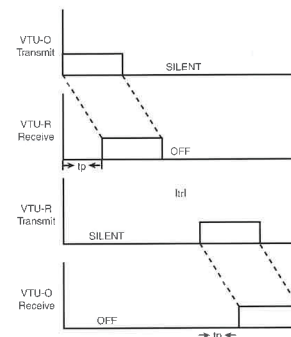
Modulation Methods for VDSL

Several different modulation methods have been developed to satisfy the requirements of VDSL. At the time of this writing, VDSL was still very early in the standardization process, and many details of the proposals are yet to be specified (for example, framing content is not yet known). Though many proposals currently exist for VDSL, most fall into two categories: a time-domain duplexing (TDD) category and a frequency-domain duplexing (FDD) category. Different modulation methods within each category exist as well. The ideal method by which standardization can be achieved is to choose a modulation method and fill in all of the implementation details necessary for a standards document. In reality, one or more of the methods are likely to become standardized (or at least implemented by service providers), mainly because of the lack of consensus or compromise on behalf of the standardization body members. The following sections describe some of the candidate implementations.

TDD Approaches

TDD is a half-duplex approach for sending signals between two endpoints. Basically, one modem transmits while its peer modem only receives. After a set amount of time, the process reverses, and the first modem only receives while its peer transmits. This basic concept is shown in Figure 9.11.

Figure 9.11 An example of TDD.



Sometimes this method is known as “ping-pong” for obvious reasons. In each direction, the entire usable frequency band can be used to send information, and thus no separate upstream and downstream PSD need be defined.

The most popular TDD approach proposed for VDSL is the synchronous DMT approach (SDMT). Essentially, this technology employs many of the same fundamental blocks used in ADSL along with the capability to gate the output of the transmitter. (Though many of the same blocks are used, the parameters, such as FFT/IFFT size, will be different.) Figure 9.12 shows a block diagram of an SDMT transmitter and receiver.

Figure 9.12 Block diagram of an SDMT transmitter and receiver.

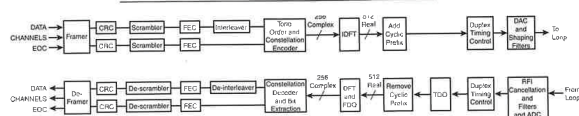


Figure 9.12 is very similar to the DMT-based ADSL diagram discussed in Chapter 7, “ADSL Modulation Specifics.” The actual implementation of these blocks will no doubt undergo changes in the early stages of VDSL. A discussion of the blocks in order to explain the main concepts of SDMT follows.

Non-Modulation-Specific Blocks

Some of the blocks in Figure 9.12 should be well understood by now, and implementation details are not necessary for an understanding of SDMT. These blocks include the scrambler and descrambler, the FEC blocks, and the interleaver and de-interleaver blocks. In addition, block implementation is not yet static. Some of the common implementations seen for other DSL systems, including self-synchronizing scramblers and descramblers, Reed-Solomon FEC blocks in GF(256), and convolutional interleaving, are likely implementation candidates.

To a lesser extent, framing and tone ordering/constellation encoding are not modulation-specific blocks. Of course, the tone-ordering and constellation-encoding algorithms must know the number of subchannels in the modulation path. For SDMT, the number of subchannels supported is 256. As with DMT-based ADSL, some of these subchannels must not be used so that the SDMT output spectrum conforms to the VDSL PSD mask. Because SDMT is a half-duplex modulation scheme, each transceiver will use 256 tones, as opposed to DMT ADSL, where upstream and downstream transmitters used a different number of tones.

SDMT Modulator

The SDMT modulator accepts a vector of 256 complex values from the constellation encoder. Each of these values represents a constellation point for a subchannel or equivalently, the cosine wave magnitude (real part) and sine wave magnitude (imaginary part) of the time-domain waveform produced for a subchannel. The complex conjugates of the 256 complex values are then appended to the vector so that the new vector has complex conjugate symmetry. An IFFT is performed, as given by Eqn. 9.1.

Eqn. 9.1

$$x_k = \sum_{l=0}^{511} Z_l e^{j2\pi k l / 512} \text{ for } k = 0..511$$

Here the vector Z represents the vector with complex conjugate symmetry. The output vector of the IFFT contains 512 time-domain values (all real). This IFFT transform is identical to that done in a downstream DMT ADSL transmitter.

