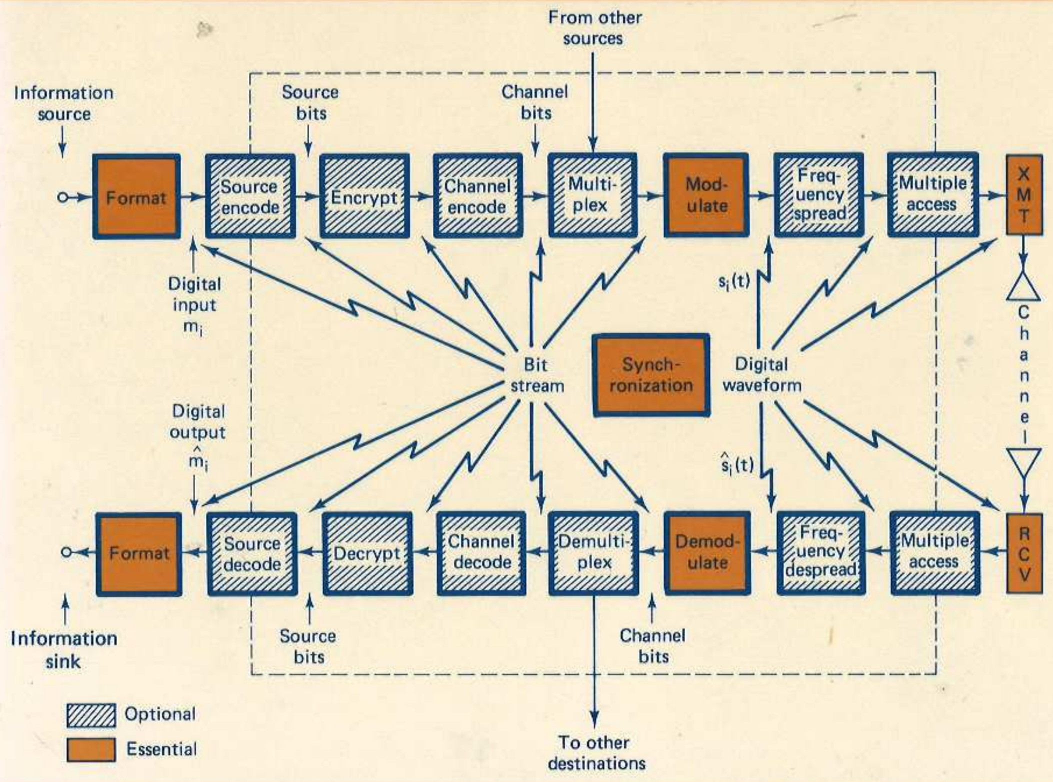
BERNARD SKLAR

DIGITAL COMMUNICATIONS
Fundamentals and Applications



DIGITAL COMMUNICATIONS
Fundamentals and Applications

BERNARD SKLAR

The Aerospace Corporation, El Segundo, California
and
University of California, Los Angeles

P T R Prentice Hall
Englewood Cliffs, New Jersey 07632

Library of Congress Cataloging-in-Publication Data

SKLAR, BERNARD (date)
Digital communications.

Bibliography: p.
Includes index.
1. Digital communications. I. Title.
TK5103.7.S55 1988 621.38'0413 87-1316
ISBN 0-13-211939-0

Editorial/production supervision and
interior design: Reynold Rieger
Cover design: Wanda Lubelska Design
Manufacturing buyers: Gordon Osbourne and Paula Benevento

Printed in the United States of America

20 19 18 17

ISBN 0-13-211939-0

Prentice-Hall International (UK) Limited, *London*
Prentice-Hall of Australia Pty. Limited, *Sydney*
Prentice-Hall Canada Inc., *Toronto*
Prentice-Hall Hispanoamericana, S.A., *Mexico*
Prentice-Hall of India Private Limited, *New Delhi*
Prentice-Hall of Japan, Inc., *Tokyo*
Simon & Schuster Asia Pte. Ltd., *Singapore*
Editora Prentice-Hall do Brasil, Ltda., *Rio de Janeiro*

To my mother, Ruth Sklar,
the memory of my father, Julius Sklar,
my wife, Gwen, and our children,
Debra, Sharon, and Dean

Contents

23

ION 51

?

90

Preface

This book is intended to provide a comprehensive coverage of digital communication systems for senior-level undergraduates, first-year graduate students, and practicing engineers. Even though the emphasis of the book is on digital communications, necessary analog fundamentals are included, since analog waveforms are used for the radio transmission of digital signals.

The key feature of a digital communication system is that it deals with a finite set of discrete messages, in contrast to an analog communication system in which messages are defined on a continuum. The objective at the receiver of the digital system is *not* to reproduce a waveform with precision; it is, instead, to determine from a noise-perturbed signal which of the finite set of waveforms had been sent by the transmitter. In fulfillment of this objective, an impressive assortment of signal processing techniques has arisen over the past two decades.

The book develops these important techniques in the context of a unified structure. The structure, in block diagram form, appears at the beginning of each chapter; blocks in the diagram are emphasized, as appropriate, to correspond to the subject of that chapter. Major purposes of the book are (1) to add organization and structure to a field that has grown rapidly in the last two decades, and (2) to ensure awareness of the "big picture" even while delving into the details. The signals and key processing steps are traced from the information source through the transmitter, channel, receiver, and ultimately to the information sink. Signal transformations are organized according to functional classes: formatting and source coding, modulation, channel coding, multiplexing and multiple access, spreading, encryption, and synchronization. Throughout the book, emphasis is placed on system goals and the need to trade off basic system parameters such as signal-to-noise ratio, probability of error, and bandwidth (spectral) expenditure.

ORGANIZATION OF THE BOOK

It is assumed that the reader is familiar with Fourier methods and convolution. Appendix A reviews these techniques, emphasizing those properties that are particularly useful in the study of communication theory. It is also assumed that the reader has a knowledge of basic probability and has some familiarity with random variables. Appendix B builds on these disciplines for a short treatment on statistical decision theory with emphasis on hypothesis testing—so important in the understanding of detection theory. Chapter 1 introduces the overall digital communication system and the basic signal transformations that are highlighted in subsequent chapters. Some basic ideas of random variables and the additive white Gaussian noise (AWGN) model are reviewed. Also, the relationship between power spectral density and autocorrelation, and the basics of signal transmission through linear systems, are established. Chapter 2 covers the signal processing step, known as formatting, the step that renders an information signal compatible with a digital system. Chapter 2 also emphasizes the *transmission* of baseband signals. Chapter 3 deals with bandpass modulation and demodulation techniques. The detection of digital signals in Gaussian noise is stressed, and receiver optimization is examined. Chapter 4 deals with link analysis, an important subject for providing overall system insight; it considers some subtleties usually neglected at the college level. Chapters 5 and 6 deal with channel coding—a cost-effective way of providing improvement in system error performance. Chapter 5 emphasizes linear block coding, and Chapter 6 emphasizes convolutional coding.

Chapter 7 considers various modulation/coding system trade-offs dealing with probability of bit error performance, bandwidth efficiency, and signal-to-noise ratio. Chapter 8 deals with synchronization for digital systems. It covers phase-locked-loop implementation for achieving carrier synchronization; bit synchronization, frame synchronization, and network synchronization; and some fundamentals of synchronization as applied to satellite links.

Chapter 9 treats multiplexing and multiple access. It explores techniques that are available for utilizing the communication resource efficiently. Chapter 10 introduces spread-spectrum techniques and their application in such areas as multiple access, ranging, and interference rejection. This technology is particularly important for most military communication systems. The subject of source coding in Chapter 11 deals with data formatting, as is done in Chapter 2; the main difference between formatting and source coding is that source coding additionally involves data redundancy reduction. Rather than considering source coding immediately after formatting, source coding has purposely been treated in a later chapter. It is felt that the reader should be involved with the fundamental processing steps, such as modulation and channel coding, early in the book, before examining some of the special considerations of source coding. Chapter 12 covers

some basic
concepts, as
public key c
If the b
the first six
second term
the course r
and 10.

ACKNOWLEDGME

This book is
Los Angeles
Corporation
pleasure to
carefully rev
tinual assist
nization. Pr
improvemen
thanks to D
with much e
I also v
considerable
12. Professo
of beneficial
and Andreas
portant impr
stitute, Dr.
fessor Ron
valuable rev
MA/COM L
in the chapte
Aerospace C
to pay speci
whose uniqu
The bl
the cover of
Sklar, "A St
IEEE Comm
IEEE to rep
My stu
ters of this b
students wh
want to exp

em parameters such
)ectral) expenditure.

ıds and convolution.
)perties that are par-
lso assumed that the
ıiliarity with random
: treatment on statis-
-so important in the
: overall digital com-
at are highlighted in
nd the additive white
elationship between
f signal transmission
he signal processing
on signal compatible
mission of baseband
ıdulation techniques.
, and receiver optim-
mportant subject for
es usually neglected
ing—a cost-effective
e. Chapter 5 empha-
utional coding.
m trade-offs dealing
iency, and signal-to-
ıl systems. It covers
:hronization; bit syn-
zation; and some fun-

explores techniques
efficiently. Chapter
tion in such areas as
:chnology is particu-
he subject of source
ı Chapter 2; the main
:e coding additionally
ıg source coding im-
:en treated in a later
he fundamental pro-
ı in the book, before
ıg. Chapter 12 covers

some basic encryption/decryption ideas. It includes some classical encryption concepts, as well as some of the proposals for a class of encryption systems called public key cryptosystems.

If the book is used for a two-term course, a simple partitioning is suggested: the first six chapters to be taught in the first term, and the last six chapters in the second term. If the book is used for a one-term only course, it is suggested that the course material be selected from the following chapters: 1, 2, 3, 4, 5, 6, 8, and 10.

ACKNOWLEDGMENTS

This book is an outgrowth of my teaching activities at the University of California, Los Angeles, and my work in the Communications Division at The Aerospace Corporation. A number of people have contributed in many ways and it is a pleasure to acknowledge them. Dr. Maurice King, my colleague at Aerospace, carefully reviewed and made important contributions to each chapter. His continual assistance has been invaluable. He also contributed Chapter 8, Synchronization. Professor Fred Harris of San Diego State University suggested many improvements and contributed Chapter 11, Source Coding. I want to pay special thanks to Dr. Marvin Simon of the Jet Propulsion Laboratory for providing me with much encouragement and many valuable suggestions.

I also want to thank Professor Jim Omura of UCLA for sharing with me his considerable knowledge of encryption and thereby helping me improve Chapter 12. Professor Raymond Pickholtz of George Washington University gave me lots of beneficial advice throughout the writing process. Professors William Lindsey and Andreas Polydoros of the University of Southern California suggested important improvements. Professor James Modestino of Rensselaer Polytechnic Institute, Dr. Adam Lender of Lockheed Palo Alto Research Laboratory, and Professor Ron Iltis of the University of California, Santa Barbara, each provided valuable reviews. Dr. Todd Citron of Hughes Aircraft, Dr. Joe Odenwalder of MA/COM Linkabit, and Dr. Unjeng Cheng of Axiomatics were extremely helpful in the chapters on channel coding. Mr. Don Martin and Mr. Ned Feldman of The Aerospace Corporation made numerous suggestions and contributions. I also want to pay special thanks to Professor Wayne Stark of the University of Michigan, whose unique critical talents enhanced the manuscript's continuity.

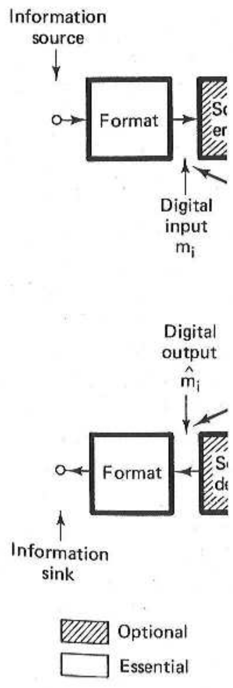
The block diagrams in Figures 1.2 and 1.3, at each chapter opening, and on the cover of the book, first appeared in the two part paper: © 1983 IEEE; B. Sklar, "A Structured Overview of Digital Communications—A Tutorial Review," *IEEE Communications Magazine*, August and October, 1983. Permission from IEEE to reprint these figures throughout the book is gratefully acknowledged.

My students at UCLA and those at Aerospace used early versions of chapters of this book and made many helpful contributions. I am indebted to all those students who have taken my courses and thus helped me with this project. I also want to express my appreciation to my management at Aerospace, Mr. Hal

McDonnell and Mr. Fred Jones, for their indulgence and moral support. I want to acknowledge and thank Ms. Cynthia Dickson for her diligence and speed in typing the entire manuscript.

Finally, I want to thank my wife, Gwen, for her very unselfish support, her understanding, and her endurance of the many months I had time for only *one* devotion—the writing of this book.

BERNARD SKLAR
Tarzana, California



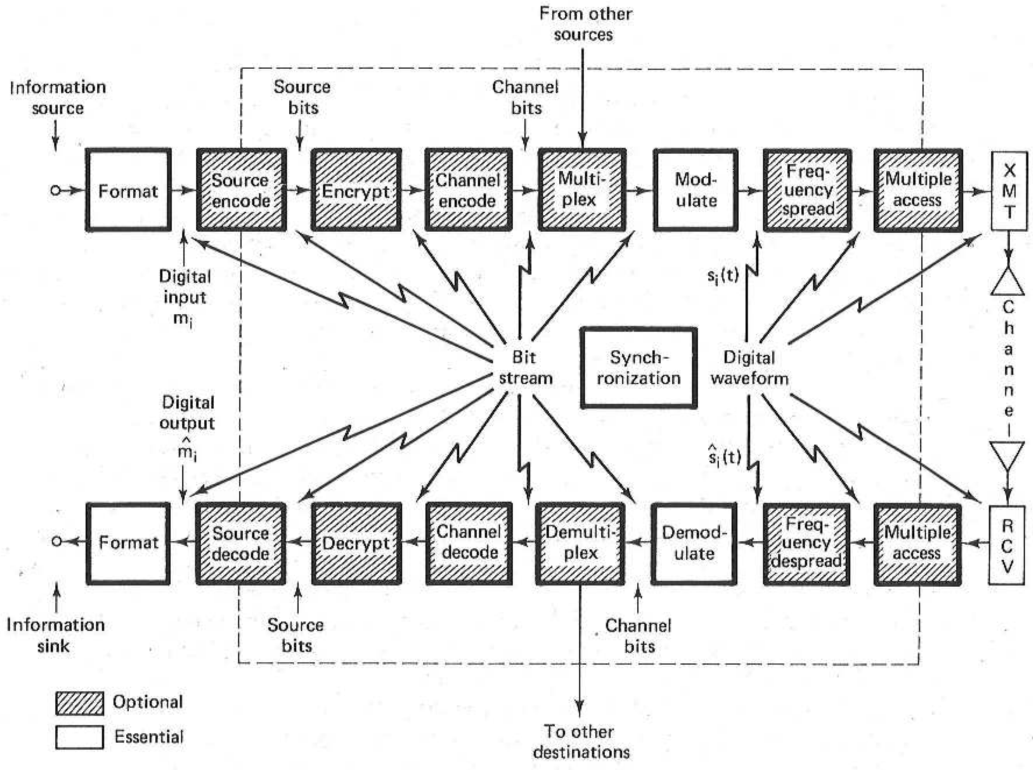
ıoral support. I want
ligence and speed in

ınselfish support, her
ad time for only *one*

BERNARD SKLAR
Tarzana, California

CHAPTER 1

Signals and Spectra



This book presents the ideas and techniques fundamental to digital communication systems. Emphasis is placed on system design goals and on the need for tradeoffs among basic system parameters such as signal-to-noise ratio (SNR), probability of error, and bandwidth expenditure. Transmission bandwidth is a finite resource; there is a growing awareness that bandwidth must be conserved, shared, and used efficiently. In general, we shall see that system performance can often be improved through the use of increased transmission bandwidth. However, such an increase is not always possible, because of physical limitations or the constraint of government regulations concerning the allocation and conservation of the usable electromagnetic spectrum.

We shall deal with the transmission of information (voice, video, or data) over a path (channel) that may consist of wires, waveguides, or free space. Frequently, the treatment will be in the context of a satellite communications link. Communication via satellites has two unique characteristics: (1) the ability to cover the globe with a flexibility that cannot be duplicated with terrestrial links, and (2) the availability of bandwidth exceeding anything previously available for intercontinental communications. Until recently, most satellite communication systems have been analog in nature. However, digital communication is becoming increasingly attractive because of the ever-growing demand for data communication and because digital transmission offers data processing options and flexibilities not available with analog transmission.

The principal feature of a digital communication system (DCS) is that during a finite interval of time, it sends a waveform from a finite set of possible waveforms, in contrast to an analog communication system, which sends a waveform

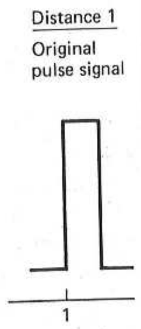
from an infi
In a DCS, th
with precisi
waveform fr
important m
(P_E).

1.1 DIGITAL COMI

1.1.1 Why I

Why are cor
There are m
signals, com
binary digit
form is affec
have some n
and (2) unw
waveform. I
tion of line
pulse can st
by the trans
its original i
perform this
regenerative

Digital
circuits. Sin
off, to be m
operating p
signal regen



ligital communication
n the need for trade-
: ratio (SNR), proba-
bandwidth is a finite
e conserved, shared,
erformance can often
vidth. However, such
ions or the constraint
nservation of the us-

oice, video, or data)
s, or free space. Fre-
communications link.
cs: (1) the ability to
with terrestrial links,
eviously available for
ellite communication
unication is becoming
d for data communi-
ing options and flex-

n (DCS) is that during
set of possible wave-
ch sends a waveform

from an infinite variety of waveform shapes with theoretically infinite resolution. In a DCS, the objective at the receiver is *not* to reproduce a transmitted waveform with precision; it is, instead, to determine from a noise-perturbed signal which waveform from the finite set of waveforms had been sent by the transmitter. An important measure of system performance in a DCS is the probability of error (P_E).

1.1 DIGITAL COMMUNICATION SIGNAL PROCESSING

1.1.1 Why Digital?

Why are communication systems, military and commercial alike, "going digital"? There are many reasons. The primary advantage is the ease with which digital signals, compared to analog signals, are regenerated. Figure 1.1 illustrates an ideal binary digital pulse propagating along a transmission line. The shape of the waveform is affected by two basic mechanisms: (1) as all transmission lines and circuits have some nonideal transfer function, there is a distorting effect on the ideal pulse; and (2) unwanted electrical noise or other interference further distorts the pulse waveform. Both of these mechanisms cause the pulse shape to degrade as a function of line length, as shown in Figure 1.1. During the time that the transmitted pulse can still be reliably identified (before it is degraded to an ambiguous state by the transmission line), the pulse is amplified by a digital amplifier that recovers its original ideal shape. The pulse is thus "reborn" or regenerated. Circuits that perform this function at regular intervals along a transmission system are called *regenerative repeaters*.

Digital circuits are less subject to distortion and interference than are analog circuits. Since binary digital circuits operate in one of two states, fully on or fully off, to be meaningful a disturbance must be large enough to change the circuit operating point from one state to the other. Such two-state operation facilitates signal regeneration and thus prevents noise and other disturbances from accu-

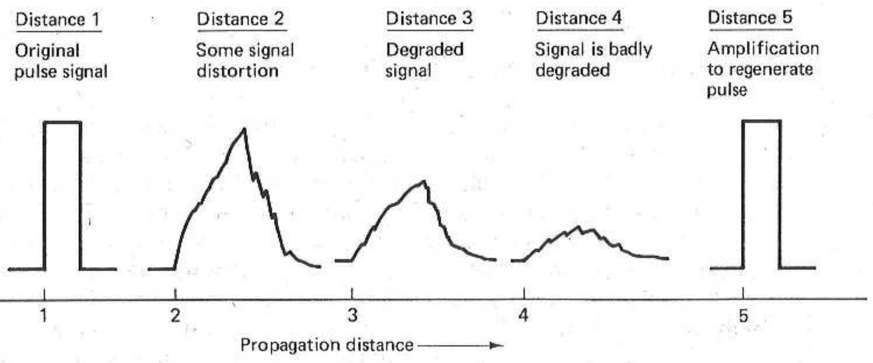


Figure 1.1 Pulse degradation and regeneration.

mulating in transmission. Analog signals, however, are *not* two-state signals; they can take an *infinite variety* of shapes. With analog circuits, even a small disturbance can render the reproduced waveform unacceptably distorted. Once the analog signal is distorted, the distortion cannot be removed by amplification. Since, with analog signals, accumulated noise is irrevocably bound to the signal, analog signals cannot be completely regenerated. Extremely low error rates producing high signal fidelity are possible through error detection and correction with digital techniques, but similar procedures are not available with analog.

There are other important advantages to digital communications. Digital circuits are *more reliable* and can be produced at lower cost than analog circuits. Also, digital hardware lends itself to *more flexible* implementation than analog hardware [e.g., microprocessors, digital switching, and large-scale integrated (LSI) circuits]. The combining of digital signals using time-division multiplexing (TDM) is *simpler* than the combining of analog signals using frequency-division multiplexing (FDM). Different types of digital signals (data, telegraph, telephone, television) can be treated as identical signals in transmission and switching—*a bit is a bit*. Also, for convenient switching, digital messages can be handled in autonomous groups called *packets*. Digital techniques lend themselves naturally to signal processing functions that protect against interference and jamming, or that provide encryption and privacy; such techniques are discussed in Chapters 10 and 12, respectively. Also, much data communication is computer to computer, or digital instrument or terminal to computer. Such digital terminations are naturally best served by digital communication links.

Most system choices entail trade-offs; system options are rarely all good or all bad. Thus far we have discussed only the *benefits* of digital transmission. What do you suppose are the *costs* or *liabilities*? A major disadvantage of digital transmission is that it typically requires a *greater system bandwidth* to communicate the same information in a digital format as compared to an analog format. Throughout this book we emphasize that bandwidth is a valuable resource, not always available. Bandwidth-efficient signaling techniques are discussed in Chapters 2 and 7. Another cost of digital transmission is that digital detection requires system synchronization (Chapter 8), whereas analog signals generally have no such requirement.

1.1.2 Typical Block Diagram and Transformations

The functional block diagram shown in Figure 1.2 illustrates the signal flow through a typical DCS. The upper blocks—format, source encode, encrypt, channel encode, multiplex, modulate, frequency spread, and multiple access—indicate the signal transformations from the source to the transmitter. The lower blocks indicate the signal transformations from the receiver to the sink; the lower blocks essentially reverse the signal processing steps performed by the upper blocks. It used to be that the only blocks within the dashed lines were the *modulator* and *demodulator*, together called a *modem*. During the past two decades, other signal processing functions were frequently incorporated within the same assembly as the modulator and demodulator. Consequently, at present, the term "modem"

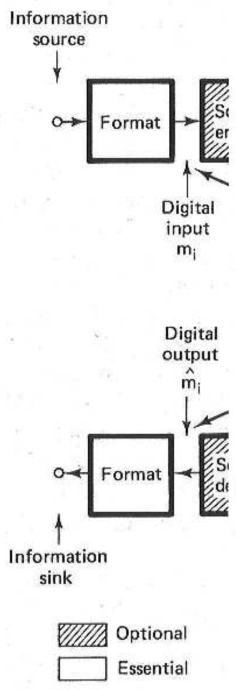


Figure 1.2
mission fr
mun. Mag

often encom
1.2; when th
system. Not
the blocks ha
or outside tl
"muscles" (
conversion s
usually consi
stage, typica
Of all t
ulation are e
design optio
formation int
processing w
which the sy
mission char
The sou

vo-state signals; they
even a small disturb-
torted. Once the an-
amplification. Since,
to the signal, analog
error rates producing
correction with digital
nalog.
munications. Digital
than analog circuits.
mentation than analog
large-scale integrated
-division multiplexing
ng frequency-division
telegraph, telephone,
ion and switching—*a*
ges can be handled in
l themselves naturally
ence and jamming, or
discussed in Chapters
computer to computer,
l terminations are nat-

s are rarely all good or
tal transmission. What
vantage of digital trans-
lwidth to communicate
nalog format. Through-
e resource, not always
iscussed in Chapters 2
tection requires system
enerally have no such

strates the signal flow
encode, encrypt, chan-
ultiple access—indicate
itter. The lower blocks
e sink; the lower blocks
by the upper blocks. It
were the *modulator* and
wo decades, other signal
n the same assembly as
ent, the term "modem"

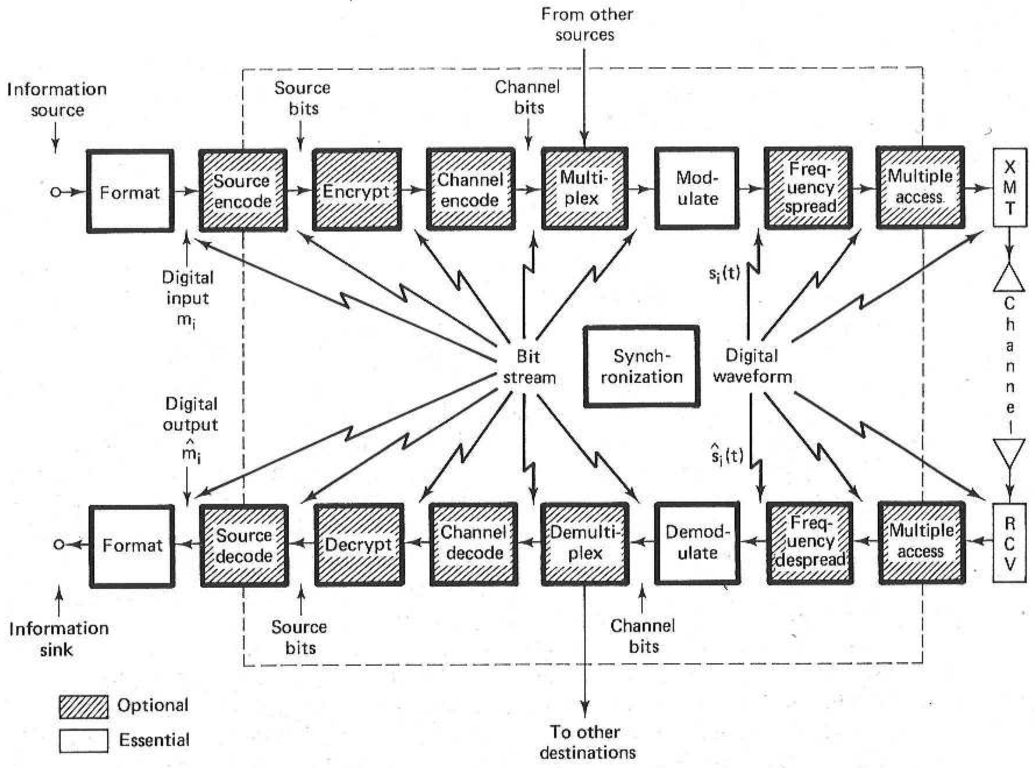


Figure 1.2 Block diagram of a typical digital communication system. (Reprinted with permission from B. Sklar, "A Structured Overview of Digital Communications," *IEEE Commun. Mag.*, August 1983, Fig. 1, p. 5. © 1983 IEEE.)

often encompasses all the processing steps shown within the dashed lines of Figure 1.2; when this is the case, the modem can be thought of as the "brains" of the system. Note that what constitutes a modem is not a precise concept; some of the blocks have purposely been shown *on* the dashed line rather than either inside or outside the modem. The transmitter and receiver can be thought of as the "muscles" of the system. The transmitter usually consists of a frequency up-conversion stage, a high-power amplifier, and an antenna. The receiver portion usually consists of an antenna, a low-noise amplifier (LNA), and a down-converter stage, typically to an intermediate frequency (IF).

Of all the signal processing steps, only formatting, modulation, and demodulation are essential for a DCS; the other processing steps within the modem are design options for specific system needs. *Formatting* transforms the source information into *digital symbols*; it makes the information compatible with the signal processing within a digital communication system. *Modulation* is the process by which the symbols are converted to *waveforms* that are compatible with the transmission channel.

The source encoding step produces analog-to-digital (A/D) conversion (for analog sources) *and* removes *redundant or unneeded* information. Encryption prevents unauthorized users from understanding messages and from injecting false messages into the system. Channel coding, for a given data rate, can reduce the probability of error (P_E), or reduce the signal-to-noise ratio (SNR) requirement, at the expense of bandwidth or decoder complexity. Channel coding can also reduce the system bandwidth requirement at the expense of SNR or P_E performance. Frequency spreading can produce a signal that is less vulnerable to interference (both natural and intentional) and can be used to enhance the privacy of the communicators. Multiplexing and multiple access procedures combine signals that might have different characteristics or might originate from different sources, so that they can share a portion of the communications resource.

The flow of the signal processing steps shown in Figure 1.2 represents a typical arrangement; however, the blocks are sometimes implemented in a different order. For example, multiplexing can take place prior to channel encoding, or prior to modulation, or—with a two-step modulation process (subcarrier and carrier)—it can be performed between the two modulation steps. Similarly, spreading can take place anywhere along the transmission chain; its precise location depends on the particular technique used. Figure 1.2 illustrates the reciprocal aspect of the procedure; any signal processing step that takes place in the transmitting chain must be reversed in the receiving chain. The figure also indicates that from the source to the modulator a message, also called a *baseband signal* or a *bit stream,* is characterized by a sequence of digital symbols. After modulation, the message takes the form of a digitally encoded waveform or *digital waveform.* Similarly, in the reverse direction, a received message appears as a digital waveform until it is demodulated. Thereafter it takes the form of a bit stream for all further signal processing steps. At various points along the signal route, noise corrupts the waveform $s(t)$ so that its reception must be termed an estimate $\hat{s}(t)$. Such noise and its deleterious effects on system performance are considered in Chapter 4.

Figure 1.3 shows the basic signal processing functions, which may be viewed as transformations from one signal space to another. The transformations are classified into seven basic groups:

1. Formatting and source coding
2. Modulation/demodulation
3. Channel coding
4. Multiplexing and multiple access
5. Spreading
6. Encryption
7. Synchronization

Although this organization has some inherent overlap, it provides a useful structure for the book. Beginning with Chapter 2, the seven basic transformations are considered individually. In Chapter 2 we discuss the basic formatting techniques for transforming the source information into digital symbols, as well as

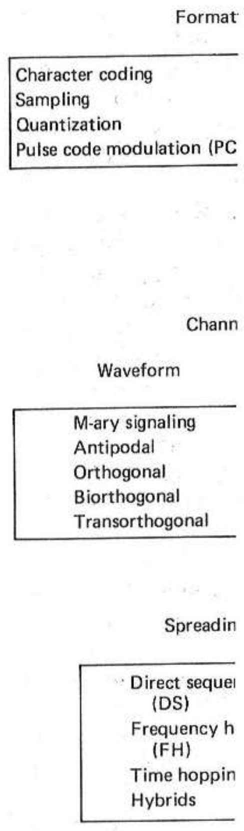


Figure 1.3
B. Sklar,
August 19

the selectior
transmission
together; th
"source cod
addition to c
11.

In Figu
basic catego
volves the d
accomplishe
all the sign
coherent; w
Both techni

ɔrmation. Encryption
nd from injecting false
ı rate, can reduce the
ɔ (SNR) requirement,
.nnel coding can also
f SNR or P_E perform-
ıs vulnerable to inter-
nhance the privacy of
dures combine signals
rom different sources,
ıource.
gure 1.2 represents a
implemented in a dif-
: to channel encoding,
ocess (subcarrier and
tion steps. Similarly,
. chain; its precise lo-
2 illustrates the recip-
hat takes place in the
. The figure also indi-
lso called a *baseband*
digital symbols. After
ːd waveform or *digital*
message appears as a
kes the form of a bit
ɪoints along the signal
ɔn must be termed an
stem performance are

which may be viewed
e transformations are

ɔ, it provides a useful
basic transformations
basic formatting tech-
ıl symbols, as well as

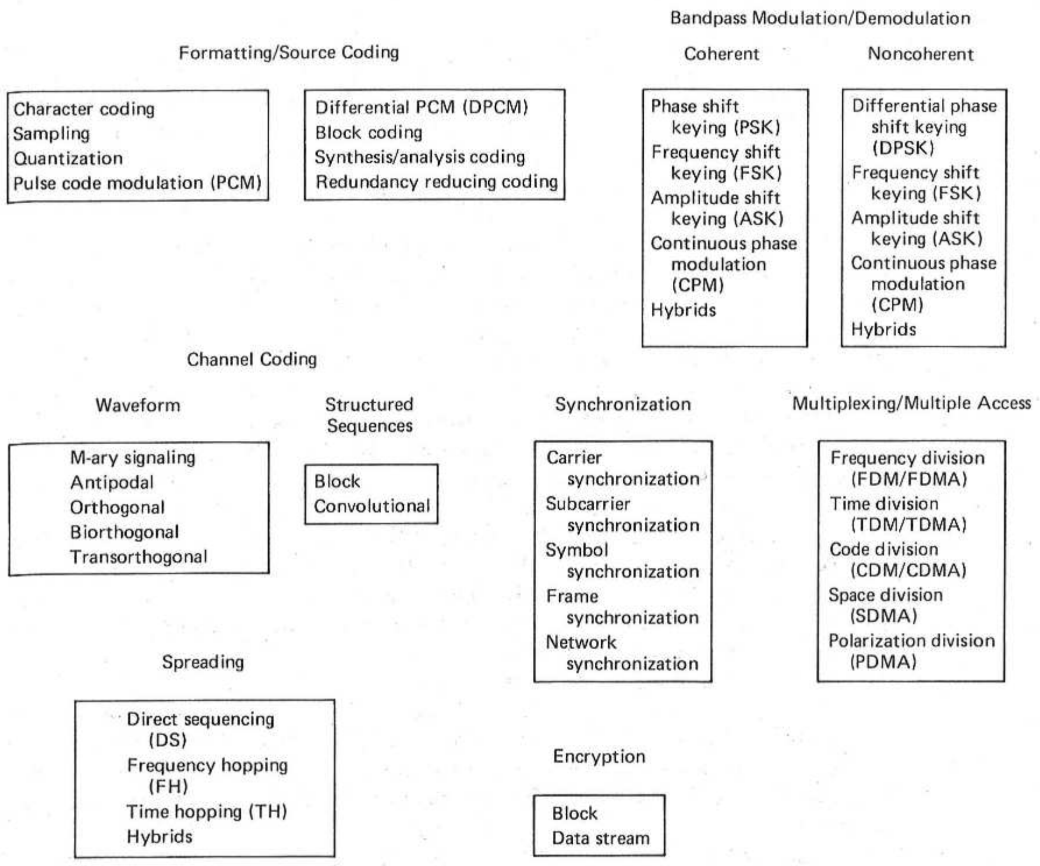


Figure 1.3 Basic digital communication transformations. (Reprinted with permission from B. Sklar, "A Structured Overview of Digital Communications," *IEEE Commun. Mag.*, August 1983, Fig. 2, p. 6. © 1983 IEEE.)

the selection of waveforms for making the symbols compatible with baseband transmission. As seen in Figure 1.3, formatting and source coding are grouped together; they are similar in that they involve data digitization. Since the term "source coding" has taken on the connotation of data redundancy reduction in addition to digitization, it is treated later, as a special formatting case, in Chapter 11.

