universal personal communications

Hiroshi Harada
Ramjee Prasad

simulation and software radio

for mobile communications

CD-ROM INCLUDED

Contents

Preface

ज्ञानं ज्ञेयं परिज्ञाता त्रिविधा कर्मचोदना ।
करणं कर्म कर्तेति त्रिविधः कर्मसंग्रहः ॥

jñānaṁ jñeyaṁ parijñātā
tri-vidhā karma-codanā
karaṇaṁ karma karteti
tri-vidhaḥ karma-saṅgrahaḥ

> Knowledge, the object of knowledge, and the knower are the three factors that motivate the action; the senses, the work, and the doer are the three constituents of action.
>
> —*The Bhagvad Gita (18.18)*

From September 1996 to August 1997, we worked together at the Delft University of Technology, The Netherlands, in order to realize future wireless personal multimedia communications (WPMC). During our joint research period, research and development were done on higher layers such as the media access control (MAC) layer and the physical layer, and many researchers came from Europe, Asia, the United States, and The Netherlands to work with us. We also started new projects and established a joint cooperation with several research institutes, companies, and universities.

Once a project is defined, many new researchers are allocated. Among the researchers, some newcomers are occasionally involved. Moreover, most projects are time-limited. In this situation, the answer to the question of how newcomers prepare to evaluate the target is one of key issues to success in the time-limited project. We considered the answer carefully.

When we instructed the newcomers on how to evaluate a communication system, we used the following procedure. First, the newcomers read some books about a basic communication theory for the target topic to obtain basic knowledge for the target topic on the project. Then they read many writings about the target topic to understand the state-of-the-art technology, defined the remaining problems, and found new solutions suitable for the target topic. Then, once the newcomers set their research direction, by using references on how to use computer languages, they began their own programs to design and evaluate the target communication system.

This procedure may seem quite logical. However, during the procedure, newcomers have many questions, which take too long to discuss. Moreover, when we viewed our research field, we found many excellent books on the basic communication theory. However, there are few books on how to design a telecommunications system using computer simulation. There is currently no single book on how to design and evaluate a telecommunications system from the physical to the upper layer.

It is too time-consuming to prepare the newcomers to design and evaluate their own communication system. We therefore decided to write a new book to describe how to design telecommunications systems and evaluate them from the physical layer to the upper layer. In writing this book, we have attempted to set our concept as follows. In each chapter, we have described a simple explanation for a target telecommunications system, before showing programs. The explanation is quite simple. If you need more concrete explanation, you can find many excellent books on the subject. We chose MATLAB, one of the most popular computer simulation languages in the world, as the computer language to design the telecommunications systems. Moreover, we showed source programs in this book and included them on the accompanying CD-ROM. The users can customize our programs to their favorite systems. We believe that this book and accompanying CD-ROM are a must-have for all engineers, researchers, academics, and students of telecommunications technology.

Chapter 1 presents a general introduction to the history of the wireless communication system and the latest information on WPMC. This chapter also describes why we need to evaluate the performance of telecommunications system by computer simulation and why we can realize the simulation.

Chapter 2 describes several key parameters to perform computer simulations smoothly. MATLAB, a good software simulation tool, is mainly used in this book. Therefore, we first describe how to use the MATLAB language. We summarize frequently used commands and functions and the methods of creating a hierarchical program, in addition to the methods of programming function blocks, which are commonly used to evaluate all communication systems.

Chapter 3 explains the basic configurations of the phase shift keying (PSK)–based digital radio transmission scheme and describes the method of evaluating transmission performance by computer simulation. The PSK-based digital radio transmission schemes—binary phase shift keying (BPSK), quadrature phase shift keying (QPSK), offset QPSK (OQPSK), minimum shift keying (MSK), and Gaussian-filtered minimum shift keying (GMSK)—and quadrature amplitude modulation (QAM) are introduced and their performances are evaluated by creating computer simulation programs. Parts of the programs are based on contributions from Takako Yamamura of the National Police Agency in Tokyo, Japan, and Ryo Sawai and Ryuhei Funada of Chuo University, also in Tokyo, Japan.

Chapter 4 presents the configuration of the orthogonal frequency division multiplexing (OFDM) transmission scheme, which can reduce the influence of multipath fading and realize broadband communication while retaining high-frequency utilization efficiency. This chapter also describes the method of simulating transmission performance using computer simulation programs. Takako Yamamura has also contributed to parts of the programs in this chapter.

Chapter 5 presents the configuration of code division multiplexing (CDM), which can retain robustness against multipath fading and is used in third-generation mobile communication systems. As did Chapter 4, this chapter also describes the method of simulating transmission performance using computer simulation programs. Makoto Okita from the National Police Agency in Tokyo also contributed to parts of the programs in this chapter.

Chapter 6 evaluates the transmission performance of a point-to-multipoint communication system with multiple-access protocols by computer simulation. Pure ALOHA, slotted ALOHA, nonpersistent carrier-sense multiple access (CSMA), and slotted nonpersistent inhibit sense multiple access (ISMA) are explained as examples of multiple-access protocols. In this chapter, the methods of simulating throughput and average delay time are described using computer simulation programs. This chapter and its programs contain contributions and suggestions from Makoto Okita.

Chapter 7 describes the basic simulation method for a multipoint-to-multipoint communication system based on a cellular telecommunications system. The dynamic channel assignment (DCA), fixed channel assignment (FCA), and adaptive cellular zone configuration (AZC) algorithms using an adaptive antenna are introduced. In this chapter, the method used to simulate call-blocking probability is described by using computer simulation programs. This chapter and all of its programs contain contributions from Fumihide Kojima and Ami Harada from the Communication Research Laboratory, Independent Administrative Institution, in Japan.

Chapter 8 describes a software radio communication system as a future application of the software programming method presented in this book. Computer simulation languages have a good relationship with software languages that configure digital signal processing hardware (DSPH) such as a digital signal processor (DSP) and/or field programmable gate array (FPGA) and/or application-specific integrated circuit (ASIC). The concept of software radio bridges the software programming examined in this book and real hardware implementation.

To make the learning process fast and easy, MATLAB code is provided at the end of each chapter and on the accompanying CD-ROM. This CD-ROM will allow users to exercise the simulations for real without having to write their own programs, which is always a barrier and prone to error. One of the main strengths of this book is that the learning process can be hands-on, allowing the readers to see the effects of varying parameters on the output of the simulation, a great aid to learning. Providing the MATLAB code at the end of each chapter is very helpful for readers. This book has a great deal to offer to researchers, practicing engineers, and to everyone in the field of wireless information and multimedia communication.

Finally, we would like to state that this book is a work in progress. We wish to develop new discussions about evaluating the communication system through worldwide computer simulations. If you have any comments or questions, please let us know. Through your comments, this book will evolve with the future. Please enjoy this book.

Acknowledgments

The material in this book is based on Dr. Harada's work during his post- doctoral fellowship at the Delft University of Technology, The Netherlands, supervised by Dr. Prasad. Based on Dr. Harada's work, the material developed into a book by adding comments and new computer simulation programs from many researchers in the Communications Research Laboratory, Independent Administrative Institution in Japan, and several other institutions.

We wish to thank Takako Yamamura and Makoto Okita from the National Police Agency in Tokyo, Japan; Fumihide Kojima and Ami Harada, from the Communications Research Laboratory, Independent Administrative Institution; and Ryo Sawai and Ryuhei Funada from Chuo University in Tokyo, Japan, for their valuable contributions to this book.

We also wish to thank Anand Raghawa Prasad from Genista Co. Ltd. in Japan, Junko Prasad, and Izumi Otsuka for helping to make our text more readable. Thanks also to Rajeev Prasad.

Hiroshi would like to thank Seiichi Sampei from Osaka University in Osaka, Japan; Yukiyoshi Kamio from the Communications Research Laboratory, Independent Administrative Institution; Sadayuki Abeta from NTT Docomo in Yokosuka, Japan; Shinya Ohtsuki from NTT in Yokosuka, Japan; and Toyoki Ue from Matsushita Communication Industrial, Co. Ltd. for their instructive suggestions for computer simulation.

In addition, we would like to sincerely thank Shingo Ohmori, Toshio Ihara, Yoshihiro Hase, and Masayuki Fujise, all from the Communication Research Laboratory, Independent Administrative Institution, for their encouragement.

We believe that exploiting this book removes the technical borders and technical differences in the research area of wireless telecommunication engineering among all over the world.

1

Introduction

This book describes a method for evaluating the wireless communication systems by computer simulation. This chapter shows how evaluations by computer simulation are beneficial in wireless communication research by reviewing the history of a mobile communication system that is representative of wireless communication systems. Then, the significance of evaluation by computer simulation is described. Finally, this chapter describes the scope of this book and presents some figures that describe the process of evaluation by computer simulation for wireless communication systems.

1.1 History of Mobile Communications

The prosperous progress of mobile communication has built the main road of the history of wireless communication. Historically, when a mobile communication system was to be standardized, many system proposals were submitted to standardization bodies. Then, these proposals were equally evaluated by using computer simulations or developed prototypes. Finally, one or a few standardized systems were chosen. The standardized mobile communication systems included many important concepts, and thus, a historical survey is the best way to understand the key points of recent mobile communication systems.

The history of mobile communications can be categorized into three periods: (1) the pioneer era, (2) the precellular era, and (3) the cellular era [1]. Table 1.1 summarizes several representative events in each era.

Table 1.1
History of Mobile Communications

Time	Significant Event
Pioneer Era	
1860s	James Clark Maxwell's *electromagnetic* (EM) wave postulates
1880s	Proof of the existence of EM waves by Heinrich Rudolf Hertz
1890s	First use of wireless and first patent of wireless communications by Gugliemo Marconi
1905	First transmission of speech and music via a wireless link by Reginald Fessenden
1912	Sinking of the *Titanic* highlights the importance of wireless communication on the seaways; in the following years marine radio telegraphy is established
Precellular Era	
1921	Detroit Police Department conducts field tests with mobile radio
1933	In the United States, four channels in the 30–40-MHz range
1938	In the United States, rules for regular services
1940	Wireless communication is stimulated by World War II
1946	First commercial mobile telephone system operated by the Bell system and deployed in St. Louis
1948	First commercial fully automatic mobile telephone system is deployed in Richmond, Virginia, in the United States
1950s	Microwave telephone and communication links are developed
1960s	Introduction of trunked radio systems with automatic channel allocation capabilities in the United States
1970	Commercial mobile telephone system operated in many countries (e.g., 100 million moving vehicles on U.S. highways, "B-Netz" in West Germany)
Cellular Era	
1980s	Deployment of analog cellular systems
1990s	Digital cellular deployment and dual-mode operation of digital systems
2000s	*Future public land mobile telecommunication systems* (FPLMTSs)/*international mobile telecommunications-2000* (IMT-2000)/*universal mobile telecommunication systems* (UMTS) will be deployed with multimedia services
2010s	*Fixed-point* (FP)–based wireless broadband communications and software radio will be available over the Internet
2010s+	Radio over fiber (such as fiber-optic microcells) will be available

In the pioneer era, a great deal of the fundamental research and development in the field of wireless communications took place. The postulates of *electromagnetic* (EM) waves by James Clark Maxwell during the 1860s in England, the demonstration of the existence of these waves by Heinrich Rudolf Hertz in the 1880s in Germany, and the invention and first demonstration of wireless telegraphy by Guglielmo Marconi during the 1890s in Italy were representative examples from Europe [2, 3]. Moreover, in Japan, the Radio Telegraph Research Division was established as a part of the Electrotechnical Laboratory at the Ministry of Communications and started to research wireless telegraphy in 1896.

From the above fundamental research and the resultant developments in wireless telegraphy, the application of wireless telegraphy to mobile communication systems started from the 1920s. The period, which is called the precellular era, began with the first land-based mobile wireless telephone system installed in 1921 by the Detroit Police Department to dispatch patrol cars, followed in 1932 by the New York City Police Department [4]. These systems were operated in the 2-MHz frequency band [4]. Unfortunately, during World War II, the progress of radio communication technologies was drastically stimulated.

In 1946, however, the first commercial mobile telephone system, which operated in the 150-MHz frequency band, was set up by Bell Telephone Laboratories in St. Louis [1, 4]. The demonstration system was a simple analog communication system with a manually operated telephone exchange.

Subsequently, in 1969, a mobile duplex communication system was realized in the 450-MHz frequency band. The telephone exchange of this modified system was operated automatically [5]. The new system, called the *improved mobile telephone system* (IMTS), was widely installed in the United States. However, because of its large coverage area, the system could not manage a large number of users or allocate the available frequency bands efficiently.

The cellular zone concept was developed to overcome this problem by using the propagation characteristics of radio waves. The concept is shown in Figure 1.1. A frequency channel in one cellular zone is used in another zone. However, the distance between the cellular zones that use the same frequency channels is sufficiently long to ensure that the probability of interference is quite low. The use of the new cellular zone concept launched the third era, known as the cellular era.

The first generation of cellular mobile communication was developed from 1980 to 1990. In this period, research and development (R&D) centered around analog cellular communication systems. Table 1.2 summarizes these analog cellular communication systems.

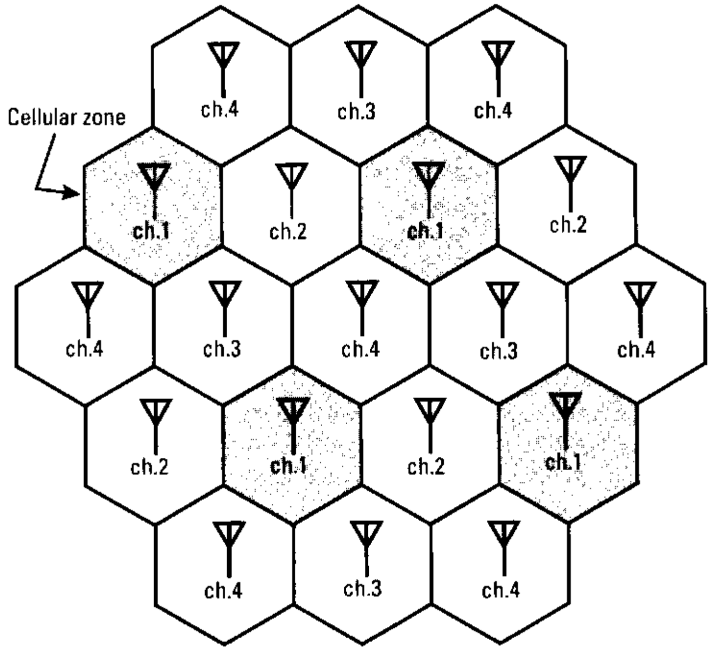


Figure 1.1 Cellular zone concept.

In the United States, an analog cellular mobile communication service called advanced mobile phone service (AMPS) was started in October 1983 in Chicago [6].

In Europe, several cellular mobile communication services were started. In Norway, *Nordic Mobile Telephones* (NMT) succeeded in the development of an analog cellular mobile communication system: NMT-450 [7].

In the United Kingdom, Motorola developed an analog cellular mobile communication system called the *total access communication system* (TACS) based on AMPS in the 1984–1985 period. In 1983, NMT started a modified NMT-450 called NMT-900. Moreover, C-450, RTMS, and Radiocom-2000 were, respectively, introduced in Germany, Italy, and France.

Meanwhile, in Japan, *Nippon Telephone and Telegraph* (NTT) developed a cellular mobile communication system in the 800-MHz frequency band and started service in Tokyo in December 1979. Furthermore, a modified TACS that changed the frequency band to adjust for Japanese frequency planning and was called JTACS was also introduced in July 1989. Subsequently, *narrowband TACS* (NTACS), which reduced the required frequency band in half, started service in October 1991.

Table 1.2
Summary of Analog Cellular Radio Systems

System	**AMPS**	**NMT-450**	**NMT-900**	**TACS**	**ETACS**
Frequency range (mobile Tx/base Tx) (MHz)	824–849/ 869–894	453–457.5/ 463–467.5	890–915/ 463–467.5	890–915/ 935–960	872–905/ 917–950
Channel spacing (kHz)	30	25	12.5*	2.5	2.5
Number of channels	832	180	1,999	1,000	1,240
Region	The Americas, Australia, China, Southeast Asia	Europe	Europe, China, India, Africa	United Kingdom	Europe, Africa

System	**C-450**	**RTMS**	**Radiocom-2000**	**JTACS/ NTACS**	**NTT**
Frequency range (mobile Tx/base) (MHz)	450–455.74/ 460–465.74	450–455/ 460–465	165.2–168.4/ 169.8–173 192.5–199.5/ 200.5–207.5 215.5–233.5/ 207.5–215.5 414.8–418/ 424.8–428	915–925/ 860–870 898–901/ 843–846 918.5–922/ 863.5–867	925–940/ 870–855 915–918.5/ 860–863.5 922–925/ 867–870
Channel spacing (kHz)	10*	25	12.5	25/12.5* 25/12.5* 12.5*	25/6.25* 6.25* 6.25*
Number of channels	573	200	256 560 640 256	400/800 120/240 280	600/2400 560 480
Region	Germany, Portugal	Italy	France	Japan	Japan

*Frequency interleaving using overlapping or "interstitial" channels; the channel spacing is half the nominal channel bandwidth.

So far, we have described the evolution of the analog cellular mobile communication system. However, the incompatibility of the various systems precluded roaming. This meant that users had to change their mobile ter- minals when they moved to another country. In addition, analog cellular mobile communication systems were unable to ensure sufficient capacity for the increasing number of users, and the speech quality was not good.

To solve these problems, the R&D of cellular mobile communication systems based on the digital radio transmission scheme was initiated. These new mobile communication systems became known as the second generation of mobile communication systems, and the analog cellular era thus is regarded as the first generation of mobile communication systems. Table 1.3 summarizes digital cellular radio systems.

In Europe, the global system for mobile communication (GSM), a new digital cellular communication system that allowed international roaming and used the 900-MHz frequency band, started service in 1992. In 1994, DCS-1800, a modified GSM that used the 1.8-GHz frequency band, was launched.

In North America, the IS-54 digital cellular communication system was standardized in 1989. Subsequently, the standard was revised to include dual-mode services between analog and digital cellular communication systems and reintroduced in 1993 with the title DAMPS, or IS-136. In addition, IS-95, which was the first standardized system based on *code-division multiple access* (CDMA), started service in 1993.

In Japan, the digital cellular communication or *personal digital cellular* (PDC) systems using the 800- and 1.5-GHz frequency bands started service in 1993 and 1994, respectively.

In addition to the above digital systems, the development of new digital cordless technologies gave birth to the second-supplement-generation systems, namely, the *personal handy-phone systems* (PHSs)—formerly PHPs—in Japan, the *digital enhanced* (formerly European) *cordless telephone* (DECT) in Europe, and *personal access communication services* (PACSs) in North America. Table 1.4 summarizes the second-supplement-generation systems [8, 9] and shows the *cordless telecommunications, second generation* (CT2) and CT2+. A detailed description of CT2 can be found in [10, 11], where CT2+ is a Canadian enhancement of the CT2 common air interface.

In the second generation of mobile communication systems, the common standardizations of some regions, such as in Europe and North America, enabled the realization of partial roaming. This feature was a unique point of the second-generation systems in comparison with the first-generation systems. The advent of common standard gave users a sense of ease of international roaming. Users have been eager to see worldwide standardization.

Table 1.3
Summary of Digital Cellular Radio Systems

Systems	**GSM** **DCS-1800**	**IS-54**	**IS-95**	**PDC**

Frequency range (base Rx/Tx, MHz)	GSM:
	Tx: 935–960;
	Rx: 890–915
	DCS-1800:
	Tx: 1,805–1,880;
	Rx: 1,710–1785
		Tx: 869–894;
		Rx: 824–849
			Tx: 869–894;
			Rx: 824–849
				Tx: 810–826;
				Rx: 940–956;
				Tx: 1,429–1,453;
				Rx: 1,477–1,501

Channel spacing (kHz)	200	30	1,250	25
Number of channels	GSM: 124; DCS-1800: 375	832	20	1,600
Number of users per channel	GSM: 8 DCS-1800: 16	3	63	3
Multiple access	TDMA/FDMA	TDMA/FDMA	CDMA/FDMA	TDMA/FDMA
Duplex	FDD	FDD	FDD	FDD
Modulation	GMSK	π/4 DQPSK	BPSK/QPSK	π/4 DQPSK
Speech coding and its rate (Kbps)	RPE-LTP 13	VSELP 7.95	QCELP 8	VSELP 6.7
Channel coding	1/2 Convolutional	1/2 Convolutional	Uplink 1/3 Downlink 1/2 Convolutional	9/17 Convolutional
Region	Europe, China, Australia, Southeast Asia	North America, Indonesia	North America, Australia, Southeast Asia	Japan

Notes: RPE-LTP: regular pulse-exciting long-term predictive coding; VSELP: vector-sum–excited linear predictive coding; GSM: global system for mobile communication.

During the period 1990–2000, the styles of wired communication as well as wireless communication were both changed by the innovation of digital signal processing. During the period, all information such as voice, data, images, and moving images could be digitized, and the digitized data could be transmitted through a worldwide computer network such as the Internet. Mobile users were also eager to be able to transmit such digitized data in a mobile communication network. However, in the second-generation mobile communication systems, the data transmission speeds were limited, creating the need for new

Table 1.4
Summary of Digital Cordless Systems

System	CT2/CT2+	DECT	PHS	PACS
Frequency range (Base Rx/Tx, MHz)	CT2: 864–868 CT2+: 944–948	1,880–1,990	1,895–1,918	Tx: 1,850–1,910 Rx: 1,930–1,990
Channel spacing (kHz)	100	1,728	300	300
Number of channels	40	10	77	96
Number of users per channel	1	12	4	8
Multiple access	FDMA	TDMA/FDMA	TDMA/FDMA	TDMA/FDMA
Duplex	TDD	TDD	TDD	TDD
Modulation	GFSK	GFSK	π/4 DQPSK	π/4 DQPSK
Speech coding	ADPCM	ADPCM	ADPCM	ADPCM
	32	32	32	32
Channel coding	None	CRC	CRC	CRC
Region	Europe, Canada, China, Southeast Asia	Europe	Japan, Hong Kong	United States

high-speed mobile communication systems. Based on this objective, R&D into third-generation mobile communication systems was started in 1995. The R&D that occurred in the 1995–2000 period can be categorized into two areas: (1) international standardized high-speed digital cellular systems with mobility as the second generation and (2) international standardized broadband mobile-access systems with low mobility.

In the first area, IMT-2000 has become the standard. IMT-2000 aims to realize 144 Kbps, 384 Kbps, and 2 Mbps under high-mobility, low-mobility, and stationary environments, respectively. Figure 1.2 shows an image of the IMT-2000 concept.

In IMT-2000, on the basis of CDMA, three radio-access schemes have been standardized: (1) *direct-sequence CDMA* (DSCDMA)-*frequency division duplex* (FDD) (DSCDMA-FDD), (2) *multicarrier CDMA* (MCCDMA)- *FDD* (MCCDMA-FDD), and (3) *direct-sequence CDMA* (DSCDMA)-*time division duplex* (TDD) (DSCDMA-TDD). WCDMA by NTT Docomo and Ericsson and CDMA2000 by Qualcomm were submitted to the ITU [12, 13]. Their basic requirements are shown in Table 1.5. IMT-2000 adopted a CDMA-based system that, by fixing the code transmission rate (chip rate), brought about the capability of offering worldwide roaming. Moreover, since the data transmission rates of third-generation mobile communication systems (144 Kbps–2 Mbps) are much higher than those of the second-generation systems

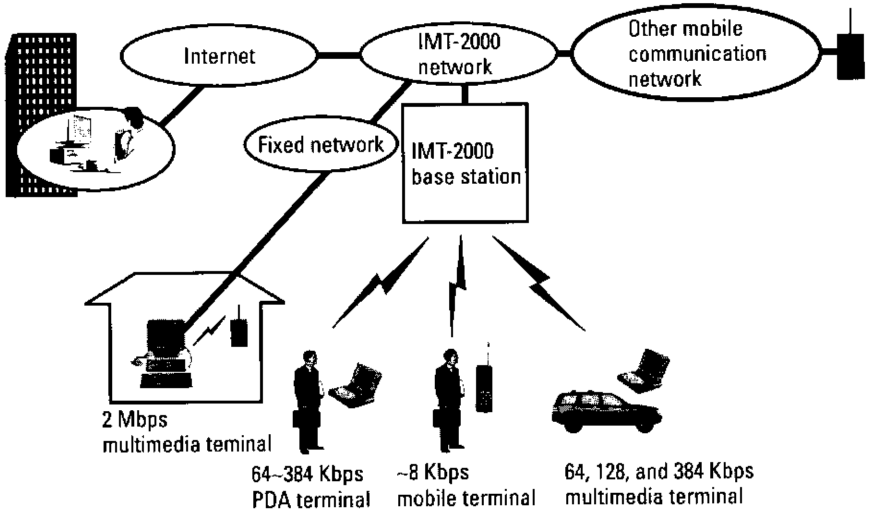


Figure 1.2 Image of IMT-2000.

Table 1.5
Summary of IMT-2000

	WCDMA	CDMA2000
Frequency	2-GHz band	—
Bandwidth	1.25/5/10/20-MHz (DSCDMA)	1.25/5/10/20-MHz (DSCDMA) 3.75/5-MHz (MCCDMA)
Chip rate	3.84-Mcps (DSCDMA-FDD, DSCDMA-TDD)	3.84-Mcps (DSCDMA-FDD) 3.6864-Mcps (MCCDMA-FDD)
Data rate	144-Kbps (high-mobility environment) 384-Kbps (low-mobility environment) 2-Mbps (stationary environment)	—
Synchronization between base stations	Asynchronous Synchronous	Asynchronous synchronous (DSCDMA-TDD) Synchronous (MCCDMA-FDD)
Exchange	GSM network	ANSI-41

(less than 64 Kbps), users can realize moving-image-based communication as well as voice and data communication using a mobile terminal.

Several high-speed wireless-access systems have been standardized [14]. These basic requirements are shown in Table 1.6. Figure 1.3 shows an image of a high-speed wireless access system. As stated in Table 1.6, most standardized systems can realize transmissions of more than 10 Mbps. It is especially so in the 5-GHz frequency band: An *orthogonal frequency-division multiplexing* (OFDM)–based high-speed wireless access system can realize several tens of megabits per second transmission rates [14]. By using such a mobile-access scheme, broadband data transmission rates, such as several tens of megabits per second, can be realized in a wireless communication network as well as a wired network.

Ultra-high-speed wireless access systems that can realize several tens of megabits per second to hundreds of megabits per second as supported data transmission rates are targets of new R&D.

Within the European *Advanced Communication Technologies and Services* (ACTS) program, there were four European Union–funded R&D projects ongoing, namely the Magic Wand [*wireless ATM* (WATM) network demonstration], the *ATM wireless access communication system* (AWACS), the *system for advanced mobile broadband applications* (SAMBA), and the wireless broadband *customer premises local area network* (CPN/LAN) for professional and residential *multimedia applications* (MEDIAN) [14–23].

In the United States, a *seamless wireless network* (SWAN) and a *broadband adaptive homing ATM architecture* (BAHAMA) along with two major projects at Bell Laboratories and the *WATM network* (WATMnet) are being developed in the *computer and communication* (C&C) research laboratories of *Nippon Electric Company* (NEC) in the United States [15–19].

Table 1.6
Summary of Broadband Mobile-Access Systems

	IEEE802.11 2 GHz	**IEEE802.11 5 GHz**	**HiperLAN2**	**MMAC**
Frequency	2.40–2.4835 GHz	5.150–5.350 GHz 5.725–5.825 GHz	5.150–5.350 GHz 5.470–5.725 GHz	5.150–5.25 GHz
Modulation scheme	CDM (DBPSK/DQPSK/CCK)	OFDM (BPSK/QPSK/16-QAM/64-QAM)	OFDM (BPSK/QPSK/16-QAM/64-QAM)	OFDM (BPSK/QPSK/16-QAM/64-QAM)
Channel access	CSMA/CA	CSMA/CA	Scheduled TDMA	DSA
Duplexing	TDD	TDD	TDD	TDD
Data rate	1, 2 Mbps, DBSK, DQSK, 5.5, 11 Mbps (CCK)	6, 9 Mbps BPSK 12, 18 Mbps QPSK 24, 36 Mbps 16-QAM 54 Mbps 64-QAM	6, 9 Mbps BPSK 12, 18 Mbps QPSK 27, 36 Mbps, 16-QAM 54 Mbps 64-QAM	6, 9 Mbps BPSK 12, 18 Mbps QPSK 27, 36 Mbps 16-QAM 54 Mbps 64-QAM
Organization	IEEE	IEEE	ETSI BRAN	ARIB MMAC

Notes: CCK: complimentary code keying; DSA: dynamic slot assignment; BRAN: broadband radio access networks; MMAC: Multimedia Mobile Project Access Communication Systems Promotion Council; IEEE: Institute of Electrical and Electronics Engineers; ETSI: European Telecommunications Standards Institute; ARIB: Association of Radio Industries and Businesses.

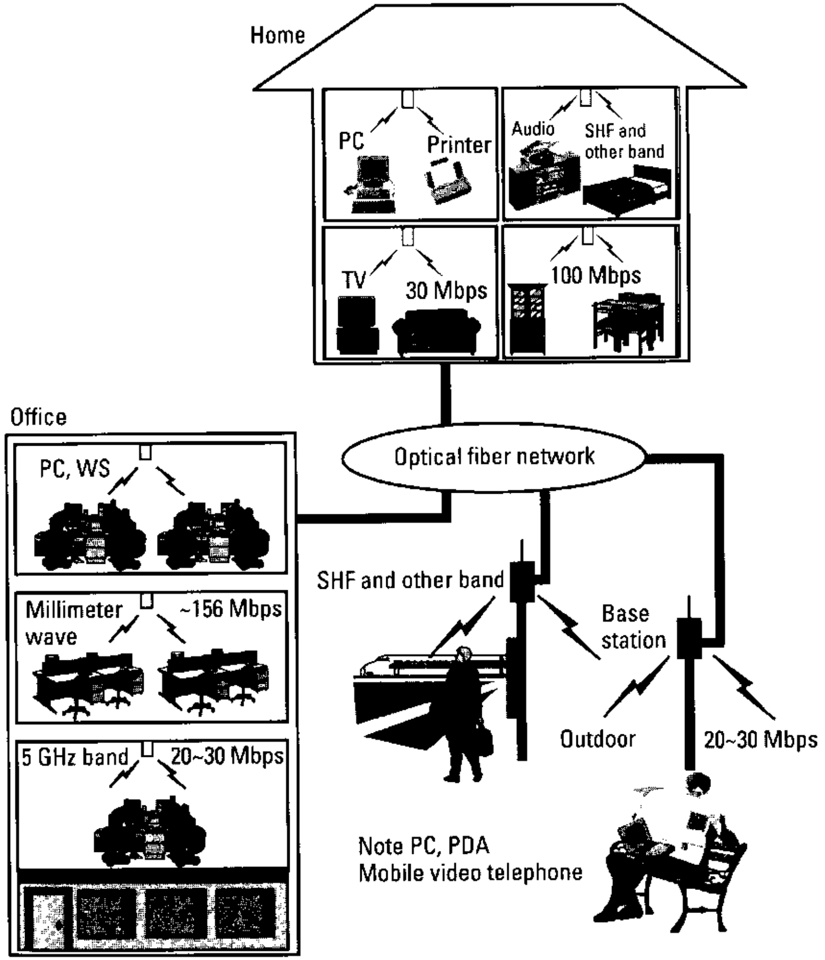


Figure 1.3 Image of a high-speed wireless access system.

In Japan, the *Communications Research Laboratory* (CRL) in the Ministry of Posts and Telecommunications is busy with several R&D projects, such as a broadband mobile communication system [24] in the *super high-frequency* (SHF) band (from 3 to 10 GHz) with a channel bit rate of up to 10 Mbps that achieves 5-Mbps transmission in a high-mobility environment where the vehicle speed is 80 km/hr [25, 26]. Moreover, an indoor high-speed wireless LAN in the millimeter-wave band with a target bit rate of up to 155 Mbps [27, 28] has also been researched, and a point-to-multipoint wireless LAN was developed that can achieve a transmission rate of 156 Mbps by using an original protocol named *reservation-based slotted idle signal multiple access* (RS-ISMA) [29].

As a mobile communication system that requires broadband transmission capability, such as several megabits per second to 10 Mbps, in a high-mobility environment, the *intelligent transport system* (ITS) is the most representative example [30–34].

In the ITS, there are many communication schemes, of which the *global positioning service* (GPS) is the most famous application. However, nowadays, the standardization of the *dedicated short-range communication* (DSRC) system has progressed. The DSRC system uses an *industrial, scientific, and medical* (ISM) band (5.725–5.875 GHz) to realize a short-distance (about up to 30m), vehicle-to-roadside communication system. The image, the applications, and the spectrum allocations for DSRC are shown in Figures 1.4–1.6, respectively [31].

To realize the DSRC, the *Comité Européen de Normalisation* (CEN) in Europe, the *American Society for Testing and Materials* (ASTM) and the IEEE in North America, and ARIB in Japan organized standardization committees for DSRC. As for the data transmission scheme, the *International Telecommunication Union-Radiocommunication* (ITU-R) recommendation M.1453 suggests two methods: the active and backscatter methods [31]. The requirements are shown in Table 1.7 [31]. Based on the recommendation, several applications are being considered. Figure 1.5 shows some examples of the intended applications. Furthermore, a full-mobility and a quasi-mobile communication system are also being considered.

There are many modulation and demodulation schemes, as well as access protocols used in mobile communication as described earlier in this section. The relationship between the first-, second-, and third-generation mobile communication systems, high-speed and ultra-high-speed wireless-access systems, and ITS is shown in Figure 1.7.

We, therefore, sometimes compare the performance of a new system with that of an old one in a common environment. Computer simulation is one of methods used to evaluate the performance of different systems in a common environment.

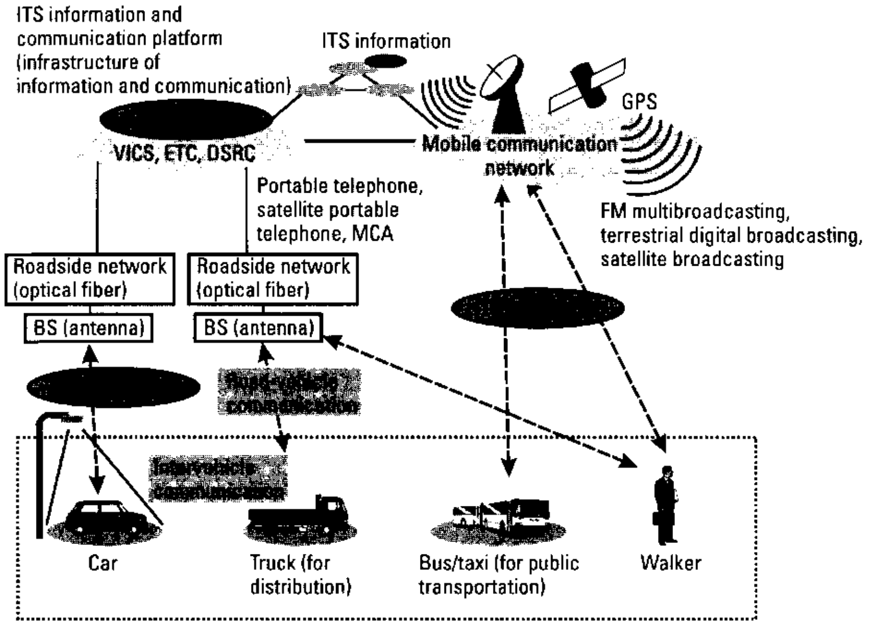

ITS information and communication platform (infrastructure of information and communication)
ITS information
GPS
VICS, ETC, DSRC
Mobile communication network
Portable telephone, satellite portable telephone, MCA
FM multibroadcasting, terrestrial digital broadcasting, satellite broadcasting
Roadside network (optical fiber)
Roadside network (optical fiber)
BS (antenna)
BS (antenna)
Road-vehicle communication
Intervehicle communication
Car
Truck (for distribution)
Bus/taxi (for public transportation)
Walker

Figure 1.4 Image of the DSRC system. (BS: base station.)

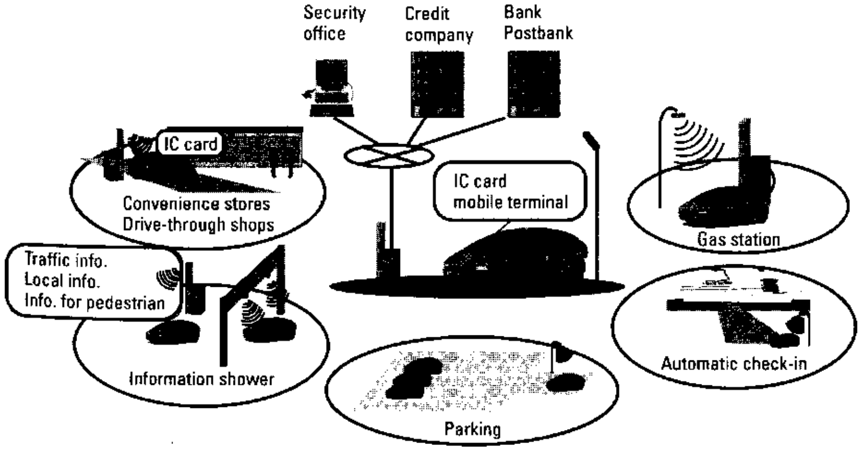


Figure 1.5 Applications of the DSRC system.

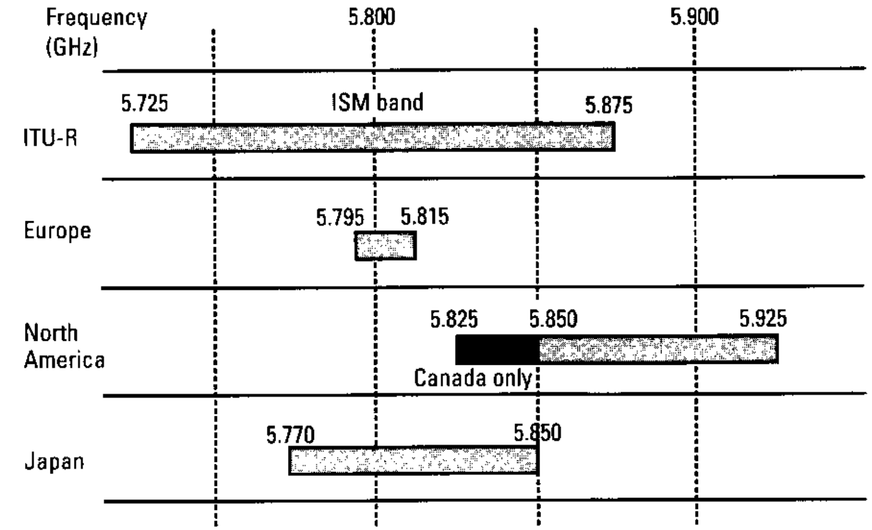


Figure 1.6 Spectrum allocations for the DSRC system.

Table 1.7
Standardized DSRC System

	Active	**Backscatter**
Organization	ARIB	CEN
RF carrier spacing	10 MHz	1.5, 12 MHz (medium data rate) 10.7 MHz (high data rate)
Allowable occupied bandwidth	Less than 8 MHz	5 MHz (medium data rate) 10 MHz (high data rate)
Modulation method	ASK (UL/DL)	ASK (DL)/PSK (UL)
Data coding	Manchester code	FMO (DL)/NRZI (UL)

Notes: DL: downlink; UL: uplink.

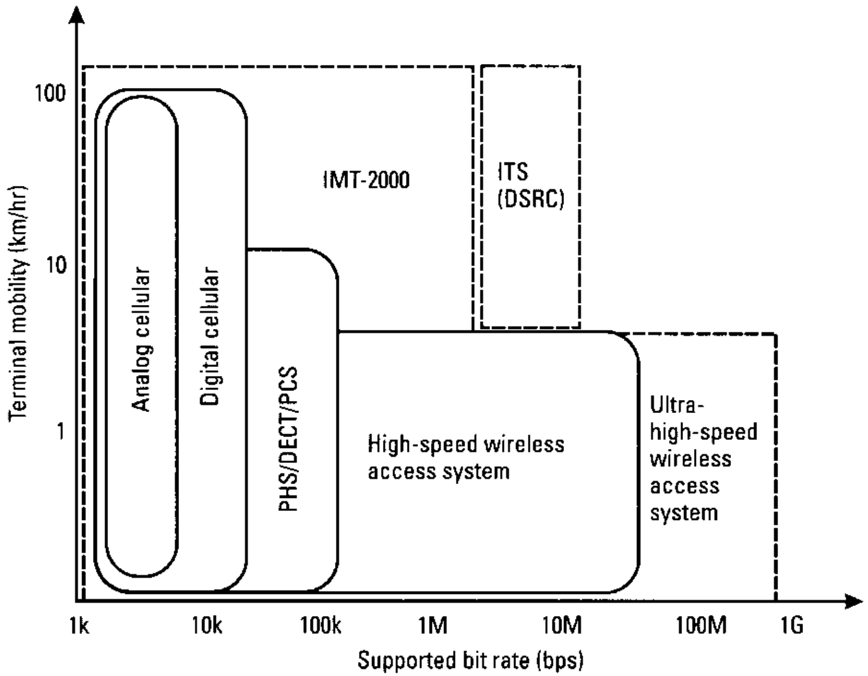


Figure 1.7 Classification of mobile communication systems.

1.2 Evaluation by Computer Simulation

The performances of several wireless communication systems can be evaluated by computer simulation without the need to develop prototypes and perform field experiments. This book focuses on the evaluation of digital wireless communication systems. The typical models of a digital wireless communication system are categorized into three types: (1) point-to-point communication, (2) point-to-multipoint communication, and (3) multipoint-to-multipoint communication. Sections 1.2.1–1.2.3 detail these three models.

1.2.1 Point-to-Point Communication

The concept of point-to-point communication is shown in Figure 1.8 [35]. In point-to-point communication, information data is first fed into a source encoder. In this encoder, the information data is digitized if it is analog data. If the volume of digitized data is large, the data is compressed by one of several encoding methods. *Motion Picture Experts Group* (MPEG) and *adaptive differential code modulation* (ADPCM) are examples of the encoding methods used for moving-image and voice-information data. Then, the source-encoded digital data is fed into a channel encoder to reduce the occurrence of bit errors under severe radio communication channels. For example, encoding techniques that use an *error-correction code* (ECC), such as a convolutional code, *Bose-Chaudhuri-Hocquenghem* (BCH) code, or Reed-Solomon code; a pilot data insertion technique; and a frame construction to estimate radio propagation characteristics are the representative examples.

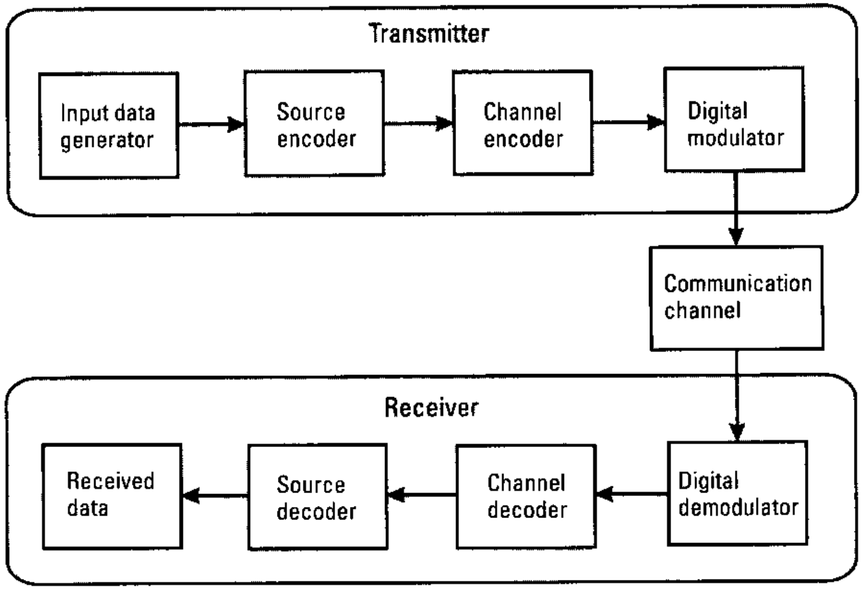


Figure 1.8 Point-to-point communication.

Next, the channel-encoded digital data is fed into a digital modulator and converted to a radio signal. The meaning of "modulation" is to vary the peculiar components that are included in carrier signal wave. The carrier signal is generally written as follows.

$$S(t) = A(t)\exp\left(2\pi f_c t + \theta(t)\right) \tag{1.1}$$

where $A(t)$, f_c and $\theta(t)$ are the time-variant amplitude, the radio carrier frequency, and the time-variant phase of the carrier wave signal, respectively. In (1.1), we have three peculiar components by which users can change the value. These are amplitude, frequency, and phase. If we change the amplitude of (1.1) in accordance with the digital information data, we call the modulation scheme *amplitude modulation* (AM). If we change the frequency of (1.1) with information data, then we call the modulation scheme *frequency modulation* (FM). Finally, if we change the phase of (1.1) in accordance with the digital information data, we call the scheme *phase modulation* (PM). The modulated signals are shown in Figure 1.9.

The process of modulation is performed in the lower frequency band as well as the carrier *radio frequency* (RF) band. In some cases, we use a quite low-frequency band in which digital signal processing can be performed. The

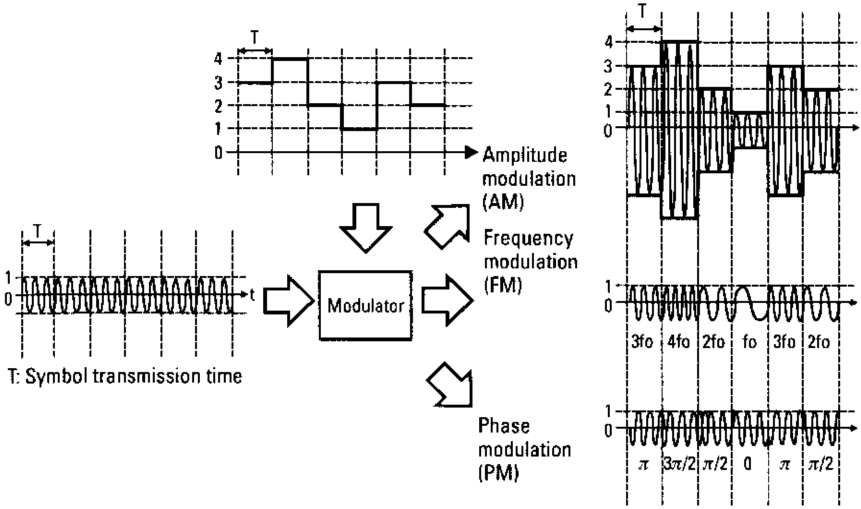

T
4
3
2
1
0
Amplitude modulation (AM)
Frequency modulation (FM)
Phase modulation (PM)
Modulator
t
T: Symbol transmission time
3fo
4fo
2fo
fo
3fo
2fo
π
3π/2
π/2
0
π
π/2

Figure 1.9 Examples of modulated signals.

frequency band is called the "baseband." If used in the baseband, we need an upconversion to the carrier RF band to make a modulated signal. If the direct conversion is difficult, we first upconvert the signal via an *intermediate frequency* (IF) band. Then, the digital modulated signal is transmitted to the receiver through a radio channel.

In the receiver, the received signal is fed into the digital demodulator and downconverted to baseband digital data. For conversion, the method where the received signal on the carrier frequency band is converted to the IF band, and then converted to the baseband is popular. Then, on the baseband, the level of amplitude, frequency, or phase is detected for the AM, FM, and PM schemes, respectively, and finally the transmitted digital data is recovered.

Next, the detected digital data is fed into the channel decoder, the various amplitudes and phases caused by the radio channel are compensated, and the ECC used in the transmitter is decoded. Finally, the channel-decoded data is fed into the source decoder, and the transmitted information data is recovered.

This book considers the model shown in Figure 1.8. Details of the shaded areas in Figure 1.8 are shown in Figure 1.10, in which the following elements are evaluated:

1. *Bit error rate* (BER);
2. *Frame-error rate* (FER)—if frame construction is used;
3. *Packet-error rate* (PER)—if packet construction is used.

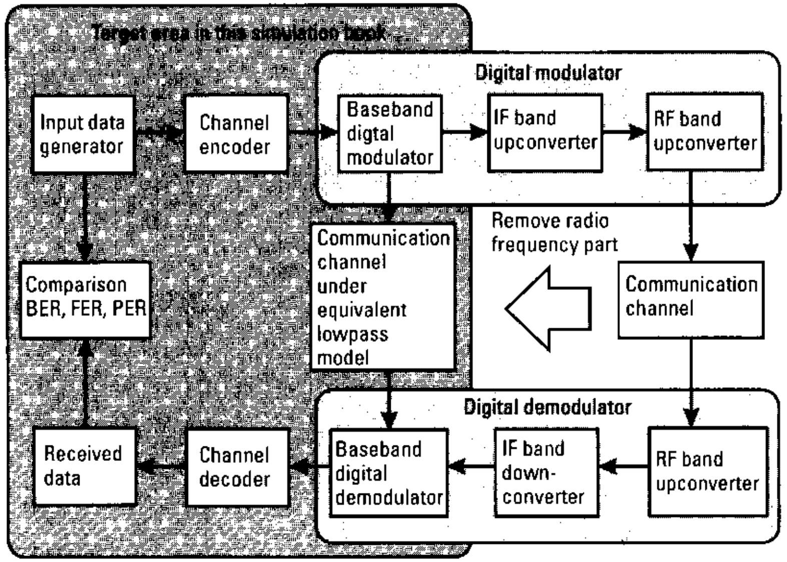


Figure 1.10 Target area of this book.

These values are evaluated by changing the following parameters:

1. Receiver noise level;
2. Level of received signal;
3. Fading environment (e.g., the level of the Doppler frequency);
4. Level of interference signals.

With the simulated digital wireless communication system shown in Figure 1.10, it is difficult to simulate a modulation and demodulation scheme in the RF band in some cases. This is because a lot of sampling data is needed to express a modulated signal in the RF band. The modulated signal is separated between the baseband frequency part and the RF band part as follows:

$$S(t) = A(t)\exp(\theta(t)) \times \exp(2\pi f_c t) \tag{1.2}$$

We can remove the RF band part from all blocks and describe all the component blocks in Figure 1.10 by a notation based on the baseband and perform an evaluation by computer simulation. This type of simulation is called a simulation under an equivalent lowpass system. The relationship between the RF band system and the equivalent lowpass system is also shown in Figure 1.10. By using a computer simulation under an equivalent lowpass system, we easily evaluate several data transmission performances under point-to-point communication.

Finally, a representative flowchart for obtaining several evaluated topics, such as BER, FER, and PER, is shown in Figure 1.10. First of all, random digital information data is generated at the input data generator. Then, these data are formatted as a packet or frame if needed at the channel encoder. Subsequently, the formatted data is modulated at the baseband digital modulator. Then, the baseband modulated signal is fed into a radio-channel block described by the equivalent lowpass model. In the receiver block, the transmitted signal expressed by the equivalent lowpass system is demodulated at the baseband digital demodulator and the recovered transmitted digital data, frame, or packet at the channel decoder block. Finally, the demodulated data, frame, or packet is compared with the transmitted ones and the BER, FER, or PER is obtained.

1.2.2 Point-to-Multipoint Communication

The image of point-to-multipoint communication is shown in Figure 1.11. In point-to-multipoint communication, an *access point* (AP) communicates with

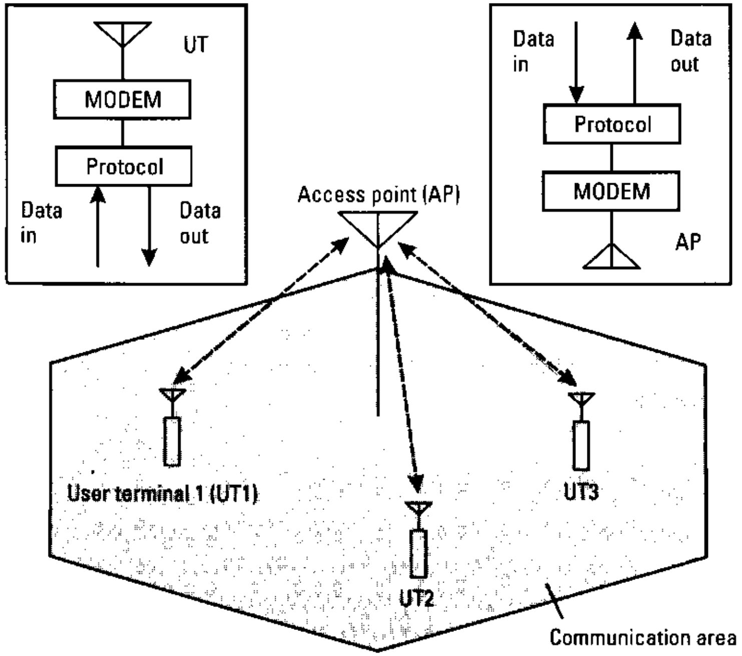


Figure 1.11 Point-to-multipoint communication.

several *user terminals* (UTs). First, users send their information data by using some protocols. The protocols are defined between an AP and several UTs. In some cases, collisions between information data from several UTs occur. By detecting the collisions and counting the number of collisions, we simulate the throughput of transmission data. Moreover, the average delay time to transmit information data from a UT is also simulated. However, even if collisions occur, the differences between levels of received signals are large. In this case, the data that has the largest signal level is received at the AP even if collisions occur. This effect is called a capture effect [1]. If we prepare some evaluation data between levels of interference signal and BER, FER, or PER in the simulation of point-to-point communication, we can simulate more precise throughput and average delay time under a capture-effect environment.

As shown above, the following items are mainly evaluated in point-to-multipoint communication in this book:

- Throughput;
- Average delay time.

These values are evaluated by changing the following parameters:

- Number of UTs;
- Volume of offered traffic;
- Level of capture effect.

A representative flowchart is shown in Figure 1.12. First of all, the traffic model, which is a data generation model in each terminal, the position of the AP, the position of UTs, and the number of UTs are determined. Then, under a programmed access protocol, we evaluate the throughput and average transmission delay by deciding whether a transmission packet was successfully transmitted at the AP.

1.2.3 Multipoint-to-Multipoint Communication

The schematic of multipoint-to-multipoint communication is shown in Figure 1.13. In multipoint-to-multipoint communication, several APs and UTs are installed, and call-blocking probability is evaluated as a new evaluation topic.

Call blocking occurs when the cellular system shown in Figure 1.13 is considered. In this cellular system, frequencies used in a cellular zone are reused at another cellular zone. Transmission signals generated by UTs located in one cellular zone sometimes interfere with APs located in other cellular zones using the same frequency band. The interference level that caused call blocking is large. By making a simulation model of multipoint-to-multipoint communication, we can evaluate the probability of call blocking.

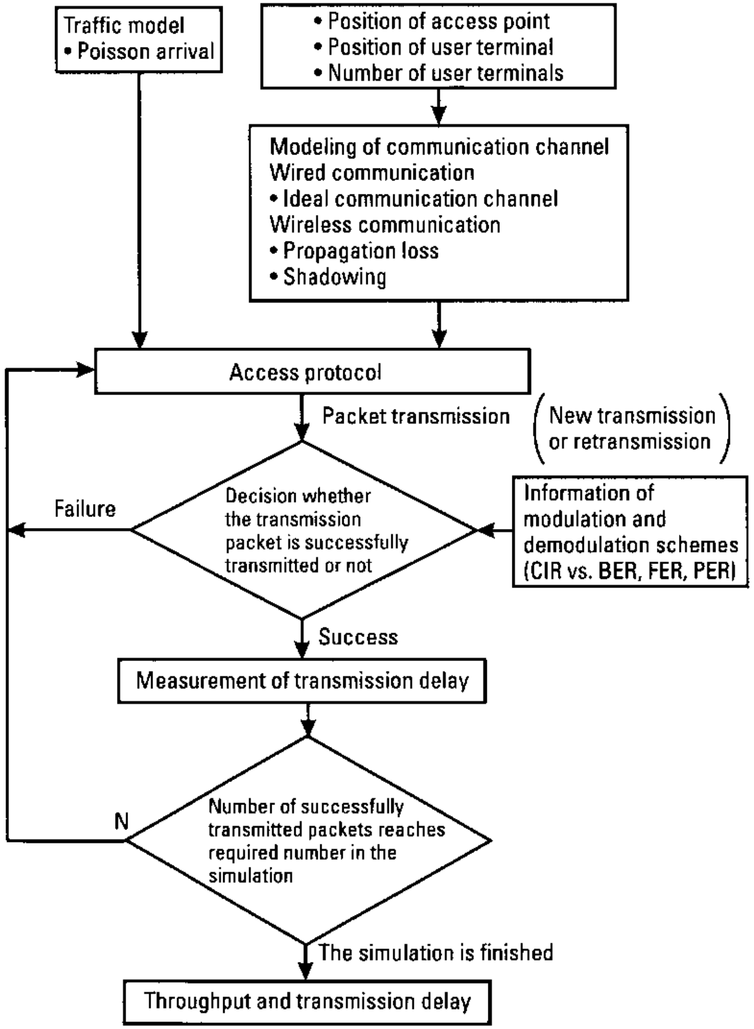

Traffic model
• Poisson arrival
• Position of access point
• Position of user terminal
• Number of user terminals
Modeling of communication channel
Wired communication
• Ideal communication channel
Wireless communication
• Propagation loss
• Shadowing
Access protocol
Packet transmission
New transmission or retransmission
Failure
Decision whether the transmission packet is successfully transmitted or not
Information of modulation and demodulation schemes (CIR vs. BER, FER, PER)
Success
Measurement of transmission delay
N
Number of successfully transmitted packets reaches required number in the simulation
The simulation is finished
Throughput and transmission delay

Figure 1.12 Flowchart to obtain transmission performance in point-to-multipoint communication.

A representative flowchart to evaluate the probability is shown in Figure 1.14. First of all, the position of UTs and APs, the number of UTs, and the size

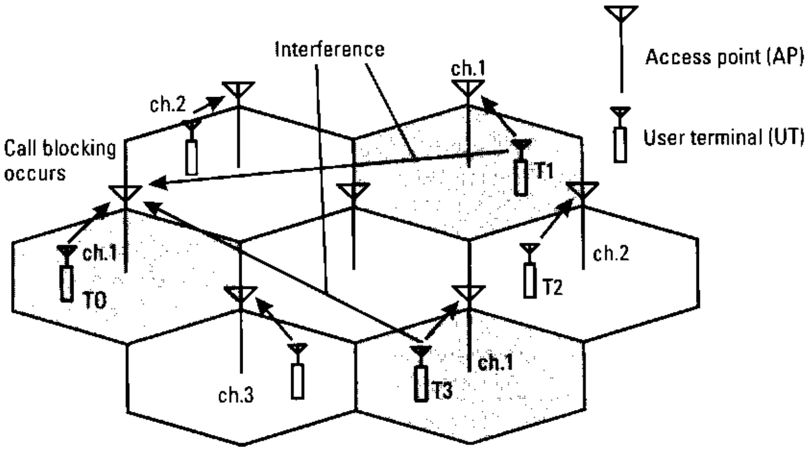


Figure 1.13 Multipoint-to-multipoint communication.

of the cellular zone are determined. At the same time, the traffic model, a data generation model in each terminal, is defined.

Then, under a programmed resource management technique, such as frequency-allocation methods, a cellular zone is configured, and the ratios between the desired signal level and undesired signal levels are measured at all APs. The ratio is calculated by considering such factors as the position of APs and UTs, the transmitted signal power, the antenna gain and pattern, and the height of antenna. By comparing the ratio and the defined threshold level, we can understand whether call blocking occurs. Finally, we obtain the call-blocking probability.

As shown earlier in this section, this book will allow readers to develop an understanding of evaluating transmission performance in point-to-point, point-to-multipoint, and multipoint-to-multipoint communication by using computer simulation.

1.3 Preview of This Book

This book is comprised of eight chapters. It covers all the necessary elements to evaluate the transmission performance of digital mobile communication systems by computer simulation. The simulation programs are presented in the appendix of each chapter and in the CD-ROM that accompanies the book.

Chapter 2 describes several key parameters to perform computer simulations smoothly. MATLAB, a good simulation software tool, is the primary tool used in this book. Therefore, the use of MATLAB language is first described.

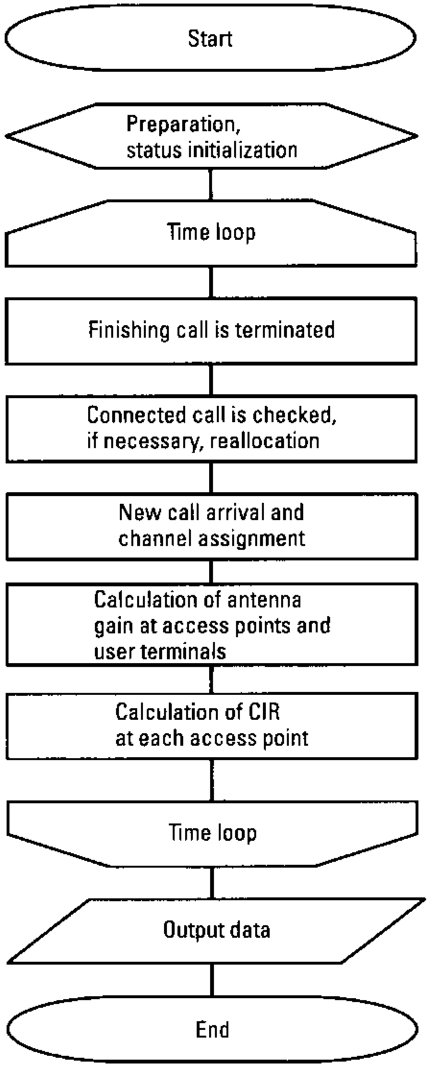

Start
Preparation, status initialization
Time loop
Finishing call is terminated
Connected call is checked, if necessary, reallocation
New call arrival and channel assignment
Calculation of antenna gain at access points and user terminals
Calculation of CIR at each access point
Time loop
Output data
End

Figure 1.14 Flowchart to obtain transmission performance in multipoint-to-multipoint communication.

Subsequently, Section 2.1 summarizes some frequently used commands and functions and the method for making a hierarchical program. In addition, Chapter 2 explains methods for programming the function blocks that are commonly used to evaluate all communication systems, such as the data-generation block, the Rayleigh-fading channel block, and the *additive white Gaussian noise* (AWGN) channel block.

Chapter 3 explains the basic configurations of the *phase shift keying* (PSK)–based digital radio transmission scheme and describes the method used to evaluate transmission performance under AWGN and a multipath Rayleigh-fading channel by computer simulation. Subsequently, the PSK-based digital radio transmission schemes—*binary PSK* (BPSK), *quadrature PSK* (QPSK), *offset QPSK* (OQPSK), *minimum shift keying* (MSK), *Gaussian-filtered MSK* (GMSK), and *quadrature AM* (QAM)—are introduced, and their performances are evaluated with computer simulation programs.

Chapter 4 presents the configuration of the OFDM transmission scheme, which can reduce the influence of multipath fading and realize broadband communication while maintaining high-frequency utilization efficiency. In addition, Chapter 4 describes the method used to simulate transmission performance under an AWGN and multipath Rayleigh-fading channel. These performances are evaluated by using programmed computer simulations.

Chapter 5 presents the configuration of CDMA. CDMA, which can maintain its strength in the face of multipath fading, is used in third-generation mobile communication systems. Chapter 5 also describes the method used to simulate transmission performance under AWGN and multipath Rayleigh fading channels. Again, these performances are evaluated by using programmed computer simulations.

Chapter 6 evaluates the transmission performance of a point-to-multipoint communication system with multiple-access protocols by computer simulation. Pure ALOHA, slotted ALOHA, nonpersistent *carrier sense multiple access* (CSMA), and slotted nonpersistent *inhibit sense multiple access* (ISMA) are explained as examples of multiple access protocols. Chapter 6 also describes the methods to simulate throughput and average delay time. Performances are evaluated by using programmed computer simulations.

Chapter 7 describes the basic simulation method for the multipoint-to-multipoint communication system based on cellular telecommunications systems. In addition, Chapter 7 introduces the *dynamic channel assignment* (DCA) algorithm, the *fixed-channel assignment* (FCA) algorithm, and the *adaptive cellular zone configuration* (AZC) algorithm using an adaptive antenna. Subsequently, Chapter 7 discusses the method used to simulate call-blocking probability. Performances are evaluated by using programmed computer simulations.

Chapter 8 describes a software radio communication system representing a future application of the software programming method studied in this book. This is because computer simulation languages have a good relationship with the software languages that configure *digital signal processing hardware* (DSPH) such as a *digital signal processor* (DSP), a *field-programmable gate array* (FPGA), or an *application-specific integrated circuit* (ASIC). The concept of software radio is one of the bridges between the software programming studied in this book and real hardware implementation.

Chapter 8 defines the software radio communication system and summarizes the advantages of software radio. In addition, Chapter 8 discusses the problems that must be overcome to realize the software radio communication system and explains the remarkable technologies developed to realize a software radio communication system. Moreover, Chapter 8 outlines the latest reported software-defined radio projects published in papers up to 2000 and explains several future applications of software radio–based communication systems.

References

[1] Prasad, R., *Universal Wireless Personal Communications,* Norwood, MA: Artech House, 1996.

[2] Dunlop, U. E., *Marconi—The Man and His Wireless,* New York: The Macmillan Company, 1937.

[3] Jolly, W. P., *Marconi,* London: Custable, 1972.

[4] Jakes, W. C., *Microwave Mobile Communications,* New York: IEEE Press, 1994.

[5] Sampei, S., *Applications of Digital Wireless Technologies to Global Wireless Communications,* Upper Saddle River, NJ: Prentice Hall, 1997.

[6] Calhoun, G., *Digital Cellular Radio,* Norwood, MA: Artech House, 1988.

[7] Cox, D. C., "Wireless Network Access for Personal Communication," *IEEE Comm. Mag.*, Vol. 30, December 1992, pp. 96–115.

[8] Padgett, J. E., C. G. Gunther, and T. Hattori, "Overview of Wireless Personal Communications," *IEEE Comm. Mag.*, Vol. 33, January 1995, pp. 28–41.

[9] Kinoshita, K., M. Kuramoto, and N. Nakajima, "Development of a TDMA Digital Cellular System Based on Japanese Standards," *Proc. IEEE VTC'91,* 1991, pp. 642–645.

[10] Tuttlebee, W. H. W. (ed.), *Cordless Telecommunications in Europe,* Berlin: Springer Verlag, 1990.

[11] Tuttlebee, W. H. W., "Cordless Personal Communications," *IEEE Comm. Mag.*, Vol. 30, December 1992, pp. 42–62.

[12] Prasad, R., *CDMA for Wireless Personal Communications,* Norwood, MA: Artech House, 1996.

[13] Ojanpera, T., and R. Prasad, *Wideband CDMA for Third-Generation Mobile Communications,* Norwood, MA: Artech House, 1998.

[14] van Nee, R., and R. Prasad, *OFDM for Wireless Multimedia Communications,* Norwood, MA: Artech House, 2000.

[15] Prasad, R., "Wireless Broadband Communication Systems," *IEEE Comm. Mag.*, Vol. 35, January 1997, p. 18.

[16] Honcharenko, W., et al., "Broadband Wireless Access," *IEEE Comm. Mag.*, Vol. 35, January 1997, pp. 20–26.

[17] Correia, L. M., and R. Prasad, "An Overview of Wireless Broadband Communications," *IEEE Comm. Mag.*, Vol. 35, January 1997, pp. 28–33.

[18] Morinaga, N., M. Nakagawa, and R. Kohno, "New Concepts and Technologies for Achieving Highly Reliable and High-Capacity Multimedia Wireless Communications Systems," *IEEE Comm. Mag.*, Vol. 35, January 1997, pp. 34–40.

[19] da Silva, J. S., et al., "Mobile and Personal Communications: ACTS and Beyond," *Proc. IEEE PIMRC '97,* September 1997.

[20] Prisoli, F. D., and R. Velt, "Design of Medium Access Control and Logical Link Control Functions for ATM Support in the MEDIAN System," *Proc. ACTS Mobile Comm. Summit '97,* October 1997, pp. 734–744.

[21] Rheinschmitt, R., A. de Haz, and M. Umehinc, "AWACS MAC and LLC Functionality," *Proc. ACTS Mobile Comm. Summit '97,* October 1997, pp. 745–750.

[22] Aldid, J., et al., "Magic into Reality, Building the WAND Modem," *Proc. ACTS Mobile Comm. Summit '97,* October 1997, pp. 734–744.

[23] Mikkonen, J., et al., "Emerging Wireless Broadband Networks," *IEEE Comm. Mag.*, February 1988, pp. 112–117.

[24] Hase, Y., et al., "R&D Project on Broadband Mobile Communications Using Microwave Band," *Proc. MDMC '96,* July 1996, pp. 158–162.

[25] Harada, H., and M. Fujise, "Experimental Performance Analysis of OCDM Radio Transmission System Based on Cyclic Modified M-Sequences for Future Intelligent Transport Systems," *Proc. IEEE VTC'99 Fall,* September 1999, pp. 764–767.

[26] Harada, H., and M. Fujise, "Field Experiments of a High Mobility and Broadband Mobile Communication System Based on a New Multicode Transmission Scheme for Future Intelligent Transport Systems," *Proc. IEEE VTC'2000 Spring,* May 1999.

[27] Wu, G., et al., "A Wireless ATM–Oriented MAC Protocol for High-Speed Wireless LAN," *Proc. IEEE PIMRC '97,* September 1997, pp. 198–203.

[28] Wu, G., Y. Hase, and M. Inoue, "An ATM-Based Indoor Millimeter-Wave Wireless LAN for Multimedia Transmissions" *IEICE Trans. Commun.*, Vol. E83-B, No. 8, August 2000, pp. 1740–1752.

[29] Wu, G., K. Mukumoto, and A. Fukuda, "Analysis of an Integrated Voice and Data Transmission System Using Packet Reservation Multiple Access," *IEEE Trans. Veh. Technol.,* Vol. 43, No. 2, May 1994, pp. 289–297.

[30] Najarian, P. B., "Status Update on ITS Activities in the U.S. and ITS America," *Proc. ITST2000,* October 2000, pp. 7–11.

[31] Ohyama, S., K. Tachikawa, and M. Sato, "DSRC Standards and ETC Systems Development in Japan," *Proc. 7th World Congress on Intelligent Transport Systems,* Torino, Italy, November 2000.

[32] Kim, J. M., "ITS Research and Development Activities at ETRI, Korea," *Proc. ITST2000,* October 2000, pp. 13–18.

[33] Armstrong, L., "New DSRC Standards Under Development for North America," *Proc. 7th World Congress on Intelligent Transport Systems,* Torino, Italy, November 2000.

[34] Rokitansky, C., C. Becker, and A. Feld, "DSRC Standardization and Conformance Testing of DSRC/EFC Equipment," *Proc. 7th World Congress on Intelligent Transport Systems,* Torino, Italy, November 2000.

[35] Proakis, J. G., *Digital Communications,* 3rd ed., New York: McGraw-Hill, 1995.

2

General Definition of Simulation Tools

2.1 Introduction

This chapter discusses several key topics to evaluate the transmission performance of wireless communication systems by computer simulation smoothly. This book primarily uses the excellent simulation software MATLAB. Therefore, first of all, Section 2.2 introduces a basic tutorial to use MATLAB by explaining selected commands that are frequently used in this book. Moreover, Section 2.2 defines several indispensable functions used frequently and commonly in the R&D of telecommunications by using MATLAB simulation programs. Section 2.3 explains "data generation" and BER by using the MATLAB program. Section 2.4 explains typical communication channels that are used to evaluate transmission performance and describes the method for programming the communication channels. In addition, Section 2.4 explains the AWGN channel and the Rayleigh fading channel. Finally, Section 2.5 concludes this chapter with a brief summary. Appendix 2A presents all the programs related to the chapter as does the CD-ROM that accompanies the book.

2.2 Basic Tutorial

This book uses the MATLAB computer-simulation software, which is produced by MathWorks, Inc. MATLAB, a sophisticated language for matrix calculations [1], stands for *matrix laboratory*. This book is not intended to serve as a tutorial of MATLAB. Readers interested in a tutorial should consult the MATLAB tutorial book [1].

2.2.1 Generation of Vectors and Matrices

First, we will provide you with a few examples demonstrating how easy it is to use the MATLAB language. When you start to use MATLAB, you will find a workspace. The workspace shows a command line.

You can directly input commands on the command line. Let's generate several vectors and matrices in the MATLAB workspace.

1. To make three vectors $u=\begin{pmatrix}2\\1\\4\end{pmatrix}$, $v = (1\ -1\ \ 1)$, and $s = 7$, type on the command line, and press enter,

   ```
   ⟩⟩  u=[ 2; 1; 4 ]
   ```

   you will see:

   ```
   u =
      2
      1
      4
   ```

   similarly type

   ```
   ⟩⟩  v=[1  -1  1 ]
   ```

   and you set

   ```
   v =
      1    -1     1
   ```

   and

   ```
   ⟩⟩ s=7
       s =
       7
   ```

2. To generate a 3-by-3 matrix

$$A = \begin{pmatrix}1 & 2 & 1\\2 & 1 & 2\\1 & 1 & 2\end{pmatrix}$$

```
⟩⟩  A=[1 2 1 ; 2 1 2; 1 1 2]
A =
   1  2  1
   2  1  2
   1  1  2
```

3. To transpose the vector and matrix, which operation-interchanges between a_{ij} and a_{ji}, you only need to put “′” after the vector and matrices. To transpose the v, type

   ```
   ⟩⟩  v′
   ```

   MATLAB will output the result of the operations as

   ```
   ans =
      1
      -1
      1
   ```

   To transpose A, type

   ```
   ⟩⟩  A′
   ```

   and set

   ```
   ans =
      1  2  1
      2  1  1
      1  2  2
   ```

2.2.2 Indexing and Subscripting

We now describe how to obtain a subpart of a matrix or vector by using the example matrix. First let’s generate the matrix.

$$B = \begin{pmatrix} 2 & 4 & 8 \\ 2 & 3 & 4 \\ 1 & 1 & 4 \end{pmatrix}$$

```
⟩⟩  B=[2 4 8; 2 3 4; 1 1 4]
B =
    2   4   8
    2   3   4
    1   1   4
```

The element in row *i* and column *j* of *B* is denoted by $B(i, j)$ in MATLAB. For example, if you need the element $B(2, 3)$, just type

```
⟩⟩  B(2,3)
ans =
     4
```

If you need any part of a matrix or vector, including elements from row *m* to row *n* and from column *p* to column *q*, you type `B(m:n,p:q)`. For example, if you need the elements from row 1 to row 2 and from column 2 to column 3, you type as follows

```
⟩⟩  B(1:2,2:3)
ans =
   4   8
   3   4
```

To access subparts of a matrix or vector in which the elements from row *m* to row *n* with increment *a* and from column *p* to column *q* with increment *b*, you execute `B(m:a:n,p:b:q)`. For example, if you need the elements from row 1 to row 3 with increment 2 and from column 1 to column 3 with increment 2, you type as follows

```
⟩⟩  B(1:2:3,1:2:3)
ans =
    2   8
    1   4
```

If you need all elements in row *m* of *B* or if you need all elements in column *p* of *B*, you also use the command "`:`". By typing `B(m,:)` and `B(:,p)`, you can obtain all elements in row *m* of *B* or all elements in column *p* of *B*, respectively.

For example, if you need all elements in row 3 of matrix *B*, type

```
⟩⟩  B(3,:)
ans =
   1  1  4
```

If you need all elements in column 2 of matrix B, type

```
〉〉 B(:,2)
ans =
   4
   3
   1
```

2.2.3 Matrix and Vector Calculations

This section introduces the basic calculation methods for matrices and vectors by MATLAB.

2.2.3.1 Addition and Subtraction

Addition and subtraction of matrices are defined just as they are for arrays—element-by-element. Addition and subtraction of matrices *A* and B are done as follows.

```
X=A+B
X =
   3  6  9
   4  4  6
   2  2  6
〉〉 Y=X-A
Y =
   2  4  8
   2  3  4

   1  1  4
```

Addition and subtraction require both matrices to have the same dimension. If the dimensions are incompatible, an error occurs.

```
〉〉 X=A+u
Error using = =>  +
```

Matrix dimensions must agree.

```
〉〉 w=v+s
w =
   8  6  8
```

2.2.3.2 Vector Products and Transpose

A row vector and a column vector of the same length can be multiplied in either order. The following are two examples.

```
x=v*u
x =
   5
⟩⟩  x=u*v
x =
   2  -2  2
   1  -1  1
   4  -4  4
```

2.2.3.3 Matrix Multiplication

The multiplication of matrices is defined in a way that reflects the composition of the underlying linear transformations and allows for the compact representation of systems of simultaneous linear equations. The matrix product $C = A * B$ is defined when the column dimension of A is equal to the row dimension of B, or when one of them is a scalar. If A is m-by-p and B is p-by-n, their product C is m-by-n. The next two examples illustrate the fact that matrix multiplication is not commutative; $A * B$ is usually not equal to $B * A$.

```
⟩⟩  X=A*B
X =
  7  11  20
  8  13  28
  6   9  20
⟩⟩  Y=B*A
Y =
   18  16  26
   12  11  16
    7   7  11
```

Moreover, the power of the matrix (e.g., $Z = A * A = A \,\hat{}\, 2$) is defined as

```
⟩⟩  Z=A^2
Z =
   6  5  7
   6  7  8
   5  5  7
```

A matrix can be multiplied on the right by a column vector and on the left by a row vector.

```
⟩⟩  x=A*u
x =
    8
   13
   11
⟩⟩  y=v*B
y =
    1  2  8
```

2.2.3.4 Element-by-Element Calculation

If you have two vectors $a = [a_1, a_2, a_3, \ldots a_N]$ and $b = [b_1, b_2, b_3, \ldots b_N]$, and you would like to obtain vector $c = [A!a_1 * b_1, a_2 * b_2, a_3 * b_3, \ldots b_N * b_N]$, you can use the “ .* ” command.

```
⟩⟩  c=a .* b
```

Moreover, if you would like to obtain vector $c = [a_1/b_1, a_2/b_2, a_3/b_3, \ldots a_N/b_N]$, you can use the “./ ” command.

```
⟩⟩  c=a ./ b
```

Moreover, if you would like to obtain vector $c = [a_1{}^\wedge b_1, a_2{}^\wedge b_2, a_3{}^\wedge b_3, \ldots a_N{}^\wedge b_N]$, you can use the “.^” command.

```
⟩⟩  c=a .^ b
```

The following two examples illustrate the above calculations by using $a = [4, 9, 8]$, $b = [4, 3, 4]$.

```
⟩⟩  a=[4,9,8]
a =
    4  9  8
⟩⟩  b=[4,3,4]
b =
    4  3  4
⟩⟩  c=a.*b
c =
    16  27  32
```

```
⟩⟩  c=a./b
c =
   1.0000  3.0000  2.0000
⟩⟩  c=a.^b
c =
   256  729  4096
```

These calculations are also valid for matrices, as shown by the following examples.

```
⟩⟩  C=A.*B
C =
  2  8  8
  4  3  8
  1  1  8
⟩⟩  C=B./A
C =
  2  2  8
  1  3  2
  1  1  2
⟩⟩  C=A.^B
C =
  1  16  1
  4  1  16
  1  1  16
```

Table 2.1 summarizes the MATLAB arithmetic operators.

2.2.4 Relational Operators

MATLAB provides the relational operators shown in Table 2.2.

These operators relate two vectors or two matrices; elements where the specified relation is true receive the value 1, whereas elements where the relation is false receive the value 0. The following illustrates the effect of several relational operators.

```
⟩⟩  A=[ 1 -1 -1; 1 1 1; -1 1 1]
A =
  1  -1  -1
   1  1   1
  -1  1   1
```

Table 2.1
Arithmetic Operators in MATLAB

+	Addition	.^	Power (element-by-element)
–	Subtraction	(vector)′	Transpose
.*	Multiplication (element-by-element)	*	Matrix multiplication
./	Right division (element-by-element)	/	Matrix, right division
+	Unary plus	^	Matrix power
–	Unary minus		

Table 2.2
Relational Operators in MATLAB

<	Less than	>=	Greater than or equal to
<=	Less than or equal to	==	Equal to
>	Greater than	~=	Not equal to

```
⟩⟩  B=[1 1 -1; -1 -1 -1; 1 -1 1]
B =
    1   1  -1
   -1  -1  -1
    1  -1   1
⟩⟩  A<B
ans =
   0  1  0
   0  0  0
   1  0  0
⟩⟩  A<=B
ans =
   1  1  1
   0  0  0
   1  0  1
⟩⟩  A>B
ans =
   0  0  0
   1  1  1
   0  1  0
⟩⟩  A>=B
ans =
  1  0  1
  1  1  1
  0  1  1
⟩⟩  A==B
ans =
  1  0  1
  0  0  0
  0  0  1
⟩⟩  A~=B
ans =
  0  1  0
  1  1  1
  1  1  0
```

2.2.5 Logical Operators

MATLAB provides the logical operators shown in Table 2.3.

An expression using the AND operator, &, is true if both operands are logically true. In numeric terms, the expression is true if both operands are nonzero. The following example shows its effect.

```
⟩⟩  u=[ 2  0  1  0  0  2 ]

u =
  2  0  1  0  0  2
⟩⟩  v=[ 2  1  3  0  1  0]
v =
  2  1  3  0  1  0
⟩⟩  u & v
ans =
  1  0  1  0  0  0
```

Elements having the value “1” indicate that the corresponding elements of *u* and *v* both are nonzero.

An expression using the OR operator, |, is true if one operand is logically true or if both operands are logically true. An OR expression is false only if both operands are false. In numeric terms, the expression is false only if both operands are zero. The following example shows its effect.

Table 2.3
Logical Operators in MATLAB

&	AND	|	OR	~	NOT

```
〉〉 u | v
ans =
  1  1  1  0  1  1
```

An expression using the NOT operator, ~, negates the operand. This produces a false result if the operand is true and a true result if it is false. In numeric terms, any nonzero operand becomes zero, and any zero operand becomes one. The following example shows its effect.

```
〉〉 ~u
ans =
  0  1  0  1  1  0
```

2.2.6 Flow Control

There are seven flow control statements in MATLAB, as shown in Table 2.4. We will use only five flow control statements: `if`, `switch`, `while`, `for`, and `break` in this book.

2.2.6.1 `if`, `else`, and `elseif`

The command, `if`, evaluates a logical expression and executes a group of statements based on the value of the expression. The syntax is:

```
if  logical_expression
  statements
elseif
  statements
end
```

Table 2.4
Flow Control Operators in MATLAB

`if`	Together with `else` and `elseif`, executes a group of statements based on some logical condition
`switch`	Together with case and otherwise, executes different groups of statements depending on the value of some logical condition
`while`	Executes a group of statements an indefinite number of times, based on some logical condition
`for`	Executes a group of statements a fixed number of times
`break`	Terminates execution of a for or while loop
`try...catch`	Changes flow control if an error is detected during execution
`return`	Causes execution to return to the invoking function

If the logical expression is true (1), MATLAB executes all the statements between the if and end lines. It resumes execution at the line following the end statement. If the condition is false (0), MATLAB skips all the statements between the if and end lines, and resumes execution at the line following the end statement. The following is an example of using the if command.

```
[Program]
  k=1
  if k==1
  fprintf( 'Ok \n');
  end

[Result]
  Ok
```

2.2.6.2 switch

The command switch executes certain statements based on the value of a variable or expression. Its basic form is:

```
switch expression ( scalar or string )
  case value1
     statements
  case value2
     statements
  .
  .
  .
  otherwise
     statements
end
```

The block consists of the following elements:

- The word switch followed by an expression to be evaluated.
- Any number of case groups. These groups consist of the word case followed by a possible value for the expression, all on a single line. Subsequent lines contain the statements to execute for the given value of the expression. These can be any valid MATLAB statement including another switch block. Execution of a case group ends when MATLAB encounters the next case statement or the otherwise statement. Only the first matching case is executed.
- An optional otherwise group. This consists of the word otherwise, followed by the statements to execute if the expression's value is not handled by any of the preceding case groups. Execution of the otherwise group ends at the end statement.

- An end statement.

The following is an example of using the switch command.

```
[Program]
  k=1;
  switch k
     case 1
        fprintf('it is 1 \n');
     otherwise
        fprintf('It is not 1 \n');
  end

[Result]
  It is 1
```

The example prints out "it is 1" on the screen of MATLAB if *k* is equal to 1; otherwise, it prints out "It is not 1".

2.2.6.3 while

The while loop executes a statement or group of statements repeatedly as long as the controlling expression is true (1). Its syntax is:

```
while expression
  statements
end
```

The following is an example of using the while command. In the example, we calculate a simple summation.

```
[Program]
  n=1;
  while n<100
     n=n+1;
  end
  fprintf('%d \n',n);

[Result]
  100
```

2.2.6.4 for

The for loop executes a statement or group of statements a predetermined number of times. Its syntax is

```
for index=start:increment:end
  statements
end
```

The default increment is 1. You can specify any increment, including a negative one. For positive indexes, execution terminates when the value of the index exceeds the end value; for negative increments, it terminates when the index is less than the end value. For example, we calculate the sum from 1 to 100.

```
[Program]
  total=0;
  for n=1:100
     total=total+n;
  end
  fprintf(‘%d \n’,total);

[Result]
  5050
```

2.2.6.5 break

The break statement terminates the execution of a for loop or while loop. When a break statement is encountered, execution continues with the next statement outside of the loop. In nested loops, break exits from the innermost loop only.

```
[Program]
  n=1;
  while n<100
     n=n+1;
     if n>=50, break, end
  end
  fprintf(‘%d \n’,n);

[Result]
  50
```

2.2.7 Custom-Made Functions

Except for the functions described in Section 2.2.6, we will make many built-in functions in MATLAB. If you are interested in these functions, please see the MATLAB textbooks [1]. This section summarizes several remarkable functions used throughout this book. These functions are detailed in Tables 2.5–2.8.

2.2.8 Hierarchical Programming

Until now, MATLAB has been presented as interpreter type software. However, MATLAB can be used as a programming language like C, C++, and FORTRAN. If you write program, the program consists of main-function and subfunction files and has a hierarchical structure. The relationship between main-function and subfunction files is shown in Figure 2.1. This section studies how to create hierarchically structured programs by means of an example.

The objectives in the example are listed as follows:

Table 2.5
Remarkable Functions Used in This Book (1)

Command	Synopsis	Description	Examples
abs	Y=abs(X)	abs(X) is the absolute value of the elements of X	abs(–5)=5 abs(3+4i)=5

acos(X)
asin(X)
atan(X)
	Y=acos(X)
	Y=asin(X)
	Y=atan(X)
		Inverse trigonometric functions	acos(cos(π))=3.1416
			asin(sin(π/2))=1.5708
			atan(tan(π/4))=0.7854

ceil	Y=ceil(X)	ceil(X) rounds the elements of X to the nearest integer which is greater than or equal to X	X=[–1.9 –0.2 3.4 5.6 7.0] ceil(X)=[–1 0 4 6 7]
conj	Y=conj(X)	conj(X) is the complex-conjugate of the elements of X.	X=[1+3j 2–5j] conj(X)=1.0000 – 3.0000i 2.0000 + 5.0000i

conv	c=conv(a,b)	c=conv(a,b) convolves vectors **a** and **b.**
		$c(k)=\sum_j a(j)b(k+1-j)$
			a=[1 2 3 4]
			b=[10 20 30]
			conv(a,b)=[10 40 100 160 170 120]

cos(X)
sin(X)
tan(X)
	Y=cos(X)
	Y=sin(X)
	Y=tan(X)
		Trigonometric functions	cos(π/3)=0.5000
			sin(π/6)=0.5000
			tan(π/4)=1.0000

erf	y=erf(x)	The error function erf(x) is the integral of the Gaussian distribution function from 0 to x
		$\mathrm{erf}(x)=\frac{2}{\sqrt{\pi}}\int_0^x e^{-t^2}$
			x=[1:30]
			y=erf(x)
			plot(x,y)

erfc	y=erfc(x)	The complementary error function erfc(x) is the value of the complementary erf(x)
		$\mathrm{erfc}(x)=\frac{2}{\sqrt{\pi}}\int_x^\infty e^{-t^2}=1-\mathrm{erf}(x)$
			x=[1:30]
			y=erfc(x)
			plot(x,y)

exp	Y=exp(X)	exp(X) returns e^X for each element of *x*	exp(1)=2.7183

fclose	fclose ('filename')	Fclose(fid) closes the specified file, if it is open and returns 0 if successful and –1 if unsuccessful	fid='parameter.txt'
			fopen(fid)
			fclose(fid)

Table 2.6
Remarkable Functions Used in This Book (2)

Command	Synopsis	Description	Examples

fft
ifft
	y=fft(x)
	y=ifft(x)
		ifft(x) is the fast Fourier transform (FFT) of vector **x**.
		fft(x,n) is the n-point inverse FFT
		ifft(x) is the inverse FFT of vector **x.**
		ifft(x,n) is the n-point inverse FFT
			X=[1 1 –1 –1 –1 1 1 –1]
			fft(ifft(X,8))=
			[1 1 –1 –1 –1 1 1 –1]

fix	Y=fix(X)	fix(X) rounds the elements of X to integers by eliminating the fractional part.	X=[–1.9 –0.2 3.4 5.6 7.0] fix(X)=[–1,0,3,5,7]
floor	Y=floor(X)	floor(X) rounds the elements of X to the nearest integers	X=[–1.9 –0.2 3.4 5.6 7.0] floor(X)=[–2 –1 3 5 7]

fopen	fopen ('filename')	fopen('filename','permission')
		fopen('filename','permission') opens the file name in the mode specified by permission
		'r': Open the file for reading
		'r+': Open the file for reading and writing
		'w': Delete the contents of an existing file or create a new file and open it for writing
		'w+': Delete the contents of an existing file or create a new file and open it for reading and writing
		'a': Create and open a new file or open an existing file for writing, appending to the end of the file
		'a+': Create and open a new file or open an existing file for reading and writing, appending to the end of the file
			fid='parameter.txt'
			fopen(fid,'r')
			fclose(fid)

1. Generation of Gaussian random data;
2. Calculation of mean and variance by using subfunctions named `value` and `disper`.