Cyclic Prefix Block

The cyclic prefix block adds the last L sample points of the 512-point time-domain vector to the beginning of the vector. L can be any integer less than or equal to 64. The new time-domain vector will be given by $(511-L, 511-L+1, 511-L+2, \dots, 0, 511, 0, 1, \dots, 511)$. Adding a cyclic prefix makes a channel appear to be circular so that equalization can be done in the frequency domain. The 512 time-domain samples along with the L sample cyclic prefix represent one SDMT symbol. The choice of L is further discussed in the next section.

Note

If the channel's impulse response is longer than the cyclic prefix, the channel will not look circular. A time-domain filter in the receiver must be used in this case.

Superframe Structure

SDMT uses a superframe structure with duration of 500 μ s. Thus the superframe frequency is 2 kHz. A superframe consists of a number of downstream symbols followed by a quiet period and then a number of upstream symbols followed by a quiet period. Figure 9.13 illustrates an SDMT frame structure.

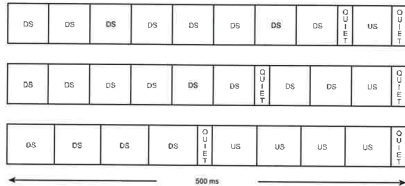
Figure 9.13 Sample SDMT frame structure.



By varying the number of downstream symbols sent and the number of upstream symbols sent, the average upstream and downstream bit rates as well as the bit rate ratio can be controlled.

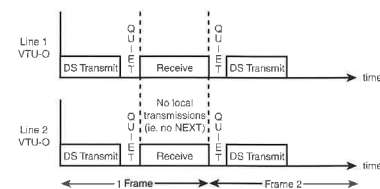
Consider the examples shown in Figure 9.14. The first shows an 8:1 downstream to upstream ratio, the second a 2:1 ratio, and the third a 1:1 ratio (symmetric example).

Figure 9.14 Possible implementations of an SDMT frame for different upstream and downstream bit-rate ratios.



Self-NEXT avoidance is relatively straightforward when lines sharing a binder group use the same symmetry ratio. In such a case, the frame structure of all lines must be aligned so that all VTU-Os transmit simultaneously and likewise receive simultaneously. If no VTU-O transmits while another VTU-O receives, then self-NEXT does not exist. Aligning the superframes of all VDSL lines in a binder group is not trivial, but is certainly possible. For the 2:1 ratio case, self-NEXT elimination is illustrated in Figure 9.15.

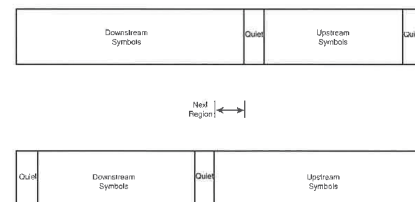
Figure 9.15 Self-Next avoidance in an SDMT system.

**Note**

A proposal on how to align the superframes of all VDSL lines in a binder group includes using a common network clock for synchronization, using stuff and delete methods,⁸ and even using GPS signals.⁹

When mixing different data rate ratio configurations within a binder group, complete self-NEXT elimination is not possible. However, frames can be arranged to reduce the occurrence of self-NEXT. See Figure 9.16 for an example of limited self-NEXT due to frame alignment of a symmetric and an asymmetric VDSL mix.

Figure 9.16 Two SDMT frames with different upstream-to-downstream ratios aligned to limit the amount of mutual NEXT.



Though it is not possible with this scheme to perfectly eliminate self-NEXT, the self-NEXT levels are predictable and line rates can be adjusted accordingly when self-NEXT is present.

Sampling Considerations

Up to now, the number of samples in a frame has been discussed, as well as the length of a superframe. Not discussed yet is the duration of a sample (or the inverse of that being the sample rate). Following from the sample rate is the number of frames in a superframe, the highest frequency used in the system, and the width of each frequency bin. Remember that all blocks up to the output of the modulator have functionality that does not depend on the output sample rate. Of course, the reason is that the length of a superframe is fixed (500 μ s); the higher number of frames per superframe implies that the blocks leading up to the modulator must perform their functions more quickly.

Three sample rates are proposed for SDMT: 11.04 MHz, 22.08 MHz, and 44.16 MHz. For these rates, Table 9.3 shows the maximum operating frequency, bin width, number of samples per superframe, and approximate number of frames per superframe. When more frames exist per superframe, the net data rate in each direction can be higher. Thus short lines supporting the extremes of the required VDSL rates use a sample rate of 44.16 MHz.