In Figure 1.3, bandpass modulation/demodulation is partitioned into two basic categories, coherent and noncoherent. The process of *demodulation* involves the detection of the baseband information. Digital demodulation is typically accomplished with the aid of reference waveforms. When the references contain all the signal attributes, particularly phase information, the process is termed *coherent*; when phase information is not used, the process is termed *noncoherent*. Both techniques are detailed in Chapter 3.

Chapter 4 is devoted to link analysis. In the past, this area has received little attention in colleges or in textbooks, probably because it was considered straightforward and not worthy of detailed discussion. However, of the many specifications, analyses, and tabulations that support a developing communication system, link analysis stands out in its ability to provide overall system insight. In Chapter 4 we bring together all the link fundamentals that are essential for the analysis of most communication systems.

Channel coding deals with the techniques used to enhance digital signals so that they are less vulnerable to such channel impairments as noise, fading, and jamming. In Figure 1.3 channel coding is partitioned into two basic categories, waveform coding and structured sequences. *Waveform coding* involves the use of new waveforms, yielding improved detection performance over that of the original waveforms. *Structured sequences* involve the use of redundant bits to determine whether or not an error has occurred due to noise on the channel. One of these techniques, known as automatic repeat request (ARQ), simply recognizes the occurrence of an error and requests that the sender retransmit the message; other techniques, called forward error correction (FEC), are capable of automatically correcting the errors (within specified limitations). Under the heading of structured sequences, we shall discuss the two prevalent techniques, block coding and convolutional coding. In Chapter 5 we consider waveform coding and linear block coding. In Chapter 6 we consider convolutional coding, Viterbi decoding (and other decoding algorithms), hard versus soft decoding procedures, and interleaving and deinterleaving.

In Chapter 7 we summarize the design goals for a communication system and present various modulation and coding trade-offs that need to be considered in the design of a system. We discuss theoretical limitations such as the Nyquist criterion and the Shannon limit. We also examine bandwidth-efficient modulation schemes.

Chapter 8 deals with synchronization. In digital communications, synchronization involves the estimation of both time and frequency. The subject is partitioned as shown in Figure 1.3. Coherent systems need to synchronize their frequency reference with the carrier (and possibly subcarrier) in both frequency and phase. For noncoherent systems, phase synchronization is not needed. The fundamental time-synchronization process is symbol synchronization. The demodulator needs to know when to start and end the symbol detection procedure; a timing error will degrade detection performance. The next time-synchronization level, frame synchronization, allows the reconstruction of the message; and network synchronization allows coordination with other users in order to use the resource efficiently. In Chapter 8 we are concerned with the alignment of the timing of spatially separated periodic processes; the alignment is illustrated for the case of a satellite communications link.

Chapter 9 deals with multiplexing and multiple access. The two terms mean very similar things. Both involve the idea of resource sharing. The main difference between the two is that *multiplexing* takes place locally (e.g., on a printed circuit board, within an assembly, or even within a facility), and *multiple access* takes place remotely (e.g., multiple users share the use of a satellite transponder). Mul-

tiplexing i
into the sy
may requir
the classic
code divisi
emerged a

Chap
communic
techniques
privacy, o

Chap
ing efficien
digital dat
reduce dat
of the syst

The f
cryption,
preventing
from the cl
injecting s
the data e
cryption s

1.1.3 Basi

Some of t
communic

Infor
by m
outpu
ampli
value
digita
quant
are d

Textu
trans
finite

Char
Char
stand
Stand
Code
and N

ıually likely signaling and

frequency $f_m = 4000$ Hz,
antization distortion must

bits per PCM word that

ıt is the resulting bit rate?

:ded for a 10-Mbits/s signal

'able system bandwidth is

quantization distortion ≤
ling rate of 8000 samples/s
the theoretical minimum

ıl waveform that is trans-
:haracteristic. The system

. 8000 samples/s. We may
ıvert them into codewords

he detection of PAM with
.?
the minimum bandwidth
les are quantized to eight

ł to a peak-to-peak swing
ıre quantized to 64 evenly
ratio of peak signal power
ınsmitted either as binary
.ndwidth is defined by the

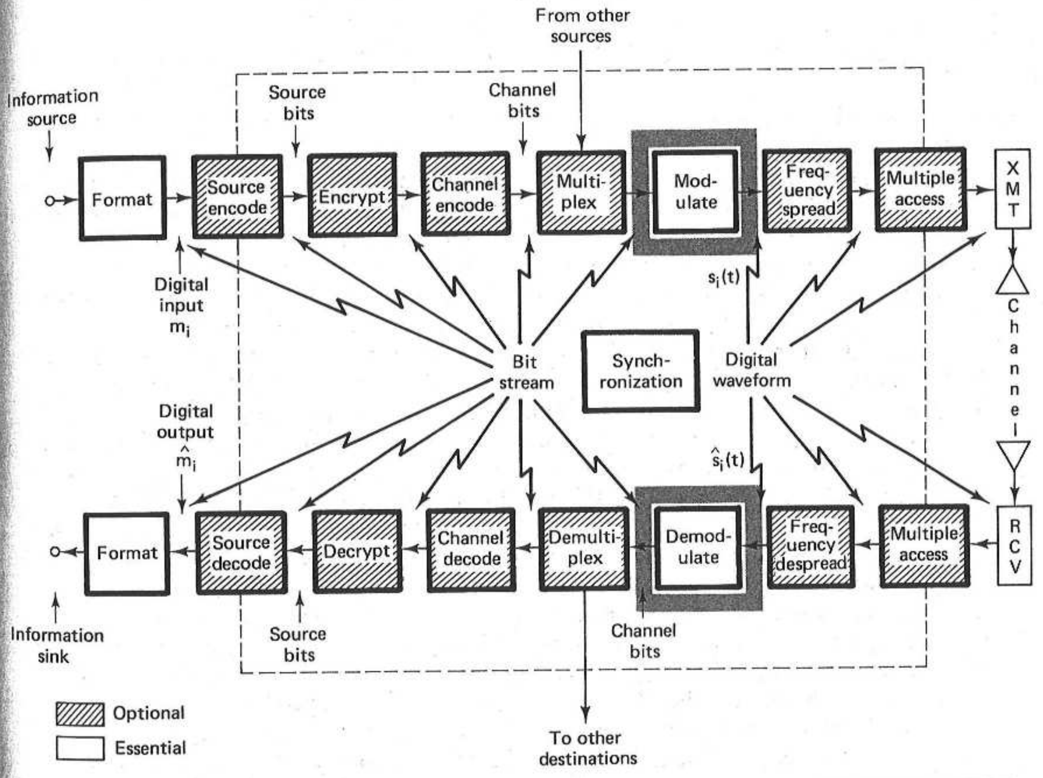
ıal and transmitted over a
ıantization levels are used
e raised cosine type with

this system without intro-

ommodated for the analog

..

Bandpass Modulation and Demodulation



3.1 WHY MODULATE?

Digital modulation is the process by which digital symbols are transformed into waveforms that are compatible with the characteristics of the channel. In the case of baseband modulation, these waveforms are pulses, but in the case of *bandpass modulation* the desired information signal modulates a sinusoid called a *carrier wave,* or simply a *carrier*; for radio transmission the carrier is converted to an electromagnetic (EM) field for propagation to the desired destination. One might ask why it is necessary to use a carrier for the radio transmission of baseband signals. The answer is as follows. The transmission of EM fields through space is accomplished with the use of antennas. To efficiently couple the transmitted EM energy into space, the dimensions of the antenna aperture should be at least as large as the wavelength being transmitted. Wavelength, λ, is equal to c/f, where c, the speed of light, is 3×10^8 m/s. For a baseband signal with frequency $f = 3000$ Hz, $\lambda = 10^5$ m $\simeq$ 60 miles. To efficiently transmit a 3000-Hz signal through space *without carrier-wave modulation*, an antenna that spans at least 60 miles would be required. Even if we were willing to inefficiently transmit the EM energy with an antenna measuring one-tenth of a wavelength, we are faced with an impossible antenna size. However, if the information to be transmitted is first modulated on a higher frequency carrier, for example a 30-GHz carrier, the equivalent antenna diameter is then less than $\frac{1}{2}$ in. For this reason, carrier-wave or bandpass modulation is an essential step for all systems involving radio transmission.

Bandpass modulation can provide other important benefits in signal transmission. If more than one signal utilizes a single channel, modulation may be used

to separate th
multiplexing, i
effects of inte
spectrum mod
bandwidth tha
for interferenc
used to place
filtering and a
quency (RF) si

3.2 SIGNALS AND N

3.2.1 Noise i

The task of tl
received wave
to which the s
signal distortic
receiver discu
function cause
The secc
variety of sou
unwanted sign
motion of elec
in amplifiers a
the received si
noise, $n(t)$. Tl
mechanics anc
The prir
plitudes are di
in Section 1.5
$p(n)$, of the ze

where σ^2 is th
noise amplitu
noise can be i
The prir
power spectra
radio commur
just as much
fluctuations—

s are transformed into
he channel. In the case
n the case of *bandpass*
ısoid called a *carrier*
ier is converted to an
lestination. One might
ısmission of baseband
Л fields through space
couple the transmitted
ture should be at least
., is equal to c/f, where
al with frequency f =
000-Hz signal through
spans at least 60 miles
ransmit the EM energy
are faced with an im-
ansmitted is first mod-
carrier, the equivalent
rier-wave or bandpass
ıdio transmission.
enefits in signal trans-
odulation may be used

to separate the different signals. Such a technique, known as *frequency-division multiplexing,* is discussed in Chapter 9. Modulation can be used to minimize the effects of interference. A class of such modulation schemes, known as *spread-spectrum modulation*, requires a system bandwidth much larger than the minimum bandwidth that would be required by the message. The trade-off of bandwidth for interference rejection is considered in Chapter 10. Modulation can also be used to place a signal in a frequency band where design requirements, such as filtering and amplification, can be easily met. This is the case when radio-frequency (RF) signals are converted to an intermediate frequency (IF) in a receiver.

3.2 SIGNALS AND NOISE

3.2.1 Noise in Radio Communication Systems

The task of the demodulator or detector is to retrieve the bit stream from the received waveform, as nearly error free as possible, notwithstanding the distortion to which the signal may have been subjected. There are two primary causes for signal distortion. The first is the filtering effects of the transmitter, channel, and receiver discussed in Section 2.11. As described there, a nonideal system transfer function causes symbol "smearing," which can produce *intersymbol interference*.

The second cause for signal distortion is the noise that is produced by a variety of sources, such as galaxy noise, terrestrial noise, amplifier noise, and unwanted signals from other sources. An unavoidable cause of noise is the thermal motion of electrons in any conducting media. This motion produces *thermal noise* in amplifiers and circuits which corrupts the signal in an additive fashion; that is, the received signal, $r(t)$, is the sum of the transmitted signal, $s(t)$, and the thermal noise, $n(t)$. The statistics of thermal noise have been developed using quantum mechanics and are well known [1].

The primary statistical characteristic of thermal noise is that the noise amplitudes are distributed according to a normal or Gaussian distribution, discussed in Section 1.5.5 and shown in Figure 1.7. The probability density function (pdf), $p(n)$, of the zero-mean noise voltage is expressed as

$$p(n) = \frac{1}{\sigma\sqrt{2\pi}} \exp\left[-\frac{1}{2}\left(\frac{n}{\sigma}\right)^2\right] \tag{3.1}$$

where σ^2 is the noise variance. In Figure 1.7 it can be seen that the most probable noise amplitudes are those with small positive or negative values. In theory, the noise can be infinitely large, but very large noise amplitudes are rare.

The primary spectral characteristic of thermal noise is that its two-sided power spectral density, $G_n(f) = N_0/2$, is flat for all frequencies of interest for radio communication systems. In other words, thermal noise, on the average, has just as much power per hertz in low-frequency fluctuations as in high-frequency fluctuations—up to a frequency of about 10^{12} hertz. When the noise power is message block before transmitting the next in the sequence. The second ARQ procedure, called *continuous ARQ with pullback*, is shown in Figure 5.10b. Here a full-duplex connection is necessary. Both terminals are transmitting simultaneously; the transmitter is sending message data and the receiver is sending acknowledgment data. Notice that a sequence number has to be assigned to each block of data. Also, the ACKs and NAKs need to reference such numbers, or else there needs to be a priori knowledge of the propagation delays so that the transmitter knows which messages are associated with which acknowledgments. In the example of Figure 5.10b there is a fixed separation of four blocks between the message being transmitted and the acknowledgment being simultaneously received. For example, when message 8 is being sent, a NAK corresponding to the corrupted message 4 is being received. In this ARQ procedure, the transmitter "pulls back" to the message in error and retransmits all message data, starting with the corrupted message. The final method, called *continuous ARQ with selective repeat*, is shown in Figure 5.10c. Here, as with the second ARQ procedure, a full-duplex connection is needed. However, in this procedure, only the corrupted message is repeated; then the transmitter continues the transmission sequence where it had left off instead of repeating any subsequent correctly received messages.

The choice of which ARQ procedure to choose is a trade-off between the requirements for efficient utilization of the communications resource and the need to provide full-duplex connectivity. The half-duplex connectivity required in Figure 5.10a is less costly than full-duplex; the associated inefficiency can be measured by the blank time slots. The more efficient utilization illustrated in Figures 5.10b and c requires the more costly full-duplex connectivity.

The major advantage of ARQ over forward error correction (FEC) is that error detection requires much simpler decoding equipment and much less redundancy than does error correction. Also, ARQ is adaptive in the sense that information is retransmitted only when errors occur. On the other hand, FEC may be desirable in place of, or in addition to, error detection, for any of the following reasons:

1. A reverse channel is not available or the delay with ARQ would be excessive.
2. The retransmission strategy is not conveniently implemented.
3. The expected number of errors, without corrections, would require excessive retransmissions.

5.3 STRUCTURED SEQUENCES

In Section 3.8 we considered digital signaling by means of $M = 2^k$ signal waveforms (M-ary signaling), where each waveform contains k bits of information. We saw that in the case of orthogonal M-ary signaling, we can decrease P_B by increasing M (expanding the bandwidth). Similarly, in Section 5.1 we showed that it is possible to decrease P_B by encoding k binary digits into one of M orthogonal codewords. The major disadvantage with such orthogonal coding techniques is the associated inefficient use of bandwidth. The required transmission bandwidth grows exponentially with k for an orthogonal set of $M = 2^k$ waveforms. In this and subsequent sections we abandon the need for antipodal or orthogonal properties and focus on a class of encoding procedures known as *parity-check codes*. Such channel coding procedures are classified as *structured sequences* because they represent methods of inserting structured redundancy into the source data so that the presence of errors can be detected or the errors corrected. Structured sequences are partitioned into two important subcategories as shown in Figure 5.1: *block coding* and *convolutional coding*. Block coding (primarily) is treated in this chapter, and convolutional coding is treated in Chapter 6. These techniques allow us to attain a P_B performance comparable to waveform encoding techniques but with lower bandwidth requirements. The codewords of these codes (structured sequences) are usually *nonorthogonal* [3].

5.3.1 Channel Models

5.3.1.1 Discrete Memoryless Channel

A *discrete memoryless channel* (DMC) is characterized by a discrete input alphabet, a discrete output alphabet, and a set of conditional probabilities, $P(j|i)$ $(1 \le i \le M, 1 \le j \le Q)$, where i represents a modulator M-ary input symbol, j represents a demodulator Q-ary output symbol, and $P(j|i)$ is the probability of receiving j given that i was transmitted. Each output symbol of the channel depends only on the corresponding input, so that for a given input sequence $\mathbf{U} = u_1, u_2, \ldots, u_m, \ldots, u_N$ the conditional probability of a corresponding output sequence $\mathbf{Z} = z_1, z_2, \ldots, z_m, \ldots, z_N$ may be expressed as

$$P(\mathbf{Z}|\mathbf{U}) = \prod_{m=1}^{N} P(z_m|u_m) \tag{5.12}$$

In the event that the channel *has memory* (i.e., noise or fading that occurs in bursts), the conditional probability of the sequence $\mathbf{Z}$ would need to be expressed as the *joint* probability of all the elements of the sequence. Equation (5.12) expresses the *memoryless* condition of the channel. Since the channel noise in a memoryless channel is defined to affect each symbol independently of all the other symbols, the conditional probability of $\mathbf{Z}$ is seen as the product of the independent element probabilities.

5.3.1.2 Binary Symmetric Channel

A *binary symmetric channel* (BSC) is a special case of a DMC; the input and output alphabet sets consist of the binary elements (0 and 1). The conditional probabilities are symmetric:

$$\begin{aligned} P(0|1) &= P(1|0) = p \\ P(1|1) &= P(0|0) = 1 - p \end{aligned} \tag{5.13}$$

Equation (5.13) states the channel *transition probabilities*. That is, given that a channel symbol was transmitted, the probability that it is received in error is p (related to the symbol energy), and the probability that it is received correctly is $(1 - p)$. Since the demodulator output consists of the discrete elements 0 and 1, the demodulator is said to make a firm or *hard decision* on each symbol. A commonly used code system consists of BPSK modulated coded data, hard decision demodulated. Then the channel symbol error probability is found using the methods discussed in Section 3.7.1 and Equation (3.84) to be

$$p = Q\left(\sqrt{\frac{2E_c}{N_0}}\right)$$

where E_c/N_0 is the channel symbol energy per noise density, and $Q(x)$ is defined in Equation (2.42).

When such hard decisions are used in a binary coded system, the demodulator feeds the two-valued *code symbols* or *channel bits* to the decoder. Since the decoder then operates on the hard decisions made by the demodulator, decoding with a BSC channel is called *hard-decision decoding*.

5.3.1.3 Gaussian Channel

We can generalize our definition of the DMC to channels with alphabets that are not discrete. An example is the *Gaussian channel* with a discrete input alphabet and a continuous output alphabet over the range $(-\infty, \infty)$. The channel adds noise to the symbols. Since the noise is a Gaussian random variable, with zero mean and variance σ^2, the resulting probability density function (pdf) of the received random variable z, conditioned on the symbol u_k (the likelihood of u_k), can be written

$$p(z|u_k) = \frac{1}{\sigma\sqrt{2\pi}} \exp\left[\frac{-(z - u_k)^2}{2\sigma^2}\right] \tag{5.14}$$

for all z, where $k = 1, 2, \ldots, M$. For this case, *memoryless* has the same meaning as it does in Section 5.3.1.1, and Equation (5.12) can be used to obtain the conditional probability for the sequence, **Z**.

When the demodulator output consists of a continuous alphabet or its quantized approximation (with greater than two quantization levels), the demodulator is said to make *soft decisions*. In the case of a coded system, the demodulator feeds such quantized code symbols to the decoder. Since the decoder then operates on the soft decisions made by the demodulator, decoding with a Gaussian channel is called *soft-decision decoding*.

In the case of a hard-decision channel, we are able to characterize the detection process with a channel symbol error probability. However, in the case of a soft-decision channel, the detector makes the kind of decisions (soft decisions) that cannot be labeled as correct or incorrect. Thus, since there are no firm decisions, there cannot be a probability of making an error; the detector can only

That is, given that a
received in error is p
s received correctly is
rete elements 0 and 1,
ı on each symbol. A
coded data, hard de-
ility is found using the
be

:y, and $Q(x)$ is defined

ed system, the demo-
to the decoder. Since
the demodulator, de-
ıg.

els with alphabets that
th a discrete input al-
$(-\infty, \infty)$. The channel
random variable, with
y function (pdf) of the
(the likelihood of u_k),

(5.14)

has the same meaning
sed to obtain the con-

s alphabet or its quan-
vels), the demodulator
tem, the demodulator
the decoder then op-
oding with a Gaussian

o characterize the de-
owever, in the case of
:isions (soft decisions)
: there are no firm de-
the detector can only

formulate a family of conditional probabilities or likelihoods of the different symbol types.

It is possible to design decoders using soft decisions, but block code soft-decision decorders are substantially more complex than hard-decision decoders; therefore, block codes are usually implemented with hard-decision decoders. For convolutional codes, both hard- and soft-decision implementations are equally popular. In this chapter we consider that the channel is a binary symmetric channel (BSC), and hence the decoder employs hard decisions. In Chapter 6 we further discuss channel models, as well as hard- versus soft-decision decoding for convolutional codes.

5.3.2 Code Rate and Redundancy

In the case of block codes, the source data are segmented into blocks of k data bits, also called information bits or message bits; each block can represent any one of 2^k distinct messages. The encoder transforms each k-bit data block into a larger block of n bits, called code bits or channel symbols. The $(n - k)$ bits, which the encoder adds to each data block, are called *redundant bits*, *parity bits*, or *check bits*; they carry no new information. The code is referred to as an (n, k) code. The ratio of redundant bits to data bits, $(n - k)/k$, within a block is called the *redundancy* of the code, and the ratio of data bits to total bits, k/n, is called the *code rate*. The code rate can be thought of as the portion of a code bit that constitutes information. For example, in a rate $\frac{1}{2}$ code, each code bit carries $\frac{1}{2}$ bit of information.

In this chapter and Chapter 6 we consider those coding techniques that provide redundancy by increasing the required transmission bandwidth. For example, an error control technique that employs a rate 1/2 code (100% redundancy) will require double the bandwidth of an uncoded system. However, if a rate 3/4 code is used, the redundancy is 33% and the bandwidth expansion is only 4/3. In Chapter 7 we consider modulation/coding techniques for bandlimited channels where complexity, instead of bandwidth, is traded for error performance improvement.

5.3.3 Parity-Check Codes

5.3.3.1 Single-Parity-Check Code

Parity-check codes use linear sums of the information bits, called *parity symbols* or *parity bits*, for error detection or correction. A single-parity check code is constructed by adding a single-parity bit to a block of data bits. The parity bit takes on the value of one or zero as needed to ensure that the summation of all the bits in the codeword yields an even (or odd) result. The summation operation is performed using modulo-2 arithmetic (exclusive-or logic), as described in Section 2.12.3. If the added parity is designed to yield an even result, the method is termed *even parity*, and if designed to yield an odd result, it is termed *odd*

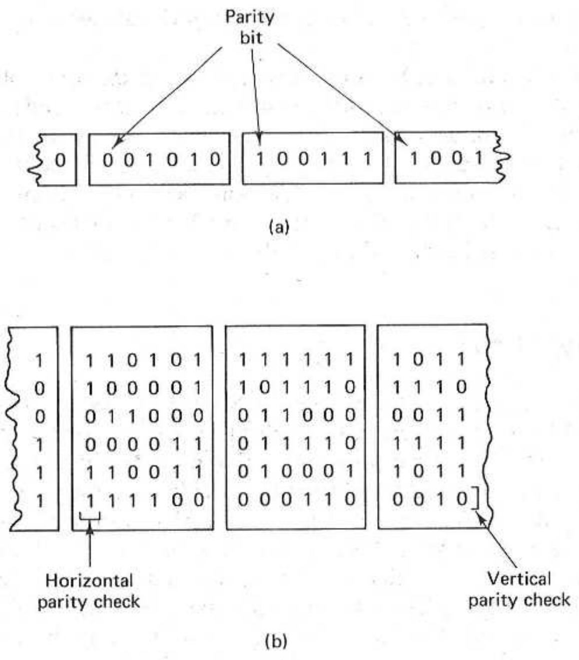


Figure 5.11 Parity checks for serial and parallel transmission. (a) Serial transmission. (b) Parallel transmission.

parity. Figure 5.11a illustrates a serial data transmission (the rightmost bit is the earliest bit). A single-parity bit is added (the leftmost bit in each block) to yield even parity.

At the receiving terminal, the decoding procedure consists of testing that the modulo-2 sum of the codeword bits yields a zero result (even parity). If the result is found to be one instead of zero, the codeword is known to contain errors. The rate of the code can be expressed as $k/(k + 1)$. Do you suppose the decoder can automatically *correct* a digit that is received in error? No, it cannot. It can only *detect* the presence of an odd number of bit errors (if an even number of bits are inverted, the parity test will appear correct; this represents the case of an *undetected error*). Assuming that all bit errors are equally likely and occur independently, we can write the probability of j errors occurring in a block of n symbols as

$$P(j, n) = \binom{n}{j} p^j (1 - p)^{n-j} \tag{5.15}$$

where p is the probability that a *channel symbol* is received in error, and where

$$\binom{n}{j} = \frac{n!}{j!(n - j)!}$$

is the number of various ways in which j bits out of n may be in error. Thus for

a single-parity erro
within a block of

Example 5.1 Even

Configure a (
pears as the
detect? Comp
symbol error
error is p =

Solution

The code is
of an unde
where in a

5.3.3.2 I

A *rectan*
data transmis
bits comprise
pended to ea
resulting in a
the rectangul

Parity checks for serial
ransmission. (a) Serial
(b) Parallel transmission.

he rightmost bit is the
n each block) to yield

onsists of testing that
lt (even parity). If the
iown to contain errors.
u suppose the decoder
? No, it cannot. It can
(if an even number of
represents the case of
ually likely and occur
curring in a block of n

(5.15)

ed in error, and where

y be in error. Thus for

a single-parity error-detection code, the probability of an undetected error, P_{nd}, within a block of n bits is computed, as follows:

$$P_{\text{nd}} = \sum_{j=1}^{\substack{n/2 \text{ (for } n \text{ even)} \\ (n-1)/2 \text{ (for } n \text{ odd)}}} \binom{n}{2j} p^{2j}(1-p)^{n-2j} \tag{5.16}$$

Example 5.1 Even-Parity Code

Configure a (4, 3) even-parity error-detection code such that the parity symbol appears as the leftmost symbol of the codeword. Which error patterns can the code detect? Compute the probability of an undetected message error, assuming that all symbol errors are independent events and that the probability of a channel symbol error is $p = 10^{-3}$.

Solution

Message	Parity	Codeword
000	0	0 000
100	1	1 100
010	1	1 010
110	0	0 110
001	1	1 001
101	0	0 101
011	0	0 011
111	1	1 111

parity message

The code is capable of detecting all single- and triple-error patterns. The probability of an undetected error is equal to the probability that two or four errors occur anywhere in a codeword.

$5{,}98 \times 10^{-6}$

$(1-p)(1-p)$

$1-p-p+p^2$

$6p^2\,(p^2-2p+1)$

$6p^4-12p^3+6p^2$

$$\begin{aligned} P_{\text{nd}} &= \binom{4}{2}p^2(1-p)^2 + \binom{4}{4}p^4 \\ &= 6p^2(1-p)^2 + p^4 \\ &= 6p^2 - 12p^3 + 7p^4 \\ &= 6(10^{-3})^2 - 12(10^{-3})^3 + 7(10^{-3})^4 \simeq 6 \times 10^{-6} \end{aligned}$$

5.3.3.2 Rectangular Code

A *rectangular code*, also called a *product code*, can be thought of as a parallel data transmission, depicted in Figure 5.11b. First we form a rectangle of message bits comprised of M rows and N columns; then a horizontal parity check is appended to each row and a vertical parity check is appended to each column, resulting in an augmented array of dimensions $(M+1) \times (N+1)$. The rate of the rectangular code, k/n, can then be written as

$$\frac{k}{n} = \frac{MN}{(M+1)(N+1)} \tag{5.17}$$

How much more powerful is the rectangular code than the single-parity code, which is only capable of error detection? Notice that any single bit error will cause a parity check failure in one of the array columns *and* in one of the array rows. Therefore, the rectangular code can correct a single error pattern since the error is uniquely located at the intersection of the error-detecting row and the error-detecting column. For the example shown in Figure 5.11b, the array dimensions are $M = N = 5$; therefore, the figure depicts a (36, 25) code that can correct a single error located anywhere in the 36 bit positions. For an error-correcting block code, we compute the probability that the decoded block has an uncorrected error by accounting for all the ways in which a *message error* can be made. Starting with the probability of j errors in a block of n symbols, expressed in Equation (5.15), we can write the probability of a message error, also called a *block error* or *word error*, P_M, for a code that can correct all t and fewer error patterns:

$$P_M = \sum_{j=t+1}^{n} \binom{n}{j} p^j (1 - p)^{n-j} \tag{5.18}$$

where p is the probability that a *channel symbol* is received in error. For the example in Figure 5.11b, the code can correct all single error patterns ($t = 1$) within the rectangular block of $n = 36$ bits. Hence the summation in Equation (5.18) starts with $j = 2$:

$$P_M = \sum_{j=2}^{36} \binom{36}{j} p^j (1 - p)^{36-j} \tag{5.19}$$

When p is reasonably small, the first term in the summation is the dominant one; we can therefore write for this (36, 25) rectangular code example

$$P_M \simeq \binom{36}{2} p^2 (1 - p)^{34}$$

The *bit error probability*, P_B, depends on the particular code and decoder. An approximation for P_B is given in Section 5.5.3.

5.3.4 Coding Gain

Figure 5.12 illustrates the probability of bit error, P_B, versus E_b/N_0 for coherent binary PSK modulation in combination with examples of various (n, k) codes over a Gaussian channel. The (1, 1) curve illustrates the uncoded PSK performance, while the (24, 12) and (127, 92) curves illustrate coded PSK performance using block codes with $(n - k) = 12$ parity bits and 35 parity bits, respectively. From Figure 3.24 we know in which direction the waterfall-like curves move, corresponding to P_B performance improvement. Look at the various curves in Figure 5.12. Can you explain why the coded curves (to which we attribute P_B performance improvement) appear to be moving in the wrong direction when compared with the uncoded curve? Where does the strength of the code manifest itself? The curves in Figure 5.12 indicate that the strength of a code is seen only after an E_b/N_0 threshold has been exceeded (approximately 5.5 dB in this example). For values of E_b/N_0 less than the threshold, the coding manifests itself only as *over-*

ıe single-parity code,
ʒle bit error will cause
ne of the array rows.
attern since the error
ıg row and the error-
the array dimensions
ıde that can correct a
ɜrror-correcting block
ȝ an uncorrected error
:an be made. Starting
xpressed in Equation
o called a *block error*
ver error patterns:

(5.18)

ved in error. For the
:rror patterns ($t = 1$)
ımmation in Equation

(5.19)

n is the dominant one;
ҳample

:ode and decoder. An

ıus E_b/N_0 for coherent
ırious (n, k) codes over
led PSK performance,
SK performance using
its, respectively. From
e curves move, corre-
ırious curves in Figure
ɜ attribute P_B perform-
ection when compared
ıde manifest itself? The
e is seen only after an
B in this example). For
ests itself only as *over-*

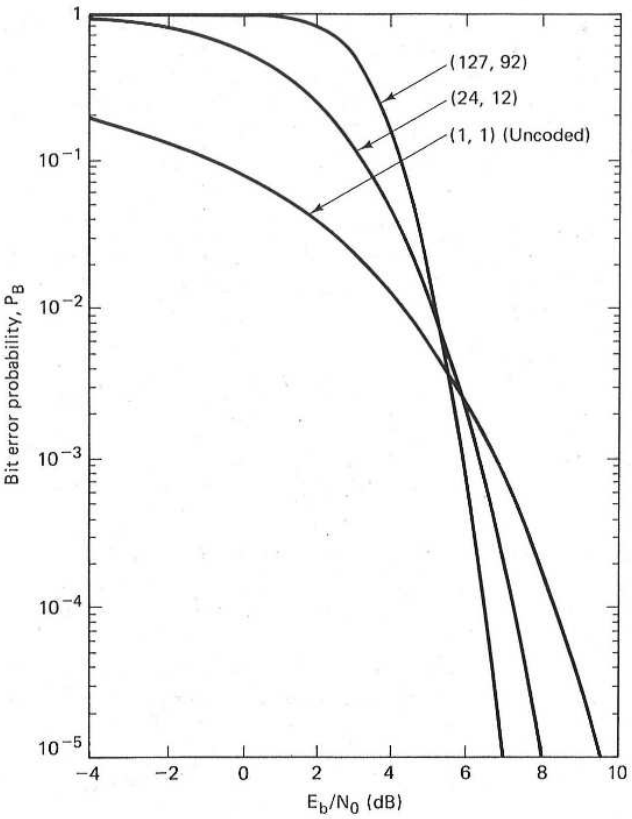


Figure 5.12 Coded versus uncoded bit error performance for coherent PSK with various (n, k) codes.

head bits resulting in *reduced energy per bit*, compared to the uncoded case; before the threshold is exceeded, the redundant bits are simply "excess baggage" without the ability to improve performance. Once the threshold is exceeded, the performance improvement of the code more than compensates for the reduction in energy per coded bit. Therefore, in Figure 5.12, once the threshold value of $E_b/N_0 = 5.5$ dB is exceeded, the relative positions of the curves reverse themselves compared to their positions at less-than-threshold E_b/N_0. *Coding gain* is defined as the reduction, expressed in decibels, in the required E_b/N_0 to achieve a specified error performance of an error-correcting coded system over an uncoded one with the same modulation. For example, in Figure 5.12, for $P_B = 10^{-5}$, the (24, 12) code has a coding gain of about 1.5 dB.