First of all, we define two subfunctions. The first function is for the calculation of mean value. For a data set indata = $[a_1, a_2, a_3, \ldots \ldots a_N]$ which consists of N symbol data, the mean E is given as:

Table 2.7
Remarkable Functions Used in This Book (3)

Command	Synopsis	Description	Examples

fprintf	fprintf(fid,’format’, A,…)
	fprintf(‘format’,A,…)
		fprintf(fid,’format’, A,…) formats the data in matrix A under control of the specified format string
		format is a string
		\n newline
		\t horizontal tab
		%d decimal numbers
		%e exponential notation
		%f fixed point notation
			a=3
			b=1.2
			c=0.0003
			fprintf(‘%d\t%f\t%e\t%e\n’, a,b,c)
			3 1.2000 3.0000e-004

fscanf	fscanf(fid,’format’, size)
	fscanf(fid,’format’)
		fscanf(fid, ’format’, size) reads data from the file specified by file identifier fid, converts it according to the specified format string, and returns it in matrix or vector
		The size is optional
		n: read n elements into a column vector
		inf read to the end of the file,
		[m,n] reads enough elements to fill an *m*-by-*n* matrix
		The format string can consist of
		%d decimal numbers
		%e,%f,%g floating point numbers
			fidd=fopen(‘test.txt’);
			fscanf(fidd,’%g’,[1,1]);

log	Y=log(X)	log(X) is the natural logarithm of the elements of X	log(exp(1))=1
log10	Y=log10(X)	log10(X) is base 10 logarithm of the elements of X	log10(1000)=3.0000
length	y=length(X)	length(X) calculates the length of vector **x**.	x=[1,2,3} length(X)=3
rand	Y=rand(m,n)	rand(m,n) is an *m*-by-*n* matrix with uniformly distributed random entries.	rand(2,3)=0.5828 0.3340 0.5798 0.4235 0.4329 0.7604

$$E(\text{indata}) = \frac{\sum_{k=1}^{N} a_k}{N} \tag{2.1}$$

Table 2.8
Remarkable Functions Used in This Book (4)

Command	Synopsis	Description	Examples
randn	Y=randn(m,n)	randn(m,n) is an *m*-by-*n* matrix with normally distributed random entries	randn(2,2)= –0.4326 0.1253 –1.1465 –1.6656 0.2877 1.1909
randperm	p=randperm(n)	p=randperm(n) is a random permutation of the integers 1:n	randperm(6) might be the vector [3 2 6 4 1 5] or it might be some other permutation of 1:6
real	X=real(Z)	real(Z) is the real part of the elements of Z	real(2+3.*i)=2
rem	rem(x,y)	rem(x,y) is x–n.*y where n=fix(x./y) is the integer part of the quotient, x./y.	rem(5,2)=1
round	Y=round(X)	round(X) rounds the elements of X to the nearest integers	X=[–1.9 –0.2 3.4 5.6 7.0] round(X)=[–2 0 3 6 7]
sqrt	Y=sqrt(X)	sqrt(X) is the square root of the elements of X	sqrt(2)=1.4142 sqrt(–2)=1.4142 i
sum	Y=sum(X)	sum(X) calculates the sum of vector **x**	X=[1,2,3] sum(X)=6

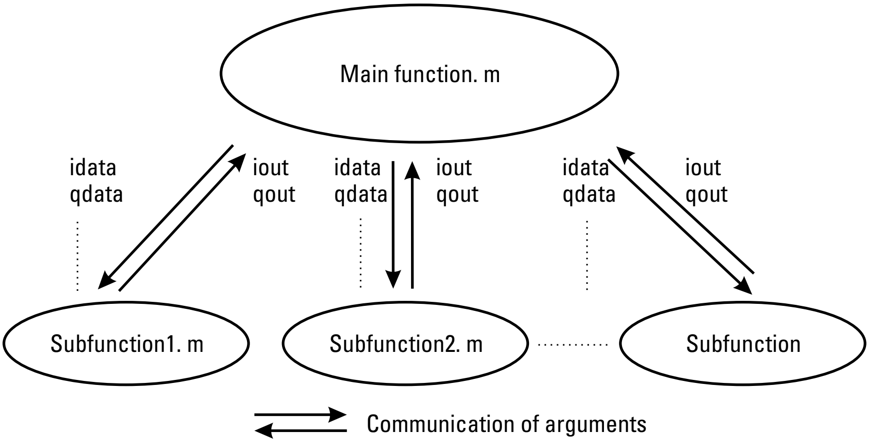


Figure 2.1 The relationship between main-function and subfunction files.

We will make a subfunction to express (2.1) in MATLAB. We use the following notation:

```
function  [output1, output2, … ] = function_name
  ( input argument 1 input argument2, … ….)
% start of files
output variable 1 = calculation equation by using
  input arguments
output variable 2 = calculation equation by using
  input arguments
.
.
% end of files
```

where “%” indicates a comment. You must save the function with a name like `function_name.m` in a directory of your computer where the extension “`.m`” is valid for MATLAB function. As for this function, it is valid if MATLAB’s current directory is the directory where the `function_name.m` is saved or if you set MATLAB’s path by using the setup menu, we will study the procedure to make subfunctions, by using an example in which we make a subfunction to calculate mean value.

First, open your favorite editor, like WordPad or mule (*emacs*) or the MATLAB editors, and create a new file. If you would like to use MATLAB editor, you just type

```
⟩⟩  edit
```

You must create the program file that has a name `mvalue.m` shown in Program 2.1. In this case, the number of input and output arguments both equal one. In the function, we add all the elements in a vector and divide the sum by the length of input vector. The use of commands `sum` and `length` is described in Section 2.2.7. Using them, we make subfunctions that can calculate dispersion and standard deviation.

The dispersion and the standard deviation for a vector `indata` = [a_1, a_2, a_3, …, a_N] is given as follows:

$$V(\text{indata}) = \frac{\sum_{k=1}^{N} a_k^2}{N} - \left(\frac{\sum_{k=1}^{N} a_k}{N} \right)^2 \tag{2.2}$$

$$\sigma(\text{indata}) = \sqrt{V(\text{indata})} \tag{2.3}$$

The next subfunction has two output arguments that are dispersion-value and standard deviation–value. We define the output dispersion and output standard deviation as sigma2 and sigma, respectively. In addition, we utilize a vector `indata` as the input argument. The input data is given in Program 2.2.

The program must be saved in the same directory as mvalue.m. Save it with the name disper.m.

So far, we have made two subfunctions. Next, we will try to make the main function. Please open a new file, and then make a program and save it with the program name main.m in the same directory as the subfunctions. The main.m is given Program 2.3 where random data is generated automatically, and the values of mean and dispersion are calculated. After making the program, enter the directory that has mvalue.m, disper.m, and main.m and execute main.m, then you should obtain the data below.

```
>> main
meanvalue = 0.522608
dispersion = 0.111834
standard deviation=0.334415
```

As a result, we can utilize mean, dispersion, and standard deviation values in main function and shrink the volume of main function. In this book, we will produce many subfunctions. By the end you should have a good database for the functions.

2.3 Data Generation and Bit Error Rate

We analyze the bit error rate to evaluate system performance. This section defines bit error probability and shows how to calculate the BER with computer software with a few examples. Using the above example, we generate data, consisting of a 1-by-10 vector and the element in the data consists of 0 or 1. The vector, which is named txdata, is given as follows:

```
>> txdata = rand(1,10) > 0.5
txdata =
  1  0  1  0  1  1  0  0  1  0
```

If an error occurs on the communication channel and txdata(1,7) changes from 0 to 1, the received data rxdata is given as follows

```
>> rxdata = txdata ;
>> rxdata(1,7)=1

>> rxdata(1,9)=0
rxdata =
  1  0  1  0  1  1  1  0  0  0,
```

where the underlined data represents errors. By using transmitted data, txdata, and received data, rxdata, we count the number of transmitted pieces of data and the number of errors.

To count the number of transmitted pieces of data, we need only the length of txdata. MATLAB has a good command for the calculation of vector size—that is, length. We define the number of transmitted data as nod:

```
>>  nod = length(txdata)
nod =
   10
```

To calculate the number of errors, we execute the following procedure. First, we subtract the transmitted data txdata from the received data rxdata. If no error exists, you obtain a zero vector with the length of nod. Otherwise, you obtain a nonzero vector in which "-1" or "1" data occurs at the error positions. The subtraction vector is defined as subdata, which is given as follows:

```
>>  subdata=rxdata-txdata
subdata =
   0   0   0   0   0   0   1   0   -1   0
```

As you can see in the vector subdata, if an element in txdata changes from 0 to 1 because of an error, the element of vector subdata at the error position becomes "1". On the other hand, if an element in txdata changes from 1 to 0 because of an error, the element of vector subdata at the error position becomes "-1". By taking the absolute value of the subdata elements, we can make vector that has "1" at each error element.

```
>>  abs(subdata)
ans =
   0   0   0   0   0   0   1   0   1   0
```

By adding all elements in the vector abs(subdata), we can calculate the number of errors. For the element summation, MATLAB also has a good command, "sum." If the number of errors is noe, we obtain, by utilizing the command "sum,"

```
>>  noe=sum(abs(subdata))
noe =
   2
```

Therefore, we can calculate the bit error probability by using noe and nod. We define the bit error rate as *ber*, which is given by dividing noe by nod as follows:

```
>>  ber=noe/nod
ber =
   0.2000
```

We use this procedure to obtain the BER throughout this book.

2.4 Definition of a Radio Communication Channel

This book analyzes system performance under AWGN and/or multipath fading environments. This section explains AWGN and multipath fading and shows how to simulate an AWGN and multipath fading environment by MATLAB.

2.4.1 AWGN Channel

If we construct a mathematical model for the signal at the input of the receiver, the channel is assumed to corrupt the signal by the addition of white Gaussian noise, as illustrated in Figure 2.2 [2]. When we define transmitted signal, white Gaussian noise, and received signal as $s(t)$, $n(t)$, and $r(t)$, the received signal is

$$r(t) = s(t) + n(t) \tag{2.4}$$

where $n(t)$ is a sample function of the AWGN process with probability density function (pdf) and power spectral density as follows:

$$\Phi_{nn}(f) = \frac{1}{2}N_0[\text{W/Hz}] \tag{2.5}$$

where N_0 is a constant and often called the noise power density. To simulate in MATLAB, we simply use the built-in function `randn`, which generates random numbers and matrices whose elements are normally distributed with mean 0 and variance 1. Therefore, if we add AWGN noise with power 1 to the digital

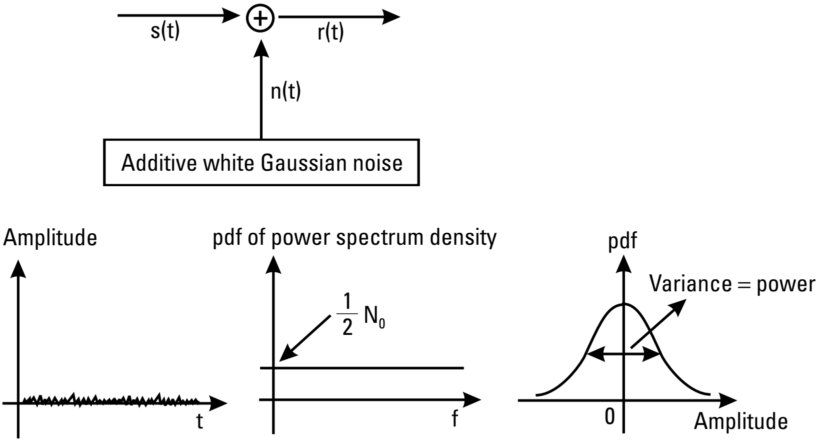


Figure 2.2 Example of an AWGN channel.

modulation signal with in-phase channel (I-channel) and quadrature-phase channel (Q-channel) data vectors, *idata* and *qdata*, respectively, the output data of I channel and Q channel, *iout* and *qout,* are given as follows:

$$\begin{aligned} iout(t) &= idata(t) + randn(t) \\ qout(t) &= qdata(t) + randn(t) \end{aligned} \tag{2.6}$$

However, in the simulation, we sometimes calculate the BER performance by varying the noise power, where we define the noise power as a variable, *npow*. *idata* and *qdata* are voltages, not powers. Therefore, we must change the notation of *npow* from power to voltage. We define a variable *attn* as the root of *npow* as

$$attn = \frac{1}{2}\sqrt{npow} \tag{2.7}$$

Therefore, the revised output data after contamination from noise with a power of *npow* becomes

$$\begin{aligned} iout(t) &= idata(t) + attn \times randn(t) \\ qout(t) &= qdata(t) + attn \times randn(t) \end{aligned} \tag{2.8}$$

Program 2.4 can add AWGN to the input data, which is expressed as digital quadrature phase modulation. For the input data, we use *idata* and *qdata*. The output data are *iout* and *qout*. We only input *attn*, *idata,* and *qdata*, to set the noise-contaminated signal.

2.4.2 Rayleigh Fading Channel

The path between the base station and mobile stations of terrestrial mobile communications is characterized by various obstacles and reflections. For example, an indoor environment has business machines and furniture, and buildings and trees constitute an outdoor environment. These have a large influence on the received signal, when the radio wave is propagated from the base station to the mobile station. The general characteristics of radio wave propagation in terrestrial mobile communications are shown in Figure 2.3. The radio wave transmitted from a base station radiates in all directions these radio waves, including reflected waves that are reflected off of various obstacles, diffracted waves, scattering waves, and the direct wave from the base station to the mobile station. In this case, since the path lengths of the direct, reflected,

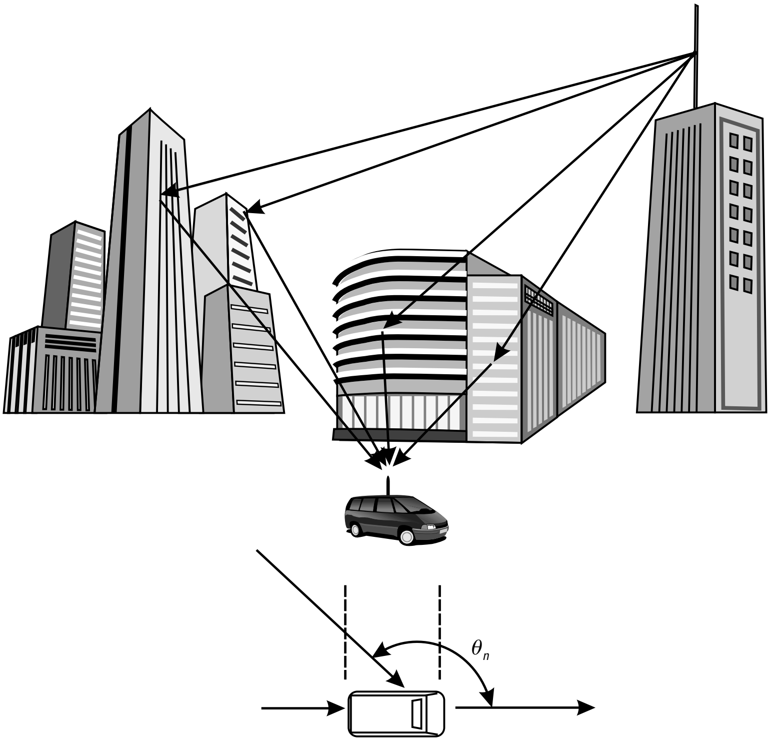


Figure 2.3 Principle of multipath channel.

diffracted, and scattering waves are different, the time each takes to reach the mobile station will be different. In addition the phase of the incoming wave varies because of reflections. As a result, the receiver receives a superposition consisting of several waves having different phase and times of arrival. The generic name of a radio wave in which the time of arrival is retarded in comparison with this direct wave is called a delayed wave. Then, the reception environment characterized by a superposition of delayed waves is called a multipath propagation environment. In a multipath propagation environment, the received signal is sometimes intensified or weakened. This phenomenon is called multipath fading, and the signal level of the received wave changes from moment to moment. Multipath fading raises the error rate of the received data, when a digital radio signal is transmitted in the mobile communication environment. A compensation method for this multipath fading must be used to ensure a high transmission performance. This section discusses the concept of multipath fading and explains a programming method for simulations of multipath fading.

Let us begin with the mechanism by which fading occurs [3, 4]. The delayed wave with incident angle θ_n is given by (2.9) corresponding to Figure 2.3, when a continuous wave of single frequency f_c (Hz) is transmitted from the base station.

$$r_n(t) = \mathrm{Re}\left[e_n(t) \exp j\left(2\pi f_c t\right)\right] \tag{2.9}$$

where Re[] indicates the real part of a complex number that gives the complex envelope of the incoming wave from the direction of the number *n*. Moreover, *j* is a complex number. $e_n(t)$ is given in (2.10) by using the propagation path length from the base station of the incoming waves: L_n (m), the speed of mobile station, v (m/s), and the wavelength, λ(m).

$$\begin{aligned} e_n(t) &= R_n(t) \exp j\left(-\frac{2\pi\left(L_n - vt\cos\theta_n\right)}{\lambda} + \phi_n\right) \\ &= x_n(t) + jy_n(t) \end{aligned} \tag{2.10}$$

where R_n and ϕ_n are the envelope and phase of the *n*th incoming wave. $x_n(t)$ and $y_n(t)$ are the in-phase and quadrature phase factors of $e_n(t)$, respectively. The incoming *n*th wave shifts the carrier frequency as v cos θ_n/λ (Hz) by the Doppler effect (Hz). This is called the Doppler shift in land mobile communication [3, 4]. This Doppler shift, which is described as f_d, has a maximum value of v/λ, when the incoming wave comes from the running direction of mobile station in cos $\theta_n = 1$. Then, this maximum is the largest Doppler shift. The delayed wave that comes from the rear of the mobile station also has a frequency shift of $-f_d$ (Hz).

It is shown by (2.11), since received wave $r(t)$ received in mobile station is the synthesis of the above-mentioned incoming waves, when the incoming wave number is made to be *N*.

$$\begin{aligned} r(t) &= \sum_{n=1}^{N} r_n(t) \\ &= \mathrm{Re}\left[\left(\sum_{n=1}^{N} e_n(t)\right) \exp j\left(2\pi f_c t\right)\right] \\ &= \mathrm{Re}\left[\left(x(t) + jy(t)\right)\left(\cos 2\pi f_c t + j\sin 2\pi f_c t\right)\right] \\ &= x(t)\cos 2\pi f_c t - y(t)\sin 2\pi f_c t \end{aligned} \tag{2.11}$$

where $x(t)$ and $y(t)$ are given by

$$\begin{aligned} x(t) &= \sum_{n=1}^{N} x_n(t) \\ y(t) &= \sum_{n=1}^{N} y_n(t) \end{aligned} \tag{2.12}$$

and $x(t)$ and $y(t)$ are normalized random processes, having an average value of 0 and dispersion of σ, when N is large enough. We have (2.13) for the combination probability density $p(x,y)$, where $x = x(t)$, $y = y(t)$

$$p(x,y) = \frac{1}{2\pi\sigma^2} \exp\left(\frac{x^2 + y^2}{2\sigma^2} \right) \tag{2.13}$$

In addition, it can be expressed as $r(t)$ using the amplitude and phase of the received wave.

$$r(t) = R(t)\cos(2\pi f_c t + \theta(t)) \tag{2.14}$$

$R(t)$ and $\theta(t)$ are given by

$$\begin{aligned} R(t) &= R = \sqrt{x^2 + y^2} \\ \theta(t) &= \theta = \tan^{-1}[y/x] \end{aligned} \tag{2.15}$$

By using a transformation of variables, $p(x, y)$ can be converted into $p(R, \theta)$

$$p(R,\theta) = \frac{R}{2\pi\sigma^2} \exp\left(-\frac{R^2}{2\sigma^2} \right) \tag{2.16}$$

By integrating $p(R, \theta)$ over θ from 0 to 2, we obtain the probability density function $p(R)$

$$p(R) = \frac{R}{\sigma^2} \exp\left(-\frac{R^2}{2\sigma^2} \right) \tag{2.17}$$

Moreover, we can obtain the probability density function $p(\theta)$ by integrating $p(R, \theta)$over R from 0 to ∞.

$$p(\theta) = \frac{1}{2\pi} \tag{2.18}$$

From these equations, the envelope fluctuation follows a Rayleigh distribution, and the phase fluctuation follows a uniform distribution on the fading in the propagation path.

Next, let us try to find an expression for simulations of this Rayleigh fading. Here, the mobile station receives the radio wave as shown in Figure 2.3, the arrival angle of the receiving incoming wave is uniformly distributed, and the wave number of the incoming waves is N. In this case, the complex fading fluctuation in an equivalent lowpass system is [3, 4]

$$\begin{aligned} r(t) &= x(t) + j \cdot y(t) \\ &= \left[\sqrt{\frac{2}{N_1 + 1}} \sum_{n=1}^{N_1} \sin\left(\frac{\pi n}{N_1}\right) \cos\left\{ 2\pi f_d \cos\left(\frac{2\pi n}{N_1}\right) t \right\} + \frac{1}{\sqrt{N_1 + 1}} \cos(2\pi f_d t) \right] \\ &\quad + j \sqrt{\frac{2}{N_1}} \sum_{n=1}^{N_1} \sin\left(\frac{\pi n}{N_1}\right) \cos\left\{ 2\pi f_d \cos\left(\frac{2\pi n}{N_1}\right) t \right\} \end{aligned} \tag{2.19}$$

where N_2 is an odd number and N_1 is given by

$$N_1 = \frac{1}{2}\left(\frac{N}{2} - 1\right) \tag{2.20}$$

In this case, the following relations are satisfied

$$\begin{aligned} E\left[x_I^2(t)\right] &= E\left[y_Q^2(t)\right] = \frac{1}{2} \\ E\left[x_I(t) y_Q(t)\right] &= 0 \end{aligned} \tag{2.21}$$

and it is shown that the result of (2.19) takes the form of Program 2.5. In Program 2.5, the output signal is obtained by inputting the complex modulating signal formed by the transmitter and expressing it in an equivalent lowpass system.

Next, let us discuss how a multipath propagation environment can be simulated. In the multipath propagation environment, the mobile station receives not only the direct wave but also delayed waves caused by reflection, diffraction, and scattering that reach the time later than the direct wave. The model of the relationship between this direct wave and the delayed wave is shown in Figure 2.4.

It is clear that the amplitude has a Rayleigh distribution and that the phase has a uniform distribution when we observe the received signal at the

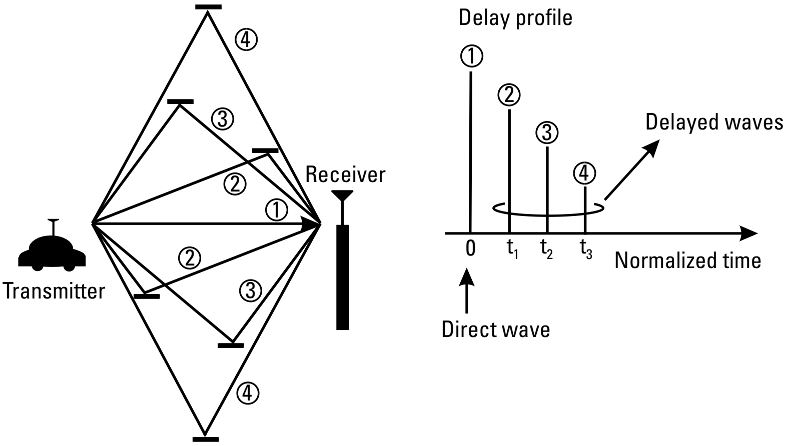


Figure 2.4 The configuration of multipath fading channel.

arrival time. It is also clear that there are fixed ratios of the average electric powers of the direct and delayed waves. That is to say, we have only to give the relative signal level and relative delay time of delayed waves in comparison with the direct wave, when this multipath propagation environment is simulated; its flowchart is shown in Figure 2.5. The program for simulations of multipath fading is given in Programs 2.6 and 2.7.

In Program 2.6, we input the time resolution, relative signal levels, and relative delay times of the direct and delayed waves, a complex modulating signal formed by the transmitter and expressed in an equivalent lowpass system, and the simulation time for one simulation loop.

Consider, as an example, the case of a simulation time at one simulation loop and a minimum time resolution of simulation use 50 μs and 0.5 μs, respectively. It is assumed that three delayed waves have mean powers of 10 dB, 20 dB, and 25 dB smaller than that of the direct wave, respectively, and that the relative arrival times were retarded with respect to the direct wave by 1, 1.5, and 2.0 μs, respectively. In this case, the input variable for the multipath fading simulator is

```
tstp = 0.5.*10.^(-6);
itau = [0 floor(1.*10.^(-6)/tstp) floor(1.5.*10.^(-6)/tstp)
  floor(2.0.*10.^(-6)/tstp)]= [0, 2, 3, 4];
dlvl = [0,10, 20,25];
nsamp = 50.*10.^(-6)/tstp = 100;
```

The parameter is set using such expressions for the simulator. Next, we describe the operation of the multipath fading simulator. To begin, the input signal is

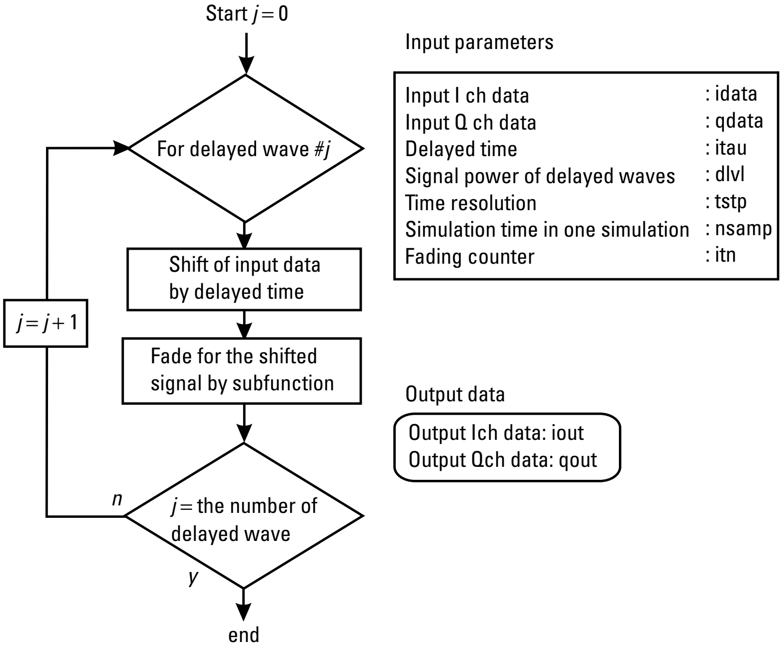


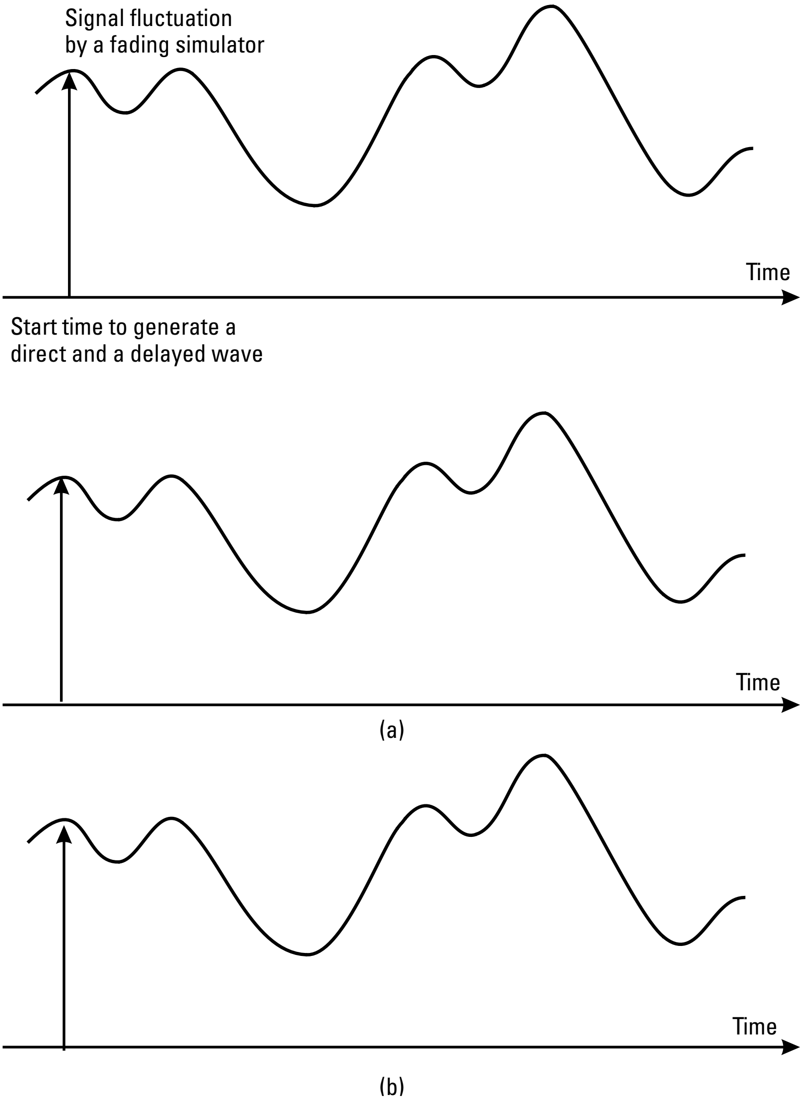
Figure 2.5 The flowchart to obtain the Rayleigh fading channel.

delayed by using the input parameters. Next, Rayleigh fading is added to the delayed signals. Only the number of delayed waves set in the parameter repeats this process. All are added afterwards. As a result, the output signal taken from the multipath Rayleigh fading is obtained.

One problem exists in this case: Generally speaking, the distribution of Rayleigh must be independent for each delay time. However, in this simulation, the distributions are the same, because fading waveforms of all transmission paths are generated by a function. This characteristic is shown in Figure 2.6. Therefore, the technique that let the fading in each delay time generate independently is needed.

There are various methods for generating an independently fading delay time, and here, the fading counter method is shown as an example. A fading counter gives the start time of fading generation to a fading generator such as `fade.m`.

Figure 2.6 shows a fluctuated signal generated by a fading generator. In the simulation, we use a fading generator; therefore, there is only one generated waveform of fading. If the fading generators for all direct and delayed waves are started simultaneously by setting fading counters for both waves the same, all fading waveforms are the same as in Figure 2.6. However, if we give different



(a)

(b)

Figure 2.6 Signal fluctuation by a fading simulator when the start time to generate a direct and a delayed wave is the same: (a) generated signal fluctuation for a direct wave and (b) generated signal fluctuation for a delayed wave.

start times of fading generation to all direct and delayed waves by setting the fading counters differently, as shown in Figure 2.7, the waveforms of the generated fadings in each propagation path will be different. Therefore, we can simulate an independently distributed Rayleigh fading environment.

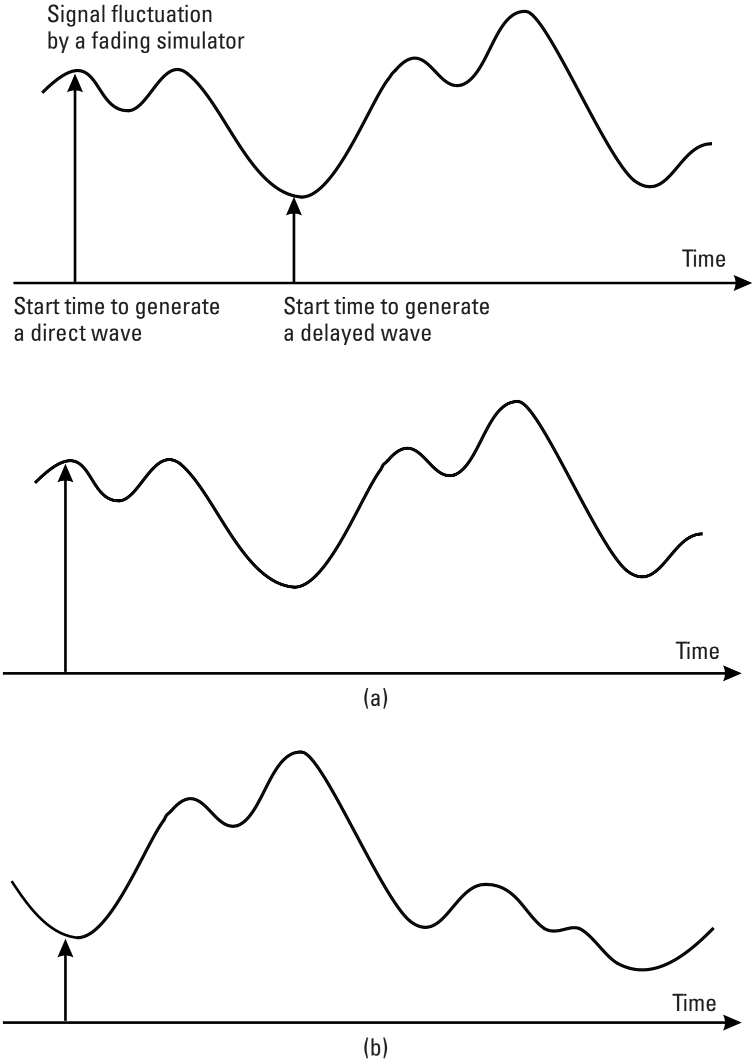


Figure 2.7 Signal fluctuation by a fading simulator when the start time to generate a direct and a delayed wave is different: (a) generated signal fluctuation for a direct wave and (b) generated signal fluctuation for a delayed wave.

For this `sefade.m`, shown in Program 2.6, we use the evaluation program shown in Program 2.8. The initial value of the fading counter is given in this evaluation program. The size of the vector of this fading counter is equal to the size of the vector that expresses the delay time of the delayed wave and the size of the vector that shows the relative power level of the delayed wave. Then, the

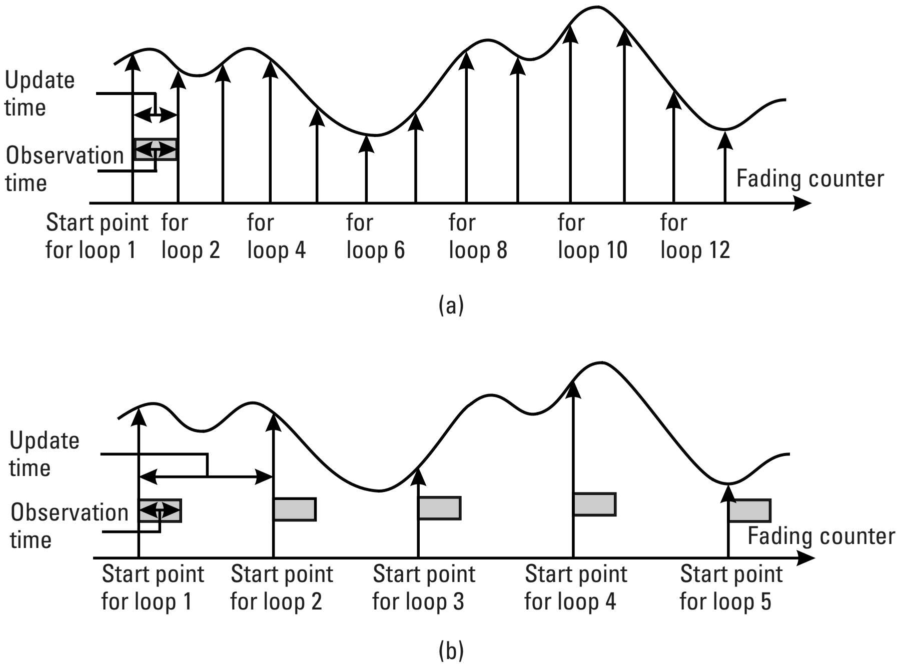


Figure 2.8 Relationship between the observation time and the update time: (a) observation time = update time and (b) observation time < update time.

Rayleigh fading is independently formed in each delay time, by setting the counter adaptively.

The fading counter is updated after each simulation loop by adding a value, `itnd0`, corresponding to the simulation time to it. One hundred points are added after each simulation loop in a case of a minimum time resolution of 0.5 μs and an observation time of 50 μs. The added value is called the update time. The update time can be adjusted to reduce the simulation time. Figure 2.8 shows the simulated signal levels caused by a fading when the update time is equal to the observation time [Figure 2.8(a)] and when the update time is larger than the observation time [Figure 2.8(b)]. When the update time is equal to the observation time, we can evaluate a transmission performance under a continuous-changed signal level. However, it takes a long time to stabilize the simulated performance, because the distribution of the signal fluctuation becomes Rayleigh by many simulation loops. On the other hand, when the update time is larger than the observation time, we can evaluate the transmission performance under Rayleigh fading with a small number of simulation loops as shown in Figure 2.8(b). However, the simulation result may not be precise. We therefore had better increase the update time when we check the transmission performance briefly. This multipath fading simulator will be used in the rest of the discussions in this book.

2.5 Conclusion

This chapter introduced the simulation language that will be used throughout this book. Using the simulation language, we created several general-purpose functions used frequently that will be employed in subsequent chapters. Chapter 3's method for simulating a mobile communication system is based on the discussions and tutorials of this chapter.

References

[1] MathWorks Inc., "MATLAB Using MATLAB."

[2] Proakis, J. G., *Digital Communications,* 3rd ed., New York: McGraw-Hill, 1995.

[3] Jakes, W. C., *Microwave Mobile Communications,* New York: IEEE Press, 1994.

[4] Sampei, S., *Applications of Digital Wireless Technologies to Global Wireless Communications,* Upper Saddle River, NJ: Prentice Hall, 1997.

Appendix 2A

Program 2.1

```
% Program 2-1
%
% calculate average
%
% Programmed by H. Harada
%

function outdata = mvalue(indata)

%*********************** variables **********************
%     indata : Input data
%********************************************************

outdata=sum(indata)/length(indata);

% ********************* end of file *********************
```

Program 2.2

```
% Program 2-2
% disper.m
%
% calculate dispersion and standard deviation
%
% Programmed by H. Harada
%

function  [sigma2, sigma]=disper(indata)

%*********************** variables **********************
%  indata  : Input data
%  sigma2  : dispersion
%  signa   : standard deviation
%********************************************************

% calculate average value
mvalue= sum(indata)/length(indata);

% calculate square average
sqmean= sum(indata.^2)/length(indata);
% calculate dispersion
sigma2=sqmean-mvalue^2;

% calculate standard deviation
sigma=sqrt(sigma2);

%******************** end of file *********************
```

Program 2.3

```
% Program 2-3
% main.m
%
% calculate mean, dispersion and standard deviation for
%  the vector data
%
% Programmed by H. Harada
%

data=rand(1,20);
mvalue2=mvalue(data);
[sigma2, sigma]= disper(data);
fprintf( ‘meanvalue = %f \n’,mvalue2);
fprintf( ‘dispersion = %f \n’, sigma2);
fprintf( ‘standard deviation=%f \n’,sigma);

%******************** end of file *********************
```

Program 2.4

```
% Program 2-4
% comb.m
%
% Generate additive white gaussian noise
%
% Programmed by H. Harada
%

function [iout,qout] = comb (idata,qdata,attn)

%*********************** variables ********************
% idata : input Ich data
% qdata : input Qch data
% iout   output Ich data
% qout   output Qch data
% attn : attenuation level caused by Eb/No or C/N
%******************************************************

iout = randn(1,length(idata)).*attn;
qout = randn(1,length(qdata)).*attn;

iout = iout+idata(1:length(idata));
qout = qout+qdata(1:length(qdata));

%******************** end of file *********************
```

Program 2.5

```
% Program 2-5
% fade.m
%
% Generate Rayleigh fading
%
% Programmed by H. Harada

function [iout,qout,ramp,rcos,rsin]=fade(idata,qdata,...
  nsamp,tstp,fd,no,counter,flat)

%********************** variables *********************
% idata  : input Ich data
% qdata  : input Qch data
% iout   : output Ich data
% qout   : output Qch data
% ramp   : Amplitude contaminated by fading
% rcos   : Cosine value contaminated by fading
% rsin   : Cosine value contaminated by fading
% nsamp  : Number of samples to be simulated
% tstp   : Minimum time resolution
% fd     : maximum doppler frequency
% no     : number of waves in order to generate fading
% counter  : fading counter
% flat     : flat fading or not

% (1-flat (only amplitude is fluctuated),0-normal
% (phase and amplitude are fluctutated))
%*******************************************************

if fd ~= 0.0
      ac0 = sqrt(1.0 ./ (2.0.*(no + 1)));
         % power normalized constant(ich)
      as0 = sqrt(1.0 ./ (2.0.*no));
         % power normalized constant(qch)
      ic0 = counter;
         % fading counter

      pai = 3.14159265;
      wm = 2.0.*pai.*fd;
      n = 4.*no + 2;
      ts = tstp;
      wmts = wm.*ts;
      paino = pai./no;

      xc=zeros(1,nsamp);
      xs=zeros(1,nsamp);
      ic=[1:nsamp]+ic0;

  for nn = 1: no
      cwn = cos( cos(2.0.*pai.*nn./n).*ic.*wmts );
      xc = xc + cos(paino.*nn).*cwn;
```

```
     xs = xs + sin(paino.*nn).*cwn;
  end

  cwmt = sqrt(2.0).*cos(ic.*wmts);
  xc = (2.0.*xc + cwmt).*ac0;
  xs = 2.0.*xs.*as0;

  ramp=sqrt(xc.^2+xs.^2);
  rcos=xc./ramp;
  rsin=xs./ramp;

  if flat ==1
     iout = sqrt(xc.^2+xs.^2).*idata(1:nsamp);
        % output signal(ich)
     qout = sqrt(xc.^2+xs.^2).*qdata(1:nsamp);
        % output signal(qch)

  else
     iout = xc.*idata(1:nsamp) - xs.*qdata(1:nsamp);
        % output signal(ich)
     qout = xs.*idata(1:nsamp) + xc.*qdata(1:nsamp);
        % output signal(qch)
  end

else
  iout=idata;
  qout=qdata;
end

%******************** end of file *******************
```

Program 2.6

```
% Program 2-6
% sefade.m
%
% This function generates frequency selecting fading...
%
% Programmed by H. Harada
%

function[iout,qout,ramp,rcos,rsin]=sefade(idata,qdata,
  itau,dlvl,th,n0,itn,n1,nsamp,tstp,fd,flat)

%******************** variables *******************
% idata  input Ich data
% qdata  input Qch data
% iout   output Ich data
% qout   output Qch data
% ramp   : Amplitude contaminated by fading
% rcos   : Cosine value contaminated by fading
% rsin   : Cosine value contaminated by fading
% itau   : Delay time for each multipath fading
% dlvl   : Attenuation level for each multipath fading
% th     : Initialized phase for each multipath fading
% n0     : Number of waves in order to generate each
%            multipath fading
% itn    : Fading counter for each multipath fading
% n1     : Number of summation for direct and delayed
%            waves
% nsamp   : Total number of symbols
% tstp   : Minimum time resolution
% fd   : Maximum doppler frequency
% flat     flat fading or not
% (1-flat (only amplitude is fluctuated),0-normal(phase
%            and amplitude are fluctuated))
%*******************************************************

iout = zeros(1,nsamp);
qout = zeros(1,nsamp);

total_attn = sum(10 .^( -1.0 .* dlvl ./ 10.0));

for k = 1 : n1

     atts = 10.^( -0.05 .* dlvl(k));
     if dlvl(k) = 40.0
            atts = 0.0;
     end

     theta = th(k) .* pi ./ 180.0;

     [itmp,qtmp] = delay (idata,qdata,nsamp,itau(k));
     [itmp3,qtmp3,ramp,rcos,rsin] = fade (itmp,qtmp,...
        nsamp,tstp,fd,n0(k),itn(k),flat);

  iout = iout + atts .* itmp3 ./ sqrt(total_attn);
  qout = qout + atts .* qtmp3 ./ sqrt(total_attn);

end

%******************** end of file *********************
```

Program 2.7

```
% Program 2-7
% delay.m
% Gives delay to input signal%
% Programmed by H. Harada
%
function [iout,qout] = delay(idata,qdata,nsamp,idel )

%********************** variables *********************
```

```
% idata  input Ich data
% qdata  input Qch data
% iout   output Ich data
% qout   output Qch data
% nsamp   Number of samples to be simulated
% idel   Number of samples to be delayed
%******************************************************

iout=zeros(1,nsamp);
qout=zeros(1,nsamp);

if idel ~= 0
  iout(1:idel) = zeros(1,idel);
  qout(1:idel) = zeros(1,idel);
end

iout(idel+1:nsamp) = idata(1:nsamp-idel);
qout(idel+1:nsamp) = qdata(1:nsamp-idel);

%******************** end of file *********************
```

Program 2.8

```
% Program 2-8
% bpskev.m
%
% Evaluation program of fading counter based BPSK
% transmission scheme
% This program is one of example simulations that
% include fading
% As for the explanation, you can check Chapter 3.
%
% Programmed by H. Harada
%

%****************** Preparation part ******************

% Time resolution
% In this case, 0.5us is used as an example
tstp = 0.5*1.0e-6;

% Symbol rate
% In this case we assume that each sample time is equal
% to 1/(symbol rate).
% In this case 200 kbps is considered.
sr = 1/tstp ;

% Arrival time for each multipath normalized by tstp
% In this simulation four-path Rayleigh fading is
% considered
itau = [0, 2, 3, 4];
```

```
% Mean power for each multipath normalized by direct
% wave.
% In this simulation four-path Rayleigh fading is
% considered.
% This means that the second path is -10dB less than the
% first direct path.
dlvl = [0 ,10 ,20 ,25];

% Number of waves to generate fading for each multipath.
% In this simulation four-path Rayleigh fading is
% considered.
% In normal case, more than six waves are needed to
% generate Rayleigh fading
n0=[6,7,6,7];

% Initial phase of delayed wave
% In this simulation four-path Rayleigh fading is
% considered.
th1=[0.0,0.0,0.0,0.0];

% Number of fading counter to skip (50us/0.5us)
% In this case we assume to skip 50 us
itnd0=100*2;

% Initial value of fading counter
% In this simulation four-path Rayleigh fading is
% considered.
% Therefore four fading counters are needed.

itnd1=[1000,2000, 3000, 4000];

% Number of direct wave + Number of delayed wave
% In this simulation four-path Rayleigh fading is
% considered
now1=4;

% Maximum Doppler frequency [Hz]
% You can insert your favorite value
fd=200;

% Number of data to simulate one loop
% In this case 100 data are assumed to consider
nd = 100;

% You can decide two mode to simulate fading by changing
% the variable flat
% flat     : flat fading or not
% (1-flat (only amplitude is fluctuated),0-normal(phase
% and amplitude are fluctuated))
flat =1;

%****************** START CALCULATION ***************
```

```
nloop = 1000;   % Number of simulation loop
noe = 0;        % Initial number of errors
nod = 0;        % Initial number of transmitted data
for iii=1:nloop

%******************* Data generation *******************

  data=rand(1,nd) > 0.5;  % rand: built in function

%******************* BPSK modulation *******************

  data1=data.*2-1;     % Change data from 1 or 0 notation
                       % to +1 or -1 notation

%******************* Fading channel ********************

% Generated data are fed into a fading simulator
% In the case of BPSK, only Ich data are fed into
% fading counter
  [data6,data7]=sefade(data1,zeros(1,length(data1)),...
      itau,dlvl,th1,n0,itnd1,now1,length(data1),tstp,...
      fd,flat);

  % Updata fading counter
  itnd1 = itnd1+ itnd0;

%***************** BPSK Demodulation *******************

  demodata=data6 > 0;

%***************** Bit Error Rate (BER) ****************

  % count number of instantaneous errors
  noe2=sum(abs(data-demodata));

  % count number of instantaneous transmitted data
  nod2=length(data);  % length: built in function

  fprintf('%d\t%e\n',iii,noe2/nod2);

  noe=noe+noe2;
  nod=nod+nod2;

end % for iii=1:nloop

%******************* Output result *********************

%ber = noe/nod;
fprintf('%d\t%d\t%e\n',noe,nod,noe/nod);

% ******************** end of file *********************
```

3

PSK-Based Digital Transmission Schemes

3.1 Introduction

When we transmit digital data like +1 or −1 by using radio waves, the best way is to modulate carrier signals with frequency f_c in accordance with the information of digital information data. The meaning of "modulate" is to vary the peculiar component that is included in the carrier signal wave. The waveform of the carrier signal is written as follows:

$$S(t) = A\cos\left\{2\pi f_c t + \theta(t)\right\} \tag{3.1}$$

where A, f_c, and $\theta(t)$ and are the amplitude, center frequency, and time-variant phase of the carrier wave signal, respectively. In (3.1), we have three peculiar components by which users can change the value. These three are amplitude, frequency, and phase, and if we change the amplitude of (3.1) in accordance with the digital information data, we call the modulation scheme AM. Moreover, if we change the frequency of (3.1) with information digital data, then we call the modulation scheme FM. Finally, if we change the phase of (3.1) in accordance with the digital information data, we call the modulation scheme PM or PSK.

This section focuses on PSK-based digital communication, describes the basic configurations of the transmitter and receiver, and explains how to express the configurations by using computer simulations. Section 3.2 describes BPSK. Then, Section 3.3 explains QPSK to increase the transmission rate with high-frequency utilization efficiency in comparison with BPSK. QPSK has several problems regarding its shared bandwidth from the viewpoint of the development of the prototype. To solve these problems, we introduce three modulation schemes: OQPSK, MSK, and GMSK in Sections 3.4–3.6, respectively. Moreover, to realize broadband data transmission in the limited bandwidth, we introduce QAM in Section 3.7. Each section also shows how to evaluate these systems by computer simulation. Finally, Section 3.8 concludes the chapter with a brief summary. All programs related to the chapter are presented in Appendix 3A and in the CD-ROM that accompanies the book.

3.2 BPSK

3.2.1 Basic Configuration of BPSK Transmission Scheme

In the modulation scheme, the input digital data 0 or 1 is directly converted to the phase of 0 or π, respectively. Therefore, the waveform is shown as follows:

$$S(t) = A\cos\left\{2\pi f_c t + \pi \cdot d_k\right\} \tag{3.2}$$

where d_k is the information data sequence.

Figure 3.1 illustrates a BPSK signal generation method. As shown in Figure 3.1, the waveform of a BPSK wave is generated by multiplying between the digital signal data and carrier wave. However, as for the limitation of the frequency bandwidth, we must control the shape with an adequate pulse-shaping filter. Therefore, in the procedure of BPSK signal generation, first of all, digital data are fed into a pulse-shaping filter circuit. Section 3.2.3 describes the method for the design of the adequate pulse-shaping. Then, the pulse-shaped signal is converted to an analog signal by a D/A converter, upconverted to the RF frequency by multiplying by carrier signal wave, and finally transmitted to the air.

At the receiver, the received wave passes a *bandpass filter* (BPF), where the spurious wave is eliminated. Then, the received signal is downconverted to the baseband by multiplying the received radio signal by the RF carrier frequency signal. Then, the signal is converted to digitally sampled data with an A/D converter, and the transmission digital data is recovered by DSPH. In the DSPH, the sampled data is filtered to eliminate the symbol inter- ference at a pulse-shaping filter circuit. Finally, a synchronization point is selected from the filtered digital sample signal. If the signal level is larger than 0 at the point, we can obtain the received digital data 1; otherwise, the received data becomes 0

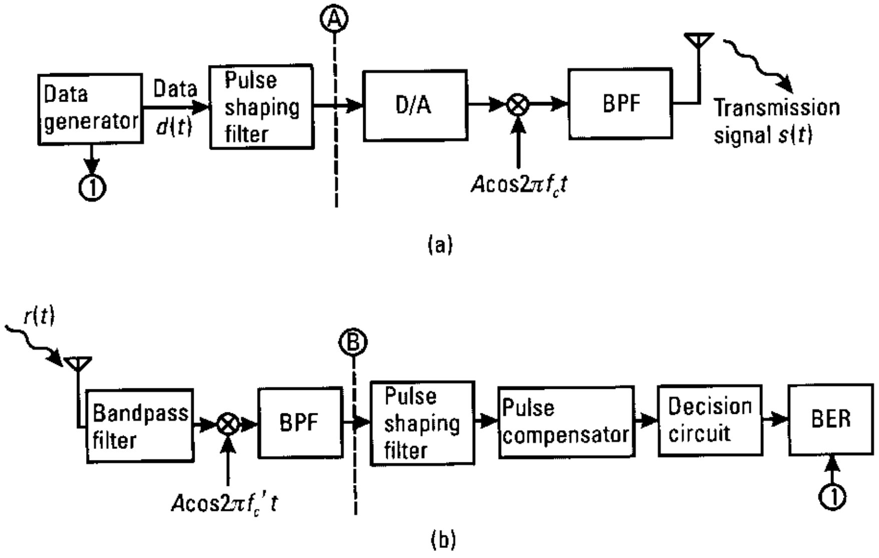


Figure 3.1 (a) Transmitter and (b) receiver of the BPSK transmission scheme.

3.2.2 Theoretical Notation of Signals

We first derived the theoretical education for the BPSK transmission scheme because we needed it for our computer-simulation program.

The transmission digital data sequence is given as:

$$d(t) = \sum_{k=-\infty}^{+\infty} d_k \cdot \left(g_T(t) \otimes \delta(t - kT_b)\right) \tag{3.3}$$

$$= \sum_{k=-\infty}^{+\infty} d_k \cdot g_T\left(t - kT_b\right) \tag{3.4}$$

where $d_k(d_k : k = 1, 2, \ldots)$, $g_T(t)$, and T_b are the transmission digital data, the pulse shape of each transmission digital data, and the bit duration, respectively. The reciprocal of T_b is the bit rate, and $\delta(t)$ is a delta function:

$$\delta(t) = \begin{cases} 1 : t = 0 \\ 0 : \text{otherwise} \end{cases} \tag{3.5}$$

When $g_T(t)$ is a trailing rectangular pulse,

$$g_T(t) = \begin{cases} 1\,; & -\dfrac{T_b}{2} \le t \le \dfrac{T_b}{2} \\ 0\,; & \text{otherwise} \end{cases} \tag{3.6}$$

where the relationship between d_k, $g_T(t)$, $\delta(t)$, and $d(t)$ is shown in Figure 3.2. In this case, $\{d_k : k = 1 \ldots 6\} = [1, 0, 1, 1, 0, 0]$ and "0" data is converted to "1." While the pulse shape in Figure 3.2 looks digital, from the viewpoint of the frequency domain, the shape radiates large spurious signals for other frequency channels.

Figure 3.3 illustrates an example $g_T(t)$ and its Fourier transmission value, $G_T(f)$. As shown in Figure 3.3, to include all information in the area $|f| < \dfrac{1}{2T_b}$, the pulse shape in the time domain must take a sinc $(= \dfrac{\sin(x)}{x})$ function. This shape differs from that shown in Figure 3.2. In this case, spurious signals appear in the area

$$|t| > \frac{T_b}{2}$$

We thus need a filter with an adequate optimum shape that can reduce the number of spurious signals in the time and frequency domains.

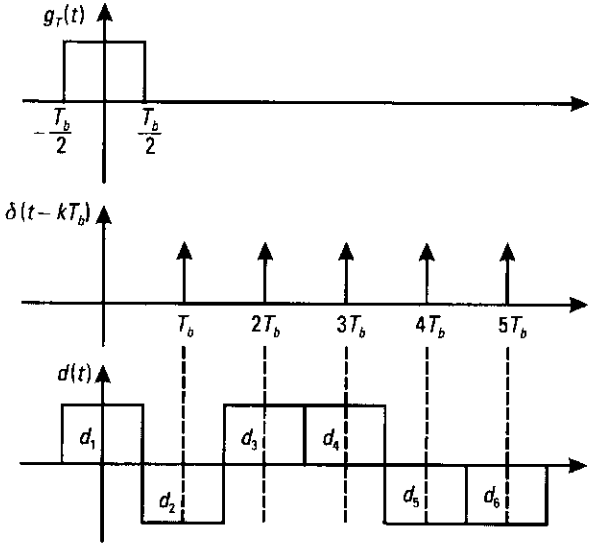


Figure 3.2 Relationship between $g_T(t)$, $\delta(t)$, and $d(t)$.

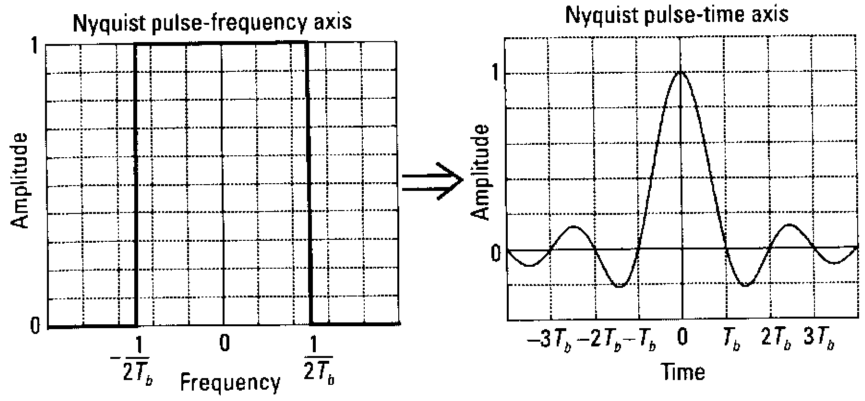


Figure 3.3 An example of $g_T(t)$ and its Fourier transmission value.

A widely used filter that reduces the number of spurious signals is the Nyquist filter. The frequency response of the Nyquist filter is given by [1]

$$G_N(f) = \begin{cases} 1 & 0 \le |f| \le \dfrac{(1-\alpha)}{2T_b} \\ \cos^2\left[\dfrac{T_b}{4\alpha}\left\{2\pi|f| - \dfrac{\pi(1-\alpha)}{T_b}\right\}\right] & \dfrac{(1-\alpha)}{2T_b} \le |f| \le \dfrac{(1+\alpha)}{2T_b} \\ 0 & |f| > \dfrac{(1+\alpha)}{2T_b} \end{cases} \tag{3.7}$$

where α is the roll-off factor, which determines the channel bandwidth.

Figure 3.4 illustrates $G_N(f)$ in the frequency domain and its impulse response in the time domain. The amplitude of the spurious signals in the frequency domain increases in the area $|f| > \dfrac{1}{2T_b}$ as roll-off factor α increases. In contrast, the amplitude of the spurious signals in the time domain decreases in the area $|t| > T_b$. Therefore, we can find a compromise value for the roll-off factor.

One feature of the Nyquist filter is that we always obtain 0 at nT_b (n is an integer) in the time domain. Therefore, when we set the synchronization point at nT_b, one symbol never interferes with other symbols at this point. Although the Nyquist filter has several useful features, there is a question of when and where to use it for an optimum result. Reference [1] discusses the optimum allocation method to maximize the synchronization point in an

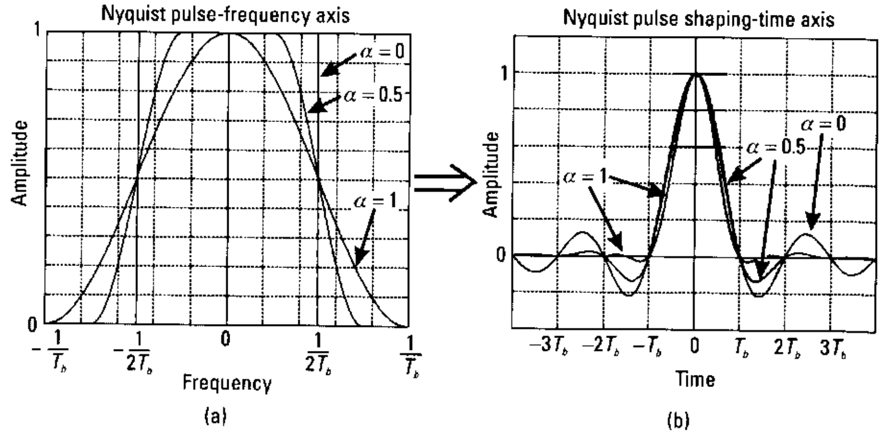


Figure 3.4 Configuration of the Nyquist filter: (a) $G_N(f)$ and (b) impulse response of $G_N(f)$.

AWGN environment. They found that two transmission and receiver filters with an equally distributed gain of $G_N(f)$, namely, the $G_T(f) = \sqrt{G_N(f)}$ and $G_R(f) = \sqrt{G_N(f)}$, where $G_T(f)$ and $G_R(f)$ are the frequency responses of the transmitter and receiver filters:

$$G_T(f) = G_R(f) = \sqrt{G_N(f)}$$

$$= \begin{cases} 1 & 0 \le |f| \le \dfrac{(1-\alpha)}{2T_b} \\ \cos\left[\dfrac{T_b}{4\alpha}\left\{2\pi|f| - \dfrac{\pi(1-\alpha)}{T_b}\right\}\right] & \dfrac{(1-\alpha)}{2T_b} \le |f| \le \dfrac{(1+\alpha)}{2T_b} \\ 0 & |f| > \dfrac{(1+\alpha)}{2T_b} \end{cases} \tag{3.8}$$

Such a filter is sometimes called a root Nyquist filter, and its impulse responses $g_T(t)$ and $g_R(t)$ are given as

$$g_T(t) = g_R(t)$$

$$= \frac{1}{\pi t} \frac{1}{1 - \left(\dfrac{4\alpha t}{T_b}\right)^2} \sin\left\{2\pi(1-\alpha)\frac{1}{T_b}\right\} + \frac{1}{\pi} \frac{4\alpha / T_b}{1 - \left(\dfrac{4\alpha t}{T_b}\right)^2} \cos\left\{2\pi(1+\alpha)\frac{1}{T_b}\right\} \tag{3.9}$$

Using (3.4) and (3.6), we can configure the BPSK transmission of the data. The configured signal is modulated by multiplying the carrier frequency signal (e.g., $\cos 2\pi fct$) and then transmitted to air. The transmitted signal is given by

$$s(t) = d(t)\cos 2\pi f_c t \tag{3.10}$$

The transmitted signal is contaminated by multipath fading and AWGN, and at the receiver it is received as

$$r(t) = \int_0^\infty h(\tau, t) \otimes s(t - \tau)d\tau + n(t) \tag{3.11}$$

where $h(\tau, t)$ is the impulse response of the radio channel at time t, and $n(t)$ is the receiver noise.

In the receiver, the received signal is first filtered by a BPF, which is assumed to have a sufficient passband so that the signal is negligibly distorted. The filtered signal is multiplied by a carrier wave that has the same frequency as the transmitter. However, the initial phase of the carrier signal source is different between the transmitter and receiver. Therefore, the carrier signal source at the receiver is mentioned as $\cos(2\pi f_c + \theta_1(t))$ where $\theta_1(t)$ is the initial phase value of the carrier signal. The multiplied signal is given by

$$r_2(t) = r(t) \times \cos(2\pi f_c t + \theta_1(t)) \tag{3.12}$$

where $r(t)$ is given by (3.11). For simplification we assume no filtering, so

$$\begin{aligned} r(t) &= R(t)\begin{pmatrix} \cos\theta_f(t) & -\sin\theta_f(t) \\ \sin\theta_f(t) & \cos\theta_f(t) \end{pmatrix}\begin{pmatrix} d(t)\cos 2\pi f_c t \\ 0 \end{pmatrix} + n(t) \\ &= \left[R(t)d(t)\left(\cos\theta_f(t) + n_1(t)\right)\right]\cos 2\pi f_c t + n_2(t) \end{aligned} \tag{3.13}$$

where $R(t)$ and $\theta_f(t)$ are the time-variant fluctuations of the amplitude and phase, respectively, by radio channel. The $n(t)$ is the receiver noise; it is given by $n(t) = R(t) \times n_1(t) \times \cos 2\pi f_c t + n_2(t)$, where $n_1(t)$ and $n_2(t)$ are the noise component around f_c and the other noise component, respectively. In addition,

$$\begin{pmatrix} \cos\theta_f(t) & -\sin\theta_f(t) \\ \sin\theta_f(t) & \cos\theta_f(t) \end{pmatrix}$$

is a rotation matrix with the phase of $\theta_f(t)$. Using (3.12), we can expand (3.11). When we use LPF for the expansion value, the high-frequency signal is eliminated, so

$$r_3(t) = \frac{1}{2}\Big(R(t)d(t)\big(\cos\theta_f(t) + n_1(t)\big)\Big)\cos\theta_1(t) \tag{3.14}$$

Then, $r_3(t)$ is filtered by a pulse-shaping, filter-based root Nyquist filter to reduce ISI.

$$\begin{aligned} r_4(t) &= r_3(t)\otimes g_R(t) \\ &= \frac{1}{2}R(t)\cos\theta_f(t)\cos\theta_1(t)\cdot\sum_{k=-\infty}^{+\infty} d_k \\ &\cdot\big(g_T(t)\otimes g_R(t)\otimes\delta(t-kT_b)\big) + \frac{1}{2}n_1(t)\big(\cos\theta_1(t)\otimes g_R(t)\big) \\ &= \frac{1}{2}R(t)\cos\theta_f(t)\cos\theta_1(t)\cdot\sum_{k=-\infty}^{+\infty} d_k \\ &\cdot\big(g_T(t)\otimes g_R(t)\otimes\delta(t-kT_b)\big) + n_3(t) \end{aligned} \tag{3.15}$$

Next, the filtered signal is oversampled at a sampling rate of $\frac{n}{T_b}$ (n is an integer) by using an A/D converter. The sampled signal is

$$r_4\left(k\cdot\frac{T_b}{n}\right) k = 0, 1, 2, \ldots$$

When $t = u\cdot T_b/n$ (u is an integer), we can obtain the maximum value of the pulse from the characteristics of the Nyquist filter. However, we must consider the synchronization method. In this section, we assume that we know the synchronization point. Then, we resample

$$r_4\left(u\cdot\frac{T_b}{n}\right)$$

every T_b or every n sample from the synchronization point. Finally, we obtain resampled data $r_5(l) = r_4$ (syncpoint $+\, l\cdot T_b$) $l = 0, 1, 2, \ldots$. We can then decide whether the received data is a 1 or a 0 by using the threshold condition equation:

$$\hat{d}_k = \begin{cases} 1 & \big(r_5(l) > 0\big) \\ 0 & \big(r_5(l) < 0\big) \end{cases} \tag{3.16}$$

By comparing $\hat{d}_k$ and d_k, we can calculate the BER, which depends on the volume of receiver noise. Theoretical BER in an AWGN and one-path Rayleigh fading channels have been reported [1–3].

$$\mathrm{BER}_{\mathrm{BPSK-AWGN}} = \frac{1}{2}\mathrm{erfc}\left(\sqrt{E_b/N_0}\right) \tag{3.17}$$

$$\mathrm{BER}_{\mathrm{BPSK-FADING}} = \frac{1}{2}\left[1 - \frac{1}{\sqrt{1+\dfrac{1}{E_b/N_0}}}\right] \tag{3.18}$$

The E_b/N_0 is the ratio between the energy per bit (E_b) and the noise power density N_0. Figure 3.5 illustrates the theoretical relationship of BPSK between E_b/N_0 (dB) and BER with AWGN and one-path Rayleigh fading channels.

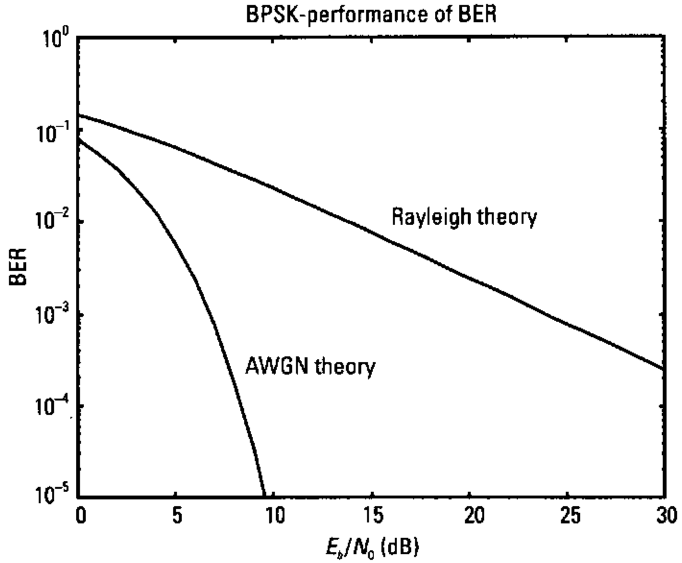


Figure 3.5 BER performance of BPSK transmission scheme under AWGN and Rayleigh fading environments (theoretical value).

3.2.3 Computer Simulation Program

To measure the BER performance of the BPSK scheme, we used computer simulation. The model we used is illustrated in Figure 3.6, and the three simulation programs we used are shown below. All functions used in this chapter are summarized in Table 3.1. Program 3.1 is the main program for BPSK. We will use it here to explain the procedures of the computer simulation.

Before using Program 3.1, we must set the values of the common variables used in the program.

```
sr=256000;        % sr : symbol rate
ml=1;             % ml : number of modulation levels
br=sr.*ml;        % br : bit rate
nd=1000;          % nd : number of symbols
ebn0=100;         % ebn0 : Eb/No
IPOINT=8;         % IPOINT : number of oversamples
```

For this program, we must assume an important point. As Section 3.2 explains, we oversample the received signal at the receiver. Therefore, we assume that the minimum time unit is equal to the oversampling duration. Namely, $\frac{1}{sr \cdot IPOINT}$ is the minimum time unit.

After defining several variables, we must set the filter coefficients of the root Nyquist filter. The filter coefficient is calculated using function hrollfcoef.m, which is in Program 3.2. By setting the last argument to 0 or 1, we generate a filter for the transmitter or receiver, respectively. To generate the transmitter filter coefficients

```
alfs=0.5;         % alfs  : roll-off factor
irfn=21;          % irfn  : number of filter taps
[xh]=hrollcoef(irfn, IPOINT, sr, alfs, 1);
```

is entered. To generate the receiver filter coefficients,

```
[xh2]=hrollcoef(irfn, IPOINT, sr, alfs, 0);
```

is entered. The information about filters is stored in xh and xh2.

Next we define several variables common to all the programs

```
nloop=100;        % nloop : number of loops
noe=0;            % noe : number of errors
nod=0;            % nod : number of data
```

After defining all of the variables, we can run the simulation to obtain the BER. First, we generate random data (0 or 1), in a 1-by-nd*ml vector called

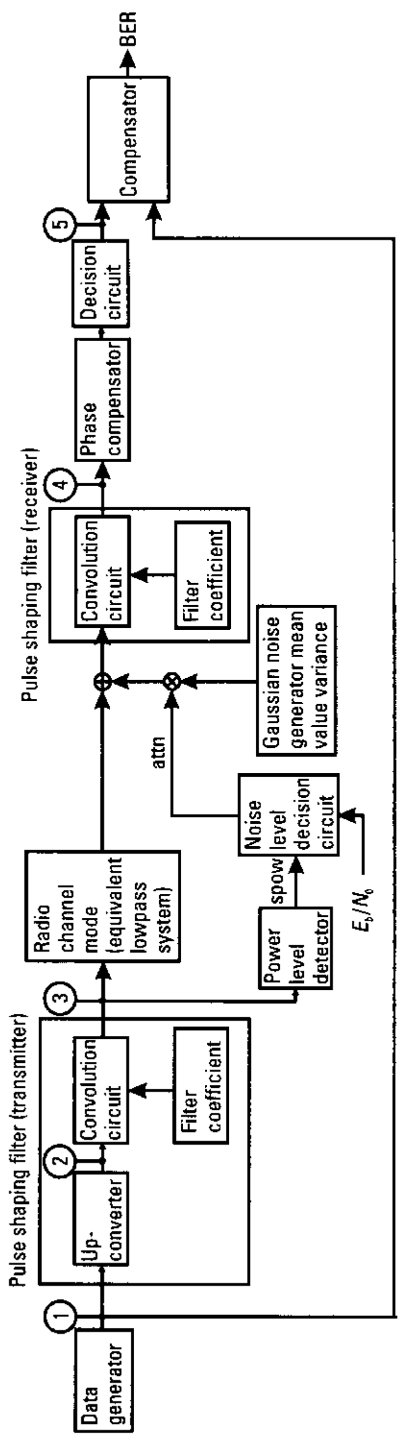


Figure 3.6 Model of the BPSK transmission system used to calculate BER performance.

Table 3.1
Overview of Simulation Functions in This Chapter

Name of Function	Function	Input Data	Output
bpsk.m (Program 3.1) bpsk_fading.m (Program 3.2)	Calculate BER performance under AWGN and one-path Rayleigh fading channel (BPSK)	None	BER: bit error rate performance
hrollfcoef.m (Program 3.3)	Calculate filter coefficients of root Nyquist filter in accordance with input symbol or sampling rate	irfn: Number of symbols to use filtering ipoint: Number of samples in one symbol sr: symbol rate alfs: roll-off coefficient ncc: 1-transmission factor 0-receiver factor	xh: coefficient value in accordance with input symbol or sampling rate
oversamp.m (Program 3.4)	Output oversampled data of IPOINT times	indata: input sequence nsymb: number of symbols in an input sequence sample: Number of oversample	outdata: oversampled data
qpsk.m (Program 3.5) qpsk_fading.m (Program 3.6)	Calculate BER performance under AWGN and one-path Rayleigh fading channel (QPSK)	None	BER: bit error rate performance
compconv.m (Program 3.7)	Perform convolution between complex signal and filter	idata: input I-channel data qdata: input Q-channel data filter: filter tap coefficient	iout: filtered I-channel data qout: filtered Q-channel data
compoversamp.m (Program 3.8)	Output oversampled I and Q channel data of IPOINT times	idata: input I-channel data qdata: input Q-channel data nsymb: number of symbols in an input sample: Number of oversample	iout: oversampled channel data qout: oversampled Q-channel data
qpskmod.m (Program 3.9)	Output modulated data of QPSK modulation	paradata: parallel serial data para: the number of parallels nd: the number of symbols ml: the number of modulations	iout: modulated I-channel data qout: modulated Q-channel data
qpskdemod.m (Program 3.10)	Output demodulated data of QPSK modulation	idata: input I-channel data qdata: input Q-channel data para: the number of parallels nd: the number of symbols ml: the number of modulations	demodata: demodulated serial data
comb.m (Program 2.4)	Add AWGN	idata: input I-channel data qdata: input Q-channel data attn: amplitude of noise in each I and Q channel	iout: output of I-channel data qout: output of Q-channel data
oqpsk.m (Program 3.11) oqpsk_fading.m (Program 3.12)	Calculate BER performance under AWGN and one-path Rayleigh fading channel (QPSK)	None	BER: bit error rate performance
msk.m (Program 3.13) msk_fading.m (Program 3.14)	Calculate BER performance under AWGN and one-path Rayleigh fading channel (MSK)	None	BER: bit error rate performance
msk2.m (Program 3.15) msk2_fading.m (Program 3.16)	Calculate BER performance under AWGN and one-path Rayleigh fading channel (MSK)	None	BER: bit error rate performance

Table 3.1 (continued)

Name of Function	Function	Input Data	Output
oversamp2.m (Program 3.17)	Output oversampled I and Q channel data of IPOINT times for MSK	in: input sequence ntot: number of symbols in an input sequence sample: Number of oversample	out: oversampled data
gmsk.m (Program 3.18) gmsk_fading.m (Program 3.19)	Calculate BER performance under AWGN and one-path Rayleigh fading channel (GMSK)	None	BER: bit error rate performance
gaussf.m (Program 3.20)	Form Gaussian filter	B: `filter` coefficient irfn: Number of symbols to use filtering ipoint: Number of samples in one symbol sr: symbol rate ncc: 1-transmission factor 0-receiver factor	xh: coefficient value in accordance with input symbol or sampling rate
qam16.m (Program 3.21) qam16_fading.m (Program 3.22)	Calculate BER performance under AWGN and one-path Rayleigh fading channel (16-QAM)	None	BER: bit error rate performance
qammod.m (Program 3.23)	Output modulated data of 16-QAM modulation	paradata: input serial data para: the number of parallels nd: the number of symbol ml: the number of modulation	iout: modulated I-channel data qout: modulated Q-channel data
qamdemod.m (Program 3.24)	Output demodulated data of 16-QAM modulation	idata: input I-channel data qdata: input Q-channel data para: the number of parallels nd: the number of symbol ml: the number of modulation	demodata: demodulated serial data

`data1`. Section 2.3 explains the method for generating the data. It is done by simply entering a command:

```
data=rand(1,nd*ml)>0.5;
```

Next, we convert the notation of the `data` from 0 and 1 digital data to −1 and 1 digital data. The converted data is stored in variable `data1`:

```
data1= 2.*data-1;
```

The data to be transmitted is shown in Figure 3.7(a). It is pulse-shaped as follows:

1. Using `data1`, we make an impulse train sequence $d(t) = \sum_{k=-\infty}^{+\infty} d_k \delta(t - kT_b)$ as shown in (3.2), where the minimum time duration is `1/(sr·IPOINT)`. The function `oversample.m` is used to generate the impulse train sequence as follows

    ```
    data2=oversamp(data1,nd, IPOINT); .
    ```

    The waveform of `data2` is shown in Figure 3.7(b).

2. To generate the impulse train sequence, we perform convolution using filter coefficient `xh`. The filtered data is given by

    ```
    data3=conv(data2, xh);  .
    ```

    The waveform of `data3` is shown in Figure 3.7(c).

The transmission signal is then passed through a radio channel (an equivalent lowpass system) and transmitted. At the receiver, the received signal is first contaminated by AWGN. The noise function used, `comb.m`, was described in Section 2.4. If we use this function in the simulation, we need not input `qch` data. Moreover, we must decide variable `attn`.

We want to determine the relationship between E_b/N_0 and BER. That means we must vary `attn` while keeping E_b/N_0 constant. The variable `attn` is calculated as follows. First, energy per bit E_b and noise power density N_0 are defined:

$$E_b = \frac{spow}{br} \quad (\mathrm{W} \cdot T/\mathrm{bit}) \qquad (3.19)$$

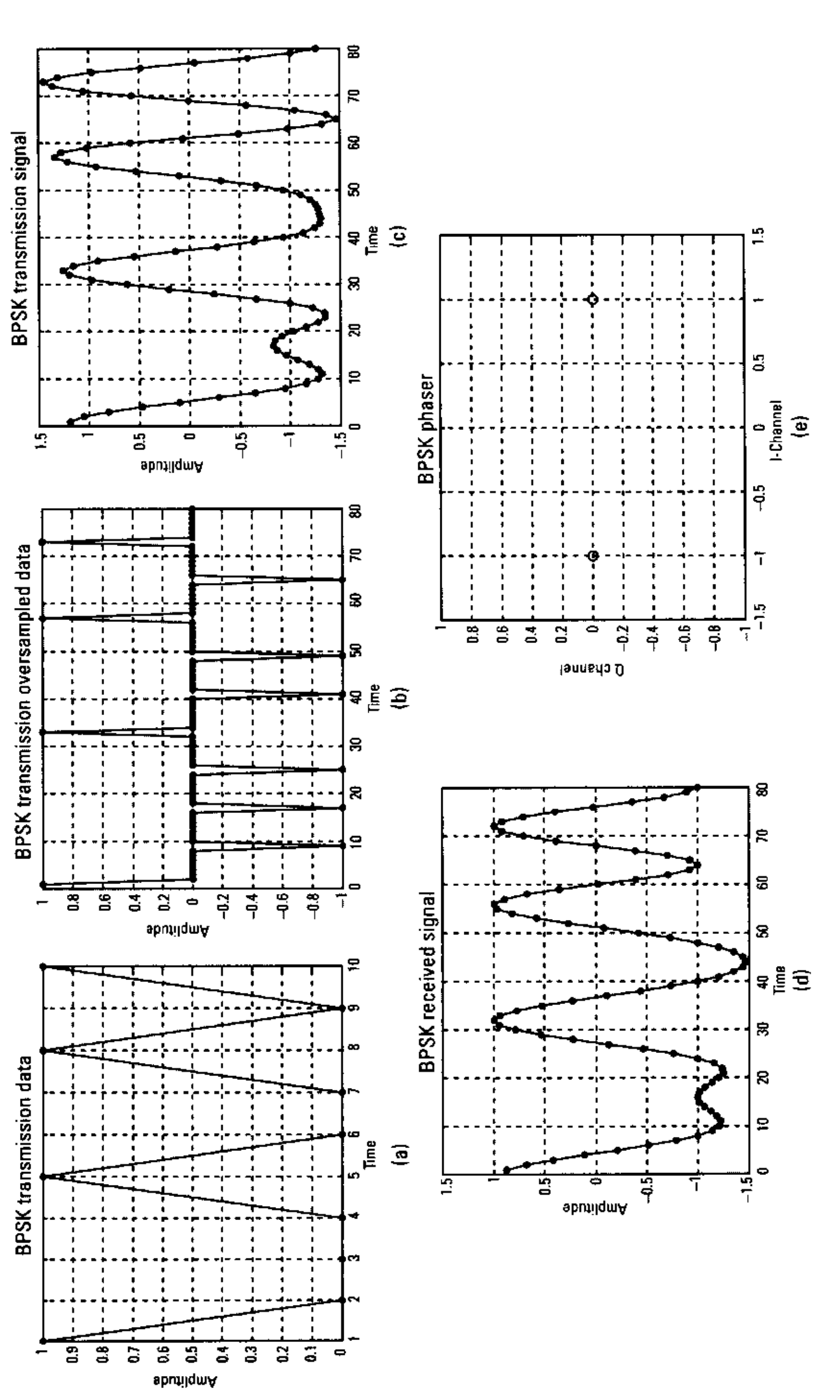


Figure 3.7 Simulated instantaneous BPSK waveform: (a) BPSK transmission data, (b) BPSK transmission oversampled data, (c) BPSK transmission signal, (d) BPSK received signal, and (e) BPSK phaser.

$$N_0 = \frac{npow}{sr} \quad (\text{W/Hz}) \tag{3.20}$$

where spow, npow, br, and sr are the signal power per symbol, the noise power per symbol, the bit rate, and the symbol rate, respectively.

From combining (3.19) and (3.20), we get

$$E_b / N_0 = \frac{spow}{br} \cdot \frac{sr}{npow} \tag{3.21}$$

After manipulation, we get

$$npow = \frac{spow}{br} \cdot \frac{sr}{E_b / N_0} \tag{3.22}$$

Since E_b/N_0 is in decibels, (3.22) can be written as

$$npow = \frac{spow}{br} \cdot \frac{sr}{10^{\frac{E_b/N_0}{10}}} \tag{3.23}$$

meaning that we can calculate npow. In this simulation data, data2, and data3 are expressed in voltage, so the noise signal must be expressed in voltage. Gaussian noise is normally distributed equally in in-phase and quadrature-phase channels. Therefore,

$$attn = \sqrt{\frac{1}{2} npow} \tag{3.24}$$

In Program 3.1, we express these relationships as

```
spow=sum(data3.*data3)/nd;
attn=0.5*spow*sr/br*10.^(-ebn0/10);
attn=sqrt(attn);
```

Using attn, we contaminate the transmitted data with AWGN.

```
inoise=randn(1,length(data3)).*attn;
data4=data3+inoise;
```

where randn(a,b) is a built-in function that generates Gaussian noise with mean a and variance b. The received signal, data4, is filtered with a root Nyquist filter, with coefficient xh2, which is set at the beginning of the simulation.

```
data5=conv(data4, xh2);
```

To use the conv.m function, we must recover the time delay due to convolution, which has length(xh). The time delay of each conv.m is $\frac{irfn + IPOINT}{2}$. Since we use two conv.m functions, the time delay is irfn+IPOINT samples. Therefore, we resample data5 from sample point irfn*IPOINT+1. We define the start point as sampl:

```
sample=irfn*IPOINT+1;
```

The filtered signal data5 is an oversampled signal and shown in Figure 3.7(d). To recover the transmitted data, we must select optimum points from the oversampled signal. As shown in Figure 3.7(d), we can obtain a 1 or −1 value by resampling data5 every IPOINT(=8) samples from the start point. The resampled value is stored in data6. From the operation, we can understand that the transmission signal is recovered. The phase of the recovered signal is shown in Figure 3.7(e).

```
data6=data5(sampl: 8:8*nd+sampl-1);
```

If the received signal is contaminated by AWGN, the value of data6 is not 1 or 0, and we need to decide whether the value is 0 or 1. The threshold value for deciding is given in (3.16). We use it to execute the following condition function:

```
demodata=data6>0;
```

If each element is larger than 0, the data is considered to be a 1, otherwise 0.

Next, we investigate the number of errors. The procedure was described in Section 2.3. In this simulation, the transmission data is data, and, the received data is data6.

```
noe2=sum(abs(data-demodata));
nod2=length(data);
noe=noe+noe2;
nod=nod+nod2;
```

Finally, the BER is given by

```
ber=noe/nod;
```

For the simulation in the Rayleigh fading environment, we can use Program 3.2. The difference between Programs 3.1 and 3.2 is shown as follows. In Program 3.2 we first set the fading parameters discussed in Section 2.4.2. In this case we try to generate one-path Rayleigh fading.

```
%*************Fading initialization **********

% Time resolution
tstp=1/sr/IPOINT;
% Arrival time for each multipath normalized by tstp
itau = [0];
% Mean power for each multipath normalized by direct wave.
dlvl = [0];
% Number of waves to generate fading for each multipath.
n0=[6];
% Initial phase of delayed wave
th1=[0.0];
% Number of fading counter to skip
itnd0=nd*IPOINT*100;

% Initial value of fading counter
itnd1=[1000];
% Number of direct wave + Number of delayed wave
now1=1;
% Maximum Doppler frequency [Hz]
fd=160;
% flat     : flat fading or not
% (1-flat (only amplitude is fluctuated),0-normal
% (phase and amplitude are fluctuated))
flat =1;
```

Moreover, by using the above parameters, the transmitted BPSK signal data3 is fluctuated by fading simulator.

```
%******************* Fading channel *******************

% Generated data are fed into a fading simulator

[ifade,qfade]=sefade(data3,zeros(1,length(data3)),itau,
  dlvl,th1,n0,itnd1,now1,length(data3),tstp,fd,flat);

% Updata fading counter
itnd1 = itnd1+ itnd0;
```

The other function is basically the same as Program 3.1. To evaluate BER performance under AWGN and fading environments by using Program 3.1 (bpsk.m) and Program 3.2 (bpsk_fading.m), we just type the following command

```
clear
bpsk
```

or

```
clear
bpsk_fading
```

The simulated results are shown in Figure 3.8.

As shown in Figure 3.8, the simulated results closely matched the theoretical value. The BPSK scheme has a simple configuration. Several modulation schemes based on BPSK have been proposed to obtain higher frequency-utilization efficiency. Sections 3.3–3.7 introduce QPSK, OQPSK, MSK, GMSK, and QAM as representative examples.

3.3 QPSK

3.3.1 Basic Configuration of Quadrature Modulation Scheme

A QPSK signal is generated by two BPSK signals. To distinguish the two signals, we use two orthogonal carrier signals. One is given by $\cos 2\pi f_c t$, and the other is given by $\sin 2\pi f_c t$. The two carrier signals remain orthogonal in the area of a period

$$\int_0^{T_c} \cos 2\pi f_c t \times \sin 2\pi f_c t = 0 \qquad (3.25)$$

where T_c is the period of the carrier signals and

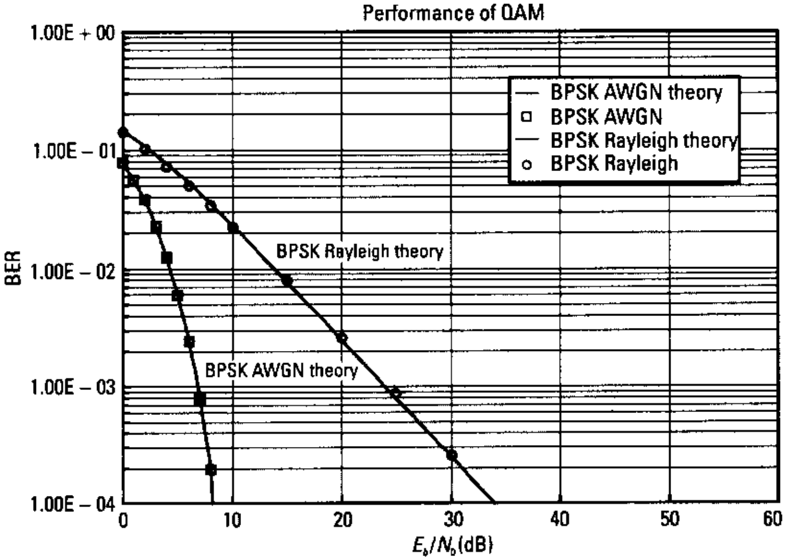


Figure 3.8 Theoretical and simulated BER performance of BPSK under AWGN and fading environments.

$$f_c = \frac{1}{T_c}$$

By using $\cos 2\pi f_c t$ and $\sin 2\pi f_c t$, we can represent QPSK signals by

$$s(t) = \frac{1}{\sqrt{2}} d_I(t) \cos(2\pi f_c t) + \frac{1}{\sqrt{2}} d_Q(t) \sin(2\pi f_c t) \qquad (3.26)$$

A channel in which $\cos 2f_c t$ is used as a carrier signal is generally called an in-phase channel, or Ich, and a channel in which $\sin 2f_c t$ is used as a carrier signal is generally called a quadrature-phase channel, or Qch. Therefore, $d_I(t)$ and $d_Q(t)$ are the data in Ich and Qch, respectively. Modulation schemes that use Ich and Qch are called *quadrature modulation schemes.* The basic configuration is shown in Figure 3.9.