Table 9.3 Maximum Operating Frequency, Bin Width, Number of Samples per Superframe, and Approximate Number of Frames per Superframe

Sampling Frequency (MHz)	Maximum Frequency (MHz) (approximate)	Bin Width (kHz)	Samples per Superframe	Frames per Superframe
11.04	5.52	21.5625	5520	10
22.08	11.04	43.125	11040	21
44.16	22.08	86.25	22080	43

For the lowest sample rate, the total number of samples per superframe is 5520. Given that each frame contains 512 base samples along with a cyclic prefix of between 0 and 64 samples, the maximum possible number of frames per superframe (assuming a 40 sample cyclic prefix) is 10, leaving no samples to be used for the quiet period (30.4 μ s). This scenario is not very practical, as some line turnaround time will be necessary for proper half-duplex operation.

The length of the quiet time can be increased by reducing the number of samples used in the cyclic prefix (which may increase the front-end filtering needs of the receiver) or by

decreasing the number of frames per superframe to nine. The latter method would leave, using the maximum length cyclic prefix, a minimum of 336 samples for the quiet times, or approximately 30.4 μ s. This time would be divided between the two required quiet periods (though it might not be evenly divided). For the higher sample rates, similar calculations can be done to determine the lengths of the cyclic prefixes and quiet times. It is intended that many of the parameters including the sample rate, the number of frames per superframe, and length of the cyclic prefix and quiet times be negotiated during startup.

RFI Cancellation Circuitry

Generally, the front end of an SDMT receiver contains an RFI cancellation block. As discussed earlier, this block is necessary to combat ingress from local RFI sources. Normally, some type of cancellation is necessary before the analog-to-digital conversion process to adequately digitize the received signal with minimal quantization noise.

Receive Blocks

Much of the functionality in the receive path of an SDMT modem follows from the discussion of the receive path of a DMT modem. Time-domain equalization is done to shorten the channel impulse response to less than the length of the cyclic prefix. An FFT is done on a block of 512 received samples, yielding a complex vector with complex conjugate symmetry. Frequency-domain equalization is performed on the lower 256 values of the FFT output, each complex value representing the received component from a single bin. Constellation decoding and error correction is done before the data is demultiplexed by a framer.

Final Words on SDMT

As a candidate for VDSL, SDMT has several advantages. First, the use of TDD allows for easy control over the upstream and downstream data rate ratio. Second, because SDMT is half-duplex, a hybrid is not needed on VDSL modems, and the sharing of digital circuitry between the transmitter and receiver (the FFT/IFFT circuitry for example) is possible, allowing for complexity reduction. Third, the nature of multicarrier modulation allows for flexibility in dealing with narrowband interference from RFI and can easily reduce egress in a band by simply reducing the power on a tone.

One issue related to SDMT is the requirement to properly synchronize frames for VDSL lines in a common binder group. If all lines emanate from a single ONU, this synchronization might be feasible by using a common clock. However, when independent ONUs feed VDSL into the same binder group, achieving frame synchronization is much more difficult.

FDD Approaches

FDD approaches to transmission are similar to those discussed for ADSL. This approach uses full-duplex transmission and separates upstream and downstream transmissions by using different frequency bands for each. CAP/QAM-based techniques lend themselves nicely to an FDD approach, as the basic spectrum can be positioned anywhere, depending on the Hilbert pair filter responses in the case of CAP and center frequency in the case of QAM. A second method discussed here is an FDD method utilizing DMT. This method is sometimes known as Zipper.

Note

FDD approaches to transmission include CAP/QAM-based ADSL and FDM ADSL. Echo-canceled ADSL is partially an FDD approach, but not completely because some of the upstream and downstream bands overlap.

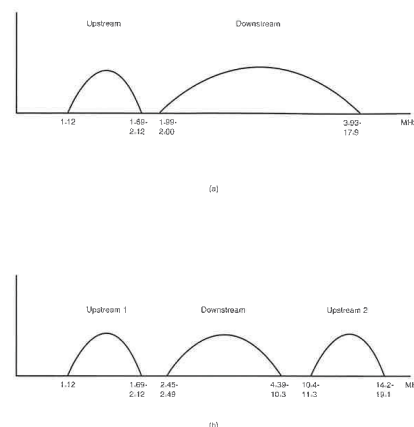
CAP/QAM-Based FDD Systems

CAP and QAM technologies offer another method of implementing VDSL. The basic blocks in the modulation system are similar to those used in CAP/QAM ADSL implementations. For VDSL, the proposed CAP/QAM implementation can run in symmetric mode as well as asymmetric mode. In asymmetric mode, the upstream channel's frequency range starts at about 1.12 MHz and can go up as high as about 2.12 MHz. Different symbol rates are defined as well as different constellation sizes to control the upstream data rate. The downstream channel begins at around 2 MHz and can range up to about 18 MHz. Again, different baud rates and constellation sizes can be used to provide different downstream data rates. Self-NEXT is avoided simply by using different frequency bands for transmission in the two directions.