Example 5.2 Coded versus Uncoded Performance

Compare the message error probability for a communications link with and without the use of error-correction coding. Assume that the uncoded transmission characteristics are: BPSK modulation, Gaussian noise, $S/N_0 = 43{,}776$, data rate $R = 4800$ bits/s. For the coded case, also assume the use of a (15, 11) error-correcting code that is capable of correcting any single-error pattern within a block of 15 bits. Consider that the demodulator makes hard decisions and thus feeds the demodulated code bits directly to the decoder, which in turn outputs an estimate of the original message.

Solution

Following Equation (3.84), let $p_u = Q\sqrt{2E_b/N_0}$ and $p_c = Q\sqrt{2E_c/N_0}$ be the uncoded and coded channel symbol error probabilities, respectively, where E_b/N_0 is uncoded bit energy per noise spectral density and E_c/N_0 is the coded bit energy per noise spectral density.

Without coding

$$\frac{E_b}{N_0} = \frac{S}{RN_0} = 9.12\ (9.6\ \text{dB})$$

$$p_u = Q\left(\sqrt{\frac{2E_b}{N_0}}\right) = Q(\sqrt{18.24}) = 1.02 \times 10^{-5} \tag{5.20}$$

where the following approximation of $Q(x)$ from Equation (2.43) was used:

$$Q(x) \simeq \frac{1}{x\sqrt{2\pi}} \exp\left(\frac{-x^2}{2}\right) \qquad \text{for } x > 3$$

The probability that the uncoded message block, P_M^u, will be received in error is 1 minus the product of the probabilities that each bit will be detected correctly. Thus

$$\begin{aligned} P_M^u &= 1 - (1 - p_u)^k \\ &= 1 - \underbrace{(1 - p_u)^{11}}_{\substack{\text{probability that all}\\ \text{11 bits in uncoded}\\ \text{block are correct}}} \qquad = \underbrace{1.12 \times 10^{-4}}_{\substack{\text{probability that at}\\ \text{least 1 bit out of}\\ \text{11 is in error}}} \end{aligned} \tag{5.21}$$

With coding:

The channel symbol rate, sometimes called the coded bit rate, R_c is 15/11 times the data bit rate.

$$R_c = 4800 \times \tfrac{15}{11} \simeq 6545 \text{ bps}$$

$$\frac{E_c}{N_0} = \frac{S}{R_cN_0} = 6.688\ (8.25\ \text{dB})$$

The E_c/N_0 for each code bit is less than that for the uncoded bit because the channel bit rate has increased but the transmitter power is assumed to be fixed.

$$p_c = Q\left(\sqrt{\frac{2E_c}{N_0}}\right) = Q(\sqrt{13.38}) = 1.36 \times 10^{-4} \tag{5.22}$$

It
channe
same ti
ment d
error ra

T
within
first te

By cor
of mes
used i

5.4 LINEAR BLOC

Linear bloc
codes that
encoder tra
block of n
elements.
binary cod
is restricte
The
tuples (seq
sequences.
2^k message
assignmen
of 2^k code
For *linear*

5.4.1 Vec

The set of
of two ele
multiplica
elements.
the conve
of additio

76, data rate $R = 4800$
error-correcting code
block of 15 bits. Con-
eeds the demodulated
stimate of the original

$\overline{2E_c/N_0}$ be the uncoded
here E_b/N_0 is uncoded
d bit energy per noise

10^{-5} (5.20)

2.43) was used:

$\cdot\, 3$

be received in error is 1
letected correctly. Thus

-4

(5.21)

ıat at
t of

ate, R_c is 15/11 times the

d bit because the channel
d to be fixed.

$\times\, 10^{-4}$ (5.22)

It can be seen by comparing the results of Equation (5.20) with (5.22) that the channel bit error probability has degraded. More bits must be detected during the same time interval, and with the same available power; the performance improvement due to the coding *is not yet apparent*. We now compute the coded message error rate, P_M^c, using Equation (5.18).

$$P_M^c = \sum_{j=2}^{n=15} \binom{15}{j} (p_c)^j (1 - p_c)^{15-j}$$

The summation is started with $j = 2$ since the code corrects all single errors within a block of $n = 15$ bits. A good approximation is obtained by using only the first term of the summation. For p_c we use the value calculated in Equation (5.22):

$$P_M^c = \binom{15}{2} (p_c)^2 (1 - p_c)^{13} = 1.94 \times 10^{-6} \tag{5.23}$$

By comparing the results of Equation (5.21) with (5.23), it is seen that the probability of message error has improved by a factor of 58 due to the error-correcting code used in this example.

5.4 LINEAR BLOCK CODES

Linear block codes (such as the one in Example 5.2) are a class of parity check codes that can be characterized by the (n, k) notation described earlier. The encoder transforms a block of k message digits (a message vector) into a longer block of n codeword digits (a code vector), constructed from a given alphabet of elements. When the alphabet consists of two elements (0 and 1), the code is a binary code comprised of binary digits (bits). Our discussion of linear block codes is restricted to binary codes, unless otherwise noted.

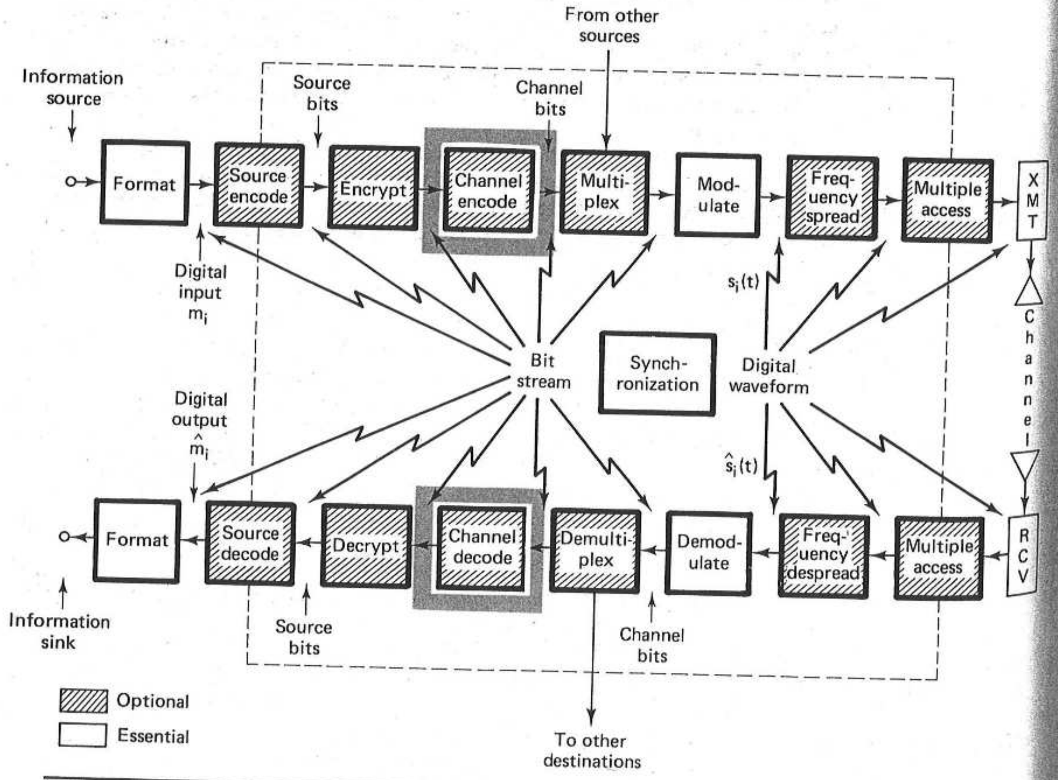
The k-bit messages form 2^k distinct message sequences referred to as *k-tuples* (sequences of k digits). The n-bit blocks can form as many as 2^n distinct sequences, referred to as *n-tuples*. The encoding procedure assigns to each of the 2^k message k-tuples *one* of the 2^n n-tuples. A block code represents a one-to-one assignment, whereby the 2^k message k-tuples are *uniquely* mapped into a new set of 2^k codeword n-tuples; the mapping can be accomplished via a look-up table. For *linear codes*, the mapping transformation is, of course, *linear*.

5.4.1 Vector Spaces

The set of all binary n-tuples, V_n, is called a *vector space* over the binary field of two elements (0 and 1). The binary field has two operations, addition and multiplication, such that the results of all operations are in the same set of two elements. The arithmetic operations of addition and multiplication are defined by the conventions of the algebraic field [4]. For example, in a binary field, the rules of addition and multiplication are as follows:

CHAPTER 6

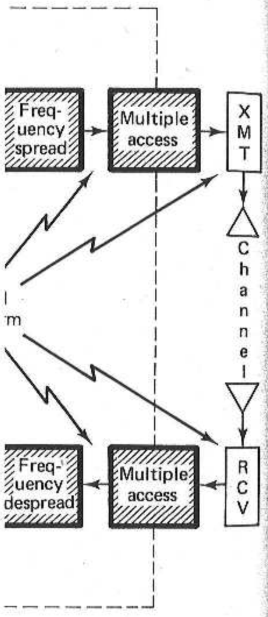
Channel Coding: Part 2



There are two r
ter 5 deals main
coding. A *linea*
matrix or polyr
to a block enc
codeword out
codeword *n*-tu
k/*n* is called th
A *convolution*
k/*n* has the sar
block codes; h
for block code
represents the
characteristic
coder has men
is not only a f
$K - 1$ input
control the re

6.1 CONVOLUTION

In Figure 1.2
system. A ve
lutional enco



There are two major categories of channel codes: block and convolutional. Chapter 5 deals mainly with block coding. This chapter deals mainly with convolutional coding. A *linear block code* is described by two integers, n and k, and a generator matrix or polynomial. The integer k is the number of data bits that form an input to a block encoder. The integer n is the total number of bits in the associated codeword out of the encoder. A characteristic of linear block codes is that each codeword n-tuple is uniquely determined by the input message k-tuple. The ratio k/n is called the *rate* of the code—a measure of the amount of added redundancy. A *convolutional code* is described by three integers, n, k, and K, where the ratio k/n has the same code rate significance (information per coded bit) that it has for block codes; however, n does *not* define a block or codeword length as it does for block codes. The integer K is a parameter known as the *constraint length*; it represents the number of k-tuple stages in the encoding shift register. An important characteristic of convolutional codes, different from block codes, is that the encoder has memory—the n-tuple emitted by the convolutional encoding procedure is not only a function of an input k-tuple, but is also a function of the previous $K - 1$ input k-tuples. In practice, n and k are small integers and K is varied to control the redundancy.

6.1 CONVOLUTIONAL ENCODING

In Figure 1.2 we presented a typical block diagram of a digital communication system. A version of this functional diagram, focusing primarily on the convolutional encode/decode and modulate/demodulate portions of the communication link, is shown in Figure 6.1. The input message source is denoted by the sequence $\mathbf{m} = m_1, m_2, \ldots, m_i, \ldots$, where each m_i represents a binary digit (bit). We shall assume that each m_i is equally likely to be a one or a zero, and independent from digit to digit. Being independent, the bit sequence lacks any redundancy; that is, knowledge about bit m_i gives no information about m_j $(i \neq j)$. The encoder transforms each sequence $\mathbf{m}$ into a unique codeword sequence $\mathbf{U} = G(\mathbf{m})$. Even though the sequence $\mathbf{m}$ uniquely defines the sequence $\mathbf{U}$, a key feature of convolutional codes is that a given k-tuple within $\mathbf{m}$ does *not* uniquely define its associated n-tuple within $\mathbf{U}$ since the encoding of each k-tuple is *not only* a function of that k-tuple but is also a function of the $K - 1$ input k-tuples that precede it. The sequence $\mathbf{U}$ can be partitioned into a sequence of branch words: $\mathbf{U} = U_1, U_2, \ldots, U_i, \ldots$. Each branch word U_i is made up of binary *code symbols*, often called *channel symbols*, *channel bits*, or *coded bits*; unlike the input message bits the code symbols are not independent.

In a typical communication application, the codeword sequence $\mathbf{U}$ modulates a waveform $s(t)$. During transmission, the waveform $s(t)$ is corrupted by noise, resulting in a received waveform $\hat{s}(t)$ and a demodulated sequence $\mathbf{Z} = Z_1, Z_2, \ldots, Z_i, \ldots$, as indicated in Figure 6.1. The task of the decoder is to produce an estimate $\hat{\mathbf{m}} = \hat{m}_1, \hat{m}_2, \ldots, \hat{m}_i, \ldots$, of the original message sequence, using the received sequence $\mathbf{Z}$ together with a priori knowledge of the encoding procedure.

A general convolutional encoder, shown in Figure 6.2, is mechanized with a kK-stage shift register and n modulo-2 adders, where K is the constraint length. The constraint length represents the number of k-bit shifts over which a single information bit can influence the encoder output. At each unit of time, k bits are shifted into the first k stages of the register; all bits in the register are shifted k stages to the right, and the outputs of the n adders are sequentially sampled to

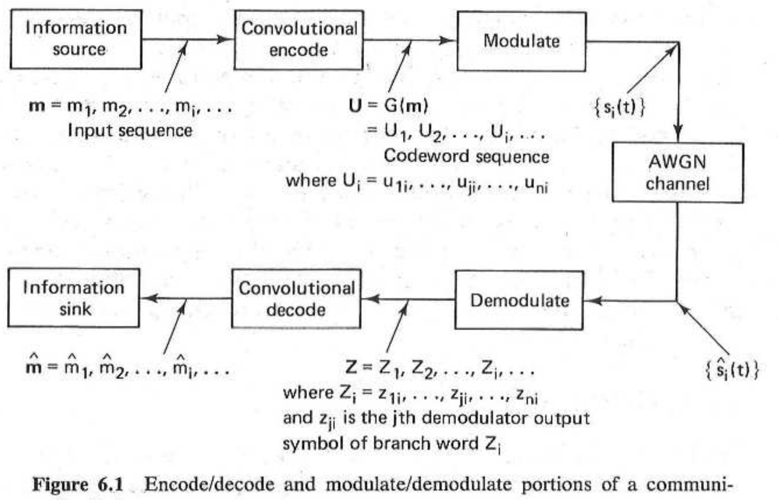
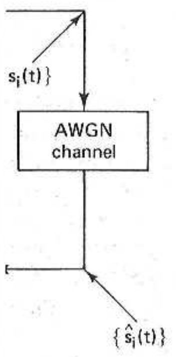

Information source
Convolutional encode
Modulate
$\mathbf{m} = m_1, m_2, \ldots, m_i, \ldots$
Input sequence
$\mathbf{U} = G(\mathbf{m})$
$= U_1, U_2, \ldots, U_i, \ldots$
Codeword sequence
where $U_i = u_{1i}, \ldots, u_{ji}, \ldots, u_{ni}$
$\{s_i(t)\}$
AWGN channel
Information sink
Convolutional decode
Demodulate
$\hat{\mathbf{m}} = \hat{m}_1, \hat{m}_2, \ldots, \hat{m}_i, \ldots$
$\mathbf{Z} = Z_1, Z_2, \ldots, Z_i, \ldots$
where $Z_i = z_{1i}, \ldots, z_{ji}, \ldots, z_{ni}$
and z_{ji} is the jth demodulator output symbol of branch word Z_i
$\{\hat{s}_i(t)\}$

Figure 6.1 Encode/decode and modulate/demodulate portions of a communication link.

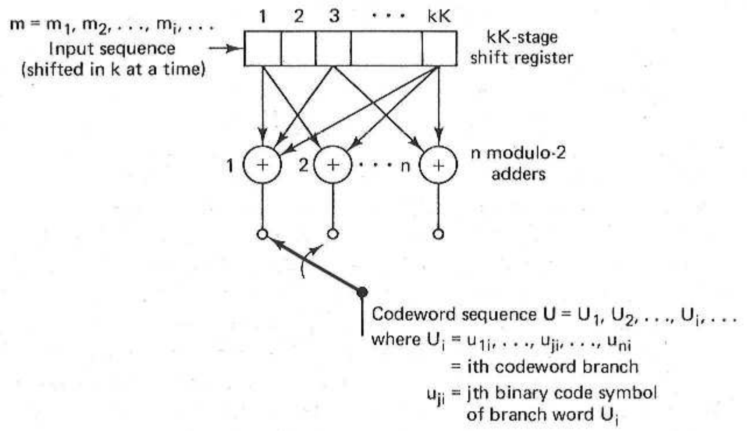
ted by the sequence
nary digit (bit). We
:o, and independent
ks any redundancy;
$(i \neq j)$. The encoder
ce $\mathbf{U} = G(\mathbf{m})$. Even
key feature of con-
uniquely define its
s *not only* a function
ples that precede it.
ch words: $\mathbf{U} = U_1$,
nary *code symbols*,
ce the input message

quence **U** modulates
corrupted by noise,
quence $\mathbf{Z} = Z_1, Z_2$,
coder is to produce
age sequence, using
ge of the encoding

is mechanized with
ne constraint length.
over which a single
it of time, k bits are
egister are shifted k
ientially sampled to



of a communi-



$\mathbf{m} = m_1, m_2, \ldots, m_i, \ldots$
Input sequence (shifted in k at a time)
1 2 3 · · · kK
kK-stage shift register
1 2 · · · n
n modulo-2 adders
Codeword sequence $\mathbf{U} = U_1, U_2, \ldots, U_i, \ldots$
where $U_i = u_{1i}, \ldots, u_{ji}, \ldots, u_{ni}$
= ith codeword branch
u_{ji} = jth binary code symbol of branch word U_i

Figure 6.2 Convolutional encoder with constraint length K and rate k/n.

yield the binary code symbols or coded bits. These code symbols are then used by the modulator to specify the waveforms to be transmitted over the channel. Since there are n coded bits for each input group of k message bits, the code rate is k/n message bit per coded bit, where $k < n$.

We shall consider only the most commonly used binary convolutional encoders for which $k = 1$, that is, those encoders in which the message bits are shifted into the encoder one bit at a time, although generalization to higher-order alphabets is straightforward [1, 2]. For the $k = 1$ encoder, at the ith unit of time, message bit m_i is shifted into the first shift register stage; all previous bits in the register are shifted one stage to the right, and as in the more general case, the outputs of the n adders are sequentially sampled and transmitted. Since there are n coded bits for each message bit, the code rate is $1/n$. The n code symbols occurring at time t_i comprise the ith branch word, $U_i = u_{1i}, u_{2i}, \ldots, u_{ni}$, where u_{ji} $(j = 1, 2, \ldots, n)$ is the jth code symbol belonging to the ith branch word. Note that for the rate $1/n$ encoder, the kK-stage shift register can be referred to simply as a K-stage register, and the constraint length K, which was expressed in units of k-tuple stages, can be referred to as constraint length in units of bits.

6.2 CONVOLUTIONAL ENCODER REPRESENTATION

To describe a convolutional code, one needs to characterize the encoding function $G(\mathbf{m})$, so that given an input sequence $\mathbf{m}$, one can readily compute the output sequence $\mathbf{U}$. Several methods are used for representing a convolutional encoder, the most popular being the *connection pictorial*, *connection vectors or polynomials*, the *state diagram*, the *tree diagram*, and the *trellis diagram*. They are each described below.

6.2.1 Connection Representation

We shall use the convolutional encoder, shown in Figure 6.3, as a model for discussing convolutional encoders. The figure illustrates a (2, 1) convolutional encoder with constraint length $K = 3$. There are $n = 2$ modulo-2 adders; thus the code rate k/n is $\frac{1}{2}$. At each input bit time, a bit is shifted into the leftmost stage and the bits in the register are shifted one position to the right. Next, the output switch samples the output of each modulo-2 adder (i.e., first the upper adder, then the lower adder), thus forming the code symbol pair making up the branch word associated with the bit just inputted. The sampling is repeated for each inputted bit. The choice of connections between the adders and the stages of the register gives rise to the characteristics of the code. Any change in the choice of connections results in a different code. The connections are, of course, *not* chosen or changed arbitrarily. The problem of choosing connections to yield good distance properties is complicated and has not been solved in general; however, good codes have been found by computer search for all constraint lengths less than about 20 [3–5].

Unlike a block code that has a fixed word length n, a convolutional code has no particular block size. However, convolutional codes are often forced into a block structure by *periodic truncation*. This requires a number of zero bits to be appended to the end of the input data sequence, for the purpose of clearing or *flushing* the encoding shift register of the data bits. Since the added zeros carry no information, the *effective code rate* falls below k/n. To keep the code rate close to k/n, the truncation period is generally made as long as practical.

One way to represent the encoder is to specify a set of *n connection vectors*, one for each of the n modulo-2 adders. Each vector has dimension K and describes the connection of the encoding shift register to that modulo-2 adder. A one in the ith position of the vector indicates that the corresponding stage in the shift register is connected to the modulo-2 adder, and a zero in a given position indicates that no connection exists between the stage and the modulo-2 adder. For the encoder example in Figure 6.3, we can write the connection vector $\mathbf{g}_1$ for the upper connections and $\mathbf{g}_2$ for the lower connections as follows:

$$\mathbf{g}_1 = 1\ 1\ 1$$

$$\mathbf{g}_2 = 1\ 0\ 1$$

Consider that a message vector $\mathbf{m} = 1\ 0\ 1$ is convolutionally encoded with the encoder shown in Figure 6.3. The three message bits are inputted, one at a time, at times t_1, t_2, and t_3, as shown in Figure 6.4. Subsequently, $(K - 1) = 2$ zeros are inputted at times t_4 and t_5 to flush the register and thus ensure that the tail end of the message is shifted the full length of the register. The output sequence is seen to be 1 1 1 0 0 0 1 0 1 1, where the leftmost symbol represents the earliest transmission. The entire output sequence, including the code symbols as a result of flushing, are needed to decode the message. To flush the message from the encoder requires one less zero than the number of stages in the register, or $K - 1$ flush bits. Another zero input is shown at time t_6, for the reader to verify that the corresponding branch word output is then 00.

Input
m

6.2.1.1 I

We can a
response of th
the contents o

Input sequen
Output seque

The output s
encoder. The
the *superpos*
follows:

Inpu

1
0
1

Modulo-

Observe tha
that *convolu*
It is from th
shifted impu
sponse of th

6.3, as a model for
(2, 1) convolutional
ıdulo-2 adders; thus
to the leftmost stage
ht. Next, the output
'st the upper adder,
aking up the branch
s repeated for each
ınd the stages of the
ınge in the choice of
f course, *not* chosen
ɔ yield good distance
ıowever, good codes
ıs less than about 20

convolutional code
ıre often forced into
mber of zero bits to
purpose of clearing
ıe added zeros carry
keep the code rate
as practical.
connection vectors,
sion K and describes
! adder. A one in the
;e in the shift register
ɔsition indicates that
ler. For the encoder
g_1 for the upper con-

lly encoded with the
ıutted, one at a time,
', $(K - 1) = 2$ zeros
s ensure that the tail
The output sequence
mbol represents the
the code symbols as
ısh the message from
es in the register, or
r the reader to verify

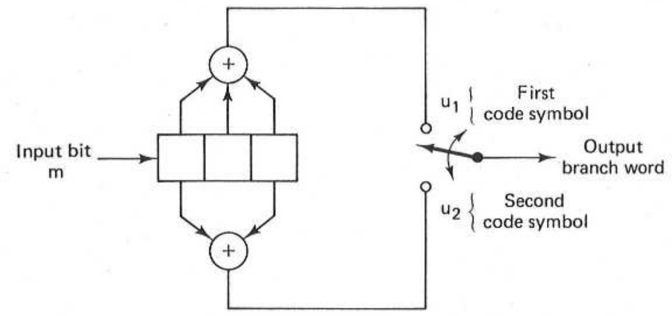


Figure 6.3 Convolutional encoder (rate $\frac{1}{2}$, $K = 3$).

6.2.1.1 Impulse Response of the Encoder

We can approach the encoder in terms of its *impulse response*—that is, the response of the encoder to a single "one" bit that moves through it. Consider the contents of the register in Figure 6.3 as a one moves through it.

Register contents	Branch word u_1	Branch word u_2
1 0 0	1	1
0 1 0	1	0
0 0 1	1	1

Input sequence:	1	0	0
Output sequence:	1 1	1 0	1 1

The output sequence for the input "one" is called the impulse response of the encoder. Then for the input sequence $\mathbf{m} = 1\ 0\ 1$, the output may be found by the *superposition* or the *linear addition* of the time-shifted input "impulses" as follows:

Input $\mathbf{m}$	Output				
1	1 1	1 0	1 1		
0		0 0	0 0	0 0	
1			1 1	1 0	1 1
Modulo-2 sum:	1 1	1 0	0 0	1 0	1 1

Observe that this is the same output as that obtained in Figure 6.4, demonstrating that *convolutional codes are linear*—just as the linear block codes of Chapter 5. It is from this property of generating the output by the linear addition of time-shifted impulses, or the convolution of the input sequence with the impulse response of the encoder, that we derive the name *convolutional encoder*. Often,

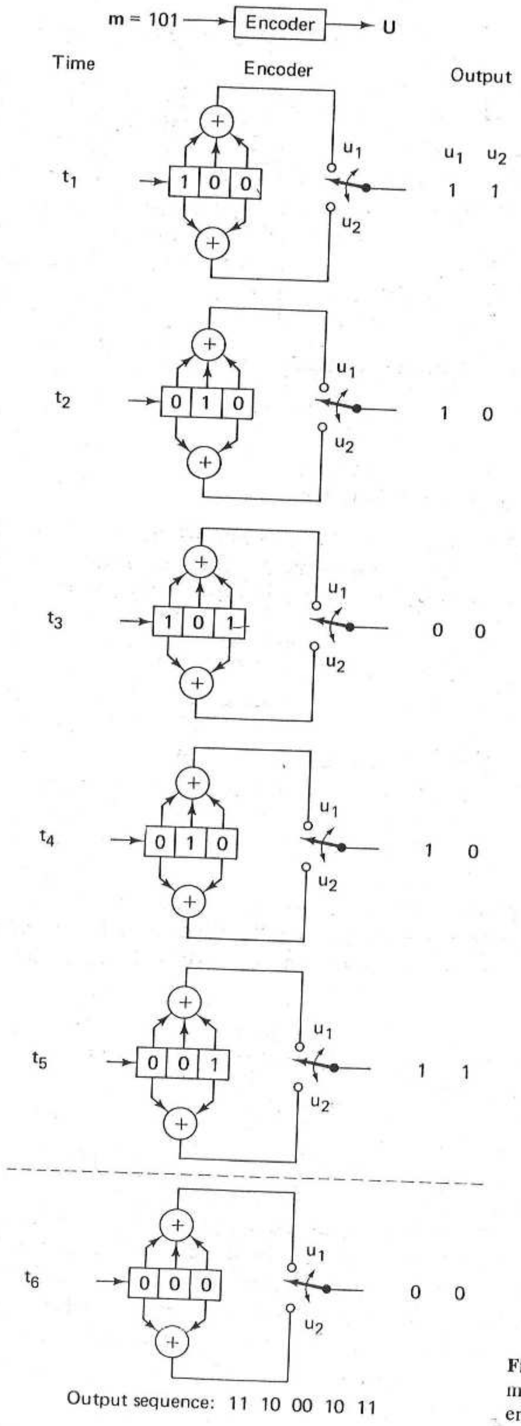


Figure 6.4 Convolutionally encoding a message sequence with a rate $\frac{1}{2}$, $K = 3$ encoder.

this encoder char
matrix [6].

Notice that
sequence and 10-
that might have b
a pair of output
bit into the enco
channel bits are
say 300 bits, the
a code rate of 3(

6.2.1.2 Pol

Sometimes
nomials, similar
ister implement
with a set of *n*
Each polynomi
encoding shift
nection vector
is either 1 or 0,
the shift regist
in Figure 6.3,
nections and g

where the low
the register. T

First, express
$1 + X^2$. We
flush the regis
of the Figure

$\mathbf{m}(X)$g

$\mathbf{m}(X)$g

$\mathbf{m}(X)$g

$\mathbf{m}(X)$

olutionally encoding a
: with a rate $\frac{1}{2}$, $K = 3$

this encoder characterization is presented in terms of an infinite-order generator matrix [6].

Notice that the *effective code rate* for the foregoing example with 3-bit input sequence and 10-bit output sequence is $k/n = \frac{3}{10}$—quite a bit less than the rate $\frac{1}{2}$ that might have been expected from the knowledge that each input data bit yields a pair of output channel bits. The reason for the disparity is that the final data bit into the encoder needs to be shifted through the encoder. All of the output channel bits are needed in the decoding process. If the message had been longer, say 300 bits, the output codeword sequence would contain 604 bits, resulting in a code rate of 300/604—much closer to $\frac{1}{2}$.

6.2.1.2 Polynomial Representation

Sometimes, the encoder connections are characterized by *generator polynomials*, similar to those used in Chapter 5 for describing the feedback shift register implementation of cyclic codes. We can represent a convolutional encoder with a set of n generator polynomials, one for each of the n modulo-2 adders. Each polynomial is of degree $K - 1$ or less and describes the connection of the encoding shift register to that modulo-2 adder, much the same way that a connection vector does. The coefficient of each term in the $(K - 1)$-degree polynomial is either 1 or 0, depending on whether a connection exists or does not exist between the shift register and the modulo-2 adder in question. For the encoder example in Figure 6.3, we can write the generator polynomial $\mathbf{g}_1(X)$ for the upper connections and $\mathbf{g}_2(X)$ for the lower connections as follows:

$$\mathbf{g}_1(X) = 1 + X + X^2$$

$$\mathbf{g}_2(X) = 1 + X^2$$

where the lowest-order term in the polynomial corresponds to the input stage of the register. The output sequence is found as follows:

$$\mathbf{U}(X) = \mathbf{m}(X)\mathbf{g}_1(X) \text{ interlaced with } \mathbf{m}(X)\mathbf{g}_2(X)$$

First, express the message vector $\mathbf{m} = 1\ 0\ 1$ as a polynomial—that is, $\mathbf{m}(X) = 1 + X^2$. We shall again assume the use of zeros following the message bits, to flush the register. Then the output polynomial, $\mathbf{U}(X)$, or the output sequence, $\mathbf{U}$, of the Figure 6.3 encoder can be found for the input message $\mathbf{m}$ as follows:

$$\mathbf{m}(X)\mathbf{g}_1(X) = (1 + X^2)(1 + X + X^2) = 1 + X + X^3 + X^4$$

$$\mathbf{m}(X)\mathbf{g}_2(X) = (1 + X^2)(1 + X^2) = 1 + X^4$$

$$\mathbf{m}(X)\mathbf{g}_1(X) = 1 + X + 0X^2 + X^3 + X^4$$

$$\mathbf{m}(X)\mathbf{g}_2(X) = 1 + 0X + 0X^2 + 0X^3 + X^4$$

$$\mathbf{U}(X) = (1, 1) + (1, 0)X + (0, 0)X^2 + (1, 0)X^3 + (1, 1)X^4$$

$$\mathbf{U} = 1\ 1 \quad 1\ 0 \quad 0\ 0 \quad 1\ 0 \quad 1\ 1$$

In this example we started with another point of view—that the convolutional encoder can be treated as a set of *cyclic code shift registers*. We represented the encoder with *polynomial generators* as used for describing cyclic codes. However, we arrived at the same output sequence as in Figure 6.4, and the same output sequence as the impulse response treatment of the preceding section. For a good presentation of convolutional code structure in the context of linear sequential circuits, see Reference [7].

6.2.2 State Representation and the State Diagram

The state of a rate $1/n$ convolutional encoder is defined as the contents of the rightmost $K - 1$ stages (see Figure 6.3). Knowledge of the state together with knowledge of the next input is necessary and sufficient to determine the next output. Let the state of the encoder at time, t_i, be defined as $X_i = m_{i-1}, m_{i-2}, \ldots, m_{i-K+1}$. The ith codeword branch, U_i, is completely determined by state X_i and the present input bit m_i; thus the state X_i represents the past history of the encoder in determining the encoder output. The encoder state is said to be *Markov*, in the sense that the probability, $P(X_{i+1}|X_i, X_{i-1}, \ldots, X_0)$, of being in state X_{i+1}, given all previous states, depends only on the most recent state, X_i; that is, the probability is equal to $P(X_{i+1}|X_i)$.

One way to represent simple encoders is with a *state diagram*; such a representation for the encoder in Figure 6.3 is shown in Figure 6.5. The states, shown in the boxes of the diagram, represent the possible contents of the rightmost $K - 1$ stages of the register, and the paths between the states represent the output branch words resulting from such state transitions. The states of the register are designated $a = 00$, $b = 10$, $c = 01$, and $d = 11$; the diagram shown in Figure 6.5 illustrates all the state transitions that are possible for the encoder in Figure

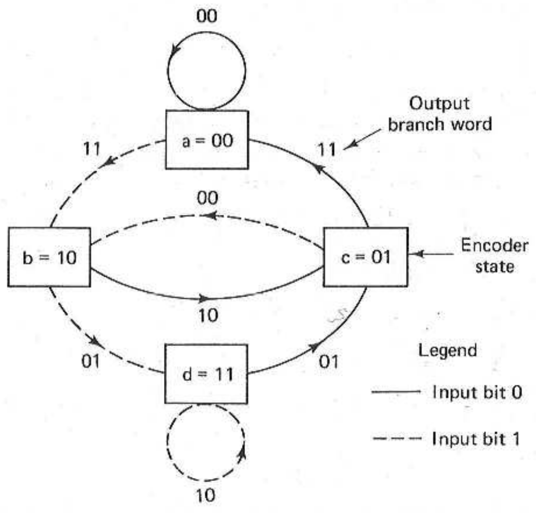


Figure 6.5 Encoder state diagram (rate $\frac{1}{2}$, $K = 3$).

6.3. There ar
the two possi
branch word
convention tl
a dashed lin
not possible
state. As a c
state transiti
present enco
00 or 10.