In this system the input digital data, d_k : $k = 1, 2, \ldots$ is first converted into parallel data with two channels, Ich and Qch. The data are represented as $d_I(t)$ and $d_Q(t)$. The data is allocated using a mapping circuit block. Then, the data allocated to Ich is filtered using a pulse-shaping filter in Ich. Then, the pulse-shaped signal is converted into an analog signal by a D/A converter and multiplied by a cos $2\pi f_c t$ carrier signal. The data allocated to Qch is also filtered using a pulse-shaping filter. The pulse-shaped signal is converted into an analog signal by a D/A converter and multiplied by a sin $2\pi f_c t$ carrier signal. Then, the Ich and Qch signals are added and transmitted to the air.

At the receiver, the received wave first passes through a BPF, where spurious waves are eliminated. The received signal is then down-converted to the baseband by multiplying it by the RF carrier frequency. For Ich and Qch, we use $\cos[2\pi f_c t + \theta_1(t)]$ and $\sin[2\pi f_c t + \theta_1(t)]$, respectively, where $\theta_1(t)$ is the phase noise of the frequency source, the difference between the frequency sources of the transmitter and receiver. Then, in both Ich and Qch channels, the downconverted signal is digitally sampled by A/D converters, and the digital data is fed to DSPH. In the DSPH, the sampled data is filtered with a pulse-shaping filter to eliminate ISI. The signals are then synchronized, and the transmitted digital data is recovered.

Based on this quadrature modulation scheme, Section 3.3.2 discusses several modulation schemes.

3.3.2 Basic Configuration of QPSK Transmission Scheme

QPSK basically uses the configuration shown in Figure 3.9 with several blocks, specialized for QPSK. They are the ones for mapping, demapping, and pulse shaping.

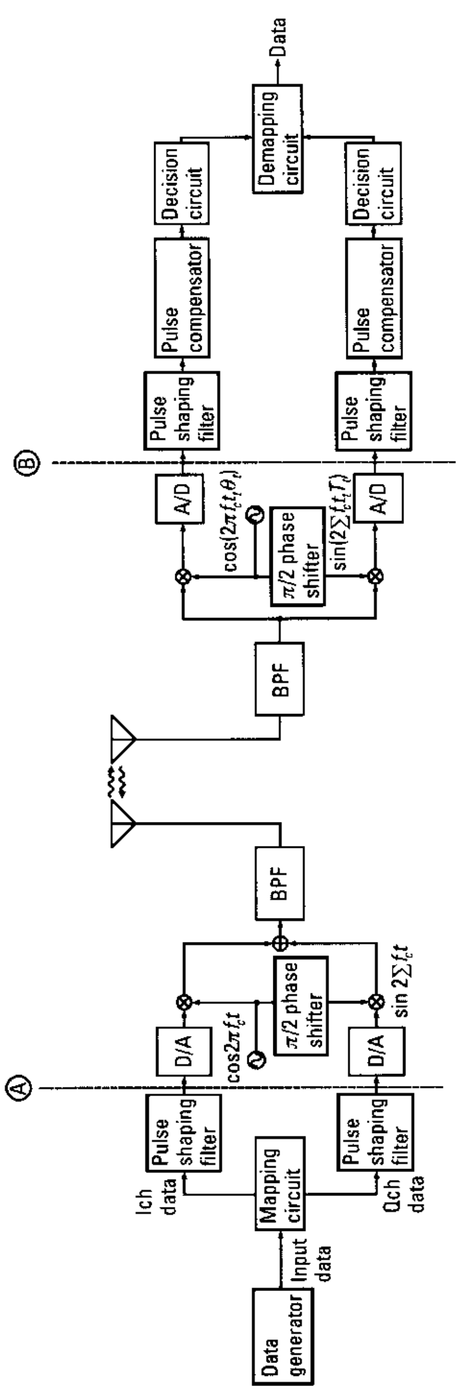


Figure 3.9 Basic configuration of the quadrature modulation scheme.

For mapping and demapping, we use the simple circuit shown in Figure 3.10. For pulse shaping, we use the root Nyquist filter described in Section 3.2.2. The theoretical BER values with AWGN and one-path Rayleigh fading have the QPSK transmission scheme given in [2].

$$\mathrm{BER}_{\mathrm{QPSK\_AWGN}} = \frac{1}{2}\mathrm{erfc}\left(\sqrt{E_b/N_0}\right) \tag{3.27}$$

$$\mathrm{BER}_{\mathrm{QPSK\_FADING}} = \frac{1}{2}\left[1 - \frac{1}{\sqrt{1+\dfrac{1}{E_b/N_0}}}\right] \tag{3.28}$$

In the simulation under the Rayleigh fading environment, we assume that the frequency rotation is compensated for.

3.3.3 Computer Simulation

The model we used to simulate the BER performance of QPSK is shown in Figure 3.11. A comparison of Figure 3.11 to Figure 3.6 shows how quadrature modulation differs from BPSK modulation. In quadrature modulation, there are two simulation flows, Ich and Qch. The simulation of these flows is independent and the same as that for BPSK. The simulation is thus quite easy to understand. The programs are Programs 3.5–3.10. In the main program, Program 3.5, we first decide which common variables are to be used throughout Program 3.5.

```
sr=256000;        % sr : Symbol rate
ml=2;             % ml : number of modulation levels
br=sr. *ml;       % br : bit rate
nd=1000;          % nd : number of symbols
ebn0=3;           % ebn0 : Eb/No
IPOINT=8;         % IPOINT : number of oversamples
```

We next decide the filter coefficients of the root Nyquist filter. The procedures are those used with BPSK (see Section 3.2.3).

Then, we define several variables common to all the programs, the same as with BPSK. (See Section 3.2.3.)

After defining all of the variables, we can run the simulation to obtain the BER. First, we generate random data (0 or 1) in a 1-by-`nd*ml` vector called as `data1`.

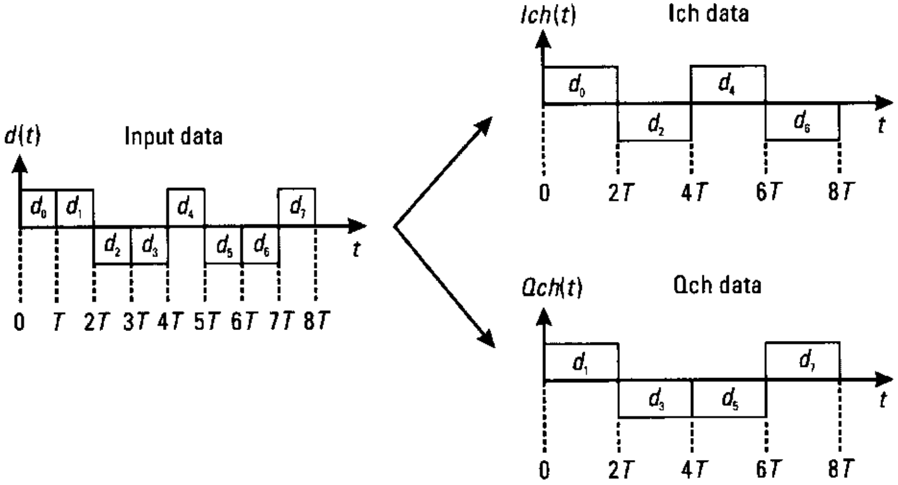


Figure 3.10 Mapping-circuit function for QPSK.

```
data1=rand(1,nd*ml) > 0.5;
```

The transmission data is shown in Figure 3.12(a).

Next, vector data1 is fed into the mapping circuit, where the serial data is converted into parallel data of two channels, Ich and Qch, using predefined mapping. Then, using compoversamp.m, the data in both channels are oversampled and put into an impulse train sequence so that the pulse-shaping filter can be used. Convolution with filter coefficient xh is used to generate for the impulse train sequence. The simulation program works as follows:

```
[ich,qch]=qpskmod(data1,1,nd,ml);
[ich1,qch1]= compoversamp(ich,qch,length(ich),
  IPOINT);
[ich2,qch2]= compconv(ich1,qch1,xh);
```

In this procedure, compoversamp.m is an extension of oversamp.m., and compconv.m becomes an extension of conv.m. The instantaneous waveforms, ich1 and qch1, are shown in Figure 3.12(b) and (c).

The filtered signal is transmitted to air, passes through a radio channel (an equivalent lowpass system), and is received by the receiver.

At the receiver, the received signal is first contaminated by AWGN. The noise function, used comb.m, was described in Section 2.4.1.

We want to determine the relationship between E_b/N_0 and the BER. That means we must vary attn while keeping E_b/N_0 constant. The variable attn is calculated using the same procedure as for BPSK. (See Section 3.2.3.)

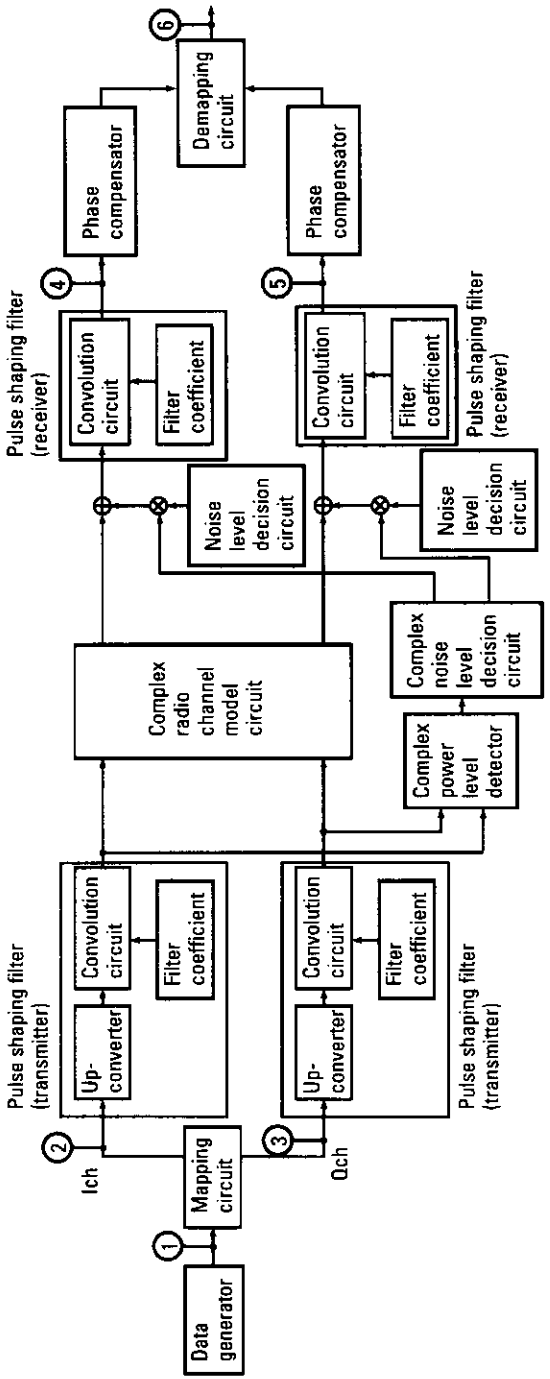

Data generator
1
Mapping circuit
2
Ich
3
Qch
Pulse shaping filter (transmitter)
Up-converter
Convolution circuit
Filter coefficient
Pulse shaping filter (transmitter)
Up-converter
Convolution circuit
Filter coefficient
Complex power level detector
Complex noise level decision circuit
Complex radio channel model circuit
Noise level decision circuit
Noise level decision circuit
Pulse shaping filter (receiver)
Convolution circuit
Filter coefficient
Pulse shaping filter (receiver)
Convolution circuit
Filter coefficient
4
5
Phase compensator
Phase compensator
Demapping circuit
6

Figure 3.11 Model of quadrature modulation scheme used to simulate BER performance.

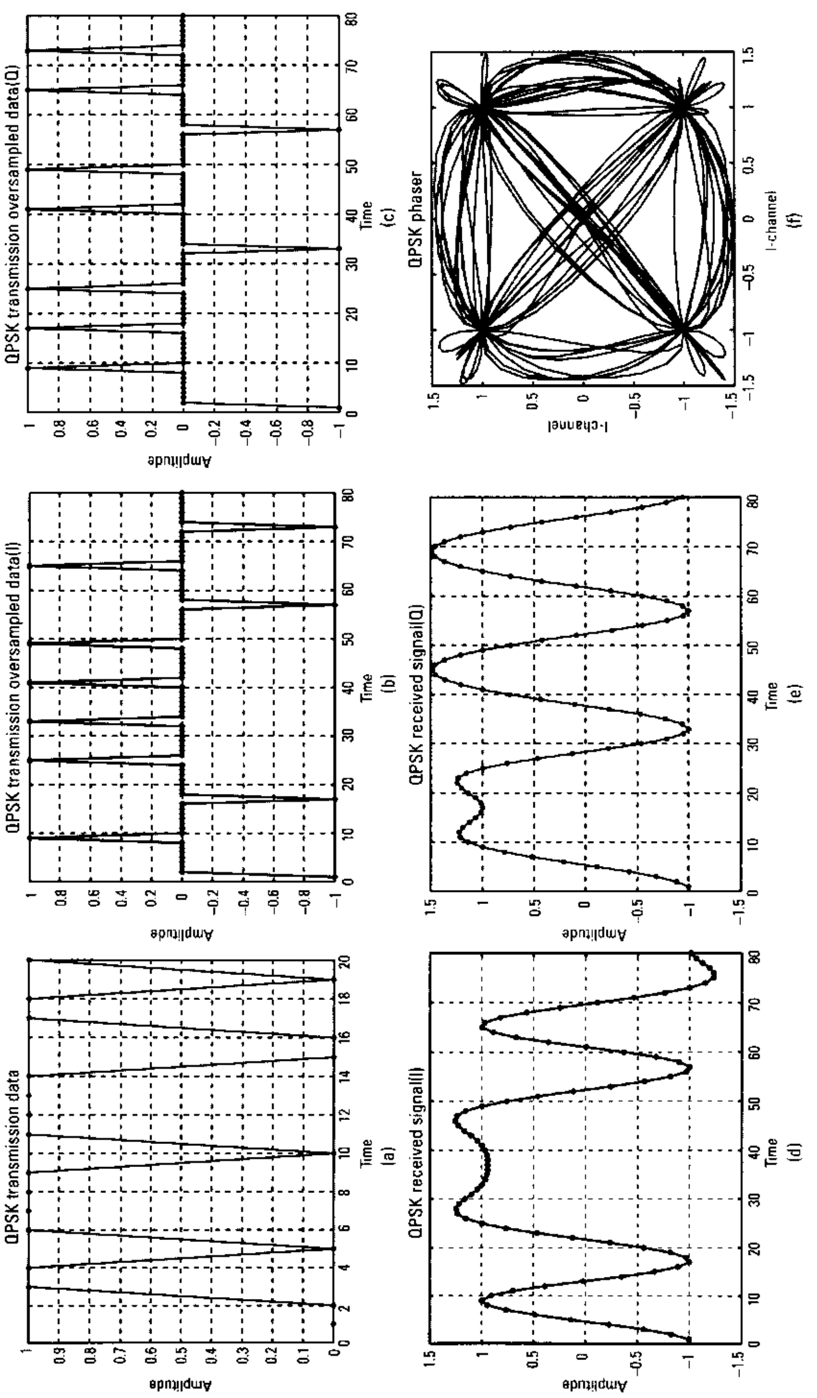


Figure 3.12 Simulated instance waveform: (a) QPSK transmission data, (b) QPSK transmission oversampled data (I), (c) QPSK transmission oversampled data (Q), (d) QPSK received signal (I), (e) QPSK received signal (Q), and (f) QPSK phaser.

Using attn and comb.m, we contaminate the transmitted data with AWGN.

```
[ich3,qch3]= comb(ich2,qch2,length(ich2),attn); ,
```

where comb.m is that described in Section 2.4.1. Then, the received signals, ich3 and qch3, are filtered with a root Nyquist filter, in which the received data is correlated with coefficients xh2.

```
[ich4,qch4]= compconv(ich3,qch3,xh2);
```

The waveforms of ich4 and qch4 are shown in Figure 3.12(d) and (e); the phase is shown in Figure 3.12(f). As mentioned in Section 3.2.3, if we use conv.m, we must recover the time delay due to convolution. In one time, conv.m causes a delay half the length of the filter taps. In this case, the length of the taps is (irfn*IPOINT)/2. Because we used a root Nyquist filter, conv.m is used twice. The delay time is thus irfn*IPOINT samples, and we can define the start point as syncpoint.

```
syncpoint=irfn*IPOINT+1;
```

Then, ich4 and qch4 are resampled at the rate of IPOINT and converted to ich5 and qch5.

```
ich5 = ich4(sampl:IPOINT:length(ich4));
qch5 = qch4(sampl:IPOINT:length(ich4));
```

Then, the resampled data are fed into demodulation function.

```
[demodata]=qpskdemod(ich5,qch5,1,nd,ml);
```

Next, we investigate the number of errors. The procedure was described in Section 2.3. In this simulation, the transmission data is data and the received data is data7.

```
noe2=sum(abs(data1-demodata);
nod2=length(data1);
noe=noe+noe2;
nod=nod+nod2;
```

Finally, the BER is given by

```
>> ber=noe/nod;
```

As shown in Figure 3.13, the simulation results are quite similar to the theoretical values. For the simulation in the Rayleigh fading environment, you can use Program 3.6, which inserted the fading parameters and simulator to Program 3.5.

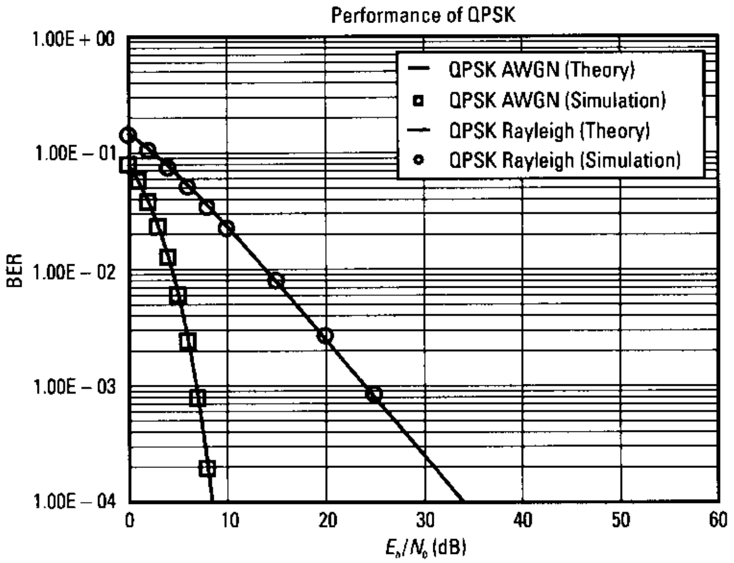


Figure 3.13 Theoretical and simulated BER performance of QPSK scheme under AWGN and one-path flat Rayleigh fading environments.

3.4 OQPSK

3.4.1 Basic Configuration of OQPSK

In the QPSK transmission scheme, the data of Ich and Qch (d_I, d_Q) has four states: A(1, 1), B(1, −1), C(−1, −1), and D(−1, +1). Because it can alternate between all candidates, the phase transmission is that shown in Figure 3.12(f). As shown in Figure 3.12(f), the phase transition sometimes passes the origin, meaning that the phase changes 180°, and the spectrum will be spread in comparison with that of other phase transitions. One way to reduce the number of origin crossings is to use OQPSK.

The configuration of OQPSK is almost the same as those of QPSK. The only different configuration is that for the mapping circuit. As shown in Figure 3.14, the data signal of the Qch signal is half a symbol delayed from that of the Ich signal. The configurations for the transmitter and receiver are the same as follows [1–3]. Therefore, the theoretical BER is the same as that for QPSK; it is shown in (3.27) and (3.28).

We can represent the OQPSK signal as those for [1–3]

$$s(t) = \frac{1}{\sqrt{2}} d_I(t)\cos(2\pi f_c t) + \frac{1}{\sqrt{2}} d_Q(t - T_b/2)\sin(2\pi f_c (t - T_b/2)) \tag{3.29}$$

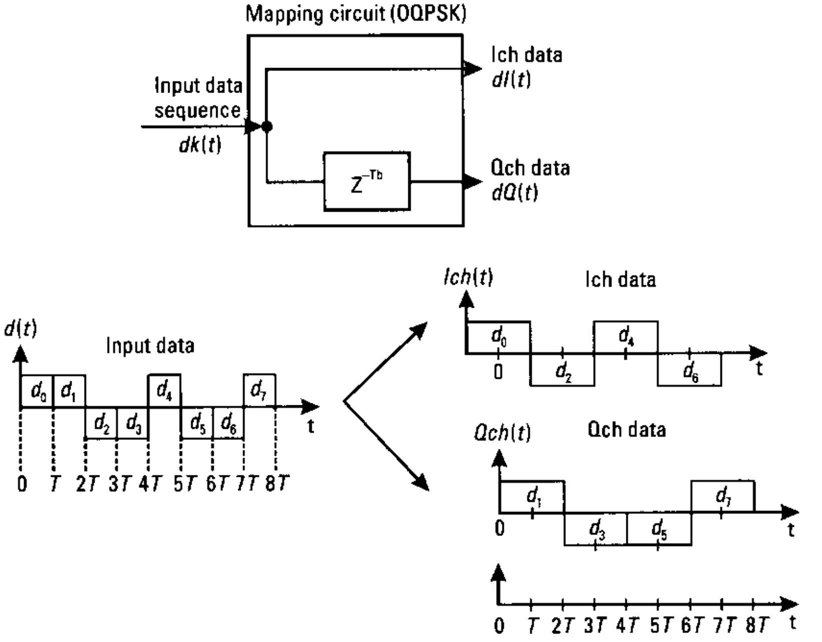


Figure 3.14 Mapping-circuit function for OQPSK.

3.4.2 Computer Simulation

The model we used to simulate the BER performance of OQPSK is based on that depicted in Figure 3.11 and shown as Program 3.11. As mentioned, the only difference is in the mapping circuit. The mapping-circuit commands for QPSK are listed as follows:

```
[ich, qch]=qpskmod(data1,1,nd,ml);
[ich1, qch1]=compoversamp(ich,qch,length(ich),IPOINT);
[ich2,qch2]=compconv(ich1,qch1,xh);
```

Those for OQPSK are listed as follows:

```
[ich,qch]=qpskmod(data1,1,nd,ml);
[ich1,qch1]=compoversamp(ich,qch,
  length(ich),IPOINT);
ich21=[ich1 zeros(1,IPOINT/2)];
qch21=[zeros(1,IPOINT/2) qch1];
[ich2, qch2]=compconv(ich21,qch21,xh); .
```

As shown above, `qch21` delays `IPOINT/2` from `ich21`. The delay must be adjusted for after pulse-shaping at the receiver to demodulate the received signal. At the point for QPSK, we use

```
    syncpoint=irfn*IPOINT+1;
    ich5=ich4(syncpoint:IPOINT:length(ich4));
    qch5=qch4(syncpoint:IPOINT:length(qch4));.
```

To compensate for the delay with OQPSK, we use

```
    syncpoint =irfn*IPOINT+1;
    ich5=ich4(syncpoint:IPOINT: length(ich4));
    qch5=qch4(syncpoint+IPOINT/2:IPOINT: length(qch4));.
```

By using the commands in Program 3.11, we obtain the OQPSK signal. Figure 3.15(a)–(e) shows several instantaneous signs of `data1`, `ich21`, `qch21`, `ich4`, and `qch4` obtained using Program 3.11. The locus of the phase with `ich4` and `qch4` without noise and fading is shown in Figure 3.15(f), and the BER performance is shown in Figure 3.16. Figure 3.15(b) and (c) shows that this simulation generates offset QPSK, and Figure 3.15(f) shows that OQPSK never passes through the origin. Therefore, spurious frequencies from other channels are reduced. Although OQPSK is good for reducing big phase changes, the amplitude is not fixed and fluctuates [see Figures 3.12(f) and 3.15(f)]. One way to stabilize the amplitude is to use MSK, which is described in the Section 3.5.

3.5 MSK

3.5.1 Basic Configuration of MSK Transmission Scheme

In this modulation scheme, the amplitude of OQPSK is weighted to eliminate the fluctuation in amplitude. The weighting functions are normally $\cos(\pi t/2T_b)$ for the Ich data and $\sin(\pi t/2T_b)$ for the Qch data. The only change is again in the configuration of the mapping circuit (Figure 3.17). We can represent the MSK signal as those for [1–3].

$$s(t) = d_I(t)\cos(\pi t/2T_b)\cos(2\pi f_c t) + d_Q(t - T_b/2)\sin(\pi(t - T_b/2)/2T_b)$$

$$\sin(2\pi f_c(t - T_b/2)) = \cos(2\pi f_c + b(t)\pi t/2T + \theta(t)) \tag{3.30}$$

where $b(t) = -d_I(t)d_Q(t)$ and

$$\theta(t) = \begin{cases} 0 & d_I(t) = 1 \\ \pi & d_Q(t) = -1 \end{cases} \tag{3.31}$$

The theoretical bit error probability with AWGN and one-path flat Rayleigh fading is the same as that of QPSK (3.27) and (3.28).

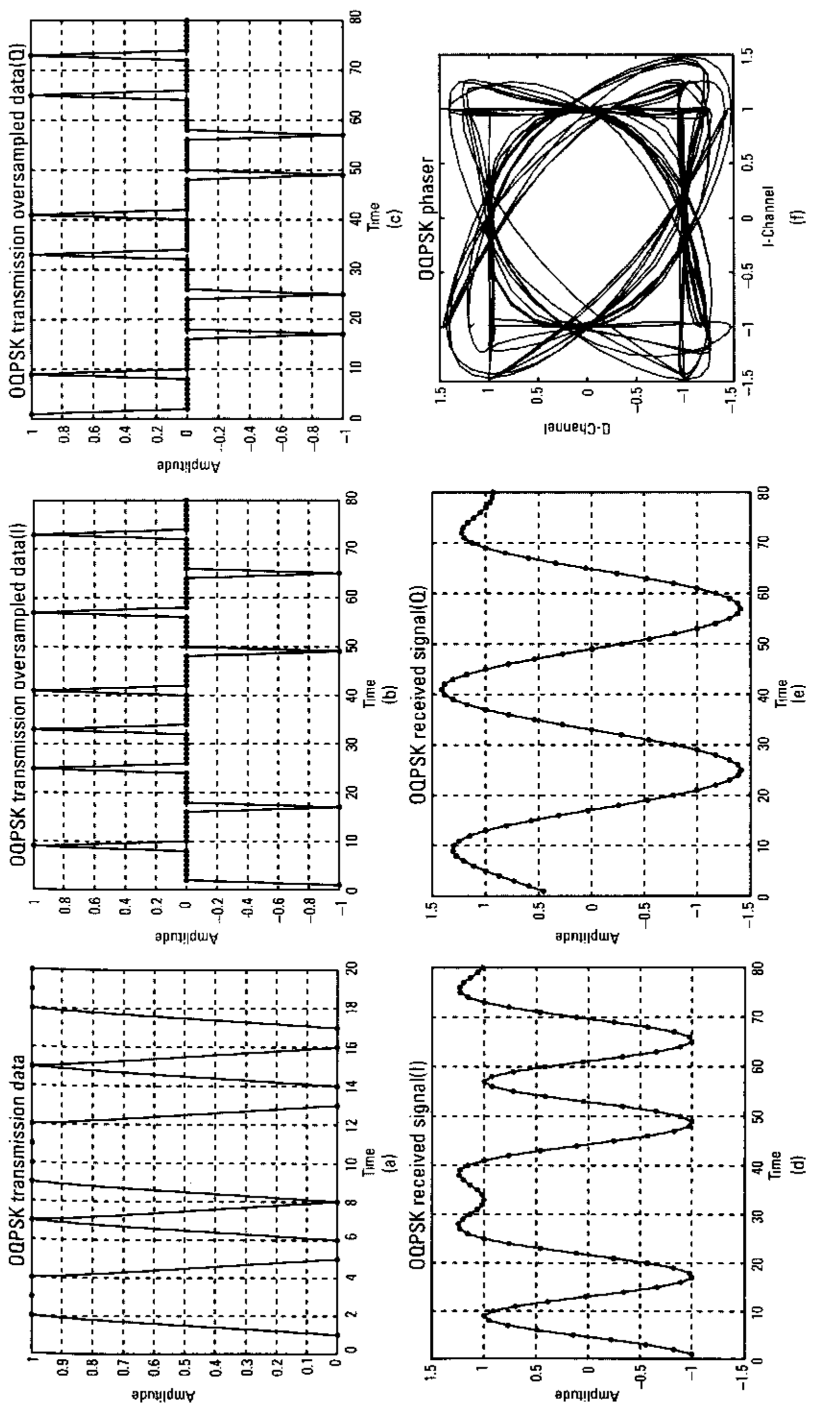


Figure 3.15 Simulated instance OQPSK waveform: (a) OQPSK transmission data, (b) OQPSK transmission oversampled data (I), (c) OQPSK transmission oversampled data (Q), (d) OQPSK received signal (I), (e) OQPSK received signal (Q), and (f) OQPSK phaser.

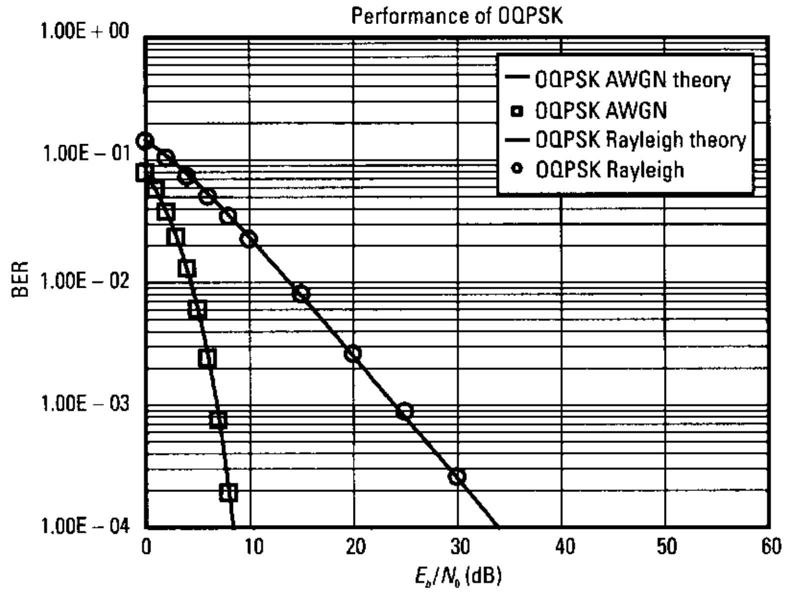


Figure 3.16 Theoretical and simulated BER performance of OQPSK scheme under AWGN and one-path flat Rayleigh fading environments.

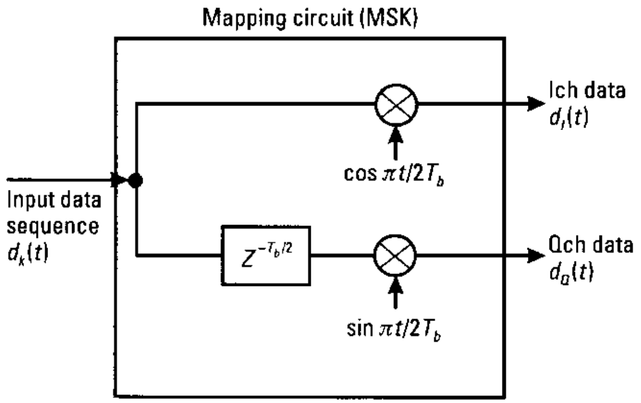


Figure 3.17 Mapping-circuit function for MSK1.

3.5.2 Computer Simulation

The model we used to simulate the BER performance of MSK is again based on that shown in Figure 3.11 and the simulation program is in Programs 3.13 and 3.14. The only difference is again in the configuration of the mapping circuit. The mapping-circuit commands for MSK are listed as follows:

```
[ich,qch]=qpskmod(data1,1,length(data2)/2,2);
smooth1=cos(pi/2*[-1:1./4*[0:IPOINT-1]);

for ii=1:length(ich)
ich2((ii-1)*IPOINT+1:ii*IPOINT)=(-1)^(ii-1)*smooth1.*ich(ii);
qch2((ii-1)*IPOINT+1:ii*IPOINT)=(-1)^(ii-1)*smooth1.*qch(ii);
end

ich21=[ich2 zeros(1,IPOINT/2)];
qch21=[zeros(1,IPOINT/2) qch2]; .
```

As shown above, `qch21` delays `IPOINT/2` from `ich21`. This delay must be adjusted for after pulse-shaping at the receiver to demodulate the received signal. At the part with OQPSK we use

```
syncpoint =irfn*IPOINT+1;
ich5=ich4(syncpoint:IPOINT:length(ich4));
qch5=qch4(syncpoint+IPOINT/2:IPOINT: length(qch4));.
```

With MSK, since we do not use any filters, we use

```
syncpoint=1;

ich5 = ich3(syncpoint+IPOINT/2:IPOINT:length(ich3));
qch5 = qch3(syncpoint+IPOINT:IPOINT:length(ich2)+IPOINT/2);


ich5(2:2:length(ich5))=-1*ich5(2:2:length(ich5));
qch5(2:2:length(ich5))=-1*qch5(2:2:length(ich5));.
```

By using these commands in Programs 3.13 and 3.14, we obtain the MSK signal. Figure 3.18(a)–(e) shows several instantaneous signals of `data1`, `ich`, `qch`, `ich5`, and `qch5` obtained by using Program 3.11. The phase by `ich5` and `qch5` without noise and fading is shown in Figure 3.18(f), and the BER performance is shown in Figure 3.19. Figure 3.18(f) shows that this simulation generates MSK and that MSK never passes through the origin. Therefore, spurious frequencies to the other channels are reduced.

In the above discussion, we approached the configuration of MSK from that of OQPSK. Another approach starts from the FSK-based transmission scheme. An MSK signal is regarded as an FSK signal with a continuous phase transition, because the phase locus is smoothly and continuously distributed, as shown in Figure 3.18(f). The configuration of the mapping circuit in this case is shown in Figure 3.20; the FSK signal with its continuous phase transition is described as

$$s(t) = A\cos\left(2\pi f_c t + \theta(t)\right) \tag{3.32}$$

$$\theta(t) = 2\pi\Delta f \int_{-\infty}^{\tau} x(\tau)d\tau \tag{3.33}$$

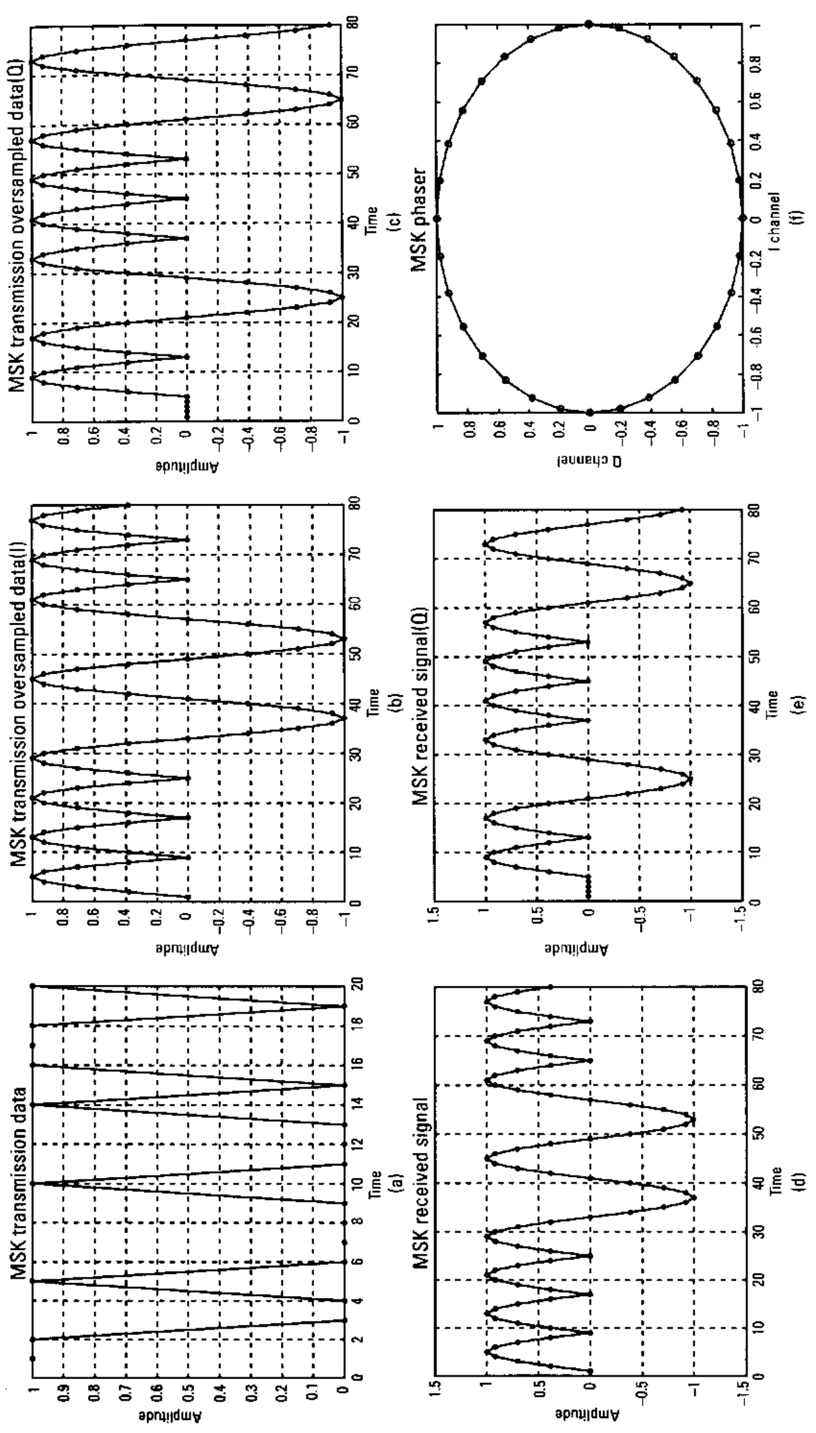


Figure 3.18 Simulated instantaneous MSK waveform (simulation 1): (a) MSK transmission data, (b) MSK transmission oversampled data (I), (c) MSK transmission oversampled data (Q), (d) MSK received signal (I), (e) MSK received signal (Q), and (f) MSK phaser.

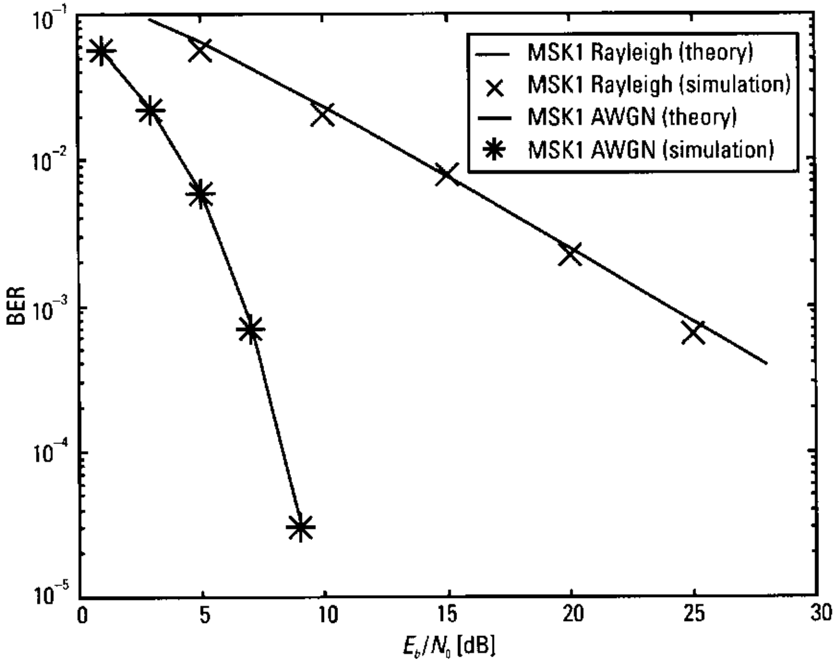


Figure 3.19 Theoretical and simulated BER performance of MSK1 scheme under AWGN and one-path Rayleigh fading environments (simulation 1).

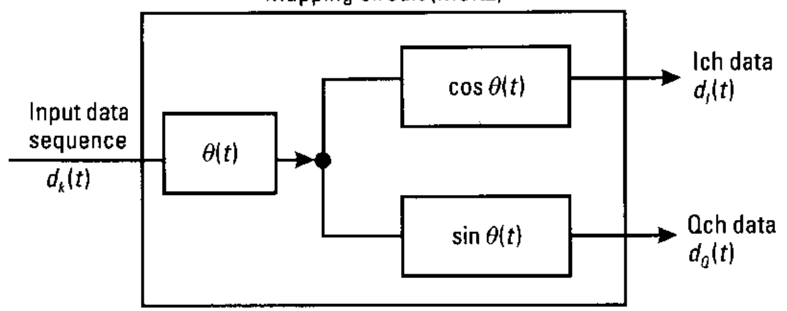


Figure 3.20 Mapping-circuit function for MSK2 (simulation 2).

$$x(t) = \sum_k d_k p(t - kT_b) \tag{3.34}$$

where $p(t)$ is a rectangular pulse:

$$p(t) = \begin{cases} 1 & 0 < t \le T_b \\ 0 & \text{otherwise} \end{cases} \tag{3.35}$$

The integer $d_k : k$ is a data sequence with 1 or -1 data. The power spectrum density of the FSK signal shown in (3.30) is [2]

$$\frac{S_{\mathrm{FSK}}(f)}{T_b} = \begin{cases} 4\left\{\dfrac{m}{\pi\left(m^2 - 4X^2\right)}\right]^2 \dfrac{\{\cos(m\pi) - \cos(2\pi X)\}^2}{1 - 2\cos(m\pi)\cos(2\pi X) + \cos^2(m\pi)} & |\cos(m\pi)| < 1 \\ \dfrac{1}{4T_b}\{\delta(2X + m) + \delta(2X - m)\} + 2\left\{\dfrac{m}{\pi\left(m^2 - 4X^2\right)}\right\}^2 \{1 - \cos(m\pi)\cos(2\pi X)\} & |\cos(m\pi)| = 1, m \ne 0 \end{cases} \tag{3.36}$$

where $m = 2\Delta f T_b$ is the modulation index and $X(=(f - f_c)\,T_b)$ is the normalized frequency. The power spectrum density of the signal MSK is

$$\frac{S_{MSK}(f)}{T_b} = \frac{16}{\pi^2}\left(\frac{\cos\left(2\pi f T_b^2\right)}{1 - 16 f^2 T_b^{\ 2}}\right) \tag{3.37}$$

The values in (3.36) and (3.37) become the same if we substitute for $m = 0.5$ (3.36). We can generate the MSK signal by using (3.32) to (3.34) and $m = 0.5$. The procedure is as follows.

1. Using input data d_k, we calculate $\theta(t)$, which is given by (3.33), by using the result of

$$\begin{aligned} \theta(kT_b) - \theta((k-1)T_b) &= 2\pi\Delta f \int_{(k-1)T_b}^{kT_b} x(\tau)d\tau \\ \theta(kT_b) &= \theta((k-1)T_b) + 2\pi\Delta f \int_{(k-1)T_b}^{kT_b} x(\tau)d\tau \\ &= \theta((k-1)T_b) + 2\pi\Delta f T_b d_k \qquad (\because (3.34)) \\ &= \theta((k-1)T_b) + \frac{\pi}{2} d_k \qquad \left(m = 2\Delta f T_b = 0.5\right) \end{aligned} \tag{3.38}$$

because $\theta(kT_b)$is continuously connected with $\theta((k-1)T_b)$, in the area $(k-1)T < t \le kT$,

$$\theta(t) = \theta((k-1)T_b) + \frac{\pi}{2} d_k \frac{t - (k-1)T_b}{T_b} \tag{3.39}$$

2. The calculated $\theta(t)$ is input to (3.30) to obtain the MSK signal.

When differential encoding and coherent detection are used [1–3],

$$\mathrm{BER}_{\mathrm{MSK\_AWGN}} = \mathrm{erfc}\left(\sqrt{E_b/N_0}\right) \tag{3.40}$$

The model we used to simulate the BER performance is again based on that shown in Figure 3.11 and the simulation program is described in Programs 3.15 and 3.16. The only difference is again in the configuration of the mapping circuit, and the mapping-circuit commands for MSK based on FSK are shown as follows:

```
data2=oversamp2(data11,length(data11),IPOINT);

th=zeros(1,length(data2)+1);
ich2=zeros(1,length(data2)+1);
qch2=zeros(1,length(data2)+1);

for ii=2:length(data2)+1
th(1,ii)=th(1,ii-1)+pi/2*data2(1,ii-1)./IPOINT;
end

ich2=cos(th);
qch2=sin(th);.
```

At the demodulator, we must recover the transmission signal from the Ich and Qch data. First, we resample the `ich3` and `qch3` data every `IPOINT`. The method is the same as with OQPSK:

```
syncpoint = 1;
ich5=ich3(syncpoint:IPOINT:length(ich4));
qch5=qch3(syncpoint+IPOINT/2:IPOINT:length(qch4)); .
```

Then, `ich5` is differentially multiplied by `qch5`.

```
demoddata2(1,1)=-1;   % Initialized data

for k=3:2:nd*ml+1
demodata3(1,k)=ich5(1,k)*qch5(1,k-1)*cos(pi*(k))>0;
end

for n=2:2:nd*ml+1
demodata2(1,n)=ich5(1,n-1)*qch5(1,n)*cos(pi*(k))>0;
end
```

Finally, the clock is multiplied by `demodata2`, and the transmission signal is regenerated.

```
[demodata]=demodata2(1,2:nd*ml+1);
```

By using these commands in Programs 3.15 and 3.16, we obtain the MSK signal. Figure 3.21 shows the transmission signal [Figure 3.21(a)], and the transition of $\theta(t)$ [Figure 3.21(b)], and transmission signals of Ich and Qch [Figure 3.21(c)]. The BER performance of MSK is shown in Figure 3.22. Figure 3.21(b) shows that the phase changes by at least $\pi/2$ for each symbol. Figure 3.22 shows that this simulation program expresses the MSK scheme well. Although MSK has many good characteristics, as shown in Figure 3.21(b), the maximum transition is limited to $\pi/2$ in the phase transition, so the fluctuation is not smooth. One way to smooth out the fluctuations is *Gaussian MSK* (GMSK), which is described in Section 3.6.

3.6 GMSK

3.6.1 Basic Configuration of GMSK

GMSK has come into wide usage because it was adopted for use in the GSM mobile phone system. GMSK is an expanded version of MSK. Like MSK, it keeps the amplitude constant, as shown in Figure 3.18(f). Unlike MSK, it keeps locus of the phase transition smooth at the kT_b: k integer point.

To generate the GMSK signal, we can use the same configurations as for MSK, except for the mapping circuit. The configuration of the mapping circuit is shown in Figure 3.23. The input data is filtered with a Gaussian filter, and its shape is stabilized. The filtered signal is fed into an FSK-based MSK transmitter and receiver. The frequency response is

$$G(f)=\exp\left\{-\frac{\ln 2}{2}\left(\frac{f}{B_{3\mathrm{dB}}}\right)^2\right\} \tag{3.41}$$

The time impulse response is

$$g(t)=\sqrt{\frac{2\pi}{\ln 2}}B_{3\mathrm{dB}}\exp\left\{-\frac{2}{\ln 2}\left(\pi B_{3\mathrm{dB}}t\right)^2\right\} \tag{3.42}$$

where $B_{3\ \mathrm{dB}}$ is a 3-dB bandwidth and filter $g(t)$ has an infinite time coefficient. In the normal case, we investigate the characteristics of a Gaussian filter by performing a convolution between $g(t)$ and rectangular pulse $g_T(t)$ with width $2a$. The convolution value is

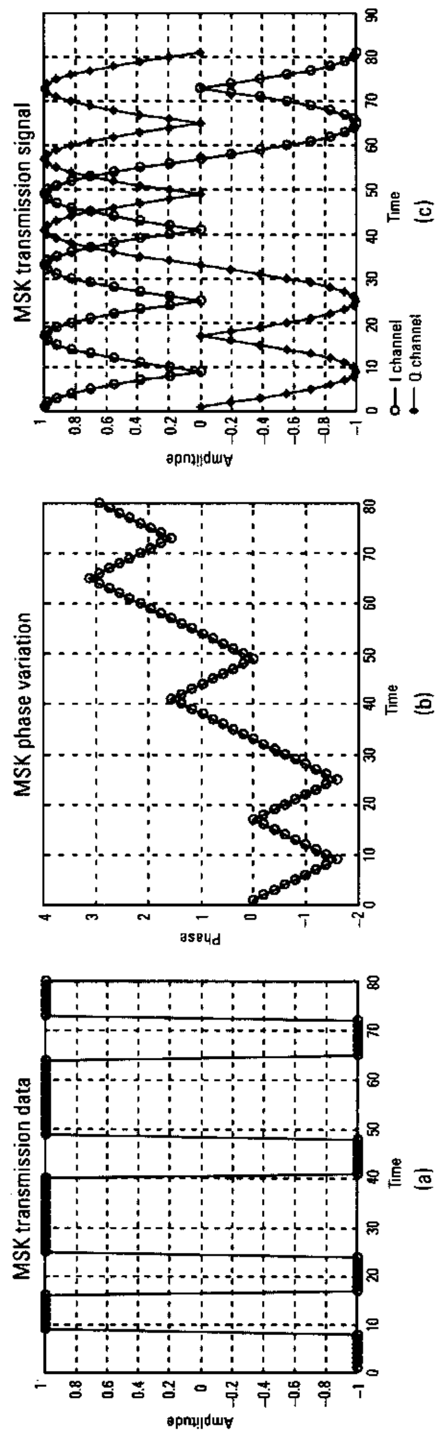


Figure 3.21 Simulated instantaneous MSK waveform (simulation 2): (a) transmission data, (b) MSK phase variation, and (c) MSK transmission signal.

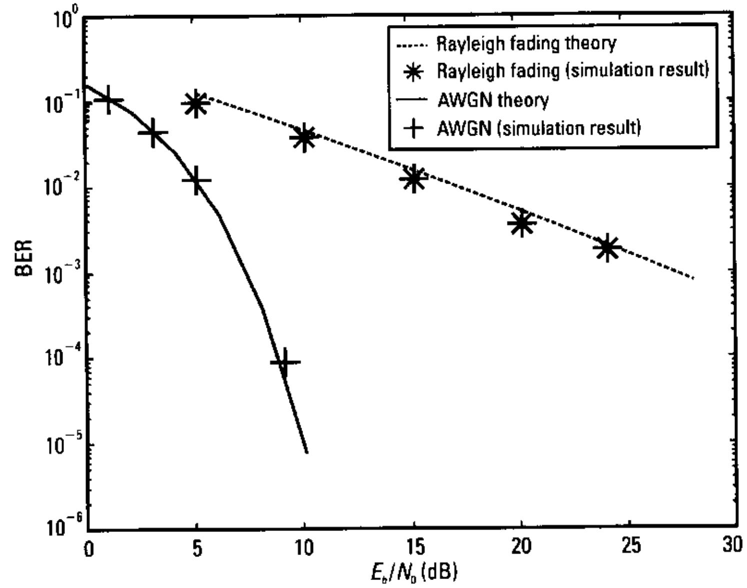


Figure 3.22 Theoretical and simulated BER performance of MSK2 scheme under AWGN and one-path flat Rayleigh fading environments (simulation 2).

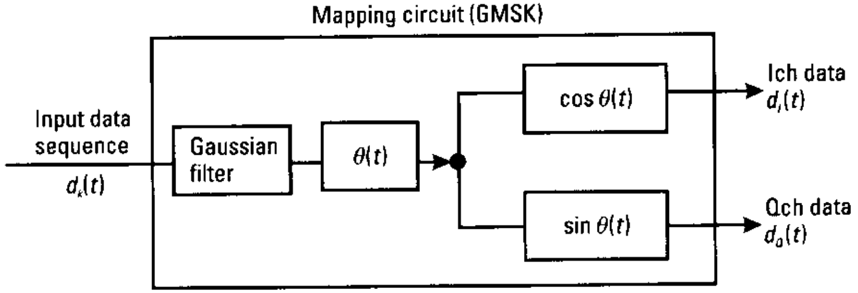


Figure 3.23 Mapping-circuit function for GMSK.

$$y(t) = g(t) \otimes g_T(t) = \sqrt{\frac{2\pi}{\ln 2}} B_{3\,\text{dB}} \int_{t-\alpha}^{t+\alpha} \exp\left\{ -\frac{2}{\ln 2} \left(\pi B_{3\,\text{dB}} \tau \right)^2 \right\} d\tau$$
$$= \frac{1}{2} \left[\text{erf}\left\{ -\sqrt{\frac{2}{\ln 2}} \pi B_{3\,\text{dB}} (t-a) \right\} + \text{erf}\left\{ \sqrt{\frac{2}{\ln 2}} \pi B_{3\,\text{dB}} (t+a) \right\} \right] \tag{3.43}$$

Figure 3.24 illustrates how the shapes of the Gaussian filter are changed by changing $B_{3\text{ dB}}$. The BER performance of the GMSK transmission has been calculated:

$$\text{BER}_{\text{GMSK\_AWGN}}\left(E_b/N_0\right) \cong \text{erfc}\left(\sqrt{\beta \cdot E_b/N_0}\right) \tag{3.44}$$

3.6.2 Computer Simulation

The model we used to simulate the BER performance of GMSK is based on Figure 3.11, and except for the Gaussian filter, all the programs are those as for MSK. The main simulation programs are Programs 3.18 and 3.19. The changes for GMSK are marked with "`### New for GMSK ###`."

By using the revised program, we obtain the GMSK signal. Figure 3.25 (a)–(d) shows several instantaneous signals of `data2`, `data3`, `th`, `ich2`, and `qch2` obtained by using Program 3.14. The locus of the phase without noise and fading is shown in Figure 3.25(e), and the BER performance is shown in

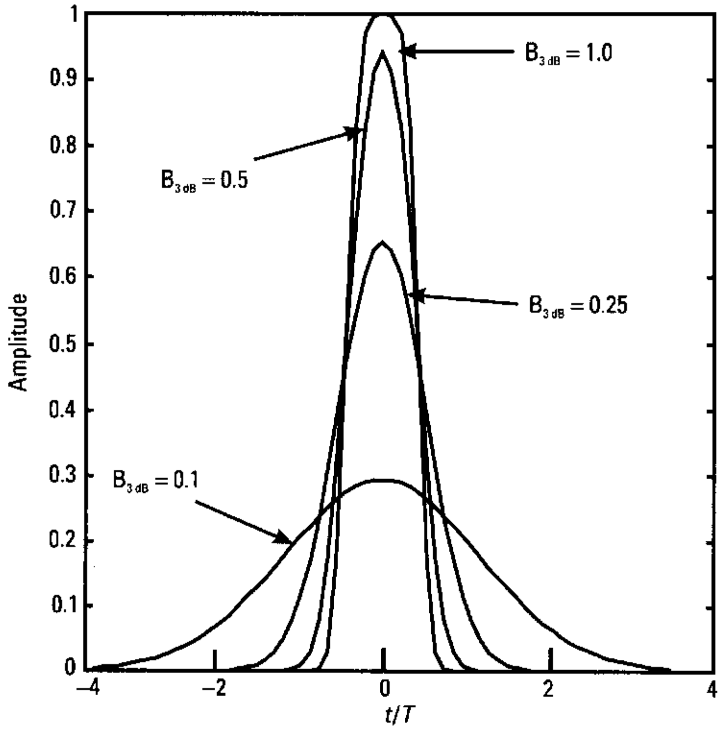


Figure 3.24 Change in Gaussian filter with change in $B_{3\text{dB}}$.

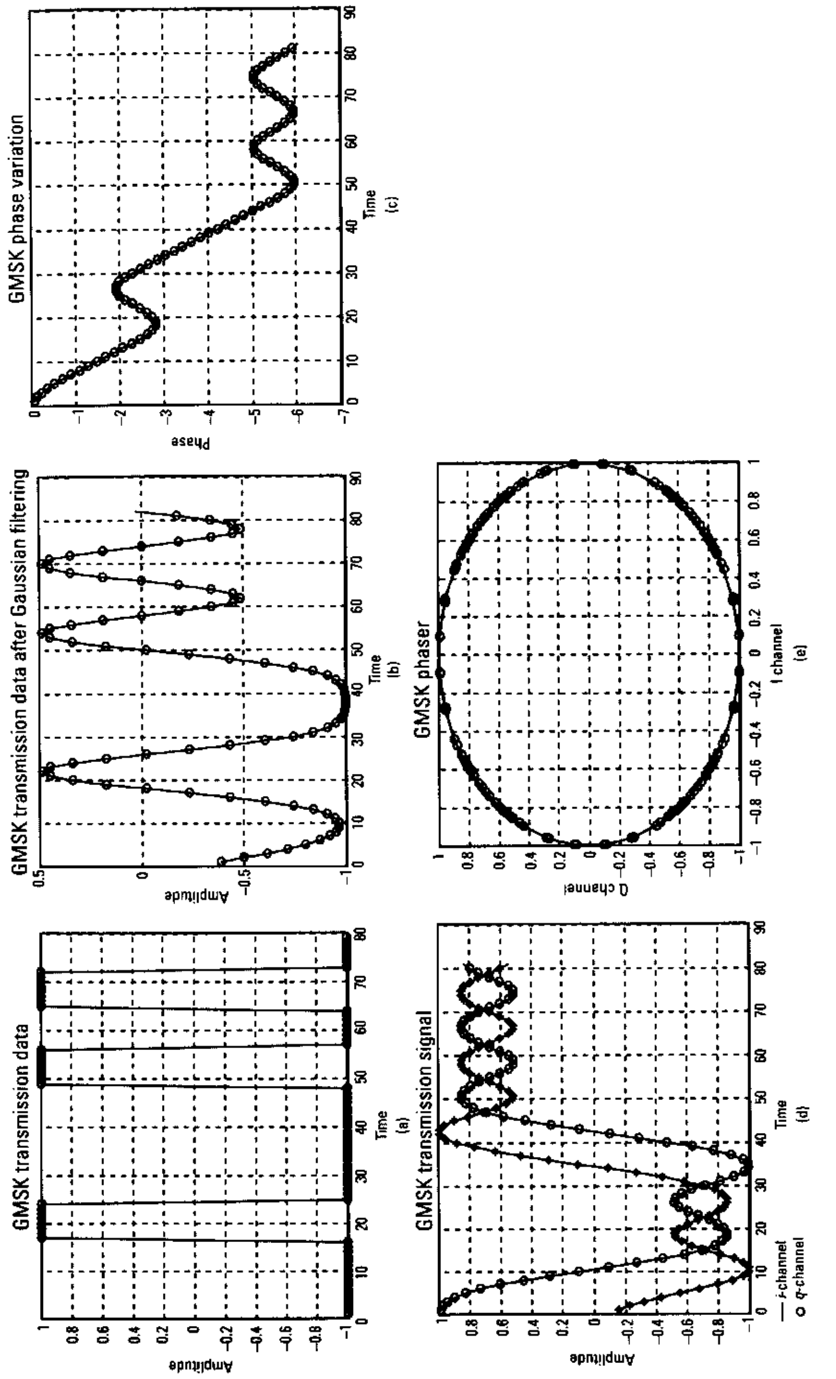


Figure 3.25 Simulated instantaneous GMSK waveform: (a) GMSK transmission data, (b) GMSK transmission data after Gaussian filtering, (c) GMSK phase variation, (d) GMSK transmission signal, and (e) GMSK phaser.

Figure 3.26. Figure 3.26 shows that this simulation generates GMSK, and Figure 3.25(e) shows that GMSK never passes through the origin, proving that the amplitude remains constant. Figure 3.25(c) demonstrates that the phase transition is quite smooth. While the system has high robustness for radio channels to increase the transmission rate, we must look for other directions. Section 3.7 introduces a scheme that increases the transmission rate while keeping high-frequency utilization efficiency.

3.7 QAM

3.7.1 Basic Configuration of QAM

In the QPSK-based modulation schemes, two bits $(a_{2n}, a_{2n+1}) : n = 1, 2\ldots$ can be transmitted during symbol time T_b. The waveform is

$$s(t) = d_I(t)\cos(2\pi f_c t) + d_Q(t)\sin(2\pi f_c t) \qquad (3.45)$$

where $d_I(t)$ and $d_Q(t)$ are the transmission amplitudes of `Ich` and `Qch`, respectively.

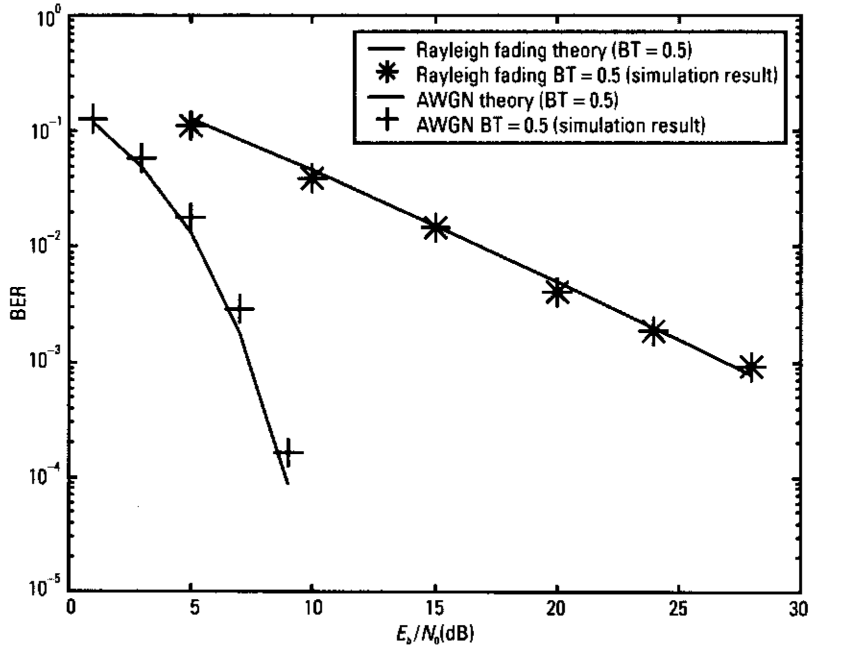


Figure 3.26 Theoretical and simulation performance of GMSK scheme under AWGN and one-path flat Rayleigh fading environments.

In the QAM transmission scheme, the amplitudes of `Ich` and `Qch` are determined using more bits. Figure 3.27(a)–(d) shows the constellations of QAM, 16-QAM, 64-QAM, and 256-QAM respectively. Here we focus on 2^{2n}QAM. In this scheme, we transmit 2^{2n} bits ($a_{2m+1}, \ldots, a_{2m+2n}$) during each symbol time. A key factor when doing this is how to assign the transmission bit to the amplitude. In this section, we introduce the Gray encoding method for 2^{2n}QAM ($n = 1, 2, 3, 4$) as an example. There are three steps in this method.

1. Categorize ($a_{2m+1}, \ldots, a_{2m+2n}$) into two groups: $g_1(a_{2m+1}, \ldots, a_{2m+n})$ and $g_2(a_{2m+n+1}, \ldots, a_{2m+2n})$.
2. For both groups, we perform two calculations

$$a_I(k) = \sum_{j=1}^{n} a_{2m+j} \times 2^{n-j} \qquad (3.46)$$

$$a_Q(k) = \sum_{j=1}^{n} a_{2m+n+j} \times 2^{n-j} \qquad (3.47)$$

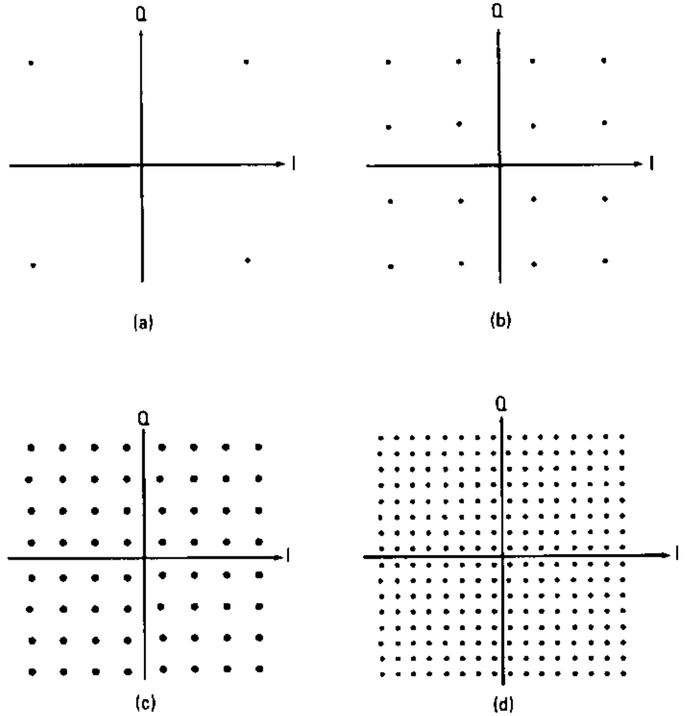


Figure 3.27 Constellations of (a) QAM, (b) 16-QAM, (c) 64-QAM, and (d) 256-QAM.

3. By using conversion map

$$C = \begin{pmatrix} -1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ -3 & -1 & 3 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ -7 & -5 & -1 & -3 & 7 & 5 & 1 & 3 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ -15 & -13 & -9 & -11 & -1 & -3 & -7 & -5 & 15 & 13 & 9 & 11 & 1 & 3 & 7 & 5 \end{pmatrix} \qquad (3.48)$$

we can determine the amplitudes of both the in-phase and quadrature-phase channels for 2^{2n}QAM, $n = 1, 2, 3, 4$.

$$d_I(k) = C\left(n, a_I(k) + 1\right) / \sum_{j=1}^{n} 2^{j-1} \qquad (3.49)$$

$$d_Q(k) = C\left(n, a_Q(k) + 1\right) / \sum_{j=1}^{n} 2^{j-1} \qquad (3.50)$$

By using (3.45), (3.49), and (3.50), we can generate a Gray-encoded 2^{2n}QAM signal. By using the reverse procedure, we can decode the signal and regenerate transmission data. The theoretical BER values for 2^{2n}QAM, $n = 1, 2, 3, 4$, under AWGN and one-path Rayleigh fading have been reported in [1–3].

$$\mathrm{BER}_{\mathrm{16QAM\_AWGN}}\left(E_b/N_0\right) = \frac{3}{8}\mathrm{erfc}\left(\sqrt{\frac{2}{5}E_b/N_0}\right) - \frac{9}{64}\mathrm{erfc}^2\left(\sqrt{\frac{2}{5}E_b/N_0}\right) \qquad (3.51)$$

$$\mathrm{BER}_{\mathrm{64QAM\_AWGN}}\left(E_b/N_0\right) = \frac{7}{24}\mathrm{erfc}\left(\sqrt{\frac{1}{7}E_b/N_0}\right) - \frac{49}{384}\mathrm{erfc}^2\left(\sqrt{\frac{1}{7}E_b/N_0}\right) \qquad (3.52)$$

$$\mathrm{BER}_{\mathrm{256QAM\_AWGN}}\left(E_b/N_0\right) = \frac{15}{64}\mathrm{erfc}\left(\sqrt{\frac{4}{85}E_b/N_0}\right) - \frac{225}{2{,}048}\mathrm{erfc}^2\left(\sqrt{\frac{4}{85}E_b/N_0}\right) \qquad (3.53)$$

$$\mathrm{BER}_{\mathrm{16QAM\_FADING}}\left(E_b/N_0\right) = \frac{3}{8}\left[1 - \frac{1}{\sqrt{1 + 5/(2E_b/N_0)}}\right] \qquad (3.54)$$

$$\mathrm{BER}_{64\mathrm{QAM\_FADING}}\left(E_b/N_0\right)=\frac{7}{24}\left[1-\frac{1}{\sqrt{1+7/\left(E_b/N_0\right)}}\right] \quad (3.55)$$

$$\mathrm{BER}_{256\mathrm{QAM\_FADING}}\left(E_b/N_0\right)=\frac{15}{64}\left[1-\frac{1}{\sqrt{1+85/\left(4E_b/N_0\right)}}\right] \quad (3.56)$$

In Rayleigh fading, we assume that the frequency rotation is compensated for.

3.7.2 Computer Simulation

We calculated the BER of 2^{2n}QAM using computer simulation. This section focuses on the case of $n = 2$ (16-QAM), as a representative example. The model we used is shown in Figure 3.11. The programs are Programs 3.21–3.24.

The only difference in the model from QPSK is in the mapping and demapping methods. The mapping circuit for QPSK is as follows:

```
[ich qch]=qpskmod(data1,1,nd,ml);
[ich1,qch1]=compoversamp(ich,qch,length(ich),IPOINT);
[ich2,qch2]=compconv(ich1,qch1,xh);.
```

These commands are used in the simulation of QPSK. The mapping circuit for QAM is as follows:

```
[ich qch]=qammod(data1,1,nd,ml);
[ich1,qch1]=compoversamp(ich,qch,length(ich),IPOINT);
[ich2,qch2]=compconv(ich1,qch1,xh);     .
```

The simulation differed from those for QPSK and QAM in mentioning the demodulation procedure. For QPSK, we used

```
[demodata]=qpskdemod(ich5,qch5,1,nd,ml); .
```

For QAM we used

```
[demodata]=qamdemod(ich5,qch5,1,nd,ml); .
```

By changing these commands in Programs 3.21 and 3.22, we obtained a 16-QAM signal. Figure 3.28(a)–(e) shows several instantaneous signals of `data1`, `ich1`, `qch1`, `ich5`, and `qch5`, respectively. The locus of the phase for `ich5` and `qch5` with no noise or fading is shown in Figure 3.28(f). As shown in Figure 3.29, the simulated BER performance is quite similar to the theoretical BER values.

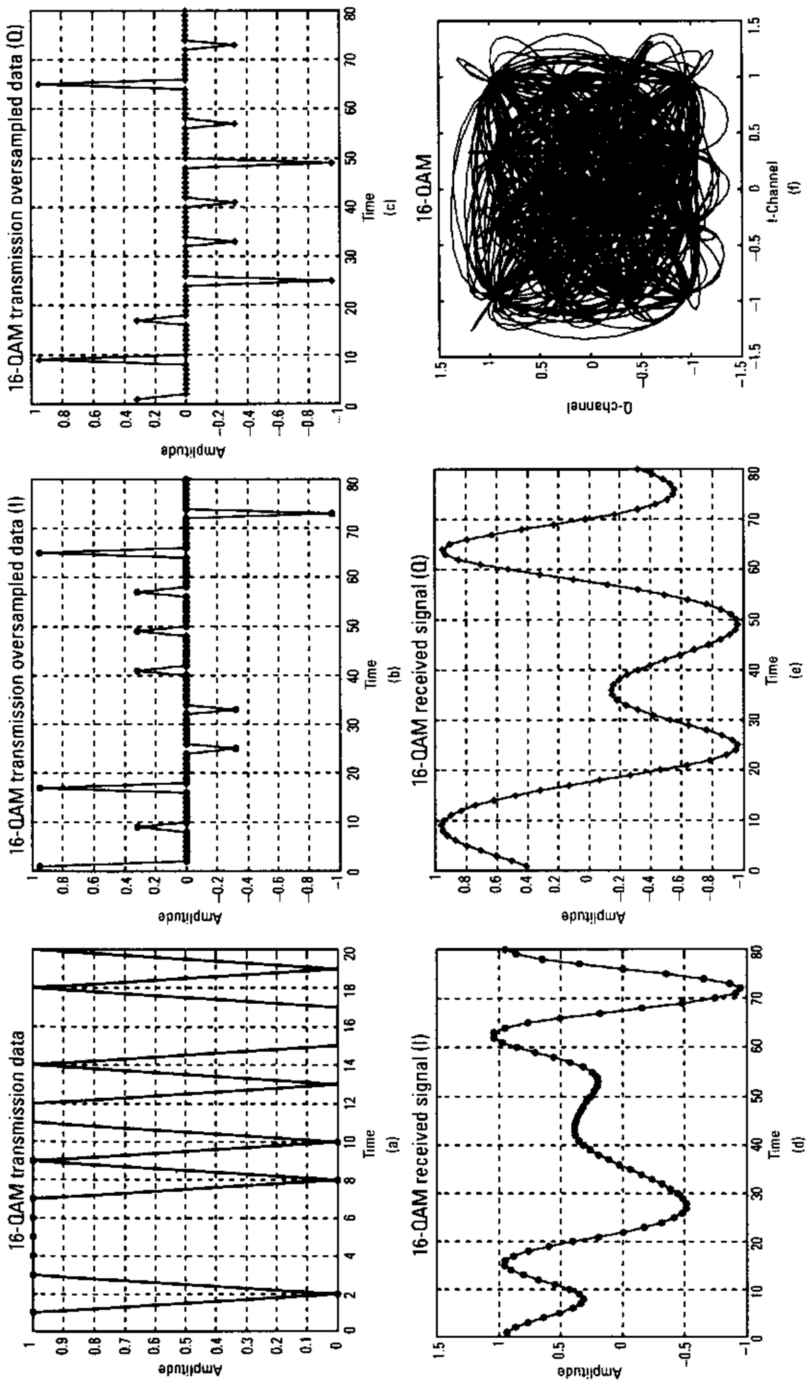


Figure 3.28 Simulated instantaneous 16-QAM waveform: (a) 16-QAM transmission data, (b) 16-QAM transmission oversampled data (I), (c) 16-QAM transmission oversampled data (Q), (d) 16-QAM received signal (I), (e) 16-QAM received signal (Q), and (f) 16-QAM.

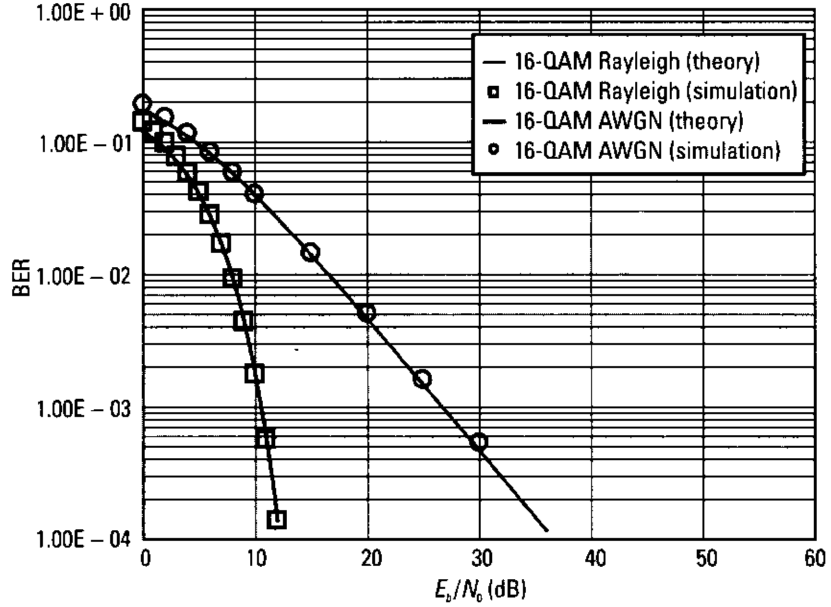


Figure 3.29 Theoretical and simulated BER performance of 16-QAM scheme with AWGN and one-path flat Rayleigh fading.

3.8 Conclusion

This section described six PSK-based digital transmission schemes: BPSK, QPSK, OQPSK, MSK, GMSK, and QAM. We described the basic configurations and the characteristics of their transmitters and receivers and explained how we evaluated them by computer simulation. These modulation schemes are the basis of all modulation schemes. In the next chapter we introduce several applications using these modulation schemes.

References

[1] Sampei, S., *Applications of Digital Wireless Technologies to Global Wireless Communications*, Upper Saddle River, NJ: Prentice Hall, 1997.

[2] Prasad, R., *Universal Wireless Personal Communications*, Norwood, MA: Artech House, 1996.

[3] Proakis, J. G., *Digital Communications*, 3rd ed., New York: McGraw-Hill, 1995.