Symmetrical service is handled in a very interesting way by CAP/QAM-based VDSL. For symmetric operation, two upstream channels exist: one from 1.12 MHz to 1.6 MHz, and one from 11.30 MHz extending up to 19.8 MHz. The downstream channel sits between the two upstream channels and ranges between 2.4 MHz and up to 10.7 MHz. Again no self-NEXT exists because the upstream and downstream channels do not overlap in frequency.

Figure 9.17 shows the various spectrums utilized by CAP/QAM VDSL for symmetric and asymmetric service.

Figure 9.17 Upstream and downstream spectra for CAP/QAM-based VDSL for (a) asymmetric rates and (b) symmetric rates.



The figure shows different downstream baud rate options (and thus frequency ranges). When mixing symmetric and asymmetric services, in some cases self-NEXT will occur. Note that the highest baud rate downstream carrier overlaps the higher upstream regions in the symmetric frequency plot, which would cause a problem. However, some of the other downstream symmetric options would not, as they end below the higher upstream channel. As CAP/QAM VDSL modems mature, some of the frequencies may change; however, the basic implementation strategy is likely to remain the same.

RF egress control is a bit more difficult for CAP/QAM-based modems. Recall that SDMT-based modems could simply lower the power in a subcarrier at frequencies where power reduction was necessary. CAP/QAM modems must use time-domain filters to do this. Digital filters can normally be implemented for this task, but their flexibility with respect to changes in the power reduced frequencies and the amount of power reduction is somewhat reduced.

DMT-Based FDD Systems

A second FDD method for VDSL uses DMT and is a relatively new modulation scheme named Zipper. Zipper is a time-synchronized version of DMT using 2,048 subchannels going up to 11.04 MHz. Each subchannel has a width of about 5.309 kHz. A subchannel in Zipper can be used for upstream transmission or for downstream transmission, but not for both. If all lines in a binder use a similar bin assignment, then no self-NEXT will be present. (The frames must be time aligned as well so that crosstalk is orthogonal to the received signal.) In addition, subchannel usage can be chosen in a way that minimizes NEXT interaction with other services. Zipper has many of the same benefits as SDMT with regard to ingress and egress handling flexibility. Though a relatively new addition to VDSL implementation, this modulation method may develop into a powerful tool for high-speed access.

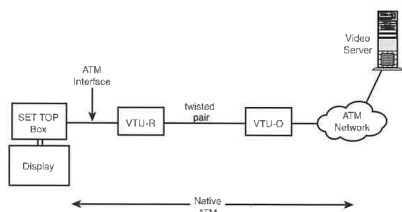
VDSL in WAN Networks

The speeds that VDSL supports make it suitable for many types of applications. Many applications that exist today may use VDSL as a transport mechanism; some applications yet to be developed may also use VDSL. A good example of VDSL usage is the full-service network described later in the chapter. The following sections describe video services and data services that utilize VDSL.

Video Services

The high-rate options of VDSL make it a very good access technology for VoD. Figure 9.18 shows a system using VDSL in this application.

Figure 9.18 VDSL used in a network providing video services.



The video server in this example sends digital video over ATM through access equipment to the VDSL loop. At the customer end, some type of video codec converts the digital video to the proper signal for the display screen. The digital video would likely be compressed, and the codec would perform decompression. It is also likely that PPP would not be used on this connection, as it might not be the most efficient protocol for carrying high-speed video. Another control channel might use PPP for authentication and billing.

Data Services

Many data service possibilities exist for VDSL. In the near future, VDSL might be overkill for residential Internet access and Web surfing. However, back-hauling data traffic between locations using twisted pairs might be more reasonable. VDSL could also be used to connect larger offices and corporations to data networks in lieu of fiber connections. As time goes on, more and more opportunities are likely to exist for VDSL to carry data.

The Full-Service Network

Because of the high bit rates supported by VDSL, it is often thought of as an access mechanism to a full-service network (FSN). This type of network is intended to serve all the communication needs of a subscriber, including voice, video, and data applications. Such an all-encompassing network could replace today's telephone system and cable system, and it could add extra features such as video telephony. A full-service access network (FSAN), consisting primarily of telephone and service providers, is working to piece together existing standards where available and stimulate new standards to define a FSN.