Example 6.1

For the
codewo
$K - 1$ =
are all z

Solutio

Example 6.2

In Exa
to the
encodi
6.1 wit
and ve
is diffe

the convolutional
Ve represented the
c codes. However,
d the same output
ection. For a good
of linear sequential

he contents of the
state together with
determine the next
$X_i = m_{i-1}, m_{i-2}$,
letermined by state
the past history of
state is said to be
$\ldots, X_0$), of being
most recent state,

liagram; such a re-
. The states, shown
ts of the rightmost
epresent the output
s of the register are
m shown in Figure
e encoder in Figure

coder state diagram (rate

6.3. There are *only two transitions* emanating from each state, corresponding to the two possible input bits. Next to each path between states is written the output branch word associated with the state transition. In drawing the path, we use the convention that a solid line denotes a path associated with an input bit, zero, and a dashed line denotes a path associated with an input bit, one. Notice that it is *not possible* in a single transition to move from a given state to *any arbitrary state*. As a consequence of shifting-in one bit at a time, there are only two possible state transitions that the register can make at each bit time. For example, if the present encoder state is 00, the *only possibilities* for the state at the next shift are 00 or 10.

Example 6.1 Convolutional Encoding

For the encoder shown in Figure 6.3, show the state changes and the resulting output codeword sequence **U** for the message sequence **m** = 1 1 0 1 1, followed by $K - 1 = 2$ zeros to flush the register. Assume that the initial contents of the register are all zeros.

Solution

Input bit m_i	Register contents	State at time t_i	State at time t_{i+1}	Branch word at time t_i u_1	u_2
–	0 0 0	0 0	0 0		–
1	1 0 0	0 0	1 0	1	1
1	1 1 0	1 0	1 1	0	1
0	0 1 1	1 1	0 1	0	1
1	1 0 1	0 1	1 0	0	0
1	1 1 0	1 0	1 1	0	1
0	0 1 1	1 1	0 1	0	1
0	$\underbrace{0\,0}_{\text{state } t_{i+1}}$ / $0\,\underbrace{0\,1}_{\text{state } t_i}$	0 1	0 0	1	1

Output sequence: **U** = 1 1 0 1 0 1 0 0 0 1 0 1 1 1

Example 6.2 Convolutional Encoding

In Example 6.1 the initial contents of the register are all zeros. This is equivalent to the condition that the given input sequence is preceded by two zero bits (the encoding is a function of the present bit and the $K - 1$ prior bits). Repeat Example 6.1 with the assumption that the given input sequence is preceded by two one bits, and verify that now the codeword sequence **U** for input sequence **m** = 1 1 0 1 1 is different than the codeword found in Example 6.1.

Solution

The entry "×" signifies "don't know."

Input bit m_i	Register contents	State at time t_i	State at time t_{i+1}	Branch word at time t_i u_1	u_2
–	1 1 ×	1 ×	1 1	–	
1	1 1 1	1 1	1 1	1	0
1	1 1 1	1 1	1 1	1	0
0	0 1 1	1 1	0 1	0	1
1	1 0 1	0 1	1 0	0	0
1	1 1 0	1 0	1 1	0	1
0	0 1 1	1 1	0 1	0	1
0	0 0 1	0 1	0 0	1	1

state t_i

state t_{i+1}

Output sequence: **U** = 1 0   1 0   0 1   0 0   0 1   0 1   1 1

By comparing this result with that of Example 6.1, we can see that each branch word of the output sequence **U** is *not only* a function of the input bit, but is also a function of the $K - 1$ prior bits.

6.2.3 The Tree Diagram

Although the state diagram completely characterizes the encoder, one cannot easily use it for tracking the encoder transitions as a function of time since the diagram cannot represent time history. The tree diagram adds the *dimension of time* to the state diagram. The tree diagram for the covolutional encoder shown in Figure 6.3 is illustrated in Figure 6.6. At each successive input bit time the encoding procedure can be described by traversing the diagram from left to right, each tree branch describing an output branch word. The branching rule for finding a codeword sequence is as follows: If the input bit is a zero, its associated branch word is found by moving to the next rightmost branch in the upward direction. If the input bit is a one, its branch word is found by moving to the next rightmost branch in the downward direction. Assuming that the initial contents of the encoder is all zeros, the diagram shows that if the first input bit is a zero, the output branch word is 00 and, if the first input bit is a one, the output branch word is 11. Similarly, if the first input bit is a one and the second input bit is a zero, the second output branch word is 10. Or, if the first input bit is a one and the second input bit is a one, the second output branch word is 01. Following this procedure we see that the input sequence 1 1 0 1 1 traces the heavy line drawn on the tree

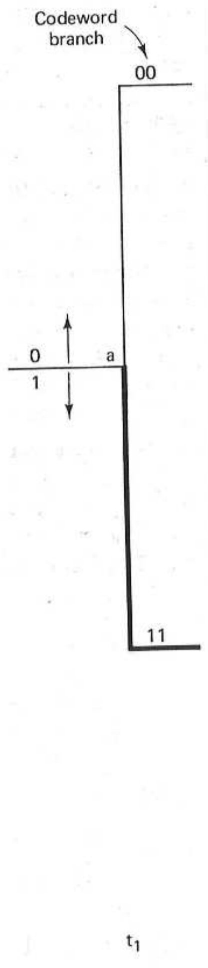


diagram in
sequence:
The a
diagram) al
ticular inpu

Branch word at time t_i	
u_1	u_2
–	
1	0
1	0
0	1
0	0
0	1
0	1
1	1

1 01 11

hat each branch
t bit, but is also

der, one cannot
f time since the
he *dimension of*
encoder shown
put bit time the
rom left to right,
g rule for finding
sociated branch
ward direction.
e next rightmost
tents of the en-
zero, the output
branch word is
it is a zero, the
and the second
g this procedure
awn on the tree

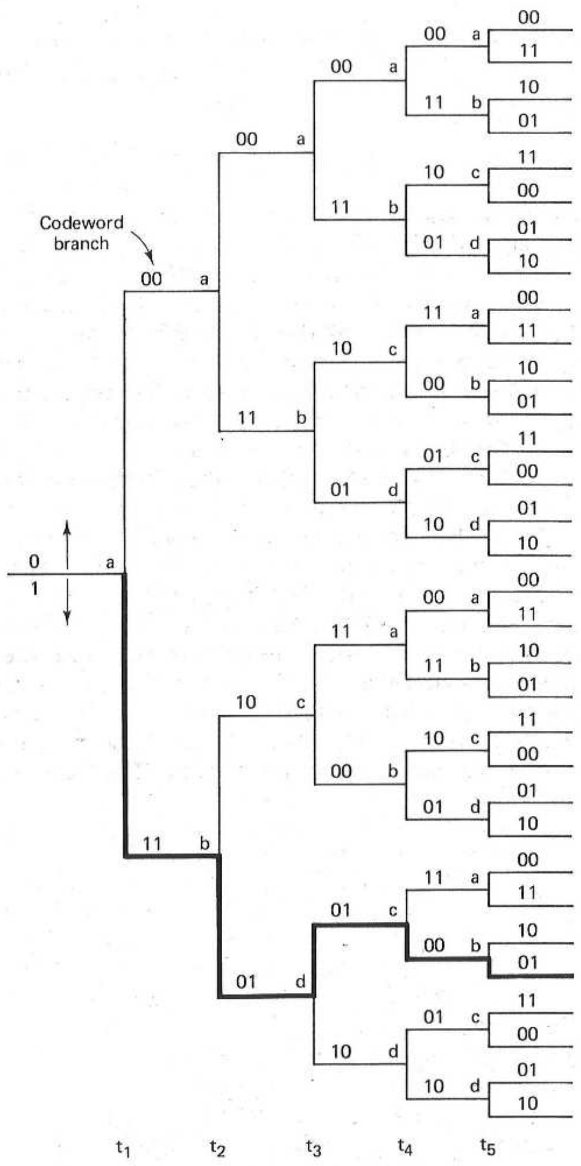


Figure 6.6 Tree representation of encoder (rate ½, $K = 3$).

diagram in Figure 6.6. This path corresponds to the following output codeword sequence: 1 1 0 1 0 1 0 0 0 1.

The added dimension of time in the tree diagram (compared to the state diagram) allows one to dynamically describe the encoder as a function of a particular input sequence. However, can you see one problem in trying to use a tree diagram for describing a sequence of any length? The number of branches increase as a function of 2^L, where L is the number of bits in the input sequence. You would quickly run out of paper, and patience.

6.2.4 The Trellis Diagram

Observation of the Figure 6.6 tree diagram shows that for this example, the structure repeats itself at time t_4, after the third branching (in general, the tree structure *repeats after K branchings*, where K is the constraint length). We label each node in the tree of Figure 6.6 to correspond to the four possible states in the shift register, as follows: $a = 00$, $b = 10$, $c = 01$, and $d = 11$. The first branching of the tree structure, at time t_1, produces a pair of nodes labeled a and b. At each successive branching the number of nodes double. The second branching, at time t_2, results in four nodes labeled a, b, c, and d. After the *third* branching there are a total of eight nodes; two of them are labeled a, two are labeled b, two are labeled c, and two are labeled d. We can see that all branches emanating from two nodes of the same state generate identical branch word sequences. From this point on, the upper and the lower halves of the tree are identical. The reason for this should be obvious from examination of the encoder in Figure 6.3. As the fourth input bit enters the encoder on the left, the first input bit is ejected on the right and no longer influences the output branch words. Consequently, the input sequences 1 0 0 x y . . . and 0 0 0 x y . . . , where the leftmost bit is the earliest bit, generate the same branch words after the ($K = 3$)rd branching. This means that any two nodes having the same state label, at the same time t_i, can be merged since all succeeding paths will be indistinguishable. If we do this to the tree structure of Figure 6.6, we obtain another diagram, called the trellis. The *trellis dia-*

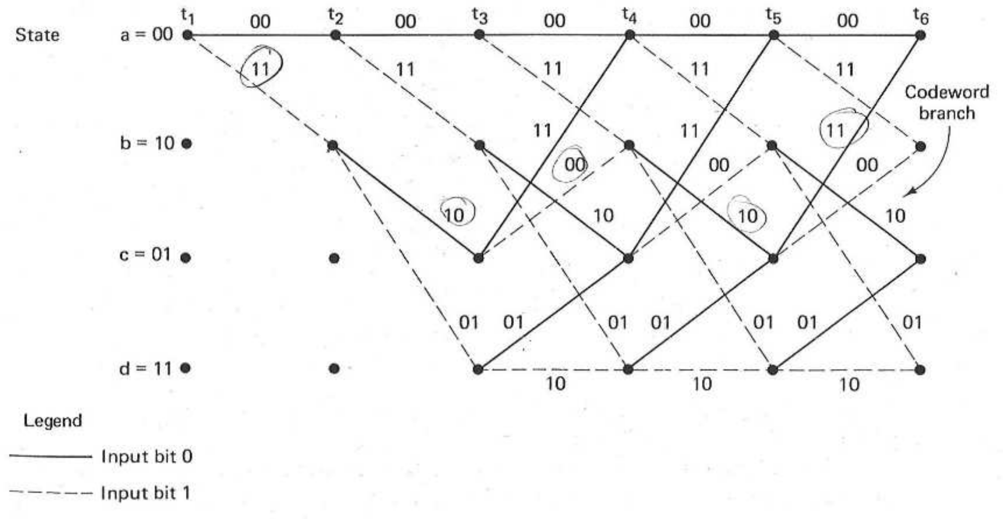


Figure 6.7 Encoder trellis diagram (rate $\frac{1}{2}$, $K = 3$).

gram, by exploi
description than
encoder of Figu
In drawin
duced with the
an input bit, ze
bit, one. The n
nodes correspo
to the states b
2^{K-1} nodes to
ample assumes
t_4). In the gene
this point, eac
Also, each of t
branches, one
input bit one.
transitions app

6.3 FORMULATION
CONVOLUTIONA

6.3.1 Maxim

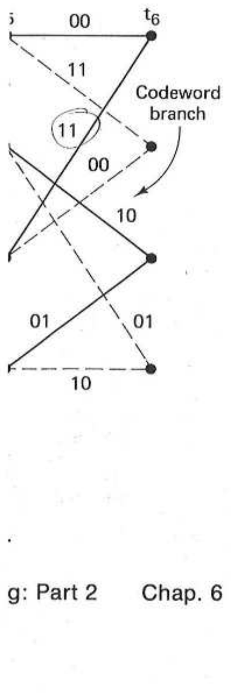
If all input m
minimum prol
also called th
and $\mathbf{U}^{(m)}$ is on
The decoder

The *maximu*
development
"common-se
the possibilit
were *only tw*
transmitted.
a received si

otherwise, t
the receiver
maximum li
a *multitude*
To be speci

f branches increase
put sequence. You

example, the struc-
l, the tree structure
We label each node
states in the shift
e first branching of
d *a* and *b*. At each
l branching, at time
ird branching there
labeled *b*, two are
es emanating from
quences. From this
cal. The reason for
Figure 6.3. As the
bit is ejected on the
equently, the input
ost bit is the earliest
nching. This means
ne t_i, can be merged
is to the tree struc-
llis. The *trellis dia-*



gram, by exploiting the repetitive structure, provides a more manageable encoder description than does the tree diagram. The trellis diagram for the convolutional encoder of Figure 6.3 is shown in Figure 6.7.

In drawing the trellis diagram, we use the same convention that we introduced with the state diagram—that a solid line denotes the output generated by an input bit, zero, and a dashed line denotes the output generated by an input bit, one. The nodes of the trellis characterize the encoder states; the first row nodes correspond to the state $a = 00$, the second and subsequent rows correspond to the states $b = 10$, $c = 01$, and $d = 11$. At each unit of time the trellis requires 2^{K-1} nodes to represent the 2^{K-1} possible encoder states. The trellis in our example assumes a fixed periodic structure after trellis depth 3 is reached (at time t_4). In the general case, the fixed structure prevails after depth K is reached. After this point, each of the states can be entered from either of two preceding states. Also, each of the states can transition to one of two states. Of the two outgoing branches, one corresponds to an input bit zero and the other corresponds to an input bit one. On Figure 6.7 the output branch words corresponding to the state transitions appear as labels on the trellis branches.

6.3 FORMULATION OF THE CONVOLUTIONAL DECODING PROBLEM

6.3.1 Maximum Likelihood Decoding

If all input message sequences are equally likely, a decoder that achieves the minimum probability of error is one that compares the conditional probabilities, also called the *likelihood functions*, $P(\mathbf{Z}|\mathbf{U}^{(m)})$, where $\mathbf{Z}$ is the received sequence and $\mathbf{U}^{(m)}$ is one of the possible transmitted sequences, and chooses the maximum. The decoder chooses $\mathbf{U}^{(m')}$ if

$$P(\mathbf{Z}|\mathbf{U}^{(m')}) = \max_{\text{all } \mathbf{U}^{(m)}} P(\mathbf{Z}|\mathbf{U}^{(m)}) \tag{6.1}$$

The *maximum likelihood* concept, as stated in Equation (6.1), is a fundamental development of decision theory (see Appendix B); it is the formalization of a "common-sense" way to make decisions when there is statistical knowledge of the possibilities. In the binary demodulation treatment in Chapters 2 and 3 there were *only two* equally likely possible signals, $s_1(t)$ or $s_2(t)$, that might have been transmitted. Therefore, to make the binary maximum likelihood decision, given a received signal, meant only to decide that $s_1(t)$ was transmitted if

$$p(z|s_1) > p(z|s_2)$$

otherwise, to decide that $s_2(t)$ was transmitted. The parameter z represents $z(T)$, the receiver output at a symbol duration time $t = T$. However, when applying maximum likelihood to the convolutional decoding problem, there are typically a *multitude* of possible codeword sequences that might have been transmitted. To be specific, an L-bit codeword sequence is a member of a set of 2^L possible sequences. Therefore, in the maximum likelihood context, we can say that the decoder chooses a particular $\mathbf{U}^{(m')}$ as the transmitted sequence if the likelihood $P(\mathbf{Z}|\mathbf{U}^{(m')})$ is greater than the likelihoods of all the other possible transmitted sequences. Such an optimal decoder, which minimizes the error probability (for the case where all transmitted sequences are equally likely), is known as a *maximum likelihood decoder*. The likelihood functions are given or computed from the specifications of the channel.

We will assume that the noise is additive white Gaussian with zero mean and the channel is *memoryless*, which means that the noise affects each code symbol *independently* of all the other symbols. For a convolutional code of rate $1/n$, we can therefore express the likelihood, $P(\mathbf{Z}|\mathbf{U}^{(m)})$ as follows:

$$P(\mathbf{Z}|\mathbf{U}^{(m)}) = \prod_{i=1}^{\infty} P(Z_i|U_i^{(m)}) = \prod_{i=1}^{\infty} \prod_{j=1}^{n} P(z_{ji}|u_{ji}^{(m)}) \tag{6.2}$$

where Z_i is the ith branch of the received sequence $\mathbf{Z}$, $U_i^{(m)}$ the ith branch of a particular codeword sequence $\mathbf{U}^{(m)}$, z_{ji} the jth code symbol of Z_i, and $u_{ji}^{(m)}$ the jth code symbol of $U_i^{(m)}$, each branch comprising n code symbols. The decoder problem consists of choosing a path through the trellis of Figure 6.7 (each possible path defines a codeword) such that

$$\prod_{i=1}^{\infty} \prod_{j=1}^{n} P(z_{ji}|u_{ji}^{(m)}) \text{ is maximized} \tag{6.3}$$

Generally, it is computationally more convenient to use the logarithm of the likelihood function since this permits the summation, instead of the multiplication, of terms. We are able to use this transformation because the logarithm is a monotonically increasing function and thus will not alter the final result in our codeword selection. We can define the log-likelihood function $\gamma_{\mathbf{U}}(m)$ as

$$\gamma_{\mathbf{U}}(m) = \log P(\mathbf{Z}|\mathbf{U}^{(m)}) = \sum_{i=1}^{\infty} \log P(Z_i|U_i^{(m)}) = \sum_{i=1}^{\infty} \sum_{j=1}^{n} \log P(z_{ji}|u_{ji}^{(m)}) \tag{6.4}$$

The decoder problem now consists of choosing a path through the tree of Figure 6.6 or the trellis of Figure 6.7 such that $\gamma_{\mathbf{U}}(m)$ is maximized. For the decoding of convolutional codes, either the tree or the trellis structure can be used. In the tree representation of the code, the fact that the paths remerge is ignored. Since the number of possible sequences for an L-symbol-long sequence is 2^L, maximum likelihood decoding of an L-bit-long received sequence, using a tree diagram, requires the "brute force" or exhaustive comparison of 2^L accumulated log-likelihood metrics, representing all the possible different codewords that could have been transmitted. Hence it is not practical to consider maximum likelihood decoding with a tree structure. It is shown in a later section that with the use of the trellis representation of the code, it is possible to configure a decoder which can discard the paths that could not possibly be candidates for the maximum likelihood sequence. The decoded path is chosen from some reduced set of *surviving paths*. Such a decoder is still optimum in the sense that the decoded path is the same

as the deco
but the ear
For a
likelihood
algorithms
problem, in
is suited to
Viterbi dec
fore optim
application
Viterbi alg

6.3.2 Cha

Before spe
cision, let
branch wo
sidered to
data and th
sequence s
the modula
The modul
or FSK). I
signal wav
maps each
waveform
When the
and then b
Cons
represente
signal is $r($
In Section
steps. In th
$= a_i + n_0$
The noise
$z(T)$ is a G
whether a
process a
comparing
$p(z|s_2)$ are
demodulat
$\{z(T)\}$, int
output can
firm or ha
the output

we can say that the
nce if the likelihood
possible transmitted
rror probability (for
is known as a *max-*
n or computed from

sian with zero mean
se affects each code
lutional code of rate
llows:

$u_{ji}^{(m)})$ (6.2)

) the *i*th branch of a
f Z_i, and $u_{ji}^{(m)}$ the *j*th
s. The decoder prob-
e 6.7 (each possible

(6.3)

the logarithm of the
of the multiplication,
the logarithm is a
ne final result in our
on $\gamma_{\mathbf{U}}(m)$ as

$|u_{ji}^{(m)})$ (6.4)

gh the tree of Figure
. For the decoding of
can be used. In the
rge is ignored. Since
ence is 2^L, maximum
sing a tree diagram,
accumulated log-like-
ords that could have
ximum likelihood de-
at with the use of the
a decoder which can
maximum likelihood
et of *surviving paths.*
ded path is the same

as the decoded path obtained from a "brute force" maximum likelihood decoder, but the early rejection of unlikely paths reduces the decoding complexity.

For an excellent tutorial on the structure of convolutional codes, maximum likelihood decoding, and code performance, see Reference [8]. There are several algorithms that yield *approximate* solutions to the maximum likelihood decoding problem, including sequential [9, 10] and threshold [11]. Each of these algorithms is suited to certain special applications, but are all suboptimal. In contrast, the *Viterbi decoding algorithm* performs maximum likelihood decoding and is therefore optimal. This does not imply that the Viterbi algorithm is best for every application; there are severe constraints imposed by hardware complexity. The Viterbi algorithm is considered in Sections 6.3.3 and 6.3.4.

6.3.2 Channel Models: Hard versus Soft Decisions

Before specifying an algorithm that will determine the maximum likelihood decision, let us describe the channel. The codeword sequence $\mathbf{U}^{(m)}$, made up of branch words, with each branch word comprised of n code symbols, can be considered to be an endless stream, as opposed to a block code, in which the source data and their codewords are partitioned into precise block sizes. The codeword sequence shown in Figure 6.1 emanates from the convolutional encoder and enters the modulator, where the code symbols are transformed into signal waveforms. The modulation may be baseband (e.g., pulse waveforms) or bandpass (e.g., PSK or FSK). In general, ℓ symbols at a time, where ℓ is an integer, are mapped into signal waveforms $s_i(t)$, where $i = 1, 2, \ldots, M = 2^\ell$. When $\ell = 1$, the modulator maps each code symbol into a binary waveform. The channel over which the waveform is transmitted is assumed to corrupt the signal with Gaussian noise. When the corrupted signal is received, it is first processed by the demodulator and then by the decoder.

Consider that a binary signal, transmitted over a symbol interval $(0, T)$, is represented by $s_1(t)$ for a binary one and $s_2(t)$ for a binary zero. The received signal is $r(t) = s_i(t) + n(t)$, where $n(t)$ is a zero-mean Gaussian noise process. In Sections 2.9 and 3.4 we described the detection of $r(t)$ in terms of two basic steps. In the first step, the received waveform is reduced to a single number, $z(T) = a_i + n_0$, where a_i is the signal component of $z(T)$ and n_0 is the noise component. The noise component, n_0, is a zero-mean *Gaussian random variable*, and thus $z(T)$ is a *Gaussian random variable* with a mean of either a_1 or a_2 depending on whether a binary one or binary zero was sent. In the second step of the detection process a decision was made as to which signal was transmitted, on the basis of comparing $z(T)$ to a threshold. The conditional probabilities of $z(T)$, $p(z|s_1)$, and $p(z|s_2)$ are shown in Figure 6.8, labeled likelihood of s_1 and likelihood of s_2. The demodulator in Figure 6.1, converts the set of time-ordered random variables, $\{z(T)\}$, into a code sequence, $\mathbf{Z}$, and passes it on to the decoder. The demodulator output can be configured in a variety of ways. It can be implemented to make a *firm or hard decision* as to whether $z(T)$ represents a zero or a one. In this case, the output of the demodulator is quantized to two levels, zero and one, and fed

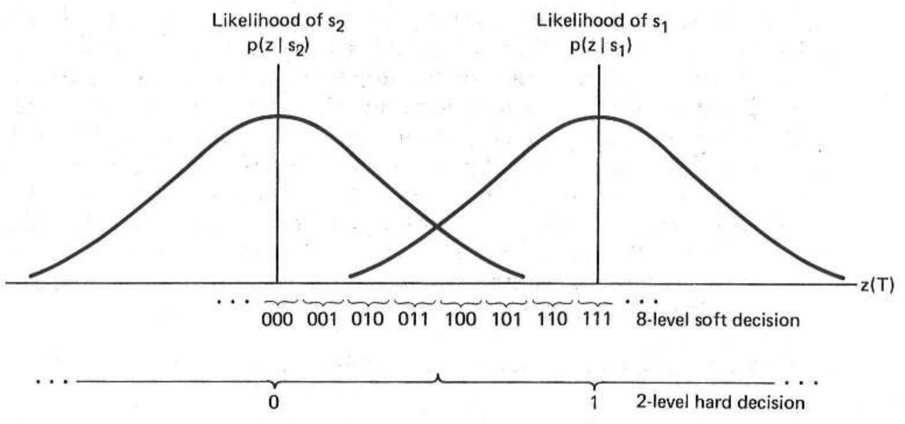


Figure 6.8 Hard and soft decoding decisions.

into the decoder (this is exactly the same threshold decision that was made in Chapters 2 and 3). Since the decoder operates on the hard decisions made by the demodulator, the decoding is called *hard-decision decoding*.

The demodulator can also be configured to feed the decoder with a *quantized value* of *z(T) greater than two levels*, or with an unquantized or analog value of *z(T)*. Such an implementation furnishes the decoder with more information than is provided in the hard-decision case. When the quantization level of the demodulator output is greater than two, the decoding is called *soft-decision decoding*. Eight levels (3-bits) of quantization are illustrated on the abscissa of Figure 6.8. When the demodulator sends a hard binary decision to the decoder, it sends it a single binary symbol. When the demodulator sends a soft binary decision, quantized to eight levels, it sends the decoder a 3-bit word describing an interval along *z(T)*. In effect, sending such a 3-bit word in place of a single binary symbol is equivalent to sending the decoder a *measure of confidence* along with the code symbol. Referring to Figure 6.8, if the demodulator sends 1 1 1 to the decoder, this is tantamount to declaring the code symbol to be a one with very high confidence, while sending a 1 0 0 is tantamount to declaring the code symbol to be a one with very low confidence. It should be clear that ultimately, every message decision out of the decoder must be a hard decision; otherwise, one might see computer printouts that read: "think it's a 1," "think it's a 0," and so on. The idea behind the demodulator *not making hard decisions* and sending more data (soft decisions) to the decoder can be thought of as an interim step to provide the decoder with more information, which the decoder then uses for recovering the message sequence (with better error performance than it could in the case of hard-decision decoding).

For a Gaussian channel, eight-level quantization results in a performance improvement of approximately 2 dB in required signal-to-noise ratio compared to two-level quantization. This means that eight-level soft-decision decoding can provide the same probability of bit error as that of hard-decision decoding, but

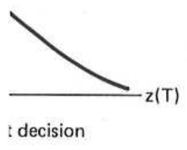
requires 2 dI
quantization)
tization; ther
dB compared
than eight le
paid for such
decision dec
eight-level qu
symbol; ther
decoding prc
in required n
Block c
devised to o
is generally
than hard-de
decoding is
decoding, sc

6.3.2.1

A bina
Section 5.3.
probabilities

as illustrate
the input sy
to the input
which mear
demodulatc
symbol, z_{ji}
indexing of
demodulat

Tra
s



ıade in Chapters
by the demod-

vith a *quantized*
analog value of
ıformation than
ı of the demod-
ision decoding.
a of Figure 6.8.
ler, it sends it a
decision, quan-
n interval along
inary symbol is
g with the code
to the decoder,
very high con-
le symbol to be
, every message
, one might see
and so on. The
ıding more data
p to provide the
· recovering the
he case of hard-

a performance
ratio compared
›n decoding can
n decoding, but

requires 2 dB *less* E_b/N_0 for the same performance. Analog (or infinite-level quantization) results in a 2.2-dB performance improvement over two-level quantization; therefore, *eight-level quantization* results in a loss of approximately 0.2 dB compared to infinitely fine quantization. For this reason, quantization to more than eight levels can yield little performance improvement [12]. What price is paid for such improved soft-decision-decoder performance? In the case of hard-decision decoding, a single bit is used to describe each code symbol, while for eight-level quantized soft-decision decoding 3 bits are used to describe each code symbol; therefore, three times the amount of data must be handled during the decoding process. Hence the price paid for soft-decision decoding is an increase in required memory size at the decoder (and possibly a speed penalty).

Block decoding algorithms and convolutional decoding algorithms have been devised to operate with hard *or* soft decisions. However, soft-decision decoding is generally not used with block codes because it is considerably more difficult than hard-decision decoding to implement. The most prevalent use of soft-decision decoding is with the *Viterbi convolutional decoding algorithm*, since with Viterbi decoding, soft decisions represent only a trivial increase in computation.

6.3.2.1 Binary Symmetric Channel

A binary symmetric channel (BSC) is a discrete memoryless channel (see Section 5.3.1) that has binary input and output alphabets and symmetric transition probabilities. It can be described by the conditional probabilities

$$\begin{aligned} P(0|1) &= P(1|0) = p \\ P(1|1) &= P(0|0) = 1 - p \end{aligned} \tag{6.5}$$

as illustrated in Figure 6.9. The probability that an output symbol will differ from the input symbol is p, and the probability that the output symbol will be identical to the input symbol is $(1 - p)$. The BSC is an example of a *hard-decision channel*, which means that, even though continuous-valued signals may be received by the demodulator, a BSC allows only firm decisions such that each demodulator output symbol, z_{ji}, as shown in Figure 6.1, consists of one of two binary values. The indexing of z_{ji} pertains to the jth code symbol of the ith branch word, Z_i. The demodulator then feeds the sequence $\mathbf{Z} = \{Z_i\}$ to the decoder.

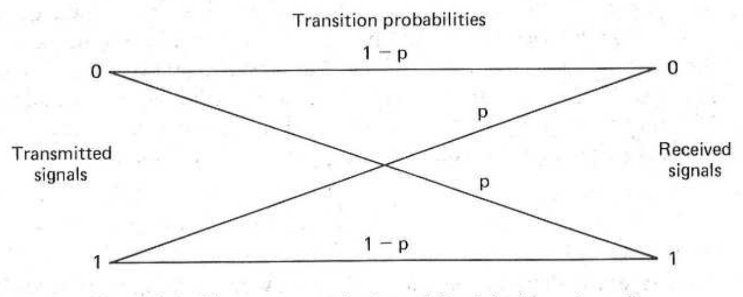


Figure 6.9 Binary symmetric channel (hard-decision channel).

Let $\mathbf{U}^{(m)}$ be a transmitted codeword over a BSC with symbol error probability p, and let $\mathbf{Z}$ be the corresponding received decoder sequence. As noted previously, a maximum likelihood decoder chooses the codeword $\mathbf{U}^{(m')}$ which maximizes the likelihood, $P(\mathbf{Z}|\mathbf{U}^{(m)})$, or its logarithm. For a BSC, this is equivalent to choosing the codeword, $\mathbf{U}^{(m')}$, that is closest in *Hamming distance* to $\mathbf{Z}$ [8]. Thus Hamming distance is an appropriate metric to describe the distance or closeness of fit between $\mathbf{U}^{(m)}$ and $\mathbf{Z}$. From all the possible transmitted sequences, $\mathbf{U}^{(m)}$, the decoder chooses the $\mathbf{U}^{(m')}$ sequence for which the distance to $\mathbf{Z}$ is minimum.

Suppose that $\mathbf{U}^{(m)}$ and $\mathbf{Z}$ are each L-bit-long sequences and that they differ in d_m positions [i.e., the Hamming distance between $\mathbf{U}^{(m)}$ and $\mathbf{Z}$ is d_m]. Then, since the channel is assumed memoryless, the probability that this $\mathbf{U}^{(m)}$ was transformed to the specific received $\mathbf{Z}$ at distance d_m from it can be written

$$P(\mathbf{Z}|\mathbf{U}^{(m)}) = p^{d_m}(1-p)^{L-d_m} \tag{6.6}$$

and the log-likelihood function is

$$\log P(\mathbf{Z}|\mathbf{U}^{(m)}) = -d_m \log\left(\frac{1-p}{p}\right) + L\log(1-p) \tag{6.7}$$

If we compute this quantity for each possible transmitted sequence, the second term will be constant in each case. Assuming that $p < 0.5$, we can express Equation (6.7) as

$$\log P(\mathbf{Z}|\mathbf{U}^{(m)}) = -Ad_m - B \tag{6.8}$$

where A and B are positive constants. Therefore, choosing the codeword $\mathbf{U}^{(m')}$ such that the Hamming distance, d_m, to the received sequence $\mathbf{Z}$ is minimized corresponds to *maximizing the likelihood or log-likelihood metric*. Consequently, over a BSC, the log-likelihood metric is conveniently replaced by the Hamming distance, and a maximum likelihood decoder will choose, in the tree or trellis diagram, the path whose corresponding sequence, $\mathbf{U}^{(m')}$, is at the *minimum Hamming distance* to the received sequence $\mathbf{Z}$.

6.3.2.2 Gaussian Channel

For a Gaussian channel, each demodulator output symbol, z_{ji}, as shown in Figure 6.1, is a value from a continuous alphabet. The symbol z_{ji} cannot be labeled as a correct or incorrect detection decision. Sending the decoder such soft decisions can be viewed as sending a family of conditional probabilities of the different symbols (see Section 5.3.1). It can be shown [8] that maximizing $P(\mathbf{Z}\,|\,\mathbf{U}^{(m)})$ is equivalent to maximizing the inner product between the codeword sequence, $\mathbf{U}^{(m)}$ (consisting of binary symbols), and the analog-valued received sequence, $\mathbf{Z}$. Thus the decoder chooses the codeword $\mathbf{U}^{(m')}$ if it maximizes

$$\sum_{i=1}^{\infty}\sum_{j=1}^{n} z_{ji}u_{ji}^{(m)} \tag{6.9}$$

This is equivalent to choosing the codeword $\mathbf{U}^{(m')}$ that is closest in *Euclidean distance* to $\mathbf{Z}$. Even though the hard- and soft-decision channels require different

metrics, the c
sequence, Z, i
tion (6.9) exa
arithmetic op
plemented dig
Does Equatio
Equation (6.9
$r(t)$, with a re
tized Gaussia
channel mode

6.3.3 The Vi

The Viterbi
1967. The Vi
however, it
structure in t
brute-force d
of the numb
calculating a
time t_i, and a
removes fro
dates for the
the one havi
This selectic
continues in
eliminating t
the decoding
gorithm is, i
timum path
maximum li
tance metric

6.3.4 An E

For simplic
measure. T
trellis diagr
the decode
cedure can
concert wit
to label ea
received c
trellis. The
sponding c
11 01 01 1

ol error proba-
ence. As noted
ord $\mathbf{U}^{(m')}$ which
his is equivalent
stance to $\mathbf{Z}$ [8].
stance or close-
equences, $\mathbf{U}^{(m)}$,
$\mathbf{Z}$ is minimum.
that they differ
Z is d_m]. Then,
$\mathbf{U}^{(m)}$ was trans-
vritten

(6.6)

(6.7)

nce, the second
n express Equa-

(6.8)

codeword $\mathbf{U}^{(m')}$
$\mathbf{Z}$ is minimized
. Consequently,
y the Hamming
e tree or trellis
minimum Ham-

z_{ji}, as shown in
annot be labeled
er such soft de-
lities of the dif-
izing $P(\mathbf{Z}|\mathbf{U}^{(m)})$
word sequence,
ed sequence, $\mathbf{Z}$.