Appendix 3A

Program 3.1

```
% Program 3-1
%
% qpsk.m
%
% Simulation program to realize QPSK transmission system
%
% Programmed by H. Harada and T. Yamamura
%

%****************** Preparation part *****************

sr=256000.0;    % Symbol rate
ml=2;           % ml:Number of modulation levels
                % (BPSK:ml=1, QPSK:ml=2, 16QAM:ml=4)
br=sr .* ml;    % Bit rate
nd = 1000;      % Number of symbols that simulates in
                % each loop
ebn0=3;         % Eb/N0
IPOINT=8;       % Number of oversamples

%*************** Filter initialization ***************

irfn=21;    % Number of taps
alfs=0.5;   % Rolloff factor
[xh] = hrollfcoef(irfn,IPOINT,sr,alfs,1);
% Transmitter filter coefficients
[xh2] = hrollfcoef(irfn,IPOINT,sr,alfs,0);
% Receiver filter coefficients

%***************** START CALCULATION *****************

nloop=100;  % Number of simulation loops

noe = 0;    % Number of error data
nod = 0;    % Number of transmitted data

for iii=1:nloop

%******************* Data generation *****************

  data1=rand(1,nd*ml) > 0.5;    % rand: built in function

%****************** QPSK Modulation ******************

   [ich,qch]=qpskmod(data1,1,nd,ml);
   [ich1,qch1]= compoversamp(ich,qch,length(ich),IPOINT);
   [ich2,qch2]= compconv(ich1,qch1,xh);

%*************** Attenuation Calculation *************
```

```
  spow=sum(ich2.*ich2+qch2.*qch2)/nd;
  attn=0.5*spow*sr/br*10.^(-ebn0/10);
  attn=sqrt(attn);  % sqrt: built in function

%****************** Fading channel ******************

  % Generated data are fed into a fading simulator
  % [ifade,qfade]=sefade(ich2,qch2,itau,dlvl,th1,n0,
  %   itnd1,now1,length(ich2),tstp,fd,flat);

  % Updata fading counter
  % itnd1 = itnd1+ itnd0;

%*********** Add White Gaussian Noise (AWGN) ***********

  [ich3,qch3]= comb(ich2,qch2,attn);
     % add white gaussian noise
  [ich4,qch4]= compconv(ich3,qch3,xh2);
  syncpoint=irfn*IPOINT+1;
  ich5=ich4(syncpoint:IPOINT:length(ich4));
  qch5=qch4(syncpoint:IPOINT:length(qch4));

%***************** QPSK Demodulation *****************

[demodata]=qpskdemod(ich5,qch5,1,nd,ml);

%*************** Bit Error Rate (BER) ***************

  noe2=sum(abs(data1-demodata));  % sum: built in function
  nod2=length(data1);  % length: built in function
  noe=noe+noe2;
  nod=nod+nod2;
  fprintf('%d\t%e\n',iii,noe2/nod2);
     % fprintf: built in function
end % for iii=1:nloop

%******************* Output result *******************

ber = noe/nod;
fprintf('%d\t%d\t%d\t%e\n',ebn0,noe,nod,noe/nod);
  % fprintf: built in function
fid = fopen('BERqpsk.dat','a');
fprintf(fid,'%d\t%e\t%f\t%f\t\n',ebn0,noe/nod,noe,nod);
  % fprintf: built in function
fclose(fid);

%******************** end of file *******************
```

Program 3.2

```
% Program 3-2
% bpsk_fading.m
%
```

```
% Simulation program to realize BPSK transmission system
% (under one path fading)
%
% Programmed by H. Harada and T. Yamamura
%

%****************** Preparation part *****************

sr=256000.0;    % Symbol rate
ml=1;           % Number of modulation levels
br=sr.*ml;      % Bit rate (=symbol rate in this case)
nd = 100;       % Number of symbols that simulates in
                %  each loop
ebn0=10;        % Eb/N0
IPOINT=8;       % Number of oversamples

%*************** Filter initialization ***************

irfn=21;        % Number of filter taps
alfs=0.5;       % Rolloff factor
[xh] = hrollfcoef(irfn,IPOINT,sr,alfs,1);
  %Transmitter filter coefficients
[xh2] = hrollfcoef(irfn,IPOINT,sr,alfs,0);
  %Receiver filter coefficients

%*************** Fading initialization ***************

% If you use fading function "sefade", you can
% initialize all of parameters.
% Otherwise you can comment out the following
% initialization.
% The detailed explanations of all of variables are
% mentioned in Program 2-8.

% Time resolution

tstp=1/sr/IPOINT;

% Arrival time for each multipath normalized by tstp
% If you would like to simulate under one path fading
% model, you have only to set
% direct wave.

itau = [0];

% Mean power for each multipath normalized by direct
% wave.
% If you would like to simulate under one path fading
% model, you have only to set
% direct wave.
dlvl = [0];
```

```
% Number of waves to generate fading for each multipath.
% In normal case, more than six waves are needed to
% generate Rayleigh fading
n0=[6];

% Initial phase of delayed wave
% In this simulation one-path Rayleigh fading is
% considered.
th1=[0.0];

% Number of fading counter to skip
itnd0=nd*IPOINT*100;

% Initial value of fading counter
% In this simulation one-path Rayleigh fading is
% considered.
% Therefore one fading counter is needed.

itnd1=[1000];

% Number of direct wave + Number of delayed wave
% In this simulation one-path Rayleigh fading is
% considered
now1=1;

% Maximum Doppler frequency [Hz]
% You can insert your favorite value
fd=160;

% You can decide two modes to simulate fading by changing
% the variable flat
% flat     : flat fading or not
% (1->flat (only amplitude is fluctuated),0->normal
% (phase and amplitude are fluctuated))
flat =1;

%****************** START CALCULATION *****************

nloop=1000;  % Number of simulation loops
noe = 0;    % Number of error data
nod = 0;    % Number of transmitted data

for iii=1:nloop

%****************** Data generation *******************

  data=rand(1,nd) > 0.5;  % rand: built in function

%****************** BPSK Modulation *******************

  data1=data.*2-1;
  [data2] = oversamp( data1, nd , IPOINT) ;
  data3 = conv(data2,xh);  % conv: built in function
```

```
%*************** Attenuation Calculation **************

      spow=sum(data3.*data3)/nd;
      attn=0.5*spow*sr/br*10.^(-ebn0/10);
      attn=sqrt(attn);

%******************* Fading channel *******************

  % Generated data are fed into a fading simulator
  % In the case of BPSK, only Ich data are fed into
  %  fading counter
  [ifade,qfade]=sefade(data3,zeros(1,length(data3)),itau,...
  dlvl,th1,n0,itnd1,now1,length(data3),tstp,fd,flat);

  % Updata fading counter
  itnd1 = itnd1+ itnd0;

%*********** Add White Gaussian Noise (AWGN) ***********

      inoise=randn(1,length(ifade)).*attn;
      % randn: built in function
      data4=ifade+inoise;
      data5=conv(data4,xh2);  % conv: built in function
      sampl=irfn*IPOINT+1;
      data6 = data5(sampl:8:8*nd+sampl-1);

%***************** BPSK Demodulation *****************

      demodata=data6 > 0;

%***************** Bit Error Rate (BER) ***************

      % count number of instantaneous errors
      noe2=sum(abs(data-demodata));
      % sum: built in function

      % count number of instantaneous transmitted data
      nod2=length(data);  % length: built in function
      noe=noe+noe2;
      nod=nod+nod2;

      fprintf('%d\t%e\n',iii,noe2/nod2);
end % for iii=1:nloop

%******************* Output result *******************

ber = noe/nod;
fprintf('%d\t%d\t%d\t%e\n',ebn0,noe,nod,noe/nod);
fid = fopen('BERbpskfad.dat','a');
fprintf(fid,'%d\t%e\t%f\t%f\t\n',ebn0,noe/nod,noe,nod);
fclose(fid);

%******************** end of file ********************
```

Program 3.3

```
% Program 3-3
% hrollfcoef.m
%
% Generate coefficients of Nyquist filter
%
% programmed by H. Harada
%

function [xh] = hrollfcoef(irfn,ipoint,sr,alfs,ncc)

%********************** variables *********************

% irfn   : Number of symbols to use filtering
% ipoint : Number of samples in one symbol
% sr     : symbol rate
% alfs   : rolloff coefficients
% ncc    : 1 -- transmitting filter  0 -- receiving filter

% *****************************************************

xi=zeros(1,irfn*ipoint+1);
xq=zeros(1,irfn*ipoint+1);

point = ipoint;
tr = sr ;
tstp = 1.0 ./ tr ./ ipoint;
n = ipoint .* irfn;
mid = ( n ./ 2 ) + 1;
sub1 = 4.0 .* alfs .* tr;     % 4*alpha*R_s

for i = 1 : n

  icon = i - mid;
  ym = icon;

  if icon == 0.0

     xt = (1.0-alfs+4.0.*alfs./pi).* tr;  % h(0)
  else
     sub2 =16.0.*alfs.*alfs.*ym.*ym./ipoint./ipoint;
     if sub2 ~= 1.0
        x1=sin(pi*(1.0-alfs)/ipoint*ym)./pi./...
           (1.0-sub2)./ym./tstp;
        x2=cos(pi*(1.0+alfs)/ipoint*ym)./...
           pi.*sub1./(1.0-sub2);
        xt = x1 + x2;  % h(t)
     else % (4alphaRst)^2 = 1
        xt = alfs.*tr.*((1.0-2.0/pi).*cos...
           pi/4.0/alfs)+(1.0+2.0./pi).*sin...
           (pi/4.0/alfs))./sqrt(2.0);
```

```
     end    % if sub2 ~= 1.0
  end    % if icon == 0.0

  if ncc == 0       % in the case of receiver
     xh( i ) = xt ./ ipoint ./ tr;  % normalization
  elseif ncc == 1 % in the case of transmitter
     xh( i ) = xt ./ tr;             % normalization
  else
     error('ncc error');
  end    %  if ncc == 0
end  % for i = 1 : n

%********************* end of file ********************
```

Program 3.4

```
% Program 3-4
% oversamp.m
%
% Insert zero data to input data
%
% Programmed by H. Harada
%

function [out] = oversamp( indata, nsymb, sample)

%********************** variables *********************

% indata : input sequence
% nsymb  : Number of symbols
% sample : Number of oversample

% *****************************************************

out=zeros(1,nsymb*sample);
out(1:sample:1+sample*(nsymb-1))=indata;

%********************* end of file ********************
```

Program 3.5

```
% Program 3-5
% qpsk.m
%
% Simulation program to realize QPSK transmission system
%
% Programmed by H. Harada and T. Yamamura
%

%****************** Preparation part *****************
```

```
sr=256000.0;     % Symbol rate
ml=2;            % ml:Number of modulation levels
br=sr .* ml;     % Bit rate
nd = 1000;       % Number of symbols
ebn0=3;          % Eb/N0
IPOINT=8;        % Number of oversamples

%*************** Filter initialization ****************

irfn=21;    % Number of taps
alfs=0.5;   % Rolloff factor
[xh] = hrollfcoef(irfn,IPOINT,sr,alfs,1);
  %Transmitter filter coefficients
[xh2] = hrollfcoef(irfn,IPOINT,sr,alfs,0);
  %Receiver filter coefficients

%***************** START CALCULATION ******************

nloop=100;  % Number of simulation loops
noe = 0;    % Number of error data
nod = 0; % Number of transmitted data

for iii=1:nloop

%****************** Data generation ******************

     data1=rand(1,nd*ml) > 0.5;  % rand: built in function

%****************** QPSK Modulation ******************

     [ich,qch]=qpskmod(data1,1,nd,ml);
     [ich1,qch1]= compoversamp(ich,qch,length(ich),...
        IPOINT);
     [ich2,qch2]= compconv(ich1,qch1,xh);

%*************** Attenuation Calculation **************

     spow=sum(ich2.*ich2+qch2.*qch2)/nd;
        % sum: built in function
     attn=0.5*spow*sr/br*10.^(-ebn0/10);
     attn=sqrt(attn);  % sqrt: built in function

 %****************** Fading channel ******************

  % Generated data are fed into a fading simulator
  % [ifade,qfade]=sefade(ich2,qch2,itau,dlvl,th1,n0,
  % itnd1,now1,length(ich2),tstp,fd,flat);

  % Updata fading counter
  % itnd1 = itnd1+ itnd0;

%*********** Add White Gaussian Noise (AWGN) ***********
```

```
     [ich3,qch3]= comb(ich2,qch2,attn);
        % add white gaussian noise
     [ich4,qch4]= compconv(ich3,qch3,xh2);

     syncpoint=irfn*IPOINT+1;
     ich5=ich4(syncpoint:IPOINT:length(ich4));
     qch5=qch4(syncpoint:IPOINT:length(qch4));

%****************** QPSK Demodulation *****************

    [demodata]=qpskdemod(ich5,qch5,1,nd,ml);

%*************** Bit Error Rate (BER) ***************

     noe2=sum(abs(data1-demodata));
        % sum: built in function
     nod2=length(data1);  % length: built in function
     noe=noe+noe2;
     nod=nod+nod2;
     fprintf('%d\t%e\n',iii,noe2/nod2);
        % fprintf: built in function

end % for iii=1:nloop

%******************** Output result ******************

ber = noe/nod;
fprintf('%d\t%d\t%d\t%e\n',ebn0,noe,nod,noe/nod);
  % fprintf: built in function
fid = fopen('BERqpsk.dat','a');
fprintf(fid,'%d\t%e\t%f\t%f\t\n',ebn0,noe/nod,noe,nod);
  % fprintf: built in function
fclose(fid);

%******************** end of file ******************
```

Program 3.6

```
% Program 3-6
% qpsk_fading.m
%
% Simulation program to realize QPSK transmission system
% (under one path fading)
%
% Programmed by H. Harada and T. Yamamura
%

%****************** Preparation part *****************

sr=256000.0;    % Symbol rate
ml=2;           % ml:Number of modulation levels
```

```
                % (BPSK:ml=1, QPSK:ml=2, 16QAM:ml=4)
br=sr .* ml;    % Bit rate
nd = 100;       % Number of symbols that simulates in
                % each loop
ebn0=10;        % Eb/N0
IPOINT=8;       % Number of oversamples

%**************** Filter initialization ***************

irfn=21;        % Number of taps
alfs=0.5;       % Rolloff factor
[xh] = hrollfcoef(irfn,IPOINT,sr,alfs,1);
  %Transmitter filter coefficients
[xh2] = hrollfcoef(irfn,IPOINT,sr,alfs,0);
  %Receiver filter coefficients

%*************** Fading initialization ****************

% If you use fading function "sefade", you can
% initialize all of parameters.
% Otherwise you can comment out the following
% initialization.
% The detailed explanations of all of variables are
% mentioned in Program 2-8.

% Time resolution

tstp=1/sr/IPOINT;

% Arrival time for each multipath normalized by tstp
% If you would like to simulate under one path fading
% model, you have only to set
% direct wave.

itau = [0];

% Mean power for each multipath normalized by direct
% wave.
% If you would like to simulate under one path fading
% model, you have only to set
% direct wave.
dlvl = [0];

% Number of waves to generate fading for each multipath.
% In normal case, more than six waves are needed to
% generate Rayleigh fading
n0=[6];

% Initial phase of delayed wave
% In this simulation one-path Rayleigh fading is
% considered.
th1=[0.0];
```

```
% Number of fading counter to skip
itnd0=nd*IPOINT*100;

% Initial value of fading counter
% In this simulation one-path Rayleigh fading is
% considered.
% Therefore one fading counter is needed.

itnd1=[1000];

% Number of direct wave + Number of delayed wave
% In this simulation one-path Rayleigh fading is
% considered
now1=1;

% Maximum Doppler frequency [Hz]
% You can insert your favorite value
fd=160;

% You can decide two modes to simulate fading by changing
% the variable flat
% flat     : flat fading or not
% (1-flat (only amplitude is fluctuated),0-normal(phase
% and amplitude are fluctuated))
flat =1;

%***************** START CALCULATION *****************

nloop=1000;  % Number of simulation loops

noe = 0;     % Number of error data
nod = 0;     % Number of transmitted data
for iii=1:nloop

%****************** Data generation ******************

      data1=rand(1,nd*ml) > 0.5;  % rand: built in function

%******************* QPSK Modulation *****************

      [ich,qch]=qpskmod(data1,1,nd,ml);
      [ich1,qch1]= compoversamp(ich,qch,length(ich),...
         IPOINT);
      [ich2,qch2]= compconv(ich1,qch1,xh);

%************** Attenuation Calculation **************

      spow=sum(ich2.*ich2+qch2.*qch2)/nd;
         % sum: built in function
      attn=0.5*spow*sr/br*10.^(-ebn0/10);
      attn=sqrt(attn);  % sqrt: built in function

%******************* Fading channel ******************
```

```
  %   Generated data are fed into a fading simulator
      [ifade,qfade]=sefade(ich2,qch2,itau,dlvl,th1,n0,...
         itnd1,now1,length(ich2),tstp,fd,flat);

  %   Updata fading counter
      itnd1 = itnd1+ itnd0;

%*********** Add White Gaussian Noise (AWGN) ***********

      [ich3,qch3]= comb(ifade,qfade,attn);
         % add white gaussian noise
      [ich4,qch4]= compconv(ich3,qch3,xh2);

      syncpoint=irfn*IPOINT+1;
      ich5=ich4(syncpoint:IPOINT:length(ich4));
      qch5=qch4(syncpoint:IPOINT:length(qch4));

%****************** QPSK Demodulation *****************

      [demodata]=qpskdemod(ich5,qch5,1,nd,ml);

%**************** Bit Error Rate (BER) ****************

      noe2=sum(abs(data1-demodata));
         % sum: built in function
      nod2=length(data1);  % length: built in function
      noe=noe+noe2;
      nod=nod+nod2;

      fprintf('%d\t%e\n',iii,noe2/nod2);
         % fprintf: built in function

end % for iii=1:nloop

%******************** Output result *******************

ber = noe/nod;
fprintf('%d\t%d\t%d\t%e\n',ebn0,noe,nod,noe/nod);
  % fprintf: built in function
fid = fopen('BERqpskfad.dat','a');
fprintf(fid,'%d\t%e\t%f\t%f\t\n',ebn0,noe/nod,noe,nod);
  % fprintf: built in function
fclose(fid);

%********************* end of file ********************
```

Program 3.7

```
% Program 3-7
% compconv.m
%
% Function to perform convolution between signal and
% filter
```

```
%
% programmed by H. Harada and M. Okita
%

function [iout, qout] = compconv(idata, qdata, filter)

% *****************************************************

% idata      : ich data sequence
% qdata      : qch data sequence
% filter     : filter tap coefficients

% *****************************************************

iout = conv(idata,filter);
qout = conv(qdata,filter);

%******************** end of file ********************
```

Program 3.8

```
% Program 3-8
% compoversamp.m
%
% Insert zero data to Ich and Qch input data
%
% programmed by H. Harada
%

function [iout,qout] = compoversamp(idata,qdata,nsymb,...
  sample)

%********************* variables ********************

% idata  : input Ich data
% qdata  : input Qch data
% iout   : output Ich data
% qout   : output Qch data
% nsymb  : Number of burst symbol
% sample : Number of oversample
% *****************************************************

iout=zeros(1,nsymb*sample);
qout=zeros(1,nsymb*sample);
iout(1:sample:1+sample*(nsymb-1))=idata;
qout(1:sample:1+sample*(nsymb-1))=qdata;

%********************* end of file ********************
```

Program 3.9

```
% Program 3-9
% qpskmod.m
%
% Function to perform QPSK modulation
%
% Programmed by H. Harada
%

function [iout,qout]=qpskmod(paradata,para,nd,ml)

%********************** variables *********************

% paradata : input data (para-by-nd matrix)
% iout :output Ich data
% qout :output Qch data
% para   : Number of parallel channels
% nd : Number of data
% ml : Number of modulation levels
% (QPSK -2  16QAM - 4)

% *****************************************************

m2=ml./2;

paradata2=paradata.*2-1;
count2=0;

for jj=1:nd

      isi = zeros(para,1);
      isq = zeros(para,1);

      for ii = 1 : m2
         isi = isi + 2.^( m2 - ii )...
             .* paradata2((1:para),ii+count2);
         isq = isq + 2.^( m2 - ii )...
             .* paradata2((1:para),m2+ii+count2);
      end

      iout((1:para),jj)=isi;
      qout((1:para),jj)=isq;

      count2=count2+ml;

end

%********************* end of file ********************
```

Program 3.10

```
% Program 3-10
% qpskdemod.m
%
% Function to perform QPSK demodulation
%
% programmed by H. Harada
%

function [demodata]=qpskdemod(idata,qdata,para,nd,ml)

%********************** variables *********************

% idata      :input Ich data
% qdata      :input Qch data
% demodata   : demodulated data (para-by-nd matrix)
% para       : Number of parallel channels
% nd         : Number of data
% ml         : Number of modulation levels
% (QPSK -2  16QAM - 4)

% *****************************************************

demodata=zeros(para,ml*nd);
demodata((1:para),(1:ml:ml*nd-1))=idata((1:para),...
   (1:nd))>=0;
demodata((1:para),(2:ml:ml*nd))=qdata((1:para),...
   (1:nd))>=0;

%********************* end of file ********************
```

Program 3.11

```
% Program 3-11
% oqpsk.m
%
% Simulation program to realize OQPSK transmission
  system
%
% Programmed by H. Harada and T. Yamamura
%

%****************** Preparation part ******************

sr=256000.0;    % Symbol rate
ml=2;           % ml:Number of modulation levels
                % (BPSK:ml=1, QPSK:ml=2, 16QAM:ml=4)
br=sr.*ml;      % bit rate
nd = 1000;      % Number of symbols that simulates in
                % each loop
```

```
ebn0=3;            % Eb/N0
IPOINT=8;          % Number of oversamples

%*************** Filter initialization ***************

irfn=21;           % Number of taps
alfs=0.5;          % Rolloff factor
[xh] = hrollfcoef(irfn,IPOINT,sr,alfs,1);
  %Transmitter filter coefficients
[xh2] = hrollfcoef(irfn,IPOINT,sr,alfs,0);
  %Receiver filter coefficients

%***************** START CALCULATION *****************

nloop=100;  % Number of simulation loops
noe = 0;    % Number of error data
nod = 0;    % Number of transmitted data

for iii=1:nloop

%****************** Data generation ******************

    data1=rand(1,nd*ml) > 0.5;  % rand: built in function

%****************** OQPSK Modulation *****************

      [ich,qch]=qpskmod(data1,1,nd,ml);
      [ich1,qch1]=compoversamp(ich,qch,length(ich),...
         IPOINT);
      ich21=[ich1 zeros(1,IPOINT/2)];
      qch21=[zeros(1,IPOINT/2) qch1];
      [ich2, qch2]=compconv(ich21,qch21,xh);

%************** Attenuation Calculation ***************

      spow=sum(ich2.*ich2+qch2.*qch2)/nd;
         % sum: built in function
      attn=0.5*spow*sr/br*10.^(-ebn0/10);
      attn=sqrt(attn);
         % sqrt: built in function

%****************** Fading channel *******************

  % Generated data are fed into a fading simulator
  % [ifade,qfade]=sefade(ich2,qch2,itau,dlvl,th1,
  % n0,itnd1,now1,length(ich1),tstp,fd,flat);

  % Updata fading counter
  % itnd1 = itnd1+ itnd0;

%*********** Add White Gaussian Noise (AWGN) ***********
```

```
      [ich3,qch3]= comb(ich2,qch2,attn);
         % add white gaussian noise
      [ich4,qch4]= compconv(ich3,qch3,xh2);

      syncpoint=irfn*IPOINT+1;
      ich5=ich4(syncpoint:IPOINT:length(ich4));
      qch5=qch4(syncpoint+IPOINT/2:IPOINT:length(qch4));

%***************** OQPSK Demodulation *****************

   [demodata]=qpskdemod(ich5,qch5,1,nd,ml);

%*************** Bit Error Rate (BER) ****************

      noe2=sum(abs(data1-demodata));
         % sum: built in function
      nod2=length(data1);
         % length: built in function
      noe=noe+noe2;
      nod=nod+nod2;

      fprintf('%d\t%e\n',iii,noe2/nod2);
         % fprintf: built in function

end % for iii=1:nloop

%******************** Output result ******************

ber = noe/nod;
fprintf('%d\t%d\t%d\t%e\n',ebn0,noe,nod,noe/nod);
  % fprintf: built in function
fid = fopen('BERoqpsk.dat','a');
fprintf(fid,'%d\t%e\t%f\t%f\t\n',ebn0,noe/nod,noe,nod);
  % fprintf: built in function
fclose(fid);

%******************** end of file ********************
```

Program 3.12

```
% Program 3-12
% oqpsk_fading.m
%
% Simulation program to realize OQPSK transmission
%  system
% (under one path fading)
%
% Programmed by H. Harada and T. Yamamura
%

%****************** Preparation part *****************
```

```
sr=256000.0;    % Symbol rate
ml=2;           % ml:Number of modulation levels
                % (BPSK:ml=1, QPSK:ml=2, 16QAM:ml=4)
br=sr .* ml;    % Bit rate
nd = 1000;      % Number of symbols that simulates in
                % each loop
ebn0=10;        % Eb/N0
IPOINT=8;       % Number of oversamples

%*************** Filter initialization ****************

irfn=21;        % Number of taps
alfs=0.5;       % Rolloff factor
[xh] = hrollfcoef(irfn,IPOINT,sr,alfs,1);
  %Transmitter filter coefficients
[xh2] = hrollfcoef(irfn,IPOINT,sr,alfs,0);
  %Receiver filter coefficients

%*************** Fading initialization ****************

% If you use fading function "sefade", you can
% initialize all of parameters.
% Otherwise you can comment out the following
% initialization.
% The detailed explanations of all of variables are
% mentioned in Program 2-8.

% Time resolution

tstp=1/sr/IPOINT;

% Arrival time for each multipath normalized by tstp
% If you would like to simulate under one path fading
% model, you have only to set
% direct wave.

itau = [0];

% Mean power for each multipath normalized by direct
% wave.
% If you would like to simulate under one path fading
% model, you have only to set
% direct wave.
dlvl = [0];

% Number of waves to generate fading for each multipath.
% In normal case, more than six waves are needed to
% generate Rayleigh fading
n0=[6];

% Initial phase of delayed wave
% In this simulation one-path Rayleigh fading is
% considered.
th1=[0.0];
```

```
% Number of fading counter to skip
itnd0=nd*IPOINT;

% Initial value of fading counter
% In this simulation one-path Rayleigh fading is
% considered.
% Therefore one fading counter is needed.

itnd1=[1000];

% Number of direct wave + Number of delayed wave
% In this simulation one-path Rayleigh fading is
% considered
now1=1;

% Maximum Doppler frequency [Hz]
% You can insert your favorite value
fd=160;

% You can decide two modes to simulate fading by changing
% the variable flat
% flat     : flat fading or not
% (1-flat (only amplitude is fluctuated),0-normal(phase
% and amplitude are fluctuated))
flat =1;

%**************** START CALCULATION *****************

nloop=200;  % Number of simulation loops

noe = 0;    % Number of error data
nod = 0;    % Number of transmitted data

for iii=1:nloop

%****************** Data generation *****************

      data1=rand(1,nd*ml) > 0.5;  % rand: built in function

%***************** OQPSK Modulation *****************

      [ich,qch]=qpskmod(data1,1,nd,ml);
      ich1,qch1]=compoversamp(ich,qch,length(ich),...
         IPOINT);
      ich21=[ich1 zeros(1,IPOINT/2)];
      qch21=[zeros(1,IPOINT/2) qch1];
      [ich2, qch2]=compconv(ich21,qch21,xh);

%************** Attenuation Calculation *************

      spow=sum(ich2.*ich2+qch2.*qch2)/nd;
         % sum: built in function
      attn=0.5*spow*sr/br*10.^(-ebn0/10);
      attn=sqrt(attn);  % sqrt: built in function

%******************** Fading channel *******************

  % Generated data are fed into a fading simulator
  [ifade,qfade]=sefade(ich2,qch2,itau,dlvl,th1,n0,...
      itnd1,now1,length(ich1),tstp,fd,flat);

  % Updata fading counter
      itnd1 = itnd1+ itnd0;

%*********** Add White Gaussian Noise (AWGN) ***********

      [ich3,qch3]= comb(ifade,qfade,attn);
         % add white gaussian noise
      [ich4,qch4]= compconv(ich3,qch3,xh2);

      syncpoint=irfn*IPOINT+1;
      ich5=ich4(syncpoint:IPOINT:length(ich4));
      qch5=qch4(syncpoint+IPOINT/2:IPOINT:length(qch4));

%***************** OQPSK Demodulation *****************

      [demodata]=qpskdemod(ich5,qch5,1,nd,ml);

%**************** Bit Error Rate (BER) ****************

      noe2=sum(abs(data1-demodata));
         % sum: built in function
      nod2=length(data1);
         % length: built in function
      noe=noe+noe2;
      nod=nod+nod2;
      fprintf('%d\t%e\n',iii,noe2/nod2);
         % fprintf: built in function
end % for iii=1:nloop

%******************* Output result ********************

ber = noe/nod;
fprintf('%d\t%d\t%d\t%e\n',ebn0,noe,nod,noe/nod);
  % fprintf: built in function
fid = fopen('BERoqpskfad.dat','a');
fprintf(fid,'%d\t%e\t%f\t%f\t\n',ebn0,noe/nod,noe,nod);
  % fprintf: built in function
fclose(fid);

%********************* end of file ********************
```

Program 3.13

```
% Program 3-13
% msk.m
%
% Simulation program to realize MSK transmission system
%
% Programmed by H. Harada and T. Yamamura
%

%****************** preparation part ******************

sr=256000.0;    % Symbol rate
ml=1;           % ml:Number of modulation levels
br=sr.*ml;      % Bit rate
nd = 1000;      % Number of symbols that simulates in
                % each loop
ebn0=3;         % Eb/N0
IPOINT=8;       % Number of oversamples

%***************** START CALCULATION ******************

nloop=100;  % Number of simulation loops

noe = 0;    % Number of error data
nod = 0;    % Number of transmitted data

for iii=1:nloop

%****************** Data generation *******************

      data1=rand(1,nd) > 0.5;  % rand: built in function

%****************** MSK Modulation ********************

      [ich,qch]=qpskmod(data1,1,nd/2,2);
      smooth1=cos(pi/2*[-1+1./4.*[0:IPOINT-1]]);
         %IPOINT point filtering

      for ii=1:length(ich)
         ich2((ii-1)*IPOINT+1:ii*IPOINT)=(-1)^(ii-1)...
             *smooth1.*ich(ii);
         qch2((ii-1)*IPOINT+1:ii*IPOINT)=(-1)^(ii-1)...
             *smooth1.*qch(ii);
      end

      ich21=[ich2 zeros(1,IPOINT/2)];
         qch21=[zeros(1,IPOINT/2) qch2];

%*************** Attenuation Calculation **************

      spow=sum(ich21.*ich21+qch21.*qch21)/nd/2;
         % sum: built in function
      attn=0.5*spow*sr/br/2*10.^(-ebn0/10);
      attn=sqrt(attn);    % sqrt: built in function

%******************** Fading channel *******************

% [ifade,qfade]=sefade2(data2,qdata1,itau,dlvl1,th1,n0,...
% itnd1,now1,length(data2),fftlen2,fstp,fd,flat);

%*********** Add White Gaussian Noise (AWGN) ***********

      [ich3,qch3]= comb(ich21,qch21,attn);
         % add white gaussian noise

      syncpoint=1;

      ich5 = ich3(syncpoint+IPOINT/2:IPOINT:length(ich2));
      qch5 = qch3(syncpoint+IPOINT:IPOINT:length(ich2)...
         +IPOINT/2);

      ich5(2:2:length(ich5))=-1*ich5(2:2:length(ich5));
      qch5(2:2:length(ich5))=-1*qch5(2:2:length(ich5));

%***************** MSK Demodulation *******************

      [demodata]=qpskdemod(ich5,qch5,1,nd/2,2);

%***************** Bit Error Rate (BER) ***************

      noe2=sum(abs(data1-demodata));
         % sum: built in function
      nod2=length(data1);
         % length: built in function
      noe=noe+noe2;
      nod=nod+nod2;

      fprintf('%d\t%e\n',iii,noe2/nod2);
         % fprintf: built in function
end % for iii=1:nloop

%******************** Output result *******************

ber = noe/nod;
fprintf('%d\t%d\t%d\t%e\n',ebn0,noe,nod,noe/nod);
  % fprintf: built in function
fid = fopen('BERmsk.dat','a');
fprintf(fid,'%d\t%e\t%f\t%f\t\n',ebn0,noe/nod,noe,nod);
  % fprintf: built in function
fclose(fid);

%********************* end of file ********************
```

Program 3.14

```
% Program 3-14
% msk_fading.m
%
% Simulation program to realize MSK transmission system
% (under one path fading)
%
% Programmed by H. Harada and T. Yamamura
%

%****************** preparation part *****************

sr=256000.0;    % Symbol rate
ml=1;           % ml:Number of modulation levels
br=sr.*ml;      % Bit rate
nd = 100;       % Number of symbols that simulates in
                % each loop
ebn0=10;        % Eb/N0
IPOINT=8;       % Number of oversamples

%*************** Fading initialization ***************

% If you use fading function "sefade", you can
% initialize all of parameters.
% Otherwise you can comment out the following
% initialization.
% The detailed explanations of all of variables are
% mentioned in Program 2-8.

% Time resolution

tstp=1/sr/IPOINT;

% Arrival time for each multipath normalized by tstp
% If you would like to simulate under one path fading
% model, you have only to set
% direct wave.
itau = [0];

% Mean power for each multipath normalized by direct
% wave.
% If you would like to simulate under one path fading
% model, you have only to set
% direct wave.
dlvl = [0];

% Number of waves to generate fading for each multipath.
% In normal case, more than six waves are needed to
% generate Rayleigh fading
n0=[6];
```

```
% Initial phase of delayed wave
% In this simulation one-path Rayleigh fading is
% considered.
th1=[0.0];

% Number of fading counter to skip
itnd0=nd*IPOINT*100;

% Initial value of fading counter
% In this simulation one-path Rayleigh fading is
% considered.
% Therefore one fading counter is needed.

itnd1=[1000];

% Number of direct wave + Number of delayed wave
% In this simulation one-path Rayleigh fading is
% considered
now1=1;

% Maximum Doppler frequency [Hz]
% You can insert your favorite value
fd=320;

% You can decide two modes to simulate fading by changing
% the variable flat
% flat    : flat fading or not
% (1-flat (only amplitude is fluctuated),0-normal(phase
% and amplitude are fluctuated))
flat =1;

%***************** START CALCULATION ******************

nloop=1000;     % Number of simulation loops
noe = 0;        % Number of error data
nod = 0;        % Number of transmitted data

for iii=1:nloop

%****************** Data generation ******************

  data1=rand(1,nd) > 0.5;  % rand: built in function

%******************* MSK Modulation ******************

      [ich,qch]=qpskmod(data1,1,nd/2,2);
      smooth1=cos(pi/2*[-1+1./4.*[0:IPOINT-1]]);
         %IPOINT point filtering

      for ii=1:length(ich)
         ich2((ii-1)*IPOINT+1:ii*IPOINT)=(-1)^(ii-1)...
             *smooth1.*ich(ii);
         qch2((ii-1)*IPOINT+1:ii*IPOINT)=(-1)^(ii-1)...
```

```
             *smooth1.*qch(ii);
      end

      ich21=[ich2 zeros(1,IPOINT/2)];
      qch21=[zeros(1,IPOINT/2) qch2];

%************** Attenuation Calculation ***************

      spow=sum(ich21.*ich21+qch21.*qch21)/nd/2;
         % sum: built in function
      attn=0.5*spow*sr/br/2*10.^(-ebn0/10);
      attn=sqrt(attn);        % sqrt: built in function

%******************** Fading channel ******************

  % Generated data are fed into a fading simulator
      [ifade,qfade]=sefade(ich21,qch21,itau,dlvl,th1,...
      n0,itnd1,now1,length(ich21),tstp,fd,flat);

  % Updata fading counter
      itnd1 = itnd1+ itnd0;

%*********** Add White Gaussian Noise (AWGN) **********

  [ich3,qch3]= comb(ifade,qfade,attn);
      % add white gaussian noise

      syncpoint=1;
      ich5 = ich3(syncpoint+IPOINT/2:IPOINT:length(ich2));
      qch5 = qch3(syncpoint+IPOINT:IPOINT:length(ich2)...
         +IPOINT/2);

      ich5(2:2:length(ich5))=-1*ich5(2:2:length(ich5));
      qch5(2:2:length(ich5))=-1*qch5(2:2:length(ich5));

%****************** MSK Demodulation *****************

      [demodata]=qpskdemod(ich5,qch5,1,nd/2,2);

%**************** Bit Error Rate (BER) ****************

      noe2=sum(abs(data1-demodata));
         % sum: built in function
      nod2=length(data1);
         % length: built in function
      noe=noe+noe2;
      nod=nod+nod2;

      fprintf('%d\t%e\n',iii,noe2/nod2);
         % fprintf: built in function
end % for iii=1:nloop

%******************** Output result *******************
```

```
ber = noe/nod;
fprintf('%d\t%d\t%d\t%e\n',ebn0,noe,nod,noe/nod);
  % fprintf: built in function
fid = fopen('BERmskfad.dat','a');
fprintf(fid,'%d\t%e\t%f\t%f\t\n',ebn0,noe/nod,noe,nod);
  % fprintf: built in function
fclose(fid);

%********************** end of file ********************
```

Program 3.15

```
% Program 3-15
% msk2.m
%
% Simulation program to realize MSK transmission system
%
% Programmed by R. Sawai and H. Harada
%

%****************** Preparation part ******************

sr=256000.0;      % Symbol rate
ml=1;             % ml:Number of modulation levels
br=sr.*ml;        % Bit rate
nd = 1000;        % Number of symbols that simulates in
                  % each loop
ebn0=5;           % Eb/N0
IPOINT=8;         % Number of oversamples

%***************** START CALCULATION ******************

nloop=100;  % Number of simulation loops
noe = 0;    % Number of error data
nod = 0;    % Number of transmitted data
for iii=1:nloop

%****************** Data generation ******************

  data1=rand(1,nd*ml) > 0.5;  % rand: built in function

%******************* MSK Modulation ******************

      data11=2*data1-1;
      data2=oversamp2(data11,length(data11),IPOINT);

      th=zeros(1,length(data2)+1);
      ich2=zeros(1,length(data2)+1);
      qch2=zeros(1,length(data2)+1);
```

```
      for ii=2:length(data2)+1
         th(1,ii)=th(1,ii-1)+pi/2*data2(1,ii-1)./IPOINT;
      end

      ich2=cos(th);
      qch2=sin(th);

%*************** Attenuation Calculation *************

      spow=sum(ich2.*ich2+qch2.*qch2)/(nd*IPOINT);
         % sum: built in function
      attn=0.5*spow*sr/br*10.^(-ebn0/10);
      attn=sqrt(attn);  % sqrt: built in function

%******************* Fading channel ******************

%  [ifade,qfade]=sefade2(data2,qdata1,itau,dlvl1,th1,n0,
%  itnd1,now1,length(data2),fftlen2,fstp,fd,flat);

%*********** Add White Gaussian Noise (AWGN) ***********

      [ich3,qch3]= comb(ich2,qch2,attn);
         % add white gaussian noise

      syncpoint = 1;
      ich5=ich3(syncpoint:IPOINT:length(ich3));
      qch5=qch3(syncpoint:IPOINT:length(qch3));

%***************** MSK Demodulation *****************

      demoddata2(1,1)=-1

      for k=3:2:nd*ml+1
         demoddata2(1,k)=ich5(1,k)*qch5(1,k-1)...
             *cos(pi*(k))0;
      end

      for n=2:2:nd*ml+1
         demoddata2(1,n)=ich5(1,n-1)*qch5(1,n)...
         *cos(pi*(n))0;
      end

      [demodata]=demoddata2(1,2:nd*ml+1);

%*************** Bit Error Rate (BER) ***************

      noe2=sum(abs(data1-demodata));
         % sum: built in function
      nod2=length(data1);
      % length: built in function
      noe=noe+noe2;
      nod=nod+nod2;
      fprintf(‘%d\t%e\n’,iii,noe2/nod2);
         % fprintf: built in function
end % for iii=1:nloop

%********************** Data file ********************

ber = noe/nod;
fprintf(‘%d\t%d\t%d\t%e\n’,ebn0,noe,nod,noe/nod);
  % fprintf: built in function
fid = fopen(‘BERmsk2.dat’,‘a’);
fprintf(fid,‘%d\t%e\t%f\t%f\t\n’,ebn0,noe/nod,noe,nod);
  % fprintf: built in function
fclose(fid);

%********************* end of file *******************
```

Program 3.16

```
% Program 3-16
% msk2_fading.m
%
% Simulation program to realize MSK transmission system
% (under one path fading)
%
% Programmed by R. Sawai and H. Harada
%

%****************** Preparation part *****************

sr=256000.0;     % Symbol rate
ml=1;            % ml:Number of modulation levels
br=sr.*ml;       % Bit rate
nd = 100;        % Number of symbols that simulates in
                 % each loop
ebn0=15;         % Eb/N0
IPOINT=8;        % Number of oversamples

%**************** Fading initialization **************

% If you use fading function "sefade", you can
% initialize all of parameters.
% Otherwise you can comment out the following
% initialization.
% The detailed explanations of all of variables are
% mentioned in Program 2-8.
% Time resolution

tstp=1/sr/IPOINT;

% Arrival time for each multipath normalized by tstp
% If you would like to simulate under one path fading
```

```
% model, you have only to set
% direct wave.

itau = [0];

% Mean power for each multipath normalized by direct
% wave.
% If you would like to simulate under one path fading
% model, you have only to set
% direct wave.
dlvl = [0];

% Number of waves to generate fading for each multipath.
% In normal case, more than six waves are needed to
% generate Rayleigh fading
n0=[6];

% Initial phase of delayed wave
% In this simulation one-path Rayleigh fading is
% considered.
th1=[0.0];

% Number of fading counter to skip
itnd0=nd*IPOINT*100;

% Initial value of fading counter
% In this simulation one-path Rayleigh fading is
% considered.
% Therefore one fading counter is needed.

itnd1=[3000];

% Number of direct wave + Number of delayed wave
% In this simulation one-path Rayleigh fading is
% considered
now1=1;

% Maximum Doppler frequency [Hz]
% You can insert your favorite value
fd=320;

% You can decide two modes to simulate fading by changing
% the variable flat
% flat     : flat fading or not
% (1-flat (only amplitude is fluctuated),0-normal(phase
% and amplitude are fluctuated))
flat =1;

%****************** START CALCULATION ****************

nloop=1000;  % Number of simulation loops
```

```
noe = 0;    % Number of error data
nod = 0;    % Number of transmitted data

for iii=1:nloop

%****************** Data generation *******************

      data1=rand(1,nd*ml) > 0.5;  % rand: built in function

%******************** MSK Modulation ******************

      data11=2*data1-1;
      data2=oversamp2(data11,length(data11),IPOINT);

      th=zeros(1,length(data2)+1);
      ich2=zeros(1,length(data2)+1);
      qch2=zeros(1,length(data2)+1);

      for ii=2:length(data2)+1
         th(1,ii)=th(1,ii-1)+pi/2*data2(1,ii-1)./IPOINT;
      end

      ich2=cos(th);
      qch2=sin(th);

%************** Attenuation Calculation ***************

      spow=sum(ich2.*ich2+qch2.*qch2)/(nd*IPOINT);
         % sum: built in function
      attn=0.5*spow*sr/br*10.^(-ebn0/10);
      ttn=sqrt(attn);  % sqrt: built in function

%******************* Fading channel *******************

   % Generated data are fed into a fading simulator
   [ifade,qfade]=sefade(ich2,qch2,itau,dlvl,th1,n0,...
      itndl,nowl,length(ich2),tstp,fd,flat);

   % Updata fading counter
      itndl = itndl+ itnd0;

%********** Add White Gaussian Noise (AWGN) ***********

      [ich3,qch3]= comb(ifade,qfade,attn);
      % add white gaussian noise

      syncpoint = 1;
      ich5=ich3(syncpoint:IPOINT:length(ich3));
      qch5=qch3(syncpoint:IPOINT:length(qch3));

%****************** MSK Demodulation ******************

      demoddata2(1,1)=-1;
```

```
      for k=3:2:nd*ml+1
         demoddata2(1,k)=ich5(1,k)*qch5(1,k-1)...
            *cos(pi*(k)) > 0;
      end

      for n=2:2:nd*ml+1
         demoddata2(1,n)=ich5(1,n-1)*qch5(1,n)...
            *cos(pi*(n)) > 0;
      end

      [demodata]=demoddata2(1,2:nd*ml+1);

%**************** Bit Error Rate (BER) ****************

      noe2=sum(abs(data1-demodata));
         % sum: built in function
      nod2=length(data1);
         % length: built in function
      noe=noe+noe2;
      nod=nod+nod2;

      fprintf(‘%d\t%e\n’,iii,noe2/nod2);
         % fprintf: built in function

end % for iii=1:nloop

%********************* Data file **********************

ber = noe/nod;
fprintf(‘%d\t%d\t%d\t%e\n’,ebn0,noe,nod,noe/nod);
  % fprintf: built in function
fid = fopen(‘BERmsk2fad.dat’,‘a’);
fprintf(fid,‘%d\t%e\t%f\t%f\t\n’,ebn0,noe/nod,noe,nod);
  % fprintf: built in function
fclose(fid);

%********************* end of file ********************
```

Program 3.17

```
% Program 3-17
% oversamp2.m
%
% Function to sample “sample” time
%
% programmed by H. Harada
%

function [out] = oversamp2( iin, ntot, sample)

%*******************************************************
```

```
% iin    : input sequence
% ntot   : Number of burst symbol
% sample : Number of oversample

% ******************************************************

for k=1:sample
  out(k:sample:k+sample*(ntot-1))=iin;
end

%******************** end of file *******************
```

Program 3.18

```
% Program 3-18
% gmsk.m
%
% Simulation program to realize GMSK transmission system
%
% Programmed by R. Sawai and H. Harada
%

%****************** Preparation part ******************

sr=256000.0;    % Symbol rate
ml=1;           % ml:Number of modulation levels
br=sr.*ml;      % Bit rate
nd = 1000;      % Number of symbols that simulates in
                % each loop
ebn0=5;         % Eb/N0
IPOINT=8;       % Number of oversamples

%*************** Filter initialization ****************

irfn=21;            % Number of taps
B=0.25*sr;
B2=0.6*sr;
[xh] = gaussf(B,irfn,IPOINT,sr,1);
% Transmitter filter coefficients
[xh2] =gaussf(B2,irfn,IPOINT,sr,0);
% Receiver filter coefficients

%**************** START CALCULATION *****************

nloop=100;  % Number of simulation loops

noe = 0;    % Number of error data
nod = 0;    % Number of transmitted data

for iii=1:nloop

%****************** Data generation *****************
```

```
     data1=rand(1,nd.*ml) > 0.5;  % rand: built in function

%****************** GMSK Modulation *****************

     data11=2*data1-1;
     data2=oversamp(data11,length(data11),IPOINT);
     data3=conv(data2,xh);      % NEW for GMSK

     th=zeros(1,length(data3)+1);
     ich2=zeros(1,length(data3)+1);
     qch2=zeros(1,length(data3)+1);

     for ii=2:length(data3)+1
        th(1,ii)=th(1,ii-1)+pi/2*data3(1,ii-1)./IPOINT;
      end

     ich2=cos(th);
     qch2=sin(th);

%*************** Attenuation Calculation *************

     spow=sum(ich2.*ich2+qch2.*qch2)/nd;
        % sum: built in function
     attn=0.5*spow*sr/br*10.^(-ebn0/10);
     attn=sqrt(attn);    % sqrt: built in function

%****************** Fading channel ******************

     % [ifade,qfade]=sefade2(data2,qdata1,itau,dlvl1,th1,
     % n0, itnd1,now1,length(data2),fftlen2,fstp,fd,flat);

%*********** Add White Gaussian Noise (AWGN) **********

     [ich3,qch3]= comb(ich2,qch2,attn);
        % add white gaussian noise

     [ich4,qch4] = compconv(ich3,qch3,xh2);

     syncpoint =irfn*IPOINT-IPOINT/2+1;
     ich5=ich4(syncpoint:IPOINT:length(ich4));
     qch5=qch4(syncpoint:IPOINT:length(qch4));

%***************** GMSK Demodulation ****************

     demoddata2(1,1)=-1;

     for k=3:2:nd*ml+1
        demoddata2(1,k)=ich5(1,k)*qch5(1,k-1)...
            *cos(pi*(k)) > 0;
     end

     for n=2:2:nd*ml+1
        demoddata2(1,n)=ich5(1,n-1)*qch5(1,n)...
```

```
            *cos(pi*(n)) > 0;
      end

      [demodata]=demoddata2(1,2:nd*ml+1);

%**************** Bit Error Rate (BER) ****************

      noe2=sum(abs(data1-demodata));
         % sum: built in function
      nod2=length(data1);
         % length: built in function
      noe=noe+noe2;
      nod=nod+nod2;

      fprintf('%d\t%e\n',iii,noe2/nod2);
         % fprintf: built in function

end % for iii=1:nloop

%****************** Output result *******************

ber = noe/nod;
fprintf('%d\t%d\t%d\t%e\n',ebn0,noe,nod,noe/nod);
  % fprintf: built in function
fid = fopen('BERgmsk.dat','a');
fprintf(fid,'%d\t%e\t%f\t%f\t\n',ebn0,noe/nod,noe,nod);
  % fprintf: built in function
fclose(fid);

%******************** end of file *******************
```

Program 3.19

```
% Program 3-19
% gmsk_fading.m
%
% Simulation program to realize GMSK transmission system
% (under one path fading)
%
% Programmed by R. Sawai and H. Harada
%

%***************** Preparation part *****************

sr=256000.0;     % Symbol rate
ml=1;            % ml:Number of modulation levels
br=sr.*ml;       % Bit rate
nd = 100;        % Number of symbols that simulates in
                 % each loop
ebn0=15;         % Eb/N0
IPOINT=8;        % Number of oversamples
```

```
%*************** Filter initialization ***************

irfn=21;        % Number of taps
B=0.25*sr;
B2=0.6*sr;
[xh] = gaussf(B,irfn,IPOINT,sr,1);
  % Transmitter filter coefficients
[xh2] =gaussf(B2,irfn,IPOINT,sr,0);
  % Receiver filter coefficients

%*************** Fading initialization ***************

% If you use fading function "sefade", you can
% initialize all of parameters.
% Otherwise you can comment out the following
% initialization.
% The detailed explanations of all of variables are
% mentioned in Program 2-8.

% Time resolution

tstp=1/sr/IPOINT;

% Arrival time for each multipath normalized by tstp
% If you would like to simulate under one path fading
% model, you have only to set
% direct wave.

itau = [0];

% Mean power for each multipath normalized by direct
% wave.
% If you would like to simulate under one path fading
% model, you have only to set
% direct wave.
dlvl = [0];

% Number of waves to generate fading for each multipath.
% In normal case, more than six waves are needed to
% generate Rayleigh fading
n0=[6];

% Initial phase of delayed wave
% In this simulation one-path Rayleigh fading is
% considered.
th1=[0.0];

% Number of fading counter to skip
itnd0=nd*IPOINT*100;

% Initial value of fading counter
% In this simulation one-path Rayleigh fading is
% considered.
% Therefore one fading counter is needed.

itnd1=[1000];

% Number of direct wave + Number of delayed wave
% In this simulation one-path Rayleigh fading is
% considered
now1=1;

% Maximum Doppler frequency [Hz]
% You can insert your favorite value
fd=320;

% You can decide two modes to simulate fading by changing
% the variable flat
% flat     : flat fading or not
% (1-flat (only amplitude is fluctuated),0-normal(phase
% and amplitude are fluctuated))
flat =1;

%***************** START CALCULATION *****************

nloop=1000;     % Number of simulation loops

noe = 0;        % Number of error data
nod = 0;        % Number of transmitted data

for iii=1:nloop

%****************** Data generation ******************

      data1=rand(1,nd.*ml) > 0.5;
         % rand: built in function

%******************* GMSK Modulation *****************

      data11=2*data1-1;
      data2=oversamp(data11,length(data11),IPOINT);
      data3=conv(data2,xh);              % NEW for GMSK

      th=zeros(1,length(data3)+1);
      ich2=zeros(1,length(data3)+1);
      qch2=zeros(1,length(data3)+1);

      for ii=2:length(data3)+1
         th(1,ii)=th(1,ii-1)+pi/2*data3(1,ii-1)./IPOINT;
      end

      ich2=cos(th);
      qch2=sin(th);

%************** Attenuation Calculation **************
```

```
      spow=sum(ich2.*ich2+qch2.*qch2)/nd;
         % sum: built in function
      attn=0.5*spow*sr/br*10.^(-ebn0/10);
      attn=sqrt(attn);       % sqrt: built in function

%******************* Fading channel *******************

  % Generated data are fed into a fading simulator
  [ifade,qfade]=sefade(ich2,qch2,itau,dlvl,th1,n0,
      itnd1,now1,length(ich2),tstp,fd,flat);

  % Updata fading counter
      itnd1 = itnd1+ itnd0;

%*********** Add White Gaussian Noise (AWGN) ***********

      [ich3,qch3]= comb(ifade,qfade,attn);
         % add white gaussian noise

      [ich4,qch4] = compconv(ich3,qch3,xh2);

      syncpoint =irfn*IPOINT-IPOINT/2+1;
      ich5=ich4(syncpoint:IPOINT:length(ich4));
      qch5=qch4(syncpoint:IPOINT:length(qch4));

%***************** GMSK Demodulation *****************

      demoddata2(1,1)=-1;

      for k=3:2:nd*ml+1
         demoddata2(1,k)=ich5(1,k)*qch5(1,k-1)...
            *cos(pi*(k)) > 0;
      end

      for n=2:2:nd*ml+1
         demoddata2(1,n)=ich5(1,n-1)*qch5(1,n)...
            *cos(pi*(n)) > 0;
      end

      [demodata]=demoddata2(1,2:nd*ml+1);

%**************** Bit Error Rate (BER) ****************

      noe2=sum(abs(data1-demodata));
         % sum: built in function
      nod2=length(data1);
         % length: built in function
      noe=noe+noe2;
      nod=nod+nod2;

      fprintf('%d\t%e\n',iii,noe2/nod2);
         % fprintf: built in function
```

```
end % for iii=1:nloop

%******************* Output result ********************

ber = noe/nod;
fprintf('%d\t%d\t%d\t%e\n',ebn0,noe,nod,noe/nod);
  % fprintf: built in function
fid = fopen('BERgmskfad.dat','a');
fprintf(fid,'%d\t%e\t%f\t%f\t\n',ebn0,noe/nod,noe,nod);
  % fprintf: built in function
fclose(fid);

%********************* end of file *******************
```

Program 3.20

```
% Program 3-20
% gaussf.m
%
% Function to form Gaussian filter
%
% programmed by H. Harada
function [xh] = gaussf(B,irfn,ipoint,sr,ncc)

%*******************************************************

% irfn       : Number of symbols to use filtering
% ipoint     : Number of samples in one symbol
% sr      : symbol rate
% B       :  filter coefficients
% ncc     : 1 — transmitting filter  0 — receiving filter
%*******************************************************

point = ipoint;

tr = sr ;
n = ipoint .* irfn;
mid = ( n ./ 2 ) + 1;
fo=B/sqrt(2*log(2));

for i = 1 : n
  icon = i - mid;
  ym = icon;

 xt=1/2*(erf(-sqrt(2/log(2))*pi*B*(ym/ipoint-1/2)/tr)+...
  erf(sqrt(2/log(2))*pi*B*(ym/ipoint+1/2)/tr));

  if ncc == 0                % in the case of receiver
     xh( i ) = xt ;
  elseif ncc == 1            % in the case of transmitter
     xh( i ) = xt;
  else
```

```
      error('ncc error');
  end
  end

%********************* end of file *******************
```

Program 3.21

```
% Program 3-21
% qam16
%
% Simulation program to realize 16QAM transmission system
%
% Programmed by H. Harada and R. Funada
%

%****************** preparation part *****************

sr=256000.0;    % Symbol rate
ml=4;           % ml:Number of modulation levels
                % (BPSK:ml=1, QPSK:ml=2, 16QAM:ml=4)
br=sr .* ml;    % Bit rate
nd = 1000;      % Number of symbols that simulates in
                % each loop
ebn0=6;         % Eb/N0
IPOINT=8;       % Number of oversamples

%*************** Filter initialization ***************

irfn=21;        % Number of taps
alfs=0.5;       % Rolloff factor
[xh] = hrollfcoef(irfn,IPOINT,sr,alfs,1);
  %Transmitter filter coefficients
[xh2] = hrollfcoef(irfn,IPOINT,sr,alfs,0);
  %Receiver filter coefficients

%****************** START CALCULATION ****************

nloop=100;  % Number of simulation loops

noe = 0;    % Number of error data
nod = 0;    % Number of transmitted data

for iii=1:nloop

%******************* Data generation *****************

  data1=rand(1,nd*ml) > 0.5;

%****************** 16QAM Modulation *****************
```

```
      [ich,qch]=qammod(data1,1,nd,ml);
      [ich1,qch1]= compoversamp(ich,qch,length(ich),IPOINT);
      [ich2,qch2]= compconv(ich1,qch1,xh);

%************** Attenuation Calculation ***************

      spow=sum(ich2.*ich2+qch2.*qch2)/nd;
      attn=0.5*spow*sr/br*10.^(-ebn0/10);
      attn=sqrt(attn);

%******************* Fading channel ******************

  % Generated data are fed into a fading simulator
  % [ifade,qfade]=sefade(ich2,qch2,itau,dlvl,th1,n0,
  % itnd1,now1,length(ich2),tstp,fd,flat);

  % Updata fading counter
  % itnd1 = itnd1+ itnd0;

%*********** Add White Gaussian Noise (AWGN) ***********

      [ich3,qch3]= comb(ich2,qch2,attn);
         % add white gaussian noise
      [ich4,qch4]= compconv(ich3,qch3,xh2);

      sampl=irfn*IPOINT+1;
      ich5 = ich4(sampl:IPOINT:length(ich4));
      qch5 = qch4(sampl:IPOINT:length(ich4));

%***************** 16QAM Demodulation ****************

      [demodata]=qamdemod(ich5,qch5,1,nd,ml);

%**************** Bit Error Rate (BER) ***************

      noe2=sum(abs(data1-demodata));
      nod2=length(data1);
      noe=noe+noe2;
      nod=nod+nod2;

      fprintf('%d\t%e\n',iii,noe2/nod2);
end % for iii=1:nloop

%******************** Output result ******************

ber = noe/nod;
fprintf('%d\t%d\t%d\t%e\n',ebn0,noe,nod,noe/nod);
fid = fopen('BERqam.dat','a');
fprintf(fid,'%d\t%e\t%f\t%f\t\n',ebn0,noe/nod,noe,nod);
fclose(fid);

%********************* end of file *******************
```

Program 3.22

```
% Program 3-22
% qam16_fading
%
% Simulation program to realize 16QAM transmission system
% (under one path fading)
%
% Programmed by H. Harada and R. Funada
%

%***************** preparation part ******************

sr=256000.0;     % Symbol rate
ml=4;            % ml:Number of modulation levels
                 % (BPSK:ml=1, QPSK:ml=2, 16QAM:ml=4)
br=sr .* ml;     % Bit rate
nd = 100;        % Number of symbols that simulates in
                 % each loop
ebn0=15;         % Eb/N0
IPOINT=8;        % Number of oversamples

%*************** Filter initialization  **************

irfn=21;            % Number of taps
alfs=0.5;           % Rolloff factor
[xh] = hrollfcoef(irfn,IPOINT,sr,alfs,1);
  %Transmitter filter coefficients
[xh2] = hrollfcoef(irfn,IPOINT,sr,alfs,0);
  %Receiver filter coefficients

%*************** Fading initialization ***************

% If you use fading function "sefade", you can
% initialize all of parameters.
% Otherwise you can comment out the following
% initialization.
% The detailed explanations of all of variables are
% mentioned in Program 2-8.

% Time resolution

tstp=1/sr/IPOINT;

% Arrival time for each multipath normalized by tstp
% If you would like to simulate under one path fading
% model, you have only to set
% direct wave.

itau = [0];

% Mean power for each multipath normalized by direct
% wave.
% If you would like to simulate under one path fading
% model, you have only to set
% direct wave.
dlvl = [0];

% Number of waves to generate fading for each multipath.
% In normal case, more than six waves are needed to
% generate Rayleigh fading
n0=[6];

% Initial phase of delayed wave
% In this simulation one-path Rayleigh fading is
% considered.
th1=[0.0];

% Number of fading counter to skip
itnd0=nd*IPOINT*100;

% Initial value of fading counter
% In this simulation one-path Rayleigh fading is
% considered.
% Therefore one fading counter is needed.

itnd1=[1000];

% Number of direct wave + Number of delayed wave
% In this simulation one-path Rayleigh fading is
% considered
now1=1;

% Maximum Doppler frequency [Hz]
% You can insert your favorite value
fd=160;

% You can decide two modes to simulate fading by changing
% the variable flat
% flat     : flat fading or not
% (1-flat (only amplitude is fluctuated),0-normal(phase
% and amplitude are fluctuated))
flat =1;

%***************** START CALCULATION ******************

nloop=1000;    % Number of simulation loops

noe = 0;       % Number of error data
nod = 0;       % Number of transmitted data

for iii=1:nloop

%******************* Data generation ******************

  data1=rand(1,nd*ml) > 0.5;
```

```
%***************** 16QAM Modulation *******************

      [ich,qch]=qammod(data1,1,nd,ml);
      [ich1,qch1]= compoversamp(ich,qch,length(ich),IPOINT);
      [ich2,qch2]= compconv(ich1,qch1,xh);

%*************** Attenuation Calculation **************

      spow=sum(ich2.*ich2+qch2.*qch2)/nd;
      attn=0.5*spow*sr/br*10.^(-ebn0/10);
      attn=sqrt(attn);

%******************* Fading channel *******************

   % Generated data are fed into a fading simulator
   [ifade,qfade,ramp]=sefade(ich2,qch2,itau,dlvl,th1,n0,...
      itnd1,now1,length(ich2),tstp,fd,flat);

      % Updata fading counter
      itnd1 = itnd1+ itnd0;

%*********** Add White Gaussian Noise (AWGN) ***********

      [ich3,qch3]= comb(ifade,qfade,attn);
         % add white gaussian noise

%***** Compensate the fluctuation of fading by ramp ****

      ich3=ich3./ramp(1:length(ramp));
      qch3=qch3./ramp(1:length(ramp));

      [ich4,qch4]= compconv(ich3,qch3,xh2);

      sampl=irfn*IPOINT+1;
      ich5 = ich4(sampl:IPOINT:length(ich4));
      qch5 = qch4(sampl:IPOINT:length(ich4));

%***************** 16QAM Demodulation *****************

      [demodata]=qamdemod(ich5,qch5,1,nd,ml);

%**************** Bit Error Rate (BER) ****************

      noe2=sum(abs(data1-demodata));
      nod2=length(data1);
      noe=noe+noe2;
      nod=nod+nod2;

      fprintf('%d\t%e\n',iii,noe2/nod2);
end % for iii=1:nloop

%******************** Output result *******************
```

```
ber = noe/nod;
fprintf(‘%d\t%d\t%d\t%e\n’,ebn0,noe,nod,noe/nod);
fid = fopen(‘BERqamfad.dat’,‘a’);
fprintf(fid,‘%d\t%e\t%f\t%f\t\n’,ebn0,noe/nod,noe,nod);
fclose(fid);

%********************* end of file ********************
```

Program 3.23

```
% Program 3-23
% qammod.m
%
% This function is used for Gray coding of 16QAM modulation
%
% programmed by R. Funada and H. Harada
%

function [iout,qout]=qammod(paradata,para,nd,ml)

%********************* variables ********************

% paradata : input data (para-by-nd matrix)
% iout :output Ich data
% qout :output Qch data
% para   : Number of parallel channels
% nd : Number of data
% ml : Number of modulation levels
% (QPSK -2  16QAM - 4)
% ****************************************************

% The constellation power

k=sqrt(10);
iv=[-3 -1 3 1];

m2=ml/2;
count2=0;

for ii=1 : nd
      isi = zeros(para,1);
      isq = zeros(para,1);

      for jj=1 : ml

          if jj <= m2
          isi = isi +2.^( m2- jj ).*paradata((1:para),...
              count2+jj);
          else
          isq = isq +2.^( ml- jj ).*paradata((1:para),...
              count2+jj);
          end
      end
```

```
      iout((1:para),ii) = iv(isi+1)./k;
      qout((1:para),ii) =iv(isq+1)./k;

      count2=count2+ml;

end

%********************* end of file ********************
```

Program 3.24

```
% Program 3-24
% qamdemod.m
%
% Function to decode 16QAM modulation
%
% programmed by R. Funada and H. Harada
%

function [demodata]=qamdemod(idata,qdata,para,nd,ml)

%********************* variables ********************

% idata :input Ich data
% qdata :input Qch data
% demodata: demodulated data (para-by-nd matrix)
% para   : Number of parallel channels
% nd : Number of data
% ml : Number of modulation levels
% (QPSK -> 2  16QAM -> 4)

% ****************************************************

k=sqrt(10);
idata=idata.*k;
qdata=qdata.*k;
demodata=zeros(para,ml*nd);

m2=ml/2;
count2=0;

for ii = 1:nd

      a=1;
      b=1;
      i_lngth=0;
      q_lngth=0;

      for jj= 1:m2

          if jj ~= 1
```

```
            if demodata((1:para),jj-1+count2)==1
               a=-a;
            end

            if demodata((1:para),m2+jj-1+count2)==1
               b=-b;
            end

            i_lngth=i_lngth+i_plrty.*2.^(m2-jj+1);
            q_lngth=q_lngth+q_plrty.*2.^(m2-jj+1);
         end

         if idata((1:para),ii >= i_lngth
            demodata((1:para),jj+count2)= a >= 0;
            i_plrty=1;
         else
            demodata((1:para),jj+count2)= a <= 0;
            i_plrty=-1;
         end

         if qdata((1:para),ii) >= q_lngth
            demodata((1:para),m2+jj+count2)= b => 0;
            q_plrty=1;
         else
            demodata((1:para),m2+jj+count2)= b <= 0;
            q_plrty=-1;
         end

      end % for jj= 1:m2

      count2=count2+m1;

end % for ii = 1:nd

%********************* end of file ********************
```

4

Orthogonal Frequency Division Multiplexing (OFDM) Transmission Technology

4.1 Introduction

As shown in Chapter 2, a mobile radio channel is characterized by a multipath fading environment. In other words, the signal offered to the receiver contains not only a direct line-of-sight radio wave, but also a large number of reflected radio waves that arrive at the receiver at different times. Delayed signals are a result of reflections from terrain features such as trees, hills, mountains, vehicles, or buildings. These reflected delayed waves interfere with the direct wave and cause *intersymbol interference* (ISI), which causes significant degradation of the network performance. A wireless network must be designed in such a way as to minimize these adverse effects.

Recently, there have been increasing attempts to extend the services available through wired public telecommunication networks to mobile/movable nonwired telecommunication-network users. These attempts have raised expectations that the development of broadband mobile communications is likely. For broadband multimedia mobile communication systems, it is necessary to use high-bit-rate transmission of at least several megabits per second. However, if digital data are transmitted at a rate of several megabits per second, the delay time of the delayed waves exceeds 1 symbol time. Because the delayed waves interfere with other symbols, the effects of this interference must be eliminated in the received signal. There are several ways to achieve this goal. Using adaptive equalization techniques at the receiver is one way to equalize the received signal. However, in practice, achieving this equalization at several megabits per second with compact and low-cost hardware is quite difficult.

To overcome such a multipath fading environment and achieve a *wireless broadband multimedia communication system* (WBMCS), it is possible to use an OFDM transmission scheme, which is described in this chapter. OFDM is based on a parallel data transmission scheme [1–3] that reduces the effects of multipath fading and renders complex equalizers unnecessary. The history of OFDM transmission technologies was described in detail in our previous book [3].

OFDM is expected to be used in future broadcasting and *wireless LAN* (WLAN) systems. For example, ETSI BRAN in Europe [4], IEEE802.11 in the United States [5], and ARIB MMAC [6] in Japan have already adopted the OFDM transmission technology as a physical layer for future broadband WLAN systems. In the standardization body, several telecommunications systems have already been compared by using computer simulation. This section explains how to generate an OFDM signal, how to configure an OFDM transmitter and an OFDM receiver, and how to evaluate the BER and *packet error rate* (PER) performance by computer simulation.

The chapter is organized as follows: First, the concept behind the parallel data transmission scheme is described in Section 4.2. Section 4.3 presents the concept of OFDM transmission technology and the configuration of an OFDM transmitter and receiver. Section 4.4 describes the method to generate an OFDM signal and to configure the OFDM transmitter and receiver by computer simulation. Section 4.5 presents a simulation method for an OFDM system that is matched to a standard OFDM-based WLAN system. Section 4.6 concludes with a brief summary. All programs related to the chapter are given in Appendix 4A and in the CD-ROM that accompanies the book.

4.2 Concept of the Parallel Transmission Scheme

The multipath fading environment in which not only a direct transmission signal but also many reflected signals arrive at the receiver at different times in the time domain is characterized by a channel impulse response, which includes the information about the relative time when the delayed signal arrived at the receiver, the power of the signal, and its phase as compared to the power and phase of the direct wave. Figure 4.1 shows a typical impulse response of multipath fading in the time and frequency domains. From the time-domain point of view, many signals with different arrival times, signal power, and phases are received at the receiver. From the frequency-domain point of view, on the other hand, the multipath fading environment is characterized by the enhancement of some frequencies and the attenuation of others. If there is mobile reception, then the relative power levels and attenuations of various reception paths will change with time. A narrowband signal will vary in quality as the peaks and the frequency response move around in the frequency domain. There will also be a noticeable variation in the phase response, which will affect all systems using the phase as a means of signaling.

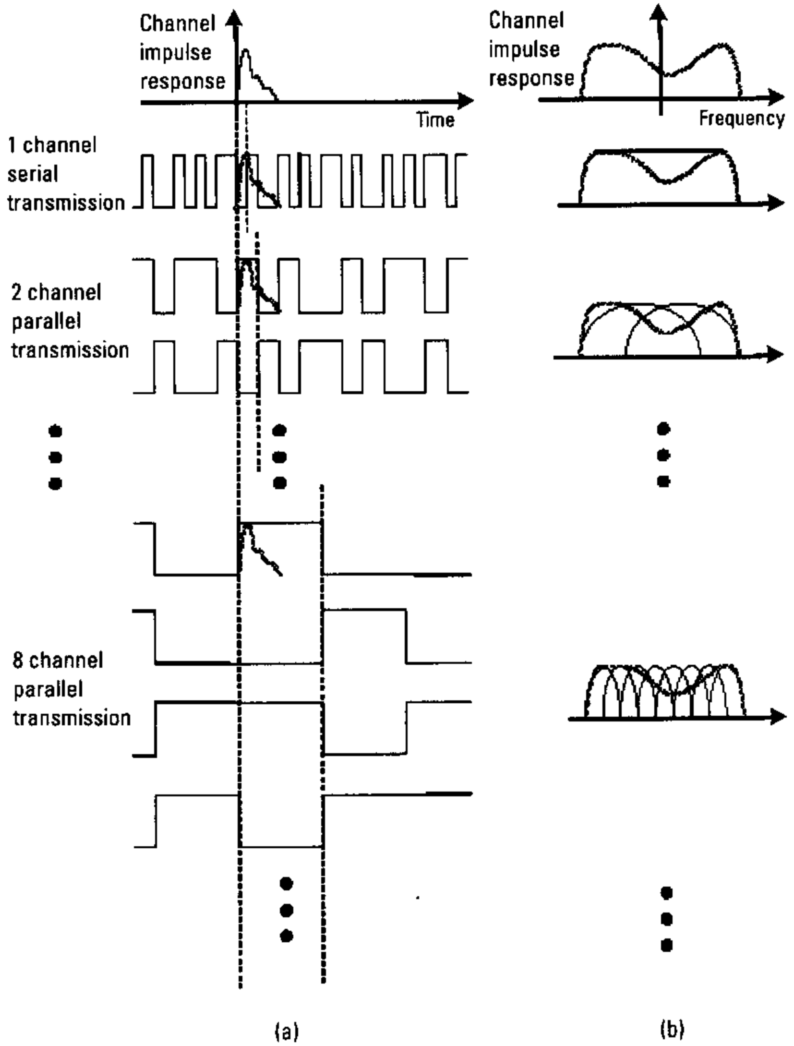

Channel impulse response
Time
Channel impulse response
Frequency
1 channel serial transmission
2 channel parallel transmission
8 channel parallel transmission
(a)
(b)

Figure 4.1 Typical impulse response of multipath fading: (a) time domain and (b) frequency domain.

Let us consider the situation where single-carrier serial high-speed wireless data is transmitted in a multipath fading environment. As mentioned in Section 4.1, if digital data are transmitted at a rate of several megabits per second, and the maximum delay time of delayed waves caused by multipath fading is larger than 1 μs, the maximum delay time of the delayed waves is greater than 1 symbol time. Figure 4.1 illustrates the waveforms of a single-carrier serial high-speed wireless-data-transmission scheme in the time domain and the spectrum when the maximum delay time of the delayed waves is greater than 1 symbol time. Both the waveforms and the spectrum are distorted, and we need to equalize the distorted signal. One way to equalize the signal is by using adaptive equalization techniques that estimate the channel impulse response at the receiver and multiply the complex conjugate of the estimated impulse response by the received data signal at the receiver. However, there are practical difficulties associated with operating this equalization at several megabits per second with high-speed, compact, and low-cost hardware, because, as shown in Figure 4.1, if we recover the transmitted data from the received data, we must store several successive symbols to equalize the received data sequentially. We therefore look for other solutions.

From the frequency-domain point of view, when a transmitted signal suffers from multipath fading, some part of the signal may suffer from constructive interference and be enhanced in level, whereas some other parts of the signal may suffer from destructive interference and be attenuated, sometimes to the point of extinction. In general, frequency bands that are close together will suffer from the same variation in the signal strength, which is well correlated. The width of the frequency bands that have high correlation value is called the coherent bandwidth [3]. For a narrowband signal, distortion is usually minimized if the bandwidth of the signal is less than the coherent bandwidth. There is, however, a significant chance that the signal will be subjected to severe attenuation in some occasions. A signal that occupies a bandwidth greater than the coherent bandwidth will be subjected to more distortion, but will suffer from less variation in the total received power, even if it is subjected to significant levels of multipath fading.

To combat the problems caused by the multipath fading environment and achieve broadband mobile communications, it is necessary to use parallel transmission, in which the transmitted high-speed data is converted to slow parallel data in several channels. These data are multiplexed using several multiplexing techniques to distinguish between the subchannels.

Figure 4.1 also shows the effect of the parallel transmission scheme. For a given overall data rate, increasing the number of parallel transmission channels reduces the data rate that each individual subchannel must convey, or, in other words, lengthens the symbol period. As a result, the delay time of the delayed waves is suppressed to within 1 symbol time.

To distinguish between the subchannels, *frequency-division multiplexing* (FDM) and *code-division multiplex* (CDM) are often used. In some cases, the first method is called *multicarrier transmission*, and the second method is called *multicode transmission*. Figure 4.2 illustrates the configuration of the two parallel transmission schemes. OFDM is a multicarrier transmission technique—and the most efficient one. Section 4.3 describes the concept of OFDM transmission.

4.3 Concept of OFDM Transmission Technology

For the history and principle of the OFDM transmission system, we refer the reader to our previous book [3]. In this section, we will focus on the development and application of the OFDM transmission scheme.

4.3.1 Transmitter Configuration

Figure 4.3(a) illustrates the configuration of an OFDM transmitter. In the transmitter, the transmitted high-speed data is first converted into parallel data

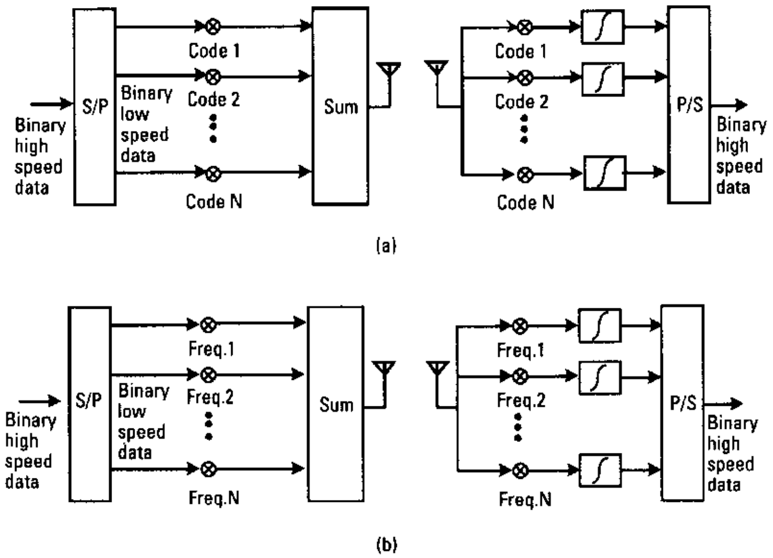
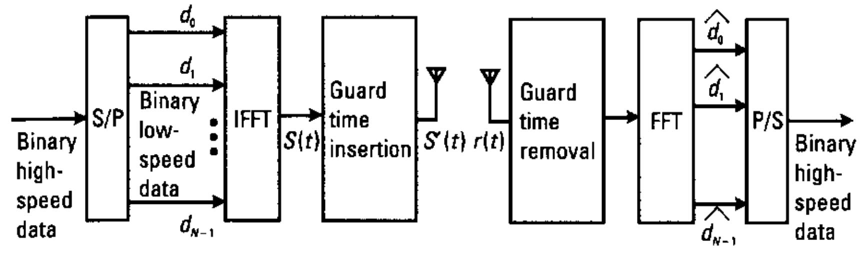


Figure 4.2 Parallel-transmission system: (a) multicode transmission scheme and (b) multicarrier transmission scheme.



(a) (b)

Figure 4.3 OFDM radio transmission system: (a) transmitter and (b) receiver.

of N subchannels. Then, the transmitted data of each parallel subchannel is modulated by PSK-based modulation discussed in Chapter 3. Consider a quadrature-modulated data sequence of the N channels (d_0, d_1, d_2, … , d_{N-1}), where each d_n is a complex number, $d_n = d_{I_n} + jd_{Q_n}$ (j = complex number), and d_{I_n} and d_{Q_n} are {1, −1} in QPSK, and {±1, ±3} in 16-QAM that has been discussed in Chapter 3. These modulated data are fed into an *inverse fast Fourier transform* (IFFT) circuit, and an OFDM signal is generated. The transmitted data is given by

$$\begin{aligned} s(t) &= \sum_{k=-\infty}^{\infty}\sum_{i=0}^{N-1} d_i(k)\exp\left(j2\pi f_i(t-kT_s)\right)f(t-kT_s) \\ &= \sum_{k=-\infty}^{\infty}\sum_{i=0}^{N-1}\left(d_{I_i}(k)+jd_{Q_i}(k)\right)\left(\cos\left(2\pi f_i(t-kT_s)\right)+j\sin\left(2\pi f_i(t-kTs)\right)\right)f(t-kT_s) \\ &= \sum_{k=-\infty}^{\infty}\sum_{i=0}^{N-1}\left(d_{I_i}(k)\cos\left(2\pi f_i(t-kT_s)\right)-d_{Q_i}(k)\sin\left(2\pi f_i(t-kTs)\right)\right)f(t-kT_s) \\ &\quad +j\sum_{k=-\infty}^{\infty}\sum_{i=0}^{N-1}\left(d_{I_i}(k)\sin\left(2\pi f_i(t-kT_s)\right)+d_{Q_i}(k)\cos\left(2\pi f_i(t-kT_s)\right)\right)f(t-kT_s) \end{aligned} \tag{4.1}$$

where T_S is the symbol duration of the OFDM signal, and f_i (i = 0, 1, 2, …) is the frequency of the ith subcarrier given by

$$f_i = f_0 + \frac{i}{T_s} \tag{4.2}$$

Here, $f(t)$ is the pulse waveform of each of the symbols and it is defined as

$$f(t) = \begin{cases} 1 & (0 \le t \le T_s) \\ 0 & (\text{otherwise}) \end{cases} \tag{4.3}$$

Figure 4.4 illustrates the waveforms of a real part and an imaginary part of an OFDM signal in each subchannel when i = 0, 1, 2, …, N − 1. As shown in Figure 4.4, the OFDM signal includes many carrier signals with their own frequencies. This OFDM signal is fed into a guard time insertion circuit to reduce ISI.

We will now describe the guard interval. The orthogonality of subchannels in OFDM can be maintained, and individual subchannels can be completely separated by using an FFT circuit at the receiver when there are no ISI and *intercarrier interference* (ICI) introduced by transmission channel distortion. In practice, however, these conditions cannot be obtained. Because the spectra of an OFDM signal are not strictly band-limited, the distortion, due to multipath fading, causes each subchannel to spread the power into the adjacent channels. Moreover, a delayed wave with the delay time larger than 11 symbol time contaminates the next symbol. To reduce the distortion, a simple solution is to increase the symbol duration or the number of carriers. However, this method may be difficult to implement in terms of carrier stability against Doppler frequency and FFT size.

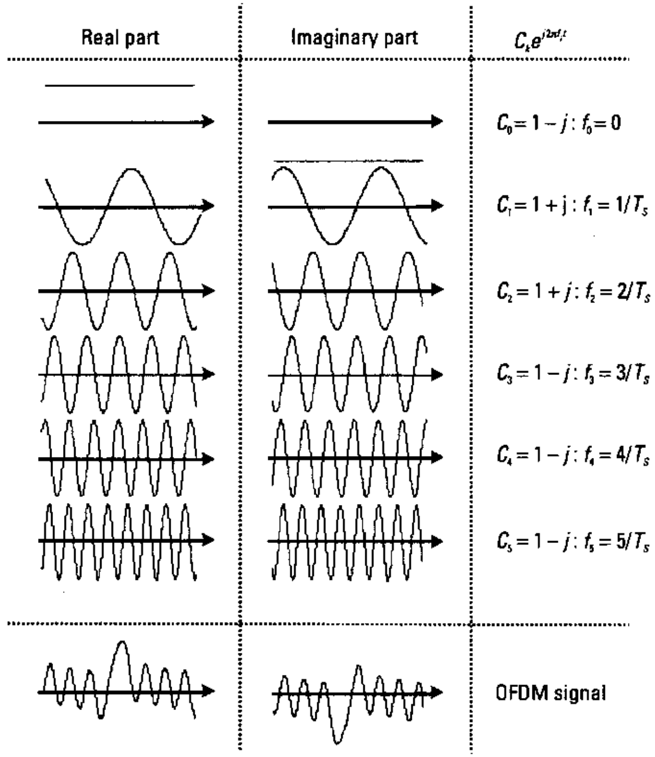

Real part
Imaginary part
$C_k e^{j2\pi f_i t}$
$C_0 = 1 - j : f_0 = 0$
$C_1 = 1 + j : f_1 = 1/T_S$
$C_2 = 1 + j : f_2 = 2/T_S$
$C_3 = 1 - j : f_3 = 3/T_S$
$C_4 = 1 - j : f_4 = 4/T_S$
$C_5 = 1 - j : f_5 = 5/T_S$
OFDM signal

Figure 4.4 OFDM transmission signal in each subcarrier.

One way to eliminate ISI is to create a cyclically extended guard interval [2], where each OFDM symbol is preceded by a periodic extension of the signal itself. The total symbol duration is $T_{\text{total}} = T_g + T_s$, where T_g is the guard interval. Figure 4.5 shows a typical guard interval. Each symbol is made of two parts. The whole signal is contained in the active symbol, the last part of which is also repeated at the start of the symbol and is called a guard interval. When the guard interval is longer than the channel impulse response, or the multipath delay, the effect of ISI can be eliminated. However, the ICI, or in-band fading, still exists. The ratio of the guard interval to the useful symbol duration is application-dependent. Because the insertion of a guard interval will reduce the data throughput, T_g is usually smaller than $T_s/4$.

After the insertion of a guard interval, the OFDM signal is given by

$$s'(t) = \sum_{k=-\infty}^{\infty} \sum_{i=0}^{N-1} d_i(k) \exp\left(j2\pi f_i\left(t - kT_{\text{total}}\right)\right) f'\left(t - kT_{\text{total}}\right) \quad (4.4)$$

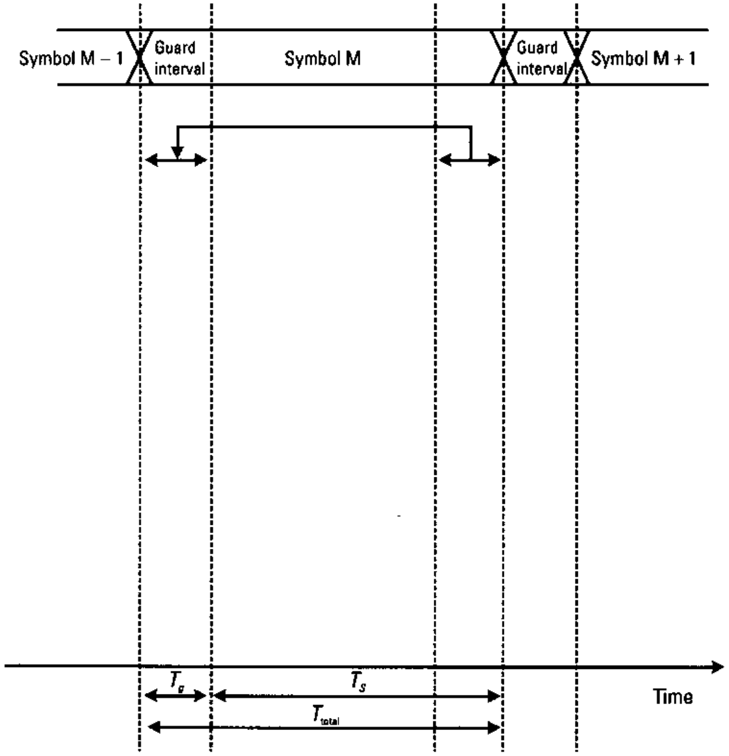


Figure 4.5 Guard interval insertion.

where $f'(t)$ is the modified pulse waveform of each symbol defined as

$$f'(t) = \begin{cases} 1 & \left(-T_g \le t \le T_s\right) \\ 0 & \left(t < -T_g, t > T_s\right) \end{cases} \quad (4.5)$$

The OFDM signal is transmitted to the receiver; however, the transmitted data, $s'(t)$, is contaminated by multipath fading and AWGN. At the receiver, the received signal is given by

$$r(t) = \int_0^{\infty} h(\tau, t) s(t - \tau) d\tau + n(t) \quad (4.6)$$

where $h(\tau, t)$ is the impulse response of the radio channel at time t, and $n(t)$ is the complex AWGN.

4.3.2 Receiver Configuration

Figure 4.3(b) shows a block diagram of an OFDM receiver. At the receiver, received signal $r(t)$ is filtered by a bandpass filter, which is assumed to have sufficiently wide passband to introduce only negligible distortion in the signal. An orthogonal detector is then applied to the signal where the signal is downconverted to the IF band. Then, an FFT circuit is applied to the signal to obtain Fourier coefficients of the signal in observation periods $[iT_{\text{total}}, iT_{\text{total}} + T_s]$. The output, $\hat{d}_i(k)$, of the FFT circuit of the ith OFDM subchannel is given by

$$\hat{d}_i(k) = \frac{1}{T_s} \int_{kT_{\text{total}}}^{T_s + kT_{\text{total}}} r(t) \exp\left(-j2\pi f_i\left(t - kT_{\text{total}}\right)\right) dt \quad (4.7)$$

If we can estimate the characteristics of the delayed wave, $\hat{h}_i(k)$, in a multipath fading environment, we can equalize the received data as follows:

$$\hat{\hat{d}}_i(k) = \frac{\hat{h}_i * (k)}{\hat{h}_i(k) \hat{h}_i * (k)} \hat{d}_i(k) \quad (4.8)$$

where * indicates the complex conjugate.

By comparing d_k and $\hat{\hat{d}}_i(k)$, we can calculate the BER performance. The BER depends on the level of the receiver's noise. In OFDM transmission, the orthogonality is preserved, and the BER performance depends on the modulation scheme in each subchannel. Therefore, if BPSK is used, the BER under AWGN and a one-path Rayleigh fading channel is equal to the theoretical ones shown in (3.17) and (3.18), respectively. For the other modulation schemes, the dependence of the BER on the modulation scheme is shown in Table 4.1. Section 4.4 presents a configuration method of the OFDM transmitter and receiver by computer simulation.

4.4 Configuration by Using Computer Simulation

This section calculates the BER of an OFDM system by using a simple computer-simulation program. A block diagram of the simulation is shown in Figure 4.6. All simulation programs used for this chapter are listed in Table 4.2. In particular, in this section, we used Programs 4.1–4.3. The main program used was Program 4.1. In Program 4.1, we first chose the common variables we used in the main program.

Table 4.1
BER Performance in Conventional Modulation Schemes

Modulation Scheme	**Theoretical BER**	
	AWGN	One-Path Rayleigh Fading
BPSK	$\frac{1}{2}\text{erfc}\left(\sqrt{E_b/N_0}\right)$	$\frac{1}{2}\left[1-\frac{1}{\sqrt{1+\frac{1}{E_b/N_0}}}\right]$
QPSK	$\frac{1}{2}\text{erfc}\left(\sqrt{E_b/N_0}\right)$	$\frac{1}{2}\left[1-\frac{1}{\sqrt{1+\frac{1}{E_b/N_0}}}\right]$
16-QAM	$\frac{3}{8}\text{erfc}\left(\sqrt{\frac{2}{5}E_b/N_0}\right)-\frac{9}{24}\text{erfc}^2\left(\sqrt{\frac{2}{5}E_b/N_0}\right)$	$\frac{3}{8}\left[1-\frac{1}{\sqrt{1+5/(2E_b/N_0)}}\right]$
64-QAM	$\frac{7}{24}\text{erfc}\left(\sqrt{\frac{1}{7}E_b/N_0}\right)-\frac{49}{384}\text{erfc}^2\left(\sqrt{\frac{1}{7}E_b/N_0}\right)$	$\frac{7}{24}\left[1-\frac{1}{\sqrt{1+7/(E_b/N_0)}}\right]$
256-QAM	$\frac{15}{64}\text{erfc}\left(\sqrt{\frac{4}{85}E_b/N_0}\right)-\frac{225}{2{,}048}\text{erfc}^2\left(\sqrt{\frac{4}{85}E_b/N_0}\right)$	$\frac{15}{64}\left[1-\frac{1}{\sqrt{1+85/(4E_b/N_0)}}\right]$

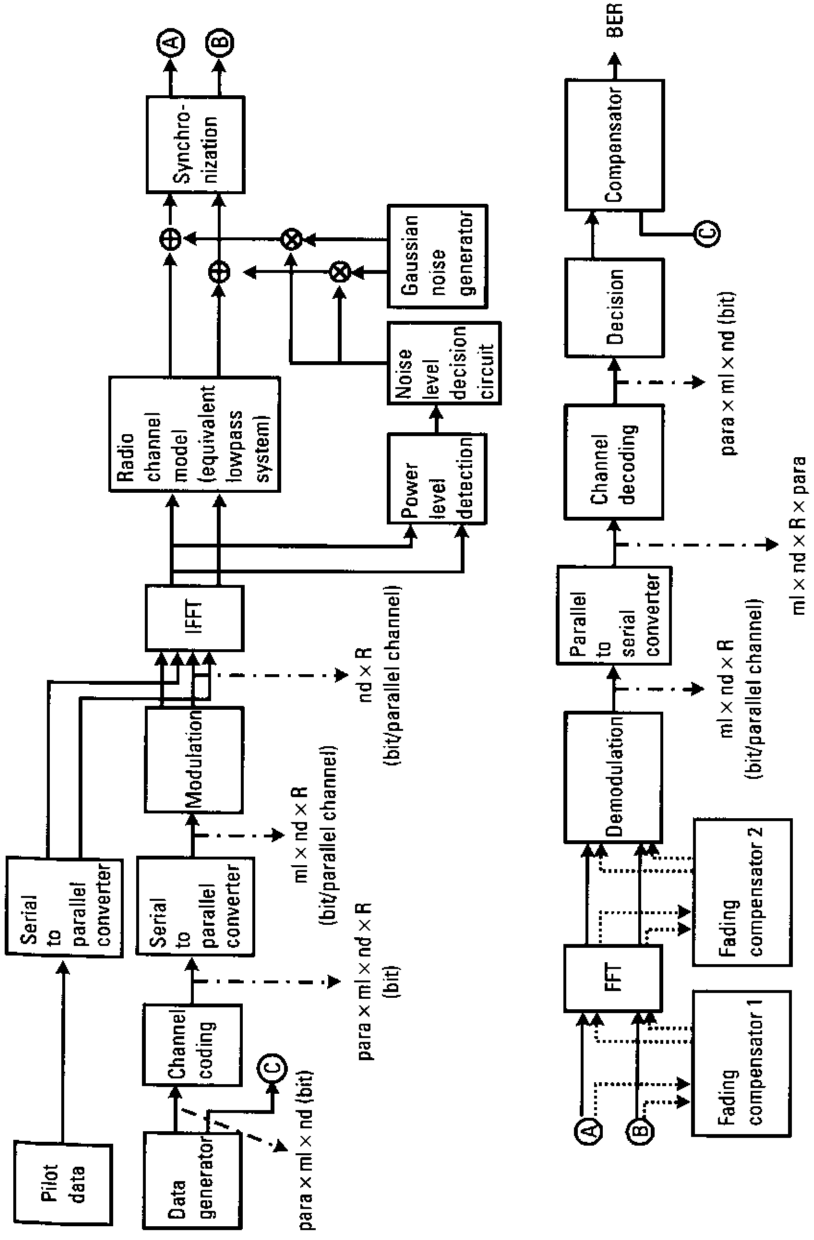


Figure 4.6 Computer simulation to calculate the BER of an OFDM system.

Table 4.2
Parameters in the OFDM Simulation

Name of Function	Function	Input Data	Output
ofdm.m (Program 4.1) ofdm_fading.m (Program 4.2)	Calculate BER performance under AWGN (Program 4.1) and one-path Rayleigh fading channel (Program 4.2)	None	ber: bit error rate performance
giins.m (Program 4.3)	Guard interval insertion	idata: input I-channel sequence qdata: input Q-channel sequence fftlen: the number of FFT point gilen: the length of guard interval nd: the number of symbol	iout: I-channel data inserted guard interval qout: Q-channel data inserted guard interval
girem.m (Program 4.4)	Guard interval removal	idata: input I-channel sequence qdata: input Q-channel sequence fftlen2: the number of point per one symbol gilen: the length of guard interval nd: the number of symbol	iout: I-channel data removed guard interval qout: Q-channel data removed guard interval
ofdmda.m (Program 4.5)	Calculate BER performance under AWGN and one-path Rayleigh fading channel	None	ber: bit error rate performance per: packet error rate performance
crmapping.m (Program 4.6)	Set data on subcarrier	idata: input I-channel data sequence qdata: input Q-channel data sequence fftlen: the number of IFFT point nd: the number of symbol	iout: mapped data in I-channel qout: mapped data in Q-channel
crdemapping.m (Program 4.7)	Separate data from carrier	idata: input I-channel data sequence qdata: input Q-channel data sequence fftlen: the number of FFT point nd: the number of symbol	iout: demapped data in I-channel qout: demapped data in Q-channel

Table 4.2 (continued)

Name of Function	Function	Input Data	Output
ofdmce.m (Program 4.8)	Calculate BER performance under AWGN and one-path Rayleigh fading channel	None	ber: bit error rate performance per: packet error rate performance
ofdmci.m (Program 4.9)	Calculate BER performance under AWGN and one-path Rayleigh fading channel	None	ber: bit error rate performance per: packet error rate performance
interwave.m (Program 4.10)	Form wave of interference	ci: carrier to interference power ratio spow: power of carrier ml: modulation level in interference nsamp: number of samples tstp: time resolution fading para: fading parameter	iout: interference of I-channel qout: interference of Q-channel

```
para=128;         % Number of parallel channel
fftlen=128;       % FFT length
noc=128;          % Number of carrier
nd=6;             % Number of OFDM symbol for one loop
ml=2;             % Modulation level: QPSK
sr=250000;        % Symbol rate
br=sr.*ml;        % Bit rate per carrier
gilen=32;         % Length of guard interval
ebn0=100          % ebn0 : Eb/No
```

As shown in Figure 4.6 and the above parameters using the program, we simulated an OFDM system with 128 subcarriers, a 4-μs symbol time (*tstp* = 1./sr), and a 1/4 *tstp* guard interval. We then defined the variables for the simulation, with QPSK used as a modulation technique in each channel.

```
nloop=100;    % Number of simulation loops
noe=0;        % Number of error data
nod=0;        % Number of transmitted data
eop=0;        % Number of error packet
nop=0;        % Number of transmitted packet
```

After defining all the variables, we began our simulation to obtain BER and PER performance. First, we generated random serial data of 0 and 1 consisting of a 1-by-para*nd*ml vector. We called the vector "seridata."

```
seridata=rand(1,para*nd*ml)>0.5;
```

The serial data vector, "seridata," was converted into a parallel data vector, "paradata," consisting of a para-by-nd*ml vector to transmit the data in parallel in order to enable parallel transmission with 128 subchannels where each channel was using a QPSK modulation scheme.

```
paradata=reshape(seridata,para,nd*ml);Next, the vector
"paradata" was fed into the mapping circuit. In the
circuit, the parallel data were converted into modulated
parallel data of two channels, Ich and Qch by a predefined
mapping method.

[ich,qch]=qpskmod(paradata,para,nd,ml);
```

The frame format of the simulation model is configured as shown in Figure 4.7. The transmission data in the I-channel and Q-channel are shown in Figure 4.8. Then, these data were increased kmod times to normalize the data as follows.

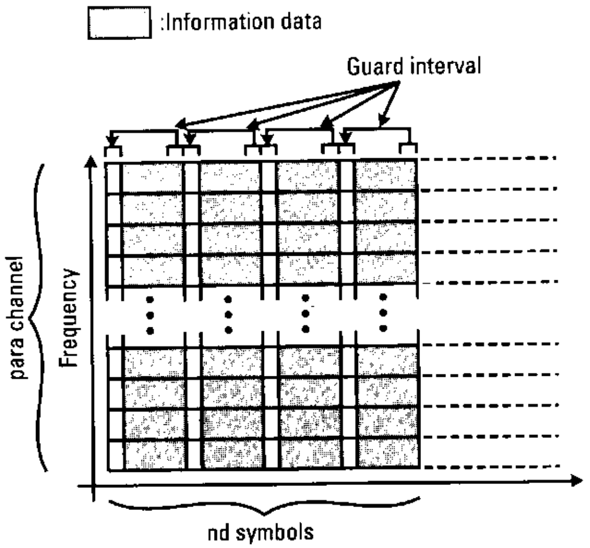


Figure 4.7 Frame format of the simulation model.

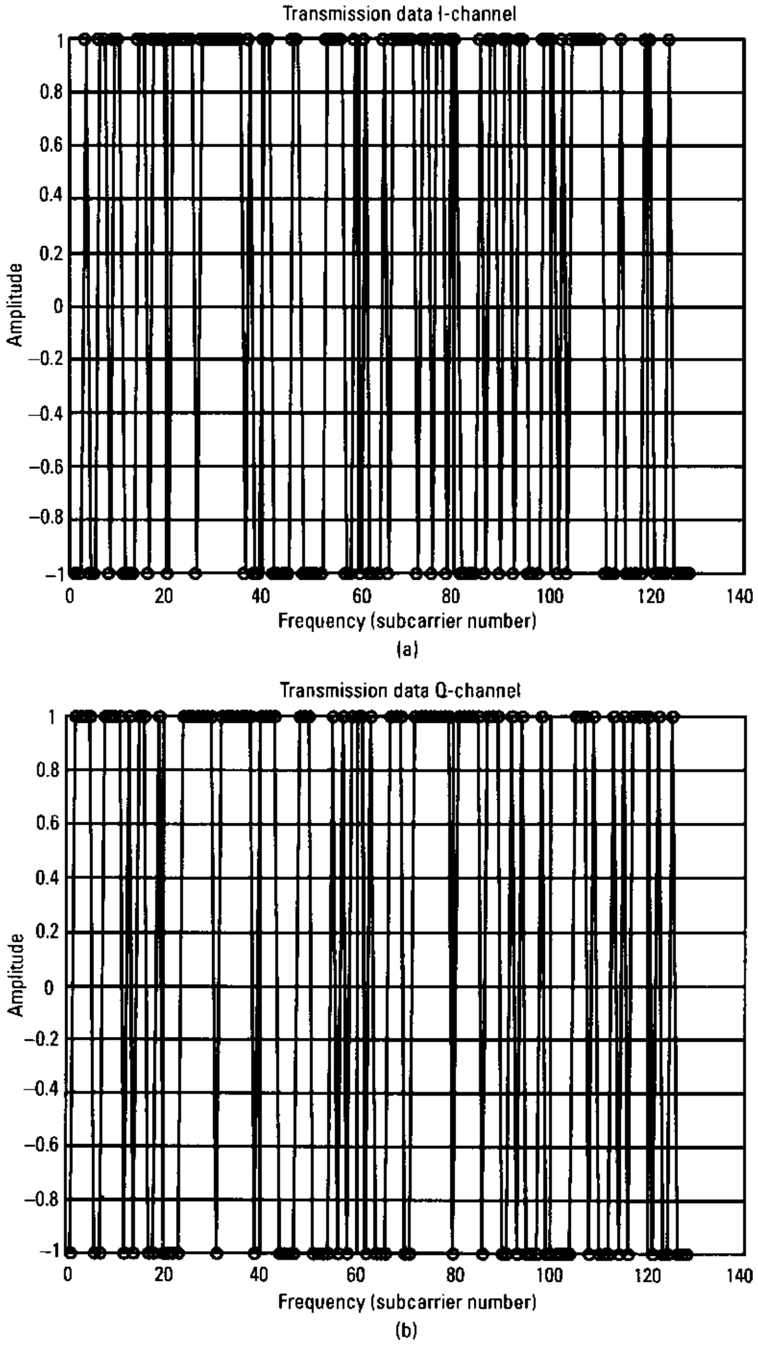


Figure 4.8 (a) Transmission data in I-channel and (b) transmission data in Q-channel.

```
kmod=1/sqrt(2);
ich1=ich.*kmod;
qch1=qch.*kmod;
```

After the mapping, these parallel data on the frequency axis were fed into the IFFT circuit. In the circuit, the parallel data were converted into serial data on the time axis by using OFDM

```
x=ich1+qch1.*i;
y=ifft(x);        %  ifft : built-in function
ich2=real(y);     %  real : built-in function
qch2=imag(y);     %  imag : built-in function
```

The input and output are shown in Figure 4.9. Then, ich2 and qch2, guard intervals, were inserted to eliminate ISI caused by multipath fading.

```
[ich3,qch3]= giins(ich2,qch2,fftlen,gilen,nd);
fftlen2=fftlen+gilen;
```

At this point, we defined fftlen2 as the length of a symbol including the guard interval. After that, the filtered signal was transmitted to the air. The OFDM-modulated data in the I-channel and Q-channel are shown in Figure 4.10. Then, the transmitted signal passed through the radio channel (equivalent lowpass system) and was transmitted to the receiver.