Summary

This chapter focused on VDSL. The basic needs that VDSL is intended to fulfill were discussed along with several candidate VDSL implementations. The approach differences of these implementations were highlighted, along with some pros and cons of each. Finally, VDSL network applications were discussed. The true direction and application drivers for VDSL are yet to be seen, but any technology promising the delivery speeds of VDSL has great potential.

Endnotes

1. K. S. Jacobsen, "Spectral Compatibility of ADSL and VDSL, Part I: The Impact of VDSL on ADSL Performance," ANSI Contribution T1E1.4/97-404, December 1997.

2. K. S. Jacobsen, "Spectral Compatibility of ADSL and VDSL, Part I: The Impact of ADSL on VDSL Performance," ANSI Contribution T1E1.4/98-035, March 1998.
3. Byung-Jang Jeong and Kyung-Hyun Yoo, "An Efficient Digital RF Canceller for DMT-Based VDSL," ANSI Contribution T1E1.4/97-422, December 1997.
4. Brian Wiese and John Bingham, "Digital Radio Frequency Cancellation for DMT VDSL," ANSI Contribution T1E1.4/97-460, December 1997.
5. J. Cioffi, et al., "Analog RF Cancellation with SDMT," ANSI Contribution T1E1.4/96-084, April 1996.
6. "Very High Speed Digital Subscriber Line System Requirements," Revision 13, ANSI Contribution T1E1.4/98-043, March 1998.
7. K. Foster, "Transmission and Multiplexing : Access Transmission Systems on Metallic Access Cables; Very High Speed Digital Subscriber Line (VDSL); Part I: Functional Requirements," ETSI TM6 Draft Technical Report, DTS/TM-06003-1, V0.0.7 February 1998.
8. J. Cioffi, "Add/Delete SDMT Solution for Unsynchronized VDSL Lines," ANSI Contribution T1E1.4/96-247, September 1996.
9. Sven-Rune Olofsson, et al., "Synchronizing Optical Networks Using GPS," ANSI Contribution T1E1.4/97-283, September 1997.

Index

Symbols

2B1Q (baseband code), 14
quars, 94

A

ABCD transmission

- parameters, 63
- bridged taps, 70
- loops, 71
- input impedance, 72-73
- insertion loss, 72-74
- source impedance, 72
- termination impedance, 73
- transfer function, 72-73
- lumped impedance, 65
- two-port networks, 63
- uniform twisted pairs, 66
- activation and acknowledgment (training DMT ADSL), 235
- addresses
 - dotted-decimal notation, 259
 - Ethernet, 255
 - Internet Protocol (IP), 259
- admittance, twisted pairs, 30
- ADSL (Asymmetric Digital Subscriber Lines), 13
- ATM, 240

- CAP demodulation, 248
- CAP modulation, 246–247
- de-interleaving blocks, 245
- demodulation, 241
 - forward error correction (FEC) blocks, 245
 - framing blocks, 242
 - interleaving blocks, 245
- modulation, 241
- QAM demodulation, 248
- QAM modulation, 248
- scramblers, 244
- trellis encoding, 245
- capacity, 142
 - calculating, 141
- CSA loops, 143
- loops with identical cross talk, 142
- standard loops, 145
- channels
 - beaver, 209
 - byte robbing, 215
 - byte stuffing, 215
 - framing, 211–214, 216
 - full duplex, 209
 - simplex downstream, 209
 - superframe structure, 211
- data frames, mux, 216
- disturbances
 - downstream, 104
 - upstream, 109

- DMT, 207
 - blokes, 209–210, 218–222, 225–228
 - downstream, 207
 - overhead, 228–230, 232
 - reduced overhead, 233, 234
 - training, 235–236, 238
 - upstream, 207
 - video, 233
- end-models, 207
- loops, 76
- models, 16
 - customer interface, 16
 - data link framing, 19
 - encapsulation, 19
 - line codes, 18
 - POTS splitter, 17
 - twisted pair, 17
- xTU-C modem, 17
- xTU-R modem, 17
- PSD
 - CAF, 112
 - DMT (downstream), 104
 - DMT (upstream), 109
 - QAM, 112
- standardization, 19
 - ADSL Forum, 20
 - Committee T1, 20
- transceivers, 207
- VDSL crosstalk, 287