(6.9)

st in *Euclidean*
equire different

metrics, the concept of choosing the codeword $\mathbf{U}^{(m')}$ that is closest to the received sequence, $\mathbf{Z}$, is the same in both cases. To implement the maximization of Equation (6.9) exactly, the decoder would have to be able to handle analog-valued arithmetic operations. This is impractical because the decoder is generally implemented digitally. Thus it is necessary to quantize the received symbols z_{ji}. Does Equation (6.9) remind you of the demodulation treatment in Chapter 3? Equation (6.9) is the discrete version of correlating an input received waveform, $r(t)$, with a reference waveform, $s_i(t)$, as expressed in Equation (3.34). The quantized Gaussian channel, typically referred to as a *soft-decision channel*, is the channel model assumed for the soft-decision decoding described earlier.

6.3.3 The Viterbi Convolutional Decoding Algorithm

The Viterbi decoding algorithm was discovered and analyzed by Viterbi [13] in 1967. The Viterbi algorithm essentially performs maximum likelihood decoding; however, it reduces the computational load by taking advantage of the special structure in the code trellis. The advantage of Viterbi decoding, compared with brute-force decoding, is that the complexity of a Viterbi decoder is not a function of the number of symbols in the codeword sequence. The algorithm involves calculating a *measure of similarity, or distance*, between the received signal, at time t_i, and all the trellis paths entering each state at time t_i. The Viterbi algorithm removes from consideration those trellis paths that could not possibly be candidates for the maximum likelihood choice. When two paths enter the same state, the one having the best metric is chosen; this path is called the *surviving path*. This selection of surviving paths is performed for all the states. The decoder continues in this way to advance deeper into the trellis, making decisions by eliminating the least likely paths. The early rejection of the unlikely paths reduces the decoding complexity. In 1969, Omura [14] demonstrated that the Viterbi algorithm is, in fact, maximum likelihood. Note that the goal of selecting the optimum path can be expressed, equivalently, as choosing the codeword with the *maximum likelihood metric*, or as choosing the codeword with the *minimum distance metric*.

6.3.4 An Example of Viterbi Convolutional Decoding

For simplicity, a BSC is assumed; thus Hamming distance is a proper distance measure. The encoder for this example is shown in Figure 6.3, and the encoder trellis diagram is shown in Figure 6.7. A similar trellis can be used to represent the decoder, as shown in Figure 6.10. The basic idea behind the decoding procedure can best be understood by examining the Figure 6.7 encoder trellis in concert with the Figure 6.10 decoder trellis. For the decoder trellis it is convenient to label each trellis branch at time t_i with the *Hamming distance* between the received code symbols and the corresponding branch word from the encoder trellis. The example in Figure 6.10, shows a message sequence, $\mathbf{m}$, the corresponding codeword sequence, $\mathbf{U}$, and a noise corrupted received sequence, $\mathbf{Z}$ = 11 01 01 10 01 . . . . The branch words seen on the *encoder trellis* branches

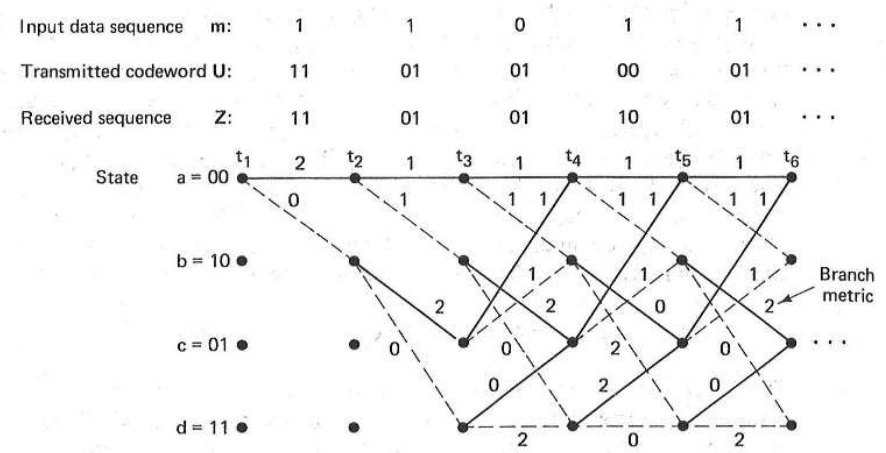

Input data sequence m: 1 1 0 1 1 . . .
Transmitted codeword U: 11 01 01 00 01 . . .
Received sequence Z: 11 01 01 10 01 . . .
State
a = 00
b = 10
c = 01
d = 11
t1 t2 t3 t4 t5 t6
Branch metric

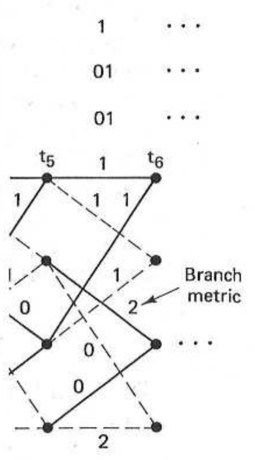
Figure 6.10 Decoder trellis diagram (rate $\frac{1}{2}$, $K = 3$).

characterize the encoder in Figure 6.3, and are known a priori to both the encoder and the decoder. These encoder branch words are the code symbols that would be expected to come from the encoder output as a result of each of the state transitions. The labels on the *decoder trellis* branches are accumulated by the decoder *on the fly*. That is, as the code symbols are received, each branch of the decoder trellis is labeled with a metric of similarity (Hamming distance) between the received code symbols and each of the branch words for that time interval. From the received sequence, **Z**, in Figure 6.10, we see that the code symbols received at time t_1 are 11. In order to label the decoder branches at time t_1 with the appropriate Hamming distance metric, we look at the Figure 6.7 encoder trellis. Here we see that a state $00 \rightarrow 00$ transition yields an output branch word of 00. But we received 11. Therefore, on the decoder trellis we label the state $00 \rightarrow 00$ transition with the Hamming distance between them, namely 2. Looking at the encoder trellis again, we see that a state $00 \rightarrow 10$ transition yields an output branch word of 11, which corresponds exactly with the code symbols we received at time t_1. Therefore, on the decoder trellis, we label the state $00 \rightarrow 10$ transition with a Hamming distance of 0. We continue labeling the decoder trellis branches in this way as the symbols are received at each time t_i. The decoding algorithm uses these Hamming distance metrics to find the *most likely* (minimum distance) path through the trellis.

The basis of *Viterbi decoding* is the following observation: If any two paths in the trellis merge to a single state, one of them can always be eliminated in the search for an optimum path. For example, Figure 6.11 shows two paths merging at time t_5 to state 00. Let us define the *cumulative Hamming path metric* of a given path at time t_i as the sum of the branch Hamming distance metrics along that path up to time t_i. In Figure 6.11 the upper path has metric 4; the lower has metric 1. The upper path cannot be a portion of the optimum path because the lower path, which enters the same state, has a lower metric. This observation

holds because of
marizes the enc
states or future
At each tin
length, and eacl
consists of com
inating one of t
t_i; then the dec
steps in our dec
input data sequ
Figure 6.10. As
(This assumptic
time t_1 the rec
sitions are to st
has branch met
are two possib
cumulative pa
responding to
branches diver
state at time t
nated, namely
of the two en
by using an a
6.12d. At this
between time
transition whi
produced by
Here we can
drawn the la
input ones.
putation had

State a



= 3).

ori to both the encoder
de symbols that would
lt of each of the state
e accumulated by the
ed, each branch of the
ning distance) between
for that time interval.
that the code symbols
ranches at time t_1 with
he Figure 6.7 encoder
an output branch word
is we label the state 00
, namely 2. Looking at
sition yields an output
le symbols we received
tate 00 → 10 transition
ecoder trellis branches
he decoding algorithm
ely (minimum distance)

ation: If any two paths
ys be eliminated in the
ows two paths merging
ming path metric of a
distance metrics along
metric 4; the lower has
mum path because the
etric. This observation

holds because of the Markov nature of the encoder state: The present state summarizes the encoder history in the sense that previous states cannot affect future states or future output branches.

At each time t_i there are 2^{K-1} states in the trellis, where K is the constraint length, and each state can be *entered by means of two paths*. Viterbi decoding consists of computing the metrics for the two paths entering each state and *eliminating one of them*. This computation is done for each of the 2^{K-1} nodes at time t_i; then the decoder moves to time t_{i+1} and repeats the process. The first few steps in our decoding example are as follows (see Figure 6.12). Assume that the input data sequence **m**, codeword **U**, and received sequence **Z** are as shown in Figure 6.10. Assume that the decoder knows the correct initial state of the trellis. (This assumption is not necessary in practice, but simplifies the explanation.) At time t_1 the received code symbols are 11. From state 00 the only possible transitions are to state 00 or state 10, as shown in Figure 6.12a. State 00 → 00 transition has branch metric 2; state 00 → 10 transition has branch metric 0. At time t_2 there are two possible branches leaving each state, as shown in Figure 6.12b. The cumulative path metrics of these branches are labeled λ_a, λ_b, λ_c, and λ_d, corresponding to the terminating state. At time t_3 in Figure 6.12c there are again two branches diverging from each state. As a result, there are two paths entering each state at time t_4. As noted previously, one path entering each state can be eliminated, namely, the one having the larger cumulative path metric. Should metrics of the two entering paths be of equal value, one path is chosen for elimination by using an arbitrary rule. The surviving path into each state is shown in Figure 6.12d. At this point in the decoding process, there is only a single surviving path between times t_1 and t_2. Therefore, the decoder can now decide that the state transition which occurred between t_1 and t_2 was 00 → 10. Since this transition is produced by an input bit one, the decoder outputs a one as the first decoded bit. Here we can see how the decoding of the surviving branch is facilitated by having drawn the lattice branches with solid lines for input zeros and dashed lines for input ones. Note that the first bit was not decoded until the path metric computation had proceeded to a much greater depth into the trellis. For a typical

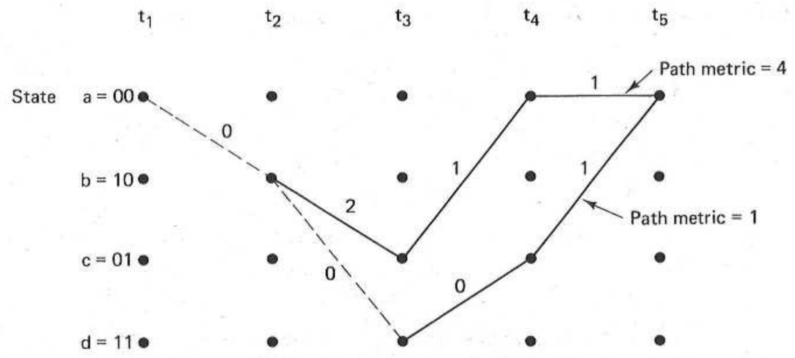


Figure 6.11 Path metrics for two merging paths.

decoder implementation, this represents a decoding delay which can be as much as five times the constraint length in bits.

At each succeeding step in the decoding process, there will always be two possible paths entering each state; one of the two will be eliminated by comparing the path metrics. Figure 6.12e shows the next step in the decoding process. Again, at time t_5 there are two paths entering each state, and one of each pair can be eliminated. Figure 6.12f shows the survivors at time t_5. Notice that in our example we cannot yet make a decision on the second input data bit because there still are two paths leaving the state 10 node at time t_2. At time t_6 in Figure 6.12g we again see the pattern of remerging paths, and in Figure 6.12h we see the survivors at time t_6. Also, in Figure 6.12h the decoder outputs one as the second decoded bit, corresponding to the single surviving path between t_2 and t_3. The decoder continues in this way to advance deeper into the trellis and to make decisions on the input data bits by eliminating all paths but one.

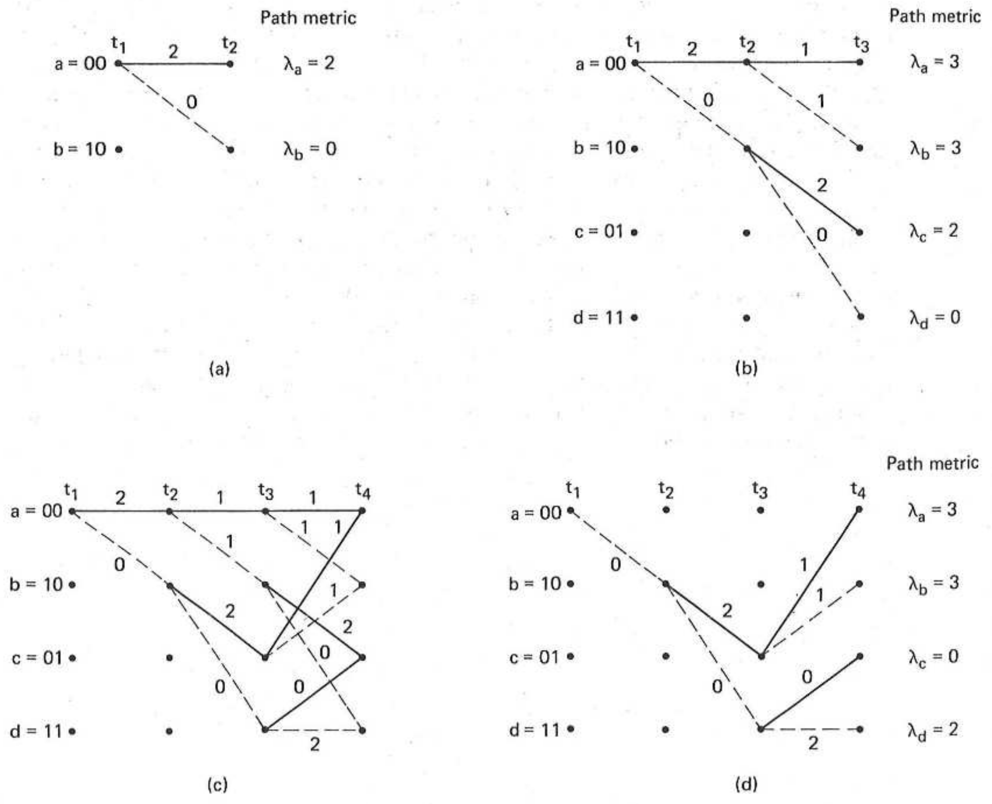
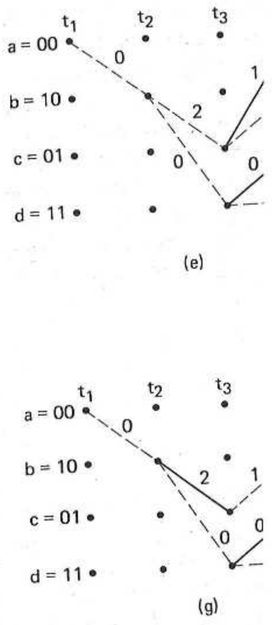


Figure 6.12 Selection of survivor paths. (a) Survivors at t_2. (b) Survivors at t_3. (c) Metric comparisons at t_4. (d) Survivors at t_4. (e) Metric comparisons at t_5. (f) Survivors at t_5. (g) Metric comparisons at t_6. (h) Survivors at t_6.



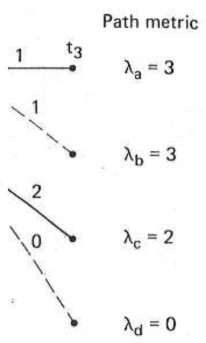
6.3.5 Path M

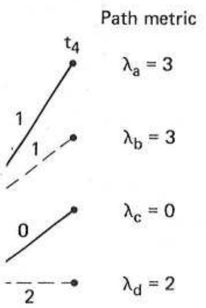
The storage
straint lengtl
after each de
disjoint very
tend to have
Thus if the c
on all paths
a *fixed amo*
time it steps
u, is [12]

where h is
which mini
decoder ou
onstrated [
for near-op
limitation

n can be as much

ill always be two
ted by comparing
g process. Again,
each pair can be
at in our example
ecause there still
Figure 6.12g we
see the survivors
second decoded
t_3. The decoder
make decisions on





s at t_3. (c) Metric
urvivors at t_5. (g)

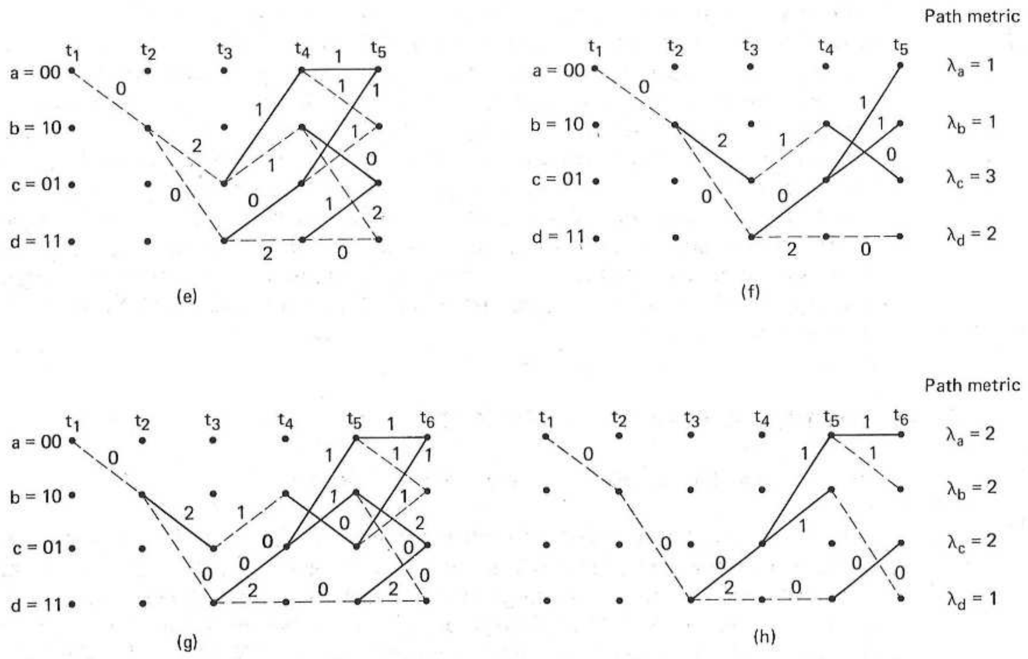


Figure 6.12 (*Continued*)

6.3.5 Path Memory and Synchronization

The storage requirements of the Viterbi decoder grow exponentially with constraint length K. For a code with rate $1/n$, the decoder retains a set of 2^{K-1} paths after each decoding step. With high probability, these paths will not be mutually disjoint very far back from the present decoding depth [12]. All of the 2^{K-1} paths tend to have a common stem which eventually branches to the various states. Thus if the decoder stores enough of the history of the 2^{K-1} paths, the oldest bits on all paths will be the same. A simple decoder implementation, then, contains a *fixed amount of path history* and outputs the oldest bit on an arbitrary path each time it steps one level deeper into the trellis. The amount of path storage required, u, is [12]

$$u = h2^{K-1} \tag{6.10}$$

where h is the length of the information bit path history per state. A refinement, which minimizes the value of h, uses the oldest bit on the most likely path as the decoder output, instead of the oldest bit on an arbitrary path. It has been demonstrated [12] that a value of h of 4 or 5 times the code constraint length is sufficient for near-optimum decoder performance. The storage requirement, u, is the basic limitation on the implementation of Viterbi decoders. The current state of the art limits decoders to a constraint length of about $K = 10$. Efforts to increase coding gain by further increasing constraint length are met by the exponential increase in memory requirements (and complexity) that follows from Equation (6.10).

Branch word synchronization is the process of determining the beginning of a branch word in the received sequence. Such synchronization can take place without new information being added to the transmitted symbol stream because the received data appear to have an excessive error rate when not synchronized. Therefore, a simple way of accomplishing synchronization is to monitor some concomitant indication of this large error rate, that is, the rate at which the path metrics are increasing or the rate at which the surviving paths in the trellis merge. The monitored parameters are compared to a threshold, and synchronization is then adjusted accordingly.

6.4 PROPERTIES OF CONVOLUTIONAL CODES

6.4.1 Distance Properties of Convolutional Codes

Let us consider the distance properties of convolutional codes in the context of our simple encoder in Figure 6.3 and its trellis diagram in Figure 6.7. We want to evaluate the distance between all possible pairs of codeword sequences. As in the case of block codes (see Section 5.5.2), we are interested in the *minimum distance* between all pairs of such codeword sequences in the code, since the minimum distance is related to the error-correcting capability of the code. Because a convolutional code is a group or *linear code* [6], there is no loss in generality in simply finding the minimum distance between each of the codeword sequences and the all-zeros sequence. Assuming that the all-zeros input sequence was transmitted, the paths of interest are those that start and end in the 00 state and do not return to the 00 state anywhere in between. An error will occur whenever the distance of any other path that merges with the $a = 00$ state at time t_i is less than that of the all-zeros path up to time t_i, causing the all-zeros path to be discarded in the decoding process. In other words, given the all-zeros transmission, an error occurs whenever the *all-zeros path does not survive*. The minimum distance for making such an error can be found by exhaustively examining every path from the 00 state to the 00 state. First, let us redraw the trellis diagram, shown in Figure 6.13, labeling each branch with its Hamming distance from the all-zeros codeword instead of with its branch word symbols. The Hamming distance between two unequal-length sequences will be found by first appending the necessary number of zeros to the shorter sequence to make the two sequences equal in length. Consider all the paths that diverge from the all-zeros path and then remerge for the first time at some arbitrary node. From Figure 6.13 we can compute the distances of these paths from the all-zeros path. There is one path at distance 5 from the all-zeros path; this path departs from the all-zeros path at time t_1 and merges with it at time t_4. Similarly, there are two paths at distance 6, one which departs at time t_1 and merges at time t_5, and the other which departs at time t_1 and merges at time t_6, and so on. We can also see from the dashed and solid lines

ts to increase coding
exponential increase
ı Equation (6.10).
ning the beginning of
ation can take place
mbol stream because
en not synchronized.
ı is to monitor some
ate at which the path
ıs in the trellis merge.
ıd synchronization is

ꞁdes in the context of
Figure 6.7. We want
ꞁord sequences. As in
sted in the *minimum*
n the code, since the
y of the code. Because
no loss in generality
codeword sequences
ıt sequence was trans-
n the 00 state and do
ll occur whenever the
e at time t_i is less than
s path to be discarded
transmission, an error
minimum distance for
ining every path from
gram, shown in Figure
he all-zeros codeword
distance between two
the necessary number
nces equal in length.
and then remerge for
we can compute the
one path at distance 5
os path at time t_1 and
distance 6, one which
ıich departs at time t_1
dashed and solid lines

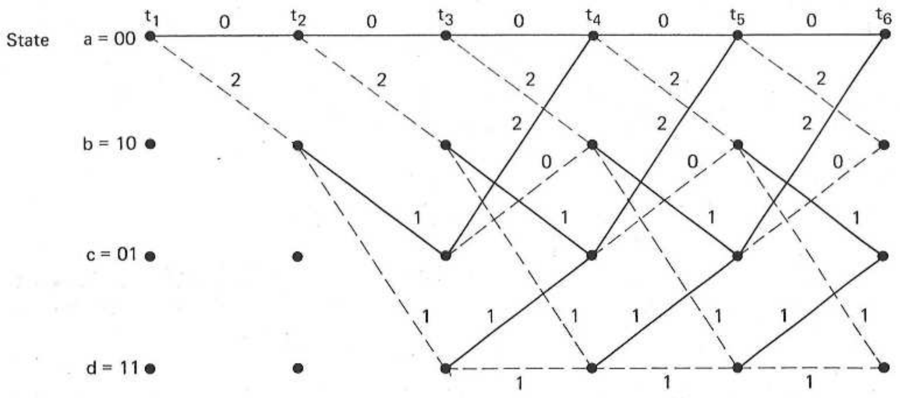


Figure 6.13 Trellis diagram, labeled with distances from the all-zeros path.

of the diagram that the input bits for the distance 5 path are 1 0 0; it differs in only one input bit from the all-zeros input sequence. Similarly, the input bits for the distance 6 paths are 1 1 0 0 and 1 0 1 0 0; each differs in two positions from the all-zeros path. The minimum distance in the set of all arbitrarily long paths that diverge and remerge, called the *minimum free distance* or simply the *free distance*, is seen to be 5 in this example. For calculating the error-correcting capability of the code, we repeat Equation (5.44) with the minimum distance, $d_{\min}$, replaced by the free distance, d_f.

$$t = \left\lfloor \frac{d_f - 1}{2} \right\rfloor \tag{6.11}$$

where $\lfloor x \rfloor$ means the largest integer no greater than x. Setting $d_f = 5$, we see that the code, characterized by the Figure 6.3 encoder, can correct any two channel errors.

Although Figure 6.13 presents the computation of free distance in a straightforward way, a more direct closed-form expression can be obtained by starting with the state diagram in Figure 6.5. First, we label the branches of the state diagram as either $D^0 = 1$, D^1, or D^2, shown in Figure 6.14, where the exponent of D denotes the Hamming distance from the branch word of that branch to the all-zeros branch. The self-loop at node a can be eliminated since it contributes nothing to the distance properties of a codeword sequence relative to the all-zeros sequence. Furthermore, node a can be split into two nodes (labeled a and e), one of which represents the input and the other the output of the state diagram. All paths originating at $a = 00$ and terminating at $e = 00$ can be traced on the modified state diagram of Figure 6.14. We can calculate the transfer function of path $a\ b\ c\ e$ (starting and ending at state 00) in terms of the indeterminate "placeholder" D, as $D^2\ D\ D^2 = D^5$. The exponent of D represents the cumulative tally of the number of ones in the path, and hence the Hamming distance from the all-zeros path. Similarly, the paths $a\ b\ d\ c\ e$ and $a\ b\ c\ b\ c\ e$ each have the transfer

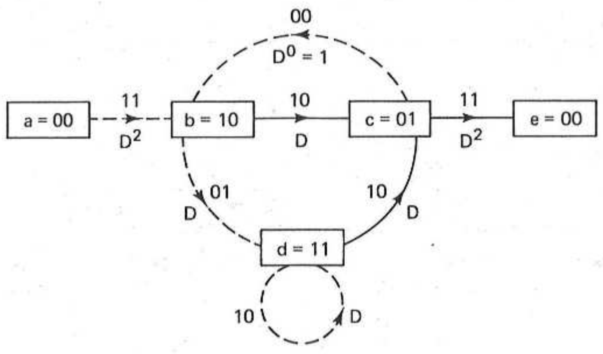


Figure 6.14 State diagram, labeled according to distance from the all-zeros path.

function D^6 and thus a Hamming distance of 6 from the all-zeros path. We now write the following state equations:

$$\begin{aligned} X_b &= D^2X_a + X_c \\ X_c &= DX_b + DX_d \\ X_d &= DX_b + DX_d \\ X_e &= D^2X_c \end{aligned} \tag{6.12}$$

where $X_a, \ldots, X_e$ are dummy variables for the partial paths to the intermediate nodes. The *transfer function*, $T(D)$, sometimes called the *generating function* of the code can be expressed as $T(D) = X_e/X_a$. By solving the state equations shown in Equation (6.12), we obtain [15, 16]

$$\begin{aligned} T(D) &= \frac{D^5}{1 - 2D} \\ &= D^5 + 2D^6 + 4D^7 + \cdots + 2^\ell D^{\ell+5} + \cdots \end{aligned} \tag{6.13}$$

The transfer function for this code indicates that there is a single path of distance 5 from the all-zeros path, two of distance 6, four of distance 7, and in general, there are 2^ℓ paths of distance $\ell + 5$ from the all-zeros path, where $\ell = 0, 1, 2, \ldots$. The free distance d_f of the code is the Hamming weight of the lowest-order term in the expansion of $T(D)$. In this example $d_f = 5$. In evaluating distance properties, the transfer function, $T(D)$, cannot be used for long constraint lengths since the complexity of $T(D)$ increases exponentially with constraint length.

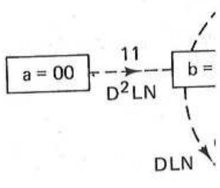
The transfer function can be used to provide more detailed information than just the distance of the various paths. Let us introduce a factor L into each branch of the state diagram so that the exponent of L can serve as a counter to indicate the number of branches in any given path from state $a = 00$ to state $e = 00$. Furthermore, we can introduce a factor N into all branch transitions caused by the input bit one. Thus, as each branch is traversed, the cumulative exponent on N increases by one, only if that branch transition is due to an input bit one. For the convolutional code characterized in our Figure 6.3 example, the additional

factors L and
now modify]

The transfer

$T(D$

Thus we ca
is one path
zeros path.
length 5, an
distance 7 p
all four pat
the all-zero
causes us to
be made.



6.4.1

In th
capability,
imum like
ever, whe

tate diagram, labeled
istance from the all-zeros

-zeros path. We now

(6.12)

ıs to the intermediate
enerating function of
tate equations shown

(6.13)

$+ \cdots$

ingle path of distance
ce 7, and in general,
ıth, where $\ell = 0, 1,$
ht of the lowest-order
ı evaluating distance
ng constraint lengths
constraint length.
ıiled information than
or L into each branch
a counter to indicate
00 to state $e = 00$.
transitions caused by
nulative exponent on
an input bit one. For
ımple, the additional

factors L and N are shown on the modified state diagram of Figure 6.15. We can now modify Equations (6.12) as follows:

$$\begin{aligned}
X_b &= D^2LNX_a + LNX_c \\
X_c &= DLX_b + DLX_d \\
X_d &= DLNX_b + DLNX_d \\
X_e &= D^2LX_c
\end{aligned} \tag{6.14}$$

The transfer function of this augmented state diagram is

$$\begin{aligned}
T(D, L, N) &= \frac{D^5L^3N}{1 - DL(1 + L)N} \\
&= D^5L^3N + D^6L^4(1 + L)N^2 + D^7L^5(1 + L)^2N^3 \\
&\quad + \cdots + D^{\ell+5}L^{\ell+3}N^{\ell+1} + \cdots
\end{aligned} \tag{6.15}$$

Thus we can verify some of the path properties displayed in Figure 6.13. There is one path of distance 5, length 3, which differs in one input bit from the all-zeros path. There are two paths of distance 6, one of which is length 4, the other length 5, and both differ in two input bits from the all-zeros path. Also, of the distance 7 paths, one is of length 5, two are of length 6, and one is of length 7; all four paths correspond to input sequences that differ in three input bits from the all-zeros path. Thus if the all-zeros path is the correct path and the noise causes us to choose one of the incorrect paths of distance 7, three bit errors will be made.

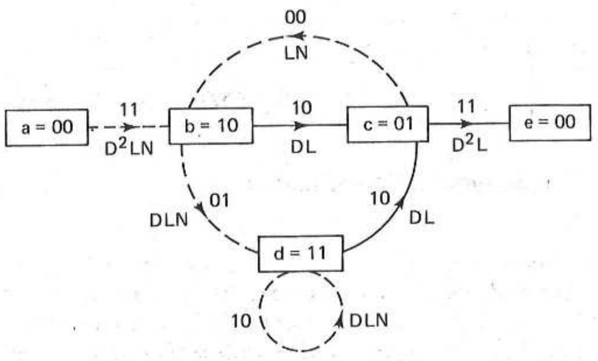


Figure 6.15 State diagram, labeled according to distance, length, and number of input ones.

6.4.1.1 Error-Correcting Capability of Convolutional Codes

In the study of block codes in Chapter 5, we saw that the error-correcting capability, t, represented the number of code symbol errors that could, with maximum likelihood decoding, be corrected in each block length of the code. However, when decoding convolutional codes, the error-correcting capability cannot be stated so succinctly. With regard to Equation (6.11), we can say that the code can, with maximum likelihood decoding, correct t errors within a few constraint lengths, where "few" here means 3 to 5. The exact length depends on how the errors are distributed. For a particular code and error pattern, the length can be bounded using transfer function methods. A computer program for convolutional decoding with the Viterbi algorithm, called VITALG, is provided in Appendix E. The interested reader can use this tool for verifying the capability of Viterbi decoding of convolutional codes with various choices of code generators, code rates, constraint lengths, and path memory lengths.

6.4.2 Systematic and Nonsystematic Convolutional Codes

A *systematic* convolutional code is one in which the input k-tuple appears as part of the output branch word n-tuple associated with that k-tuple. Figure 6.16 shows a binary, rate $\frac{1}{2}$, $K = 3$ systematic encoder. For linear block codes, any nonsystematic code can be transformed into a systematic code with the same block distance properties. This is not the case for convolutional codes. The reason for this is that convolutional codes depend largely on *free distance*; making the convolutional code systematic, in general, *reduces* the maximum possible free distance for a given constraint length and rate.

Table 6.1 shows the maximum free distance for rate $\frac{1}{2}$ systematic and nonsystematic codes for $K = 2$ through 8. For large constraint lengths the results are even more widely separated [17].

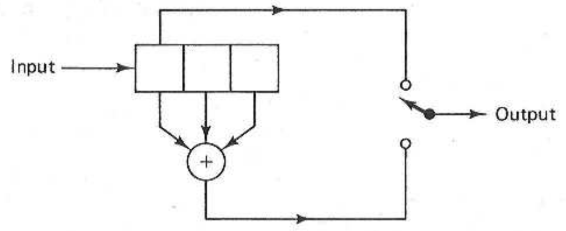


Figure 6.16 Systematic convolutional encoder, rate $\frac{1}{2}$, $K = 3$.