At the receiver, the received signal was first contaminated by AWGN. The noise function was already discussed in Section 3.2.3 where we introduced

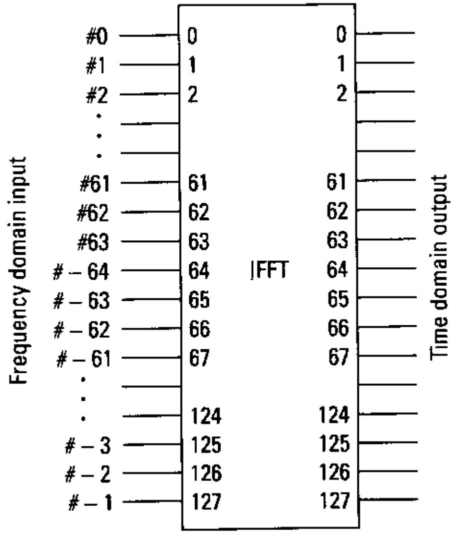


Figure 4.9 Input and output of IFFT.

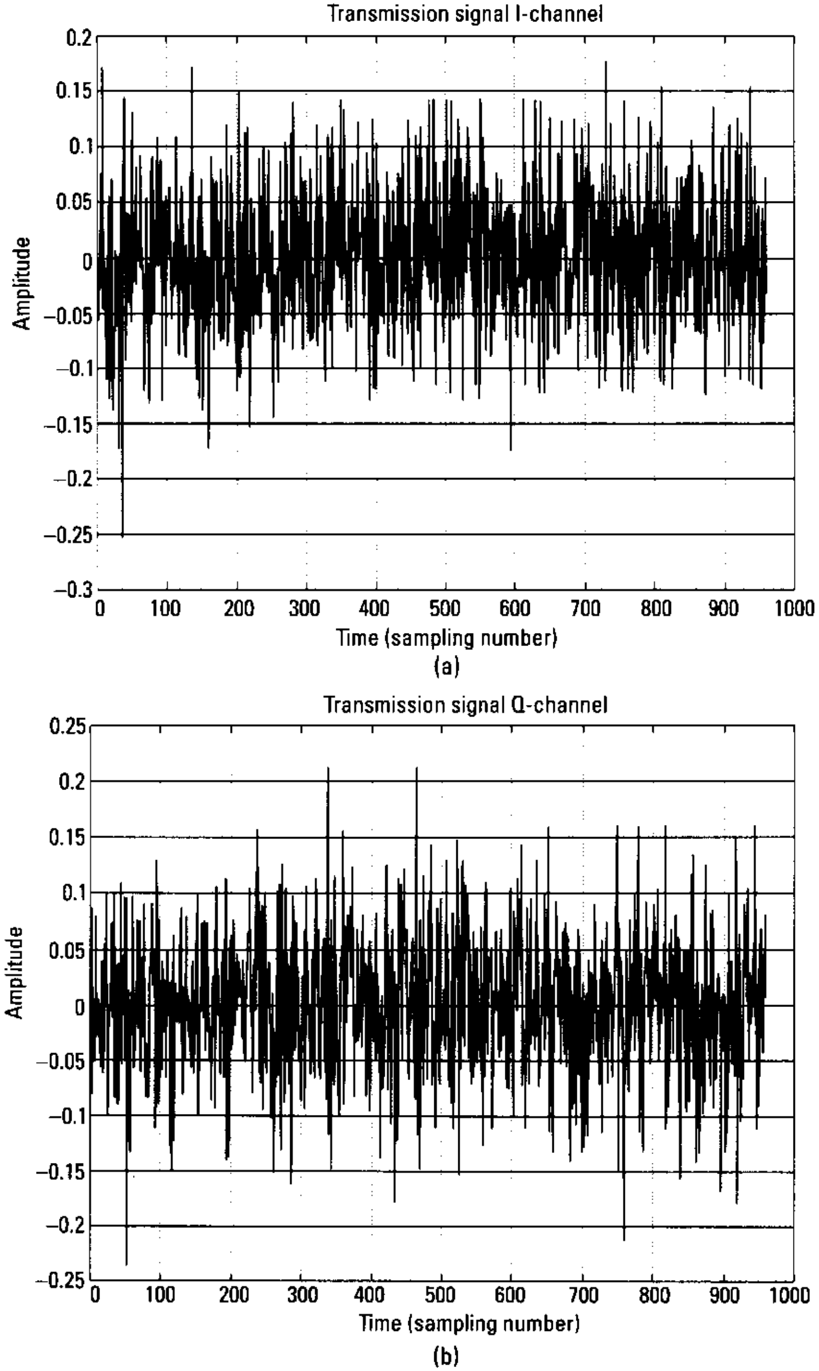


Figure 4.10 (a) Transmitted signal in I-channel and (b) transmitted signal in Q-channel.

a function comb.m. In this simulation, we wanted to create a graph showing the relationship between E_b/N_0 and BER. This means that we had to change variable "attn" in accordance with given E_b/N_0. Variable "attn" was calculated by a procedure similar to that used in BPSK. Here, "spow" refers to the signal power per carrier per symbol. For the OFDM system, "spow" had to be divided by "para," which indicates the number of parallel subcarriers.

```
spow=sum(ich3.^qch3)./nd./para;
attn=0.5*spow*sr/br*10.^(-ebn0/10);
attn=sqrt(attn);
```

By using "attn" and comb.m, the transmitted data was contaminated by AWGN.

```
[ich4,qch4]= comb(ich3,qch3,length(ich3),attn);
```

Then, the guard interval was removed from received signals ich4 and qch4.

```
[ich5,qch5]= girem(ich4,qch4,fftlen2,gilen,nd);
```

These data, "ich5" and "qch5," on the time axis, were fed into the FFT circuit. In the circuit, the serial data were converted into parallel data on the frequency axis.

```
rx=ich5+qch5.*i;
ry=fft(rx);
ich6=real(ry);
qch6=imag(ry);
```

Then, the converted data were divided by "kmod" in each channel to unnormalize the data and were fed into the demodulation function.

```
ich7=ich6./kmod;
qch7=qch6./kmod;
[demodata]=qpskdemod(ich7,qch7,para,nd,ml);
```

After that, the demodulated data were converted into a 1-by-para*nd*ml vector. The data were called "demodata1."

```
demodata1=reshape(demodata,1,para*nd*ml);
```

Next, we calculated the number of bit errors. The procedure is described in Section 2.3. At the same time, we calculated the number of packet errors. In this simulation, the transmitted data are referred to as "seridata" and the received data are referred to as "demodata1." The calculations were performed as follows.

```
% instantaneous number of errors and data bits
noe2=sum(abs(seridata-demodata1); nod2=length(seridata);
% cumulative number of errors and data bits in noe and nod
noe=noe+noe2;
nod=nod+nod2;
% calculating PER
if noe2~=0
      eop=eop+1;
else
      eop=eop;
end
eop;
nop=nop+1;
```

We then obtained the BER and PER by using the following operation.

```
ber=noe/nod;
per=eop/nop;
```

By running the main program

```
ofdm
```

The BER and PER are obtained after a long simulation. The BER performance is shown in Figure 4.11 where it is compared with theoretical value. In the simulation result, there was a 0.9691-dB shift from the theoretical value. The

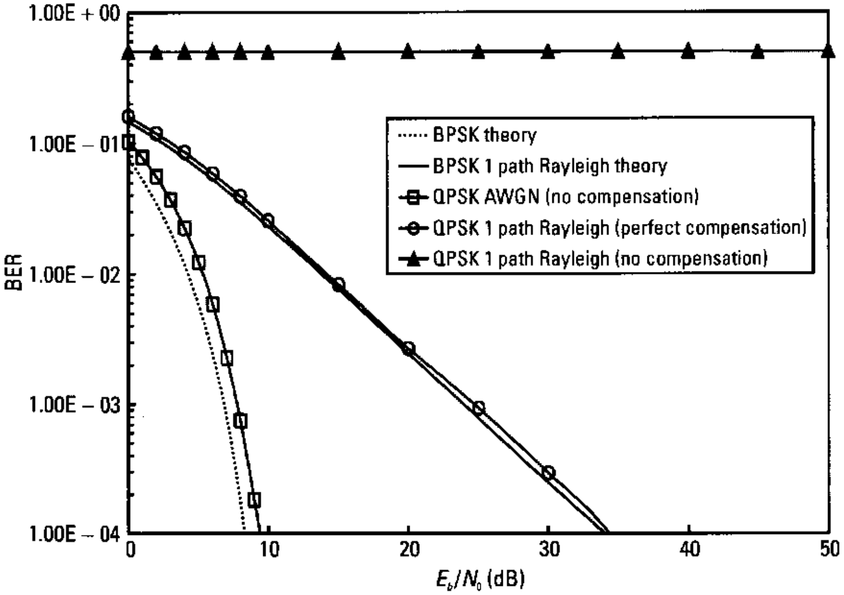


Figure 4.11 BER performance (para = 128).

shift was caused by the cutting off of the guard interval power from the received signal. It is calculated as follows.

$$\text{shifted value (dB)} = -10\log 10\left(\frac{\texttt{gilen}}{\texttt{fftlen2}}\right) \tag{4.9}$$

Previously, we defined the length of the guard interval and the length of one symbol including the guard interval as `gilen` and `fftlen2`, respectively. In Figure 4.11, we also show the BER performance under one-path flat Rayleigh fading. For the simulation, we use Program 4.2, where we first determine the fading parameters described in Chapter 3 and add the following parameters to generate fading.

```
% Generated data are fed into a fading simulator
[ifade,qfade]=sefade(ich3,qch3,itau,dlvl,th1,n0,itndl,...
nowl,length(ich3),tstp,fd,flat);
% Updata fading counter
itndl=itndl+itnd0
```

By defining variable "`flat`" as 1 or 0, we could determine, by using simulation, whether the fading channel can be compensated for automatically or not. For the BER performance under one-path Rayleigh fading, if we can compensate for amplitude and phase fluctuation caused by propagation characteristics perfectly, we can obtain a 0.969-dB shift from the theoretical value. However, if we cannot compensate for the fluctuation characteristics, we cannot recover the data. It is important to estimate the propagation characteristics in real time. One of the estimation methods is pilot-symbol insertion in which known pilot symbols are inserted at a known period as shown in Figure 4.12. At the receiver, we can estimate the channel characteristics for every symbol. By using the estimated propagation characteristics, we can recover the transmitted data. Section 4.5 discusses this method in detail.

4.5 A Pilot Symbol–Aided OFDM Modulation Scheme

This section describes a method to compensate for the fluctuation of amplitude and phase due to fading. This method is called the pilot symbol–aided OFDM modulation scheme. In this method, pilot symbols are inserted at the transmitter at fixed time intervals as shown in Figure 4.12, and at the receiver, we estimate the channel characteristics by using the pilot symbols. Because the level of fluctuation is independent in each subcarrier channel, we can insert pilot carriers in all frequency domains at a known time period. Then, by using the estimated channel characteristics, we can recover the transmitted data. This section shows the configuration of the pilot symbol–aided OFDM modulation scheme and evaluates the BER performance by computer simulation. A block diagram of the simulation is shown in Figure 4.6. Compared to the simulation described in Section 4.4, this simulation is more similar to the real conditions. In this simulation, we use an OFDM-based WLAN system used in ETSI BRAN, IEEE 802.11, and ARIB MMAC projects. The parameters are listed as follows.

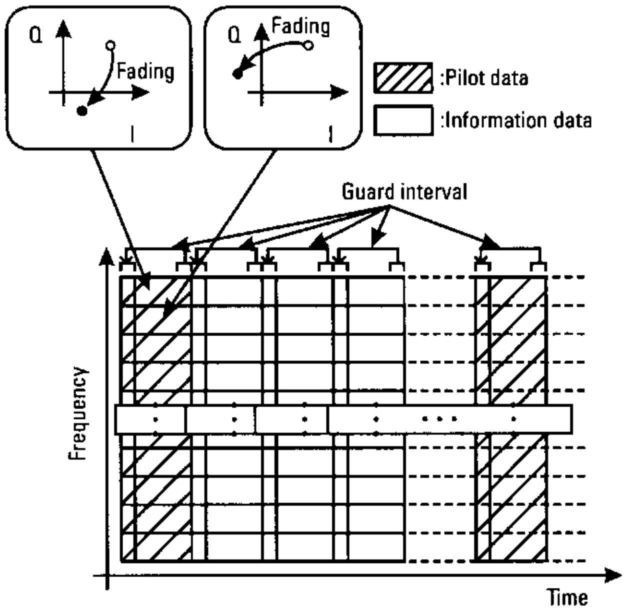


Figure 4.12 Frame format of the simulation model.

- *Number of subcarriers:* Fifty-two subcarriers are adopted and generated by a 64-point FFT circuit. Of the 52 subcarriers, 48 are used for the information data. The rest are used to compensate for the phase noise. In this simulation, we do not simulate a phase-noise environment. Therefore, we input data into all 52 carriers. Figure 4.13 shows the frequency allocation for the 52 carriers in the 64-point IFFT circuit.
- *Guard interval: 800 ns:* To avoid the effects of multipath fading where delay time is greater than the symbol length, a cyclically extended signal was inserted before each OFDM signal. We used a guard interval of 800 ns, because we must consider using the OFDM-based wireless communication system for not only an indoor environment but also in an outdoor microcellular environment. For a 5-GHz environment, an interval of 800 ns is enough to cover major delayed waves. Figure 4.14

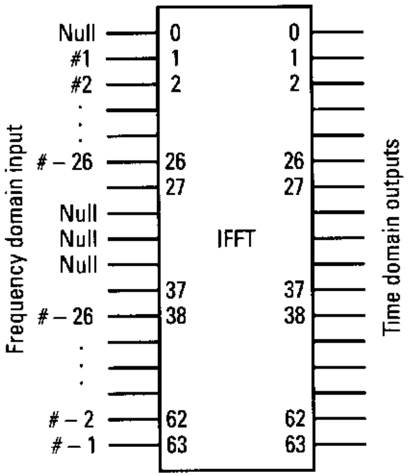


Figure 4.13 Input and outputs of IFFT.

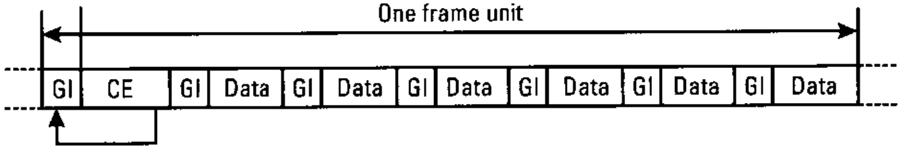


Figure 4.14 Frame format of the simulated OFDM transmission.

shows the basic frame format and configuration of each OFDM symbol.

- *Sampling rate:* The sampling rate (20 MHz) was the same as the input-signal rate of the IFFT input signal. This is because we wanted to achieve a total throughput of more than 20 Mbps.
- *Modulation scheme:* In a WLAN environment, differential encoding and detection-based modulation schemes, such as D8PSK, are used. However according to several standardization committees, the use of a broadband data terminal is possible not only in an indoor environment but also in an outdoor microcellular environment. Therefore, we discuss coherent detection-based modulation schemes—such as BPSK, QPSK, 8PSK, and 16-QAM—that are used to improve the quality of the transmitted data and preserve robustness against multipath fading not only in an indoor but also in an outdoor environment. This subsection, (to compare the results presented in Section 4.4) uses a coherent QPSK-based OFDM system.

- *FEC:* Basically, FEC is based on convolutional coding and soft-decision Viterbi decoding with R = 1/2 and K = 7 (R = coding rate, K = constraint length). For other coding rates, we use punctured convolutional coding and soft-decision Viterbi decoding. This subsection, however, does not discuss any encoding methods.
- *Frame format:* Figure 4.14 shows the frame format of our simulated OFDM system. The frame is divided into two parts: channel-estimation (CE) symbol and transmitted data symbols. This chapter uses one CE symbol and six transmitted data symbols as one frame unit. In the CE symbol, the amplitude and phase deviation from the pilot data are measured by using a pilot signal. Based on the measured propagation characteristics, the deviation of the amplitude and phase of the six OFDM data symbols caused by multipath fading is compensated for. To ensure smooth communication based on the results from the previous section, we simulate the following procedure.

  A. Fifty-two-carrier non-pilot-symbol–assisted QPSK-OFDM transmission scheme;

  B. Fifty-two-carrier pilot-symbol–assisted QPSK-OFDM transmission scheme;

  C. Fifty-two-carrier pilot-symbol–assisted QPSK-OFDM transmission scheme (E_b/N_0 versus BER or E_b/N_0 versus PER under AWGN, one-path Rayleigh fading, two-path Rayleigh fading);

  D. Fifty-two-carrier pilot-symbol–assisted QPSK-OFDM transmission scheme (carrier-to-interference ratio versus BER and PER).

- *Radio channel model:* In this simulation, we use a normal one-path Rayleigh fading and a two-path Rayleigh fading channel. In the two-path Rayleigh fading environment, the delay time of the delayed wave is quite important. In this simulation, the guard interval is 800 ns. Therefore, we set 250 ns for the delay time. In addition, in this simulation, we always use the Doppler frequency of *fd* = 50 Hz (3 m/s @5 GHz) or *fd* = 150 Hz (15 m/s @5 GHz).

To evaluate the system performance, we simulate not only the bit error probability but also the PER in which the packet is defined as the number of transmitted data bits in one frame unit. In this case, six OFDM symbols exist in one frame unit. If more than one of these transmitted data bits in one frame makes a mistake, a packet error occurs.

For simulation A, all the programs were used from Program 4.5 to 4.7. In Program 4.5, first of all, we first decided on the common variables we used throughout Program 4.1.

```
para=52;     % Number of parallel channel
fftlen=64;   % FFT length
noc=53;      % Number of carriers
nd=6;        % Number of OFDM symbol for one loop
ml=2;        % Modulation level: QPSK
sr=250000;   % OFDM symbol rate
br=sr.*ml;   % Bit rate per carrier
gilen=16;    % Length of guard interval (points)
ebn0=3;      % EbN0
```

We then initialized the fading parameters.

```
tstp=1/sr(fftlen+gilen);  % Time resolution
itau=[0];                 % Arrival time
dlvl1=[0];                % Mean power for each
                          % multipath
n0=[6];                   % Number of waves to generate
                          % fading
th1=[0.0];                % Initial phase of delayed
                          % wave
itnd1=[1000];             % Set fading counter
now1=1;                   % Number of direct wave +
                          % delayed wave
fd=150;                   % Maximum Doppler frequency
flat=0;                   % Flat or not (see
                          % ofdm_fading.m)
itnd0=nd*(fftlen+gilen)*20;% Number of fading counter
```

Next, we defined the variables for the simulation.

```
nloop=1000; % Number of simulation loops
noe=0;      % Number of error data
nod=0;      % Number of transmitted data
eop=0;      % Number of error packet
nop=0;      % Number of transmitted packet
```

After defining all the variables, we began our simulation to obtain the BER and PER performance. First, we generated random serial data of 0 and 1, considering a 1-by-para*nd*ml vector. We called the data "seridata."

```
seridata=rand(1,para*nd*ml) > 0.5;
```

The "seridata" were converted into "paradata" consisting of a para-by-nd*ml vector to transmit in parallel.

```
paradata=reshape(seridata,para,nd*ml);
```

Next, the vector paradata were fed into the mapping circuit. In the circuit, the parallel data were converted into the parallel data of two channels, Ich and Qch, by using a predefined mapping method.

```
[ich,qch]=qpskmod(paradata,para,nd,ml);
```

Then, these data were increased kmod times to normalize the data as follows.

```
kmod=1/sqrt(2);
ich1=ich.*kmod;
qch1=qch.*kmod;
```

The transmitted data in Ich and Qch in the frequency domain were configured according to the format shown in Figure 4.13. The waveforms are shown in Figure 4.15.

After the mapping, these parallel data on the frequency axis were fed into the IFFT circuit. In this case, we achieved 52-subcarrier transmission by using an OFDM technique based on a 64-point IFFT circuit. The allocation method is shown in Figure 4.13.

```
[ich1,qch1]=crmapping(ich,qch,fftlen,nd);
```

After the allocation, ich1 and qch1 were fed into the IFFT circuit, and their waveforms were configured into OFDM waveforms.

```
X=ich1+qch1.*i;     %  i : complex number
y=ifft(x);          %  ifft : built-in function
ich2=real(y);       %  real : built-in function
qch2=imag(y);       %  imag : built-in function
```

These two serial data symbols had a guard interval inserted in each channel to eliminate ISI caused by multipath fading.

```
fftlen2=fftlen+gilen;
[ich4,qch4]= giins(ich2,qch2,fftlen,gilen,nd);
```

At this point, we defined fftlen2 as the length of a symbol including the guard interval. The filtered signal was then transmitted to the air. The OFDM-modulated data in Ich and Qch are shown in Figure 4.16.

Then, the transmitted signal passed through a radio channel (equivalent lowpass system) and was transmitted to the receiver. The transmitted signal was first contaminated by multipath fading. In the simulation, we used command "%" in the function. However, if you remove command "%," you can simulate the BER and PER performance in a Rayleigh fading environment.

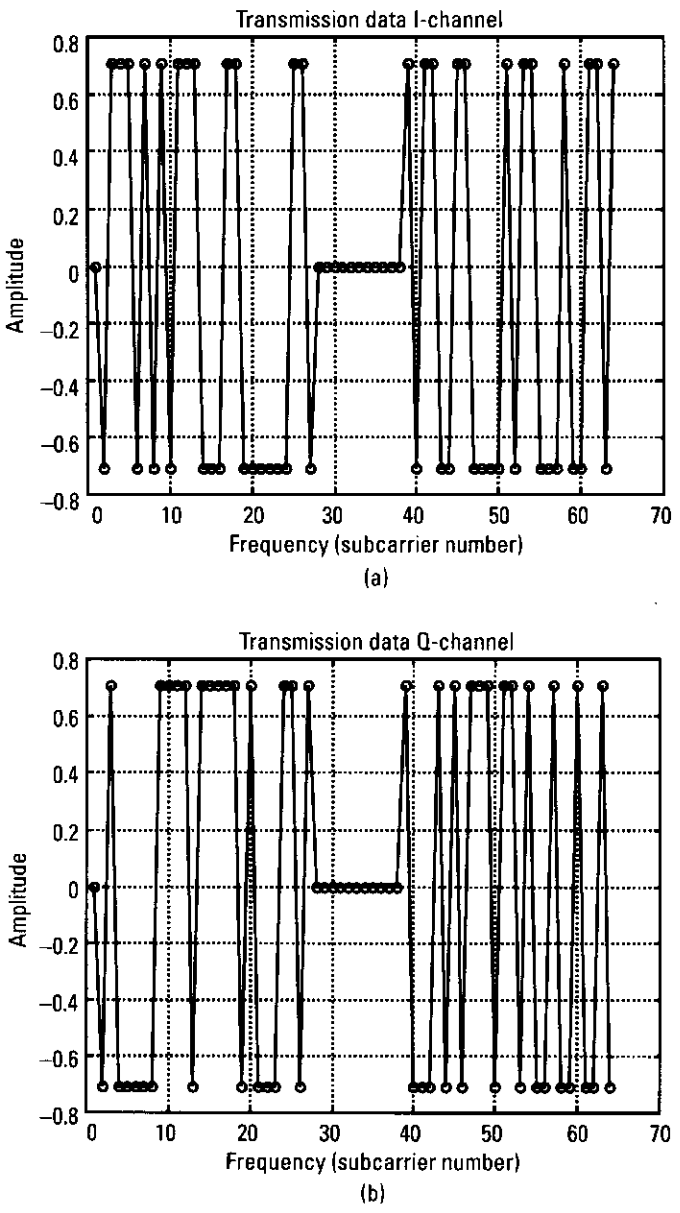


Figure 4.15 (a) Transmission data in I-channel and (b) transmission data in Q-channel.

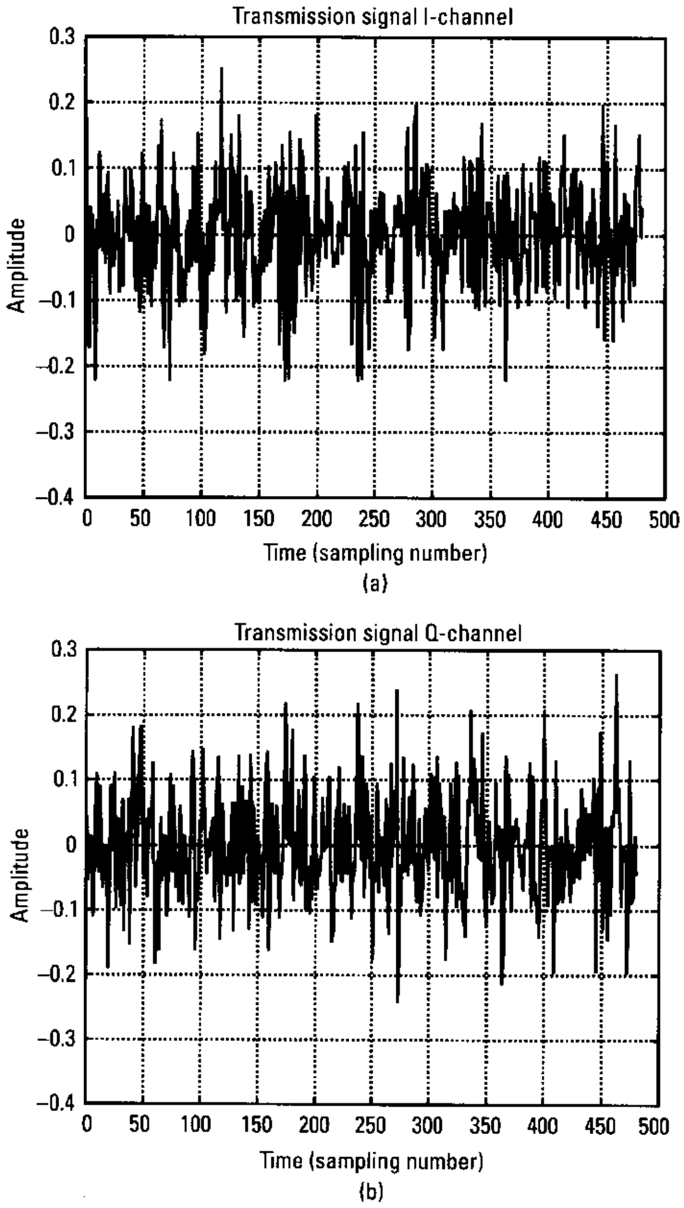


Figure 4.16 (a) Transmitted signal in I-channel and (b) transmitted signal in Q-channel.

At the receiver, the received signal was contaminated by AWGN. The noise function was already discussed in Section 3.2.3 where we used function `comb.m`. In this simulation, we wanted to make a graph showing the relationship between E_b/N_0 and BER. This means that we had to change "`attn`" in accordance with given E_b/N_0. Variable "`attn`" was calculated by a procedure similar to that used in BPSK. Here "`spow`" refers to the signal power per carrier per symbol. For OFDM, "spow" had to be divided by "para" indicating the number of parallel subcarriers.

```
spow=sum(ich4.^2+qch4.^2)/nd./para;
attn=0.5*spow*sr/br*10.^(-ebn0/10);
attn=sqrt(attn);
```

By using attn and comb.m, the transmitted data was contaminated by AWGN.

```
[ich5,qch5]=comb(ich4,qch4,attn);
```

Then, the guard interval was removed from received signals ich5 and qch5.

```
[ich6,qch6]= girem(ich5,qch5,fftlen2,gilen,nd);
```

Then, these serial data, "ich6" and "qch6," on the time axis, were fed into the FFT circuit. In the circuit, the serial data were converted into parallel data on the frequency axis.

```
rx=ich6+qch6.*i;
ry=fft(rx);   %  fft : built in function
ich7=real(ry);
qch7=imag(ry);
```

As a result, we obtained a 64-channel output, because the FFT circuit performed 64-point FFT. However, the information was in the 52 channels. Therefore, we brought out the 52 channels from the 64-channel output.

```
[ich8,qch8]=crdemapping(ich7,qch7,fftlen,nd);
```

Then, the converted data were divided by "kmod" in each channel to unnormalize the data and were fed into a demodulation function.

```
[ich8,qch8]=crdemapping(ich7,qch7,fftlen,nd);
ich9=ich8./kmod;
qch9=qch8./kmod;
[demodata]=qpskdemod(ich9,qch9,para,nd,ml);
```

After that, the demodulated data was converted to a 1-by-para*nd*ml vector. We called the data "demodata1."

```
demodata1=reshape(demodata,1,para*nd*ml);
```

Next we calculated the number of bit errors. The procedure is described in Section 2.3. At the same time, we calculated the number of packet errors. In this simulation, the transmitted data were "seridata" and the received data were "demodata1." The calculations were performed as follows.

```
% instantaneous number of errors and data
noe2=sum(abs(demodata1-seldata));   nod2=length(seldata);
%  length : built in function
% calculating BER
noe=noe+noe2;
nod=nod+nod2;
% calculating PER
if noe2~=0
  eop=eop+1;
else
  eop=eop;
end
  eop;
  nop=nop+1;
```

We then obtained the BER and PER by using the following operation.

```
per=eop/nop;
ber=noe/nod;
```

After a long simulation of ofdmda.m, we obtained the BER and PER. The BER performance is shown in Figure 4.17 where it is compared with theoretical value. The PER performance is shown in Figure 4.18. In the simulation result for the BER, there was a +0.9691-dB shift from the theoretical value. The shift of the value was caused by the guard interval power for the received signal. It is calculated as follows:

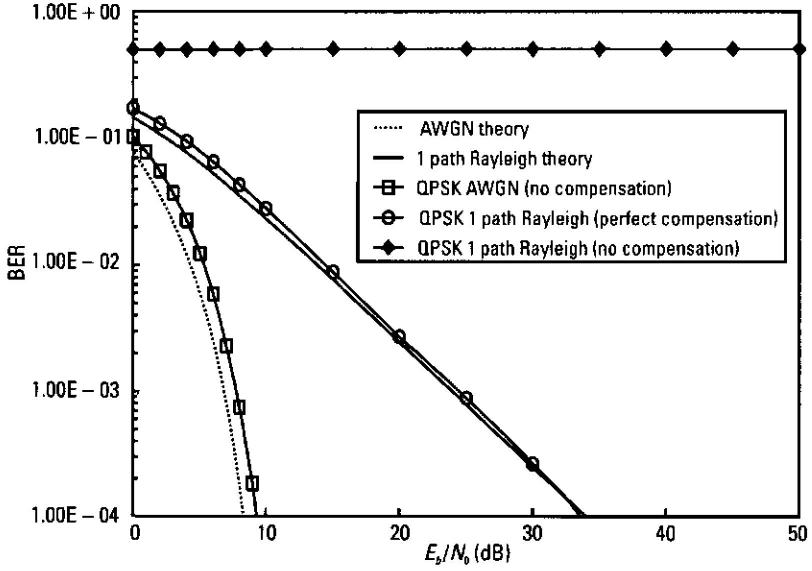


Figure 4.17 Performance of BER (DATA).

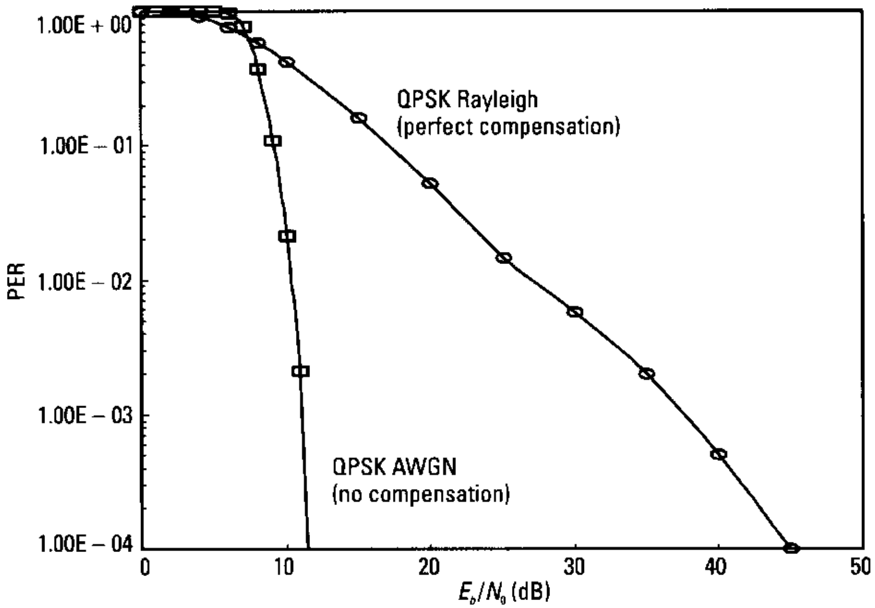


Figure 4.18 Performance of PER (DATA).

$$\text{shifted value} = -10\log 10\left(\frac{\texttt{gilen}}{\texttt{fftlen2}}\right) \tag{4.10}$$

Previously we defined the length of the guard interval and the length of 1 symbol including the guard interval as `gilen` and `fftlen2`, respectively. In Figure 4.17, we also show the BER performance under one-path flat Rayleigh fading. For the simulation, we removed command "`%`" in the program of the fading channel in Program 4.5. By changing a variable "`flat`" between 1 and 0, we could determine by simulation whether the fading channel can be compensated for or not. From the BER performance under one-path Rayleigh fading, we found that if we can compensate for the amplitude and phase fluctuations caused by fading perfectly, we can obtain a 0.9691-dB shifted from the theoretical value. However, if we cannot compensate for the fluctuations, we cannot recover the data. One of the methods to estimate the propagation characteristics is the pilot-symbol insertion, in which known pilot symbols are inserted as known periods as shown in Figure 4.12.

At the receiver, we can estimate the channel characteristics for every symbol. By using the estimated propagation characteristics, we can recover the transmitted data. Next, we will show how to add a pilot data-insertion circuit to Program 4.8.

In Program 4.8, we must insert a pilot data insertion program. Please refer to "`% CE data generation`" in Program 4.8. This is where the pilot data are configured. The pilot data here are random data, and they are inserted only in "`Ich`."

```
% CE data generation
kndata=zeros(1,fftlen);
kndata0=2.*(rand(1,52)>0.5)-1;
kndata(2:27)=kndata0(1:26);
kndata(39:64)=kndata0(27:52);
ceich=kndata; % CE:BPSK
ceqch=zeros(1,64);
```

The data are inserted in the time domain before information OFDM data. Please refer to comment "`%--------data mapping (DC=0)--------`" in Program 4.8. Then,

```
[ich1,qch1]=crmapping(ich,qch,fftlen,nd);
ich2=[ceich.' ich1]; % I-channel transmitted data
qch2=[ceqch.' qch1]; % Q-channel transmitted data
```

By using the formatted transmitted data, `ich2` and `qch2`, we can perform FFT.

```
%--------- IFFT ---------
x=ich2+qch2.*i;
y=ifft(x);
ich3=real(y);
qch3=imag(y);
```

Then, a cyclic guard signal is inserted and transmitted to the air. After that, as mentioned in Program 4.5, the transmitted signal is contaminated by multipath fading and AWGN. Then, at the receiver, the inserted guard interval is removed. Subsequently, the removed signal is fed into the FFT circuit.

```
rx=ich6+qch6.*i;
ry=fft(rx);
ich7=real(ry);
qch7=imag(ry);
```

By using CE symbols, we can then estimate the propagation characteristics. Please refer to the comment "`%-------- fading compensation by CE symbol --------`." First, we take out the pilot symbols from the received and FFT-performed data `ich7` and `qch7`. In this simulation, the pilot data were located in the first symbol time as shown in Figure 4.14. Therefore,

```
% taking pilot data out of received data
ce=1;
ice1=ich7(:,ce);
qce1=qch7(:,ce);
```

At the same time, we prepared pilot data that were used in the transmitter.

```
% preparation known CE data
ce=1;
ice0=ich2(:,ce);
qce0=qch2(:,ce);
```

The relationship between $\begin{pmatrix}\texttt{ice1}\\ \texttt{qce1}\end{pmatrix}$ and $\begin{pmatrix}\texttt{ice0}\\ \texttt{qce0}\end{pmatrix}$ is given by

$$\begin{pmatrix}\texttt{ice1}\\ \texttt{qce1}\end{pmatrix}=A\begin{pmatrix}\texttt{ice0}\\ \texttt{qce0}\end{pmatrix} \tag{4.11}$$

where A is the transition matrix of the fading environment and shown as follows:

$$A=\begin{pmatrix}iv & -qv\\ qv & iv\end{pmatrix} \tag{4.12}$$

because fading is a function with a phase rotation and amplitude fluctuation. To compensate for the fading rotation, we multiplied all the received data by A^{-1}. A^{-1} is given by

$$A^{-1}=\frac{1}{\sqrt{iv^2+qv^2}}\begin{pmatrix}iv & qv\\ -qv & iv\end{pmatrix} \tag{4.13}$$

From (4.11) to (4.13), the values of iv and qv are given by

$$iv=\frac{1}{\sqrt{\texttt{ice1}+\texttt{qce1}}}(\texttt{ice0}\times\texttt{ice1}+\texttt{qce0}\times\texttt{qce1}) \tag{4.14}$$

$$qv=\frac{1}{\sqrt{\texttt{ice1}+\texttt{qce1}}}(\texttt{qce0}\times\texttt{ice1}-\texttt{ice0}\times\texttt{qce1}) \tag{4.15}$$

In this simulation, we calculate parameters iv and qv.

```
% calculating reverse rotation
iv=real((1./(ice1.^2+qce1.^2)).*(ice0+i.*qce0).*...
      (ice1-i.*qce1));
qv=imag((1./(ice1.^2+qce1.^2)).*(ice0+i.*qce0).*...
      (ice1-i.*qce1));
```

By using iv and qv, two inverse rotation matrices were produced as follows:

```
% matrices for reverse rotation
ieqv1=[iv iv iv iv iv iv iv];
qeqv1=[qv qv qv qv qv qv qv];
```

Because there is one data-frame unit, there is one channel estimation symbol and six data symbols, the length of the vector is 7 in both parameters. Using the vectors, we performed an inverse rotation for received signals ich7 and qch7. The inverse rotated data were icompen and qcompen, and the relationship between ich7, qch7, icompen, qcompen, ieq1, and qeq1 is given by

$$\texttt{icompen}=\texttt{ich7}\times\texttt{ieqv1}-\texttt{qch7}\times\texttt{qeqv1} \tag{4.16}$$

$$\texttt{qcompen}=\texttt{qch7}\times\texttt{ieqv1}+\texttt{ich7}\times\texttt{qeqv1} \tag{4.17}$$

In the simulation, we calculated the above values.

```
% reverse rotation
icompen=real((ich7+i.*qch7).*(ieqv1+i.*qeqv1));
qcompen=imag((ich7+i.*qch7).*(ieqv1+i.*qeqv1));
ich7=icompen;
qch7=qcompen;
```

We removed the pilot symbols from the phase-compensated data as follows.

```
%-------- CE symbol removal ---------
ich8=ich7(:,knd+1:nd+1);
qch8=qch7(:,knd+1:nd+1);
```

After that, the procedure we used was the same as that described in Program 4.5.

By using the above phase compensation in our computer simulation, we obtained the BER and PER. The BER performance is shown in Figure 4.19 together with the theoretical value. From the BER performance under one-path Rayleigh fading, we found that if we can compensate for the amplitude and phase fluctuations caused by fading, we can obtain a 0.9691-dB shift from the theoretical value. However, if we cannot compensate for the fluctuations, we cannot recover the data. On the other hand, if we use a pilot signal-assisted OFDM transmission scheme, we can obtain a 2-dB shift from the theoretical value. Figure 4.19 shows the simulated BER performance when there is perfect compensation for the amplitude and phase fluctuation caused by propagation characteristics in a one-path fading environment. For this simulation, one only needs to remove comments "%" in the area of %----Fading compensation by CE symbol—in Program 4.8. For the case of perfect compensation, the

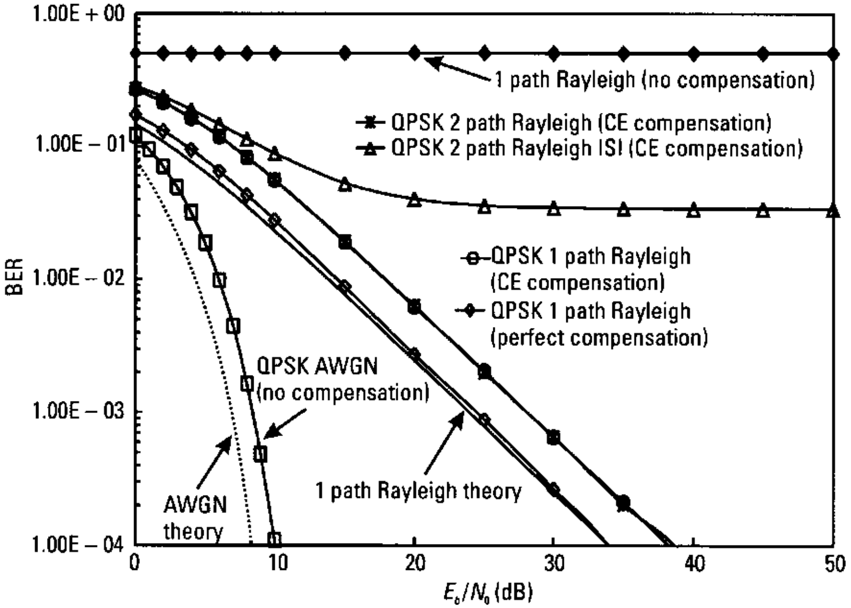


Figure 4.19 Performance of BER (CE+DATA).

value of BER was the same as that obtained in the pilot-assisted OFDM system. This means that the BER shifted 2 dB compared to the theoretical value. This was because we input pilot data of 1/7 in one frame unit. As a result, as the number of inserted pilot data symbols increased, the performance degraded and BER performance became even worse when we inserted the guard.

In a two-path channel, the BER performance of the pilot signal–assisted OFDM system partly depends on the position of the delayed wave in two-path Rayleigh fading. If the delay time of the delayed wave is shorter than the guard interval, all fluctuations of the amplitude and phase can be removed by using pilot signals, and the BER performance is the same as that in one-path Rayleigh fading. However, if the delay time is longer than the guard integral, ISI contaminates the next symbol and the BER performance degrades. The PER performance is shown in Figure 4.20. As can be seen in Figure 4.20, the PER degraded.

We also calculated the carrier-to-interference ratio (C/I) versus BER and C/I versus PER by using computer simulation. In this simulation, we used other pilot signal–based OFDM signals as interference noise. The interference was added by using Program 4.9. Using Program 4.10, we can obtain a pilot signal–based OFDM signal, whose power depends on the value of C/I and whose data source and contaminated fading are different for desired and undesired signals. Program 4.9 is the main program to calculate C/I versus BER and C/I versus PER. In Program 4.9, we first define some parameters for the interference signal.

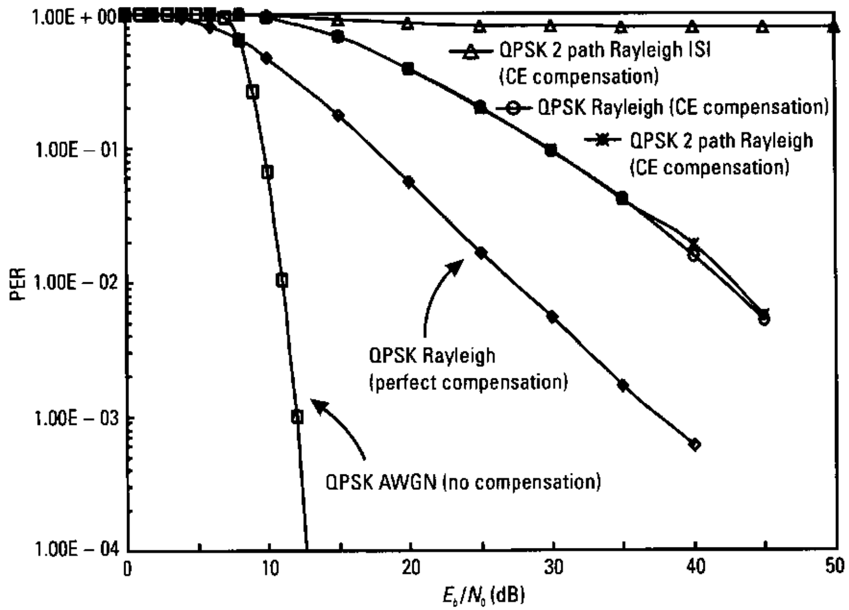


Figure 4.20 Performance of PER (CE+DATA).

```
%-------interference wave initialization -------
ci=10;              % C/I ratio
ml2=2;              % modulation level
itau2=[0];
dlvl2=[0];
n02=[6];
th2=[0.0];
itnd2=[10000+floor(rand(1)*10)*1000];
now2=1;
fd2=fd;
flat2=0;
% Number of fading counter to skip
itnd02=nd*(fftlen+gilen)*300;
```

Then, the parameters of fading for the interference signal are stored to a matrix `fadingpara`.

```
% store all parameters in one matrix "fadingpara"
fadingpara=zeros(8,length(itau2));
fadingpara(1,:)=itau2;
fadingpara(2,:)=dlvl2;
fadingpara(3,:)=n02;
fadingpara(4,:)=th2;
fadingpara(5,:)=itnd2;
fadingpara(6,:)=now2;
fadingpara(7,:)=fd2;
fadingpara(8,:)=flat2;
```

Then, by using the matrix `fadingpara`, the interference signal is generated and added to the desired signal.

```
%%% interference wave addition
% interference
[iintw,qintw]=interwave(ci,spow,ml2,length(ich4),tstp,
  fadingpara);
      itnd2=itnd2+itnd02;
fadingpara(5,:)=itnd2;
ich4=ich4+iintw;
qch4=qch4+qintw;
```

By changing the value of C/I, we can obtain the BER and PER as shown in Figures 4.21 and 4.22, respectively.

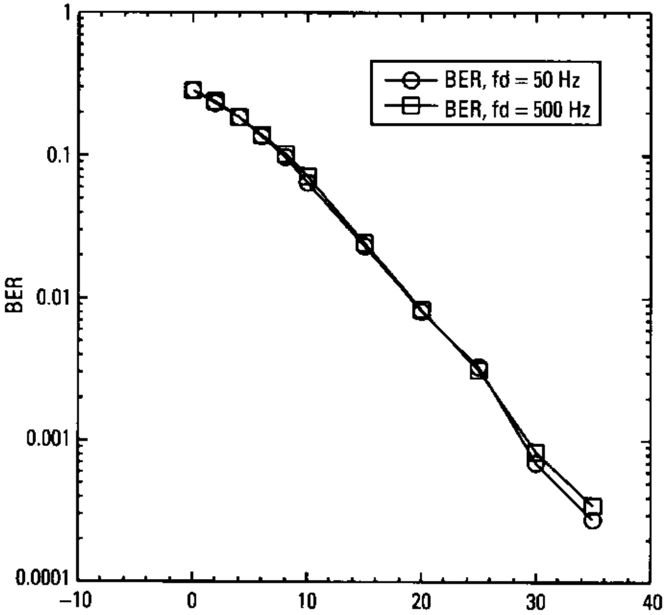


Figure 4.21 Performance of BER (CE+DATA).

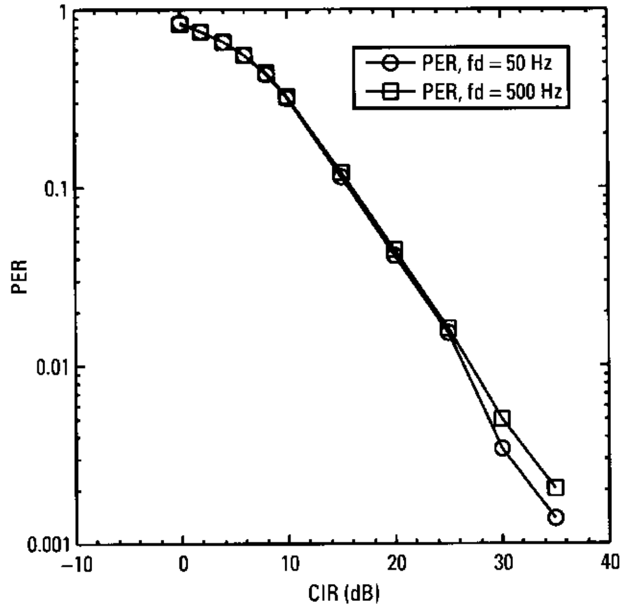


Figure 4.22 Performance of PER (CE+DATA).

4.6 Conclusions

This chapter describes the concept behind parallel data transmission and the configuration of an OFDM transmitter and receiver. We explained how we configured the OFDM transmitter and receiver by using computer simulation. We discussed the results of our extended simulation of OFDM where we simulated the broadband WLAN system of the future by using a pilot symbol–assisted OFDM transmission system. We showed the effectiveness of the pilot symbol–assisted OFDM transmission system. If we develop a prototype system, we must simulate the synchronization method for OFDM signals to remove the guard interval at the optimum point. When comparing the transmission performance of the system with that of other modulation schemes, we must consider the nonlinearity of the amplifier and phase noise.

References

[1] Chang, R. W., and R. A. Gibby, "A Theoretical Study of the Performance of an Orthogonal Multiplexing Data Transmission Scheme," *IEEE Trans. Commun.*, Vol. COM-16, No. 4, August 1968, pp. 527–540.

[2] van Nee, R., and R. Prasad, *OFDM for Wireless Multimedia Communications*, Norwood, MA: Artech House, 1999.

[3] Prasad, R., *Universal Wireless Personal Communications*, Chapter 10, Norwood, MA: Artech House, 1998.

[4] http://www.etsi.org/bran/.

[5] IEEE802.11, IEEE Standard for Wireless LAN Medium Access Control and Physical Layer Specifications, November 1997.

[6] http://www.arib.or.jp/mmac/e/index.htm.

Appendix 4A

Program 4.1

```
% Program 4-1
% ofdm.m
%
% Simulation program to realize OFDM transmission system
%
% programmed by T. Yamamura and H. Harada
%

%****************** preparation part ******************

para=128;     % Number of parallel channel to transmit
fftlen=128;   % FFT length
noc=128;      % Number of carrier
nd=6;         % Number of information OFDM symbol for
              % one loop
ml=2;         % Modulation level : QPSK
sr=250000;    % Symbol rate
br=sr.*ml;    % Bit rate per carrier
gilen=32;     % Length of guard interval (points)
ebn0=3;       % Eb/N0

%******************* main loop part ******************

nloop=100;    % Number of simulation loops

noe = 0;      % Number of error data
nod = 0;      % Number of transmitted data
eop=0;        % Number of error packet
nop=0;        % Number of transmitted packet

for iii=1:nloop

%******************* transmitter *********************

%****************** Data generation *******************

seldata=rand(1,para*nd*ml) > 0.5;
  % rand : built in function

%*********** Serial to parallel conversion ************

paradata=reshape(seldata,para,nd*ml);
  %  reshape : built in function

%****************** QPSK modulation ******************

[ich,qch]=qpskmod(paradata,para,nd,ml);
kmod=1/sqrt(2); %  sqrt : built in function
```

```
ich1=ich.*kmod;
qch1=qch.*kmod;

%*********************** IFFT ***********************

x=ich1+qch1.*i;
y=ifft(x);         % ifft : built in function
ich2=real(y);      % real : built in function
qch2=imag(y);      % imag : built in function

%************* Guard interval insertion *************

[ich3,qch3]= giins(ich2,qch2,fftlen,gilen,nd);
fftlen2=fftlen+gilen;

%*************** Attenuation Calculation ************

spow=sum(ich3.^2+qch3.^2)/nd./para;
  %  sum : built in function
attn=0.5*spow*sr/br*10.^(-ebn0/10);
attn=sqrt(attn);

%********************  Receiver  ********************

%******************* AWGN addition ******************

[ich4,qch4]=comb(ich3,qch3,attn);

%**************** Guard interval removal ************

[ich5,qch5]= girem(ich4,qch4,fftlen2,gilen,nd);

%**********************  FFT  ***********************

rx=ich5+qch5.*i;
ry=fft(rx);     % fft : built in function
ich6=real(ry);  % real : built in function
qch6=imag(ry);  % imag : built in function

%******************** demodulation ******************

ich7=ich6./kmod;
qch7=qch6./kmod;
[demodata]=qpskdemod(ich7,qch7,para,nd,ml);

%********** Parallel to serial conversion ***********

demodata1=reshape(demodata,1,para*nd*ml);

%*************** Bit Error Rate (BER) ***************

% instantaneous number of error and data
```

```
noe2=sum(abs(demodata1-seldata));
  %  sum : built in function
nod2=length(seldata);
  %  length : built in function

% cumulative the number of error and data in noe and nod

noe=noe+noe2;
nod=nod+nod2;

% calculating PER

if noe2~=0
  eop=eop+1;
else
  eop=eop;
end
  eop;
nop=nop+1;

fprintf('%d\t%e\t%d\n',iii,noe2/nod2,eop);
  %  fprintf : built in function

end

%****************** Output result *******************

per=eop/nop;
ber=noe/nod;

fprintf('%f\t%e\t%e\t%d\t\n',ebn0,ber,per,nloop);
fid = fopen('BERofdm.dat','a');
fprintf(fid,'%f\t%e\t%e\t%d\t\n',ebn0,ber,per,nloop);
fclose(fid);

%******************** end of file *******************
```

Program 4.2

```
% Program 4-2
% ofdm_fading.m
%
% Simulation program to realize OFDM transmission system
% (under one path fading)
%
% programmed by T. Yamamura and H. Harada
%

%***************** preparation part *****************

para=128;       % Number of parallel channel to transmit
fftlen=128;     % FFT length
```

```
noc=128;          % Number of carrier
nd=6;             % Number of information OFDM symbol for
                  % one loop
ml=2;             % Modulation level : QPSK
sr=250000;        % Symbol rate
br=sr.*ml;        % Bit rate per carrier
gilen=32;         % Length of guard interval (points)
ebn0=10;          % Eb/N0

%************** Fading initialization ***************

% If you use fading function "sefade", you can
% initialize all of the parameters.
% Otherwise you can comment out the following
% initialization.
% The detailed explanations of all of variables are
% mentioned in Program 2-8.

% Time resolution

tstp=1/sr/(fftlen+gilen);

% Arrival time for each multipath normalized by tstp
% If you would like to simulate under one path fading
% model, you have only to set
% direct wave.

itau = [0];

% Mean power for each multipath normalized by direct
% wave.
% If you would like to simulate under one path fading
% model, you have only to set
% direct wave.
dlvl = [0];

% Number of waves to generate fading for each multipath.
% In normal case, more than six waves are needed to
% generate Rayleigh fading
n0=[6];

% Initial phase of delayed wave
% In this simulation one-path Rayleigh fading is
% considered.
th1=[0.0];

% Number of fading counter to skip
itnd0=nd*(fftlen+gilen)*10;

% Initial value of fading counter
% In this simulation one-path Rayleigh fading is
% considered.
% Therefore one fading counter is needed.
```

```
itnd1=[1000];

% Number of direct wave + Number of delayed wave
% In this simulation one-path Rayleigh fading is
% considered
now1=1;

% Maximum Doppler frequency [Hz]
% You can insert your favorite value
fd=320;

% You can decide two modes to simulate fading by changing
% the variable flat
% flat     : flat fading or not
% (1-flat (only amplitude is fluctuated),0-normal(phase
% and amplitude are fluctuated))
flat =1;

%****************** main loop part ******************

nloop=500;  % Number of simulation loops

noe = 0;    % Number of error data
nod = 0;    % Number of transmitted data
eop=0;      % Number of error packet
nop=0;      % Number of transmitted packet

for iii=1:nloop

%******************** transmitter *******************

%****************** Data generation ******************

seldata=rand(1,para*nd*ml) > 0.5;
  % rand : built in function

%*********** Serial to parallel conversion ***********

paradata=reshape(seldata,para,nd*ml);
  % reshape : built in function

%****************** QPSK modulation ******************

[ich,qch]=qpskmod(paradata,para,nd,ml);
kmod=1/sqrt(2); %  sqrt : built in function
ich1=ich.*kmod;
qch1=qch.*kmod;
```

```
%************************* IFFT ************************

x=ich1+qch1.*i;
y=ifft(x);          %  ifft : built in function
ich2=real(y);       %  real : built in function
qch2=imag(y);       %  imag : built in function

%************** Guard interval insertion **************

[ich3,qch3]= giins(ich2,qch2,fftlen,gilen,nd);
fftlen2=fftlen+gilen;

%*************** Attenuation Calculation **************

spow=sum(ich3.^2+qch3.^2)/nd./para;
  %  sum : built in function
attn=0.5*spow*sr/br*10.^(-ebn0/10);
attn=sqrt(attn);

%******************* Fading channel *******************

% Generated data are fed into a fading simulator
[ifade,qfade]=sefade(ich3,qch3,itau,dlvl,th1,n0,itnd1,...
  now1,length(ich3),tstp,fd,flat);

% Updata fading counter
itnd1 = itnd1+ itnd0;

%********************** Receiver  ********************

%******************* AWGN addition *******************

[ich4,qch4]=comb(ifade,qfade,attn);

%*************** Guard interval removal ***************

[ich5,qch5]= girem(ich4,qch4,fftlen2,gilen,nd);

%***********************  FFT  ***********************

rx=ich5+qch5.*i;
ry=fft(rx);         % fft : built in function
ich6=real(ry);      % real : built in function
qch6=imag(ry);      % imag : built in function

%******************* demodulation *******************

ich7=ich6./kmod;
qch7=qch6./kmod;
[demodata]=qpskdemod(ich7,qch7,para,nd,ml);

%*********** Parallel to serial conversion  ***********
```

```
demodata1=reshape(demodata,1,para*nd*ml);

%**************** Bit Error Rate (BER) ****************

% instantaneous number of error and data

noe2=sum(abs(demodata1-seldata));
  %  sum : built in function
nod2=length(seldata);
  %  length : built in function

% cumulative the number of error and data in noe and nod
noe=noe+noe2;
nod=nod+nod2;

% calculating PER
if noe2~=0
  eop=eop+1;
else
  eop=eop;
end
  eop;
nop=nop+1;

fprintf('%d\t%e\t%d\n',iii,noe2/nod2,eop);
  %  fprintf : built in function

end

%****************** Output result *******************

per=eop/nop;
ber=noe/nod;

fprintf('%f\t%e\t%e\t%d\t\n',ebn0,ber,per,nloop);
fid = fopen('BERofdmfad.dat','a');
fprintf(fid,'%f\t%e\t%e\t%d\t\n',ebn0,ber,per,nloop);
fclose(fid);

%******************** end of file *******************
```

Program 4.3

```
% Program 4-3
% giins.m
%
% Function to insert guard interval into transmission
%  signal
%
% Programmed by T. Yamamura and H. Harada
%
```

```
function [iout,qout]= giins(idata,qdata,fftlen,gilen,nd);

%********************** variables ********************

% idata    : Input Ich data
% qdata    : Input Qch data
% iout     : Output Ich data
% qout     : Output Qch data
% fftlen   : Length of FFT (points)
% gilen    : Length of guard interval (points)

% *****************************************************

idata1=reshape(idata,fftlen,nd);
qdata1=reshape(qdata,fftlen,nd);
idata2=[idata1(fftlen-gilen+1:fftlen,:); idata1];
qdata2=[qdata1(fftlen-gilen+1:fftlen,:); qdata1];

iout=reshape(idata2,1,(fftlen+gilen)*nd);
qout=reshape(qdata2,1,(fftlen+gilen)*nd);

%******************** end of file ********************
```

Program 4.4

```
% Program 4-4
% girem.m
%
% Function to remove guard interval from received signal
%
% Programmed by T. Yamamura and H. Harada
%

function [iout,qout]= girem(idata,qdata,fftlen2,gilen,nd);

%********************** variables ********************

% idata     : Input Ich data
% qdata     : Input Qch data
% iout      : Output Ich data
% qout      : Output Qch data
% fftlen2   : Length of FFT (points)
% gilen     : Length of guard interval (points)
% nd        : Number of OFDM symbols

% *****************************************************

idata2=reshape(idata,fftlen2,nd);
qdata2=reshape(qdata,fftlen2,nd);
iout=idata2(gilen+1:fftlen2,:);
qout=qdata2(gilen+1:fftlen2,:);

%********************* end of file *******************
```

Program 4.5

```
% Program 4-5
% ofdmda.m
%
% Simulation program to realize OFDM transmission system
%
% Programmed by T. Yamamura and H. Harada
% Please refer to Program 4-2
%
% GI data GI data ....(data 6symbols)
%

%%%%%%%%%%%%%%%%%%%%% preparation part %%%%%%%%%%%%%%%%%

para=52;        % Number of parallel channel to transmit
                % (points)
fftlen=64;      % FFT length
noc=53;         % Number of carriers
nd=6;           % Number of information OFDM symbol for
                % one loop
ml=2;           % Modulation level : QPSK
sr=250000;      % OFDM symbol rate (250 ksymbol/s)
br=sr.*ml;      % Bit rate per carrier
gilen=16;       % Length of guard interval (points)
ebn0=3;         % Eb/N0

%%%%%%%%%%%%%%%%%%% fading initialization %%%%%%%%%%%%%%%

tstp=1/sr/(fftlen+gilen);    % Time resolution
itau=[0];             % Arrival time for each multipath
                      % normalized by tstp
dlvl1=[0];            % Mean power for each multipath
                      % normalized by direct wave.
n0=[6];               % Number of waves to generate fading
n0(1),n0(2)
th1=[0.0];            % Initial phase of delayed wave
itnd1=[1000];         % set fading counter
now1=1;               % Number of direct wave + Number of
                      % delayed wave
fd=150;               % Maximum Doppler frequency
flat=0;               % Flat or not (see ofdm_fading.m)
itnd0=nd*(fftlen+gilen)*20;
                      % Number of fading counter to skip

%****************** main loop part *****************

nloop=1000;     % Number of simulation loops

noe = 0;        % Number of error data
nod = 0;        % Number of transmitted data
eop=0;          % Number of error packet
nop=0;          % Number of transmitted packet
```

```
%******************** transmitter *********************

for iii=1:nloop

seridata=rand(1,para*nd*ml) > 0.5;
  %  rand : built in function
paradata=reshape(seridata,para,nd*ml);
  %  reshape : built in function

%----------------- ml modulation ---------------------

[ich,qch]=qpskmod(paradata,para,nd,ml);
kmod=1/sqrt(2);    %  sqrt : built in function
ich=ich.*kmod;
qch=qch.*kmod;

%--------------- data mapping (DC=0)------------------

[ich1,qch1]=crmapping(ich,qch,fftlen,nd);

%---------------------- IFFT ----------------------

x=ich1+qch1.*i;
y=ifft(x);            %  ifft : built in function
ich2=real(y);         %  real : built in function
qch2=imag(y);         %  imag : built in function

%------------- Guard interval insertion -------------

fftlen2=fftlen+gilen;
[ich4,qch4]= giins(ich2,qch2,fftlen,gilen,nd);

%-------------- Attenuation Calculation -------------

spow=sum(ich4.^2+qch4.^2)/nd./para;
  %  sum : built in function
attn=0.5*spow*sr/br*10.^(-ebn0/10);
attn=sqrt(attn);

%****************** fading channel ******************

% If you would like to simulate performance under
% fading, please remove "*"
% from the following four sentences
% [ifade,qfade,ramp,rcos,rsin]=sefade(ich4,qch4,itau,
% dlvl1,th1,n0,itnd1,now1,length(ich4),tstp,fd,flat);
% itnd1 = itnd1+itnd0;    % Updata fading counter
% ich4=ifade;
% qch4=qfade;

%********************* Receiver  ********************

%------------------- AWGN addition -----------------
```

```
[ich5,qch5]=comb(ich4,qch4,attn);

%----- 1 Path Rayleigh Fading perfect compensation -----

% If you would like to simulate performance under perfect
% compensation, please remove "*"
% from the following four sentences
% ifade2=1./ramp.*(rcos(1,:).*ich5+rsin(1,:).*qch5);
% qfade2=1./ramp.*(-rsin(1,:).*ich5+rcos(1,:).*qch5);
% ich5=ifade2;
% qch5=qfade2;

%--------------- Guard interval removal -------------

[ich6,qch6]= girem(ich5,qch5,fftlen2,gilen,nd);

%------------------------ FFT -----------------------

rx=ich6+qch6.*i;
ry=fft(rx);     %  fft : built in function
ich7=real(ry);
qch7=imag(ry);

%-------------------- demodulation -----------------

[ich8,qch8]=crdemapping(ich7,qch7,fftlen,nd);
ich9=ich8./kmod;
qch9=qch8./kmod;
[demodata]=qpskdemod(ich9,qch9,para,nd,ml);

%------------------ error calculation --------------

demodata1=reshape(demodata,1,para*nd*ml);
noe2=sum(abs(demodata1-seridata));
  %  sum : built in function
nod2=length(seridata);
  %  length : built in function

% calculating PER
if noe2~=0
  eop=eop+1;
else
  eop=eop;
end
  eop;
nop=nop+1;

% calculating BER
noe=noe+noe2;
nod=nod+nod2;

fprintf(‘%d\t%e\t%d\n’,iii,noe2/nod2,eop);
  %  fprintf : built in function
```

```
end
per=eop/nop;
ber=noe/nod;

%******************* Output result *******************

fprintf('%f\t%e\t%e\t%d\t%d\n',ebn0,ber,per,nloop,fd);
fid = fopen('BERofdmda.dat','a');
fprintf(fid,'%f\t%e\t%e\t%d\t%d\t\n',ebn0,ber,per,...
  nloop,fd);
fclose(fid);

%******************** end of file ********************
```

Program 4.6

```
% Program 4-6
% crmapping.m
%
% Function to set data on subcarrier
% (DC=0)
%
% Programmed by T. Yamamura and H. Harada
%

function [iout,qout]=crmapping(idata,qdata,fftlen,nd);

%********************* variables ********************

% idata     : Input Ich data
% qdata     : Input Qch data
% iout      : Output Ich data
% qout      : Output Qch data
% fftlen    : Length of FFT (points)
% nd        : Number of OFDM symbols

% ****************************************************

iout=zeros(fftlen,nd);
qout=zeros(fftlen,nd);
iout(2:27,:)=idata(1:26,:);
qout(2:27,:)=qdata(1:26,:);
iout(39:64,:)=idata(27:52,:);
qout(39:64,:)=qdata(27:52,:);

%********************* end of file *******************
```

Program 4.7

```
% Program 4-7
% crdemapping.m
```

```
%
% Function to separate data from the subcarrier
% (DC=0)
%
% MATLAB version
% programmed by T. Yamamura
%

function [iout,qout]=crdemapping(idata,qdata,fftlen,nd);

%********************** variables ********************

% idata      : Input Ich data
% qdata      : Input Qch data
% iout       : Output Ich data
% qout       : Output Qch data
% fftlen     : Length of FFT (points)
% nd         : Number of OFDM symbols

% ****************************************************

iout(1:26,:)=idata(2:27,:);
qout(1:26,:)=qdata(2:27,:);
iout(27:52,:)=idata(39:64,:);
qout(27:52,:)=qdata(39:64,:);

%********************* end of file *******************
```

Program 4.8

```
% Program 4-8
% ofdmce.m
%
% Simulation program to realize OFDM transmission system
%
% Programmed by T. Yamamura and H. Harada
%
% GI CE GI data GI data...(data 6symbols)
% (CE: Channel estimation symbol, GI Guard interval)
%

%***************** preparation part *****************

para=52;     % Number of parallel channel to transmit
fftlen=64;   % FFT length
noc=53;      % Number of carriers
nd=6;        % Number of information OFDM symbol for one
             % loop
knd=1;       % Number of known channel estimation (CE)
             % OFDM symbol
ml=2;        % Modulation level : QPSK
sr=250000;   % OFDM symbol rate (250 ksymbol/s)
```

```
br=sr.*ml;  % Bit rate per carrier
gilen=16;   % Length of guard interval (points)
ebn0=3;     % Eb/N0

%%%%%%%%%%%%%%%%% fading initialization %%%%%%%%%%%%%%%

tstp=1/sr/(fftlen+gilen); % Time resolution
itau=[0];       % Arrival time for each multipath
                % normalized by tstp
dlvl1=[0];      % Mean power for each multipath
                % normalized by direct wave.
n0=[6];         % Number of waves to generate fading
                % n0(1),n0(2)
th1=[0.0];      % Initial phase of delayed wave
itnd1=[1000];   % set fading counter
now1=1;         % Number of direct wave + Number of delayed
                % wave
fd=150;         % Maximum Doppler frequency
flat=0;         % Flat or not (see ofdm_fading.m)
itnd0=nd*(fftlen+gilen)*20;
                % Number of fading counter to skip

%****************** main loop part ******************

nloop=1000; % Number of simulation loops

noe = 0;    % Number of error data
nod = 0;    % Number of transmitted data
eop=0;      % Number of error packet
nop=0;      % Number of transmitted packet

%******************** transmitter *******************

for iii=1:nloop

seridata=rand(1,para*nd*ml) > 0.5;

paradata=reshape(seridata,para,nd*ml);
  %size(51  *  nd*ml)

%-------------------- ml modulation ----------------

[ich,qch]=qpskmod(paradata,para,nd,ml);
kmod=1/sqrt(2);
ich=ich.*kmod;
qch=qch.*kmod;

% CE data generation
kndata=zeros(1,fftlen);
kndata0=2.*(rand(1,52) < 0.5)-1;
kndata(2:27)=kndata0(1:26);
kndata(39:64)=kndata0(27:52);
```

```
ceich=kndata; % CE:BPSK
ceqch=zeros(1,64);

%---------------- data mapping (DC=0)----------------

[ich1,qch1]=crmapping(ich,qch,fftlen,nd);

ich2=[ceich.' ich1]; % I-channel transmission data
qch2=[ceqch.' qch1]; % Q-channel transmission data

%----------------------- IFFT -----------------------

x=ich2+qch2.*i;
y=ifft(x);
ich3=real(y);
qch3=imag(y);

%-------------- Guard interval insertion ------------

fftlen2=fftlen+gilen;
[ich4,qch4]= giins(ich3,qch3,fftlen,gilen,nd+1);

%-------------- Attenuation Calculation -------------

spow=sum(ich4.^2+qch4.^2)/nd./para;
attn=0.5*spow*sr/br*10.^(-ebn0/10);
attn=sqrt(attn);

%***************** fading channel *******************

% If you would like to simulate performance under fading,
% please remove "*"
% from the following four sentences
% [ifade,qfade,ramp,rcos,rsin]=sefade(ich4,qch4,itau,
% dlvl1,th1,n0,itnd1,now1,length(ich4),tstp,fd,flat);
% itnd1 = itnd1+itnd0;  % Updata fading counter
% ich4=ifade;
% qch4=qfade;

%******************** Receiver  *********************

%------------------ AWGN addition -------------------

[ich5,qch5]=comb(ich4,qch4,attn);

%--- Perfect fading compensation for one path fading ---

% If you would like to simulate performance under perfect
% compensation, please remove "*"
% from the following four sentences
% ifade2=1./ramp.*(rcos(1,:).*ich5+rsin(1,:).*qch5);
% qfade2=1./ramp.*(-rsin(1,:).*ich5+rcos(1,:).*qch5);
% ich5=ifade2;
% qch5=qfade2;

%--------------- Guard interval removal ---------------

[ich6,qch6]= girem(ich5,qch5,fftlen2,gilen,nd+1);

%------------------------ FFT -----------------------

rx=ich6+qch6.*i;
ry=fft(rx);
ich7=real(ry);
qch7=imag(ry);

%---------- Fading compensation by CE symbol ----------

%
% If you would like to simulate performance under
% CE-based compensation, please remove "*"
% in this area
%

% preparation known CE data
% ce=1;
% ice0=ich2(:,ce);
% qce0=qch2(:,ce);

% taking CE data out of received data
% ice1=ich7(:,ce);
% qce1=qch7(:,ce);

% calculating reverse rotation
% iv=real((1./(ice1.^2+qce1.^2)).*(ice0+i.*qce0)...
% .*(ice1-i.*qce1));
% qv=imag((1./(ice1.^2+qce1.^2)).*(ice0+i.*qce0)...
% .*(ice1-i.*qce1));

% matrix for reverse rotation
% ieqv1=[iv iv iv iv iv iv iv];
% qeqv1=[qv qv qv qv qv qv qv];

% reverse rotation
% icompen=real((ich7+i.*qch7).*(ieqv1+i.*qeqv1));
% qcompen=imag((ich7+i.*qch7).*(ieqv1+i.*qeqv1));
% ich7=icompen;
% qch7=qcompen;

%----------------- CE symbol removal ----------------

ich8=ich7(:,knd+1:nd+1);
qch8=qch7(:,knd+1:nd+1);
```

```
% DC and pilot data removal
[ich9,qch9]=crdemapping(ich8,qch8,fftlen,nd);

%------------------- demodulation -------------------

ich10=ich9./kmod;
qch10=qch9./kmod;
[demodata]=qpskdemod(ich10,qch10,para,nd,ml);

%------------------ error calculation ----------------

demodata1=reshape(demodata,1,para*nd*ml);
noe2=sum(abs(demodata1-seridata));
nod2=length(seridata);

% calculating PER
if noe2~=0
  eop=eop+1;
else
  eop=eop;
end
  eop;
nop=nop+1;

% calculating BER
noe=noe+noe2;
nod=nod+nod2;

fprintf('%d\t%e\t%d\n',iii,noe2/nod2,eop);

end

per=eop/nop;
ber=noe/nod;

%****************** Output result *******************

fprintf('%f\t%e\t%e\t%d\t%d\n',ebn0,ber,per,nloop,fd);

fid = fopen('BERofdmce.dat','a');
fprintf(fid,'%f\t%e\t%e\t%d\t\n',ebn0,ber,per,nloop);
fclose(fid);

%******************** end of file *******************
```

Program 4.9

```
% Program 4-9
% ofdmci.m
%
% Simulation program to realize OFDM transmission system
% Simulate effect of interference noise
```

```
%
% Programmed by T. Yamamura and H. Harada
%
% GI CE GI data GI data...(data 6symbols)
%

%****************** preparation part *****************

para=52;         % Number of parallel channel to transmit
                 % (points)
fftlen=64;       % FFT length
noc=53;          % Number of carriers
nd=6;            % Number of information OFDM symbol for
                 % one loop
knd=1;           % Number of known channel estimation
                 % (CE) OFDM symbol
ml=2;            % Modulation level : QPSK
sr=250000;       % OFDM symbol rate (250 ksymbol/s)
br=sr.*ml;       % Bit rate per carrier
gilen=16;        % Length of guard interval (points)
ebn0=1000;       % Eb/N0

%--------------- fading initialization --------------

tstp=1/sr/(fftlen+gilen); % Time resolution
itau=[0];        % Arrival time for each multipath
                 % normalized by tstp
dlvl1=[0];       % Mean power for each multipath
                 % normalized by direct wave.
n0=[6];          % Number of waves to generate fading
                 % n0(1),n0(2)
th1=[0.0];       % Initial phase of delayed wave
itnd1=[1000];    % set fading counter
now1=1;          % Number of direct wave + Number of
                 % delayed wave
fd=150;          % Maximum Doppler frequency
flat=0;          % Flat or not (see ofdm_fading.m)
itnd0=nd*(fftlen+gilen)*10;
                 % Number of fading counter to skip

%---------- interference wave initialization ----------

ci=10;      % C/I ratio
ml2=2;      % modulation level

itau2=[0];
dlvl2=[0];
n02=[6];
th2=[0.0];
itnd2=[10000+floor(rand(1)*10)*1000];
now2=1;
fd2=fd;
lat2=0;
```

```
itnd02=nd*(fftlen+gilen)*300;
  % Number of fading counter to skip

%% store all parameters in one matrix "fadingpara"

fadingpara=zeros(8,length(itau2));
fadingpara(1,:)=itau2;
fadingpara(2,:)=dlvl2;
fadingpara(3,:)=n02;
fadingpara(4,:)=th2;
fadingpara(5,:)=itnd2;
fadingpara(6,:)=now2;
fadingpara(7,:)=fd2;
fadingpara(8,:)=flat2;

%****************** main loop part ******************

nloop=1000;      % Number of simulation loops

noe = 0;         % Number of error data
nod = 0;         % Number of transmitted data
eop=0;           % Number of error packet
nop=0;           % Number of transmitted packet

%******************** transmitter *******************

for iii=1:nloop

seldata=rand(1,para*nd*ml) > 0.5;
paradata=reshape(seldata,para,nd*ml);
  %size(51  *  nd*ml)

%------------------- ml modulation -----------------

[ich,qch]=qpskmod(paradata,para,nd,ml);
kmod=1/sqrt(2);
ich=ich.*kmod;
qch=qch.*kmod;

% CE data generation
kndata=zeros(1,fftlen);
kndata0=2.*(rand(1,52) < 0.5)-1;
kndata(2:27)=kndata0(1:26);
kndata(39:64)=kndata0(27:52);
ceich=kndata; % CE:BPSK
ceqch=zeros(1,64);

%---------------- data mapping (DC=0)----------------

[ich1,qch1]=crmapping(ich,qch,fftlen,nd);

ich2=[ceich.' ich1]; % I-channel transmission data
qch2=[ceqch.' qch1]; % Q-channel transmission data

%---------------------- IFFT -----------------------
```

```
x=ich2+qch2.*i;
y=ifft(x);
ich3=real(y);
qch3=imag(y);

%------------- Guard interval insertion -------------

fftlen2=fftlen+gilen;
[ich4,qch4]= giins(ich3,qch3,fftlen,gilen,nd+1);

%-------------- Attenuation Calculation -------------

spow=sum(ich4.^2+qch4.^2)/nd./52;
attn=0.5*spow*sr/br*10.^(-ebn0/10);
attn=sqrt(attn);

%****************** fading channel ******************

% If you would like to simulate performance under fading,
% please remove "*"
% from the following four sentences
[ifade,qfade,ramp,rcos,rsin]=sefade(ich4,qch4,itau,
  dlvl1,th1,n0,itnd1,now1,length(ich4),tstp,fd,flat);
itnd1 = itnd1+itnd0;  % Updata fading counter
ich4=ifade;
qch4=qfade;

%%% interference wave addition
% interference
[iintw,qintw]=interwave(ci,spow,ml2,length(ich4),tstp,...
  fadingpara);
itnd2 = itnd2+itnd02;
fadingpara(5,:)=itnd2;
ich4=ich4+iintw;
qch4=qch4+qintw;

%******************** Receiver  *********************

%------------------ AWGN addition ------------------

[ich5,qch5]=comb(ich4,qch4,attn);

%--- Perfect fading compensation for one path fading ---

% If you would like to simulate performance under perfect
% compensation, please remove "*"
% from the following four sentences
% ifade2=1./ramp.*(rcos(1,:).*ich5+rsin(1,:).*qch5);
% qfade2=1./ramp.*(-rsin(1,:).*ich5+rcos(1,:).*qch5);
% ich5=ifade2;
% qch5=qfade2;

%-------------- Guard interval removal -------------

[ich6,qch6]= girem(ich5,qch5,fftlen2,gilen,nd+1);
```

```
%---------------------- FFT ----------------------

rx=ich6+qch6.*i;
ry=fft(rx);
ich7=real(ry);
qch7=imag(ry);

%---------- Fading compensation by CE symbol ---------

%
% If you would like to simulate performance under
% CE-based compensation, please remove "*"
% in this area
%

% preparation known CE data
ce=1;
ice0=ich2(:,ce);
qce0=qch2(:,ce);

% taking CE data out of received data
ice1=ich7(:,ce);
qce1=qch7(:,ce);

% calculating reverse rotation
iv=real((1./(ice1.^2+qce1.^2)).*(ice0+i.*qce0)...
   .*(ice1-i.*qce1));
qv=imag((1./(ice1.^2+qce1.^2)).*(ice0+i.*qce0)...
   .*(ice1-i.*qce1));

% matrix for reverse rotation
ieqv1=[iv iv iv iv iv iv iv];
qeqv1=[qv qv qv qv qv qv qv];

% reverse rotation
icompen=real((ich7+i.*qch7).*(ieqv1+i.*qeqv1));
qcompen=imag((ich7+i.*qch7).*(ieqv1+i.*qeqv1));
ich7=icompen;
qch7=qcompen;

%---------------- CE symbol removal ---------------

ich8=ich7(:,knd+1:nd+1);
qch8=qch7(:,knd+1:nd+1);

%------------- DC and pilot data removal ------------

[ich9,qch9]=crdemapping(ich8,qch8,fftlen,nd);

%------------------ demodulation ------------------

ich10=ich9./kmod;
qch10=qch9./kmod;
[demodata]=qpskdemod(ich10,qch10,para,nd,ml);

%---------------- error calculation ---------------
```

```
demodata1=reshape(demodata,1,para*nd*ml);
noe2=sum(abs(demodata1-seldata));
nod2=length(seldata);

% calculating PER
if noe2~=0
  eop=eop+1;
else
  eop=eop;
end
  eop;
nop=nop+1;

% calculating BER
noe=noe+noe2;
nod=nod+nod2;

fprintf('%d\t%e\t%d\n',iii,noe2/nod2,eop);

end
per=eop/nop;
ber=noe/nod;

%******************* Output result *******************

fprintf('%f\t%e\t%e\t%d\t%d\n',ci,ber,per,nloop,fd);

fid = fopen('BERofdmci.dat','a');
fprintf(fid,'%f\t%e\t%e\t%d\t%d\n',ci,ber,per,nloop,fd);
fclose(fid);

%********************* end of file *******************
```

Program 4.10

```
% Program 4-10
% interwave.m
%
% Function to add interference wave
%
% Programmed by T. Yamamura and H. Harada
%

function [iout,qout]=interwave(ci,spow,ml,nsamp,tstp,...
  fadingpara);

%********************* variables ********************

% ci          : Carrier to interference ratio
% spow        : Power of desired signal
% ml          : Modulation level
% nsamp       : Number of samples
```

```
% tstp        : Time resolution
% fadingpara : Fading parameter
% iout        : Output Ich signal
% qout        : Output Qch signal

% ****************************************************

itau=fadingpara(1,:);
dlvl1=fadingpara(2,:);
n0=fadingpara(3,:);
th1=fadingpara(4,:);
itnd1=fadingpara(5,:);
now1=fadingpara(6,:);
fd=fadingpara(7,:);
flat=fadingpara(8,:);
if ci > 40;

%%%%%%%%%%%%%%%%%% preparation part %%%%%%%%%%%%%%%%%%

%%% frame format
para=52;
fftlen=64;
noc=53;       % the number of carrier
nd=6;         % the number of information symbol
knd=1;        % the number of known data symbol

sr=250000;  % symbol rate
br=sr.*ml;  % bit rate per carrier
gilen=16;   % the length of guard interval

%%% Set CE data load
kndata=zeros(1,fftlen);
kndata0=2.*(rand(1,52)0.5)-1;
kndata(2:27)=kndata0(1:26);
kndata(39:64)=kndata0(27:52);

%%% Simulation start
%%% fading initialization

%%%%%%%%%%%%%%%%%%%%% transmitter %%%%%%%%%%%%%%%%%%%%%

seridata=rand(1,para*nd*ml) > 0.5;  %  DC=0

paradata=reshape(seridata,para,nd*ml);
  % size(51 * nd*ml)

%%% ml modulation

[ich,qch]=qpskmod(paradata,para,nd,ml);
kmod=1/sqrt(2);
ich=ich.*kmod;
qch=qch.*kmod;
```

```
% CE modulation

ceich=kndata; % CE:BPSK
ceqch=zeros(1,64);

%%% data mapping (DC=0)

[ich2,qch2]=crmapping(ich,qch,fftlen,nd);

% addition of pilot carrier and CE symbol

ich22=[ceich.' ich2]; % I-channel transmission data
qch22=[ceqch.' qch2]; % Q-channel transmission data

%%% IFFT

x=ich22+qch22.*i;
y=ifft(x);
ich3=real(y);
qch3=imag(y);

%%% Guard interval insertion

% guard interval insertion
[ich5,qch5]= giins(ich3,qch3,fftlen,gilen,nd+1);

%%% fading calculation
[ifade2,qfade2,ramp,rcos,rsin]=sefade(ich5,qch5,itau,...
  dlvl1,th1,n0,itnd1,now1,length(ich5),tstp,fd,flat);

%%% C/I reduction
spowintw=sum(ich5.^2+qch5.^2)/(nd)/52;
rint=spow/spowintw*10^(-ci/10);
iout=ifade2.*sqrt(rint);
qout=qfade2.*sqrt(rint);

else

iout=zeros(1,nsamp);
qout=zeros(1,nsamp);

end

%********************** end of file ********************
```

5

Code Division Multiple Access (CDMA) Transmission Technology

5.1 Introduction

CDMA protocols constitute a class of protocols in which multiple-access capability is primarily achieved by means of coding. In CDMA, each user is assigned a unique code sequence that he or she uses to encode his or her information signal. The receiver, knowing the code sequences of the user, decodes the received signal after reception and recovers the original data. Because the bandwidth of the code signal is chosen to be much larger than the bandwidth of the information signal, the encoding process enlarges (spreads) the spectrum of the signal and is therefore also known as *spread-spectrum* (SS) modulation. The resulting encoded signal is also called an SS signal, and CDMA protocols are often denoted as *SS multiple-access* (SSMA) protocols.

The origin of SS communications may be difficult to pinpoint because the modern SS communication is an outcome of many developments, including high-resolution radars, direction finding, guidance, correlation detection, matched filtering, interference rejection, jamming avoidance, information theory, and secured communications [1–10].

SS modulation techniques were originally developed for use in military radar and communication systems because of their resistance to jamming signals and low probability of detection. Only in recent years, with the development of new, inexpensive technologies and a decrease of the military market have researchers and manufacturers of SS equipment become interested in the civil applications of their techniques.

For a technique to be classified as an SS modulation technique, two criteria must be met [9]:

1. The transmission bandwidth must be much larger than the information bandwidth.
2. The resulting radio-frequency bandwidth must be determined by a function other than the information being sent. This excludes such modulation techniques as FM and PM.

SS modulation transforms an information signal into a transmission signal with a much larger bandwidth. This transformation is achieved by encoding the information signal with a code signal that is independent of the data and has a much larger spectral width than that of the data signal. This encoding spreads the original signal power over a much broader bandwidth, which results in a low power density. The ratio of the transmitted bandwidth to the information bandwidth is called a processing gain, G_p, of the SS system,

$$G_p = \frac{B_t}{B_i} \tag{5.1}$$

where B_t is the transmission bandwidth, and B_i is the bandwidth of the information signal.

The receiver correlates the received signal with a synchronously generated replica of the code signal used at the transmitter to recover the original information signal. This implies that the receiver must know the code signal used to modulate the data.

Because of the coding and the resulting enlarged bandwidth, SS signals have a number of properties that differ from the properties of narrowband signals. The most interesting ones from the communication systems point of view are discussed next. For better understanding, each property is briefly explained with the help of illustrations:

1. *Multiple-access capability:* If multiple users transmit an SS signal at the same time, the receiver will still be able to distinguish between the users if each user has a unique code that has a sufficiently low cross-correlation with the other codes. Correlating the received signal with the code signal from a certain user will then only despread the signal of this user, while the other SS signals will remain spread over a large bandwidth. Thus, within the information bandwidth, the power of the desired user will be much larger than the interfering power provided there are not too many interferers, and the desired signal can be extracted. The multiple-access capability is illustrated in Figure 5.1. In Figure 5.1(a), two users generate an SS signal from their narrowband data signals. In Figure 5.1(b), both users transmit their SS signals at the same time. At the receiver, only the signal of user 1 is "despread" and the data recovered.