6.4.3 Catastrophic Error Propagation in Convolutional Codes

A *catastrophic error* is defined as an event whereby a finite number of code symbol errors cause an infinite number of decoded data bit errors. Massey and Sain [18] have derived a necessary and sufficient condition for convolutional codes to display catastrophic error propagation. For rate $1/n$ codes with register taps designated by polynomial generators, as described in Section 6.2.1, the condition for catastrophic error propagation is that the generators have a *common polynomial factor* (of degree at least one). For example, Figure 6.17a illustrates a rate $\frac{1}{2}$, $K = 3$ encoder with upper polynomial $g_1(X)$ and lower polynomial $g_2(X)$, as follows:

$$g_1(X) = 1 + X$$
$$g_2(X) = 1 + X^2 \tag{6.16}$$

an say that the code
hin a few constraint
lepends on how the
n, the length can be
ım for convolutional
ided in Appendix E.
apability of Viterbi
de generators, code

es

uple appears as part
:. Figure 6.16 shows
codes, any nonsys-
'ith the same block
ıdes. The reason for
ce; making the con-
m possible free dis-

systematic and non-
t lengths the results

ystematic convolutional
, $K = 3$.

mber of code symbol
Aassey and Sain [18]
utional codes to dis-
ı register taps desig-
.1, the condition for
common polynomial
llustrates a rate $\frac{1}{2}$, K
ial $g_2(X)$, as follows:

(6.16)

TABLE 6.1 Comparison of Systematic and Nonsystematic Free Distance, Rate $\frac{1}{2}$

Constraint length	Free distance systematic	Free distance nonsystematic
2	3	3
3	4	5
4	4	6
5	5	7
6	6	8
7	6	10
8	7	10

Source: A. J. Viterbi and J. K. Omura, *Principles of Digital Communication and Coding*, McGraw-Hill Book Company, New York, 1979, p. 251.

The generators $g_1(X)$ and $g_2(X)$ have in common the polynomial factor, $1 + X$, since

$$1 + X^2 = (1 + X)(1 + X)$$

Therefore, the encoder in Figure 6.17a can manifest *catastrophic error propagation*.

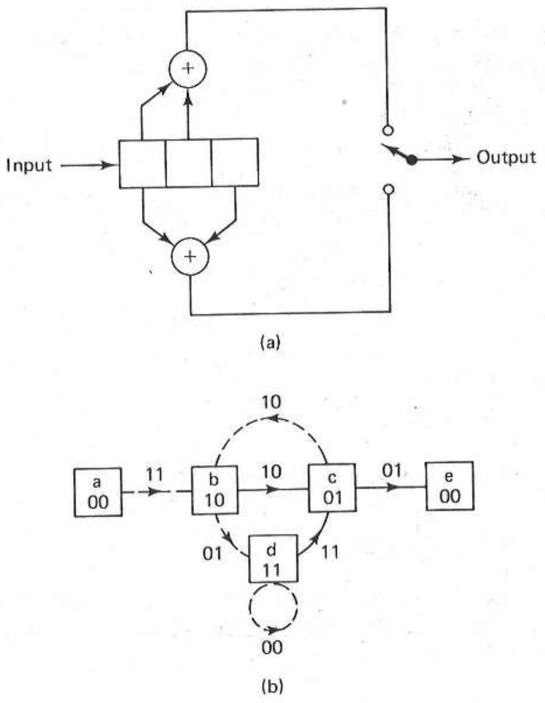


Figure 6.17 Encoder displaying catastrophic error propagation. (a) Encoder. (b) State diagram.

In terms of the state diagram for any-rate code, catastrophic errors can occur if, and only if, any closed-loop path in the diagram has zero weight (zero distance from the all-zeros path). To illustrate this, consider the example of Figure 6.17. The state diagram in Figure 6.17b is drawn with the state $a = 00$ node split into two nodes, a and e, as before. Assuming that the all-zeros path is the correct path, the incorrect path $a\ b\ d\ d\ \ldots\ d\ c\ e$ has exactly 6 ones, no matter how many times we go around the self-loop at node d. Thus for a BSC, for example, three channel errors may cause us to choose this incorrect path. An arbitrarily large number of errors (two plus the number of times the self-loop is traversed) can be made on such a path. We observe that for rate $1/n$ codes, if each adder in the encoder has an even number of connections, the self-loop corresponding to the all-ones data state will have zero weight, and consequently, *the code will be catastrophic*.

The only advantage of a systematic code, described earlier, is that it can never be catastrophic, since each closed loop must contain at least one branch generated by a nonzero input bit, and thus each closed loop must have a nonzero code symbol. However, it can be shown [19] that only a small fraction of non-systematic codes (excluding those where all adders have an even number of taps) are catastrophic.

6.4.4 Performance Bounds for Convolutional Codes

The probability of bit error, P_B, for a binary convolutional code using hard-decision decoding can be shown [8] to be upper bounded as follows:

$$P_B \leq \left. \frac{dT(D, N)}{dN} \right|_{N=1, D=2\sqrt{p(1-p)}} \tag{6.17}$$

where p is the probability of channel symbol error. For the example of Figure 6.3, $T(D, N)$ is obtained from $T(D, L, N)$ by setting $L = 1$ in Equation (6.15).

$$T(D, N) = \frac{D^5 N}{1 - 2DN} \tag{6.18}$$

and

$$\left. \frac{dT(D, N)}{dN} \right|_{N=1} = \frac{D^5}{(1 - 2D)^2} \tag{6.19}$$

Combining Equations (6.17) and (6.19), we can write

$$P_B \leq \frac{\{2[p(1-p)]^{1/2}\}^5}{\{1 - 4[p(1-p)]^{1/2}\}^2} \tag{6.20}$$

For coherent BPSK modulation over an additive white Gaussian noise (AWGN) channel, it can be shown [8] that the bit error probability is bounded by

$P_B \leq Q$

where

$E_c/N_0 = rE$

$E_b/N_0 =$ ra

$E_c/N_0 =$ ra

$r = k/$

and where $Q(x$
B.1. Therefore
coherent BPSK

P

6.4.5 Coding

Coding gain i
required E_b/N
an uncoded sy
6.2 lists an u
BPSK, for se
lengths varyin
The table illus

TABLE 6.2 (

K
3
4
5
6
7
8
9

Source: V. K
Satellite, Johi

c errors can occur
ght (zero distance
le of Figure 6.17.
00 node split into
ath is the correct
s, no matter how
SC, for example,
th. An arbitrarily
loop is traversed)
les, if each adder
op corresponding
itly, *the code will*

ier, is that it can
least one branch
st have a nonzero
l fraction of non-
n number of taps)

de using hard-de-
ws:

(6.17)

xample of Figure
Equation (6.15).

(6.18)

(6.19)

(6.20)

e Gaussian noise
ability is bounded

$$P_B \le Q\left(\sqrt{2d_f \frac{E_c}{N_0}}\right) \exp\left(d_f \frac{E_c}{N_0}\right) \frac{dT(D, N)}{dN}\Bigg|_{N=1, D=\exp(-E_c/N_0)} \qquad (6.21)$$

where

$E_c/N_0 = rE_b/N_0$

E_b/N_0 = ratio of information bit energy to noise power spectral density

E_c/N_0 = ratio of channel symbol energy to noise power spectral density

$r = k/n$ = rate of the code

and where $Q(x)$ is defined in Equations (2.42) and (2.43) and tabulated in Table B.1. Therefore, for the rate $\frac{1}{2}$ code with free distance $d_f = 5$, in conjunction with coherent BPSK and hard-decision decoding, we can write

$$\begin{aligned} P_B &\le Q\left(\sqrt{\frac{5E_b}{N_0}}\right) \exp\left(\frac{5E_b}{2N_0}\right) \frac{\exp(-5E_b/2N_0)}{[1 - 2\exp(-E_b/2N_0)]^2} \\ &\le \frac{Q(\sqrt{5E_b/N_0})}{[1 - 2\exp(-E_b/2N_0)]^2} \end{aligned} \qquad (6.22)$$

6.4.5 Coding Gain

Coding gain is defined as the reduction, usually expressed in decibels, in the required E_b/N_0 to achieve a specified error probability of the coded system over an uncoded system with the same modulation and channel characteristics. Table 6.2 lists an upper bound on the coding gains, compared to uncoded coherent BPSK, for several maximum free distance convolutional codes with constraint lengths varying from 3 to 9 over a Gaussian channel with hard-decision decoding. The table illustrates that it is possible to achieve significant coding gain even with

TABLE 6.2 Coding Gain Upper Bounds for Some Convolutional Codes

Rate $\frac{1}{2}$ codes			Rate $\frac{1}{3}$ codes		
K	d_f	Upper bound (dB)	K	d_f	Upper bound (dB)
3	5	3.97	3	8	4.26
4	6	4.76	4	10	5.23
5	7	5.43	5	12	6.02
6	8	6.00	6	13	6.37
7	10	6.99	7	15	6.99
8	10	6.99	8	16	7.27
9	12	7.78	9	18	7.78

Source: V. K. Bhargava, D. Haccoun, R. Matyas, and P. Nuspl, *Digital Communications by Satellite*, John Wiley & Sons, Inc., New York, 1981.

a simple convolutional code. The actual coding gain will vary with the required bit error probability [20].

Table 6.3 lists the measured coding gains, compared to uncoded coherent BPSK, achieved with hardware implementation or computer simulation over a Gaussian channel with soft-decision decoding [21]. The uncoded E_b/N_0 is given in the leftmost column. From Table 6.3 we can see that coding gain increases as the bit error probability is decreased. However, the coding gain cannot increase indefinitely; it has an upper bound as shown in the table. This bound in decibels can be shown [21] to be

$$\text{coding gain} \leq 10 \log_{10} (rd_f) \tag{6.23}$$

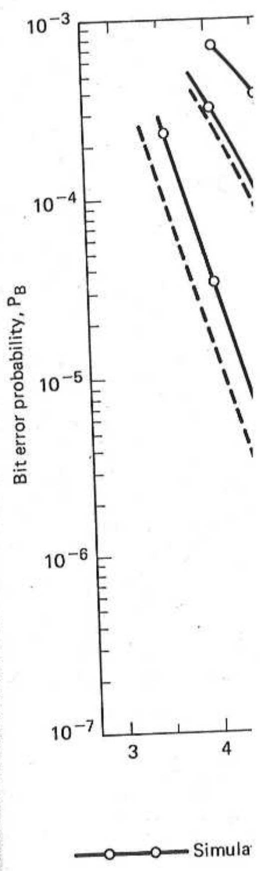
where r is the code rate and d_f is the free distance. Examination of Table 6.3 also reveals that at $P_B = 10^{-7}$, for code rates of $\frac{1}{2}$ and $\frac{2}{3}$, the weaker codes tend to be closer to the upper bound than are the more powerful codes.

Typically, Viterbi decoding is used over binary input channels with either hard or 3-bit soft quantized outputs. The constraint lengths vary between 3 and 9, the code rate is rarely smaller than $\frac{1}{3}$, and the path memory is usually a few constraint lengths [12]. The path memory refers to the depth of the input bit history stored by the decoder. From the Viterbi decoding example in Section 6.3.4, one might question the notion of a fixed path memory. It seems from the example that the decoding of a branch word, at any arbitrary node, can take place as soon as there is only a single surviving branch at that node. That is true; however, to actually implement the decoder in this way would entail an extensive amount of processing to continually check when the branch word can be decoded. Instead, *a fixed delay is provided*, after which the branch word is decoded. It has been shown [12, 22] that a fixed amount of path history, namely 4 or 5 times the constraint length, is sufficient to limit the degradation from the optimum decoder performance to about 0.1 dB for the BSC and Gaussian channels. Typical error performance curves are shown in Figure 6.18 for rate $\frac{1}{2}$ codes using coherent BPSK over a soft (8-level) quantized channel, with Viterbi decoding, and a 32-bit path memory. Also plotted are the transfer function bounds for infinitely fine quantized received data [12]. Figure 6.19 gives the simulation results for Viterbi decoding with hard decision quantization [12]. Notice that each increment in constraint

TABLE 6.3 Basic Coding Gain (dB) for Soft Decision Viterbi Decoding

Uncoded E_b/N_0 (dB)	Code rate		$\frac{1}{3}$		$\frac{1}{2}$			$\frac{2}{3}$		$\frac{3}{4}$	
	P_B	K	7	8	5	6	7	6	8	6	9
6.8	10^{-3}		4.2	4.4	3.3	3.5	3.8	2.9	3.1	2.6	2.6
9.6	10^{-5}		5.7	5.9	4.3	4.6	5.1	4.2	4.6	3.6	4.2
11.3	10^{-7}		6.2	6.5	4.9	5.3	5.8	4.7	5.2	3.9	4.8
Upper bound			7.0	7.3	5.4	6.0	7.0	5.2	6.7	4.8	5.7

Source: I. M. Jacobs, "Practical Applications of Coding," *IEEE Trans. Inf. Theory*, vol. IT20, May 1974, pp. 305–310.



length impro

10^{-5}. Also,

approximatel

6.4.6 Best

The connec

usually sele

is to select

the maximu

number of p

represent, s

by consider

until only o

rate $\frac{1}{2}$, K =

by Odenwa

y with the required

) uncoded coherent
r simulation over a
ɔded E_b/N_0 is given
ng gain increases as
;ain cannot increase
is bound in decibels

(6.23)

ion of Table 6.3 also
eaker codes tend to
des.
channels with either
vary between 3 and
ory is usually a few
f the input bit history
n Section 6.3.4, one
is from the example
in take place as soon
is true; however, to
extensive amount of
be decoded. Instead,
lecoded. It has been
4 or 5 times the con-
ne optimum decoder
annels. Typical error
using coherent BPSK
ng, and a 32-bit path
initely fine quantized
for Viterbi decoding
rement in constraint

$\frac{2}{3}$	$\frac{3}{4}$	
8	6	9
3.1	2.6	2.6
4.6	3.6	4.2
5.2	3.9	4.8
6.7	4.8	5.7

ry, vol. IT20, May 1974;

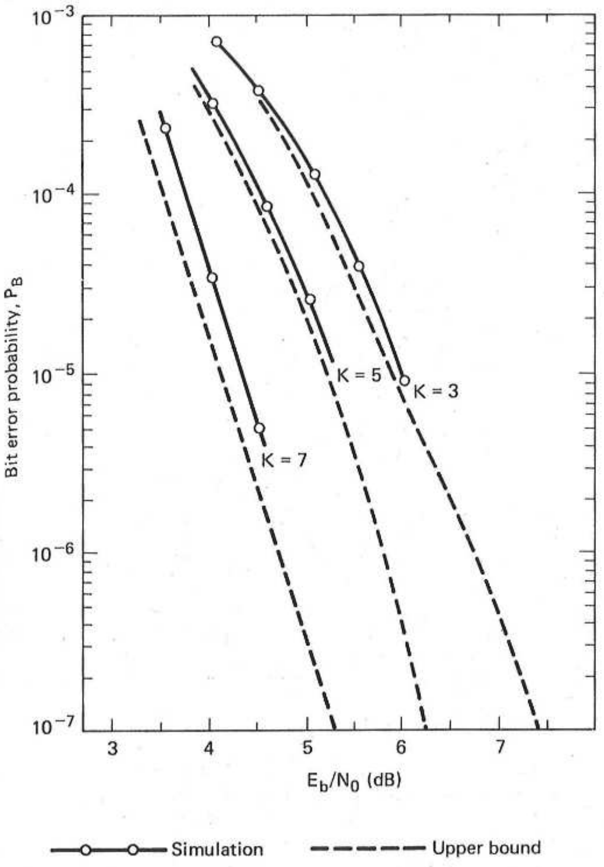


Figure 6.18 Bit error probability versus E_b/N_0 for rate $\frac{1}{2}$ codes using coherent BPSK over a soft quantized channel, Viterbi decoding, and a 32-bit path memory. (Reprinted with permission from J. A. Heller and I. M. Jacobs, "Viterbi Decoding for Satellite and Space Communication," *IEEE Trans. Commun. Technol.*, vol. COM19, no. 5, October 1971, Fig. 5, p. 84. © 1971 IEEE.)

length improves the required E_b/N_0 by a factor of approximately 0.5 dB at $P_B = 10^{-5}$. Also, as expected, the 3-bit soft decisions of the channel output result in approximately a 2-dB gain over the hard quantized BSC.

6.4.6 Best Known Convolutional Codes

The connection vectors or polynomial generators of a convolutional code are usually selected based on the code's free distance properties. The first criterion is to select a code that does not have catastrophic error propagation and that has the maximum free distance for the given rate and constraint length. Then the number of paths at the free distance d_f, or the number of data bit errors the paths represent, should be minimized. The selection procedure can be further refined by considering the number of paths or bit errors at $d_f + 1$, at $d_f + 2$, and so on, until only one code or class of codes remains. A list of the best known codes of rate $\frac{1}{2}$, $K = 3$ to 9, and rate $\frac{1}{3}$, $K = 3$ to 8, based on this criterion was compiled by Odenwalder [3, 23] and is given in Table 6.4. The connection vectors in this

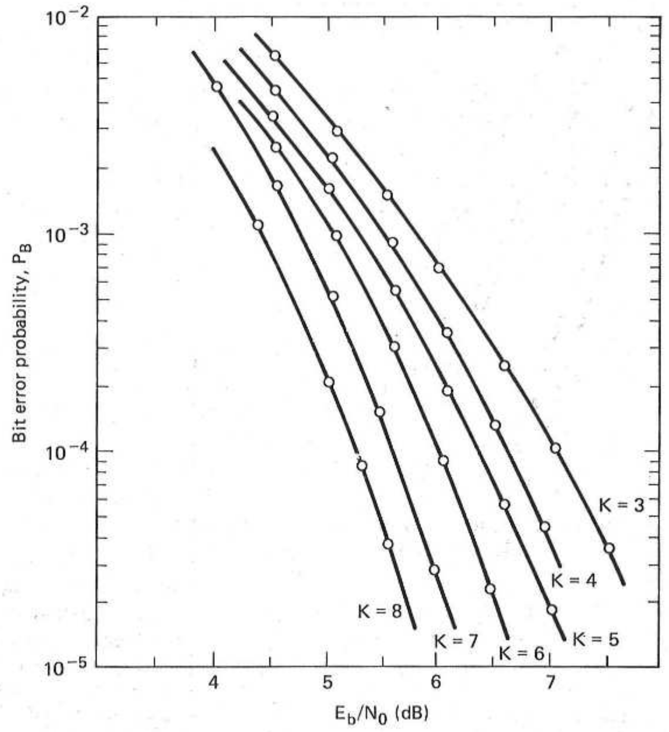


Figure 6.19 Bit error probability versus E_b/N_0 for rate $\frac{1}{2}$ codes using coherent BPSK over a BSC, Viterbi decoding, and a 32-bit path memory. (Reprinted with permission from J. A. Heller and I. M. Jacobs, "Viterbi Decoding for Satellite and Space Communication," *IEEE Trans. Commun. Technol.*, vol. COM19, no. 5, October 1971, Fig. 7, p. 84. © 1971 IEEE.)

table represent the presence or absence (1 or 0) of a tap connection on the corresponding stage of the convolutional encoder. The leftmost term corresponds to the leftmost stage of the encoder register, and the rightmost term corresponds to the rightmost stage, following the notation established in Figure 6.3. It is interesting to note that these connections can be inverted (leftmost and rightmost can be interchanged in the above description). Under the condition of Viterbi decoding, the inverted connections give rise to codes with identical distance properties, and hence identical performance, as those in Table 6.4.

6.4.7 Convolutional Code Rate Trade-Off

6.4.7.1 Performance with Coherent PSK Signaling

The error-correcting capability of a coding scheme increases as the number of channel symbols n per information bit k increases or the rate, k/n, decreases. However, the channel bandwidth and the decoder complexity both increase with n. The advantage of lower code rates when using convolutional codes with co-

TA
(F
I
herent PSK
rates), per
power, or p
shown [16,
$\frac{1}{2}$ to $\frac{1}{3}$ resul
the corresp
ues of cod
coding con
further dec
Sec. 6.4

es using coherent
/. (Reprinted with
ɔding for Satellite
vol. COM19, no.

:onnection on the cor-
st term corresponds to
st term corresponds to
Figure 6.3. It is inter-
ıost and rightmost can
ition of Viterbi decod-
:al distance properties,

ıcreases as the number
ıe rate, k/n, decreases.
xity both increase with
utional codes with co-

TABLE 6.4 Optimum Short Constraint Length Convolutional Codes (Rate $\frac{1}{2}$ and Rate $\frac{1}{3}$)

Rate	Constraint length	Free distance	Code vector
$\frac{1}{2}$	3	5	111 101
$\frac{1}{2}$	4	6	1111 1011
$\frac{1}{2}$	5	7	10111 11001
$\frac{1}{2}$	6	8	101111 110101
$\frac{1}{2}$	7	10	1001111 1101101
$\frac{1}{2}$	8	10	10011111 11100101
$\frac{1}{2}$	9	12	110101111 100011101
$\frac{1}{3}$	3	8	111 111 101
$\frac{1}{3}$	4	10	1111 1011 1101
$\frac{1}{3}$	5	12	11111 11011 10101
$\frac{1}{3}$	6	13	101111 110101 111001
$\frac{1}{3}$	7	15	1001111 1010111 1101101
$\frac{1}{3}$	8	16	11101111 10011011 10101001

Source: J. P. Odenwalder, *Error Control Coding Handbook*, Linkabit Corp., San Diego, Calif., July 15, 1976.

herent PSK, is that the required E_b/N_0 is decreased (for a large range of code rates), permitting the transmission of higher data rates for a given amount of power, or permitting reduced power for a given data rate. Simulation studies have shown [16, 22] that for a fixed constraint length, a decrease in the code rate from $\frac{1}{2}$ to $\frac{1}{3}$ results in a reduction of the required E_b/N_0 of roughly 0.4 dB. However, the corresponding increase in decoder complexity is about 17%. For smaller values of code rate, the improvement in performance relative to the increased decoding complexity diminishes rapidly [22]. Eventually, a point is reached where further decrease in code rate is characterized by a reduction in coding gain.

6.4.7.2 Performance with Noncoherent Orthogonal Signaling

In contrast to PSK, there is an optimum code rate of about $\frac{1}{2}$ for noncoherent orthogonal signaling. Error performance at rates of $\frac{1}{3}$, $\frac{2}{3}$, and $\frac{3}{4}$ are each worse than those for rate $\frac{1}{2}$. For a fixed constraint length, the rate $\frac{1}{3}$, $\frac{2}{3}$, and $\frac{3}{4}$ codes typically degrade by about 0.25, 0.5, and 0.3 dB, respectively, relative to the rate $\frac{1}{2}$ performance [16].

6.5 OTHER CONVOLUTIONAL DECODING ALGORITHMS

6.5.1 Sequential Decoding

Prior to the discovery of an optimum algorithm by Viterbi, other algorithms had been proposed for decoding convolutional codes. The earliest was the *sequential decoding algorithm*, originally proposed by Wozencraft [24, 25] and subsequently modified by Fano [2]. A sequential decoder works by generating hypotheses about the transmitted codeword sequence; it computes a metric between these hypotheses and the received signal. It goes forward as long as the metric indicates that its choices are likely; otherwise, it goes backward, changing hypotheses until, through a systematic trial-and-error search, it finds a likely hypothesis. Sequential decoders can be implemented to work with hard or soft decisions, but soft decisions are usually avoided because they greatly increase the amount of the required storage and the complexity of the computations.

Consider that using the encoder shown in Figure 6.3, a sequence $\mathbf{m} = 1\ 1\ 0\ 1\ 1$ is encoded into the codeword sequence $\mathbf{U} = 1\ 1\ 0\ 1\ 0\ 1\ 0\ 0\ 0\ 1$, as shown in Example 6.1. Assume that the received sequence $\mathbf{Z}$ is, in fact, a *correct* rendition of $\mathbf{U}$. The decoder has available a replica of the encoder code tree, shown in Figure 6.6, and can use the received sequence $\mathbf{Z}$ to penetrate the tree. The decoder starts at the time t_1 node of the tree and generates both paths leaving that node. The decoder follows that path which agrees with the received n code symbols. At the next level in the tree, the decoder again generates both paths leaving that node, and follows the path agreeing with the second group of n code symbols. Proceeding in this manner, the decoder quickly penetrates the tree.

Suppose, however, that the received sequence $\mathbf{Z}$ is a *corrupted* version of $\mathbf{U}$. The decoder starts at the time t_1 node of the code tree and generates both paths leading from that node. If the received n code symbols coincide with one of the generated paths, the decoder follows that path. If there is not agreement, the decoder follows the *most likely path* but keeps a cumulative count on the number of disagreements between the received symbols and the branch words on the path being followed. If two branches appear equally likely, the receiver uses an arbitrary rule, such as following the zero input path. At each new level in the tree, the decoder generates new branches and compares them with the next set of n received code symbols. The search continues to penetrate the tree along the most likely path and maintains the cumulative disagreement count.

If the disagreement count exceeds a certain number (which may increase as

we penetrate th
out of the patl
pathways to av
encoder in Fig
into the codew
and seventh bi

Time:

Mess
seque

Trans
seque

Rece
seque

Let us fc
that a cumula
trying an alte
represent the

1. At time
leaving
2. The mo
an inpu
bit one
3. At time
availabl
4. There i
(or bra
ment o
5. At time
availab
6. Again,
(or bra
7. At tim
availab
8. Again,
branch
9. But a
"back
to 2.
10. The al

ut ½ for noncoherent
are each worse than
nd ¾ codes typically
ve to the rate ½ per-

other algorithms had
t was the *sequential*
25] and subsequently
ng hypotheses about
between these hy-
the metric indicates
ing hypotheses until,
pothesis. Sequential
cisions, but soft de-
e amount of the re-

3, a sequence **m** =
0 1 0 1 0 0 0 1, as
is, in fact, a *correct*
encoder code tree,
o penetrate the tree.
es both paths leaving
the received *n* code
generates both paths
cond group of *n* code
netrates the tree.
corrupted version of
and generates both
ls coincide with one
re is not agreement,
ulative count on the
d the branch words
y likely, the receiver
h. At each new level
es them with the next
netrate the tree along
ment count.
hich may increase as

we penetrate the tree), the decoder decides that it is on an incorrect path, backs out of the path, and tries another. The decoder keeps track of the discarded pathways to avoid repeating any path excursions. For example, assume that the encoder in Figure 6.3 is used to encode the message sequence **m** = 1 1 0 1 1 into the codeword sequence **U** as shown in Example 6.1. Suppose that the fourth and seventh bits of the transmitted sequence **U** are received in error, such that:

Time:		t_1	t_2	t_3	t_4	t_5
Message sequence:	**m** =	1	1	0	1	1
Transmitted sequence:	**U** =	1 1	0 1	0 1	0 0	0 1
Received sequence:	**Z** =	1 1	0 0	0 1	1 0	0 1

Let us follow the decoder path trajectory with the aid of Figure 6.20. Assume that a cumulative path disagreement count of 3 is the criterion for backing up and trying an alternative path. On Figure 6.20 the numbers along the path trajectory represent the current disagreement count.

1. At time t_1 we receive symbols 11 and compare them with the branch words leaving the first node.
2. The most likely branch is the one with branch word 11 (corresponding to an input bit one or downward branching), so the decoder decides that input bit one is the correct decoding, and moves to the next level.
3. At time t_2, the decoder receives symbols 00 and compares them with the available branch words 10 and 01 at this second level.
4. There is no "best" path, so the decoder arbitrarily takes the input bit zero (or branch word 10) path, and the disagreement count registers a disagreement of 1.
5. At time t_3, the decoder receives symbols 01 and compares them with the available branch words 11 and 00 at this third level.
6. Again, there is no best path, so the decoder arbitrarily takes the input zero (or branch word 11) path, and the disagreement count is increased to 2.
7. At time t_4, the decoder receives symbols 10 and compares them with the available branch words 00 and 11 at this fourth level.
8. Again, there is no best path, so the decoder takes the input bit zero (or branch word 00) path, and the disagreement count is increased to 3.
9. But a disagreement count of 3 is the turnaround criterion, so the decoder "backs out" and tries the alternative path. The disagreement counter is reset to 2.
10. The alternative path is the input bit one (or branch word 11) path at the t_4

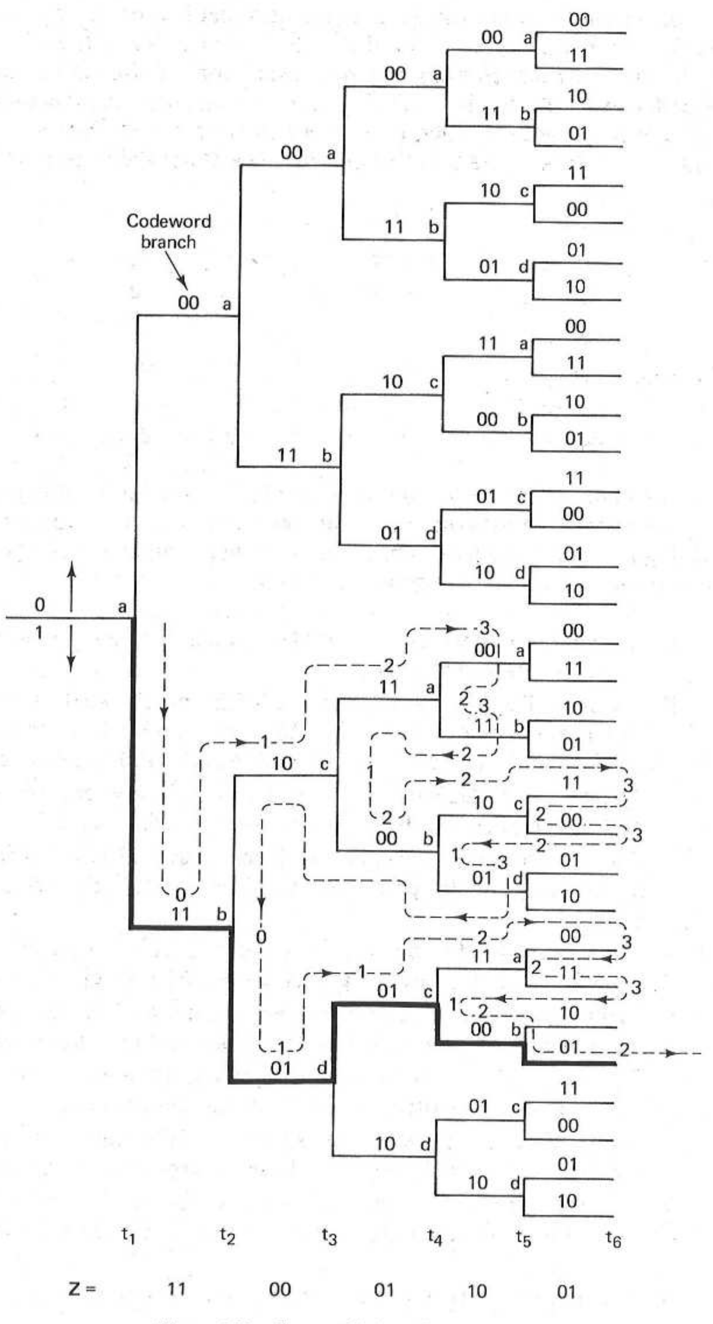


Figure 6.20 Sequential decoding example.

level. The (
is still a di:

11. But, 3 bei
and the co
at this t_4 le
to 1.

12. At the t_3 n
01, with tl
is increase

13. At the t_4 1
code syml

14. At the t_5 n
as is the r

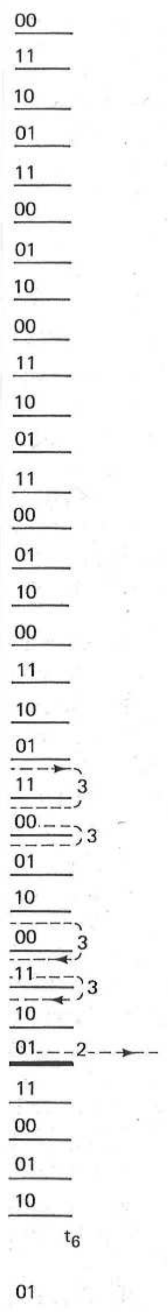
15. At this co
alternativ
a disagre
counter i:

16. The deco
alternativ
to the no

17. The deco
since the
the deco
other pat
to zero.

18. At the t
there is
the cour

The dec
which has not
decoded mes
trial-and-errc
performs the
at a time. If a
be wrong. Tl
metric. The
a road map.
respond to tl
landmarks (a
that he is or
recognize th
an alternativ



level. The decoder tries this, but compared to the received symbols 10, there is still a disagreement of 1, and the counter is reset to 3.

11. But, 3 being the turnaround criterion, the decoder backs out of this path, and the counter is reset to 2. All of the alternatives have now been traversed at this t_4 level, so the decoder returns to the node at t_3, and resets the counter to 1.

12. At the t_3 node, the decoder compares the symbols received at time t_3, namely 01, with the untried 00 path. There is a disagreement of 1, and the counter is increased to 2.

13. At the t_4 node, the decoder follows the branch word 10 that matches its t_4 code symbols of 10. The counter remains unchanged at 2.

14. At the t_5 node, there is no best path, so the decoder follows the upper branch, as is the rule, and the counter is increased to 3.

15. At this count, the decoder backs up, resets the counter to 2, and tries the alternative path at node t_5. Since the alternate branch word is 00, there is a disagreement of 1 with the received code symbols 01 at time t_5, and the counter is again increased to 3.

16. The decoder backs out of this path, and the counter is reset to 2. All of the alternatives have now been traversed at this t_5 level, so the decoder returns to the node at t_4 and resets the counter to 1.

17. The decoder tries the alternative path at t_4, which raises the metric to 3 since there is a disagreement in two positions of the branch word. This time the decoder must back up all the way to the time t_2 node because all of the other paths at higher levels have been tried. The counter is now decremented to zero.

18. At the t_2 node, the decoder now follows the branch word 01, and because there is a disagreement of 1 with the received code symbols 00 at time t_2, the counter is increased to 1.