2. *Protection against multipath interference:* In a radio channel, there is not just one path between a transmitter and a receiver. Due to reflections (and refractions), a signal will be received from a number of different paths. The signals of the different paths are all copies of the transmitted signal but with different amplitudes and phases. Adding these signals at the receiver will be constructive at some of the frequencies and destructive at others. In the time domain, this results in a dispersed signal. SS modulation can combat this multipath interference; however, the way in which this is done depends very much on the type of modulation used. In Section 5.2, where CDMA protocols based on different modulation methods are discussed, we will show for each protocol how multipath interference rejection is obtained.
3. *Privacy:* The transmitted signal can only be despread and the data recovered if the code is known to the receiver.
4. *Interference rejection:* Cross-correlating the code signal with a narrowband signal will spread the power of the narrowband signal thereby reducing the interfering power in the information bandwidth. This is illustrated in Figure 5.2. The SS signal (s) receives a narrowband interference (i). At the receiver, the SS signal is "despread" while the

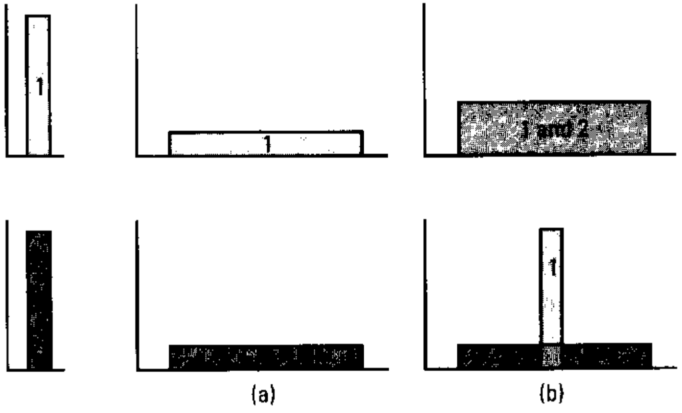


Figure 5.1 Principles of SS multiple access: (a) spread and (b) despread.

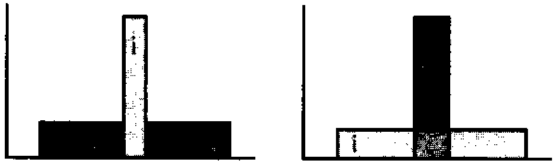


Figure 5.2 Interference rejection.

interference signal is spread, which makes it appear as background noise compared to the despread signal.

5. *Antijamming capability, especially narrowband jamming:* This is more or less the same as interference rejection. This property makes SS modulation attractive for military applications.
6. *Low probability of interception (LPI), or covert operation:* Because of its low power density, the SS signal is difficult to detect.

There are a number of modulation techniques that generate SS signals. We will briefly discuss the most important ones.

The chapter is organized as follows. First, the concept behind the CDMA scheme is described in Section 5.2. Section 5.3 presents a method for generating a spreading code by introducing three representative code sequences, namely, an M-sequence, a Gold sequence, and an orthogonal Gold sequence. Section 5.4 discusses the configuration of a transmitter and a receiver by using MATLAB program. Section 5.5 shows the BER performance in an AWGN channel and a Rayleigh fading environment. Section 5.6 concludes the chapter. All programs related to the chapter are presented in Appendix 5A and in the CD-ROM that accompanies the book.

5.2 Concept Behind CDMA Transmission Scheme

SSMA, or CDMA, protocols can be classified in two different ways: by the concept behind the protocols or by the modulation method used. The first classification gives us two groups of protocols, namely, averaging systems and avoidance systems. Averaging systems reduce interference by averaging the interference over a long time interval. Avoidance systems reduce interference by avoiding it for a large part of the time.

Classifying the CDMA protocols by the modulation method gives us five groups of protocols: (1) *direct-sequence* (DS) (or pseudo-noise), (2) frequency-hopping, (3) time-hopping, (4) chirp SS, and (5) hybrid. Of these, the first (DS) is an averaging SS protocol, and hybrid protocols can be averaging protocols depending on whether DS is used as part of the hybrid method. All the other protocols are avoidance protocols. Table 5.1 illustrates both ways of classification.

Sections 5.2.1–5.2.5 discuss the CDMA protocols in terms of the modulation technique on which they are based.

5.2.1 Direct Sequence

In the DS-CDMA protocols, the modulated information signal (the data signal) is directly modulated by a digital code signal. The data signal can be either an analog or a digital signal. In most cases, it is digital. In the case of a digital signal, the data modulation is often omitted, the data signal is directly multiplied by the code signal, and the resulting signal modulates the wideband carrier. It is from this direct multiplication that the DS-CDMA protocol gets its name.

Figure 5.3 shows a block diagram of a DS-CDMA transmitter. The binary data signal modulates an RF carrier. The modulated carrier is then modulated by the code signal. The code signal consists of a number of code bits called "chips" that can be either +1 or −1. To obtain the desired spreading of

Table 5.1
Classifying SSMA Protocols

	DS	TH	FH	Chirp	Hybrid
Averaging	x				x
Avoidance		x	x	x	x

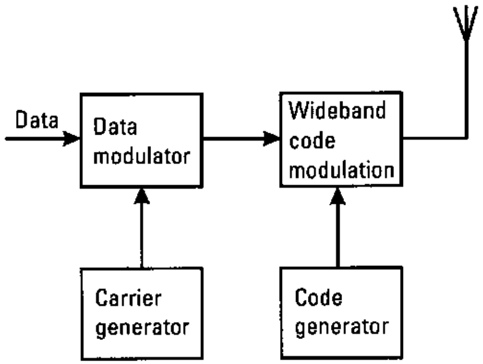


Figure 5.3 Block diagram of a DS-SSMA transmitter.

the signal, the chip rate of the code signal must be much higher than the chip rate of the information signal. For the code modulation, various modulation techniques can be used, but forms of PSK, such as BPSK, or MSK are employed. If we omit the data modulation and use BPSK for the code modulation, we get the block diagram shown in Figure 5.4.

The DS-SS signal resulting from this transmitter is shown in Figure 5.5. In the figure, 10 code chips per information signal are transmitted (the code chip rate is 10 times the information chip rate), so the processing gain is equal to 10. In practice, the processing gain will be much larger.

After transmission of the signal, the receiver (which can be seen in Figure 5.6) uses coherent demodulation to despread the SS signal by using a locally generated code sequence. To be able to perform the despreading operation, the receiver must not only know the code sequence used to spread the signal, but

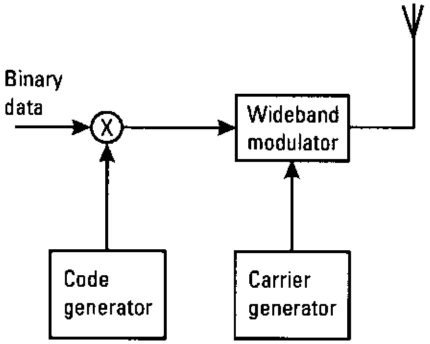


Figure 5.4 Modified block diagram of a DS-SS transmitter.

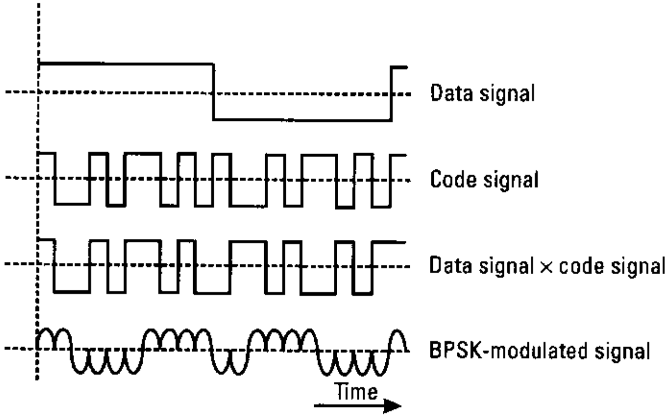


Figure 5.5 Generation of a BPSK-modulated SS signal.

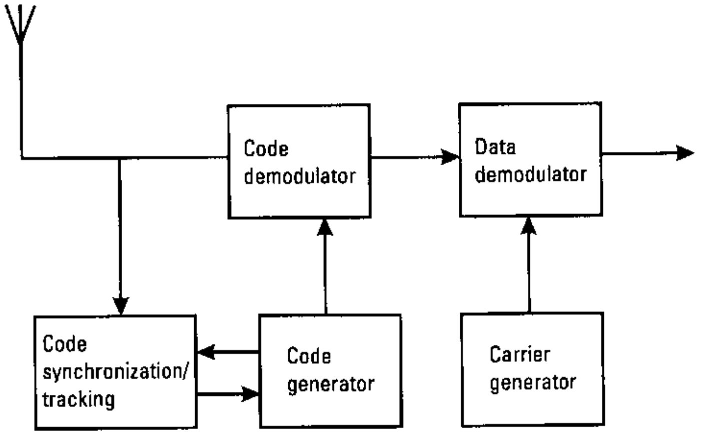


Figure 5.6 Receiver of a DS-SS signal.

also synchronize the codes of the received signal and the locally generated code. This synchronization must be accomplished at the beginning of the reception and maintained until the whole signal has been received. A synchronization/tracking block performs this operation. After despreading, a data-modulated signal is generated and after demodulation, the original data can be recovered.

Section 5.1 discusses a number of the advantageous properties of SS signals. The most important of those properties from the viewpoint of CDMA are the multiple-access capability, multipath interference rejection, narrowband interference rejection, and, with respect to secure/private communication, the LPI. We will now discuss these four properties for the case of DS-CDMA.

- *Multiple access:* If multiple users use the same channel at the same time, there will be multiple DS signals overlapping in time and frequency. At the receiver, coherent demodulation is used to remove the code modulation. This operation concentrates the power of the desired user in the information bandwidth. If the cross-correlation between the code of the desired user and the code of the interfering user is small, coherent detection will put only a small part of the power of the interfering signals in the information bandwidth.
- *Multipath interference:* If the code sequence has an ideal autocorrelation function, then the correlation function is zero outside the interval $[-T_c, T_c]$ where T_c is the chip duration. This means that if the desired signal and the copied signal that is delayed for more than $2T_c$ are received, coherent demodulation will treat the delayed signal as an

interfering signal, putting only a small part of the power of this signal in the information bandwidth.

- *Narrowband interference:* The coherent detection at the receiver involves a multiplication of the received signal by a locally generated code sequence. However, as we saw at the transmitter, multiplying a narrowband signal by a wideband code sequence spreads the spectrum of the narrowband signal so that its power in the information bandwidth decreases by a factor equal to the processing gain.
- *LPI:* Because the DS signal uses the whole signal spectrum all the time, it will have a very low transmitted power per hertz. This makes it very difficult to detect a DS signal.

Apart from the above-mentioned properties, the DS-CDMA protocols have a number of other specific properties that we can divide into advantageous (+) and disadvantageous (–) ones:

+ The generation of the coded signal is easy. It can be done by a simple multiplication.

+ Because only one carrier frequency has to be generated, the frequency synthesizer (carrier generator) is simple.

+ Coherent demodulation of the SS signal is possible.

+ No synchronization among users is necessary.

– Obtaining and maintaining the synchronization of the locally generated code signal and received signal are difficult. Synchronization has to take place within a fraction of the chip time.

– For correct reception, the locally generated code sequence and the received code sequence must be synchronized within a fraction of the chip time. Combined with the nonavailability of large contiguous frequency bands, this property practically limits the spread bandwidth to 10–20 MHz.

– The power received from users close to the base station is much higher than that received from users farther away. Because a user continuously transmits over the whole bandwidth, a user close to the base station will be constantly creating a lot of interference for users far from the base station, making their reception impossible. This near-far effect can be eliminated by using a power control algorithm so that the signals of all users can be received by the base station with the same average power. However, this control mechanism, which is called power control, is quite difficult to attain.

5.2.2 Frequency Hopping

In *frequency-hopping* (FH) CDMA (FH-CDMA) protocols, the carrier frequency of the modulated information signal is not constant but changes periodically. During time intervals *T*, the carrier frequency remains the same, but after each time interval the carrier hops to another (or possibly the same) frequency. The hopping pattern is determined by the code signal. The set of available frequencies that the carrier can attain is called a hop set. The frequency occupation of an FH-SS system differs considerably from that of a DS-SS system. A DS system occupies the whole frequency band when it transmits, whereas an FH system uses only a small part of the bandwidth when it transmits, but the location of this part differs in time.

Suppose that an FH system is transmitting in a narrow frequency band (see Figure 5.7). A DS system transmitting in the same time period spreads its signal power over the whole frequency band and as a result, the power transmitted in this frequency band is much lower than that of the FH system. However, the DS system transmits in the frequency band during all time periods, whereas the FH system uses this band only part of the time. On average, both systems will transmit the same power in the frequency band. The difference between the DS-SS and FH-SS systems is illustrated in Figure 5.7. Figure 5.8 shows a block diagram of an FH-SS transmitter and an FH-SS receiver.

The data signal is baseband modulated on a carrier. Several modulation techniques can be used for the modulation, and it does not really matter which one is used for the application of frequency hopping. Usually, FM modulation is used for analog signals, and FSK modulation is used for digital signals. Using a fast frequency synthesizer controlled by the code signal, the carrier frequency is converted up to the transmission frequency.

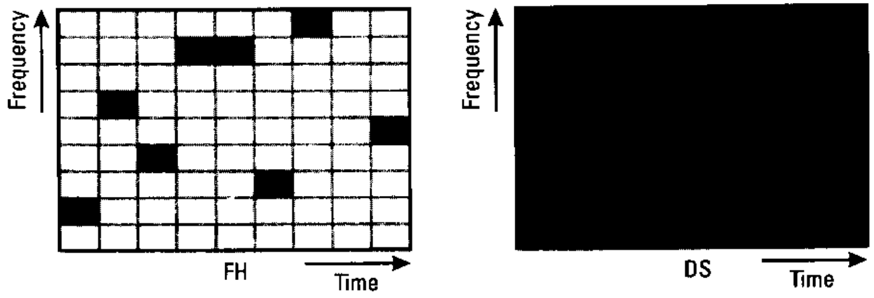


Figure 5.7 Time/frequency occupancy of FH and DS signals.

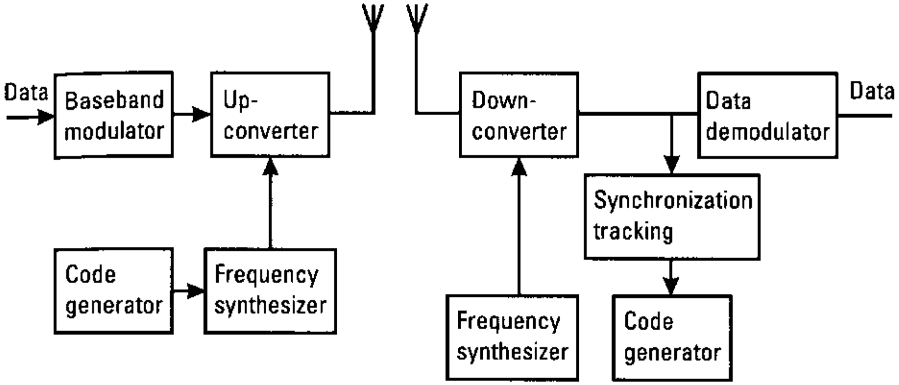


Figure 5.8 Block diagram of an FH-SS transmitter and receiver.

The inverse process takes place at the receiver. Using a locally generated sequence, the received signal is converted down to the baseband-modulated carrier. The data are recovered after the (baseband) demodulation. The synchronization/tracking circuit ensures that the hopping of the locally generated carrier is synchronized with the hopping pattern of the received carrier, so that the signal is despread correctly.

Within the FH-CDMA protocols, a distinction is made that is based on the hopping rate of the carrier. If the number of hops is (much) greater than the data rate, one speaks of a *fast FH* (F-FH) CDMA protocol. In this case, the carrier frequency changes a number of times during the transmission of one bit, so that one bit is transmitted at different frequencies. If the number of hops is (much) smaller than the data rate, one speaks of *slow FH* (S-FH) CDMA protocols. In this case, multiple bits are transmitted at the same frequency.

The occupied bandwidth of the signal on one of the hopping frequencies depends not only on the bandwidth of the information signal but also on the shape of the hopping signal and the hopping frequency. If the hopping frequency is much smaller than the information bandwidth (which is the case in S-FH), then the information bandwidth is the main factor that determines the occupied bandwidth. If, however, the hopping frequency is much greater than the information bandwidth, the pulse shape of the hopping signal will determine the occupied bandwidth at one hopping frequency. If this pulse shape is very abrupt (resulting in very abrupt frequency changes), the frequency band will be very broad, limiting the number of hop frequencies. If we make sure that the frequency changes are smooth, the frequency band at each hopping frequency will be about $1/T_h$ times the frequency bandwidth where T_h is equal to the hopping frequency. We can make the frequency changes smooth by decreasing the transmitted power before a frequency hop and increasing it again when the hopping frequency has changed.

As has been done for the DS-CDMA protocols, we will discuss the properties of FH-CDMA with respect to multiple access capability, multipath interference rejection, narrowband interference rejection, and probability of interception.

- *Multiple access:* It is quite easy to understand how the F-FH and S-FH CDMA protocols obtain their multiple access capability. In the F-FH protocol, one bit is transmitted in different frequency bands. If the desired user is the only one to transmit in most of the frequency bands, the received power of the desired signal will be much higher than the interfering power, and the signal will be received correctly. In the S-FH protocol, multiple bits are transmitted at one frequency. If the probability of other users transmitting in the same frequency band is rather low, the desired user is signed correctly most of the time. For the times when interfering users transmit in the same frequency band, error-correcting codes are used to recover the data transmitted during that period.
- *Multipath interference:* In the F-FH CDMA protocol, the carrier frequency changes a number of times during the transmission of one bit. Thus, a particular signal frequency will be modulated and transmitted on a number of carrier frequencies. The multipath effect is different at the different carrier frequencies. As a result, signal frequencies that are amplified at one carrier frequency will be attenuated at another carrier frequency and vice versa. At the receiver, the responses at the different hopping frequencies are averaged, thus reducing the multipath interference. This is not as effective as the multipath interference rejection in a DS-CDMA system but it does improve the transmission.
- *Narrowband interference:* Suppose that a narrowband signal is interfering on one of the hopping frequencies. If there are G_p hopping frequencies (where G_p is the processing gain), the desired user will (on average) use the hopping frequency where the interferer is located $1/G_p$ percent of the time. The interference is therefore reduced by a factor of G_p.
- *LPI:* The difficulty in intercepting an FH signal lies not in its low transmission power. During a transmission, it uses as much power per hertz as does a continuous transmission. However, the frequency at which the signal is going to be transmitted is unknown and the duration of the transmission at a particular frequency is quite small.

Therefore, although the signal is more readily intercepted than a DS signal, it is still a difficult task to perform.

Apart from the above-mentioned properties, the FH-CDMA protocols have a number of other specific properties that we can divide into advantageous (+) and disadvantageous (–) ones:

+ Synchronization is much easier with FH-CDMA than it is with DS-CDMA. With FH-CDMA, synchronization has to be within a fraction of the hop time. Because spectral spreading is obtained not by using a very high hopping frequency but by using a large hop set, the hop time will be much longer than the chip time of a DS-CDMA system. Thus, an FH-CDMA system allows for a larger synchronization error.
+ The different frequency bands that an FH signal can occupy do not have to be contiguous, because we can make the frequency synthesizer easily skip over certain parts of the spectrum. Combined with the easier synchronization, this allows for much wider SS bandwidths.
+ Because FH-CDMA is an avoidance SS system, the probability of multiple users transmitting in the same frequency band at the same time is small. A user transmitting far from the base station will be received by the base station even if users close to the base station are transmitting, because those users will probably be transmitting at other frequencies. Thus, the near-far performance of FH-CDMA is much better than that of DS.
+ Because of the larger possible bandwidth an FH system can use, it offers a greater possible reduction of narrowband interference than does a DS system.
– A highly sophisticated frequency synthesizer is necessary.
– An abrupt change of the signal when changing frequency bands will lead to an increase in the frequency band occupied. To avoid this, the signal has to be turned off and on when changing frequency.
– Coherent demodulation is difficult because of the problems in maintaining phase relationships during hopping.

5.2.3 Time Hopping

In *time-hopping* (TH) CDMA (TH-CDMA) protocols, the data signal is transmitted in rapid bursts at time intervals determined by the code assigned to the user.

The time axis is divided into frames, and each frame is divided into *M* time slots. During each frame, the user will transmit in one of the *M* time slots. Which of the *M* time slots is transmitted depends on the code signal assigned to the user. Because a user transmits all of his or her data in 1, instead of the *M* time slots, the frequency the user needs for the transmission increases by a factor of *M*. A block diagram of a TH-CDMA system is shown in Figure 5.9. Figure 5.10 shows a time-frequency plot of the TH-CDMA system. Comparing Figure 5.10 with Figure 5.7, we see that the TH-CDMA protocol uses the whole wideband spectrum for short periods instead of using parts of the spectrum all of the time.

Following the same procedure as for the previously described CDMA protocols, we will discuss the properties of TH-CDMA with respect to

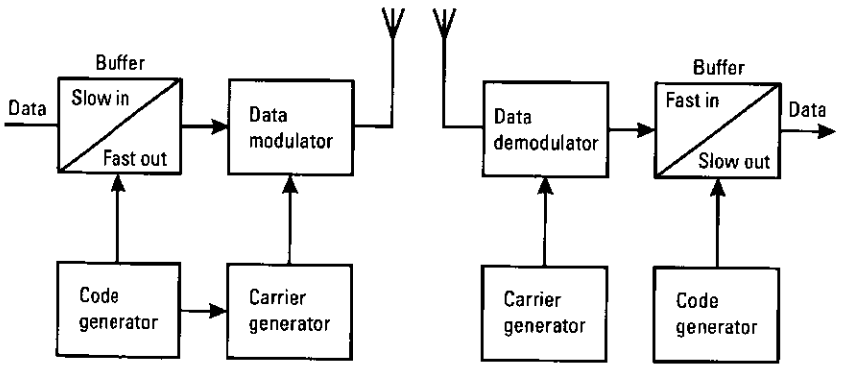


Figure 5.9 Block diagram of a TH-CDMA transmitter and a TH-CDMA receiver.

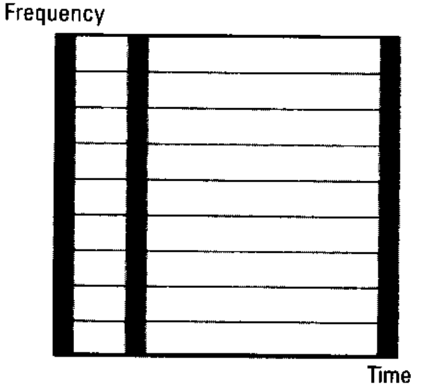


Figure 5.10 Time-frequency plot of the TH-CDMA protocol.

multiple-access capability, multipath interference rejection, narrowband interference rejection, and probability of interception.

- *Multiple access:* The multiple access capability of TH-SS signals is acquired in the same manner as that of the FH-SS signals, namely, by making the probability of users' transmissions in the same frequency band at the same time small. In the case of TH, all transmissions are in the same frequency band, so the probability of more than one transmission at the same time is small. This is again achieved by assigning different codes to different users. If multiple transmissions do occur, error-correcting codes ensure that the desired signal can still be recovered. If there is synchronization among the users, and the assigned codes are such that no more than one user transmits at a particular slot, then the TH-CDMA protocol changes to a TDMA protocol where the slot in which a user transmits is not fixed but changes from frame to frame.
- *Multipath interference:* In the TH-CDMA protocol, a signal is transmitted in reduced time. The signaling rate, therefore, increases, and the dispersion of the signal will now lead to an overlap of adjacent bits. Therefore, no advantage is to be gained with respect to multipath interference rejection.
- *Narrowband interference:* A TH-CDMA signal is transmitted in reduced time. This reduction is equal to $1/G_p$ where G_p is the processing gain. At the receiver, we will only receive an interfering signal during the reception of the desired signal. Thus, we only receive the interfering signal $1/G_p$ percent of the time, reducing the interfering power by a factor of G_p.
- *LPI:* With TH-CDMA, the frequency at which a user transmits is constant, but the times at which the user transmits are unknown, and the duration of the transmissions is very short. Particularly when multiple users are transmitting, it is difficult for an intercepting receiver to distinguish the beginning and end of a transmission and to determine which transmissions belong to which user.

Apart from the above-mentioned properties, the TH-CDMA protocols have a number of other specific properties that we can divide into advantageous (+) and disadvantageous (–) ones:

+ Implementation of TH-CDMA is simpler than that of FH-CDMA protocols.
+ It is a very useful method when the transmitter is average-power–limited but not peak-power–limited because the data are transmitted in short bursts at high power.
+ As with the FH-CDMA protocols, the near-far problem is much less of a problem because TH-CDMA is an avoidance system, which means that most of the time a terminal far from the base station can transmit alone and is not hindered by transmissions from the stations close by.
– Code synchronization takes a long time, and the time in which the receiver has to perform the synchronization is short.
– If multiple transmissions occur, a lot of data bits are lost, so a good error-correcting code and data interleaving are necessary.

5.2.4 Chirp Spread Spectrum

Although chirp SS systems have not yet been adapted as a CDMA protocol, for the sake of completeness, a short description of these systems is included in this chapter. A chirp SS system spreads the bandwidth by linear frequency modulation of the carrier. This is shown in Figure 5.11. The processing gain, G_p, is a product of bandwidth B, in which the frequency varies, and duration T of a given signal waveform:

$$G_p = BT \tag{5.2}$$

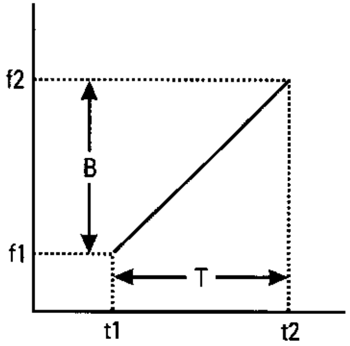


Figure 5.11 Chirp modulation.

5.2.5 Hybrid Systems

The hybrid CDMA systems include all CDMA systems that employ a combination of two or more of the above-mentioned SS modulation techniques. If we limit our discussion to the DS, FH, and TH modulations, we have four possible hybrid systems: DS/FH, DS/TH, FH/TH, and DS/FH/TH.

The idea behind the hybrid system is to combine the specific advantages of each of the modulation techniques. If we take, for example, a combined DS/FH system, we have the advantage of the anti-multipath property of the DS system combined with the favorable near-far operation of the FH system. Of course, the disadvantage lies in the increased complexity of the transmitter and receiver. Figure 5.12 shows a block diagram of a combined DS/FH CDMA transmitter.

The data signal is first spread using a DS code signal. The SS is then modulated on a carrier whose frequency hops according to a different code sequence. A code clock ensures a fixed relationship between the two codes.

5.3 Generation of a Spreading Code

In the previously described CDMA systems, the choice of the type of code sequence is important with respect to the resistance against both multipath interference and multiuser interference. To overcome the interference, several requirements must be satisfied:

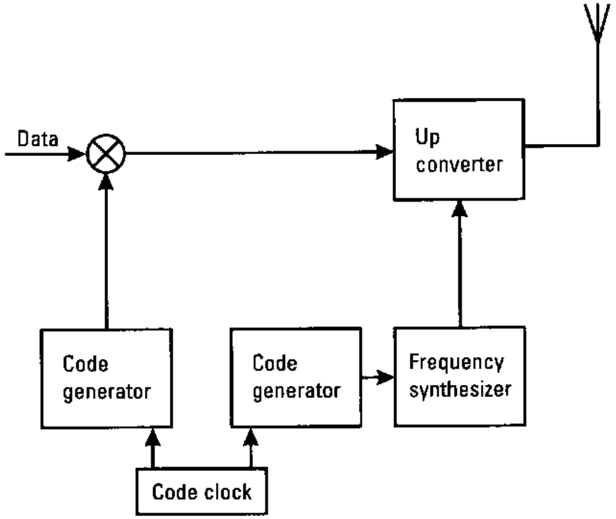


Figure 5.12 Hybrid DS-FH transmitter.

1. Each code sequence generated from a set of code-generation functions must be periodic with a constant length.
2. Each code sequence generated from a set of code-generation functions must be easy to distinguish from its time-shifted code.
3. Each code sequence generated from a set of code-generation functions must be easy to distinguish from other code sequences.

The first and second requirements are important with respect to the multipath propagation effects that occur in mobile outdoor and indoor radio environments. The third requirement is important with respect to the multiple access capability of communications systems. To measure the distinction level of the codes for requirements (1) and (2), an autocorrelation function and a cross-correlation function are used, respectively. The autocorrelation function is used to measure the distinction level, and it is defined as follows:

$$r_{XX}(t)=\frac{1}{T}\int_0^T X(t)X(t+\tau)d\tau \tag{5.3}$$

For the MATLAB function showing the autocorrelation function, `autocorr.m` must be used in Program 5.1. The arguments of this function are the name of the sequence and the number of periods of the code for which the autocorrelation function is to be obtained. For example, to calculate the autocorrelation function of a code of $X(t) = [1, 1, 1, -1, -1, 1, -1]$, the following command must be typed in the MATLAB window:

```
>>  X=[1,1,1,-1,-1,1,-1];
>>  Rxx=autocorr(X);
```

In this case, we use a three-stage M-sequence with a code length of 7, which will be described in this section. The autocorrelation value is 7 at $t = k \cdot T\,(k = 1, 2, 3, \ldots)$ and -1 at other points. To calculate this value, we can calculate the value of correlation function R as `Rxx`.

```
>>  Rxx
```

In contrast, the cross-correlation function is a value of correlation between distinct code sequences $X(t)$ and $Y(t)$, and it is defined as follows.

$$r_{XX}(t)=\frac{1}{T}\int_0^T X(t)Y(t+\tau)d\tau \tag{5.4}$$

For the MATLAB function showing the cross-correlation function, crosscorr.m must be used in Program 5.2. The arguments of this function are the name of the sequence and the number of periods of the code for which the autocorrelation function is to be obtained. For example, to calculate the cross-correlation function of codes $X(t) = [1, 1, 1, -1, -1, 1, -1]$ and $Y(t) = [1, -1, 1, -1, 1, -1, 1]$, the following command must be typed in the MATLAB window.

```
>>  X=[1,1,1,-1,-1,1,-1];
>>  Y=[1,-1,1,-1,1,-1,1];
>>  Rxy=crosscorr(X,Y);
```

In this case, we use a three-stage M-sequence and a random sequence with a code length of 7. The correlation between the two codes is quite small. To calculate the value, we can calculate the value of correlation function Rxy.

```
>>  Rxy
```

By using the autocorrelation and cross-correlation functions, the spread code is evaluated.

This section introduces three code sequences as examples of such spreading codes. These are (1) an M-sequence, (2) a Gold sequence, and (3) an orthogonal Gold sequence. This section shows how to generate these codes by using the MATLAB program and discusses the characteristics of the generated codes.

5.3.1 Code Generation by Linear Feedback Shift Registers

There are several ways to generate code sequences. One is by the use of feedback shift registers, and this method is the one generally used in CDMA systems. A shift register consists of a number of cells (numbered from 1 to r), and each cell is a storage unit that, under the control of a clock pulse, moves its contents to its output while reading its new contents from its input. In the standard configuration of a feedback shift register, the input of cell m will be a function of the output of cell $m - 1$, and the output of cell r (the last cell of the shift register) forms the desired code sequence.

The function combining the outputs of cell $m - 1$ and cell r with the input of cell m can be either a linear or a nonlinear function. This gives us two types of feedback shift registers: linear *feedback shift registers* (linear FSRs) and nonlinear FSRs. The difference between these two types has been explained in other works [1, 11]. This book describes several sequences generated from linear FSRs. Figure 5.13 shows a single linear binary shift register, which can generate a sequence from generation polynomial $h(x) = x^5 + x^2 + 1$. In general, the configuration of a linear binary shift register of n sections is described by a generator polynomial, which is a binary polynomial of degree n. Number n is the number of sections of the shift register.

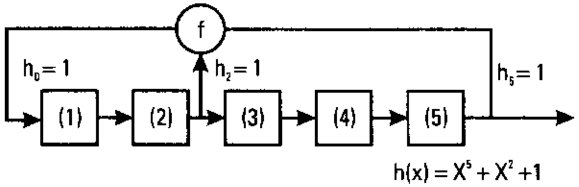


Figure 5.13 A single linear binary shift register.

$$h(x) = h_n x^x + h_{n-1} x^{n-1} + \ldots h_1 x^1 + 1 \quad \left(h_i \in \{0,1\}\right) \tag{5.5}$$

In Figure 5.13, $h(x) = x^5 + x^2 + 1$, $h_0 = h_2 = 1$, and $h_n = 0$. By using these shift registers, most spread code sequences are generated.

5.3.2 M-Sequence

M-sequences are generated by a single LSR. In particular, a sequence with the maximum possible period, ($N_c = 2^n - 1$), is generated by an n-stage binary shift register with linear feedback. To generate an M-sequence, the generator polynomial must be a generation polynomial of degree n. Thus, the periodic autocorrelation function of an M-sequence is given by

$$r_{xx}(t) = \begin{cases} 1 & t = 0 \bmod N_c \\ -1/N_c & \text{otherwise} \end{cases} \tag{5.6}$$

If $n \neq 0 \bmod 4$, there exist pairs of maximum-length sequences with a three-valued cross-correlation function [12], where the two values are $\{-t(n), t(n)-2\}$ with

$$t(n) = \begin{cases} 1 + 2^{(n+1)/2} & n: \text{odd} \\ 1 + 2^{(n+2)/2} & n: \text{even} \end{cases} \tag{5.7}$$

The function to generate M-sequences is programmed in Program 5.3 as mseq.m. The number of registers, the initial values of the registers, and the position of the feedback taps are given as arguments of mseq.m. For example, suppose that the number of registers is 3, the initial values of the registers are [1, 1, 1], and the position of the feedback tap is in the first and third taps. The generation polynomial is expressed as $h(x) = x^3 + x + 1$, and the shift register to configure an M-sequence is shown in Figure 5.14. When an M-sequence is generated by using a generation polynomial, the following command is performed

```
>> m1=mseq(3,[1,3],[1,1,1])
```

As a result, a three-stage M-sequence [1, 1, 1, 0, 1, 0, 0] is generated as a vector. Here, `mseq.m` has the fourth argument, which denotes the number of outputs. If the number of outputs, N, is given, we can obtain N one-chip shifted M-sequences. For example, another three-stage M-sequence is generated by the following command:

```
>> m2=mseq(3, [2,3],[1,1,1],3)
```

When this command is performed, three one-chip shifted M-sequences are obtained.

```
ans =
1  1  1  0  0  1  0
0  1  1  1  0  0  1
1  0  1  1  1  0  0
```

In Program 5.4, `shift.m` used in function `mseq.m` is one of the functions that shift the number of chips given by the users for the vector or matrix.

By using functions `autocorr.m` and `crosscorr.m` programmed in Programs 5.1 and 5.2, the characteristics of the M-sequences can be evaluated. First, because the generated code sequences consist of 0 and 1, the code sequences are converted into code sequences consisting of −1 and 1 by the following command:

```
>> m1=m1*2-1;
>> m2=m2*2-1;
```

The autocorrelation function of three-stage M-sequence `m1` is calculated by the following command:

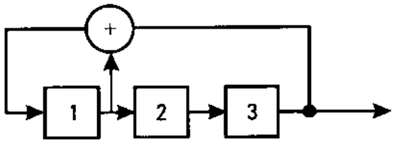


Figure 5.14 A three-stage M-sequence.

```
>> autocorr(m1)
```

As a result, an autocorrelation value of [7, −1, −1, −1, −1, −1, −1] is obtained. The autocorrelated value satisfies (5.6).

Next, the cross-correlation function between `m1` and `m2(1,:)` is obtained by the following command:

```
>> crosscorr(m1,m2(1,:))
```

As a result, a cross-correlation value of [3, −1, 3, −1, −1, −5, 3] is obtained. This result takes three values, namely, $[-1, -t(n), t(n)-2]$, where $t(n) = 5$ from (5.7). Therefore, `m1` and `m2(1,:)` have the characteristics of a preferred pair. The number of M-sequences is very small. Table 5.2 shows the information of the feedback tap that can generate a several-stage M-sequence. Based on the M-sequence, other spread code sequences are configured.

5.3.3 Gold Sequence

The M-sequence has good autocorrelation characteristics. However, the number of mobile communication systems that use the M-sequence as a function is very small. This is because the number of M-sequences that have the same code length and the same correlation characteristics is limited. When a CDMA system where many users communicate to each other is realized, we need sequences with many different codes that have the same correlation value. The Gold sequence is one of such sequences.

Table 5.2
Feedback-Tap Information

Number of Stage	Period	Number of M-Sequence	Feedback Tap (Example)	Number of Preferred Pair
3	7	2	(1,3)(2,3)	2
4	15	2	(1,4)	0
5	31	6	(2,5)(2,3,4,5)(1,2,4,5)	3
6	63	6	(1,6)(1,2,5,6)(2,3,5,6)	2
7	129	18	(4,7)(1,2,3,7)(2,3,4,7) (1,7)(1,3,6,7)(2,4,6,7)	6
8	255	16	(2,3,4,8)(3,5,6,8) (2,5,6,8)(1,3,5,8)	0
9	511	48	(4,9)(3,4,6,9)(4,5,8,9) (1,4,8,9)(2,3,5,9)	2

The Gold sequence was developed by Gold [13]. It is generated by *exclusive OR* (EXOR) of two M-sequences whose relationship is that of a preferred pair described in Section 5.3.2. The generation circuit is shown in Figure 5.15 where a three-stage shift register is used. The number of Gold sequences generated by a generation circuit is $2^n + 1$[1, 11], and it is obtained by changing the initial value of the register and adding two M-sequences when we use an n-stage shift register. In the Gold sequence generated from two preferred-pair M-sequences, the cross-correlation value takes three values, namely, $\{-1, -t(n), t(n)-2\}$, where $t(n)$ is shown in (5.7).

The generation function of a Gold sequence is programmed in Program 5.5 as `goldseq.m`. To generate a Gold sequence, two preferred-pair M-sequences are given in `goldseq.m`. For example, to generate a three-stage Gold sequence from two M-sequences, the following commands must be performed:

```
>> m1=mseq(3,[1,3],[1,1,1]);
>> m2=mseq(3,[2,3],[1,1,1]);
>> g1=goldseq(m1,m2)
```

As a result, we can obtain a three-stage Gold sequence of [0 0 0 0 1 1 0] with a length of 7 as a vector. By changing the initial data, a different Gold sequence can be obtained. To calculate the cross-correlation value, another Gold sequence is generated.

```
>> m3=mseq(3,[1,3],[1,0,0]);
>> m4=mseq(3,[2,3],[1,0,1]);
>> g2=goldseq(m3,m4)
```

Here, `goldseq.m` has the third argument, which denotes the number of outputs. If the number of outputs, N, is given as an argument, we can obtain N one-chip shifted Gold sequences. For example, when the following command is performed:

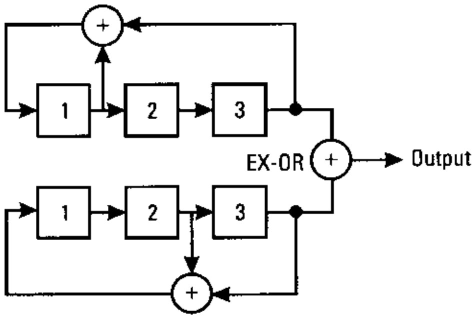


Figure 5.15 A three-stage Gold sequence.

```
>> g1=goldseq(m1,m2,3)
```

three one-chip shifted Gold sequences are obtained.

```
ans =
0   0   0   0   1   1   0
1   0   0   1   1   0   1
0   1   0   1   0   0   0
```

By using functions `autocorr.m` and `crosscorr.m` programmed in Programs 5.1 and 5.2, the characteristics of an M-sequence can be evaluated. First, the autocorrelation function of three-stage Gold sequence `g1(1,:)` is calculated by the following command:

```
>> g1=g1*2-1;
>> autocorr(g1(1,:))
```

As a result, an autocorrelation value of [7, 3, −1, −1, −1, 3] is obtained. The autocorrelation value is a large value at the synchronization point. However, at other points, the data fluctuate.

Next, the cross-correlation function between `g1(1,:)` and `g1(2,:)` is obtained by the following command:

```
>> crosscorr(g1(1,:),g1(2,:))
```

As a result, a cross-correlation value of [−1, 3, −1, −5, −1, 3, −1] is obtained. This result takes values of $\{-1, -t(n), t(n)-2\}$, where $t(n) = 3$ from (5.7).

5.3.4 Orthogonal Gold Sequence

The Gold sequence has many different codes compared to those of the M-sequence. However, there are several problems associated with the Gold sequence:

- The proportion of 0 to 1 is not always balanced.
- The cross-correlation value of the Gold sequence is not 0 in a synchronized environment.
- The code length is an odd number. As a result, a special clock is needed to generate the Gold sequence.

To solve the above problems, one chip is added to the Gold sequence to balance the proportion of 0 to 1. This sequence is called an orthogonal Gold sequence. The cross-correlation value of the orthogonal Gold sequence is 0 at the synchronous point. At other points, the characteristics of the sequence are similar to those of the Gold sequence.

5.4 Configuration of a Transmitter and a Receiver by Using a Computer Program

This section shows the procedure to obtain the BER of a synchronous DS-CDMA. The configuration of the synchronous DS-CDMA is shown in Figure 5.16. In the synchronous DS-CDMA, users employ their own sequences to spread the information data. At each user's terminal, the information data are modulated by the first modulation scheme (e.g., QPSK and BPSK). Then, the first bits of the modulated data are spread by a code sequence, such as an M-sequence or a Gold sequence. The spread data of all the users are transmitted to the base station at the same time. The base station detects the information data of each user by correlating the received signal with a code sequence allocated to each user. In the simulation, QPSK is used as the first modulation scheme. The main program used is shown in Program 5.6, and the simulation program is shown in Programs 5.7–5.11. In simulation, because the first modulation is QPSK, the program is based on the program of QPSK discussed in Chapter 3. Therefore, the description of the common features between the programs for QPSK and a synchronous DS-CDMA is omitted.

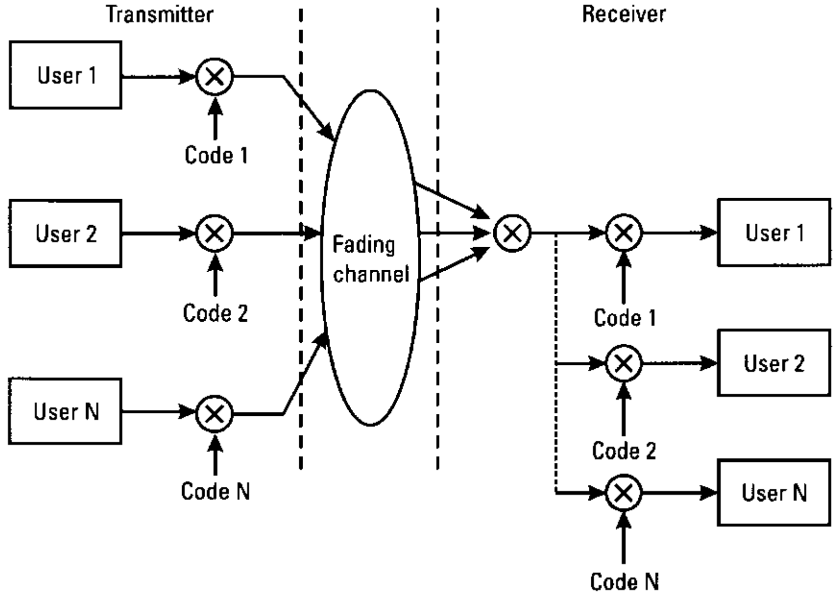


Figure 5.16 Synchronous DS-CDMA system.

In the main program used in the simulation shown in Program 5.6, the parameters used for the simulation are defined as follows.

```
sr        = 256000.0;      % Symbol rate
ml        = 2;             % Number of modulation levels
br        = sr * ml;       % Bit rate
nd        = 100;           % Number of symbols
ebn0      = 10;            % Eb/No
irfn      = 21;            % Number of filter taps
IPOINT    =  8;            % Number of oversamples
alfs      =  0.5;          % Roll-off factor
```

The coefficient of the filter is defined as shown in the program that evaluates the performance of QPSK.

```
[xh] = hrollfcoef(irfn,IPOINT,sr,alfs,1);
      % T FILTER FUNCTION
[xh2]= hrollfcoef(irfn,IPOINT,sr,alfs,0);
      % R FILTER FUNCTION
```

The difference between the programs for QPSK and those for DS-CDMA is in the parameters used to define the number of users and in the generation of a spread code. In this simulation, three types of spread sequences, namely, an M-sequence, a Gold sequence, and an orthogonal Gold sequence, are used. By denoting variables as `seq` 1, 2, or 3, a code sequence is selected. Next, the number of registers is set to generate an M-sequence. In synchronous DS-CDMA, the number of code sequences that can be allocated to different users is equal to the number of code lengths. Therefore, the length of the code sequence must be larger than the number of users. To generate a code sequence, we must specify the number of registers, the position of the feedback tap, and the initial values of the registers. To generate a Gold sequence and an orthogonal Gold sequence, two M-sequences are needed. Therefore, the following parameters are used.

```
user      = 1;                 % Number of users
seq       = 1;                 % 1:M-sequence  2:Gold
                               % 3:Orthogonal Gold
stage     = 3;                 % Number of stages
ptap1     = [1 3];             % Position of taps for 1st
ptap2     = [2 3];             % Position of taps for 2nd
regi1     = [1 1 1];           % Initial value of
                               % register for 1st
regi2     = [1 1 1];           % Initial value of
                               % register for 2nd
```

By using these parameters, a spread code is generated, and the generated code is stored as variable code. Here, code is a matrix with a sequence of the number of users multiplied by the length of the code sequence. An M-sequence is generated by function mseq.m, and a Gold sequence is generated by function goldseq.m. An orthogonal Gold sequence can be generated by adding a 0 bit of data to the top or bottom of a Gold sequence.

```
switch seq
case 1                    % M-sequence
  code = mseq(stage,ptap1,regi1,user);
case 2                    % Gold sequence
  m1     = mseq(stage,ptap1,regi1);
  m2     = mseq(stage,ptap2,regi2);
  code   = goldseq(m1,m2,user);
case 3                    % Orthogonal Gold sequence
  m1     = mseq(stage,ptap1,regi1);
  m2     = mseq(stage,ptap2,regi2);
  code   = [goldseq(m1,m2,user),zeros(user,1)];
end
```

Because the generated code sequence consists of 0 and 1, the program converts it into a sequence consisting of –1 and 1.

```
code = code * 2 - 1;
clen = length(code);
```

Next, the parameters for the fading simulator are defined. When rfade is 0, the simulation evaluates the BER performance in an AGWN channel. When rfade is 1, the simulation evaluates the BER performance in a Rayleigh fading environment.

```
rfade    = 0;                  % Rayleigh fading
                               % 0:nothing 1:consider
itau     = [0,8];              % Delay time
dlvl1    = [0.0,40.0];         % Attenuation level
n0       = [6,7];              % Number of waves to
                               % generate fading
th1      = [0.0,0.0];          % Initial phase of
                               % delayed wave
itnd1    = [3001,4004];        % Set fading counter
now1     = 2;                  % Number of direct waves +
                               % delayed waves
tstp     = 1/sr/IPOINT/clen;   % Frequency resolution
fd       = 160;                % Doppler frequency [Hz]
flat     = 1;                  % Flat Rayleigh environment
itndel   = itndel=nd*IPOINT*clen*30;
```

Then, the number of simulation loops is set. The variables that count the number of transmitted data bits and the number of errors are initialized.

```
nloop     = 100;          % Simulation number of times
noe       = 0;            % Number of errors
nod       = 0;            % Number of data
```

The transmitted data in the in-phase channel and quadrature phase modulated by QPSK are multiplied by the code sequence used to spread the transmitted data. The spread data are then oversampled and filtered by a roll-off filter and transmitted to a communication channel. Here, functions compoversamp2.m and compconv2.m are modified functions of compoversamp.m (Program 3.8) and compconv.m (Program 3.7). The format used to input these new functions does not depend on the vector or matrix. These programs are shown in Programs 5.9 and 5.10.

```
data  = rand(user,nd*ml)  0.5;
[ich, qch]   = qpskmod(data,user,nd,ml);
  % QPSK modulation
[ich1,qch1]= spread(ich,qch,code);
  % Spreading
[ich2,qch2]= compoversamp2(ich1,qch1,IPOINT);
  % Oversampling
[ich3,qch3]= compconv2(ich2,qch2,xh);
  % T filter
```

As shown in Figure 5.16, the signals transmitted from the users are synthesized.

```
if user == 1                % Number of users is 1
  ich4 = ich3;
  qch4 = qch3;
else                        % Number of users is plural
  ich4 = sum(ich3);
  qch4 = sum(qch3);
end
```

Then, the synthesized signal is contaminated in a Rayleigh fading channel. In reality, the transmitted signals of all users are contaminated by distinctive Rayleigh fading. However, in this simulation, the synthesized signal is contaminated by Rayleigh fading.

```
if rfade == 0                 % in AWGN
  ich5 = ich4;
  qch5 = qch4;
else                          % Rayleigh fading channel
   [ich5,qch5] = sefade
     (ich4,qch4,itau,dlvl1,th1,n0,itnd1, ...
     now1,length(ich4),tstp,fd,flat);
  itnd1 = itnd1 + itndel;     % fading counter
end
```

At the receiver, AWGN is added to the received data, as shown in the simulation for the QPSK transmission. Then, the contaminated signal is filtered by using a the root roll-off filter. Here, `comb2.m` is a modified function of `comb.m` (Program 2.4) to accommodate the vector of `attn`. The modified program is shown in Program 5.11.

```
spow  = sum(rot90(ich3.^2+qch3.^2))/nd;
% attenuation Calculation
attn  = sqrt(0.5 * spow * sr / br * 10^(-ebn0/10));
[ich6,qch6]  = comb2(ich5,qch5,attn);
% Add White Gaussian Noise
[ich7,qch7]  = compconv2(ich6,qch6,xh2);% R filter
sampl = irfn * IPOINT + 1;
ich8  = ich7(:,sampl:IPOINT:IPOINT*nd*clen+sampl-1);
  % Resampling
qch8  = qch7(:,sampl:IPOINT:IPOINT*nd*clen+sampl-1);
```

The resampled data are now the synthesized data of all the users. By correlating the synthesized data with the spread code used at the transmitter, the transmitted data of all the users are detected. The correlation is performed by Program 5.8.

```
[ich9 qch9] = despread(ich8,qch8,code); % Despreading
```

The correlated data are demodulated by QPSK. Then, the total number of errors for all the users is calculated. Finally, the BER is calculated.

```
demodata = qpskdemod(ich9,qch9,user,nd,ml);
  % QPSK demodulation
noe2      = sum(sum(abs(data-demodata)));
nod2      = user * nd * ml;
noe       = noe + noe2;
nod       = nod + nod2;
```

5.5 BER Performance

5.5.1 BER Performance in an AWGN Environment

By running the main program, Program 5.6,

```
>> dscdma
```

the BER performance in an AWGN environment of a synchronous DS-CDMA discussed above is simulated and the results are shown in Figures 5.17 and 5.18. In the M-sequence, the cross-correlation value is not 0 at the synchronized point. Therefore, this nonzero correlation becomes interference for

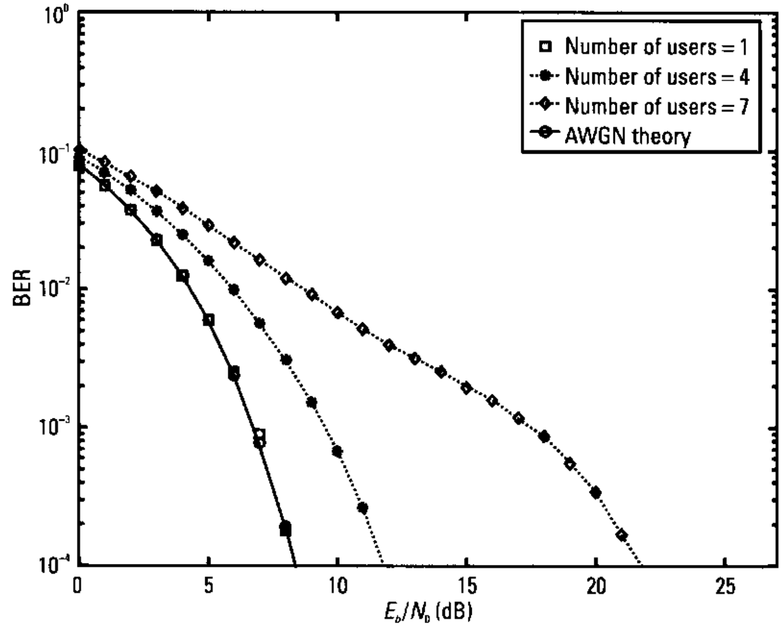


Figure 5.17 BER performance of DS-CDMA with M-sequence in AWGN.

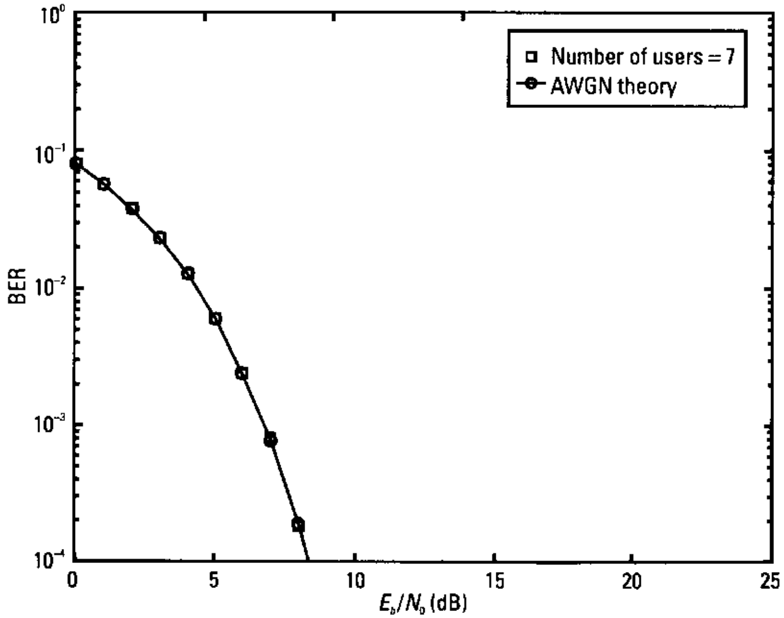


Figure 5.18 BER performance of DS-CDMA with orthogonal Gold sequence in AWGN.

other users. As a result, as the number of users increases, the BER degrades. In contrast, in the orthogonal Gold sequence, the value of cross-correlation between the users is 0 at the synchronized point. This means that as the number of users increases, the value of BER approaches the theoretical value.

5.5.2 BER Performance in a Rayleigh Fading Environment

The BER performance of a synchronous DS-CDMA in a Rayleigh fading environment is shown in Figures 5.19 and 5.20. In the M-sequence, as the number of users increases, the BER degrades. This is because the value of cross-correlation between the codes is not 0, and this is interference. In contrast, in the orthogonal Gold sequence, the orthogonality of the codes keeps the cross-correlation at the synchronized point. As a result, as the number of users increases, the BER approaches the theoretical value.

5.6 Conclusion

This chapter discusses some of the methods used to evaluate the transmission performance of CDMA systems. The basic CDMA was described. Three well-known code sequences were analyzed. The BER performance of a synchronized DS-CDMA was evaluated by computer simulation. Using this chapter, readers can design their own code sequences or new DS-CDMA systems. CDMA is one of the key systems for third- and fourth-generation wireless communication systems. The authors are looking forward to evaluating these new systems based on the program outlined in this chapter.

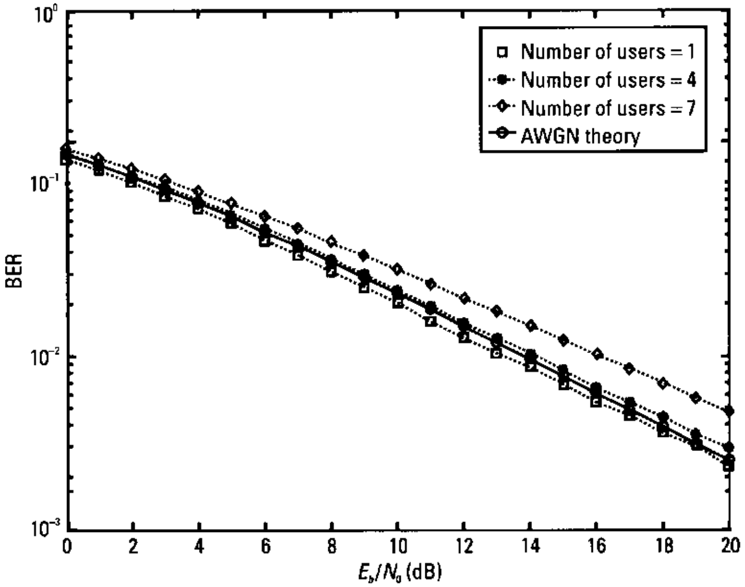


Figure 5.19 BER performance of DS-CDMA with M-sequence in a Rayleigh fading environment.

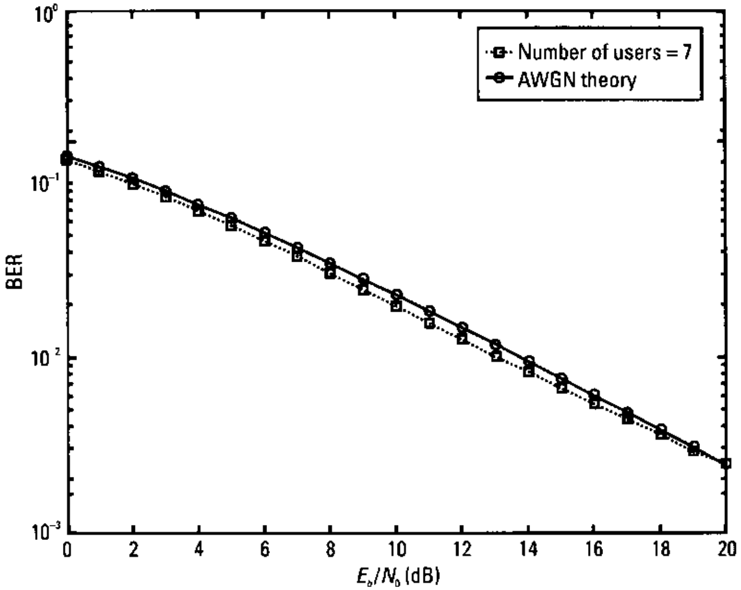


Figure 5.20 BER performance of DS-CDMA with orthogonal Gold sequence in a Rayleigh fading environment.

References

[1] Prasad, R., *CDMA for Wireless Personal Communications*, Norwood, MA: Artech House, 1996.

[2] Special issue on spread-spectrum communication, *IEEE Trans. Commun.*, Vol. COM-30, May 1982.

[3] Simon, M. K., et al., "Spread-Spectrum Communications," Vol. I, II, III, *Comp. Sci.*, 1985.

[4] Scholtz, R. A., "The Spread-Spectrum Concept," *IEEE Trans. Commun.*, Vol. COM-25, August 1977, pp. 748–755.

[5] Torrieri, D. J., *Principles of Secure Communication Systems*, Norwood, MA: Artech House, 1992.

[6] Cooper, G. R., and C. D. McGillem, *Modern Communication and Spread-Spectrum*, New York: McGraw-Hill, 1986.

[7] Glisic, S. G., and P. A. Leppanen (eds.), *Code Division Multiple Access Communications*, Norwell, MA: Kluwer Academic Publishers, 1995.

[8] Viterbi, A. J., *CDMA Principles of Spread-Spectrum Communications*, Reading, MA: Addison-Wesley, 1995.

[9] Dixon, R. C., *Spread-Spectrum Systems*, New York: John Wiley & Sons, 1984.

[10] Viterbi, A. M., and A. J. Viterbi, "Erlang Capacity of a Power-Controlled CDMA System," *IEEE J. Selected Areas in Comm.*, Vol. 11, August 1993, pp. 892–899.

[11] Prasad, R., *Universal Wireless Personal Communications*, Norwood, MA: Artech House, 1996.

[12] Golomb, S. W., *Shift Register Sequences*, San Francisco, CA: Holden-Day, 1967.

[13] Gold, R., "Optimal Binary Sequences for Spread Spectrum Multiplexing," *IEEE Trans. Inf. Theory*, Vol. IT-13, October 1967, pp. 619–621.

Appendix 5A

Program 5.1

```
% Program 5-1
% autocorr.m
%
% Autocorrelation function of a sequence
%
% Programmed by M. Okita and H. Harada
%

function [out] = autocorr(indata, tn)

% *****************************************************

% indata : input sequence
% tn     : number of period
% out    : autocorrelation data

% *****************************************************

if nargin < 2
      tn = 1;
end

ln  = length(indata);
out = zeros(1,ln*tn);

for ii=0:ln*tn-1
      out(ii+1) = sum(indata.*shift(indata,ii,0));
end

%******************** end of file ********************
```

Program 5.2

```
% Program 5-2
% crosscorr.m
%
% Crosscorrelation function of a sequence
%
% Programmed by M. Okita and H. Harada
%

function [out] = crosscorr(indata1, indata2, tn)

% *****************************************************

% indata1    : input sequence1
% indata2    : input sequence2
% tn          : number of period
% out         : crosscorrelation data

% ******************************************************

if nargin < 3
      tn = 1;
end

ln  = length(indata1);
out = zeros(1,ln*tn);

for ii=0:ln*tn-1
      out(ii+1) = sum(indata1.*shift(indata2,ii,0));
end

%******************** end of file *******************
```

Program 5.3

```
% Program 5-3
% mseq.m
%
% The generation function of M-sequence
%
% An example
%    stg     = 3
%    taps    = [ 1 , 3 ]
%    inidata = [ 1 , 1 , 1 ]
%    n       = 2
%
% Programmed by M. Okita and H. Harada
%

function [mout] = mseq(stg, taps, inidata, n)

% ******************************************************

% stg         : Number of stages
% taps        : Position of register feedback
% inidata     : Initial sequence
% n           : Number of output sequence(It can be omitted)
% mout        : output M sequence

% ******************************************************

if nargin < 4
      n = 1;
end

mout = zeros(n,2^stg-1);
fpos = zeros(stg,1);
```

```
fpos(taps) = 1;

for ii=1:2^stg-1

      mout(1,ii) = inidata(stg);     % storage of the
                                     % output data
      num = mod(inidata*fpos,2);     % calculation of the
                                     % feedback data

      inidata(2:stg) = inidata(1:stg-1);    % one shifts
                                            % the register
      inidata(1)     = num;                 % return
                                            % feedback data

end

if n > 1
      for ii=2:n
         mout(ii,:) = shift(mout(ii-1,:),1,0);
      end
end

%******************** end of file *******************
```

Program 5.4

```
% Program 5-4
% shift.m
%
% Shift the contents of the register.
%
% Programmed by M. Okita and H. Harada
%

function [outregi] = shift(inregi,shiftr,shiftu)

% ******************************************************

% inrege        : Vector or matrix
% shiftr        : The amount of shift to the right.
% shiftu        : The amount of shift to the top.
% outregi       : Register output

% ******************************************************

[h, v]  = size(inregi);
outregi = inregi;

shiftr = rem(shiftr,v);
shiftu = rem(shiftu,h);
```

```
if shiftr > 0
      outregi(:,1  :shiftr) = inregi(:,v-shiftr+1:v   );
      outregi(:,1+shiftr:v   ) = inregi(:,1  :v-shiftr);
elseif shiftr < 0
      outregi(:,1  :v+shiftr) = inregi(:,1-shiftr:v   );
      outregi(:,v+shiftr+1:v  ) = inregi(:,1  :-shiftr);
end

inregi = outregi;

if shiftu > 0
      outregi(1   :h-shiftu,:) = inregi(1+shiftu:h,  :);
      outregi(h-shiftu+1:h,  :) = inregi(1   :shiftu,:);
elseif shiftu < 0
      outregi(1  :-shiftu,:) = inregi(h+shiftu+1:h,  :);
      outregi(1-shiftu:h,  :) = inregi(1   :h+shiftu,:);
end

%******************** end of file ********************
```

Program 5.5

```
% Program 5-5
% goldseq.m
%
% The generation function of Gold sequence
%
% Programmed by M. Okita and H. Harada
%

function [gout] = goldseq(m1, m2, n)

% ****************************************************

% m1 : M-sequence 1
% m2 : M-sequence 2
% n  : Number of output sequence(It can be omitted)
% gout : output Gold sequence

% ****************************************************

if nargin < 3
      n = 1;
end

gout = zeros(n,length(m1));

for ii=1:n
      gout(ii,:)    = xor(m1,m2);
      m2            = shift(m2,1,0);
end
```

```
%******************** end of file ********************
```

Program 5.6

```
% Program 5-6
%
% Simulation program to realize DS-CDMA system
%
% dscdma.m
%
% Programmed by M. Okita and H. Harada
%

%****************** Preparation part *****************

sr   = 256000.0;          % symbol rate
ml   = 2;                 % number of modulation levels
br   = sr * ml;           % bit rate
nd   = 100;               % number of symbol
ebn0 = 3;                 % Eb/N0

%*************** Filter initialization ***************

irfn   = 21;              % number of filter taps
IPOINT =  8;              % number of oversample
alfs   =  0.5;            % roll off factor
[xh]   = hrollfcoef(irfn,IPOINT,sr,alfs,1);   % T FILTER
                                              % FUNCTION
[xh2]  = hrollfcoef(irfn,IPOINT,sr,alfs,0);   % R FILTER
                                              % FUNCTION

%*********** Spreading code initialization ***********

user  = 1;                % number of users
seq   = 1;                % 1:M-sequence  2:Gold
                          % 3:Orthogonal Gold
stage = 3;                % number of stages
ptap1 = [1 3];            % position of taps for 1st
ptap2 = [2 3];            % position of taps for 2nd
regi1 = [1 1 1];          % initial value of register
                          % for 1st
regi2 = [1 1 1];          % initial value of register
                          % for 2nd

%********** Generation of the spreading code **********

switch seq
case 1                       % M-sequence
      code = mseq(stage,ptap1,regi1,user);
case 2                       % Gold sequence
      m1   = mseq(stage,ptap1,regi1);
      m2   = mseq(stage,ptap2,regi2);
      code = goldseq(m1,m2,user);
case 3                          % Orthogonal Gold sequence
      m1   = mseq(stage,ptap1,regi1);
      m2   = mseq(stage,ptap2,regi2);
      code = [goldseq(m1,m2,user),zeros(user,1)];
end
code = code * 2 - 1;
clen = length(code);

%*************** Fading initialization ****************

rfade  = 0;      % Rayleigh fading 0:nothing 1:consider
itau   = [0,8]; % delay time
dlvl1  = [0.0,40.0];   % attenuation level
n0     = [6,7]; % number of waves to generate fading
th1    = [0.0,0.0];    % initial phase of delayed wave
itnd1  = [3001,4004];  % set fading counter
now1   = 2;      % number of direct wave + delayed wave
tstp   = 1 / sr / IPOINT / clen;     % time resolution
fd     = 160;    % doppler frequency [Hz]
flat   = 1;      % flat Rayleigh environment
itndel = nd * IPOINT * clen * 30;    % number of fading
                                     % counter to skip

%****************** START CALCULATION *****************

nloop = 1000;          % simulation number of times
noe   = 0;
nod   = 0;

for ii=1:nloop

%******************** Transmitter ********************

    data = rand(user,nd*ml) > 0.5;

      [ich, qch]   = qpskmod(data,user,nd,ml);  % QPSK
                                                % modulation
      [ich1,qch1] = spread(ich,qch,code);  % spreading
      [ich2,qch2] = compoversamp2(ich1,qch1,IPOINT);
         % over sampling
      [ich3,qch3] = compconv2(ich2,qch2,xh);
         % filter

      if user == 1        % transmission
         ich4 = ich3;
         qch4 = qch3;
      else
         ich4 = sum(ich3);
         qch4 = sum(qch3);
      end

%******************* Fading channel *******************
```

```
     if rfade == 0
        ich5 = ich4;
        qch5 = qch4;
     else
        [ich5,qch5] = ... % fading channel
        sefade(ich4,qch4,itau,dlvl1,th1,n0,itnd1, ...
           now1,length(ich4),tstp,fd,flat);
        itnd1 = itnd1 + itndel;
     end

%********************** Receiver **********************

     spow = sum(rot90(ich3.^2 + qch3.^2)) / nd;
        % attenuation Calculation
     attn = sqrt(0.5 * spow * sr / br * 10^(-ebn0/10));

     [ich6,qch6] = comb2(ich5,qch5,attn);
        % Add White Gaussian Noise (AWGN)
     [ich7,qch7] = compconv2(ich6,qch6,xh2);
        % filter

     sampl = irfn * IPOINT + 1;
     ich8  = ich7(:,sampl:IPOINT:IPOINT*nd*clen+sampl-1);
     qch8  = qch7(:,sampl:IPOINT:IPOINT*nd*clen+sampl-1);
     [ich9 qch9] = despread(ich8,qch8,code);
     % despreading

     demodata = qpskdemod(ich9,qch9,user,nd,ml);
     % QPSK demodulation

%*************** Bit Error Rate (BER) *****************

     noe2 = sum(sum(abs(data-demodata)));
     nod2 = user * nd * ml;
     noe  = noe + noe2;
     nod  = nod + nod2;

     fprintf('%d\t%e\n',ii,noe2/nod2);

end

%********************* Data file **********************

ber = noe / nod;
fprintf('%d\t%d\t%d\t%e\n',ebn0,noe,nod,noe/nod);
  % fprintf: built in function
fid = fopen('BER.dat','a');
fprintf(fid,'%d\t%e\t%f\t%f\t\n',ebn0,noe/nod,noe,nod);
  % fprintf: built in function
fclose(fid);

%******************** end of file ********************
```

Program 5.7

```
% Program 5-7
% spread.m
%
% Data spread function
%
% Programmed by M. Okita and H. Harada
%

function [iout, qout] = spread(idata, qdata, code1)

% ********************************************************

%   idata    : ich data sequence
%   qdata    : qch data sequence
%   iout     : ich output data sequence
%   qout     : qch output data sequence
%   code1    : spread code sequence

% ********************************************************

switch nargin
case { 0 , 1 }
      error('lack of input argument');
case 2
      code1 = qdata;
      qdata = idata;
end

[hn,vn] = size(idata);
[hc,vc] = size(code1);

if hn > hc
      error('lack of spread code sequences');
end

iout = zeros(hn,vn*vc);
qout = zeros(hn,vn*vc);

for ii=1:hn
      iout(ii,:) = reshape(rot90(code1(ii,:),3)...
          *idata(ii,:),1,vn*vc);
      qout(ii,:) = reshape(rot90(code1(ii,:),3)...
          *qdata(ii,:),1,vn*vc);
end

%******************** end of file ********************
```

Program 5.8

```
% Program 5-8
% despread.m
```

```
%
% Data despread function
%
% Programmed by M. Okita and H. Harada
%

function [iout, qout] = despread(idata, qdata, code1)

% ********************************************************

%   idata    : ich data sequence
%   qdata    : qch data sequence
%   iout     : ich output data sequence
%   qout     : qch output data sequence
%   code1    : spread code sequence

% ********************************************************

switch nargin
case { 0 , 1 }
      error('lack of input argument');
case 2
      code1 = qdata;
      qdata = idata;
end

[hn,vn] = size(idata);
[hc,vc] = size(code1);

vn      = fix(vn/vc);

iout    = zeros(hc,vn);
qout    = zeros(hc,vn);

for ii=1:hc
      iout(ii,:) = rot90(flipud(rot90(reshape(idata...
          (ii,:),vc,vn)))*rot90(code1(ii,:),3));
      qout(ii,:) = rot90(flipud(rot90(reshape(qdata...
          (ii,:),vc,vn)))*rot90(code1(ii,:),3));
end

%******************** end of file ********************
```

Program 5.9

```
% Program 5-9
% compoversamp2.m
%
% Function to sample "sample" time
%
% programmed by H. Harada and M. Okita
%
```

```
function [iout,qout] = compoversamp2(iin, qin, sample)

% ****************************************************

% iin        : input Ich sequence
% qin        : input Qch sequence
% iout       : ich output data sequence
% qout       : qch output data sequence
% sample     : Number of oversamples

% ****************************************************

[h,v] = size(iin);

iout = zeros(h,v*sample);
qout = zeros(h,v*sample);

iout(:,1:sample:1+sample*(v-1)) = iin;
qout(:,1:sample:1+sample*(v-1)) = qin;

%******************** end of file ********************
```

Program 5.10

```
% Program 5-10
% compconv2.m
%
% Function to perform convolution between signal and
% filter
%
% Programmed by H. Harada and M. Okita
%

function [iout, qout] = compconv2(idata, qdata, filter)

% ****************************************************

%   idata    : ich data sequence
%   qdata    : qch data sequence
%   iout     : ich output data sequence
%   qout     : qch output data sequence
%   filter   : filter tap coefficients

% ****************************************************

iout = conv2(idata,filter);
qout = conv2(qdata,filter);

%********************* end of file *******************
```

Program 5.11

```
% Program 5-11
% comb2.m
%
% Function to add white gaussian noise
%
% Programmed by H. Harada and M. Okita
%

function [iout, qout] = comb2(idata, qdata, attn)

%********************* variables ********************

% idata  : input Ich data
% qdata  : input Qch data
% iout   : output Ich data
% qout   : output Qch data
% attn   : attenuation level caused by Eb/N0 or C/N

%****************************************************

v = length(idata);
h = length(attn);

iout = zeros(h,v);
qout = zeros(h,v);

for ii=1:h
     iout(ii,:) = idata + randn(1,v) * attn(ii);
     qout(ii,:) = qdata + randn(1,v) * attn(ii);
end

%******************** end of file ********************
```

6

Multiple Access Protocols

6.1 Introduction

Agreement among users on the means of communication is known as a protocol. When users employ a common medium for communications, it is called multiple access. Thus, the multiple access protocol is defined as the agreement and set of rules among users for the successful transmission of information using a common medium. Whenever a resource is used and accessed by more than one independent user, the need for a multiple access protocol arises. In the absence of such a protocol, conflicts occur if more than one user tries to access the resource at the same time. Therefore, the multiple access protocol should avoid or at least resolve these conflicts. Thus, the multiple access technique is defined as the function-sharing (limited) common transmission resource among (distributed) terminals in a network.

The multiple access protocols addressed in this chapter are those used in communication systems in which the resource to be shared is the communication channel [1–27]. In this case, the reason for sharing the resource is mainly the connectivity environment. In wireless communication systems, an added reason is efficient resource utilization [1]. When using a common medium with an access point for every user, there is a network that consists of point-to-point links (different medium), but one of these links is simultaneously shared by many users; thus, a multiplexing technique is needed. Multiplexing is the transmission of different information over the same physical link. Table 6.1 shows the differences between multiple access and multiplexing.

Table 6.1
Difference Between Multiple Access and Multiplexing

	Multiple Access	**Multiplexing**
Resource	Network	Link
Terminal connectivity	Matrix	Point-to-(multi)point
Switching	Yes (≤3)	No (OSI≤2)
Possible	(OSI: open systems interconnection)	
Topologies	Bus Star Ring Tree	Path(s)
Control	Central Distributed	Terminal

The design of a protocol is usually accomplished with a specific goal (environment) in mind, and the properties of the protocol are mainly determined by the design goal. However, if we rule out the environment-specific properties, we can still address a number of properties that any good multiple access protocol should possess:

- The first and foremost task of the multiple access protocols addressed here is to share a common transmission channel among several users in the system. To do this, the protocol must control the way in which the users access (transmit onto) the channel by requiring that the users conform to certain rules. The protocol controls the allocation of channel capacity to the users.
- The protocol should perform the allocation such that the transmission medium is used efficiently. Efficiency is usually measured in terms of channel throughput and the delay of transmissions.
- The allocation should be fair toward individual users; that is, by not taking into account any priorities that might be assigned to the users, each user should (on the average) receive the same allocated capacity.
- The protocol should be flexible in allowing different types of traffic (e.g., voice and data).
- The protocol should be stable. This means that if the system is in equilibrium, an increase in load should move the system to a new equilibrium point. With an unstable protocol an increase in load will force the system to continue to drift to even a higher load and a lower throughput.

- The protocol should be robust with respect to equipment failure and changing conditions. If one user does not operate correctly, this should affect the performance of the rest of the system as little as possible.

This chapter focuses on the wireless mobile environment. In such an environment, we can be more specific about some of the protocol properties, especially regarding the robustness with respect to changing conditions. In a wireless mobile environment, the protocol should be able to deal with the following:

- The hidden terminal problem [two terminals are out of range (hidden from) each other by a hill, a building, or some other physical obstacle opaque to *ultra high frequency* (UHF) signals but both within the range of the central or base station];
- The near-far effect (transmissions from distant users are more attenuated than transmissions from users close by);
- The effects of multipath fading and shadowing experienced in radio channels;
- The effects of cochannel interference in cellular wireless systems caused by use of the same frequency band in an adjacent cell.

Many of the protocol properties mentioned above are in conflict with one another, and a tradeoff has to be made during the protocol design. The tradeoff depends on the environment and the specific use for the protocol.

This chapter introduces some typical protocols for wireless communication and methods to evaluate their performance by computer simulation.

This chapter is organized as follows. Section 6.2 presents the classification of multiple access protocols. Some random access protocols are introduced in Section 6.3. The universal conditions in the computer simulation are shown in Section 6.4. In Sections 6.5–6.8, the configurations and computer simulation methods of pure ALOHA, slotted ALOHA, nonpersistent CSMA, and slotted and nonpersistent ISMA are introduced. Throughput and transmission delay characteristics are also discussed. All programs related to the chapter are presented in Appendix 6C and in the CD-ROM that accompanies the book.

6.2 Classification of Multiple Access Protocols

Starting in 1970 with the ALOHA protocol (discussed later in this section), a number of multiple access protocols have been developed. Numerous ways have been suggested to divide these protocols into groups [14, 16]. This book classifies multiple access protocols into three main groups (Figure 6.1): the contentionless protocols, the contention protocols, and the class of CDMA protocols [1].

The contentionless (or scheduling) protocols avoid the situation in which two or more users access the channel at the same time by scheduling the transmissions of the users. This is either done in a fixed fashion where each user is allocated part of the transmission capacity, or in a demand-assigned fashion where the scheduling only takes place between the users that have something to transmit.

With the contention (or random access) protocols, a user cannot be sure that a transmission will not collide because other users may be transmitting (accessing the channel) at the same time. Therefore, these protocols need to resolve conflicts if they occur.

The CDMA protocols do not belong to either the contentionless or the contention protocols. As explained in Chapter 5, CDMA falls between the two groups. In principle, it is a contentionless protocol where a number of users are allowed to transmit simultaneously without conflict. However, if the number of simultaneously transmitting users rises above a threshold, contention occurs.

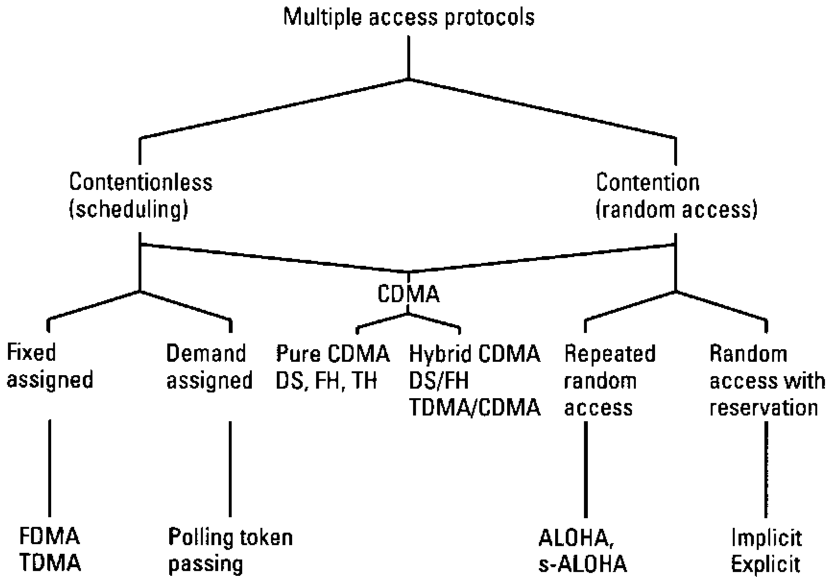


Figure 6.1 Classification of the multiple access protocols.

The contention protocols are further subdivided into repeated random protocols and random protocols with reservation. In the latter, the initial transmission of a user uses a random-access method to get access to the channel. However, once the user accesses the channel, further transmissions of that user are scheduled until the user has nothing more to transmit. The two major types of protocols are known as implicit and explicit reservations. Explicit reservation protocols use a short reservation packet to request transmission at scheduled times. Implicit reservation protocols are designed without the use of any reservation packet.

CDMA protocols are subdivided into pure CDMA and hybrid CDMA. With CDMA forming a separate group and taking into account the subgroups of the contentionless and contention protocols, we end up with six categories.

6.2.1 Contentionless (Scheduling) Multiple Access Protocols

The contentionless multiple access protocols avoid the situation in which multiple users try to access the same channel at the same time by scheduling the transmissions of all the users. The users transmit in an orderly scheduled manner so that every transmission will be a successful one. The scheduling can take two forms:

1. *Fixed-assignment scheduling:* With these types of protocols, the available channel capacity is divided among the users such that each user is allocated a fixed part of the capacity, independent of its activity. The division is done in time or frequency. The time division results in the TDMA protocol, where the transmission time is divided into frames, and each user is assigned a fixed part of each frame, not overlapping with parts assigned to other users. TDMA is illustrated in Figure 6.2(a). The frequency division results in the *frequency division multiple access* (FDMA) protocol, where the channel bandwidth is divided into nonoverlapping frequency bands, and each user is assigned a fixed band. Figure 6.2(b) illustrates FDMA.
2. *Demand-assignment scheduling:* A user is only allowed to transmit if it is active (if it has something to transmit). Thus, the *active* (or ready) users transmit in an orderly scheduled manner. Within the demand-assignment scheduling, we distinguish between centralized control and distributed control. With centralized control, a single entity schedules the transmissions. An example of such a protocol is the roll-call polling protocol. With distributed control, all users are involved in the scheduling process and such a protocol is the token-passing protocol.

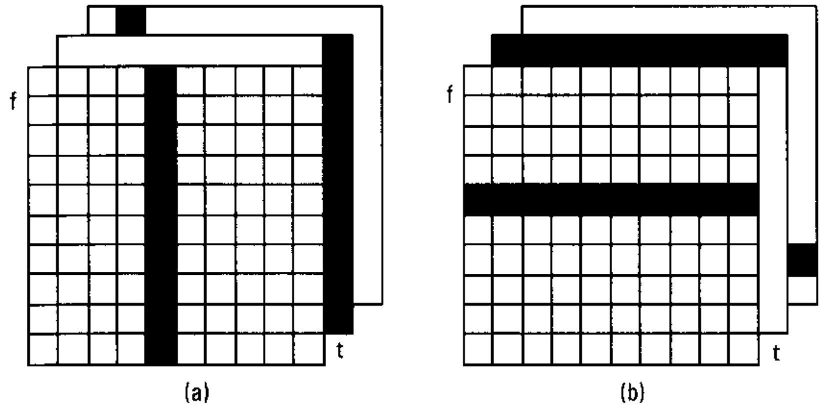


Figure 6.2 (a) TDMA and (b) FDMA.

6.2.2 Contention (Random) Multiple Access Protocols

With contention multiple access protocols there is no scheduling of transmissions. This means that a user getting ready to transmit does not have exact knowledge of when it can transmit without interfering with the transmissions of other users. The user may or may not know of any ongoing transmissions (by sensing the channel), but it has no exact knowledge about other ready users. Thus, if several ready users start their transmissions more or less at the same time, all of the transmissions will fail. This possible transmission failure makes the occurrence of a successful transmission a more or less random process. The random access protocol should resolve the contention that occurs when several users transmit simultaneously.

We subdivide contention multiple access protocols into two groups, the repeated random access protocols [e.g., pure (p)-ALOHA, slotted (s)-ALOHA, CSMA, and ISMA protocols], and random access protocols with reservation [e.g., reservation ALOHA (r-ALOHA), and the *packet reservation multiple access protocols* (PRMA)]. With the former protocols, every transmission a user makes is as described above. With every transmission there is the possibility of contention occurring. With the latter protocols, only in its first transmission does a user not know how to avoid collisions with other users. However, once a user has successfully completed its first transmission (once the user has access to the channel), future transmissions of that user will be scheduled in an orderly fashion so that no contentions can occur. Thus, after a successful transmission, part of the channel capacity is allocated to the user, and other users will refrain from using that capacity. The user loses its allocated capacity if, for some time, it had nothing to transmit.

6.3 Random Access Protocols

We consider mobile terminals communicating with a centrally located base station using packet radio. The best-known method for random access packet communication, ALOHA [2], suffers heavily due to collisions among packets. The CSMA protocol [3, 4] offers higher capacity, but its performance is affected by the problem of "hidden terminals." A third category (ISMA) [5–11], in which the central base station controls the flow of packets from the mobile terminals, reduces these two problems of collisions among data packets and hidden terminals.

The inhibit delay fraction [5–11] is a decisive parameter in designing a packet communication system using ISMA and is defined as the fraction of the packet duration necessary to inhibit other mobile packet transmissions by a broadcast outbound-signaling channel (base-to-terminal) in the event that the inbound multiple-access channel (terminal-to-base) is already busy.

ISMA is a class of protocols that we can divide into two subclasses: the nonpersistent ISMA protocol and the p-persistent ISMA protocols.

In the nonpersistent ISMA protocols, if the inbound channel is idle, a user can transmit a packet; otherwise, the user will wait a random time and then try again. At first sight it may seem that this protocol totally avoids collisions; however, due to the propagation delay between two users, it is possible that a user considers the inbound channel idle and starts transmission while another transmission is already in progress.

A user is informed of a collision by the absence of an acknowledgment packet from the receiving station. Upon detecting the collision, the packet is rescheduled for transmission at a randomly selected time later.

A special case of the p-persistent ISMA protocols is the 1-persistent ISMA protocol. The protocol is the same as the nonpersistent ISMA protocol except for when a user knows the inbound channel is busy. In this case, the transmission is not rescheduled for a randomly selected time later. Instead, the user keeps monitoring the channel until it becomes idle, and then immediately transmits its packet.

As a result, all users that become ready during a busy channel will transmit as soon as the channel becomes idle, which leads to the high probability of a collision occurring at the end of a successful transmission.

To avoid the collision of packets that accumulated while the channel was busy, the start of the transmission times of the accumulated packets can be randomized. This can be done by letting all users that generate a packet during a busy channel transmit as soon as the channel becomes idle with a probability p. With a probability 1 minus p, they will defer their transmission for τ seconds (with τ being the maximum propagation delay between any two users in the system). After the τ seconds, the deferred terminal will sense the channel again and apply the same algorithm as before.

With this p-persistent ISMA protocol, the probability of more than one ready user transmitting when the channel becomes idle can be made quite small by choosing a small value for p.

With the nonpersistent and p-persistent ISMA protocols, a user will not learn about a collision until after its whole packet has been transmitted. The reason for this is, of course, that an acknowledgment packet will only be sent after the complete packet has been received by the receiving user. Since a collision can only occur within the propagation delay after the start of the transmission, it is a waste of time to transmit more of the packet if a collision has occurred within this period. For this reason the ISMA with collision detect (ISMA-CD) protocols have been developed. With these protocols, a user keeps monitoring the channel while it is transmitting. If it detects a collision, it aborts its transmission as soon as possible, thus saving time.

6.4 Universal Condition in the Computer Simulation

6.4.1 The Modeling of the Packet Communication System

This section summarizes the universal assumption used when we perform computer simulation accurately and effectively.

6.4.1.1 Whole Configuration of Packet Communication System

As for the whole configuration of the simulated packet-communication system, each user terminal accesses an access point by using an access protocol, as shown in Figure 6.3.