The decoder continues in this way. As shown in Figure 6.20, the final path, which has not increased the counter to its turnaround criterion, yields the correctly decoded message sequence, 1 1 0 1 1. Sequential decoding can be viewed as a trial-and-error technique for searching out the correct path in the code tree. It performs the search in a sequential manner, always operating on just a single path at a time. If an incorrect decision is made, subsequent extensions of the path will be wrong. The decoder can eventually recognize its error by monitoring the path metric. The algorithm is similar to the case of an automobile traveler following a road map. As long as the traveler recognizes that the passing landmarks correspond to those on the map, he continues on the path. When he notices strange landmarks (an increase in his dissimilarity metric) the traveler eventually assumes that he is on an incorrect road, and he backs up to a point where he can now recognize the landmarks (his metric returns to an acceptable range). He then tries an alternative road.

6.5.2 Comparisons and Limitations of Viterbi and Sequential Decoding

The major drawback of the Viterbi algorithm is that while error probability decreases exponentially with constraint length, the number of code states, and consequently decoder complexity, *grows exponentially with constraint length.* On the other hand, the computational complexity of the Viterbi algorithm is independent of channel characteristics (compared to hard-decision decoding, soft-decision decoding requires only a trivial increase in the number of computations). Sequential decoding achieves asymptotically the same error probability as maximum likelihood decoding but without searching all possible states. In fact, with sequential decoding the number of states searched is essentially *independent of constraint length*, thus making it possible to use very large ($K = 41$) constraint

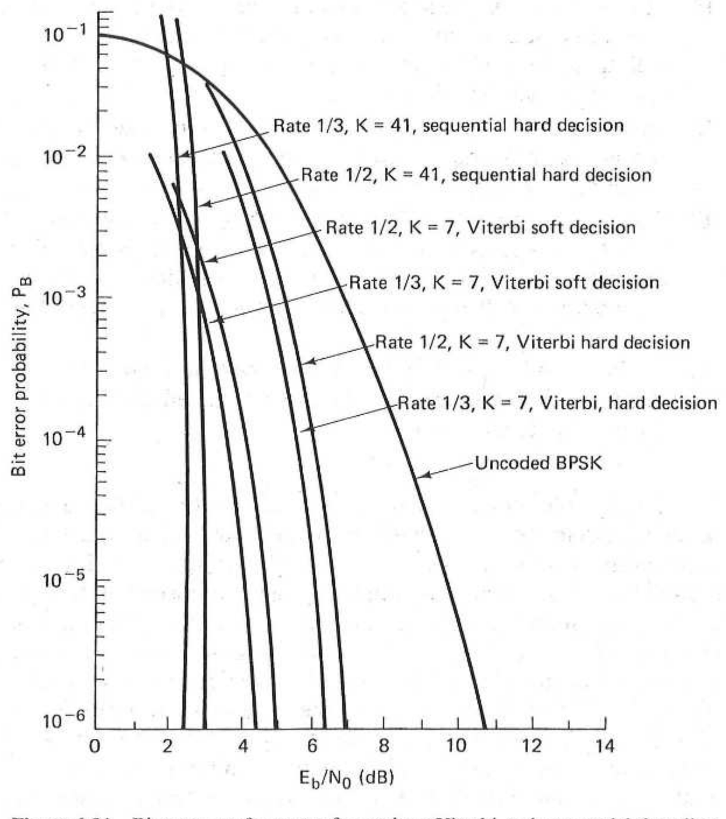


Figure 6.21 Bit error performance for various Viterbi and sequential decoding schemes using coherent BPSK over an AWGN channel. (Reprinted with permission from J. K. Omura and B. K. Levitt, "Coded Error Probability Evaluation for Antijam Communication Systems," *IEEE Trans. Commun.*, vol. COM30, no. 5, May 1982, Fig. 4, p. 900. © 1982 IEEE.)

lengths. This i
major drawba
searched is a ı
poor hypothes
a low SNR, n
this variability
riving sequenc
the decoder is
rate exceeds t
how large it is
error-free dat
through a rec
function of S
cation is the
In Figur
solutions to tl
decoding, illu
AWGN chan
sion, $K = 7$)
sequential de
Figure 6.21 tl
with sequent
of approxima
that the ma
accomplished

6.5.3 Feedb

A *feedback*
metrics com
positive inte
received coc
coder input t
the data bit
distance pat
The detailed
consider the
in Figure 6.
feedback de
considers th
Beginr
Hamming p
minimum di
if the minin
received sec
from time *t*

error probability de-
:ode states, and con-
onstraint length. On
bi algorithm is inde-
sion decoding, soft-
er of computations).
probability as max-
states. In fact, with
ially *independent of*
(K = 41) constraint

ision

decision

cision

decision

ıard decision

ɔi, hard decision

K

14

ntial decoding
d with permis-
lity Evaluation
l. COM30, no.

lengths. This is an important factor in providing such low error probabilities. The major drawback of sequential decoding is that the number of state metrics searched is a random variable. For sequential decoding, the expected number of poor hypotheses and backward searches is a function of the channel SNR. With a low SNR, more hypotheses must be tried than with a high SNR. Because of this variability in computational load, buffers must be provided to store the arriving sequences. Under low SNR, the received sequences must be buffered while the decoder is laboring to find a likely hypothesis. If the average symbol arrival rate exceeds the average symbol decode rate, the buffer will overflow, no matter how large it is, causing a loss of data. The sequential decoder typically puts out error-free data until the buffer overflows, at which time the decoder has to go through a recovery procedure. The buffer overflow threshold is a very sensitive function of SNR. Therefore, an important part of a sequential decoder specification is the *probability of buffer overflow*.

In Figure 6.21, some typical P_B versus E_b/N_0 curves for these two popular solutions to the convolutional decoding problem, Viterbi decoding and sequential decoding, illustrate their comparative performance using coherent BPSK over an AWGN channel. The curves compare Viterbi decoding (rates $\frac{1}{2}$ and $\frac{1}{3}$ hard decision, $K = 7$) versus Viterbi decoding (rates $\frac{1}{2}$ and $\frac{1}{3}$ soft decision, $K = 7$) versus sequential decoding (rates $\frac{1}{2}$ and $\frac{1}{3}$ hard decision, $K = 41$). One can see from Figure 6.21 that coding gains of approximately 8 dB at $P_B = 10^{-6}$ can be achieved with sequential decoders. Since the work of Shannon [26] foretold the potential of approximately 11 dB of coding gain compared to uncoded BPSK, it appears that the major portion of what is theoretically possible can already be accomplished.

6.5.3 Feedback Decoding

A *feedback decoder* makes a hard decision on the data bit at stage j based on metrics computed from stages $j, j + 1, \ldots, j + m$, where m is a preselected positive integer. *Look-ahead length*, L, is defined as $L = m + 1$, the number of received code symbols, expressed in terms of the corresponding number of encoder input bits that are used to decode an information bit. The decision of whether the data bit is zero or one depends on which branch the minimum Hamming distance path traverses in the *look-ahead window* from stage j to stage $j + m$. The detailed operation is best understood in terms of a specific example. Let us consider the use of a feedback decoder for the rate $\frac{1}{2}$ convolutional code shown in Figure 6.3. Figure 6.22 illustrates the tree diagram and the operation of the feedback decoder for $L = 3$. That is, in decoding the bit at branch j, the decoder considers the paths at branches $j, j + 1$, and $j + 2$.

Beginning with the first branch, the decoder computes 2^L or eight cumulative Hamming path metrics and decides that the bit for the first branch is zero if the minimum distance path is contained in the upper part of the tree, and decides one if the minimum distance path is in the lower part of the tree. Assume that the received sequence is $\mathbf{Z} = 1\ 1\ 0\ 0\ 0\ 1\ 0\ 0\ 0\ 1$. We now examine the eight paths from time t_1 through time t_3 in the block marked A in Figure 6.22, and compute

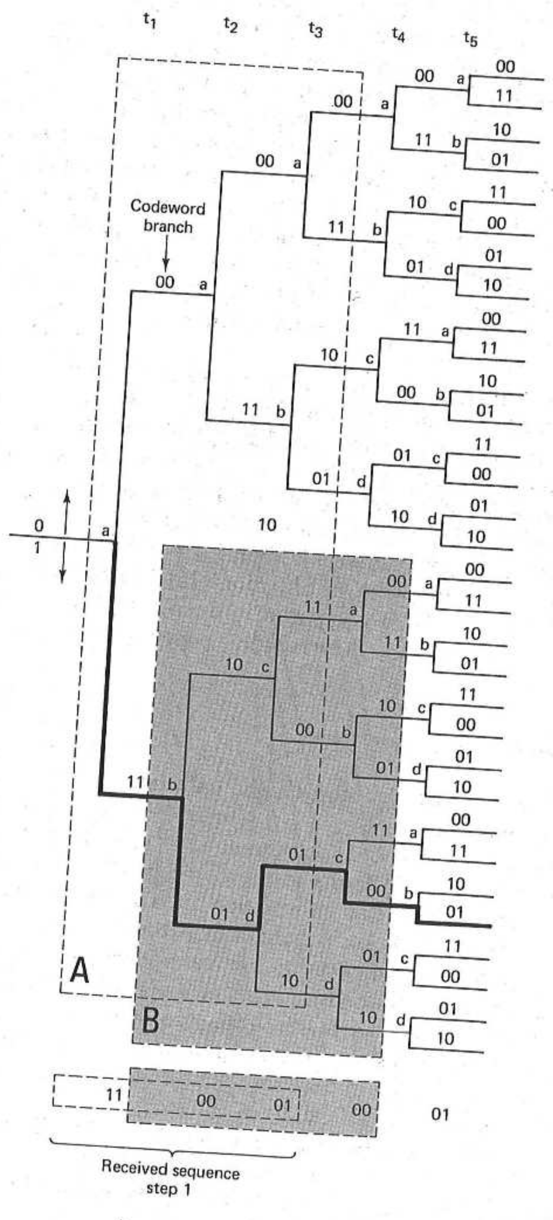


Figure 6.22 Feedback decoding example.

metrics comparing
branches deep tin
path metrics (star

We see that the r
fore, the first dec
tree). The next s
one stage deeper
Having decoded
to the right and
place in the blo
path to bottom

For the assume
half of block *B*

The same
decoder is call
to the decoder
next. On the B
decoder [17] i
all those of w
An important
ahead length.
implementatio

6.6 INTERLEAVING

Throughout
memoryless,
independent
dependent s
fading chann
time. Anoth
the receiver
signals *arri*
is distorted
suffer from
and other l
jamming).

metrics comparing these eight paths with the first six received code symbols (three branches deep times two symbols per branch). Listing the Hamming cumulative path metrics (starting from the top path), they are:

Upper-half metrics: 3, 3, 6, 4

Lower-half metrics: 2, 2, 1, 3

We see that the minimum metric is contained in the lower part of the tree. Therefore, the first decoded bit is one (characterized by a downward movement on the tree). The next step is to extend the lower part of the tree (the part that survived) one stage deeper, and again compute eight metrics, this time from t_2 through t_4. Having decoded the first two code symbols, we now slide over two code symbols to the right and again compute the path metrics for six code symbols. This takes place in the block marked B in Figure 6.22. Again, listing the metrics from top path to bottom path, they are:

Upper-half metrics: 2, 4, 3, 3

Lower-half metrics: 3, 1, 4, 4

For the assumed received sequence, the minimum metric is found in the lower half of block B. Therefore, the second decoded bit is one.

The same procedure continues until the entire message is decoded. The decoder is called a *feedback decoder* because the detection decisions are *fed back* to the decoder in determining the subset of code paths that are to be considered next. On the BSC, the feedback decoder can perform nearly as well as the Viterbi decoder [17] in that it can correct all the more probable error patterns, namely all those of weight $(d_f - 1)/2$ or less, where d_f is the free distance of the code. An important design parameter for feedback convolutional decoders is L, the look-ahead length. Increasing L increases the coding gain but also increases the decoder implementation complexity.

6.6 INTERLEAVING AND CONCATENATED CODES

Throughout this chapter and Chapter 5 we have assumed that the channel is *memoryless,* since we have considered codes that are designed to combat random independent errors. A channel that has *memory* is one that exhibits mutually dependent signal transmission impairments. An example of such a channel is a fading channel, particularly when the fading varies slowly compared to one symbol time. Another type of impairment, called *multipath,* involves signal arrivals at the receiver over two or more paths of different lengths. The effect is that the signals *arrive out of phase* with each other, and the cumulative received signal is distorted. High-frequency (HF) and tropospheric propagation radio channels suffer from such phenomena. Also, some channels suffer from switching noise and other burst noise (e.g., telephone channels or channels disturbed by pulse jamming). All of these time-correlated impairments result in statistical dependence among successive symbol transmissions. That is, the disturbances tend to cause errors that occur in bursts, instead of as isolated events.

Under the assumption that the channel has memory, the errors no longer can be characterized as single randomly distributed bit errors whose occurrence is independent from bit to bit. Most block or convolutional codes are designed to combat random independent errors. The result of a channel having memory on such coded signals is to cause degradation in error performance. Coding techniques for channels with memory have been proposed [27, 28], but the greatest problem with such coding is the difficulty in obtaining accurate models of the often time-varying statistics of such channels. One technique, which only requires a knowledge of the *duration or span* of the channel memory, *not* its exact statistical characterization, is the use of time diversity or *interleaving*.

Interleaving the coded message before transmission and deinterleaving after reception causes bursts of channel errors to be spread out in time and thus to be handled by the decoder as if they were random errors. Since, in all practical cases, the channel memory decreases with time separation, the idea behind interleaving is to separate the codeword symbols in time. The intervening times are similarly filled by the symbols of other codewords. Separating the symbols in time effectively transforms a channel with memory to a *memoryless* one, and thereby enables the random-error-correcting codes to be useful in a burst-noise channel.

The interleaver shuffles the code symbols over a span of several block lengths (for block codes) or several constraint lengths (for convolutional codes). The span required is determined by the burst duration. The details of the bit redistribution pattern must be known to the receiver in order for the symbol stream to be deinterleaved before being decoded. Figure 6.23 illustrates a simple interleaving example. In Figure 6.23a we see seven uninterleaved codewords, A through G. Each codeword is comprised of seven code symbols. Let us assume that the code has a single-error-correcting capability within each seven-symbol sequence. If the memory span of the channel is one codeword in duration, such a seven-symbol-time noise burst could destroy the information contained in one or two codewords. However, suppose that, after having encoded the data, the code symbols were then *interleaved* or shuffled, as shown in Figure 6.23b. That is, each code symbol of each codeword is separated from its preinterleaved neighbors by a span of seven symbol times. The interleaved stream is then used to modulate a waveform that is transmitted over the channel. A contiguous channel noise burst occupying seven symbol times is seen in Figure 6.23b, to affect one code symbol from each of the original seven codewords. Upon reception, the stream is first deinterleaved so that it resembles the original coded sequence in Figure 6.23a. Then the stream is decoded. Since each codeword possesses a single-error-correcting capability, the burst noise has no degrading effect on the final sequence.

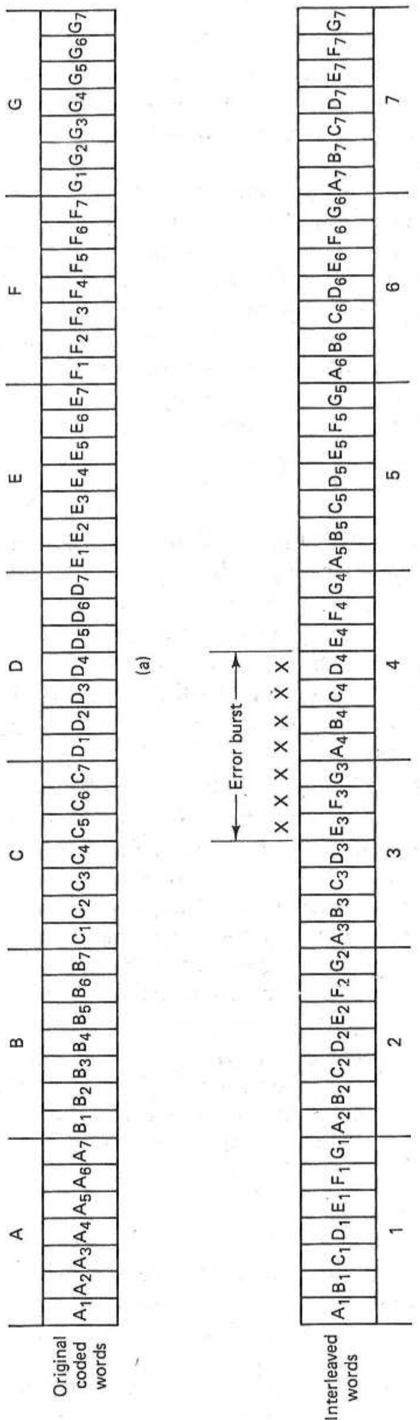
Interleaving techniques have proven useful for all the convolutional and block codes described here and in Chapter 5. Two types of interleavers are commonly used, *block interleavers* and *convolutional interleavers*. They are each described below.

bances tend to cause

the errors no longer
ɪrs whose occurrence
ɪl codes are designed
ɪnnel having memory
ɪmance. Coding tech-
28], but the greatest
curate models of the
ɪ, which only requires
ɪot its exact statistical
ɪ.

d deinterleaving after
n time and thus to be
, in all practical cases,
ɪa behind interleaving
ng times are similarly
ymbols in time effec-
one, and thereby en-
urst-noise channel.
ɪpan of several block
convolutional codes).
The details of the bit
for the symbol stream
strates a simple inter-
leaved codewords, A
mbols. Let us assume
in each seven-symbol
vord in duration, such
ation contained in one
encoded the data, the
in Figure 6.23b. That
s preinterleaved neigh-
stream is then used to
A contiguous channel
re 6.23b, to affect one
. Upon reception, the
nal coded sequence in
vord possesses a single-
ling effect on the final

the convolutional and
f interleavers are com-
ɪavers. They are each



Original coded words
A B C D E F G
$A_1 A_2 A_3 A_4 A_5 A_6 A_7 B_1 B_2 B_3 B_4 B_5 B_6 B_7 C_1 C_2 C_3 C_4 C_5 C_6 C_7 D_1 D_2 D_3 D_4 D_5 D_6 D_7 E_1 E_2 E_3 E_4 E_5 E_6 E_7 F_1 F_2 F_3 F_4 F_5 F_6 F_7 G_1 G_2 G_3 G_4 G_5 G_6 G_7$
(a)
Error burst
X X X X X X X
Interleaved words
$A_1 B_1 C_1 D_1 E_1 F_1 G_1 A_2 B_2 C_2 D_2 E_2 F_2 G_2 A_3 B_3 C_3 D_3 E_3 F_3 G_3 A_4 B_4 C_4 D_4 E_4 F_4 G_4 A_5 B_5 C_5 D_5 E_5 F_5 G_5 A_6 B_6 C_6 D_6 E_6 F_6 G_6 A_7 B_7 C_7 D_7 E_7 F_7 G_7$
1 2 3 4 5 6 7
(b)

Figure 6.23 Interleaving example. (a) Original uninterleaved codewords, each comprised of seven code symbols. (b) Interleaved code symbols.

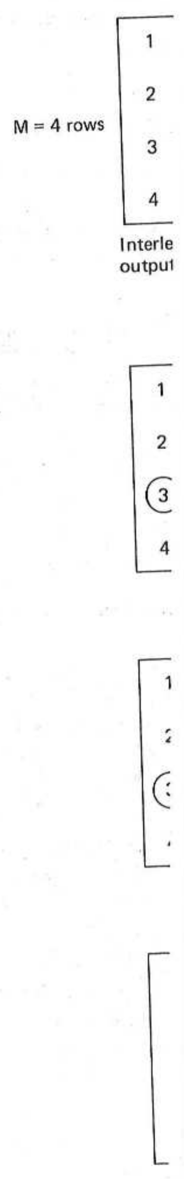
6.6.1 Block Interleaving

A block interleaver accepts the coded symbols in blocks from the encoder, permutes the symbols, and then feeds the rearranged symbols to the modulator. The usual permutation of the block is accomplished by *filling the columns* of an M-row-by N-column ($M \times N$) array with the encoded sequence. After the array is completely filled, the symbols are then fed to the modulator *one row at a time* and transmitted over the channel. At the receiver, the deinterleaver performs the inverse operation; it accepts the symbols from the demodulator, deinterleaves them, and feeds them to the decoder. Symbols are entered into the deinterleaver array by rows, and removed by columns. Figure 6.24a illustrates an example of an interleaver with $M = 4$ rows and $N = 6$ columns. The entries in the array illustrate the order in which the 24 code symbols are placed into the interleaver. The output sequence to the transmitter consists of code symbols removed from the array by rows, as shown in the figure. The most important characteristics of such a block interleaver are as follows:

1. Any burst of less than N contiguous channel symbol errors results in isolated errors at the deinterleaver output that are separated from each other by at least M symbols.
2. Any bN burst of errors, where $b > 1$, results in output bursts from the deinterleaver of no more than $\lceil b \rceil$ symbol errors. Each output burst is separated from the other bursts by no less than $M - \lfloor b \rfloor$ symbols. The notation $\lceil x \rceil$ means the smallest integer no less than x, and $\lfloor x \rfloor$ means the largest integer no greater than x.
3. A periodic sequence of single errors spaced N symbols apart results in a single burst of errors of length M at the deinterleaver output.
4. The interleaver/deinterleaver end-to-end delay is approximately $2MN$ symbol times. To be precise, only $M(N - 1) + 1$ memory cells need to be filled before transmission can begin (as soon as the first symbol of the last column of the $M \times N$ array is filled). A corresponding number needs to be filled at the receiver before decoding begins. Thus the minimum end-to-end delay is $(2MN - 2M + 2)$ symbol times, not including any channel propagation delay.
5. The memory requirement is MN symbols for each location (interleaver and deinterleaver). However, since the $M \times N$ array needs to be (mostly) filled before it can be read out, a memory of $2MN$ symbols is generally implemented at each location to allow the emptying of one $M \times N$ array while the other is being filled, and vice versa.

Example 6.3 Interleaver Characteristics

Using the $M = 4$, $N = 6$ interleaver structure of Figure 6.24a, verify each of the block interleaver characteristics described above.



ıe encoder, per-
modulator. The
ılumns of an M-
ıfter the array is
ıe row at a time
ver performs the
ır, deinterleaves
he deinterleaver
s an example of
ries in the array
ı the interleaver.
s removed from
haracteristics of

esults in isolated
each other by at

bursts from the
tput burst is sep-
ols. The notation
ıeans the largest

ıpart results in a
ıut.
ıately 2*MN* sym-
need to be filled
f the last column
ds to be filled at
d-to-end delay is
nnel propagation

ı (interleaver and
be (mostly) filled
generally imple-
× *N* array while

verify each of the

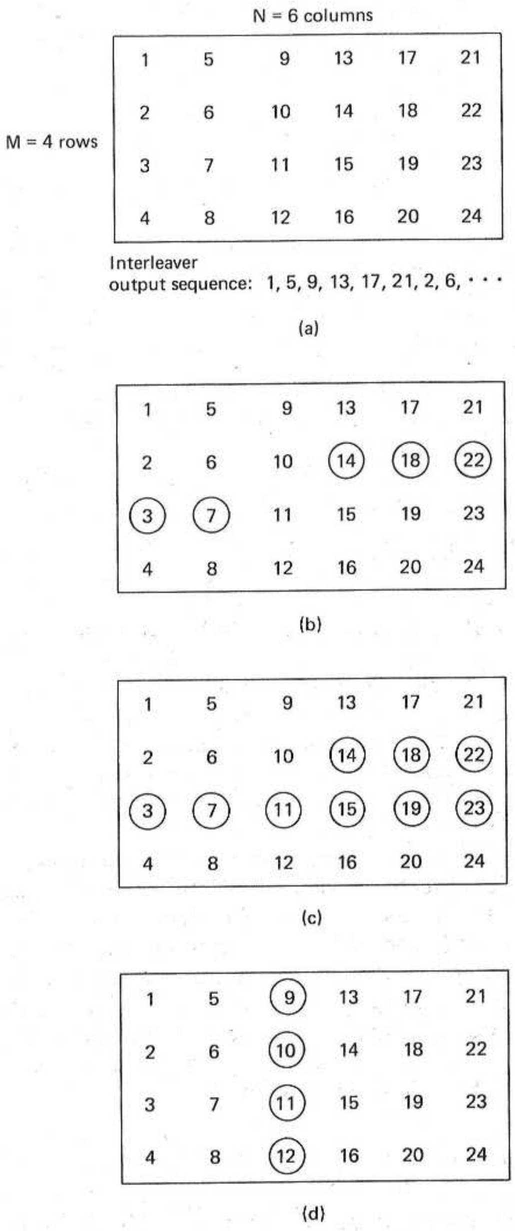

N = 6 columns
M = 4 rows
Interleaver output sequence: 1, 5, 9, 13, 17, 21, 2, 6, · · ·
(a)
(b)
(c)
(d)

Figure 6.24 Block interleaver example. (a) $M \times N$ block interleaver. (b) Five-symbol error burst. (c) Nine-symbol error burst. (d) Periodic single-error sequence spaced $N = 6$ symbols apart.

Solution

1. Let there be a noise burst of five symbol times, such that the symbols shown encircled in Figure 6.24b experience errors in transmission. After deinterleaving at the receiver, the sequence is

   1 2 ③ 4 5 6 ⑦ 8 9 10 11 12

   13 ⑭ 15 16 17 ⑱ 19 20 21 ㉒ 23 24

   where the encircled symbols are in error. It is seen that the smallest separation between symbols in error is $M = 4$.
2. Let $b = 1.5$ so that $bN = 9$. Figure 6.24c illustrates an example of a nine-symbol error burst. After deinterleaving at the receiver, the sequence is

   1 2 ③ 4 5 6 ⑦ 8 9 10 ⑪ 12

   13 ⑭ ⑮ 16 17 ⑱ ⑲ 20 21 ㉒ ㉓ 24

   Again, the encircled symbols are in error. It is seen that the bursts consist of no more than $\lceil 1.5 \rceil = 2$ contiguous symbols and that they are separated by at least $M - \lfloor 1.5 \rfloor = 4 - 1 = 3$ symbols.
3. Figure 6.24d illustrates a sequence of single errors spaced by $N = 6$ symbols apart. After deinterleaving at the receiver, the sequence is

   1 2 3 4 5 6 7 8 ⑨ ⑩ ⑪ ⑫

   13 14 15 16 17 18 19 20 21 22 23 24

   It is seen that the deinterleaved sequence has a single error burst of length $M = 4$ symbols.
4. End-to-end delay: The minimum end-to-end delay due to the interleaver and deinterleaver is $(2MN - 2M + 2) = 42$ symbol times.
5. Memory requirement: The interleaver and the deinterleaver arrays are each of size $M \times N$. Therefore, storage for $MN = 24$ symbols is required at each end of the channel. As mentioned earlier, storage for $2MN = 48$ symbols would generally be implemented.

Typically, for use with a single-error-correcting code the interleaver parameters are selected such that the number of columns N overbounds the *expected burst length*. The choice of the number of rows M is dependent on the coding scheme used. For block codes, M should be larger than the code block length, while for convolutional codes, M should be larger than the constraint length. Thus a burst of length N can cause at most a single error in any block codeword; similarly, with convolutional codes, there will be at most a single error in any decoding constraint length. For t-error-correcting codes, the choice of N need only overbound the expected burst length divided by t.

6.6.2 Convolutional Interleaving

Convolutional interleavers have been proposed by Ramsey [29] and Forney [30]. The structure proposed by Forney appears in Figure 6.25. The code symbols are

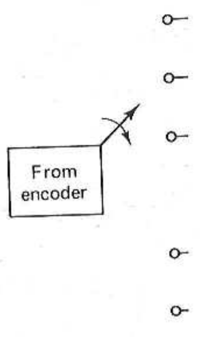


Figure 6
deinterlea

sequentially shi
J symbols mor
no storage (the
the commutato
in while the ol
transmitter. Af
register and st
the input and
be synchroniz

Figure 6.
($J = 1$) interle
deinterleaver
decoder. Figu
known states.
and the entry
9 to 12 enter
symbols, but
shows symbo
terleaver, syn
in this way un
is presented t

The per
block interle
leaving is tha
symbols, wh
of the chann
over the blo

that the symbols shown
ion. After deinterleaving

20 21 ㉒ 23 24

t the smallest separation

ɜxample of a nine-symbol
quence is

20 21 ㉒ ㉓ 24

t the bursts consist of no
are separated by at least

aced by $N = 6$ symbols
e is

19 20 21 22 23 24

rror burst of length $M =$

the interleaver and dein-

eaver arrays are each of
s is required at each end
$= 48$ symbols would gen-

the interleaver param-
erbounds the *expected*
pendent on the coding
the code block length,
constraint length. Thus
any block codeword;
t a single error in any
the choice of N need

y [29] and Forney [30].
The code symbols are

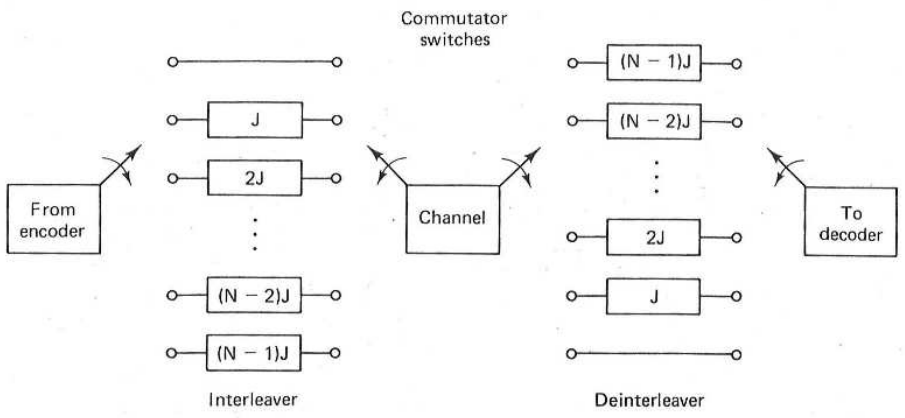


Figure 6.25 Shift register implementation of a convolutional interleaver/deinterleaver.

sequentially shifted into the bank of N registers; each successive register provides J symbols more storage than did the preceding one. The zeroth register provides no storage (the symbol is transmitted immediately). With each new code symbol the commutator switches to a new register, and the new code symbol is shifted in while the oldest code symbol in that register is shifted out to the modulator/transmitter. After the $(N - 1)$th register, the commutator returns to the zeroth register and starts again. The deinterleaver performs the inverse operation, and the input and output commutators for both interleaving and deinterleaving must be synchronized.

Figure 6.26 illustrates an example of a simple convolutional four-register ($J = 1$) interleaver being loaded by a sequence of code symbols. The synchronized deinterleaver is shown simultaneously feeding the deinterleaved symbols to the decoder. Figure 6.26a shows symbols 1 to 4 being loaded; the $\times$s represent unknown states. Figure 6.26b shows the first four symbols shifted within the registers and the entry of symbols 5 to 8 to the interleaver input. Figure 6.6c shows symbols 9 to 12 entering the interleaver. The deinterleaver is now filled with message symbols, but nothing useful is being fed to the decoder yet. Finally, Figure 6.6d shows symbols 13 to 16 entering the interleaver, and at the output of the deinterleaver, symbols 1 to 4 are being passed to the decoder. The process continues in this way until the entire codeword sequence, in its original preinterleaved form, is presented to the decoder.

The performance of a convolutional interleaver is very similar to that of a block interleaver. The important advantage of convolutional over block interleaving is that with convolutional interleaving the end-to-end delay is $M(N - 1)$ symbols, where $M = NJ$, and the memory required is $M(N - 1)/2$ at both ends of the channel. Therefore, there is a reduction of one-half in delay and memory over the block interleaving requirements [16].

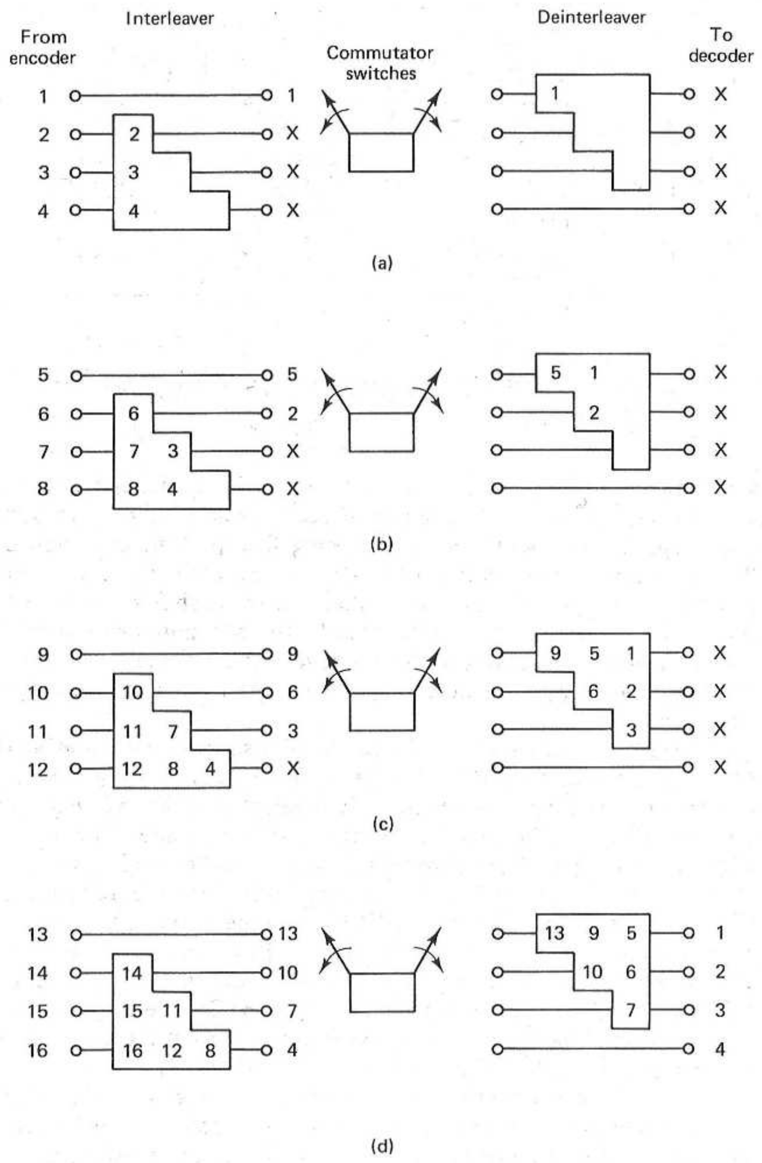
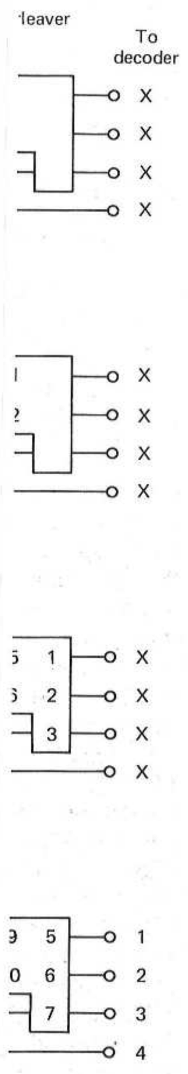


Figure 6.26 Convolutional interleaver/deinterleaver example.

6.6.3 Concaten

A concatenated
outer code, to a
order of encodi
modulator/demc
channel errors.
then reduces th
for using a con
plementation c
single coding o
coding steps. T
at the output o
One of th
coded convolu
interleaving be
E_b/N_0 in the r
from the Shan
tem, the demo
lutional decod
errors to the R
to occur in bu
binary data st
R–S code dep
is undisturbed
error, the R–S
one bit being i
performance i
Hence the in
(not at the bi
cation of suc



xample.

6.6.3 Concatenated Codes

A concatenated code is one that uses two levels of coding, an inner code and an outer code, to achieve the desired error performance. Figure 6.27 illustrates the order of encoding and decoding. The inner code, the one that interfaces with the modulator/demodulator and channel, is usually configured to correct most of the channel errors. The outer code, usually a higher-rate (lower-redundancy) code, then reduces the probability of error to the specified level. The primary reason for using a concatenated code is to achieve a low error rate with an overall implementation complexity which is less than that which would be required by a single coding operation. In Figure 6.27 an interleaver is shown between the two coding steps. This is usually required to spread any error bursts that may appear at the output of the inner coding operation.

One of the most popular concatenated coding systems uses a Viterbi-decoded convolutional inner code and a Reed–Solomon (R–S) outer code, with interleaving between the two coding steps [23]. Operation of such systems with E_b/N_0 in the range 2.0 to 2.5 dB to achieve $P_B = 10^{-5}$ (only about 4 dB away from the Shannon limit) is now feasible with practical hardware [16]. In this system, the demodulator outputs soft quantized code symbols to the inner convolutional decoder, which in turn outputs hard quantized code symbols with bursty errors to the R–S decoder. (In a Viterbi-decoded system, the output errors tend to occur in bursts.) The outer R–S code is formed from m-bit segments of the binary data stream (see Section 5.7.4). The performance of such a (nonbinary) R–S code depends only on the number of *symbol errors* in the block. The code is undisturbed by burst errors within an m-bit symbol. That is, for a given symbol error, the R–S code performance is the same whether the symbol error is due to one bit being in error or m bits being in error. However, the concatenated system performance is severely degraded by correlated errors among successive symbols. Hence the interleaving between codes needs to take place at the symbol level (not at the bit level). In the next section we consider a popular consumer application of such symbol interleaving in a concatenated system.

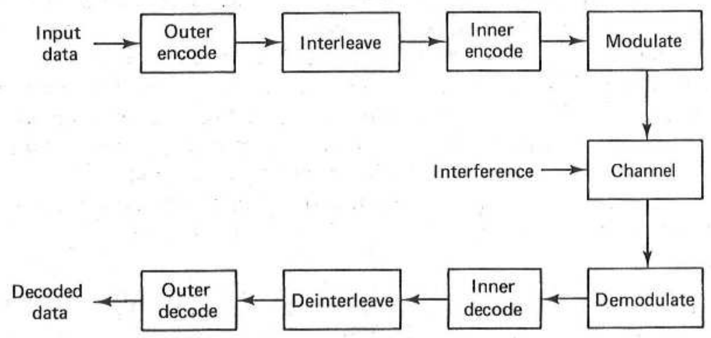
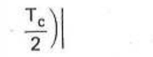
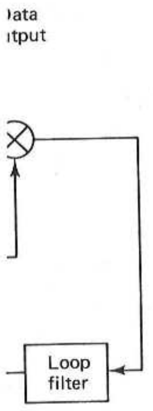


Figure 6.27 Block diagram of a concatenated coding system.



.tes the necessity of
aths. The signal-to-
ıan that of the DLL
ensive treatment of



10.6 SPREAD-SPECTRUM APPLICATIONS

10.6.1 Code-Division Multiple Access

Spread-spectrum multiple access techniques allow multiple signals occupying the same RF bandwidth to be transmitted simultaneously without interfering with one another. The application of spread-spectrum techniques to the problem of multiple access was discussed in Chapter 9 for a frequency hopped code-division multiple access (FH/CDMA) scheme. Here we consider CDMA using direct sequence (DS/CDMA). In these schemes, each of N user groups is given its own code, $g_i(t)$, where $i = 1, 2, \ldots, N$. The user codes are approximately orthogonal, so that the cross-correlation of two different codes is near zero. The main advantage of a CDMA system is that all the participants can share the full spectrum of the resource asynchronously; that is, the transition times of the different users' symbols do not have to coincide.

A typical DS/CDMA block diagram is shown in Figure 10.25. The first block illustrates the data modulation of a carrier, $A \cos \omega_0 t$. The output of the data modulator belonging to a user from group 1, $s_1(t)$, is shown below. The waveform is very general in form; no restriction has been placed on the type of modulation that can be used.

$$s_1(t) = A_1(t) \cos [\omega_0 t + \phi_1(t)] \tag{10.37}$$

Next, the data-modulated signal is multiplied by the spreading signal $g_1(t)$ belonging to user group 1, and the resulting signal, $g_1(t)s_1(t)$, is transmitted over the channel. Simultaneously, users from group 2 through N multiply their signals by their own code functions. Frequently, each code function is kept secret, and its use is restricted to the community of authorized users. The signal present at the receiver is the linear combination of the emanations from each of the users. Neglecting signal delays, we show this linear combination below.

$$g_1(t)s_1(t) + g_2(t)s_2(t) + \cdots + g_N(t)s_N(t) \tag{10.38}$$

As mentioned earlier, multiplication of $s_1(t)$ by $g_1(t)$ produces a signal whose spectrum is the convolution of the spectrum of $s_1(t)$ with the spectrum of $g_1(t)$. Thus, assuming that the signal $s_1(t)$ is relatively narrowband compared with the code or spreading signal $g_1(t)$, the product signal $g_1(t)s_1(t)$ will have approximately the bandwidth of $g_1(t)$. Assume that the receiver is configured to receive messages from user group 1. Assume, too, that the $g_1(t)$ code, generated at the receiver, is perfectly synchronized with the received signal from a group 1 user. The first stage of the receiver multiplies the incoming signal of Equation (10.38) by $g_1(t)$. The output of the multiplier will yield the following terms:

Desired signal: $g_1^2(t)s_1(t)$

Plus a composite of

undesired signals: $g_1(t)g_2(t)s_2(t) + g_1(t)g_3(t)s_3(t)$
$$+ \cdots + g_1(t)g_N(t)s_N(t) \tag{10.39}$$

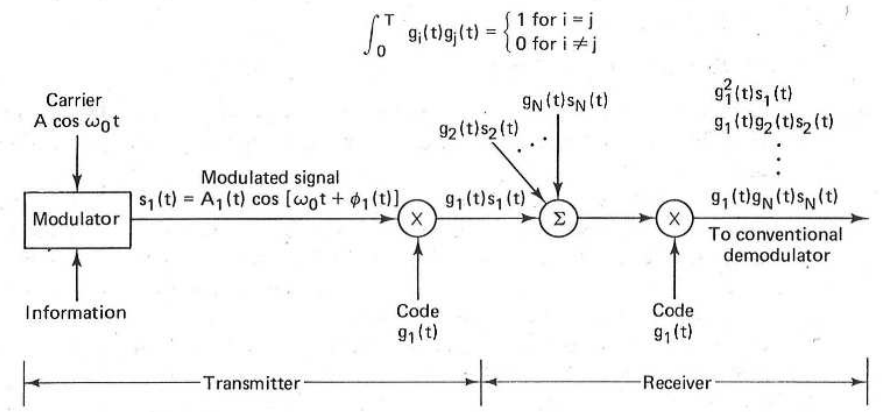


Figure 10.25 Code-division multiple access.

If the code functions, $\{g_i(t)\}$, are chosen with orthogonal properties, similar to Equation (10.14), the desired signal can be extracted perfectly in the absence of noise since $\int_0^T g_i^2(t) = 1$, and the undesired signals are easily rejected, since $\int_0^T g_i(t)g_j(t)\,dt = 0$ for $i \neq j$. In practice, the codes are not perfectly orthogonal; hence the cross-correlation between user codes introduces performance degradation, which limits the maximum number of simultaneous users.

Consider the frequency-domain view of the DS/CDMA receiver. Figure 10.26a illustrates the wideband input to the receiver; it consists of wanted and unwanted signals, each spread by its own code with code rate R_p, and each having a power spectral density of the form $\text{sinc}^2(f/R_p)$. Receiver thermal noise is also shown as having a flat spectrum across the band. The combined waveform of Equation (10.39) (desired plus undesired signals) is applied to the input of the receiver correlator driven by a synchronous replica of $g_1(t)$. Figure 10.26b illustrates the spectrum after correlation with the code $g_1(t)$ (despreading). The desired

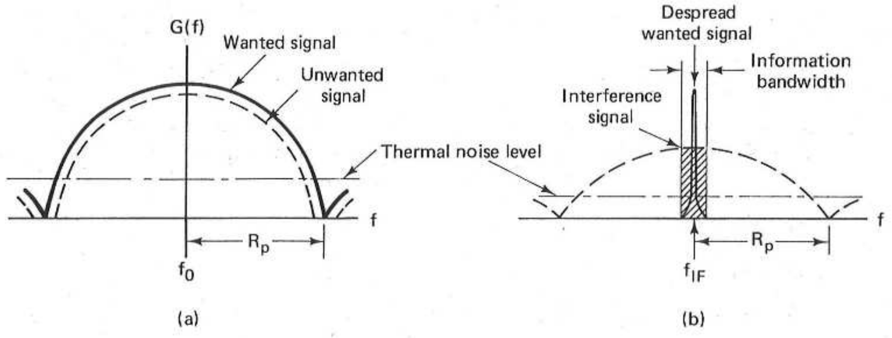
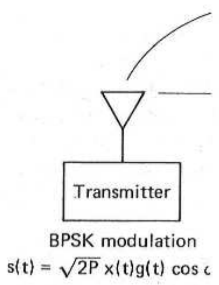


Figure 10.26 Spread-spectrum signal detection. (a) Spectrum at the input to receiver. (b) Spectrum after correlation with the correct and synchronized PN code.

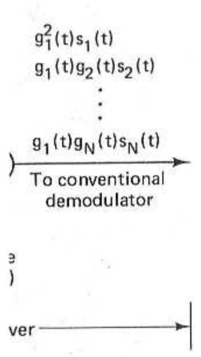
signal, occ
quency (IF
wide enoug
tion (10.39)
trum of the
will cause i
Pursle
using DS, t
Geraniotis
FH and DS

10.6.2 Mul

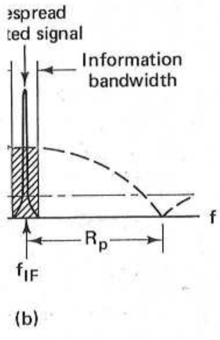
Consider a
channel tha
multiple pa
from buildi
signal level
a different
paths. Figu
multipath
television r
ditions, cor
In a d
is synchror
signal can l
$r(t) = Ax$
where $x(t)$ i
process, an
in the inter
distributed



Fig



properties, similar
ctly in the absence
sily rejected, since
rfectly orthogonal;
erformance degra-
iers.
A receiver. Figure
ists of wanted and
R_p, and each having
ermal noise is also
bined waveform of
to the input of the
Figure 10.26b illus-
ading). The desired



:he input to re-
ized PN code.

signal, occupying the information bandwidth centered at an intermediate frequency (IF), is then applied to a conventional demodulator, with bandwidth just wide enough to accommodate the despread signal. The undesired signals of Equation (10.39) remain effectively spread by $g_1(t)g_i(t)$. Only that portion of the spectrum of the unwanted signals falling in the information bandwidth of the receiver will cause interference with the desired signal.

Pursley [17] presents an excellent treatment on the performance of SSMA using DS, taking correlation properties of the code sequences into account. Also, Geraniotis [18] and Geraniotis and Pursley [19, 20] evaluate the performance of FH and DS multiple access systems subject to interference.

10.6.2 Multipath Channels

Consider a DS binary PSK communication system operating over a multipath channel that has more than one path from the transmitter to the receiver. Such multiple paths may be due to atmospheric reflection or refraction, or reflections from buildings or other objects, and may result in fluctuations in the received signal level. The different paths may consist of several discrete paths each with a different attentuation and time delay, or they might consist of a continuum of paths. Figure 10.27 illustrates a communication link with two discrete paths. The multipath wave is delayed by some time, τ, compared to the direct wave. In television receivers, signals such as these cause "ghosts," or under extreme conditions, complete loss of picture synchronization.

In a direct-sequence spread-spectrum system, if we assume that the receiver is synchronized to the time delay and RF phase of the direct path, the received signal can be expressed as

$$r(t) = Ax(t)g(t)\cos\omega_0 t + \alpha Ax(t-\tau)g(t-\tau)\cos(\omega_0 t + \theta) + n(t) \quad (10.40)$$

where $x(t)$ is the data signal, $g(t)$ the code signal, $n(t)$ a zero-mean Gaussian noise process, and τ the differential time delay between the two paths, assumed to be in the interval $0 < \tau < T$. The angle θ is a random phase, assumed to be uniformly distributed in the range $(0, 2\pi)$, and α is the attenuation of the multipath signal

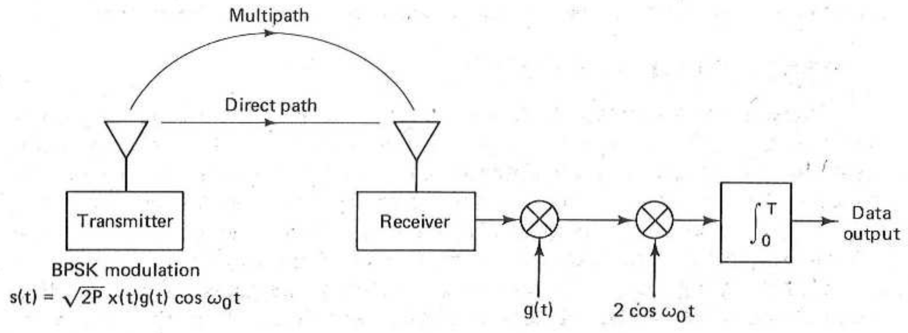


Figure 10.27 Direct-sequence BPSK system operating over a multipath channel.

relative to the direct path signal. For the receiver, synchronized to the direct path signal, the output of the correlator, $z(t = T)$, can be written as

$$z(t = T) = \int_0^T [Ax(t)g^2(t) \cos \omega_0 t$$

$$+ \alpha Ax(t - \tau)g(t)g(t - \tau) \cos (\omega_0 t + \theta) + n(t)g(t)]2 \cos \omega_0 t \, dt \quad (10.41)$$

where $g^2(t) = 1$. Also, for $\tau > T_c$, $g(t)g(t - \tau) \approx 0$ (for codes with long periods), where T_c is the chip duration. Therefore, if T_c is less than the differential time delay between the multipath and direct path signals, we can write

$$z(t = T) = \int_0^T 2Ax(t)\cos^2 \omega_0 t + 2n(t)g(t)\cos \omega_0 t \, dt = Ax(T) + n_0(T) \quad (10.42)$$

where $n_0(T)$ is a zero-mean Gaussian random variable. We see that the spread-spectrum system, similar to the case of CDMA, effectively eliminates the multipath interference by virtue of its code-correlation receiver.

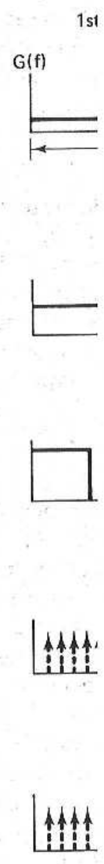
If frequency hopping (FH) is used against the multipath problem, improvement in system performance is also possible but through a different mechanism. FH receivers avoid multipath losses by rapid changes in the transmitter frequency band, thus avoiding the interference by changing the receiver band position before the arrival of the multipath signal.

10.6.3 The Jamming Game

The goals of a jammer are to deny reliable communications to his adversary and to accomplish this at minimum cost. The goals of the communicator are to develop a jam-resistant communication system under the following assumptions: (1) complete invulnerability is not possible; (2) the jammer has a priori knowledge of most system parameters, such as frequency bands, timing, traffic, and so on; (3) the jammer has *no* a priori knowledge of the PN spreading or hopping codes. The signaling waveform should be designed so that the jammer cannot gain any appreciable jamming advantage by choosing a jammer waveform and strategy other than wideband Gaussian noise (i.e., being clever should gain nothing for the jammer). The fundamental design rule in specifying a jam-resistant system is to make it as costly as possible for the jammer to succeed in jamming the system.

10.6.3.1 Jammer Waveforms

There are many different waveforms that can be used for jamming communication systems. The most appropriate choice depends on the targeted system. Figure 10.28 shows power spectral density plots of examples of jammer waveforms versus a communicator's frequency hopped M-ary FSK (FH/MFSK) tone. The range of the abscissa represents the spread-spectrum bandwidth W_{ss}. The three columns in the figure represent three instances in time (three hop times) when symbols having spectra G_1, G_2, and G_3, respectively, are being transmitted. Figure 10.28a illustrates a relatively low-level noise jammer occupying the full spread-spectrum bandwidth. In Figure 10.28b the jammer strategy is to trade bandwidth



Figur
Stepp

occupancy
curve, rem
does not al
the effect c
to jam only
increased, l
at random
niques to a
tones, inst
and steppe
FH system

to the direct path
s

os $\omega_0 t \, dt$ (10.41)

vith long periods),
e differential time
rite

$) + n_0(T)$ (10.42)

e that the spread-
liminates the mul-

problem, improve-
ferent mechanism.
nsmitter frequency
and position before

his adversary and
ator are to develop
umptions: (1) com-
knowledge of most
and so on; (3) the
opping codes. The
annot gain any ap-
and strategy other
nothing for the jam-
t system is to make
the system.

for jamming com-
he targeted system.
jammer waveforms
/MFSK) tone. The
idth W_{ss}. The three
e hop times) when
ng transmitted. Fig-
ying the full spread-
to trade bandwidth

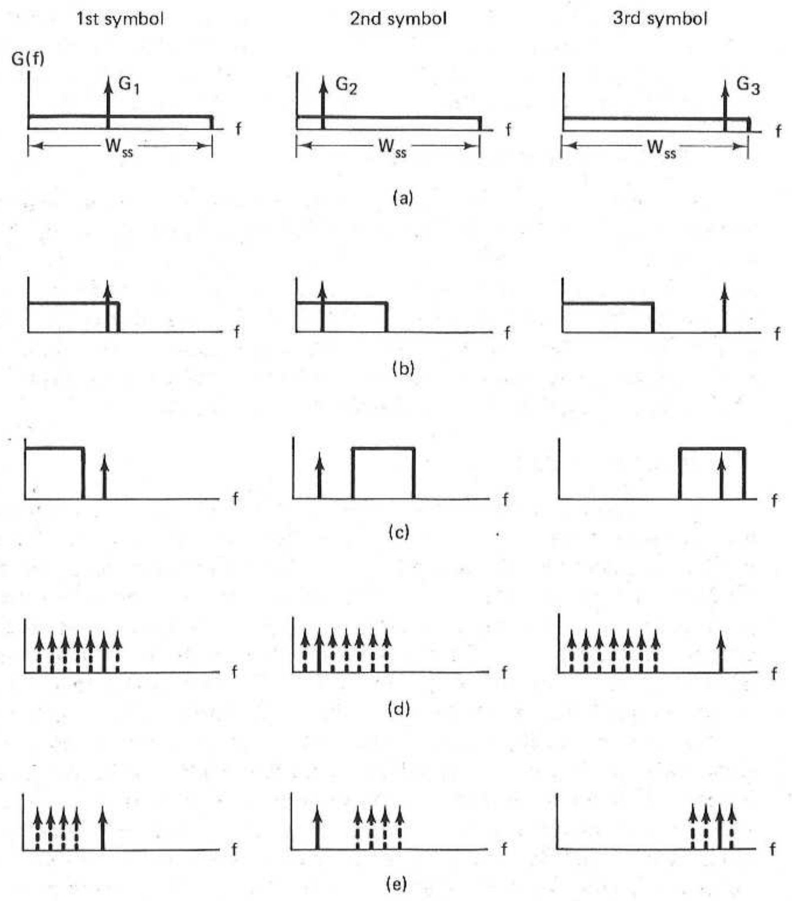


Figure 10.28 Jammer waveforms. (a) Full-band noise. (b) Partial-band noise. (c) Stepped noise. (d) Partial-band tones. (e) Stepped tones.

occupancy for greater power spectral density (the total power, or area under the curve, remains the same). The figure indicates that in this case, the jammer noise does not always share the same bandwidth region as the signal, but when it does, the effect can be destructive. In Figure 10.28c the noise jammer strategy is again to jam only part of the band, so that the jammer power spectral density can be increased, but in this case the jammer steps through different regions of the band at random times, thus preventing the communicator from using adaptive techniques to avoid the jamming. In Figure 10.28d and e the jammer uses a group of tones, instead of a continuous frequency band, in partial-band (Figure 10.28d) and stepped fashion (Figure 10.28e). This is a technique most often used against FH systems. Another jamming technique, not shown in Figure 10.28, is a pulse jammer, consisting of pulse-modulated bandlimited noise. Unless otherwise stated, we shall assume that the jammer waveform is wideband noise and that the jammer strategy is to jam the entire bandwidth W_{ss} continuously. The effects of partial band jamming and pulse jamming are considered later.

10.6.3.2 Tools of the Communicator

The usual design goal for an anti-jam (AJ) communication system is to force a jammer to expend its resources over (1) a wide-frequency band, (2) for a maximum time, and (3) from a diversity of sites. The most prevalent design options are (1) frequency diversity, by the use of direct-sequence and frequency hopping spread-spectrum techniques; (2) time diversity, by the use of time hopping; (3) spatial discrimination, by the use of a narrow-beam antenna which forces a jammer to enter the receiver via an antenna sidelobe and hence suffer, typically, a 20- to 25-dB disadvantage, and (4) combinations of the above.

10.6.3.3 *J/S* Ratio

In Chapter 4 we were concerned primarily with link error performance as a function of thermal noise interference. Emphasis was placed on the signal-to-noise ratio parameters—required E_b/N_0 and available E_b/N_0 for meeting a specified error performance. In this section we are similarly concerned with link error performance as a function of interference. However, here the source of interference is the noise power of a jammer in addition to thermal noise. Therefore, the SNR of interest is $E_b/(N_0 + J_0)$, where J_0 is the noise power spectral density due to the jammer. Unless otherwise specified, J_0 is assumed equal to J/W_{ss}, where J is the average received jammer power (jammer power referred to the receiver front end) and W_{ss} is the spread-spectrum bandwidth. Since the jammer power is generally much greater than the thermal noise power, the SNR of interest in a jammed environment is usually taken to be E_b/J_0. Therefore, similar to the thermal noise case, we define $(E_b/J_0)_{\text{reqd}}$ as the bit energy per jammer noise power spectral density *required* for maintaining the link at a specified error probability. The parameter E_b can be written as

$$E_b = ST_b = \frac{S}{R}$$

where S is the received signal power, T_b the bit duration, and R the data rate in bits/s. Then we can express $(E_b/J_0)_{\text{reqd}}$ as

$$\left(\frac{E_b}{J_0}\right)_{\text{reqd}} = \left(\frac{S/R}{J/W_{ss}}\right)_{\text{reqd}} = \frac{W_{ss}/R}{(J/S)_{\text{reqd}}} = \frac{G_p}{(J/S)_{\text{reqd}}} \tag{10.43}$$

where $G_p = W_{ss}/R$ is denoted the *processing gain,* and $(J/S)_{\text{reqd}}$ can be written

$$\left(\frac{J}{S}\right)_{\text{reqd}} = \frac{G_p}{(E_b/J_0)_{\text{reqd}}} \tag{10.44}$$

The ratio $(J/S)_{\text{reqd}}$ is a figure of merit that provides a measure of how *invulnerable*

Unless otherwise
nd noise and that
ously. The effects
:r.

system is to force
ınd, (2) for a max-
ent design options
requency hopping
time hopping; (3)
ch forces a jammer
typically, a 20- to

or performance as
d on the signal-to-
or meeting a spec-
ned with link error
source of interfer-
ise. Therefore, the
pectral density due
al to J/W_{ss}, where
red to the receiver
the jammer power
NR of interest in a
nilar to the thermal
oise power spectral
ır probability. The

l R the data rate in

— (10.43)
eqd

reqd can be written

(10.44)

of how *invulnerable*

a system is to interference. Which system has better jammer-rejection capability: one with a larger $(J/S)_{\text{reqd}}$ or a smaller $(J/S)_{\text{reqd}}$? The *larger* the $(J/S)_{\text{reqd}}$, the *greater* is the system's noise rejection capability, since this figure of merit describes how much noise power relative to signal power is *required* in order to degrade the system's specified error performance. Of course, the communicator would like the communication system *not* to degrade at all.

Another way of describing the relationship in Equation (10.44) is as follows. An adversary would like to employ a jamming strategy that forces the effective $(E_b/J_0)_{\text{reqd}}$ to be as large as possible. The adversary may employ pulse, tone, or partial-band jamming rather than wideband noise jamming. A large $(E_b/J_0)_{\text{reqd}}$ implies a small $(J/S)_{\text{reqd}}$ ratio for a fixed processing gain. This may force the communicator to employ a larger processing gain to increase the $(J/S)_{\text{reqd}}$. The system designer strives to choose a signaling waveform such that the jammer can gain no special advantage by using a jamming strategy other than wideband Gaussian noise.

10.6.3.4 Anti-Jam Margin

Sometimes the $(J/S)_{\text{reqd}}$ ratio is referred to as the *anti-jam* (AJ) *margin* since it characterizes the system jammer-rejection capability. But this is not really a good use of the phrase since AJ margin usually means the safety margin against a *particular threat*. Using the same approach as in Chapter 4 (for calculating the margin against thermal noise), we can define the AJ margin M_{AJ}, as follows:

$$M_{\text{AJ}}(\text{dB}) = \left(\frac{E_b}{J_0}\right)_r (\text{dB}) - \left(\frac{E_b}{J_0}\right)_{\text{reqd}} (\text{dB}) \qquad (10.45)$$

where $(E_b/J_0)_r$ is the E_b/J_0 *actually received*. Following the same format as Equation (10.43), we can express $(E_b/J_0)_r$ as

$$\left(\frac{E_b}{J_0}\right)_r = \frac{G_p}{(J/S)_r} \qquad (10.46)$$

where $(J/S)_r$, or simply J/S, is the ratio of the actually received jammer power to signal power. We can now combine Equations (10.43), (10.45), and (10.46), as follows:

$$M_{\text{AJ}}\ (\text{dB}) = \frac{G_p}{(J/S)_r} (\text{dB}) - \frac{G_p}{(J/S)_{\text{reqd}}} (\text{dB}) \qquad (10.47)$$

$$= \left(\frac{J}{S}\right)_{\text{reqd}} (\text{dB}) - \left(\frac{J}{S}\right)_r (\text{dB}) \qquad (10.48)$$

Example 10.2 Satellite Jamming

Figure 10.29 illustrates a satellite jamming scenario. The airplane terminal is equipped with a frequency hopping (FH) spread-spectrum system transmitting with an $\text{EIRP}_T = 20$ dBW. The data rate is $R = 100$ bits/s. The jammer is transmitting wideband Gaussian noise, continually, with an $\text{EIRP}_J = 60$ dBW. Assume that $(E_b/J_0)_{\text{reqd}} = 10$ dB and that the path loss is identical for both the airplane terminal and the jammer.

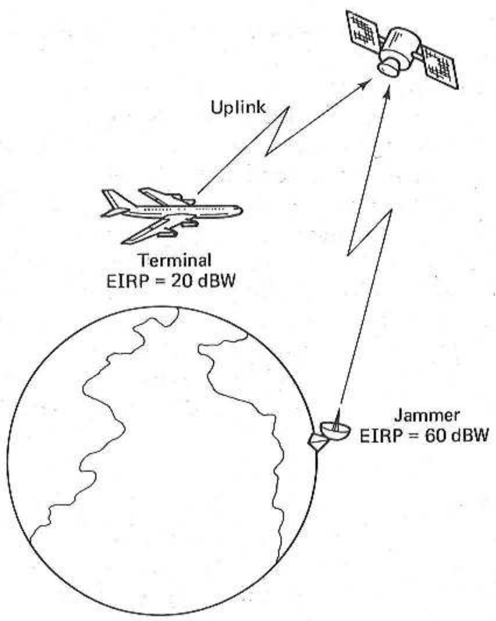


Figure 10.29 Satellite jamming scenario.

(a) Should the communicators be concerned more with the jamming of the uplink or with that of the downlink?
(b) If it is desired to have an AJ margin of 20 dB, what should be the value of the hopping bandwidth W_{ss}?

Solution

(a) Jamming the uplink is of much greater concern, since such single-point interference could degrade the communications of a multitude of terminals that are simultaneously using the satellite transponder. To achieve an equivalent degradation by jamming the downlink, the jammer would have to jam each of the receiving terminals. Downlink jamming is of some concern for critical military missions, but of less concern than uplink jamming.
(b) With the assumption that the path loss is the same for both the communicator and the jammer, we can replace $(J/S)_r$ in Equation (10.48) with the ratio of *transmitted* jammer-to-signal power, $\text{EIRP}_J/\text{EIRP}_T$. Therefore, we can write

$$\begin{aligned} M_{\text{AJ}}\ (\text{dB}) &= (J/S)_{\text{reqd}}\ (\text{dB}) + \text{EIRP}_T\ (\text{dBW}) - \text{EIRP}_J\ (\text{dBW}) \\ &= G_p\ (\text{dB}) - \left(\frac{E_b}{J_0}\right)_{\text{reqd}} (\text{dB}) + \text{EIRP}_T\ (\text{dBW}) - \text{EIRP}_J\ (\text{dBW}) \\ G_p &= 20\ \text{dB} + 10\ \text{dB} - 20\ \text{dBW} + 60\ \text{dBW} = 70\ \text{dB} \\ W_{ss} &= G_p\ (\text{dB}) + R\ (\text{dB-Hz}) = 70\ \text{dB} + 20\ \text{dB-Hz} \\ &= 90\ \text{dB-Hz} = 1\ \text{GHz} \end{aligned}$$

Example 1

In E
and
the
ence
the
tern
is L
mar

Solı

For
airp
sho

whe

and

Cor

10.7 FURTHER

10.7.1 E

If the jaı
noise pr
terest, t
density
occupy.
W_{ss}, wit
jammer,

In

atellite jamming scenario.

jamming of the uplink

ıld be the value of the

ıuch single-point inter-
e of terminals that are
ːve an equivalent deg-
ave to jam each of the
ərn for critical military

ɔoth the communicator
with the ratio of *trans-*
, we can write

(dBW)

V) − EIRP$_J$ (dBW)

= 70 dB

-Hz

Example 10.3 Satellite Downlink Jamming

In Example 10.2 the distance from the transmitting airplane to the receiving satellite and the distance from the jammer to the satellite were assumed identical. Certainly, the closer the jammer gets to the receiver, the greater will be the jamming interference. Consider a downlink jamming scenario where the satellite EIRP$_s$ = 35 dBW, the jammer EIRP$_J$ = 60 dBW, the space loss from the satellite to the receiving terminal is L_s = 200 dB, and the space loss from the jammer to the receiving terminal is L_s' = 160 dB. How much processing gain is needed to close the link with an AJ margin of 0 dB? Assume that $(E_b/J_0)_{\text{reqd}}$ = 10 dB.

Solution

For the downlink jamming scenario the proximity of the jammer to the receiving airplane is much closer than that of the satellite to the airplane. These distances show up as the space losses in the (J/S)$_r$ term of Equation (10.48), as follows:

$$M_{\text{AJ}}\ (\text{dB}) = \left(\frac{J}{S}\right)_{\text{reqd}} (\text{dB}) - \left(\frac{J}{S}\right)_r (\text{dB})$$

where

$$\left(\frac{J}{S}\right)_r (\text{dB}) = \text{EIRP}_J\ (\text{dBW}) - L_s'\ (\text{dB}) - \text{EIRP}_s\ (\text{dBW}) + L_s\ (\text{dB})$$

and

$$\left(\frac{J}{S}\right)_{\text{reqd}} (\text{dB}) = \frac{W_{ss}}{R}\ (\text{dB}) - \left(\frac{E_b}{J_0}\right)_{\text{reqd}} (\text{dB})$$

Combining the above equations, and solving for processing gain, $G_p = W_{ss}/R$, yields

$$G_p\ (\text{dB}) = 75\ \text{dB}$$

10.7 FURTHER JAMMING CONSIDERATIONS

10.7.1 Broadband Noise Jamming

If the jamming signal is modeled as a zero-mean wide-sense-stationary Gaussian noise process with a flat power spectral density over the frequency range of interest, then for a fixed jammer received power, J, the jammer power spectral density J_0' is equal to J/W, where W is the bandwidth that the jammer chooses to occupy. If the jammer strategy is to jam the entire spread-spectrum bandwidth, W_{ss}, with its fixed power, the jammer is referred to as a wideband or *broadband jammer,* and the jammer power spectral density is

$$J_0 = \frac{J}{W_{ss}} \tag{10.49}$$

In Chapter 3 it was shown that the bit error probability P_B for a coherently