6.4.1.2 Access Terminal

The performance of each user terminal is the same. Moreover, each terminal has a buffer. When each user terminal generates a transmission packet, the packet is stored in the buffer. Then, the stored packets are transmitted in accordance with the order by which packets came into the buffer. That is, the buffer circuit is a *first-in and first-out* (FIFO) buffer. As for the buffer circuit, we can categorize into two types: a buffer with infinite memory and a buffer with finite memory. When we use a buffer with finite memory and the memory is full, the packet generated in the user terminal is abandoned. This is called "call blocking," and it is different from the failure of a packet transmission to an access point. Moreover, if the number of access terminals is unlimited, we call the model an infinite call-source model. On the other hand, we call it a finite

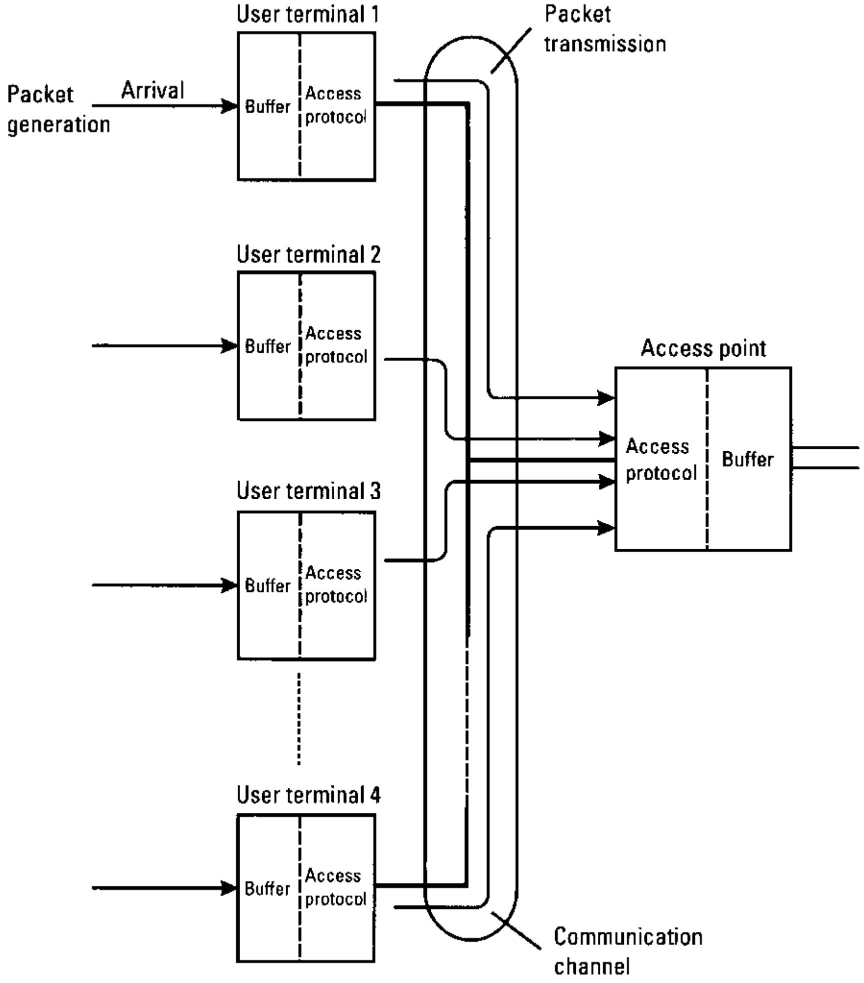


Figure 6.3 Configuration of the packet communication system.

call-source model when the number of access terminals is limited. In theoretical analysis, we sometimes assume the finite call-source model.

6.4.1.3 Communication Channel

The typical difference between wired and wireless communication systems is the modeling of the communication channel; it is modeled as follows.

- In the wired communication system, no transmission error occurs, and the signal powers received at the access point from the user terminals are the same. This is the most fundamental model in evaluating the access protocol.

- In the wireless communication system, the quality of the communication channel is time-variant. In this chapter, propagation loss—which is dependent on the distance between access terminals and access point and shadowing and is caused by obstacles between the access terminals and the access point—is taken into account. The propagation loss and shadowing are modeled as follows:
    - *Propagation loss:* The received-power monotonously decreases when the length between the access point and an access terminal increases. This is called the propagation loss, and it decreases in proportion to $\gamma^{-\alpha}$, where γ is the median value of received signal in large scale and α is attenuation constant. In a normal case, the value of α is 2 to 5 [28].
    - *Shadowing:* The transmission radio signal is received as a multiple radio wave that has passed through several routes by such events as reflection and diffraction. The received power intensely fluctuates by receiving the multiple radio wave. The fluctuation is called fading. The median value of the fading over several tens of wavelengths is called the median value in a small scale. The median value of the median value in a small scale over several tens of small scale is also fluctuated by obstacles and called shadowing. It is known that the fluctuation follows a logarithmic normal distribution with a standard deviation of 6–7 dB [28].

6.4.1.4 Packet Generation

Each access terminal is assumed to generate packets randomly and independently. Such packet generation is called the Poisson distribution. The following are characteristics of the Poisson distribution.

- *Independence:* The generation of a packet is independent of the past generation.
- *Constancy:* The probability of the packet does not change in each time slot of the simulation.
- *Rareness:* In a very small period, the probability in which more than two packets generated can be disregarded. Moreover, if the number of packets generated follows a Poisson distribution, the period until a packet is generated follows an exponential distribution.

6.4.1.5 Collision

Collisions of packets occur when several packets collide on the communication channel, as shown in Figure 6.4. In wired and wireless communication systems, the collided packets are handled as follows.

- *Wired communication system:* All collided packets are destroyed, and the transmission of a packet is regarded as a failure, because the signal levels of all packets are same. If no collisions occur, the generated packet is successively transmitted to the destination.
- *Wireless communication system:* The received power of each packet is dependent on the position of the access terminal and the condition of communication channel. Therefore, even if several packets come into collision, the packet that has the largest received power sometimes survives. Generally, the phenomenon is called the capture effect. On the other hand, even if a collision does not occur, a packet-transmission error occurs because the received power at the access point is smaller than the required power when the condition of the communication channel is worse.

In a real communication system, the access point decides whether the packets generated from the access terminals are successfully transmitted, and the results of the decision are transmitted to the access terminals. Moreover, if a packet-transmission error occurs, the packets are transmitted to the access point after some intervals.

6.4.1.6 Offered Traffic

In this chapter, the total quantity of the packets that include newly generated packets and retransmission packets at the access point in a time interval is called offered traffic and the normalized offered traffic by a transmission data rate is shown as G. If the transmission data rate is R (bps) and T_t (bit) is requested to transmit, G is as follows:

$$G = \frac{T_t}{R} \tag{6.1}$$

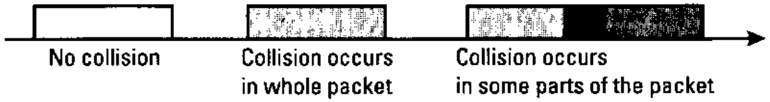

No collision
Collision occurs in whole packet
Collision occurs in some parts of the packet

Figure 6.4 Collision of the transmission packet.

If no packets are generated, $G = 0$.

6.4.1.7 Throughput

In this chapter, the total quantity of the packet that is successfully transmitted to the access point in a time interval is called *throughput*, and the normalized throughput by the data transmission data rate is shown as S. If the transmission data rate and the quantity of information in a packet are defined R(bps) and T(bit) and n packets are successfully transmitted in a time unit, S is as follows:

$$S = \frac{T \times n}{R} \tag{6.2}$$

If no transmission packets are generated and all transmission packets are destroyed by collisions, S becomes the minimum value 0. On the other hand, if all packets can be transmitted over all time units perfectly, the throughput becomes 1.

6.4.1.8 Average Transmission Delay

The period until a packet is generated at an access terminal, transmitted to the access point, and received at the access point is called the average transmission delay. The average transmission delay is dependent on the length of packet. Therefore, the normalized average transmission delay by the length of the packet is shown as D. Originally, the average transmission delay is dependent on the period when a packet is generated and transmitted from an access terminal, and the distance between the access point and an access terminal and signal processing time at access point. However, the period assumed is quite small.

6.4.1.9 Evaluation of the Access Protocol

The most fundamental elements used to evaluate the access protocol are offered traffic G, throughput S, and average transmission delay D. In the ideal access protocol, the throughput is given by

$$S = \begin{cases} G & (G < 1) \\ 1 & (G \geq 1) \end{cases} \tag{6.3}$$

However, as shown in Figure 6.5, S increases as G increases. In addition, if S is larger than a threshold value, S decreases as G increases. If G is larger than 1, D drastically increases and finally reaches infinity. The ideal average-transmission delay is obtained when all of the packets can be scheduled to transmit to the destination perfectly by using a transmission trial and there is no processing time needed for scheduling. When using the notation by Kendall, the process is M/D/1 as shown in Figure 6.6.

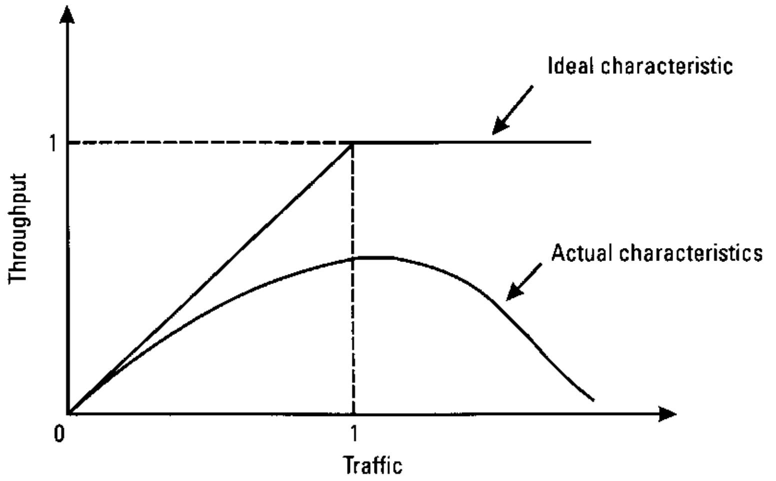


Figure 6.5 Characteristics of traffic and throughput.

6.4.2 Fundamental Configuration of the Simulation Program

This section describes the procedure to evaluate the throughput and delay of an access protocol by computer simulation. The basic configuration of the

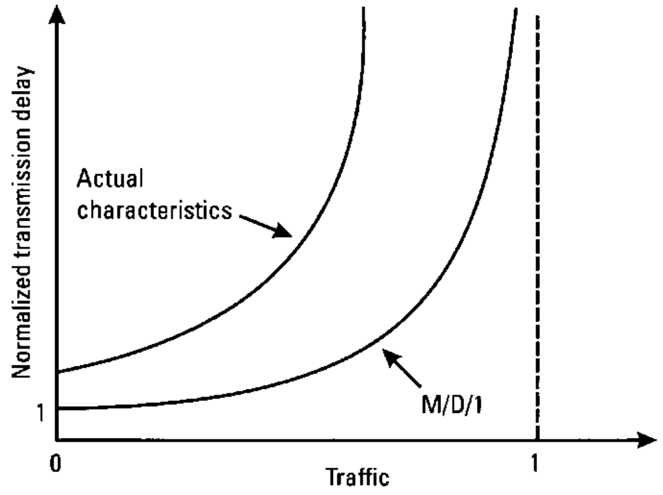


Figure 6.6 Characteristics of throughput and average transmission delay.

computer simulation is shown in Figure 6.7. In the computer simulation discussed in this section, it is assumed that propagation loss and shadowing are

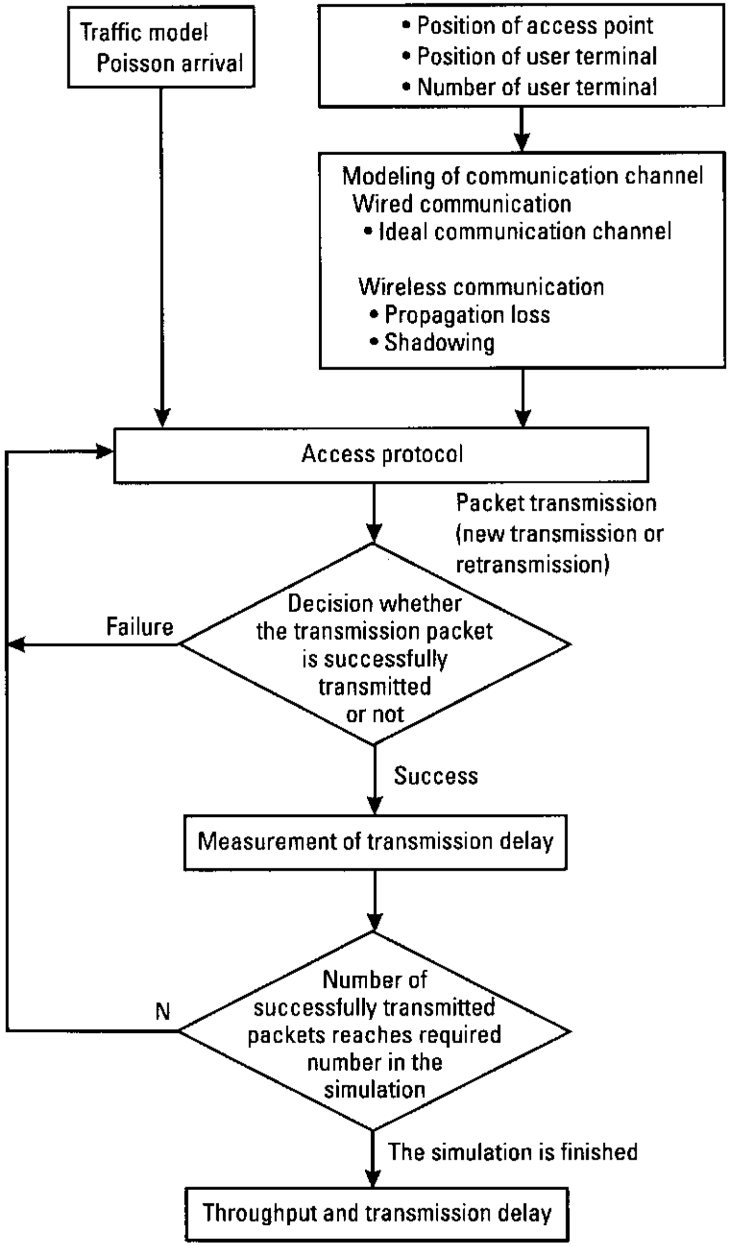


Figure 6.7 Basic configuration of computer simulation.

constant. Until the number of successfully transmitted packets is equal to the required number, the computer simulation is continued. Moreover, by making and changing several subprograms, the performance of several access protocols are evaluated.

6.4.3 Main Program

The main program is shown in Program 6.1. In the program, the following procedures are performed. All simulation parameters are summarized in Table 6.2.

1. The variables that are commonly used in both the main and the subprograms are defined as globally variable, to save memory and make the simulation run faster.

```
global STANDBY TRANSMIT COLLISION PERMIT
global Srate Plen Ttime Dtime
global Mnum Mplen Mstime Mstate
global Tint Rint
global Spnum Splen Tplen Wtime
```

2. The constant used in the program is universally defined.

```
STANDBY     = 0;
TRANSMIT    = 1;
COLLISION   = 2;
PERMIT      = 3;
```

3. The name of the subprogram for each access protocol is defined by the array configuration. By arraying the name of subprogram and using `feval` function, it is easy to add and delete the names of subprograms in comparison with the usage of statement: `if` or `switch`.

```
protocol = { 'paloha'   ...% Pure ALOHA          : pno = 1
             'saloha'   ...% Slotted ALOHA       : pno = 2
             'npcsma'   ...% Np-CSMA             : pno = 3
             'snpisma'  ...% Slotted np-ISMA     : pno = 4
         } ;
```

4. The parameters for the simulation are defined. Here, `tcn` is the threshold value of the received-power by which no packet error occurs at the access point. The `tcn` is defined as C/N (dB) at the access point and named capture ratio. The `mcn` is the transmission power of an access terminal and is defined as C/N (dB) at the access point when the transmission data generated at the end-of-service area arrives at

Table 6.2
Simulation Parameters

Variable Name	Note
brate	Bit rate (bps)
Srate	Symbol rate (sps)
plen	Length of a packet (symbol)
Ttime	Transmission time of a packet(s)
Dtime	Normalized propagation delay
alfa	Decline fixed number of propagation loss
sigma	Standard deviation of shadowing (dB)
capture	Capture effect: 0:nothing 1:consider
r	Service area radius (m)
bxy	Position of the access point (x, y, z) (m)
tcn	Capture ratio (dB)
Mnum	Number of the access terminals
mxy	Position of the access terminals (x, y, z) (m)
mcn	C/N at the access point when transmitted from area edge (dB)
Mplen	Length of a packet (symbol)
Mstate	State of the access terminals
mgtime	Packet generation time in the access terminal (seconds)
mtime	State change time of the access terminal (seconds)
Mstime	Packet transmitting time of the access terminal (seconds)
G	Offered traffic
Tint	Expectation value of the packet generation interval (seconds)
Rint	Expectation value of the packet resending interval (seconds)
Spnum	Number of successfully transmitted packets
spend	Number of packets that simulation is finished
Splen	Data amount of successfully transmitted packets (symbol)
Tplen	Data amount of packets that is tried the transmission (symbol)
Wtime	Transmission delay time (seconds)

the access point suffering from propagation loss. The pno selects an access protocol by designating an index from an array shown in procedure (3). Moreover, if we choose not to include capture effect in the simulation, the program is simulated under a wired communication

channel (ideal communication channel). Moreover, if we select to include capture effect in the simulation, the program is simulated under a wireless communication channel. Then, the total number of packets that can be transmitted to the destination successfully is the condition to terminate the simulation program and is shown as spend. In the variable outfile, the name of the file to store the simulation results is designated as:

```
brate     = 512e3;      % bit rate
Srate     = 256e3;      % symbol rate
Plen      = 128;        % length of packet
Dtime     = 0.01;       % normalized propagation delay
alfa      = 3;          % fixed number of
                        % propagation loss
sigma     = 6;          % standard deviation of
                        % shadowing
r         = 100;        % service area radius
bxy       = [0, 0, 5];  % position of access point
tcn       = 10;         % capture ratio
Mnum      = 100;        % number of access terminals
mcn       = 30;         % C/N
pno       = 2;          % protocol number
capture   = 0;          % capture effect
spend     = 1000;       % simulation termination
                        % condition
outfile   = 'test.dat'; % name of file to store the
                        % result
```

5. From the parameters defined in procedure (4), the period to transmit a packet to the destination and the C/N at the access point are calculated. To calculate the C/N at the access point, it is assumed that the access terminal transmits packets from the point of 0[m].

```
Ttime = Plen / Srate;
mpow  = 10^(mcn/10) * sqrt(r^2+bxy(3)^2)^alfa;
```

6. The parameters defined in procedure (4) are stored in a file.

```
fid = fopen(outfile,'w');
fprintf(fid,'Protocol                   = %d\n',pno);
fprintf(fid,'Capture                    = %d\n',capture);
fprintf(fid,'Normalize_delay_time       = %f\n',Dtime);
fprintf(fid,'Bit_rate (bps)             = %d\n',brate);
fprintf(fid,'Symbol_rate (sps)          = %d\n',Srate);
fprintf(fid,'Length_of_Packet(sbl)      = %d\n',Plen);
fprintf(fid,'Number_of_mobile_station = %d\n',Mnum);
fprintf(fid,'Transmission_power (C/N)    = %f\n',mcn);
fprintf(fid,'Capture_ratio (dB)          = %f\n',tcn);
fprintf(fid,'Number_of_Packet           = %d\n',spend);
```

7. All access terminals are arranged randomly. Then, shadowing between access terminals and access points is given from a random value with normal distribution.

```
mxy  = position(r,Mnum,0);          % access terminals
                                    % configuration
randn('state',sum(100*clock));      % reset of table of
                                    % random numbers
mrnd = randn(1,Mnum);               % shadowing
```

8. Offered traffic in the simulation is given by the index `G` in the `for` loop. Here, the value of offered traffic must be less than the number of access terminals. If `G` is larger than the number of access terminals, the simulation is terminated.

```
for G=[0.1:0.1:2.0]             % G : offered traffic
   if G >= Mnum
      break
   end
```

9. The expectation value of the generation interval of the packet is calculated from the number of terminals. Then, the expectation value of the packet resending interval is set to be equal to the expectation value of the generation interval. In the real system, the packet-resending interval is dependent on the simulated access protocol. For example, in the case of slotted ALOHA, packets are sent within 10 time slots. However, if the packet-resending interval is fixed within a special range, the actual offered traffic is different from the setting parameter of `G`. Here, to obtain the throughput and average transmission delay in `G` is one of the purposes of the simulation program. Therefore, the packet-resending interval must follow the exponential distribution that is used in the packet-generation interval.

```
Tint  = -Ttime / log(1-G/Mnum);  % generation interval
Rint  = Tint;                    % re-sending interval
```

where the derivation of `Tint` and `Rint` is shown in Appendix 6A by using Poisson and exponential distribution.

10. The variables used in the simulation are initialized.

```
Spnum = 0;
Splen = 0;
Tplen = 0;
Wtime = 0;
now_time = feval(char(protocol(pno)),-1);
```

11. The simulation program in the `while` loop is a core program. The access protocol is programmed in the subprogram. Each detail of pure ALOHA, slotted ALOHA, np-CSMA, and slotted np-ISMA is explained in Sections 6.5, 6.6, 6.7, and 6.8, respectively. The fundamental operation is shown as follows. The states of all access terminals are stored in a global variable `Mstate`. Then, the time in which the terminal condition changes is returned.

```
next_time = feval(char(protocol(pno)),now_time);
```

12. If the total number of packets that transmits from access terminals to the access point successfully is larger than a defined variable `spend`, the simulation in a `G` is terminated.

```
if Spnum >= spend
  break
end
```

13. The access terminals that transmit their packets to the access point are counted.

```
idx = find(Mstate==TRANSMIT | Mstate==COLLISION);
```

14. This is where it is determined whether the transmitted packets from the access terminals are successfully received at the access point when the capture effect is not considered. In procedure (12), if there are more than two terminals that transmit packets, a collision occurs. In this case, because the transmission of packets failed, the data `COLLISION` is stored in the variable of `Mstate`.

```
if capture == 0              % without capture effect
   if length(idx)  1         % transmission terminals
                             % is over two
      Mstate(idx) = COLLISION;   % collision occurs
   end
```

15. This is where it is determined whether the transmitted packets from access terminals are successfully received at the access point, when the capture effect is considered. By the capture effect, the packet with maximum received power may survive at the access point. Therefore, first of all, the distance between the access point and the access terminals that transmit packets is calculated. Then, the C/N is calculated at the access point for all packets that are transmitted from access terminals, and the packet that obtains the maximum C/N is selected. Then, for the selected packet, the instantaneous C/N is calculated again by

regarding the other packets as noise. If the calculated C/N is larger than threshold, the instantaneous data in the packet is transmitted to the access point successfully, and the result is stored in the variable Mstate. On the other hand, if the calculated C/N is lower than the threshold, all packets from users are not transmitted to the access point, and the result is stored in the variable Mstate. When Mstate is 'TRANSMIT' during a packet transmission, the packet transmission is successfully finished; otherwise, collision occurs.

```
else                                % with capture effect
   if length(idx) > 1
       dxy  = distance(bxy,mxy(idx,:),1);
       pow  = mpow * dxy.^-alfa .*...
           10.^(sigma/10*mrnd(idx));
       [maxp no] = max(pow);
       if Mstate(idx(no)) == TRANSMIT
           if length(idx) == 1
               cn = 10 * log10(maxp);
           else
               cn = 10 * log10(maxp/(sum(pow)-maxp+1));
           end
           Mstate(idx) = COLLISION;
           if cn = tcn
           end
           Mstate(idx(no)) = TRANSMIT;
       else
           Mstate(idx) = COLLISION;
       end
   end
end
```

16. The time counter increases when the state of each access terminal is changed (e.g., a new packet is generated).

```
now_time = next_time;
```

17. When a simulation is finished for a value of G, the results, such as average traffic, throughput, and average transmission delay, are shown on the display, and these data are saved in the file. Here, average traffic is defined as the value that divides the period to transmit all packets by the total simulation period. Moreover, the throughput is defined as the value that divides the period into transmit packets that are successfully received at the access point by the total simulation period.

```
traffic   = Tplen / Srate / now_time;
ts        = theorys(pno,traffic,Dtime);
fprintf(fid,'%f\t%f\t%f\t%f\t%f\t%f\t%f\n',G,now_time ...
```

```
      ,Splen/Srate,Tplen/Srate,Wtime,Tint,Rint);
fprintf('G=%f\tS=%f\tTS=%f\n',traffic,Splen/Srate/...
   now_time,ts);
```

18. When all simulations are finished in setting the parameter of G, the file that stores the simulation results is closed, and the performance will be shown in the display.

```
fclose(fid);
graph(outfile);
```

6.4.4 Explanation of Subprogram

6.4.4.1 Subprogram for Positioning Access Terminals

Subprogram position.m is shown in Program 6.2. The input arguments are r, n, and h, where r is the radius of a cell. n users are positioned randomly in a circular zone with the radius of r, as shown in Figure 6.8. The center of the cell is the origin (0,0). The positions for all users must be different. h means the height of each access terminal. When setting h=1, the height of each access terminal takes a random value from 1 to 4. The information of the decided positions for all users becomes a matrix of the number of user terminals by 3.

6.4.4.2 Subprogram for Calculating the Distance Between the Access Point and the Access Terminals

Subprogram distance.m is shown in Program 6.3. The input arguments are bstn, mstn, and scl. When scl is omitted, scl=1. bstn indicates the position

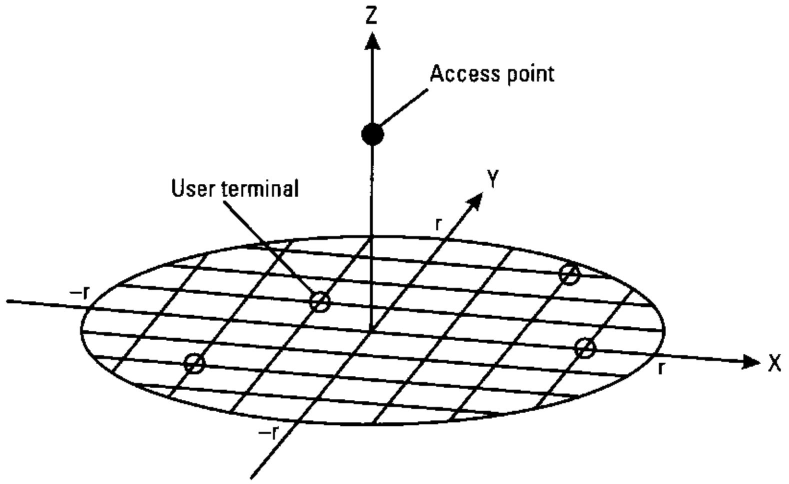


Figure 6.8 Layout of access point and user terminals.

of an access point by using x, y, and z coordinates shown in Figure 6.8, and mstn indicates the positions of access terminals by using x, y, and z coordinates shown in Figure 6.8. The output is scl times the distance between bstn and mstn.

6.4.4.3 Subprogram to Calculate the Theoretical Value of Throughput

Subprogram theorys.m is shown in Program 6.4. The input arguments are no, g, and a. The no denotes a selection number. When no=1, 2, 3, and 4, pure ALOHA, slotted ALOHA, nonpersistent CSMA, and slotted np-ISMA are selected as the access protocol, respectively. g is a given traffic and a is the normalized transmission delay. By inputting g and a for the protocol with a selection number no, the theoretical throughput is calculated.

6.4.4.4 Subprogram to Show the Simulation Results and Theoretical Value as a Figure

Subprogram graph.m is shown in Program 6.5. The graph displays the results between offered traffic and throughput and between offered traffic and average transmission delay. The input argument is filename, which designates the name of file that stores the simulation results.

6.5 Pure ALOHA

6.5.1 Principle

Pure ALOHA is a protocol in which access terminals transmit their packets as they like. The protocol was proposed by the University of Hawaii in 1970. In this case, all the access terminals do not mind whether the communication channel is busy or not. The basic configuration is shown in Figure 6.9.

If the length of each packet is fixed and the period to transmit a packet is T, a packet can be successfully transmitted to the access point when other packets do not start to transmit during $2T$ from $t_1 - T$ to $t_1 + T$, as shown in Figure 6.10. Therefore, the throughput S of pure ALOHA for the offered traffic G is shown as follows:

$$S = Ge^{-2G} \tag{6.4}$$

where the infinite call-source model is assumed, and the maximum throughput is 0.184 when $G = 0.5$. The detailed derivation of the throughputs is shown in Appendix 6B.

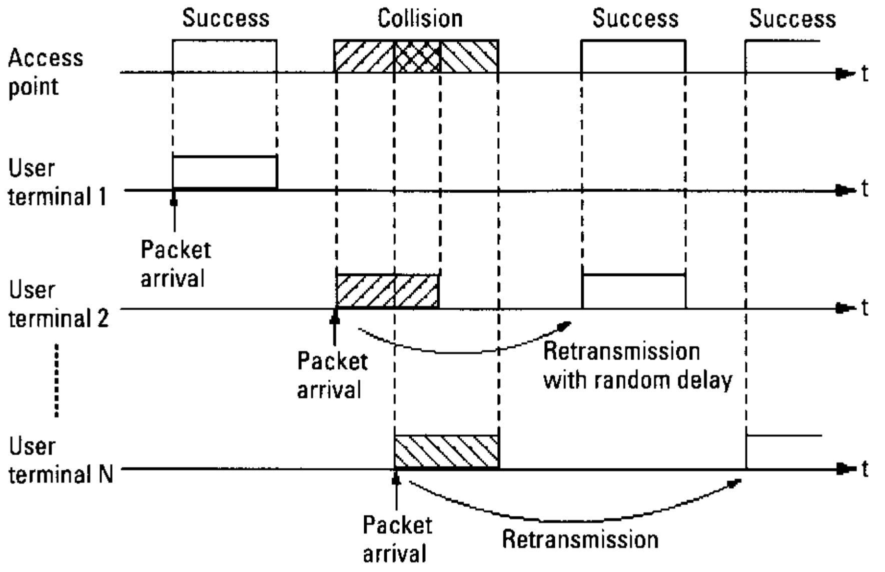


Figure 6.9 Pure ALOHA.

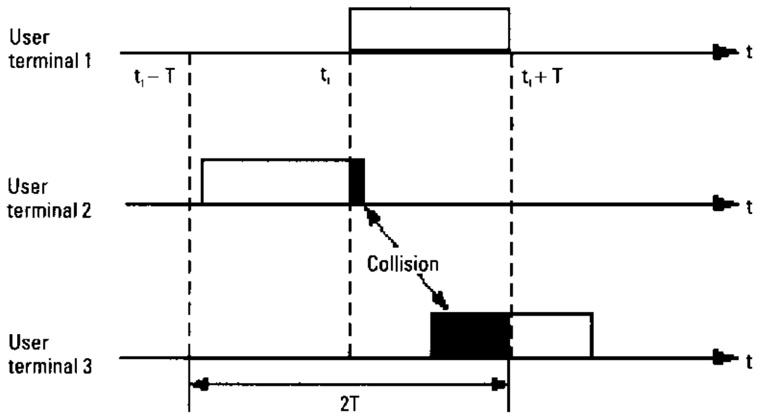


Figure 6.10 Collision of packets in pure ALOHA system.

6.5.2 Simulation Program

Subprogram paloha.m is shown in Program 6.6. The input argument is now_time, which shows the current time. And the output is next_time, which shows the time when the state of each access terminal is changed after performing the function in paloha. The function of paloha can be divided into six functionalities. Table 6.3 shows a simulation condition.

Table 6.3
Simulation Condition

r:	Service area radius	100m
bxy:	Height of the access point	5m
Mnum:	The number of terminals	100
Srate:	Symbol rate	256 ksymbol/second
Plen:	Length of the packet	128 symbol
alfa:	Distance attenuation fixed number	3
sigma:	Standard deviation of the logarithm normal distribution	6 dB
mcn:	C/N in the access point when transmitted from the area edge	30 dB
tcn:	Capture ratio	10 dB

1. In the first block, global variables and static variables are defined.

```
global STANDBY TRANSMIT COLLISION
global Srate Plen
global Mnum Mplen Mstate
global Tint Rint
global Spnum Splen Tplen Wtime
persistent mgtime mtime
```

2. In the second block, the states of all access terminals are initialized when `now_time` < 0. In this block, the initial time when the first packet is generated and the length of packet are defined.

```
if now_time < 0
   rand('state',sum(100*clock));
   mgtime              = -Tint * log(1-rand(1,Mnum));
   mtime               = mgtime;
   Mstate              = zeros(1,Mnum);
   Mplen(1:Mnum)       = Plen;
   next_time           = min(mtime);
   return
end
```

3. In the third block, when packet transmission is successfully finished, the number of packets that are successfully transmitted is counted, the transmission delay is calculated, and the time when the new packet is generated is calculated for all terminals. Here, in each terminal the new packet is not generated until the packet generated in each channel is successfully transmitted to the access point.

```
idx = find(mtime==now_time & Mstate==TRANSMIT);
if length(idx) > 0
   Spnum          = Spnum + 1;
   Splen          = Splen + Mplen(idx);
   Wtime          = Wtime + now_time - mgtime(idx);
   Mstate(idx)    = STANDBY;
   mgtime(idx)    = now_time - Tint * log(1-rand);
   mtime(idx)     = mgtime(idx);
end
```

4. In the fourth block, when packet transmission is finished and the transmission has failed, the time when the packet is retransmitted is calculated for all terminals.

```
idx = find(mtime==now_time & Mstate==COLLISION);
if length(idx) > 0
   Mstate(idx)         = STANDBY;
   mtime(idx)          = now_time - Rint * ...
       log(1-rand(1,length(idx)));
end
```

5. In the fifth block, the access terminals that transmit a packet at `now_time` are found for all access terminals. Then, the state changes to `TRANSMIT` mode, the end time to finish the packet transmission is calculated, and the number of transmitted packets is counted.

```
idx = find(mtime==now_time);
if length(idx) > 0
   Mstate(idx)         = TRANSMIT;
   mtime(idx)          = now_time + Mplen(idx) / Srate;
   Tplen               = Tplen + sum(Mplen(idx));
end
```

6. In the sixth block, by finding the minimum value from `mtime`, `next_time`, which is the nearest time when the state of each access terminal is changed, is decided.

```
next_time = min(mtime);
```

6.5.3 Simulation Results

By executing the main program, `main.m`

```
>>main
```

the performances of throughput and average transmission delay and evaluation are determined; the simulation results are shown in Figures 6.11 and 6.12.

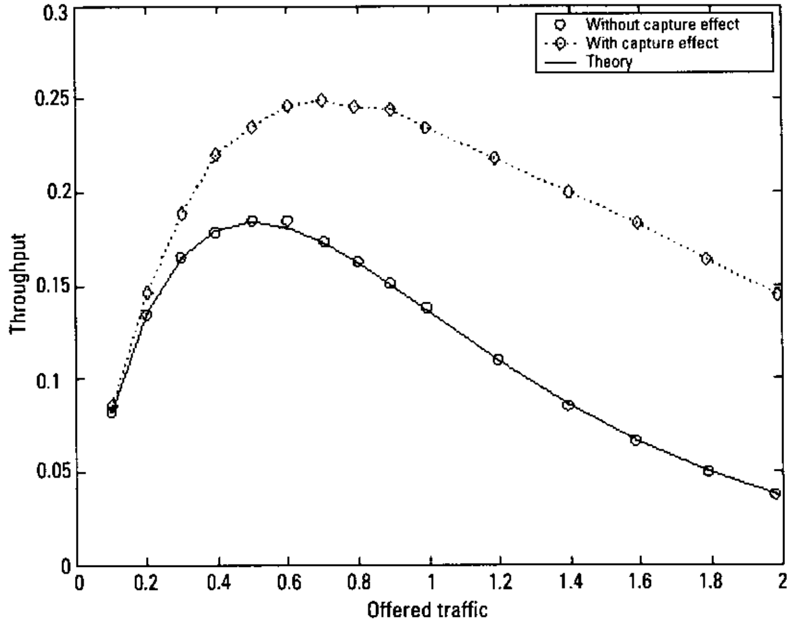


Figure 6.11 Offered traffic and throughput of pure ALOHA.

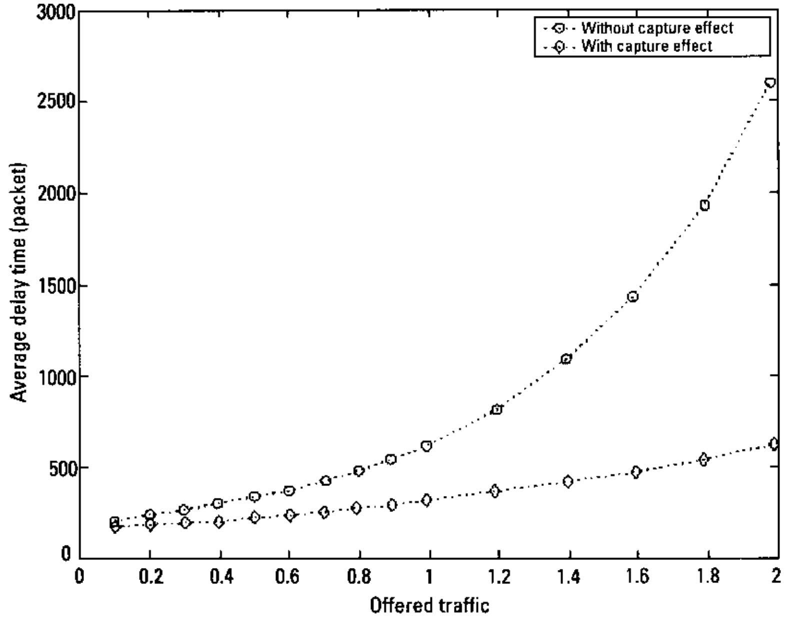


Figure 6.12 Offered traffic and average delay time of pure ALOHA.

When the capture effect is not considered, the throughput is close to the theoretical value even if the number of users is 100. Moreover, when the capture effect is considered, the throughput is larger than when the capture effect is not considered. In addition, the average transmission delay is reduced. In the case of pure ALOHA, collisions often occur; therefore, the capture effect is the reason to increase the throughput.

6.6 Slotted ALOHA

6.6.1 Principle

With a simple modification of pure ALOHA, namely, that the messages are required to be sent in the slot time between two synchronization pulses, and can be started only at the beginning of a time slot, the rate of collisions can be reduced by half [29]. The protocol is called slotted ALOHA. The configuration is shown in Figure 6.13. In Figure 6.13, the packet generated in a time slot is transmitted in the next time slot. To transmit the packet to the access point successfully, the number of packets that is generated in a time slot must become 1, as shown in Figure 6.14. If more than two packets are generated in a time

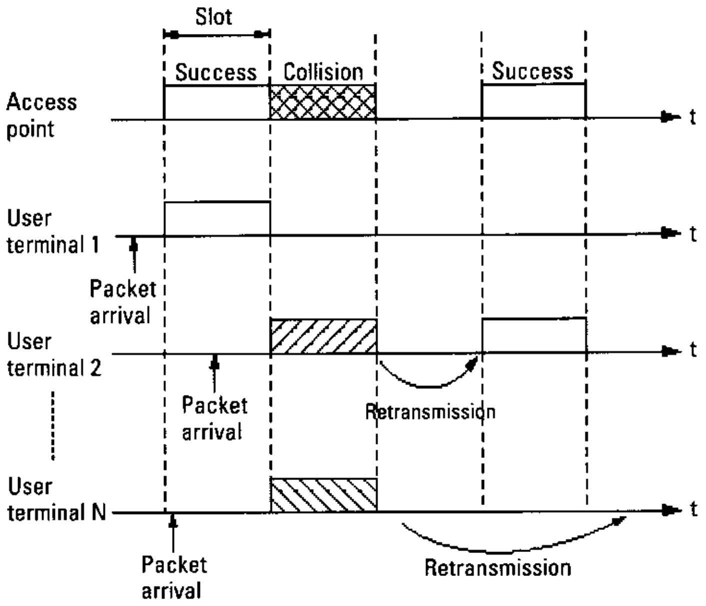


Figure 6.13 Slotted ALOHA.

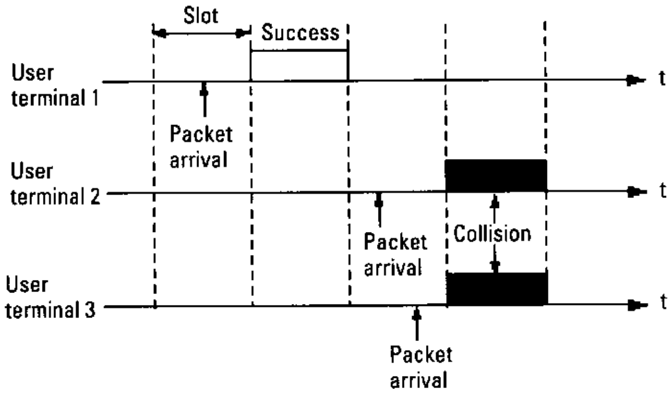


Figure 6.14 Collision of packet in slotted ALOHA system.

slot, a collision occurs. Therefore, the throughput S of slotted ALOHA for the offered traffic G is given as follows:

$$S = Ge^{-G} \tag{6.5}$$

where infinite call-source model is assumed, and the maximum throughput is 0.368 when $G = 1$. The detailed derivation of the throughputs is shown in Appendix 6B.

6.6.2 Simulation Program

Subprogram `saloha.m` is shown in Program 6.7. The program is fundamentally equal to `paloha.m`. However, the timing to transmit the packet is synchronized with the time slot.

6.6.3 Simulation Results

The simulation results of throughput and average transmission delay are shown in Figures 6.15 and 6.16. The influence of the capture effect is remarkable, as shown in Section 6.5.3.

6.7 Nonpersistent CSMA

6.7.1 Principle

The name carrier stems from the fact that the existence of the carrier wave on the communication channel is sensed by access terminals. By using carrier sense, it is possible to judge whether other access terminals are transmitting their packets, because each access terminal does not transmit any carrier wave

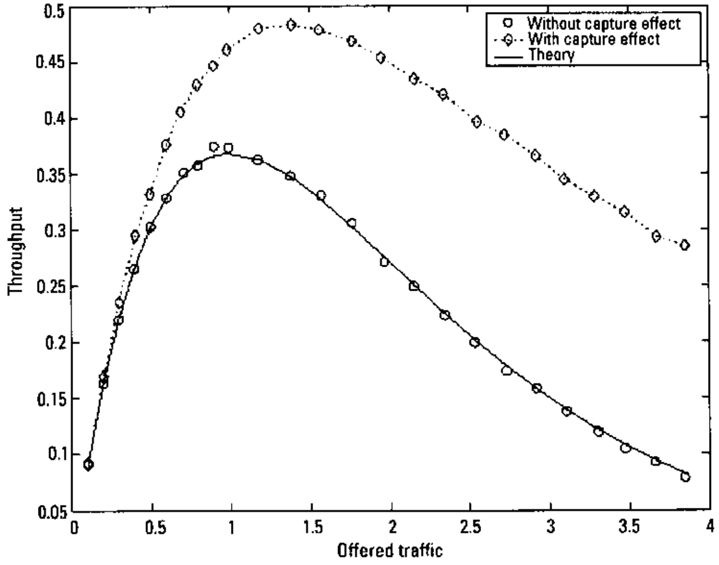


Figure 6.15 Offered traffic and throughput of slotted ALOHA.

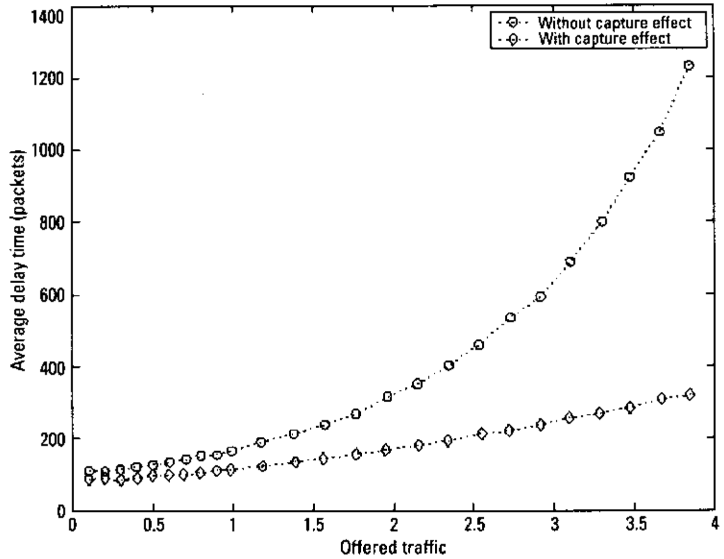


Figure 6.16 Offered traffic and average delay time of slotted ALOHA.

except for its packet transmission, as shown in Figure 6.17. If a carrier wave is sensed on the communication channel, the condition is called "busy"; otherwise, it is called "idle." The CSMA is a protocol that decides whether packet transmission should start as the result of a carrier sense. When the result is "busy," the next action to avoid collision is needed.

Nonpersistent CSMA, presented in this section, is one of the protocols that avoids collision. In the nonpersistent CSMA, when packets are generated in an access terminal, the access terminal starts the carrier sense. If the result of a carrier sense is "idle," the packet is transmitted to the access point immediately. However, if the result of carrier sense is "busy," the access terminal stops the carrier sense, waits for a while, and then starts the carrier sense again. The waiting time is a key point to realize a system with high throughput.

In the CSMA, the collision of packets occurs although each access terminal performs carrier sense. One of the reasons is propagation delay time. In a real communication system, when an access terminal transmits its packets, other access terminals detect the transmission from the propagation delay time. If other terminals transmit their packets during the propagation delay time, collisions occur at the access point, as shown in Figure 6.18. The propagation delay time is dependent on the distance between access terminals. In most cases, it is assumed that the propagation delay time is the same in the system and that the normalized propagation delay time that is normalized by the period is needed for packet transmission from an access point to each access terminal. Moreover, in a wireless communication system, the carrier sometimes cannot be sensed at some access terminals even if an access terminal transmits packets, as shown in Figure 6.19, because some obstacles exist between the access terminals. The problem is called the "hidden terminal" problem.

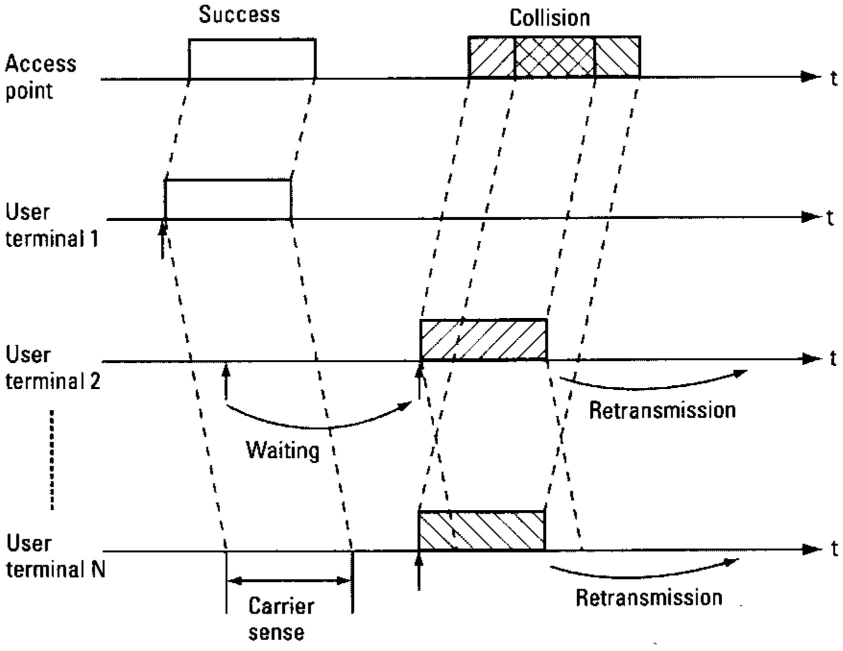


Figure 6.17 Nonpersistent CSMA.

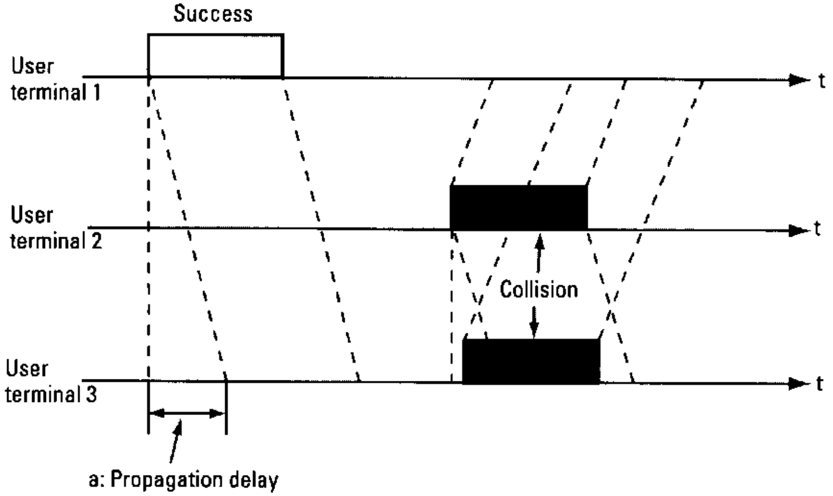


Figure 6.18 Collision of packet in np-CSMA.

The throughput S of nonpersistent CSMA for the offered traffic G is shown as follows:

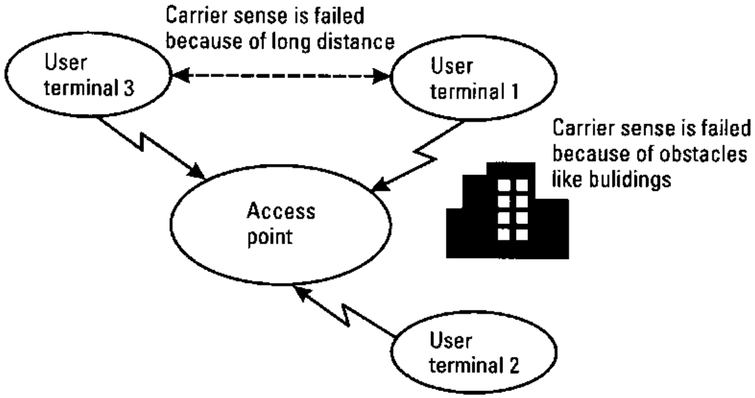

Carrier sense is failed because of long distance
User terminal 3
User terminal 1
Carrier sense is failed because of obstacles like bulidings
Access point
User terminal 2

Figure 6.19 Hidden terminal problem.

$$S = \frac{Ge^{-aG}}{G(1+2a)+e^{-aG}} \tag{6.6}$$

where a is the normalized propagation delay mentioned in this section. The throughput S is obtained under an ideal communication environment where an infinite call-source model is assumed, and no hidden terminal exists. The detailed derivation of the throughput is shown in Appendix 6B.

6.7.2 Simulation Model

Subprogram `npcsma.m` is shown in Program 6.8. This subprogram is fundamentally equal to `paloha.m`. The configuration of `npcsma` is categorized into the five blocks discussed in Section 6.5.2 and is different between `npcsma` and `paloha`. In `npcsma`, the carrier is sensed before sending a packet for each access terminal by using a function of `carriersense.m`. If the result is “‘idle,” a packet is transmitted to the access point. If the result is “busy,” resending time of packet is scheduled. All information on the sending times of packets for all access terminals is stored in the global variable `Mstime`.

```
idx = find(mtime==now_time & Mstate==STANDBY);
if length(idx) > 0
   Tplen = Tplen + sum(Mplen(idx));
   for ii=1:length(idx)
      jj = idx(ii);
      if carriersense(jj,now_time) == 0
         Mstate(jj) = TRANSMIT;
         Mstime(jj) = now_time;
         mtime(jj)  = now_time + Mplen(jj) / Srate;
      else
         mtime(jj) = now_time - Rint * log(1-rand);
      end
   end
end
```

Next, subprogram `carriersense.m` is shown in Program 6.9. In this program, `Mstime`, which shows the sending time of a packet, is added to the propagation delay time, which is dependent on the distance between access terminals. Then, the carrier is sensed for all terminals. If the carrier is sensed, then the condition is set as “busy” (`result=1`), and otherwise the condition is set as “idle” (`result=0`).

6.7.3 Simulation Results

The simulation results of throughput and average transmission delay are shown in Figures 6.20 and 6.21. In the simulation, the normalized propagation delay

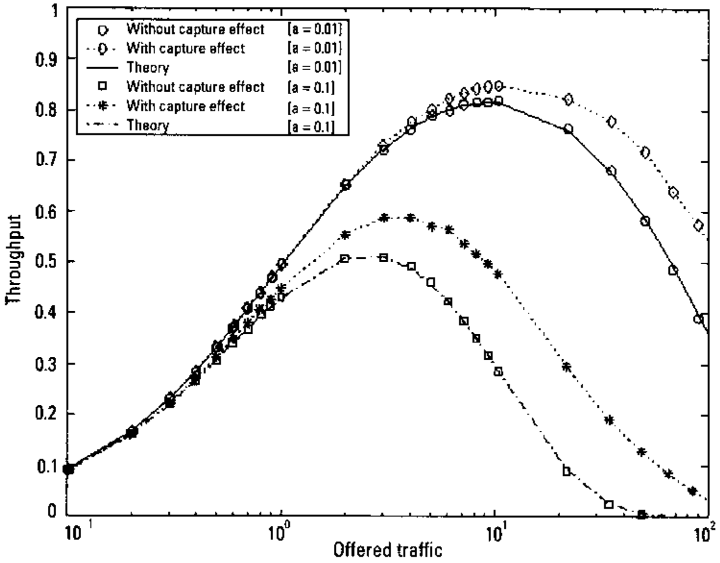


Figure 6.20 Offered traffic and throughput of np-CSMA.

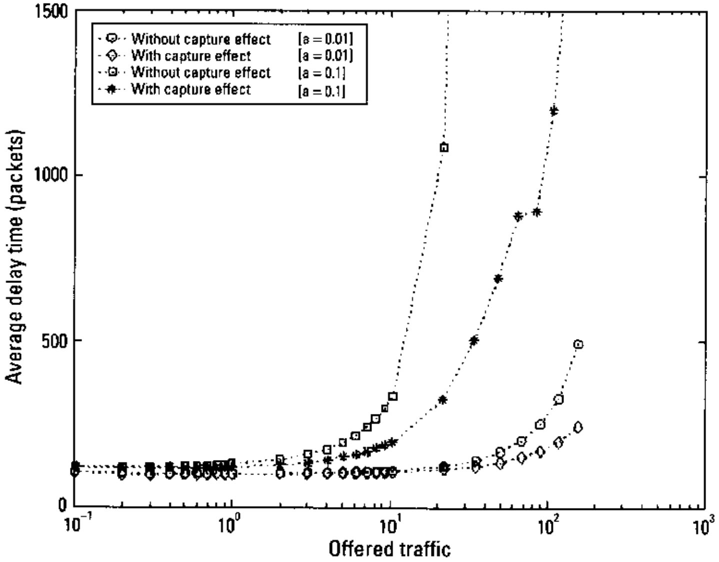


Figure 6.21 Offered traffic and average delay time of np-CSMA.

a was 0.01 or 0.1. The maximum throughput of np-CSMA was dependent on the normalized propagation delay. If a is small, the maximum throughput of np-CSMA is higher than that of pure ALOHA. However, if *a* is large, the performance is close to pure ALOHA. This is because other access terminals transmit their packets during a large propagation delay time even if an access terminal transmits its packet after sensing the carrier. As a result, collision occurs. Moreover, if the capture effect is considered, a transmitted packet sometimes survives because of the difference of received-power between transmitted access terminals. Therefore, throughput increases.

6.8 Slotted Nonpersistent ISMA

6.8.1 Principle

CSMA reduces the probability of collisions by allowing terminals to detect the carrier of the other users' transmissions. But CSMA cannot avoid collision if two terminals are out of range of each other or if they are separated by some physical obstacle. Two such terminals are considered hidden from each other. ISMA is one solution that has been offered to solve the hidden access-terminal problem.

The ISMA is a system that can solve hidden access-terminal problems by informing the communication channel of busy or idle conditions from the access point to the access terminal. The concept is shown in Figure 6.22.

In the ISMA, the access point sends a busy signal to all access terminals when the access point is receiving packets from access terminals. On the other hand, the access point sends an idle signal when the access point is not receiving any packets. When each access terminal receives the idle signal, each access terminal must decide whether to transmit packets to the access point or not. When each access terminal receives the busy signal, the packet transmission of each access terminal is inhibited. Therefore, the protocol is called ISMA. Fundamentally, ISMA may be regarded as CSMA without considering a hidden terminal. However, the propagation delay time of ISMA is twice that of CSMA, because all access terminals detect only the signal from the access point, and the access point needs to receive packets from the access terminals and then sends a busy signal to all access terminals. In CSMA, an access terminals sense the signal from the other access terminals. Therefore, ISMA needs the twice-propagation delay. If other terminals transmit their packets during the propagation delay time, collisions occur at the access point, as shown in Figure 6.23. Here, the normalized propagation delay time is normalized by the period it needs for the transmission of one packet. Moreover, in ISMA, the downlink is from the access point to access terminals, and uplink is from access terminals to the access point.

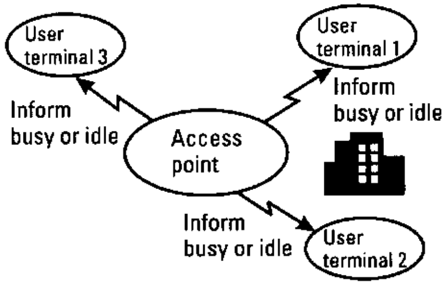


Figure 6.22 ISMA.

This section focuses on slotted nonpersistent ISMA, which is shown in Figure 6.24. In this protocol, an uplink is slotted by the period of one packet *T*. When each access terminal generates its packet, the access terminal performs inhibit sense. If it receives an idle signal, the access terminal transmits a packet at the next time slot. On the other hand, if it receives a busy signal, the access terminal does not transmit and waits until the next time slot, which is decided randomly and then starts the inhibit sense. Here, in the access point, the time slot when the access point transmits a busy signal to all the access terminals is called a busy slot. Moreover, the time slot when the access point transmits an idle signal to all access terminals is called an idle slot. The access point informs all access terminals of the next timing to transmit packets when it has finished receiving packets from all users. However, if the access point does not receive

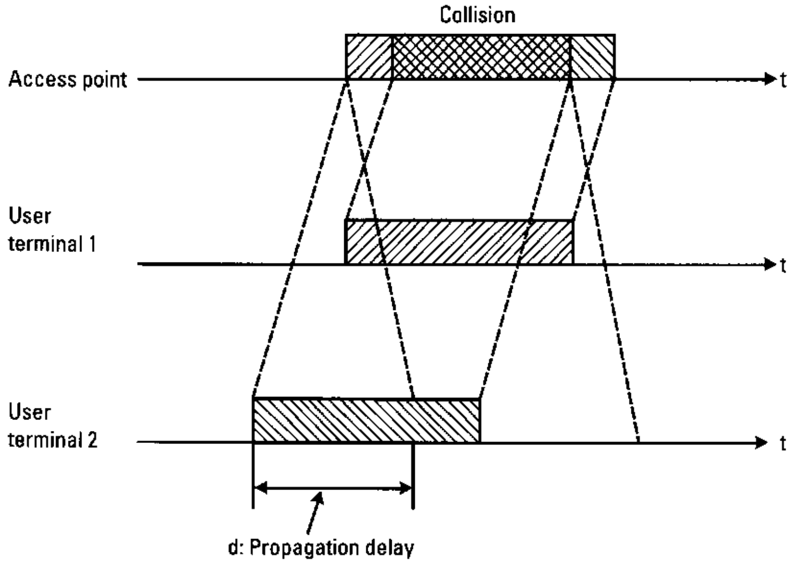


Figure 6.23 Collision of packet in ISMA.

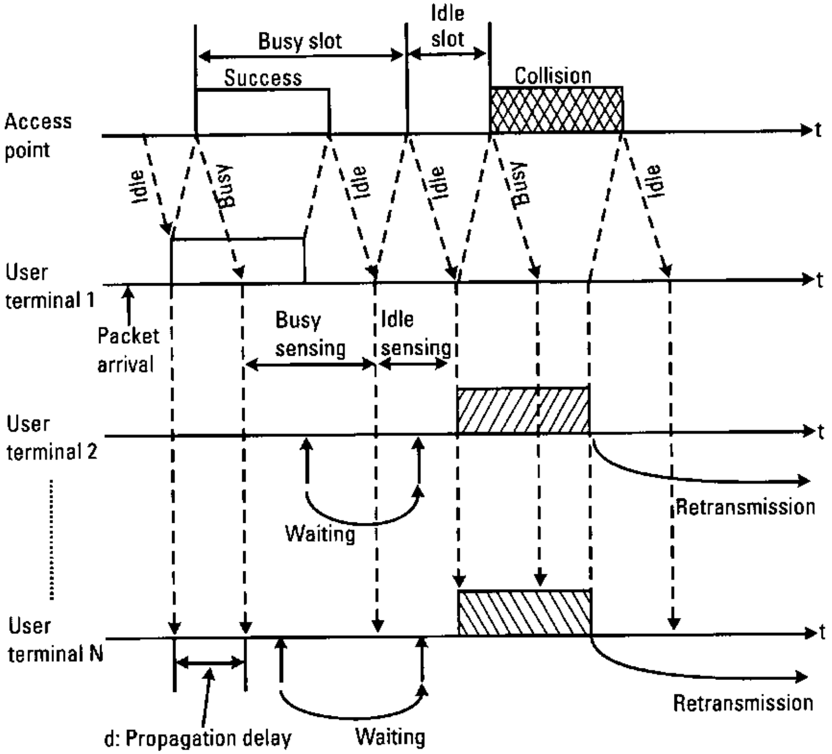


Figure 6.24 Slotted nonpersistent ISMA.

any packets, the access point also informs the terminals of the next timing. By taking a normalized propagation delay d, the busy slot must be more than $(1 + d)T$ and the idle slot must be at least more than dT. The throughput S of slotted nonpersistent ISMA for the offered traffic G is shown as follows:

$$S = \frac{dG^{-dG}}{1 + d - e^{-dG}} \tag{6.7}$$

The throughput is obtained under an ideal communication channel where the infinite call-source model is assumed. The detailed derivation of the throughput is shown in Appendix 6B.

6.8.2 Simulation Model

Subprogram `snpisma.m` is shown in Program 6.10. This program is fundamentally equal to `npcsma.m` shown in Program 6.8. The configuration of `snpisma` is also categorized into the seven blocks discussed in Section 6.5.2. The first four blocks and the seventh block are the same between `snpisma` and `npcsma`.

In the fifth block of `snpisma`, the access terminals, which generate packets or wait to transmit their packets at `now_time`, perform an inhibit sense. If the result is idle, `now_time` is updated until the next timing to transmit packets. If the result is busy, the time to perform the next inhibit sense is determined.

6.8.3 Simulation Results

The simulation results of throughput and average transmission delay are shown in Figures 6.25 and 6.26. In the simulation, the normalized propagation delay d was 0.02 and 0.2. The maximum throughput of the slotted nonpersistent ISMA was dependent on the normalized propagation delay. Moreover, if the capture effect is considered, a transmitted packet sometimes survived because the received powers for arrival packets were different from each other. Therefore, the throughput increased.

6.9 Conclusion

This chapter introduced some typical protocols for wireless communication and the methods to evaluate the throughput and average transmission delay by

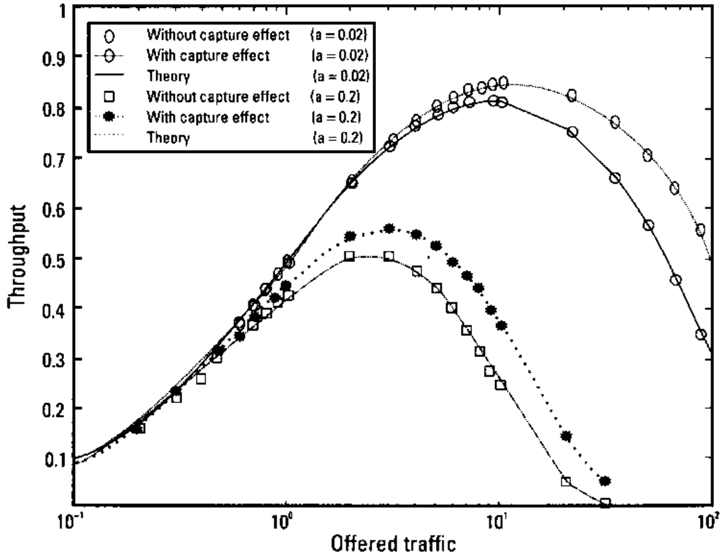


Figure 6.25 Offered traffic and throughput of slotted np-ISMA.

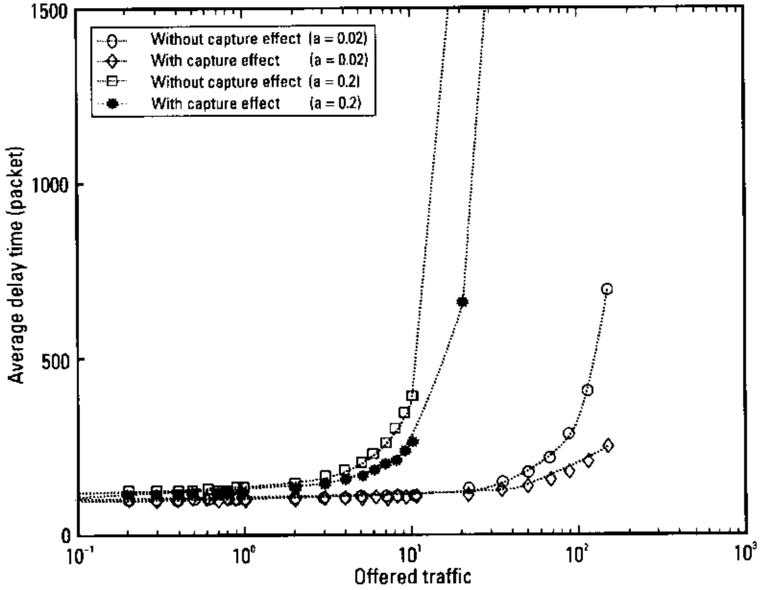


Figure 6.26 Offered traffic and average delay time of slotted np-ISMA.

computer simulation. Based on the discussed programs, readers can make their own new protocol, evaluate its performance, and compare the results with the conventional proposed protocols.

References

[1] Prasad, R., *CDMA for Wireless Personal Communication,* Norwood, MA, Artech House, 1996.

[2] Abramson, N., "The ALOHA System—Another Alternative for Computer Communications," *Proc. Fall Joint Computer Conference (AFIPS),* Vol. 37, 1970, pp. 281–285.

[3] Kleinrock, L., and F. A. Tobagi, "Packet Switching in Radio Channels, Part I-Carrier Sense Multiple Access Nodes and Their Throughput—Delay Characteristics," *IEEE Trans. Comm.,* Vol. 23, December 1975, pp. 1400–1416.

[4] Tobagi, F. A., and L. Kleinrock, "Packet Switching in Radio Channels: Part II—The Hidden Terminal Problem in Carrier Sense Multiple-Access and the Busy Tone Solution," *IEEE Trans. Comm.,* Vol. 23, December 1975, pp. 1417–1433.

[5] Krebs, J., and T. Freeburg, "Method and Apparatus for Communicating Variable Length Messages Between a Primary Station and Remote Stations at a Data Communication System," U.S. Patent No. 4519068, 1985.

[6] Zdunek, K. J., D. R. Ucci, and J. L. Locicero, "Throughput of Non-Persistent Inhibit Sense Multiple Access with Capture," *Electron. Lett.,* January 1989, pp. 30–32.

[7] Prasad, R., and J. C. Arnbak, "Capacity Analysis of Non-Persistent Inhibit Sense Multiple Access in Channels with Multipath Fading and Shadowing," *Proc. 1989 Workshop on Mobile and Cordless Telephone Communications, IEEE,* King's College, University of London, September 1989, pp. 129–134.

[8] Prasad, R., "Throughput Analysis of Non-Persistent Inhibit Sense Multiple Access in Multipath Fading and Shadowing Channels," *European Trans. on Telecommunications and Related Technologies,* Vol. 2, May–June 1991, pp. 313–317.

[9] Prasad, R., "Performance Analysis of Mobile Packet Radio Networks in Real Channels with Inhibit Sense Multiple Access," *IEE Proc.-I,* Vol. 138, No. 5, October 1991, pp. 458–464.

[10] Prasad, R., and C. Y. Liu, "Throughput Analysis of Some Mobile Packet Radio Protocols in Rician Fading Channels," *IEE Proc.-I,* Vol. 139, No. 3, June 1992, pp. 297–302.

[11] Widipangestu, I., A. J. 't Jong, and R. Prasad, "Capture Probability and Throughput Analysis of Slotted ALOHA and Unslotted np-ISMA in a Rician/Rayleigh Environment," *IEEE Trans. on Vehicular Technology,* Vol. 43, August 1994, pp. 457–465.

[12] Abramson, N., "The Throughput of Packet Broadcasting Channels," *IEEE Trans. Comm.,* Vol. 25, January 1977, pp. 117–128.

[13] Tanenbaum, A. S., *Computer Networks,* Englewood Cliffs, NJ: Prentice Hall, 1989.

[14] Rom, R., and M. Sidi, *Multiple Access Protocols Performance and Analysis,* New York: Springer-Verlag, 1990.

[15] Spragins, J. D., J. L. Hammond, and K. Pawlikowski, *Telecommunications Protocols and Design,* Reading, MA: Addison-Wesley, 1991.

[16] Sunshine, C. A., *Computer Network Architecture and Protocols,* New York: Plenum Press, 1989.

[17] Schwartz, M., *Computer-Communications Network Design and Analysis,* Englewood Cliffs, NJ: Prentice-Hall, 1977.

[18] Arnbak, J. C., and W. van Blitterswijk, "Capacity of Slotted ALOHA in Rayleigh-Fading Channels," *IEEE J. Selected Areas in Comm.,* Vol. SAC-5, No. 2, February 1987, pp. 261–269.

[19] Tsybakov, B. S., "Survey of USSR Contributions to Random Multiple Access Communications," *IEEE Trans. Inf. Theory,* Vol. IT-31, No. 2, March 1985, pp. 143–165.

[20] Nijhof, J. A. M., R. D. Vossenaar, and R. Prasad, "Stack Algorithm in Mobile Radio Channels," *Proc. IEEE 44th Vehicular Technology Conference,* Stockholm, Sweden, June 1994, pp. 1193–1197.

[21] Roberts, L. G., "Dynamic Allocation of Satellite Capacity Through Packet Reservation," *Proc. National Computer Conference (AFIPS),* Vol. 42, June 1973, pp. 711–716.

[22] Crowther, W., et al., "A System for Broadcast Communications: Reservation ALOHA," *Proc. Sixth Hawaii International Conference on System Sciences*, January 1973, pp. 371–374.

[23] Binder, R., et al., "A Dynamic Packet-Switching System for Satellite Broadcast Channels," *Proc. IEEE International Conference on Communications*, San Francisco, CA, June 1975, pp. 41.1–41.5.

[24] Goodman, D. J., and S. X. Wei, "Factors Affecting the Bandwidth Efficiency of Packet Reservation Multiple Access," *Proc. 39th IEEE Vehicular Technology Conference*, San Francisco, CA, May 1989, pp. 292–299.

[25] van den Broek, C., D. Sparreboom, and R. Prasad, "Performance Evaluation of Packet Reservation Multiple Access Protocol Using Steady-State Analysis of a Markov Chain Model," *Proc. COST 227/231 Workshop*, Limerick, Ireland, September 6–10, 1993, pp. 454–462.

[26] van den Broek, C., and R. Prasad, "Effect of Capture on the Performance of the PRMA Protocol in an Indoor Radio Environment with BPSK Modulation," *Proc. IEEE 44th Vehicular Technology Conference*, Stockholm, Sweden, June 1994, pp. 1223–1227.

[27] Prasad, R., and J. C. Arnbak, "Enhanced Throughput in Packet Radio Channels with Shadowing," *Electron. Lett.*, Vol. 24, August 1988, pp. 986–988.

[28] Kasahara, H., S. Hara, and N. Morinaga, "Modeling and Simulation Analysis of Indoor Packet Radio Communication Systems," *IEICE Trans. Fundamentals*, Vol. J78-A, No. 8, August 1995, pp. 947–956.

[29] Prasad, R., *Universal Wireless Personal Communications*, Norwood, MA: Artech House, 1996.

Appendix 6A

The probability in which a terminal generates n packets at time t is shown in (6A.1), if the number of the generation of the packet follows a Poisson distribution.

$$P_n(t) = \frac{e^{-\lambda t}(\lambda t)^n}{n!} \tag{6A.1}$$

and the expectation value of the generation interval becomes $\frac{1}{\lambda}$ because the generation interval becomes an exponential distribution. The probability that no packet is generated during the period from time 0 to t is given as follows:

$$P_0(t) = e^{-\lambda t} \tag{6A.2}$$

Therefore, the probability that the first packet generates after the time t is given as follows:

$$p(t) = 1 - e^{-\lambda t} \tag{6A.3}$$

where all the traffic, the number of the access terminals, the period to transmit a packet, and the expectation value of the packet generation interval are shown as G, M_{num}, T_{time}, and T_{int}, respectively, $t = T_{\text{time}}$ and $\lambda = \frac{1}{T_{\text{int}}}$ in (6A.3). The probability of $p(t)$ is given as follows:

$$p(t) = \frac{G}{M_{\text{num}}} = 1 - e^{\frac{T_{\text{time}}}{T_{\text{int}}}} \tag{6A.4}$$

From (6A.4), T_{int} is derivated as follows.

$$T_{\text{int}} = -\frac{T_{\text{time}}}{\log\left(1 - \frac{G}{M_{\text{num}}}\right)} \tag{6A.5}$$

In the simulation, the generation time of the packet is required. The time is calculated from (6A.6) on t.

$$x = 1 - e^{-\frac{t}{T_{\mathrm{int}}}} \tag{6A.6}$$

Finally, we can obtain the generation time of the packet.

$$t = -T_{\mathrm{int}} \times \log(1 - x) \tag{6A.7}$$

where the uniform random number of under 1 over 0 may be given in x.

Appendix 6B

This appendix derives the throughputs for pure ALOHA, slotted ALOHA, nonpersistent CSMA, and slotted nonpersistent CSMA theoretically.

6B.1 Pure ALOHA

Since the number of the generation of the packets is assumed to follow a Poisson distribution, the probability to generate n packets in the period t is given as (6B.1) when the expected number of packets to generate in a unit time is assumed as λ.

$$P_n(t) = \frac{e^{-\lambda t}(\lambda t)^n}{n!} \tag{6B.1}$$

When the period to transmit a packet is defined as τ, the traffic G is given as follows:

$$G = \lambda\tau \tag{6B.2}$$

To transmit a packet that is generated at a user terminal at the time t_1 successfully from the user terminal to access point means that the other terminals do not transmit any packets at the time from $t_1 - \tau$ to $t_1 + \tau$. In the pure ALOHA system, when a packet is generated at a terminal, the packet is transmitted to the access point immediately. Therefore, the probability of transmitting a packet generated at a user terminal to the access point successfully, P_{succ}, is equal to the probability not to generate any packet within the period of 2τ.

$$P_{\mathrm{succ}} = P_0(2\tau) = \frac{e^{-2\lambda\tau}(2\lambda\tau)^0}{0!} = e^{-2\lambda\tau} = e^{-2G} \tag{6B.3}$$

The throughput S is expressed as an expectation number of the packets that are transmitted to the access point successfully in unit time. Therefore, the value is given as follows:

$$S = GP_{\mathrm{succ}} = Ge^{-2G} \tag{6B.4}$$

6B.2 Slotted ALOHA

To transmit a packet generated from a user terminal to an access point successfully, a packet is transmitted in each time slot. Therefore, the probability of transmitting a packet generated at a user terminal to the access point successfully, P_{succ}, is equal to the probability to not generate any other packets in the period of τ where τ is equal to a time slot of the slotted ALOHA system.

$$P_{succ} = P_0(\tau) = \frac{e^{-\lambda\tau}(\lambda\tau)^0}{0!} = e^{-\lambda\tau} = e^{-G} \tag{6B.5}$$

Therefore, throughput, S, is shown as follows:

$$S = GP_{succ} = Ge^{-G} \tag{6B.6}$$

The above throughput is equal to the probability in which only one packet is generated in a time slot of τ.

$$P_1(\tau) = \frac{e^{-\lambda\tau}(\lambda\tau)^1}{1!} = \lambda\tau e^{-\lambda\tau} = Ge^{-G} = S \tag{6B.7}$$

6B.3 Nonpersistent CSMA

In the nonpersistent CSMA, a packet generated at a user terminal is transmitted to the access point after the carrier sensing becomes "idle." The CSMA timing is shown in Figure 6B.1. Here, when the expectation time length of a "busy" period is defined as B and the expectation time length of an "idle" period is defined as I, and the expectation time length in which no collision occurs and packets are successfully transmitted is defined as U, the throughput S is given by

$$S = \frac{U}{B+I} \tag{6B.8}$$

where the length of each packet is normalized and the transmission delays to transmit a packet are defined as 1 and a, respectively, Therefore, to transmit a packet generated at a user terminal at time t_1 to the access point successfully is equal to the probability not to generate any other packets in the period from t_1 to $t_1 + a$. Therefore, the expectation time length in which no collisions occur and packets are successfully transmitted, defined as U, is given by

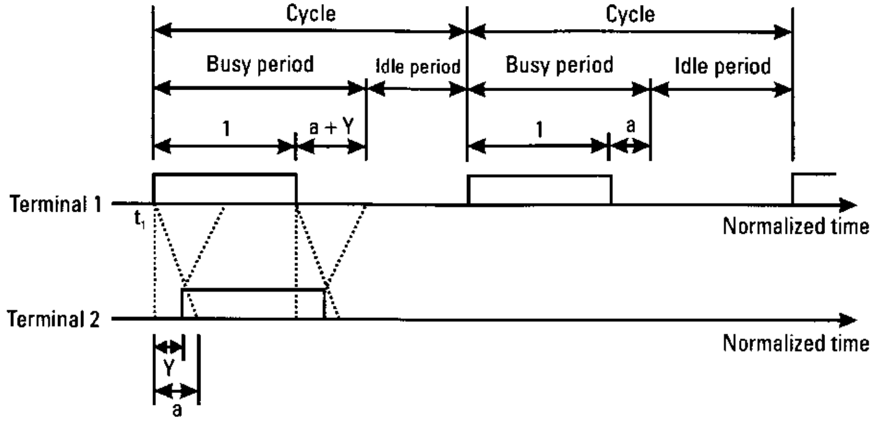


Figure 6B.1 Nonpersistent CSMA timing.

$$U = Ge^{-aG} \tag{6B.9}$$

The expectation time length of "idle" period is defined as I, in which no packet generates, follows the exponential distribution. The time length is given by

$$I = \frac{1}{G} \tag{6B.10}$$

When the transmitted time in which the last packet is transmitted in the period from t_1 to $t_1 + a$ is defined as $t_1 + Y$ and the expectation value of Y is defined as $\overline{Y}$, the expectation time length of the "busy" period defined as B is given by

$$B = 1 + a + \overline{Y} \tag{6B.11}$$

Since the distribution function of Y is given as follows

$$\begin{aligned} F_Y(y) &\cong \Pr\{Y \le y\} = \Pr\{\text{no arrival occurs in an interval of length } a - y\} \\ &= e^{-(a-y)G} \qquad (y \le a) \end{aligned} \tag{6B.12}$$

the expectation value of Y, $\overline{Y}$, is given by

$$\overline{Y} = a - \frac{1}{G}\left(1 - e^{-aG}\right) \tag{6B.13}$$

Putting all these together, we get the throughput

$$S = \frac{Ge^{-aG}}{G(1+2a)+e^{-aG}} \tag{6B.14}$$

6B.4 Slotted Nonpersistent ISMA

From Figure 6B.2, the length of an idle period is at least one time slot. When the idle period is only one slot long, it means that there is at least one arrival in the first slot of the idle period. For the period to be two slots long means that there were no arrivals at the first slot but that there was at least one arrival in its second slot. Continuing this reasoning and considering the Poisson scheduling process, we have

$$I = \frac{d}{1-e^{-dG}} \tag{6B.15}$$

A collision might occur if two or more packets arrive within the same slot and are scheduled for transmission in the next slot. A busy period will contain *k* transmission periods if there is at least one arrival in the slot of each of the first *k* – 1 transmission periods and no arrival in the last slot of the *k*th transmission period. Thus, the busy period is

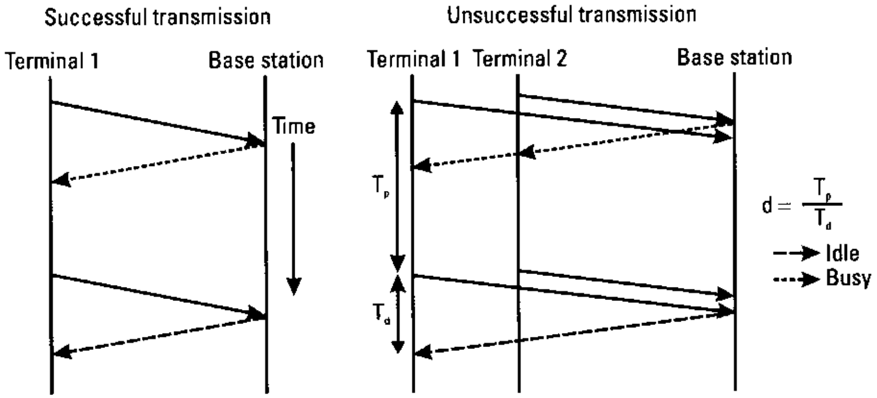


Figure 6B.2 Slotted nonpersistent ISMA packet timing.

$$B = \frac{1+d}{e^{-dG}} \tag{6B.16}$$

The expected useful time is found as

$$U = \frac{B}{1+d} P_{\text{succ}} \tag{6B.17}$$

where P_{succ} is the probability of a successful transmission period. We have

$$P_{\text{succ}} = \frac{\Pr[\text{single arrival within a slot}]}{\Pr[\text{more arrivals within a slot}]} = \frac{dGe^{-dG}}{1-e^{-dG}} \tag{6B.18}$$

Putting all these together, we get the throughput

$$S = \frac{U}{B+I} = \frac{dGe^{-dG}}{1+d-e^{-dG}} \tag{6B.19}$$

Appendix 6C

Program 6.1

```
% Program 6-1
% main.m
%
% Packet communication system
%
% MATLAB version
% Programmed by M. Okita
% Checked by H. Harada
%

clear;

                     % definition of the global variable
global STANDBY TRANSMIT COLLISION PERMIT
global Srate Plen Ttime Dtime
global Mnum Mplen Mstime Mstate
global Tint Rint
global Spnum Splen Tplen Wtime

STANDBY    = 0;     % definition of the fixed number
TRANSMIT   = 1;
COLLISION  = 2;
PERMIT     = 3;

                     % definition of the protocol
protocol = {'paloha'  ...    % Pure ALOHA      : pno = 1
            'saloha'  ...    % Slotted ALOHA   : pno = 2
            'npcsma'  ...    % np-CSMA         : pno = 3
            'snpisma' ...    % Slotted np-ISMA : pno = 4
      };

                     % definition of
                     % communication channel
brate = 512e3;       % bit rate
Srate = 256e3;       % symbol rate
Plen  = 128;         % length of a packet
Dtime = 0.01;        % normalized propagation delay
alfa  = 3;           % decline fixed number of
                     % propagation loss
sigma = 6;           % standard deviation of shadowing [dB]

                     % definition of the access point
r = 100;             % service area radius [m]
bxy = [0, 0, 5];     % position of the access point
                     % (x,y,z)[m]
tcn = 10;            % capture ratio [dB]

                     % definition of the access terminals
Mnum  = 100;         % number of the access terminal
```

```
mcn    = 30;           % C/N at the access point when
                       % transmitted from area edge

                          % simulation condition
pno = 1;                  % protocol number
capture = 0;              % capture effect    0:nothing
                          %                   1:consider
spend    = 10000;         % number of packets that
                          % simulation is finished
outfile = 'test.dat';  % result output file name

Ttime = Plen / Srate;  % transmission time of one packet
mpow  = 10^(mcn/10) * sqrt(r^2+bxy(3)^2)^alfa;
                          % true value of C/N

fid = fopen(outfile,'w');
fprintf(fid,'Protocol                            = %d\n',pno);
fprintf(fid,'Capture                             = %d\n',capture);
fprintf(fid,'Normalize_delay_time                = %f\n',Dtime);
fprintf(fid,'Bit_rate                 (bps)  = %d\n',brate);
fprintf(fid,'Symbol_rate              (sps)  = %d\n',Srate);
fprintf(fid,'Length_of_Packet(sbl)           = %d\n',Plen);
fprintf(fid,'Number_of_mobile_station        = %d\n',Mnum);
fprintf(fid,'Transmission_power (C/N)        = %f\n',mcn);
fprintf(fid,'Capture_ratio           (dB)    = %f\n',tcn);
fprintf(fid,'Number_of_Packet                = %d\n',spend);

fprintf('\n********* Simulation Start *********\n\n');
if capture == 0
      fprintf(' %s without capture effect\n\n',...
      char (protocol(pno)));
else
      fprintf(' %s with capture effect\n\n',...
      char (protocol(pno)));
end

mxy  = position(r,Mnum,0);          % positioning of the
                                    % access terminals
randn('state',sum(100*clock));      % resetting of the
                                    % random table
mrnd = randn(1,Mnum);               % decision of the
                                    % shadowing

for G=[0.1:0.1:1,1.2:0.2:2]         % offered traffic
      if G >= Mnum
         break
      end

      Tint  = -Ttime / log(1-G/Mnum);
         % expectation value of the packet generation
         % interval
      Rint  = Tint;
```

```
    % expectation value of the packet resending
    % interval
Spnum = 0;
Splen = 0;
Tplen = 0;
Wtime = 0;

now_time = feval(char(protocol(pno)),-1);
    % initialize of the access terminals

while 1
    next_time = feval(char(protocol(pno)),now_time);
    if Spnum >= spend
        break
    end
    idx = find(Mstate==TRANSMIT | Mstate==COLLISION);
    if capture == 0          % without capture effect
        if length(idx) > 1
            Mstate(idx) = COLLISION;
                         % collision occurs
        end
    else                     % with capture effect
        if length(idx) > 1
            dxy  = distance(bxy,mxy(idx,:),1);
                % calculation of the distance
            pow  = mpow * dxy.^-alfa .*...
                10.^(sigma/10*mrnd(idx));
                % calculation of received power
            [maxp no] = max(pow);
            if Mstate(idx(no)) == TRANSMIT
                if length(idx) == 1
                    cn = 10 * log10(maxp);
                else
                    cn = 10 * log10(maxp/(sum(pow)...
                        -maxp+1));
                end
                Mstate(idx) = COLLISION;
                if cn >= tcn
                    % received power larger than
                    % capture ratio
                    Mstate(idx(no)) = TRANSMIT;
                        % transmitting success
                end
            else
                Mstate(idx) = COLLISION;
            end
        end
    end
    now_time = next_time;
        % time is advanced until the next state
        % change time
end
```

```
      traffic = Tplen / Srate / now_time;
% calculation of the traffic
      ts = theorys(pno,traffic,Dtime);
% calculation of the theory value of the throughput

fprintf(fid,'%f\t%f\t%f\t%f\t%f\t%f\t%f\n',G,now_time ...
        ,Splen/Srate,Tplen/Srate,Wtime,Tint,Rint);
fprintf('G=%f\tS=%f\tTS=%f\n',traffic...
        ,Splen/Srate/now_time,ts);
end

fprintf('\n********** Simulation End **********\n\n');
fclose(fid);
graph(outfile);

%%%%%%%%%%%%%%%%%%%%%% end of file %%%%%%%%%%%%%%%%%%%%%%
```

Program 6.2

```
% Program 6-2
% position.m
%
% Positioning of the access terminals in the area of the
%  radius r.
%
% Input arguments
%  r  :  The radius r that an access point is an origin.
%  n  :  The number of access terminals.
%  h  : h=0 - z=0  h=1 -> z=1~4
%
% Output argument
%  posxy : (x,y,z)
%
% Programmed by M. Okita
% Checked by H. Harada
%

function [posxy] = position(r, n, h)

ms = 4 * r;            % calculation of the number of
                       % maximum position
ms = ms + 4 * sum(fix(sqrt(r^2-[1:r-1].^2)));

if n > ms
      error('n exceeds the number of position.');
end

posxy = zeros(n,3);    % initialize

for ii=1:n
```

```
  while 1
     xx = round(r*rand) * sign(sin(2*pi*rand));
        % x and y are decided at random
     yy  = round(r*rand) * sign(cos(2*pi*rand));
     if xx^2+yy^2 <= r^2 & (xx~=0 | yy~=0)
        % (xx,yy) is not (0,0) in the area
        if length(find(posxy(:,1)==xx & ...
           posxy(:,2)==yy)) == 0
           % (xx,yy) are vacant
           break
        end
     end
  end
  posxy(ii,[1 2]) = [xx yy];
  if h == 1
     while 1
        posxy(ii,3) = round(50*rand) / 10;
        if 1 <= posxy(ii,3) & posxy(ii,3) <= 4
           break
        end
     end
  end
end

%%%%%%%%%%%%%%%%%%%% end of file %%%%%%%%%%%%%%%%%%%%
```

Program 6.3

```
% Program 6-3
% distance.m
%
% The calculation of distance between access point and
%  access terminal.
%
% Input arguments
%  bstn   : coordinate of access point(x,y,z)
%  mstn   : coordinate of access terminals(x,y,z) (mstn is
%           vector or matrix)
%  scl    : scale (if scl is omitted, scl=1.)
%
% Output argument
% d       : distance
%
% Programmed by M. Okita
% Checked by H. Harada
%

function [d] = distance(bstn, mstn, scl)

if nargin < 3             % scl is omitted
  scl = 1;
end
```

```
[v,h] = size(mstn);
d = sqrt(sum(rot90(((repmat(bstn,v,1)-mstn)*scl).^2)));

%%%%%%%%%%%%%%%%%%%% end of file %%%%%%%%%%%%%%%%%%%%
```

Program 6.4

```
% Program 6-4
% theorys.m
%
% The calculation of the theory value of the throughput
%
% Input arguments
%  no  : protocol number    1 : Pure ALOHA
%                           2 : Slotted ALOHA
%                           3 : np-CSMA
%                           4 : Slotted np-ISMA
%  g   : offered traffic (scalar or vector)
%  a   : normalized propagation delay
%
% Output argument
%  ts  : the theory value of the throughput
%        (scalar or vector)
%
% Programmed by M. Okita
% Checked by H. Harada
%

function [ts] = theorys(no,g,a)

switch no
case 1             % Pure ALOHA
  ts = g .* exp(-2*g);
case 2             % Slotted ALOHA
  ts = g .* exp(-g);
case 3             % Non-Persistent Carrier Sense
                   % Multiple Access
     ts = g .* exp(-a*g) ./ (g*(1+2*a)+exp(-a*g));
case 4             % Slotted Non-Persistent ISMA
  ts = a * g .* exp(-a*g) ./ (1+a-exp(-a*g));
end

%%%%%%%%%%%%%%%%%%%% end of file %%%%%%%%%%%%%%%%%%%%
```

Program 6.5

```
% Program 6-5
% graph.m
%
% The function of drawing the graph of simulation
%  result.
```

```
%
% Input argument
%  filename : name of the file in which simulation result
%             was stored.
%
% Output argument
%  nothing
%
% Programmed by M. Okita
% Checked by H. Harada
%

function graph(filename)

mtitle1 = {'Throughput of Pure ALOHA system'  ...
  % definition of the title
      'Throughput of Slotted ALOHA system'     ...
      'Throughput of np CSMA system'           ...
      'Throughput of Slotted np ISMA system'   ...
      };
mtitle2 = {'Average Delay time of Pure ALOHA system' ...
  % definition of the title
      'Average Delay time of Slotted ALOHA system' ...
      'Average Delay time of np CSMA system'       ...
      'Average Delay time of Slotted ISMA system'  ...
      };

fid      = fopen(filename,'r');
nul      = fscanf(fid,'%s',2);
protocol = fscanf(fid,'%d',1);
nul      = fscanf(fid,'%s',2);
capture  = fscanf(fid,'%d',1);
nul      = fscanf(fid,'%s',2);
dtime    = fscanf(fid,'%f',1);
nul      = fscanf(fid,'%s',2);
brate    = fscanf(fid,'%d',1);
nul      = fscanf(fid,'%s',2);
srate    = fscanf(fid,'%d',1);
nul      = fscanf(fid,'%s',2);
plen     = fscanf(fid,'%d',1);
nul      = fscanf(fid,'%s',1);
mnum     = fscanf(fid,'%d',1);
nul      = fscanf(fid,'%s',2);
mcn      = fscanf(fid,'%f',1);
nul      = fscanf(fid,'%s',2);
tcn      = fscanf(fid,'%f',1);
nul      = fscanf(fid,'%s',2);
spend    = fscanf(fid,'%d',1);

idx = 0;
while 1
  data0 = fscanf(fid,'%f',1);
  data1 = fscanf(fid,'%f',1);
```

```
  data2 = fscanf(fid,'%f',1);
  data3 = fscanf(fid,'%f',1);
  data4 = fscanf(fid,'%f',1);
  data5 = fscanf(fid,'%f',1);
  data6 = fscanf(fid,'%f',1);

  if feof(fid) == 1              % eof
    break
  end

  idx = idx + 1;

  tg(idx)  = data0;              % offered traffic
  g(idx)   = data3 / data1;      % actual offered traffic
  s(idx)   = data2 / data1;      % actual throughput
  w(idx)   = data4 / spend * srate / plen;
                                 % average delay
end

fclose(fid);

ts = theorys(protocol,g,dtime);      % calculation of the
                                     % throughput

if protocol < 3                  % Pure ALOHA & Slotted ALOHA
  plot(g,s,'bo:',g,ts,'r-');       % normal graph
  legend('result','theory',0);     % legend
else                             % np-CSMA & Slotted np-ISMA
  semilogx(g,s,'bo:',g,ts,'r-');      % semi log graph
  if protocol == 3                    % legend
     legend(strcat('result (a=',num2str(dtime),')'),...
        'theory',0);
  else
     legend(strcat('result (d=',num2str(dtime),')'),...
        'theory',0);
  end
end
title(char(mtitle1(protocol)),'FontSize',16);
  % title
xlabel('Traffic(Simulation result)','FontSize',14);
  % x axis label
ylabel('Throughput','FontSize',14);
  % y axis label

figure(2);            % preparation of the new graph windows
                      % (line-feed)
if protocol < 3       % Pure ALOHA & Slotted ALOHA
  plot(g,w,'bo:');            % normal graph
  legend('result');           % legend
else                          % np-CSMA & Slotted np-ISMA
  semilogx(g,w,'bo:');        % semi log graph
  if protocol == 3            % legend
     legend(strcat('result (a=',num2str(dtime),')'),0);
  else
```

```
     legend(strcat('result (d=',num2str(dtime),')'),0);
  end
end
title(char(mtitle2(protocol)),'FontSize',16);
  % title
xlabel('Traffic(Simulation result)','FontSize',14);
  % x axis label
ylabel('Average Delay time(packet)','FontSize',14);
  % y axis label

%%%%%%%%%%%%%%%%%%%%%% end of file %%%%%%%%%%%%%%%%%%%%
```

Program 6.6

```
% Program 6-6
% paloha.m
%
% Pure ALOHA System
%
% Input argument
% now_time   : now time   but, now_time < 0 initializes
% the access terminals
%
% Output argument
% next_time : next state change time
%
% Programmed by M. Okita
% Checked by H. Harada
%

function [next_time] = paloha(now_time)

global STANDBY TRANSMIT COLLISION   % definition of the
                                    % global variable
global Srate Plen
global Mnum Mplen Mstate
global Tint Rint
global Spnum Splen Tplen Wtime

persistent mgtime mtime             % definition of the
                                    % static variable
if now_time < 0           % initialize access terminals
  rand('state',sum(100*clock));     % resetting of the
                                    % random table
  mgtime     = -Tint * log(1-rand(1,Mnum));
     % packet generation time
  mtime      = mgtime;    % packet transmitting time
  Mstate     = zeros(1,Mnum);
  Mplen(1:Mnum) = Plen;   % packet length
  next_time     = min(mtime);
  return
end
```

```
idx = find(mtime==now_time & Mstate==TRANSMIT);
  % finding of the terminal which transmission succeeded
if length(idx) > 0
  Spnum       = Spnum + 1;
  Splen       = Splen + Mplen(idx);
  Wtime       = Wtime + now_time - mgtime(idx);
  Mstate(idx) = STANDBY;
  mgtime(idx) = now_time - Tint * log(1-rand);
     % next packet generation time
  mtime(idx)  = mgtime(idx);
     % next packet transmitting time
end

idx = find(mtime==now_time & Mstate==COLLISION);
  % finding of the terminal which transmission failed
if length(idx) > 0
  Mstate(idx) = STANDBY;
  mtime(idx)  = now_time - Rint * ...
     log (1-rand(1,length(idx))); % resending time
end

idx = find(mtime==now_time);
  % finding of the terminal which transmission start
if length(idx) > 0
  Mstate(idx)   = TRANSMIT;
  mtime(idx)    = now_time + Mplen(idx) / Srate;
     % end time of transmitting
  Tplen         = Tplen + sum(Mplen(idx));
end

next_time = min(mtime);      % next state change time

%%%%%%%%%%%%%%%%%%%%% end of file %%%%%%%%%%%%%%%%%%%%%
```

Program 6.7

```
% Program 6-7
% saloha.m
%
% Slotted ALOHA System
%
% Input argument
% now_time : now time   but, now_time < 0 initializes the
%            access terminals
%
% Output argument
% next_time : next state change time
%
% Programmed by M. Okita
% Checked by H. Harada
%
```

```
function [next_time] = saloha(now_time)

global STANDBY TRANSMIT COLLISION
  % definition of the global variable
global Srate Plen
global Mnum Mplen Mstate
global Tint Rint
global Spnum Splen Tplen Wtime

persistent mgtime mtime slot
  % definition of the static variable

if now_time < 0         % initialize access terminal
  rand('state',sum(100*clock));     % resetting of the
                                    % random table
  slot       = Plen / Srate;        % slot length
  mgtime     = -Tint * log(1-rand(1,Mnum));
      % packet generation time
  mtime      = (fix(mgtime/slot)+1) * slot;
      % packet transmitting time
  Mstate     = zeros(1,Mnum);
  Mplen(1:Mnum) = Plen;      % packet length
  next_time     = min(mtime);
  return
end

idx = find(mtime==now_time & Mstate==TRANSMIT);
  % finding of the terminal which transmission succeeded
if length(idx) > 0
  Spnum          = Spnum + 1;
  Splen          = Splen + Mplen(idx);
  Wtime          = Wtime + now_time - mgtime(idx);
  Mstate(idx)    = STANDBY;
  mgtime(idx)    = now_time - Tint * log(1-rand);
      % next packet generation time
  mtime(idx)     = (fix(mgtime(idx)/slot)+1) * slot;
      % next packet transmitting time
end

idx = find(mtime==now_time & Mstate==COLLISION);
  % finding of the terminal which transmission failed
if length(idx) > 0
  Mstate(idx) = STANDBY;
  mtime(idx) = now_time - Rint * ...
      log (1-rand(1,length(idx)));     % waiting time
  mtime(idx)  = (fix(mtime(idx)/slot)+1) * slot;
      % resending time
end

idx = find(mtime==now_time);
  % finding of the terminal which transmission start
if length(idx) > 0
  Mstate(idx)    = TRANSMIT;
```

```
  mtime(idx)      = now_time + Mplen(idx) / Srate;
     % end time of transmitting
  mtime(idx)      = round(mtime(idx)/slot) * slot;
  Tplen           = Tplen + sum(Mplen(idx));
end

next_time = min(mtime);       % next state change time

%%%%%%%%%%%%%%%%%%%%%% end of file %%%%%%%%%%%%%%%%%%%%%%
```

Program 6.8

```
% Program 6-8
% npcsma.m
%
% non-persistent CSMA System
%
% Input argument

% now_time   : now time   but, now_time < 0 initializes the
%              access terminals
%
% Output argument
% next_time : next state change time
%
% Programmed by M. Okita
% Checked by H. Harada
%

function [next_time] = npcsma(now_time)

global STANDBY TRANSMIT COLLISION
  % definition of the global variable
global Srate Plen
global Mnum Mplen Mstime Mstate
global Tint Rint
global Spnum Splen Tplen Wtime

persistent mgtime mtime
  % definition of the static variable

if now_time < 0         % initialize access terminals
rand('state',sum(100*clock));     % resetting of the
                                  % random table
  mgtime        = -Tint * ...
      log (1-rand(1,Mnum));   % packet generation time
  Mstime        = zeros(1,Mnum) - inf;
      % packet transmitting time
  mtime         = mgtime;
      % state change time
  Mstate        = zeros(1,Mnum);
  Mplen(1:Mnum) = Plen;
  next_time     = min(mtime);
  return
end

idx = find(mtime==now_time & Mstate==TRANSMIT);
  % finding of the terminal which transmission succeeded
if length(idx) > 0
  Spnum       = Spnum + 1;
  Splen       = Splen + Mplen(idx);
  Wtime       = Wtime + now_time - mgtime(idx);
  Mstate(idx)  = STANDBY;
  mgtime(idx)  = now_time - Tint * log(1-rand);
      % next packet generation time
  mtime(idx)   = mgtime(idx);
      % next packet transmitting time
end

idx = find(mtime==now_time & Mstate==COLLISION);
  % finding of the terminal which transmission failed
if length(idx) > 0
  Mstate(idx)  = STANDBY;
  mtime(idx)   = now_time - Rint * ...
      log (1-rand(1,length(idx)));  % resending time
end

idx = find(mtime==now_time & Mstate==STANDBY);
  % finding of the terminal which carrier sensing
if length(idx) > 0
  Tplen = Tplen + sum(Mplen(idx));
  for ii=1:length(idx)
      jj = idx(ii);
      if carriersense(jj,now_time) == 0
         % channel is idle
         Mstate(jj) = TRANSMIT;
                % packet transmitting
         Mstime(jj) = now_time;
                % start time of transmitting
         mtime(jj)  = now_time + Mplen(jj) / Srate;
                % end time of transmitting
      else      % channel is busy
         mtime(jj) = now_time - Rint * log(1-rand);
                % waiting time
      end
  end
end

next_time = min(mtime);      % next state change time

%%%%%%%%%%%%%%%%%%%%%% end of file %%%%%%%%%%%%%%%%%%%%%%
```

Program 6.9

```
% Program 6-9
% carriersense.m
%
% The function of the carrier sense
%
% Input arguments
%  no          : terminal which carrier sensing
%  now_time    : now time
%
% Output argument
%  result      : 0: idle  1:busy
%
% Programmed by M. Okita
% Checked by H. Harada
%

function [result] = carriersense(no,now_time)

global Mnum Mstime Ttime Dtime
  % definition of the global variable

delay = Dtime * Ttime;
  % calculation of the delay time

idx = find((Mstime+delay) <= & now_time <= Mstime+delay+Ttime));
  % carrier sense
if length(idx) > 0             % carrier is detected
  result = 1;                  % channel is busy
else                           % carrier can't be detected
  result = 0;                  % channel is idle
end

%%%%%%%%%%%%%%%%%%%%%% end of file %%%%%%%%%%%%%%%%%%%%%%
```

Program 6.10

```
% Program 6-10
% snpisma.m
%
% Slotted non-persistent ISMA System
%
% Input argument
%  now_time : now time but, now_time < 0 initializes the
%             access terminals
%
% Output argument
%  next_time : next state change time
%
% Programmed by M. Okita
% Checked by H. Harada
%

function [next_time] = snpisma(now_time)

global STANDBY TRANSMIT COLLISION PERMIT
  % definition of the global variable
global Srate Plen Dtime
global Mnum Mplen Mstime Mstate
global Tint Rint
global Spnum Splen Tplen Wtime

persistent mgtime mtime slot
  % definition of the static variable

if now_time < 0             % initialize access terminals
  rand('state',sum(100*clock));
                          % resetting of the random table
  mgtime = -Tint * log(1-rand(1,Mnum));
                          % packet generation time
  Mstime = zeros(1,Mnum) - inf;
                          % packet transmitting time
  mtime  = mgtime;        % inhibit sensing time
  Mstate = zeros(1,Mnum);
  Mplen(1:Mnum) = Plen;   % packet length
  next_time = min(mtime); % state change time
  slot = Plen / Srate * Dtime;   % idle slot length
  return
end

idx = find(mtime==now_time & Mstate==TRANSMIT);
  % finding of the terminal which transmission succeeded
if length(idx) > 0
  Spnum       = Spnum + 1;
  Splen       = Splen + Mplen(idx);
  Wtime       = Wtime + now_time - mgtime(idx);
  Mstate(idx) = STANDBY;
  mgtime(idx) = now_time - Tint * ...
      log (1-rand);% next packet generation time
  mtime(idx)  = mgtime(idx);
end

idx = find(mtime==now_time & Mstate==COLLISION);
  % finding of the terminal which transmission failed
if length(idx) > 0
  Mstate(idx) = STANDBY;
  mtime(idx)  = now_time - Rint * ...
      log (1-rand(1,length(idx)));  % resending time
end

idx = find(mtime==now_time & Mstate==STANDBY);
  % finding of the terminal which inhibit sensing
if length(idx) > 0
```

```
  Tplen = Tplen + sum(Mplen(idx));
  for ii=1:length(idx)
      jj = idx(ii);
      if inhibitsense(jj,now_time) == 0
            % channel is idle
         Mstate(jj) = PERMIT;
         [temp1 kk] = max(Mstime);
            % calculation of the transmitting start
            % time slot
         if temp1 < 0
            mtime(jj) = ceil(now_time/slot) * slot;
         else
            temp1 = temp1 + Mplen(kk) / Srate;
            temp2 = now_time - temp1;
            if temp2 < 0
               mtime(jj) = temp1 + slot;
            else
               mtime(jj) = temp1 + ceil(temp2/slot)...
                   * slot;
            end
         end
      else                               % channel is busy
         mtime(jj) = now_time - Rint * log(1-rand);
                                    % waiting time
      end
  end
end

idx = find(mtime==now_time & Mstate==PERMIT);
  % finding the terminal which transmission start

if length(idx) > 0
  Mstate(idx) = TRANSMIT;
  Mstime(idx) = now_time;
  mtime(idx)  = now_time + Mplen(idx) / Srate;
      % end time of transmitting
end

next_time = min(mtime);      % next state change time

%%%%%%%%%%%%%%%%%%%%%% end of file %%%%%%%%%%%%%%%%%%%%%
```

Program 6.11

```
% Program 6-11
% inhibitsense.m
%
% The function of the inhibit sense
%
% Input arguments
% no        : terminal which inhibit sensing
% now_time  : now time
```

```
%
% Output argument
% inhibit    : 0: idle   1: busy
%
% Programmed by M. Okita
% Checked by H. Harada
%

function [inhibit] = inhibitsense(no,now_time)

global Mnum Mstime Ttime Dtime
   % definition of the global variable

delay   = Dtime * Ttime;  % calculation of delay time
inhibit = 0;

idx = find((Mstime+delay) <= now_time ...
   & now_time <= (Mstime+delay+Ttime));% inhibit sensing
if length(idx) > 0
   idx = find(idx~=no);             % except itself
   if length(idx) > 0               % inhibit is detected
       inhibit = 1;                 % channel is busy
   end
end

%%%%%%%%%%%%%%%%%%%%% end of file %%%%%%%%%%%%%%%%%%%%%
```

7

Cellular Telecommunications Systems

7.1 Introduction

In Chapters 1–6, the performance of point-to-point and point-to-multipoint transmissions was evaluated by using several transmission schemes. However, in a mobile communication system, a base station communicates with many users. This means point-to-point transmission is just one contributing factor in a mobile communication system. Thus, we need to evaluate not only the performance of point-to-point transmission, but also the per- formance of multipoint access schemes—or how performance is affected by several independent users. Particularly in a mobile communication system, the cellular telecommunication model is frequently used as the basic access scheme, because it can obtain high utilization efficiency of frequencies. This chapter will discuss a cellular telecommunication system model that is actually in use and achieving high system capacity [1, 2]. It is necessary to use computer simulations to evaluate for total system performance, such as system capacity, and blocking probability or forced termination probability of a call.

One advantage of using an access scheme model based on a cellular system is that it can be combined with some of the transmission schemes presented in previous chapters, enabling advanced evaluation. Indeed, it is often the case that a cellular system simulation is conducted by combining other complex factors of the communication process such as transmission power control and antenna array techniques, making primitive theoretical analysis of the system performance nearly impossible.

We therefore introduce the fundamental model of cellular system simulations. Several techniques used in our simulation are essential to real evaluations, and at the same time are easily applicable to the above-mentioned probable situations.

The basic concept of a cellular telecommunication system and intercell interference is explained in Section 7.2. Then, in Section 7.3, some key techniques to construct the computer simulation model for a cellular system using specific software are introduced. Moreover, a technique to achieve the DCA algorithm, a well-known channel assignment scheme, is presented there. We present our simulation results using that algorithm in Section 7.4. Section 7.5 describes the model for a cellular system using an array antenna. Section 7.6 concludes the chapter. All programs related to the chapter are presented in Appendix 7A and in the CD-ROM accompanying the book.

7.2 Concept of Cellular System and Channel Assignment Algorithm

The basic concept of a cellular system is shown in Figure 7.1. In this system, the entire service area is divided into several small areas called cells. A base station located in the center of each cell allocates several traffic channels for mobile terminals within the cell when they initiate their calls. One of the most common features of this cellular system is that the same traffic channel can be concurrently allocated in different cells if they are sufficiently spaced apart. If the same channels are allocated in closer cells, each cell causes cochannel interference to the other cells and degrades the transmission quality. However, if the cells with the same channels are adequately spaced from the others, that interference is diminished by the distance between the cells; transmission quality is never corrupted by such interference, and the radio resources for the entire system can be utilized effectively.

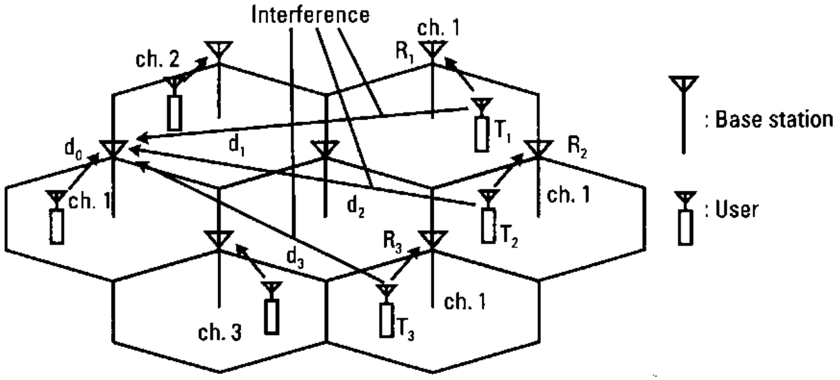


Figure 7.1 Cellular system concept.

In Figure 7.1, the four base stations are R_0, R_1, R_2, and R_3, and the four users are T_0, T_1, T_2, and T_3. Here, it is assumed that user T_i is connected to base station R_i, and that all of four users are allocated the same traffic channels. On this assumption, we can derive the carrier-to-noise plus interference ratio (C/(N + I)) R_{cni} for user T_0 as follows,

$$R_{cni} = \frac{AP_0 d_0^{-\alpha} 10^{\frac{\xi_0}{10}}}{N + \sum_{i-1}^{3} AP_i d_i^{-\alpha} 10^{\frac{\xi_i}{10}}} \tag{7.1}$$

Here, α is a path loss factor, A is a proportional coefficient, P_i is the transmitted power of user T_i, d_i is the distance between user T_i and base station R_0. In addition, ξ_i is the distortion caused by shadowing between T_i and R_0, the value of which is denoted by decibels in (7.1). From (7.1), we can see that the longer distances of d_1, d_2, d_3 result in a higher C/(N + I) ratio, which is equivalent to a higher quality of transmission. Therefore, to achieve an adequate C/(N + I) ratio for preliminary service quality, several channel assignment algorithms are being used in the current systems.

Two major examples of such channel assignment algorithms are FCA and DCA [3]. FCA is an algorithm that conducts a fixed assignment of channels per base station. An example of FCA is shown in Figure 7.2, where four channels are allocated to each base station so that users in a cell suffer from minimum

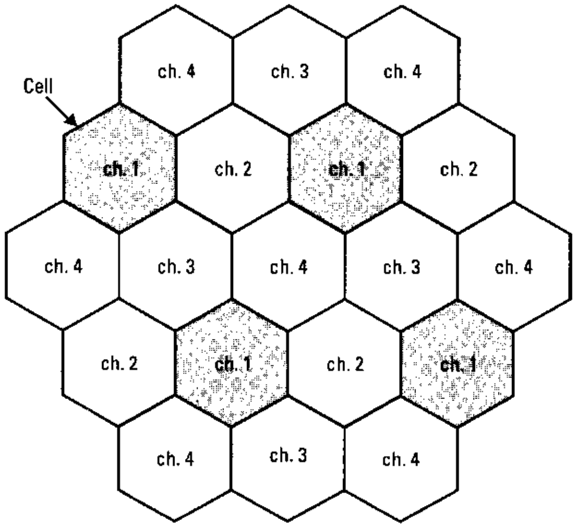


Figure 7.2 FCA algorithm concept.

cochannel interference from other cells. On the other hand, in the case of DCA, a base station can handle all the channels for the system and dynamically allocates suitable channels to minimize interference conditions. A concept of DCA is shown in Figure 7.3. The DCA algorithm can achieve higher capacity and cope more flexibly with fluctuations of traffic than FCA can, because all the channels are adaptively allocated according to users' demands. In the case of FCA, as shown in Figure 7.3(a), the second user in a cell faces call blocking, because the channels per base station are fixedly restricted. On the other hand, in case of DCA shown in Figure 7.3(b), the base stations all share the channels, which are adaptively allocated, thereby decreasing incidents of call blocking that occur with FCA. However, DCA needs a more complicated control scheme than FCA does, such as interference measurement and channel searching.

7.3 Simulation Program for Dynamic Channel Assignment System

This section presents a simulation program to realize a cellular system using the DCA algorithm. The main program is shown as Program 7.1, and subprograms used by the main program are shown as Programs 7.2–7.7. Table 7.1 summarizes the functions of each program.

7.3.1 Performance Measures

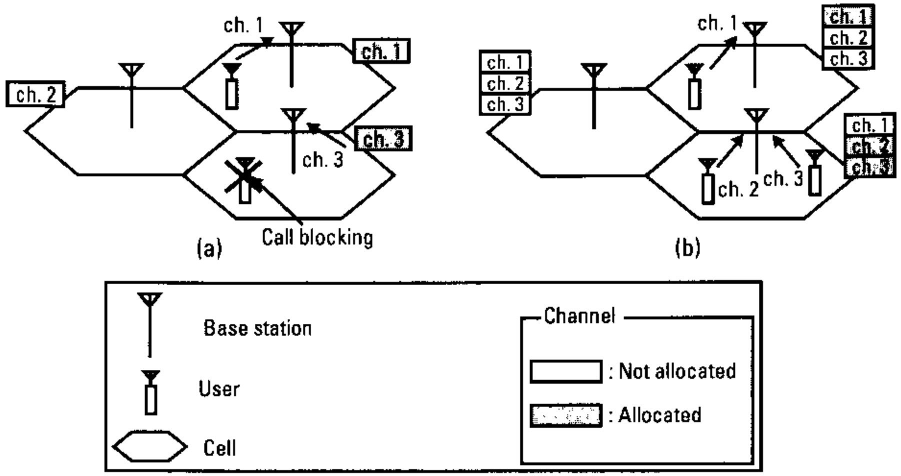


Figure 7.3 DCA algorithm concept: (a) FCA and (b) DCA.

Table 7.1
Functions of Programs 7.1–7.7

Program	Name in the Simulation	Function
Program 7.1	dcamain.m	Main program
Program 7.2	basest.m	Determine location of each base station
Program 7.3	wrap.m	Determine relationship of base station positions introducing cell-wrapping
Program 7.4	cellmesh.m	Distribute a cell into meshes and store the data about each mesh coordinate
Program 7.5	holdtime.m	Generates call holding time, which is exponentially distributed
Program 7.6	shadow.m	Generates an attenuation due to shadowing, which has log-normal distribution
Program 7.7	dist.m	Generates a path loss due to distance

Before giving a detailed explanation about our simulation and programs, we will describe the performance measures in our simulation. We are focusing on two measures, blocking probability and forced termination probability. The blocking probability is defined as the statistical probability that a new call will fail to find suitable channels that satisfy the C/(N + I) ratio condition mentioned in Section 7.2. Although the blocking probability is the measure pertaining to new calls, a connected call can be interrupted before it finishes due to rapid degradation of the C/(N + I) ratio condition. Thus, we define forced termination probability as the statistical probability that a connected call will be interrupted before its conclusion. If we further define `callnum`, `blocknum`, and `forcenum` as the number of generated calls, blocked calls, and forced terminated calls, respectively, we truly introduce the variables of the same names in the simulation as shown in following sections. Blocking probability P_{bl} and forced termination probability P_{fo} are given as follows:

$$P_{bl} = \frac{blocknum}{callnum} \tag{7.2}$$

$$P_{fo} = \frac{forcenum}{callnum - blocknum} \tag{7.3}$$

Introducing those two performance measures enables several potential evaluations of cellular systems.

7.3.2 Flowchart of Entire Simulation

Figure 7.4 shows a flowchart of the entire simulation. Our simulation consists of three parts: the preparation part, the main loop part, and the output part. In the preparation part, several pieces of information needed for the simulation are introduced, such as the cell layout or traffic parameters, before the main loop is started.

The main loop of our simulation is activated by the `while` loop with preliminary finish time "`timeend`." Throughout the entire loop, the status of present users is checked and, if necessary, renewed in a short time interval, the period "`timestep`." Several status indicators are successively stored in matrix "`userinfo`." Table 7.2 shows the contents of matrix "`userinfo`."

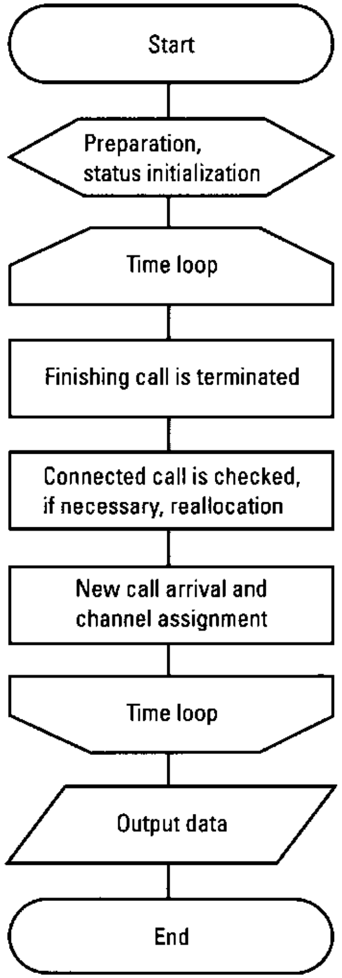

Start
Preparation, status initialization
Time loop
Finishing call is terminated
Connected call is checked, if necessary, reallocation
New call arrival and channel assignment
Time loop
Output data
End

Figure 7.4 Simulation flowchart.

In every time period in the loop, each user causes several events, such as call initiation, channel searching, channel allocation, channel reallocation, and call termination based on the status matrix. In a time period, the following events are considered in turn.

1. Calls of connected users are terminated if they finish in this period.
2. Calls of still-connected users are examined. If a desirable interference condition is not satisfied, reallocation is attempted by searching for new channels.
3. With a preliminary probability, users that are not connected start new calls and search for channels that satisfy the interference conditions.
4. Return to (1).

Several types of numerical data are also measured in every period.

Finally, in the output part, measured and accumulated data in the main loop are organized into output, in the form of output matrix "`check`," or "`check2`," or "`output`," or text data "`data.txt`."

7.3.3 Cell Layout and Cell-Wrapping Technique Used

Figure 7.5 shows the cell layout employed in our simulation. We used 19 hexagonal cells having a cell radius of 1. Such cells are determined by the position of the 19 base stations. The base station positions are determined by Program 7.2, "`basest.m`," and this information is stored in the 19 × 2 matrix "`baseinfo`." For example, "`baseinfo(5, 1)`" and "`baseinfo(5, 2)`" respectively reveal x and y coordinates of the fifth base stations. In our simulation,

Table 7.2
Information Stored in "`userinfo`"

Column	Stored Information
userinfo(:, :, 1)	X-coordinates of the user
userinfo(:, :, 2)	Y-coordinates of the user
userinfo(:, :, 3)	Path loss of the user (if connected)
userinfo(:, :, 4)	Usage; 0:not connected, 1:connected
userinfo(:, :, 5)	Call termination time
userinfo(:, :, 6)	Allocated channel number (if connected)

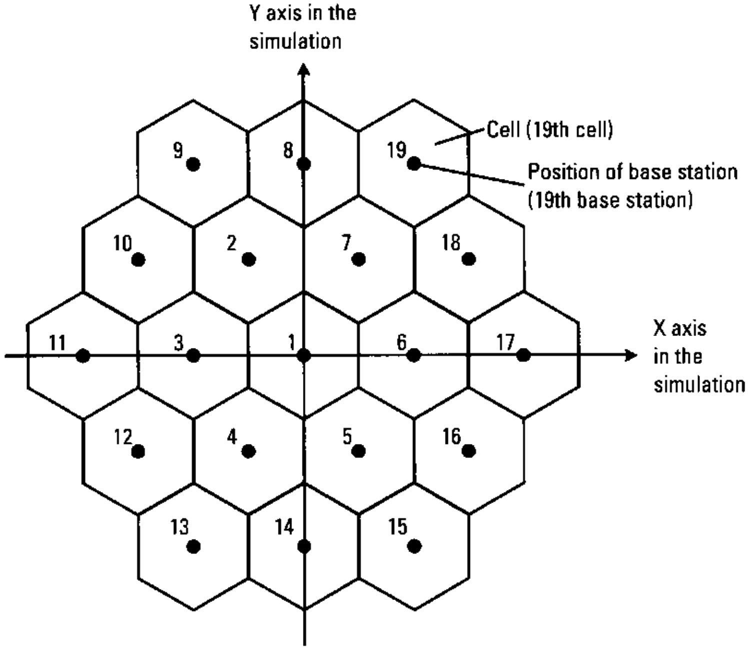


Figure 7.5 Cell layout.

regulated numbers of users are scattered in each of the 19 cells from which data are taken.

In the case of a cellular system using the DCA algorithm, we should take into account not only one sample cell, but also neighboring cells, because cochannel interference from neighboring cells has a significant effect on the performance of the sample cell. For example, the 5th cell is subject to interference from the 1st, 4th, 14th, 15th, 16th, and 6th cells. Cells located farther away, such as the 2nd or 3rd ones, could interfere with the 5th cell. However, it is assumed that such interference is decreased enough by the distance that it can be ignored, and we take into account only the six immediate neighboring cells in the simulation. On the other hand, in the case of the 9th cell, which is located on the boundary of the cell layout, it has only three neighboring cells, the 10th, 2nd, and 8th. Such a "boundary cell" has different performance than an "inner-located cell," for example the 9th cell, which would show better performance than the 2nd cell, because fewer cells cause interference in the 9th cell. Consequently, taking user activity in the boundary cells as well as that in an inner cell into account does not adequately evaluate DCA performance.

Thus, to avoid such a problem, two solutions can be used. One is to take data only from inner cells such as the 1st, 2nd, 3rd, 4th, 5th, 6th, and 7th cells in Figure 7.5, and exclude boundary cells. These inner cells are all subject to interference from six neighboring cells and are expected to reveal effective performances of the DCA algorithm. However, because boundary cells do not contribute to output data, we need a larger number of cells to construct the entire cell layout to obtain well-averaged data, making the simulation burden heavy.

The other solution is to use a cell-wrapping technique. Figure 7.6 shows a concept of this technique. In this technique, boundary cells are regarded as neighbors of the boundary cells located almost directly opposite the cell layout. In Figure 7.6, only the 19 shaded cells are cells that really exist, and the other cells are copies of the real cells having the same number. As a result, the 9th cell suffers from interference not only from the 10th, 2nd, and 8th cells, but also copies of the 13th, 17th, and 14th cells in the neighboring positions. On this assumption, every cell in the cell layout can be regarded as being an "inner-located cell" having six neighbors.

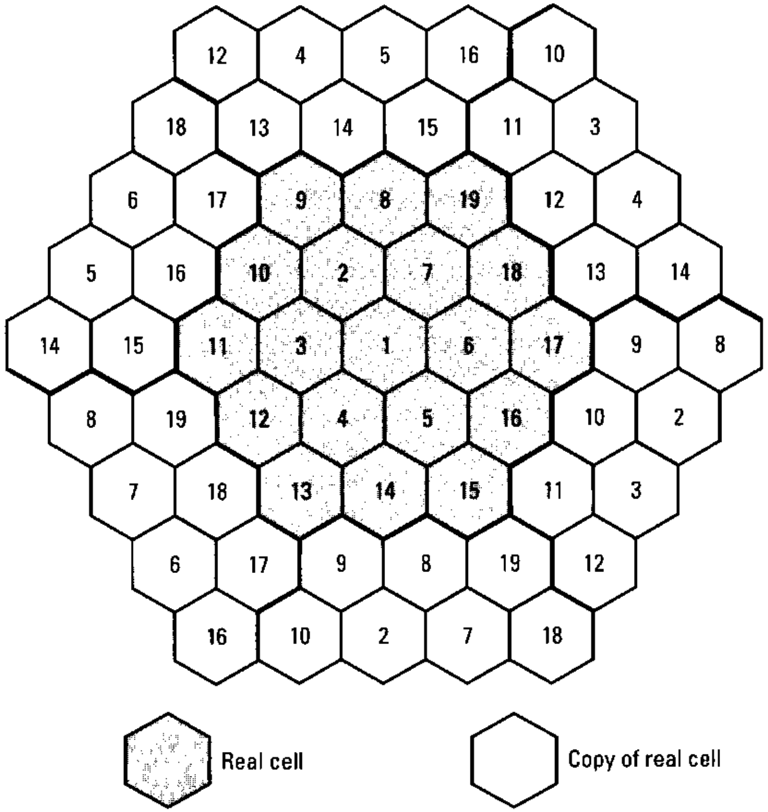


Figure 7.6 Cell-wrapping technique concept.

To realize cell wrapping, a 19 × 19 matrix `wrapinfo` is generated in the program `wrap.m`, and introduced in the main program. The `wrapinfo` matrix reveals the relationship among the 19 cells, including cell wrapping. Thus, for example, `wrapinfo(5, :)` stores information for fifth cell about other cells, especially `wrapinfo(5, 2)` to `wrapinfo(5, 7)`, which are the numbers of its six neighbors. (See Program 7.3 with Figure 7.6.) The concrete usage of this matrix will be explained later.

7.3.4 Cell Mesh Construction

In our simulation, user distribution is considered to be uniform over one cell as well as over the entire cell layout. Such a condition is realized by cells distributed into a lot of small meshes. Figure 7.7 shows a cell and meshes used in our simulation. Program `cellmesh.m` plays the role of a mesh generator in our simulation and outputs the number of meshes in a cell as a variable `cellnum` and outputs the `cellnum` × 2 matrix `meshposition`, which stores the position of each mesh. We can also see a result of `cellmesh.m` in the command line of MATLAB as follows:

```
>>[meshnum meshposition] = cellmesh;
>>plot (meshposition(:, 1),meshposition(:, 2),'.').
```

Figure 7.8 shows the results, where "`Fineness`" is a parameter for mesh fineness and is usually set to "`Fineness = 50`" in our simulation. Here, discrete meshes are used to allocate a certain amount of traffic into the specially shaped area such as a hexagonal cell. If we introduce a larger value of "`Fineness`," such a discrete mesh structure gets close to the continuous distribution model, as expected in Figure 7.8.

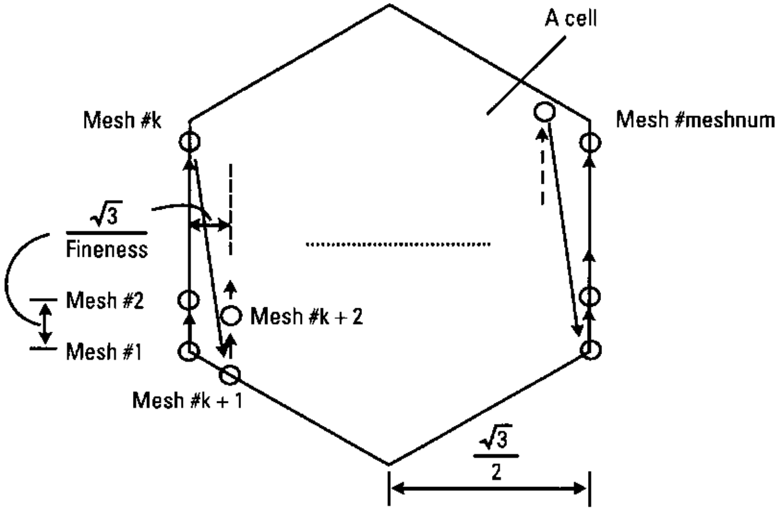


Figure 7.7 Meshes constructed in cell.

In the simulation, when a user initiates a call, we generate a random integer "`mesh`" whose value uniformly fluctuates from 1 to "`meshnum`" and locates a user on a location of "`meshposition(mesh, :)`" in a cell. As a result, each user is scattered in a certain point of "`meshnum`" points in a cell with equal probability at its call initiation, and we can obtain uniform user distribution over a cell.

7.3.5 Traffic Parameters of a Call

Each call generation is subject to the Poisson process with its mean arrival rate for each user of "lambda" (calls/hour). To realize such an arrival rate, we examine each user that is not connected in every time period "`timestep`" using a random function "`rand`." If a value of "`rand`" is not more than "`lambda * timestep / 3600`," which presents an average arrival rate during "`timestep`," then the user is regarded as having started a call, and the number of generated calls is counted as a variable "`callnum`."

On the other hand, every initiated call has its own call holding time, and the initiated call is terminated after such a time. The holding time of each call is subject to exponential distribution with a mean value of "`ht`" (second). In our simulation, we obtain the value of the holding time as the output of the function "`holdtime(ht)`," which outputs a random value subject to exponential distribution with an average value "`ht`." The "`holdtime(ht)`" is provided by program "`holdtime.m`," and Figure 7.9 shows a *probability distribution function* (pdf) of the "`holdtime(ht)`" output, which is measured by MATLAB.

Moreover, the number of users is also a significant traffic parameter. We can set up the number of users existing in a cell and investigate the effects of such fluctuations of the user numbers. In the simulation results as shown later, we evaluate the performance of DCA mainly according to such numbers of users per cell, which is revealed by the variable "`user`."

7.3.6 Propagation Conditions

The strength of the received signal, regardless of the desired wave, or interference wave, is one of the most important issues in our simulation. The transmitted signal suffers from attenuation caused by such factors as distance and obstruction. In the simulation, we introduce path loss and shadowing as such attenuation factors.

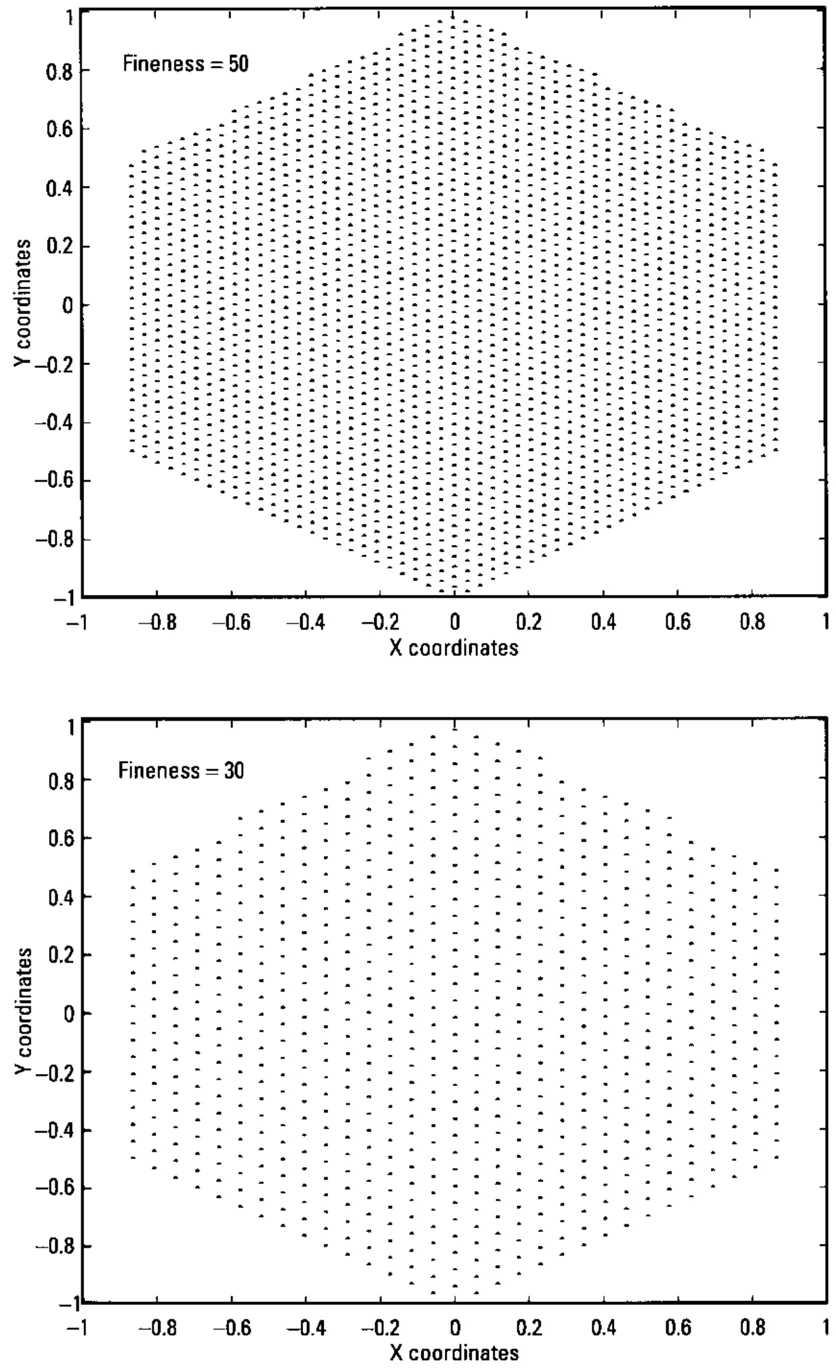


Figure 7.8 Mesh pattern examples.

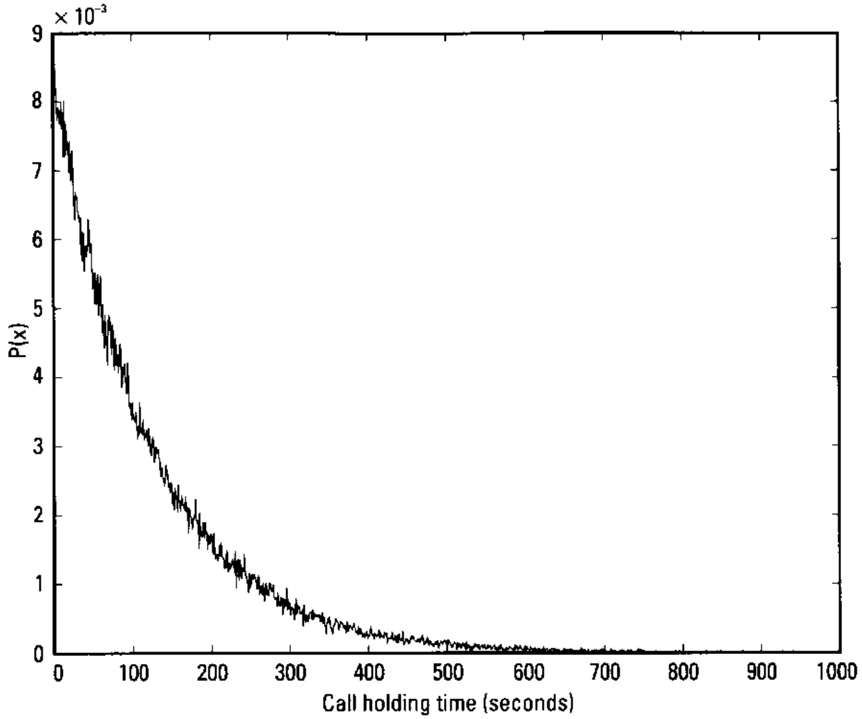


Figure 7.9 Call holding time pdf.

We assume the transmitted signal is subject to path loss with a decay factor of α as in (7.1), which is provided as variable "`alpha`" and set "`alpha = 3.5`" in the simulation. Calculating this path loss is conducted using the program "`dist.m`." This program provides a function "`dist(a, b, alpha)`," where "`a`" and "`b`" are a 1 × 2 matrix and respectively denote coordinates of two points. Then, value of "`dist(a, b, alpha)`" reveals the path loss between these two points, a and b. We describe a more practical usage of this function later.

Moreover, shadowing is assumed to be subject to log-normal distribution with a standard deviation of "`sigma`," corresponding to ξ_i in (7.1). This "`sigma`" is set to "`sigma = 6.5`" in the simulation. We also obtain the value of shadowing as the output of the function "`shadow(sigma)`," which outputs a random value subject to log-normal distribution with its standard deviation "`sigma`." The symbol "`shadow(sigma)`" is provided by program "`shadow.m`," and Figure 7.10 shows a pdf of the "`shadow(sigma)`" output, which is measured by MATLAB.

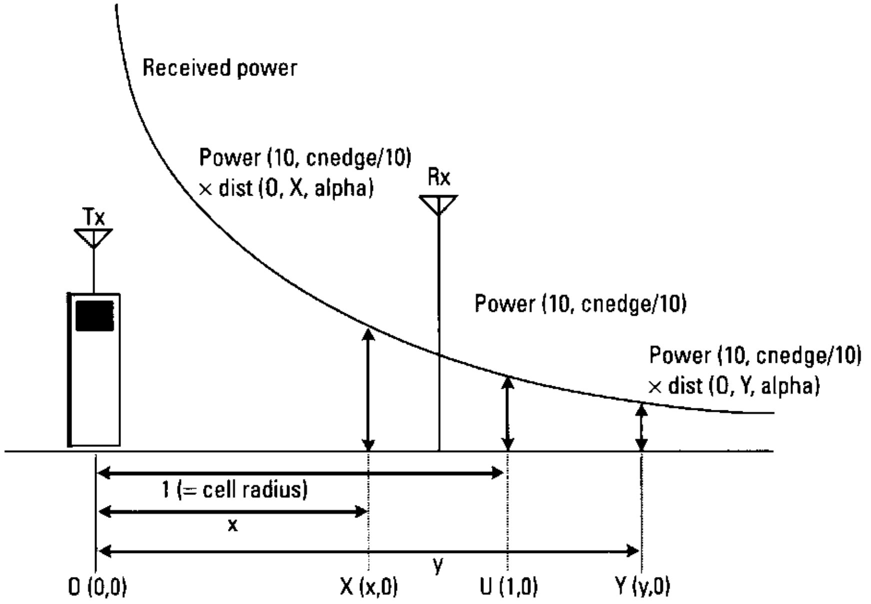


Figure 7.10 Path loss conditions.

7.3.7 Channel Assignment

The channel assignment process in the simulation is explained in this section. The routine is activated in the main program, "dcamain.m."

First of all, uplink power strength from a call-initiating user to the base station is examined. This power is provided by the C/N. We present this C/N using the simulation parameter "cnedge" (decibels), which reveals the C/N when the distance between the transmitter and receiver is 1. Figure 7.11 shows these conditions, where only path loss is considered.

In the program, we first input the coordinates of the user's position and the base station, obtain the path loss value as "dist(here, there, alpha)," then add shadowing attenuation generated by "shadow(sigma)" to that value. After that we obtain the attenuation value of the desired signal, which we call "dwave" in the program. Using the "dwave" value, we can obtain the C/N of the signal received by the base station. This C/N corresponds to "power(10.0, cnedge/10.0) * dwave," and is represented by the "cn" variable.

After calculating "cn," we search for an available channel that is not in use, and that satisfies the interference conditions of the C/(N + I). The channel search is conducted in the loop of channel number "ch," so that the provided number ("chnum") of channels is examined in its entirety. If another user is allocated the same channel "ch," we calculate the interference from that user. In a loop of "for around = 2:7," the variable "othercell = wrapinfo(numcell, around)" means the cell that is equivalent to the neighboring cell of the "numcell"th cell. We also calculate the attenuation value of the interference signal from this neighboring cell, the sum of which is called "uwave," in the same manner as "dwave." However, please note that one of the two coordinates matrix, in this case, the matrix "there" is provided somewhat differently, as follows:

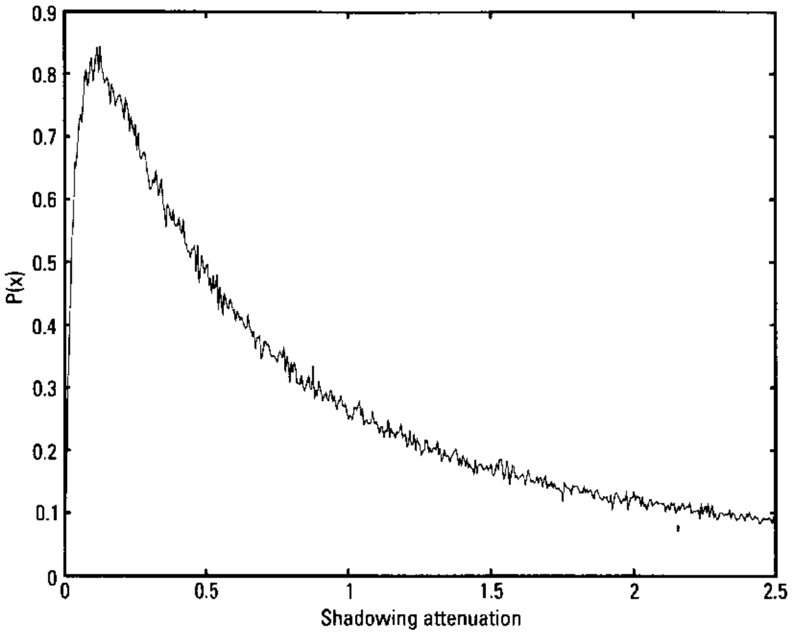


Figure 7.11 The pdf of shadowing attenuation.

```
>> userposi(1, 1:2) = userinfo(othercell, other, 1:2);
>> here = baseinfo(numcell, :);
>> there = userposi - baseinfo(othercell, :) ...
           + baseinfo(around, :) + baseinfo(numcell, :);
```

compared with "dwave" case below:

```
>> userposi(1, 1:2) = userinfo(numcell, numuser, 1:2);
>> here = baseinfo(numcell, :);
>> there = userposi;
```

In the case of "uwave," there are three additive terms. These three terms play the role of compensating for a miscalculation that could be caused by cell wrapping.

Figure 7.12 shows the compensation by using two such terms. In the figure, interference from a user in the 15th cell to the 19th base station in 19th cell is evaluated. In this case the, the values are "numcell = 19," "othercell = 15," and "around = 2." (See wrap.m.) If we use the matrix "there" with the three terms as follows,

```
>> userposi(1, 1:2) = userinfo(15, other, 1:2);
>> here = baseinfo(19, :);
>> there = userposi - baseinfo(15, :) + baseinfo(2, :) ...
       + baseinfo(19, :);
```

we can obtain a suitable calculation of path loss "dist(here, there, alpha)" including the cell wrapping, shown as in Figure 7.12(a). However, if we ignore the three terms as follows,

```
>> userposi(1, 1:2) = userinfo(15, other, 1:2);
>> here = baseinfo(19, :);
>> there = userposi;
```

we obtain an incorrect value of path loss shown as in Figure 7.12(b). We calculate interference from every user that can cause interference and finally add them up as the total interference "dwave."

Now, as a goal, we have to obtain the C/(N + I) ratio R_{cni}, which can be described as follows:

$$R_{cni} = \frac{C}{N+I} = \frac{1}{N/C + I/C} = \frac{1}{(\mathrm{C/N})^{-1} + (\mathrm{C/I})^{-1}} \quad (7.4)$$

From (7.2), we can see that the C/(N + I) is revealed by using the C/N and C/I. We have already provided the value of the C/N as "power(10.0, cnedge/10.0) * dwave," and the value of the C/I corresponds to "dwave/ uwave." By using such values, we can calculate the value of the C/(N + I) ratio "cnirdb" (decibels) in the simulation.

Finally, we examine if the achieved value of the C/(N + I) can satisfy the interference condition, that is, if it is greater than the C/(N + I) threshold "cnirth." If the achieved C/(N + I) exceeds the threshold, a call is accepted, and a channel is allocated to the user. In this case, the function "holdtime()" is activated and the output is also allocated to the user as its call holding time,

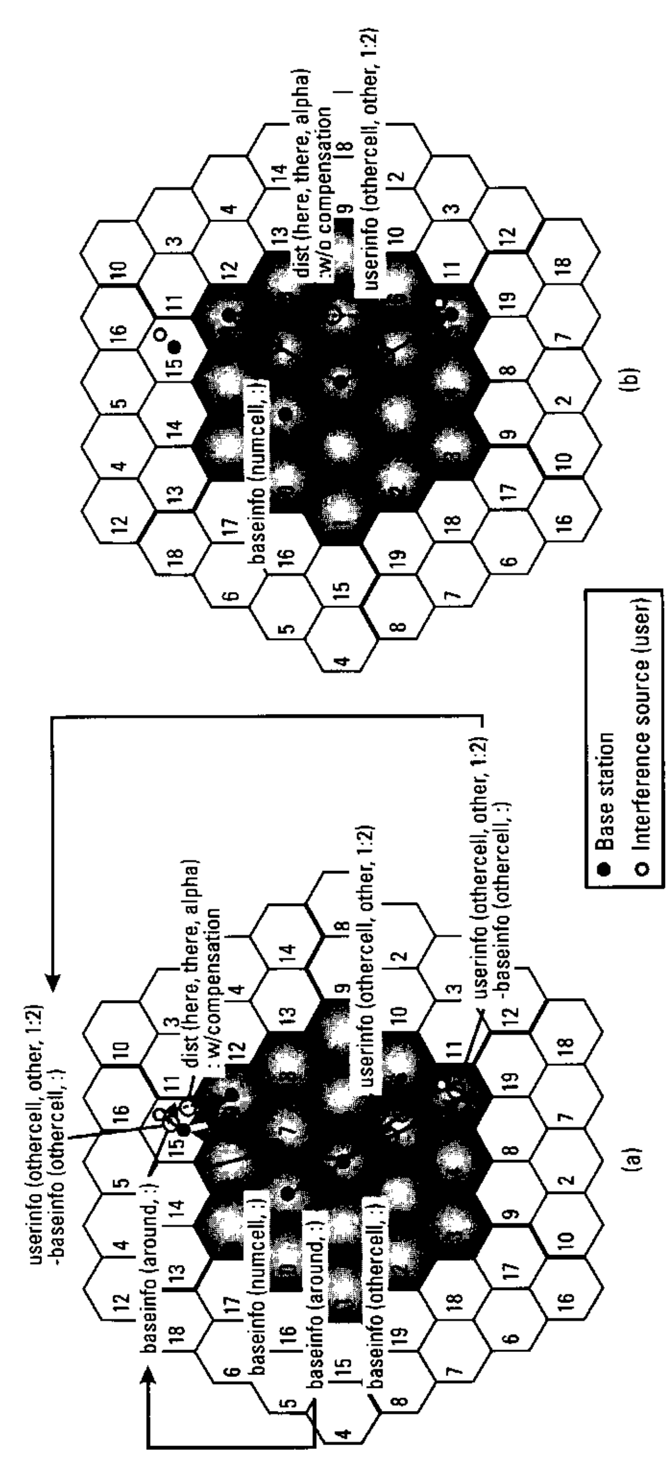

userinfo (othercell, other, 1:2)
-baseinfo (othercell, :)
dist (here, there, alpha)
: w/compensation
userinfo (othercell, other, 1:2)
userinfo (othercell, other, 1:2)
-baseinfo (othercell, :)
baseinfo (around, :)
baseinfo (numcell, :)
baseinfo (around, :)
baseinfo (othercell, :)
(a)
dist (here, there, alpha)
:w/o compensation
userinfo (othercell, other, 1:2)
baseinfo (numcell, :)
(b)
● Base station
○ Interference source (user)

Figure 7.12 Compensation for cell-wrapping: (a) with compensation and (b) without compensation.

which provides the time the call is finished. If the interference condition is not satisfied, we regard such a call as being blocked, and the number of blocked calls is counted as the variable "blocknum."

7.3.8 Channel Check and Reallocation

We also check the interference conditions for connected users in every time period. This routine is conducted in a similar way to channel assignment, and the C/(N + I) of allocated channel for each connected user is examined. If the C/(N + I) of the channel is not satisfied, the user releases a currently allocated channel and tries to find an alternative channel in just the same way as channel assignment. At this point, if a channel condition is not satisfied, such an event is not regarded as blocking but as the forced termination of a call. Therefore, in that case, the number of forced terminations of calls is counted as the variable "forcenum."

7.3.9 Output

Finally, after the entire main loop part is executed, the accumulated data are calculated and stored in the output matrix "output." We have already counted the essential numerical values such as "callnum," "blocknum," and "forcenum," thereby simply calculating the previously defined blocking probabilityP_{bl} and forced termination probability P_{fo} as in the manner of (7.2) and (7.3).

Moreover, in Program 7.1, for convenience, we provide five simulations using different parameters for "usernum." Measured data in every "usernum" is all stored in matrix "output," which is also written as a text file "data.txt." Contents of the matrix "output" are shown in Table 7.3.

We also introduce other output matrices "check" and "check2" in the simulation. In every time period we store the current value of blocking probability and forced termination probability in "check" and "check2," respectively. Consequently, "check" and "check2" give transitional data in every time period, so we can determine suitable loop lengths for our simulation.

Table 7.3
Contents of "output"

Column	Contents
output(1, :)	The number of calls
output(2, :)	The number of blocked calls
output(3, :)	Blocking probability
output(4, :)	Forced termination probability

7.4 Simulation Results of Cellular System with DCA Algorithm

This section will evaluate the suitable performance of a cellular system using DCA.

7.4.1 Simulation Results

Using the same output as in previous sections, we can evaluate the results of our simulation. For example, we can see the blocking probability performance according to the number of users in the following manner:

```
>> semilogy (usernum, output(3, :))
```

Figure 7.13 shows the blocking probability versus the number of users. In the figures, "cnedge" [Figure 7.13(a)] and "chnum" [Figure 7.13(b)] are evaluated as parameters. We can see that a higher received power and a greater number of channels result in a lower blocking probability.

Figure 7.14 shows that the transitional blocking probability obtained by "check," which means the length of the simulation (we use "timeend = 5000"), is adequate for the evaluation.

7.4.2 Theoretical Analysis

This section evaluates the theoretical result for blocking probability. For simplicity, we make the assumption that the received power of each user is sufficiently high that the interference from other users can be ignored. By this assumption, we can denote the blocking probability by using Engset's loss formula [4] as follows:

$$P_{bl-theo} = \frac{\binom{n-1}{s}(vh)^s}{\sum_{i=0}^{s}\binom{n-1}{i}(vh)^i} \tag{7.5}$$

Here, n and s are the number of users and channels, respectively, and v and h are the average call arrival rates per nonconnected user and the average call holding time, which respectively correspond to "lambda" and "ht" in our simulation.

Figure 7.15 shows the comparison between the simulation results and theoretical value. We confirmed that our simulation results fit the theoretical value.

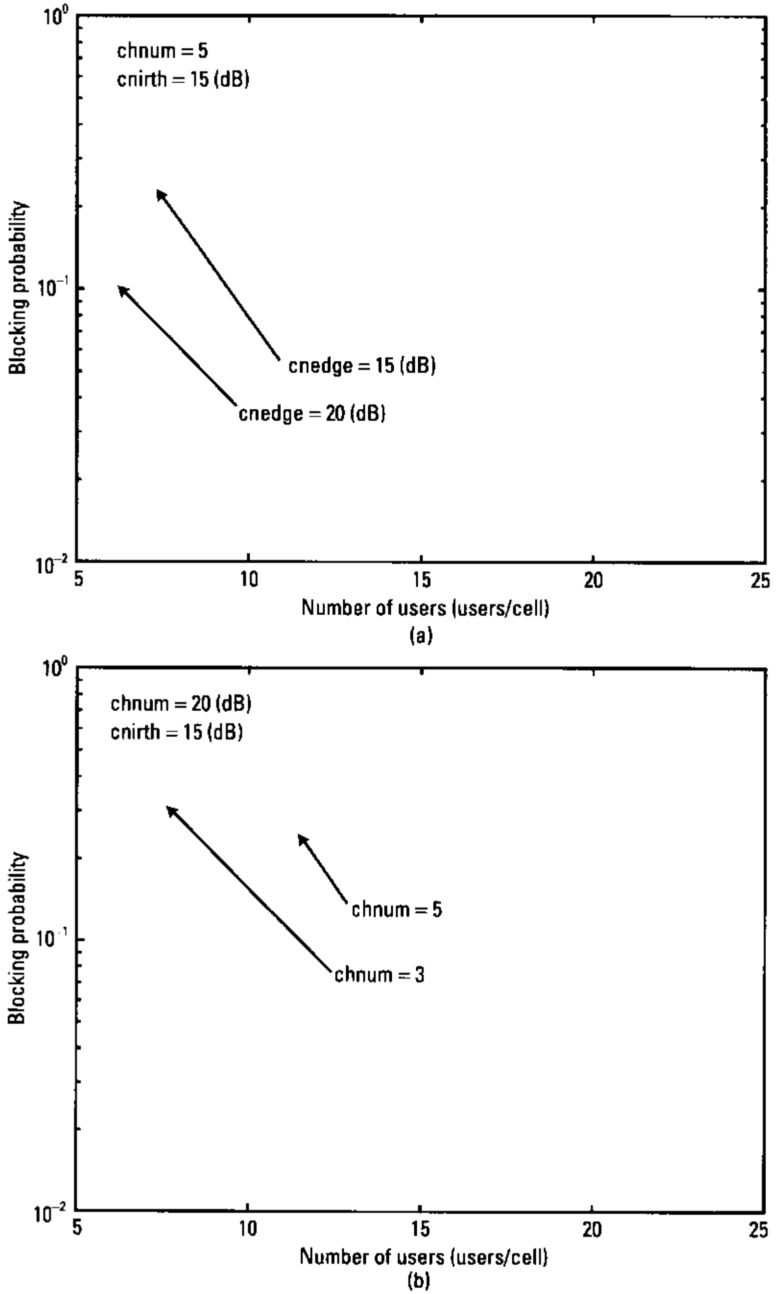


Figure 7.13 Blocking probability versus number of users: (a) parameter: `cnedge`, and (b) parameter: `chnum`.

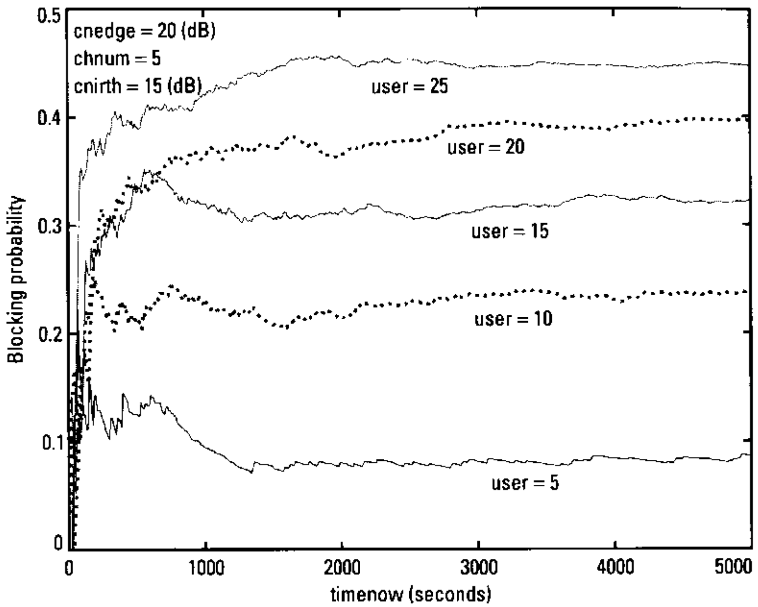


Figure 7.14 Transitional performances of blocking probability.

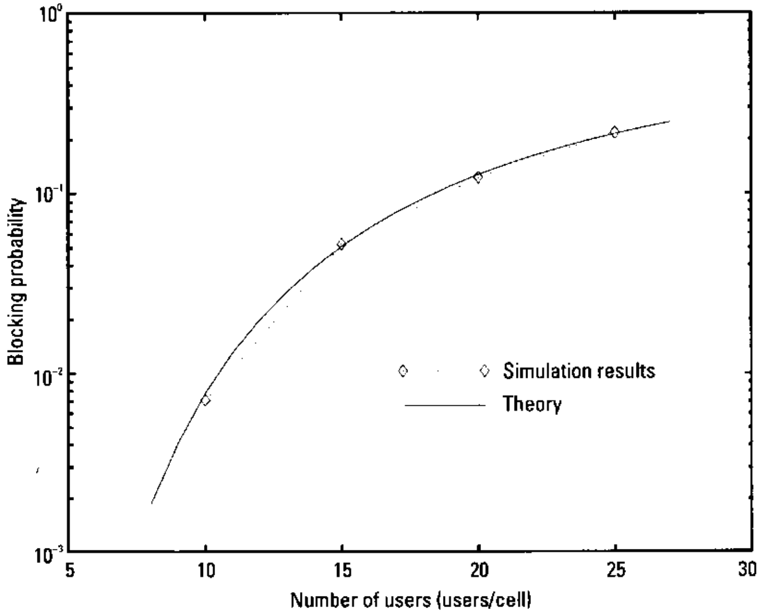


Figure 7.15 Theoretical analysis.

7.5 Beamforming Technique Using Array Antennas for Cellular Systems

The beamforming technique is widely used in mobile communication systems to optimize the use of frequency resources in the space domain [5]. By implementing this technique at the base station, the base station can change the shape of zones dynamically according to the locations of the mobile stations [6]. The cochannel interference and the reception of excess multipath fading from undesired directions are reduced at the base station, because it can issue high antenna gain in the desired direction and low antenna gain in the undesired one. Similarly, from the viewpoint of the mobile station, interference signals from other base stations are reduced. This is because base stations can avoid transmitting signals in undesired directions. As a result, the quality of service improves at both the base station and the desired mobile station. System capacity also increases because channel-reuse distances can be shortened. Naturally, using the beamforming technique at the mobile station is necessary for further improvement. The ratio of the desired signal (carrier) power to total cochannel interference signal power (CIR) are the key parameters that determine the capacity in mobile communication systems. Because the beamforming technique can greatly suppress interference, it is a powerful tool for improving system capacity.

Beamforming control provides several benefits to all conventional access schemes for wireless communication systems. These systems include FDMA, *time division multiple access* (TDMA), and CDMA. Moreover, *space division multiple access* (SDMA) has been proposed and studied as a scheme for increasing channel capacity within a limited frequency resource. In SDMA, a base station can allocate individual beams for each mobile station, and the mobile stations communicate with the base station by using the same frequency within a cell.

There are several measures for evaluating the effect of beamforming control, and these depend on the access scheme. This section focuses only on the CIR because it is a common parameter for all schemes. Here, we disregard thermal noise, because the CIR is far more influential in mobile communication systems.

For simplicity, this section explains the evaluation of the uplink improvement at the base station. Figure 7.16 shows a base station using the beamforming technique. In Figure 7.16, there are three cells using the same frequency, with the frequency reuse distance *D*. This distance is set to satisfy the required *quality of service* in the service model. Three base stations (B_0, B_1, and B_2) and three mobile stations (M_0, M_1, and M_2) are shown. Here, we assume that M_0, M_1, and M_2 transmit signals to B_0, B_1, and B_2, respectively, using the same frequency. When we notice the B_0, the ratio of the power of the desired signal from M_0 to the power of interference signals from M_1 and M_2 is given as follows:

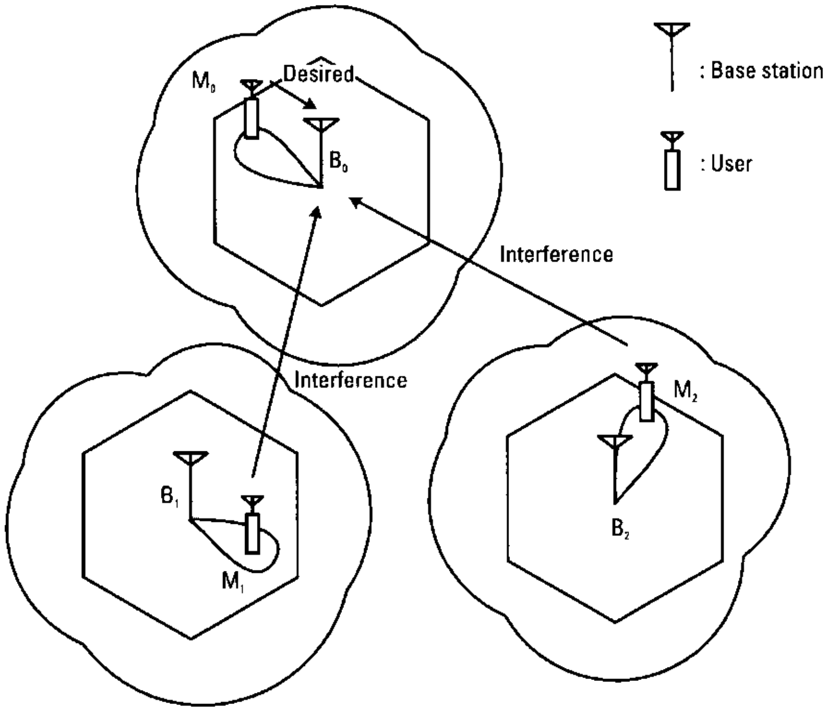


Figure 7.16 Use of beamforming technique at base stations in cellular systems.

$$R_{CI} = \frac{AP_0 d_0{}^{-\alpha} P 10^{\frac{\xi_0}{10}} \times 10^{\frac{G_{HB_0}(\theta_{HM_0})}{10}} \times 10^{\frac{G_{VB_0}(\theta_{VM_0})}{10}} \times 10^{\frac{G_{HM_0}(\theta_{HB_0})}{10}} \times 10^{\frac{G_{VM_0}(\theta_{VB_0})}{10}}}{\sum_{i=1}^{2} AP_i d_i^{-\alpha} 10^{\frac{\xi_i}{10}} \times 10^{\frac{G_{HB_0}(\theta_{HM_i})}{10}} \times 10^{\frac{G_{VB_0}(\theta_{VM_i})}{10}} \times 10^{\frac{G_{HM_i}(\theta_{HB_0})}{10}} \times 10^{\frac{G_{VM_i}(\theta_{VB_0})}{10}}} \tag{7.6}$$

where α is the path loss factor, A is a proportional coefficient, P_j is the transmitted power at M_j, d_j is the distance between M_j and B_0, and ξ_j is the distortion due to shadowing between M_j and B_0 ($j = 0, 1, 2$). The symbol $G_{HB_0}(\theta_{HM_j})$ represents the horizontal antenna gain of B_0 toward θ_{HM_j}, where θ_{HM_j} is the direction of M_j from B_0 in the horizontal direction. The symbol $G_{VB_0}(\theta_{VM_j})$ represents the vertical antenna gain of B_0 toward θ_{VM_j}, where θ_{VM_j} is the vertical direction between M_j and B_0. In the same way, $G_{HM_j}(\theta_{HB_0})$ represents the horizontal antenna gain of M_j toward θ_{HB_0}, where θ_{HB_0} is the horizontal direction between B_0 and M_j, the symbol $G_{VM_j}(\theta_{VB_0})$ represents the vertical antenna gain of M_j toward θ_{VB_0}, where θ_{VB_0} is the vertical direction between B_0 and M_j. The relationship between θ_{HM_j} and θ_{VM_j} is shown in Figure 7.17. The terms θ_{HB_0} and θ_{VB_0} correspond to $(360-\theta_{HM_j})$ and $(90-\theta_{VM_j})$, respectively.

7.5.1 Simulation Programs for Evaluating CIR Improvement

This section uses a simulation program to evaluate CIR improvement in a mobile cellular system using the beamforming technique and an array antenna. The main program is shown as Program 7.8, and subprograms used in the main program are shown in Programs 7.9–7.11. Table 7.4 summarizes the function of each program. Figure 7.18(a) shows a simulation model. It is assumed that there are 19 cells using the same frequency. The distances between all base stations are defined by the size of the clusters. Figure 7.18(b) and (c) shows the relationship between the cell arrangement and the size of the clusters. Here the centered cell is numbered 0 and the surrounding cells are numbered 1 through 18.

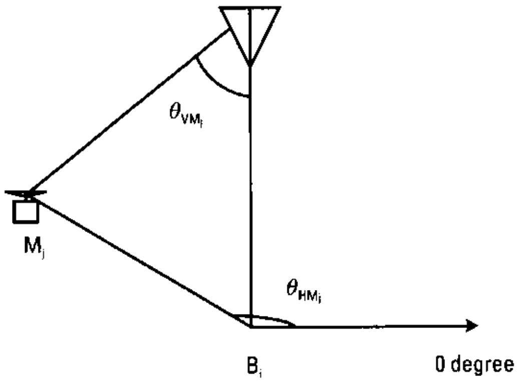


Figure 7.17 The relationship between a base station and a mobile station.

Table 7.4
Function of Programs 7.8–7.11

Program	Name in Simulation	Function
Program 7.8	main.m	Main program
Program 7.9	set_D.m	Determines distance between base stations
Program 7.10	stationinit.m	Determines position of base station
Program 7.11	antgain.m	Determines characteristics of the antenna gain
Program 7.6	shadow.m	Generates an attenuation due to shadowing that has log-normal distribution

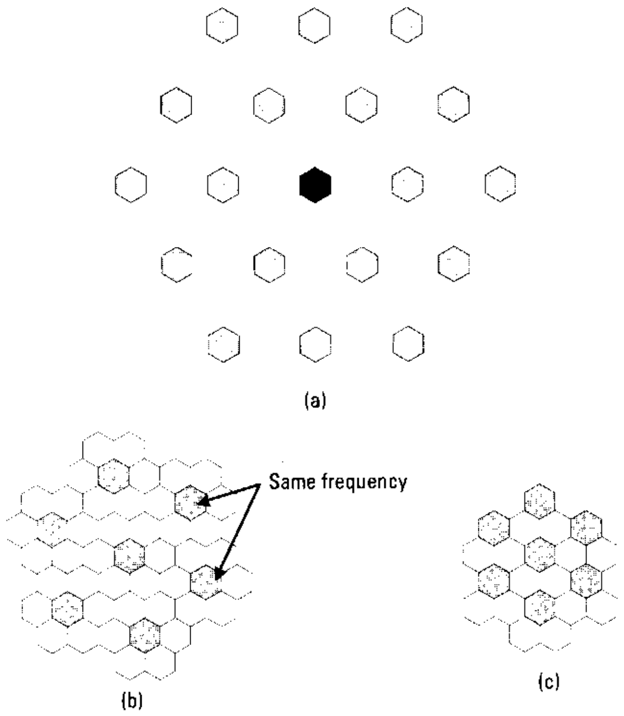


Figure 7.18 The arrangement of cells in simulation: (a) the arrangement of 19 cells using the same frequency, (b) the cell arrangement when the cluster size is 7, and (c) the cell arrangement when the cluster size is 3.

The program calculates the CIR for an uplink at a central base station and outputs the difference in the CIR statistics for two cases:

- Case 1: beamforming control is applied.
- Case 2: beamforming control is not applied.

The simulation process is as follows:

- Define the positions of the base stations.
- Define the position of the mobile stations in each cell i ($i = 0 \ldots 18$).

- Calculate the following:
    - The distance and angle between the mobile station in the *i*th cell and the target *i*th base station;
    - The distance and angle between the mobile station in the *i*th cell and the centered base station;
    - The antenna gain for the direction of the mobile station in the *i*th cell at the central base station;
    - The antenna gain for the direction of the centered base station at the mobile station in the *i*th cell;
    - The effect of shadowing.
- CIR calculation for cases 1 and 2.

Figure 7.19 shows a simulation flowchart. The following is an outline of the main program.

```
[Status initialization]
```

First, the cluster size is determined by setting parameters *I* and *J*. Here, cluster size is given by the following equation:

$$I^2 + J^2 + I \cdot J \tag{7.7}$$

To illustrate, when $I = 1$ and $J = 1$, the cluster size is 3. Next, based on the cluster size, we determine the arrangement of the base stations. We also decide on the height of the base station and the radius of the cell. If we set the height of the base station to 0, we can approximate the macrocell model.

Second, we determine the characteristics of the antenna gain at both the base station (BS) and the mobile station (MS). Here we define BS of the *i*th cell as B_i. We also define MS in the *i*th cell as M_i. In this simulation, we assume $G_{HB_i}(\phi) = G_{HM_i}(\phi) = G_{VB_i}(\phi) = G_{VM_i}(\phi) = 0$ $(i = 0 \ldots 18)$ for all direction ϕ in case 2. On the other hand, we define specific values for these antenna gain parameters in case 1. We assume that the characteristics of antenna gain can be approximated such that,

$$G_{HB_i}(\phi) = \begin{cases} 10\log_{10}\cos^n(\phi), & -\pi/2 \le \phi \le \pi/2 \\ x, & \text{otherwise} \end{cases} \tag{7.8}$$

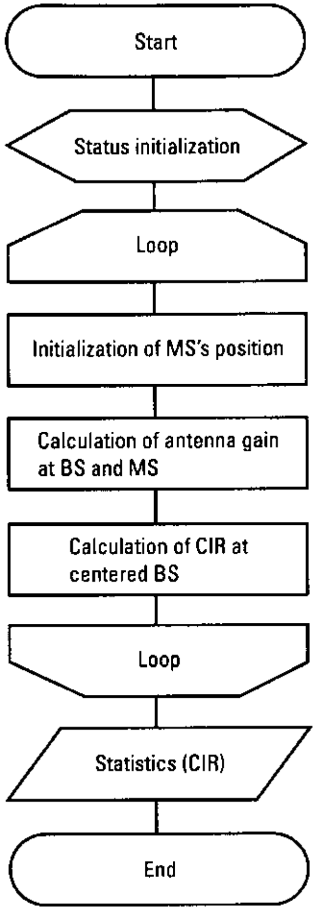

Start
Status initialization
Loop
Initialization of MS's position
Calculation of antenna gain at BS and MS
Calculation of CIR at centered BS
Loop
Statistics (CIR)
End

Figure 7.19 Simulation flowchart.

where x is the antenna gain for the back direction. The parameter n is used to determine the beamwidth θ. This width is within 3 dB of the peak. The value of n is determined by

$$n = \frac{\log_{10}\sqrt{1/2}}{\log_{10}\cos(\theta\pi/180)} \tag{7.9}$$

We can determine the characteristics of $G_{HB_i}(\phi)$, $G_{HM_i}(\phi)$, $G_{VB_i}(\phi)$, and $G_{VM_i}(\phi)$, using (7.8). Of course, we can determine them freely in order to evaluate the CIR improvement using another characteristic of antenna gain.

```
[Loop]
```

The positions of the mobile stations in each cell are determined randomly.

First, we calculate the distances between M_i and B_i to find the required transmission power P_i of M_i ($i = 0\ldots18$). By controlling P_i, we can implement a power control technique. This power control technique is also an effective way to improve CIR and has real relevance in beam control. Using this technique, each mobile station can control the transmission power dynamically in accordance with the distance between the mobile station itself and the desired base station.

Second, we calculate the distance between the central B_0 and M_i to find the path loss $d_i^{-\alpha}$.

Third, we calculate the horizontal and vertical angles θ_{HB_i} and θ_{VB_i} for the target B_i at M_i. In case 1 each M_i sets the peak antenna gain for the target B_i. Namely, since G_{HM_i} has a peak when $\phi = 0$ as shown in (7.8), $G_{HM_i}(0)$ is set at the direction θ_{HB_i} ($i = 0\ldots18$). In the same way, the peak direction of the vertical antenna gain, $G_{VM_i}(0)$, is set for direction θ_{VB_i}. Next, we calculate the horizontal and vertical angles θ_{HB_0} and θ_{VB_0} for the centered B_0 M_j ($j = 1\ldots18$). The difference between θ_{HB_i} and θ_{HB_0} and the difference between θ_{VB_i} and θ_{VB_0} are used to calculate the antenna gains $G_{HM_j}(\theta_{HB_0})$ and $G_{VM_j}(\theta_{VB_0})$, respectively.

In the same way, we calculate the horizontal and vertical angles θ_{HM_0} and θ_{VM_0} for M_0 from the centered B_0, and set the peak direction of the antenna gains $G_{HB_0}(0)$ and $G_{VB_0}(0)$ for θ_{HM_0} and θ_{VM_0}, respectively. Next, we calculate the horizontal and vertical angles θ_{HM_j} and θ_{VM_j} for M_j ($j = 1\ldots18$) from the centered B_0. By using the difference between θ_{HM_j} and θ_{HM_0} and the difference between θ_{VM_j} and θ_{VM_0}, we obtain the antenna gains $G_{HB_0}(\theta_{HM_j})$ and $G_{VB_0}(\theta_{VM_j})$.

Finally, based on (7.6), the CIR of the uplink in the centered base station for cases 1 and 2 are calculated.

```
[statistics (CIR)]
```

After repeating the `[Loop]`, the CIR statistics for cases 1 and 2 are calculated.

7.5.2 Simulation Results

Figure 7.20 shows the results for the CIR improvement of the uplink at the centered base station, highlighting the difference between cases 1 and 2. The parameters are as follows:

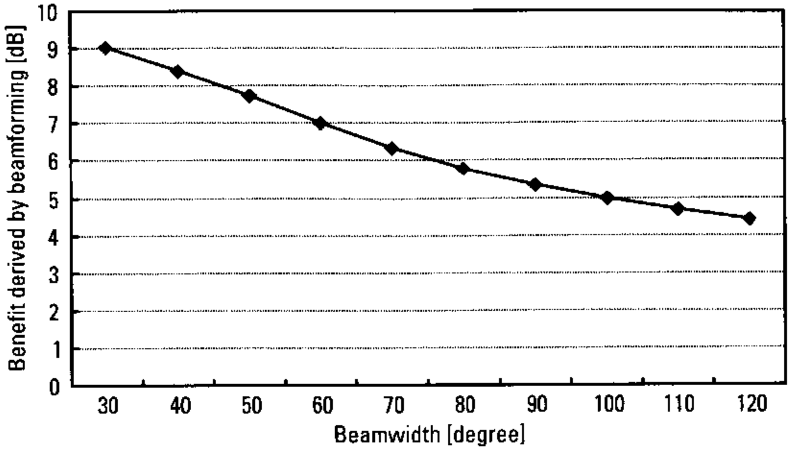


Figure 7.20 CIR improvement.

- $I = 1, J = 1$;
- $G_{HB_i}(\phi)$ is defined by (7.8) where $x = -100$ [dB], $n = 2.41$ ($\theta = 60$);
- $G_{VB_i}(\phi) = G_{HM_i}(\phi) = G_{VM_i}(\phi) = 0$ for all direction ϕ;
- Standard deviation of shadowing sigma = 0;
- Path loss factor alpha = 3.5;
- There is no power control.
- The height of the base station is 0[m] (the macrocell situation was assumed).

In Figure 7.20, the horizontal axis shows the beamwidth at the base station, and the vertical axis shows the difference in CIR between cases 1 and 2 at the center base station. The average CIR with beamforming control shows an improvement of more than 8 dB as compared to that without beamforming control.

Naturally, arranging the program to calculate the downlink improvement is easy. Note that the parameters between the base station and the mobile station are replaced. Note, as well, that we can arrange the program so that multiple users can use the same frequency in a cell. In this case, we must consider the interference calculation from intercell users. When we arrange the program, we must pay close attention to the application of the model and a target user or a base station that is focused in the CIR calculation.

7.6 Conclusions

This chapter focuses on an access scheme simulation to evaluate the total system performance considering multipoint situations. After explaining the concept of a cellular system and a channel assignment algorithm, we introduced an example of a simulation program for a cellular system having DCA, as well as several key techniques that are very useful for conducting these kinds of simulations. Moreover, we also introduced DCA with an array antenna and described the effectiveness of constructing dynamic zones.

References

[1] Balston, D. M., and R. C. V. Macario, *Cellular Radio Systems,* Norwood, MA: Artech House, 1993.

[2] Sampei, S., *Applications of Digital Wireless Technologies to Global Wireless Communications,* Upper Saddle River, NJ: Prentice Hall, 1997.

[3] Gelenbe, E., and G. Pujolle, *Introduction to Queueing Networks,* New York: John Wiley & Sons, 1987.

[4] Lee, W. C. Y., *Mobile Cellular Telecommunications Systems,* New York: McGraw-Hill, 1989.

[5] Godara, L. C., "Applications of Antenna Arrays to Mobile Communications, Part I: Performance Improvement, Feasibility, and System Considerations," *Proc. of IEEE,* Vol. 85, No. 8, August 1997, pp. 1195–1245.

[6] Litva, J., and T. K. Lo, *Digital Beamforming in Wireless Communications,* Norwood, MA: Artech House, 1996.

Appendix 7A

Program 7.1

```
% Program 7-1
%
% dcamain.m
%
% Simulation program to realize DCA algorithm
%
% Programmed by F. Kojima
%

%%%%%%%%%%%%%%%%%%% preparation part %%%%%%%%%%%%%%%%%%%

cnedge = 20.0;  % CNR on cell edge (dB)
cnirth = 15.0;  % CNIR threshold (dB)
lambda = 6.0;   % average call arrival rate (times/hour)
ht = 120.0;     % average call holding time (second)
timestep = 10;  % time step of condition check (second)
timeend = 5000; % time length of simulation (second)
chnum = 5;      % number of channels per each base station

alpha = 3.5;            % path loss factor
sigma = 6.5;            % standard deviation of shadowing

usernum = [5,10,15,20,25]; % number of users per cell

output = zeros(4,5); % output matrix

check = zeros(5,floor(timeend/timestep));
% matrix for transient status

for parameter = 1:5

  rand('state',5);
  randn('state',1);

  user = usernum(parameter); %number of users per cell

  baseinfo = zeros(19, 2);
  %baseinfo(cell #, informations)
  %%%%%baseinfo(:, 1): x coordinates
  %%%%%baseinfo(:, 2): y coordinates

  userinfo = zeros(19, user, 15);
  %userinfo(cell #, user #, informations)
  %%%%%userinfo(:, :, 1): x axis
  %%%%%userinfo(:, :, 2): y axis
  %%%%%userinfo(:, :, 3): attenuation
  %%%%%userinfo(:, :, 4): usage 0->non-connected
  %%%%%1->connected
```

```
%%%%userinfo(:, :, 5): call termination time
%%%%userinfo(:, :, 6): allocated channel #

[baseinfo] = basest;
[wrapinfo] = wrap;

[meshnum, meshposition] = cellmesh;

timenow = 0;
blocknum = 0;
forcenum = 0;
callnum = 0;
users = 0; % number of connected users

%%%%%%%%%%%%%%%%% main loop part %%%%%%%%%%%%%%%%

while timenow timeend

   callnumold = callnum;
   blocknumold = blocknum;
   forcenumold = forcenum;

   %finished calls
   for numcell = 1:19
      for numuser = 1:user
         if userinfo(numcell, numuser, 4) == 1 &
            userinfo(numcell, numuser, 5) < timenow
            userinfo(numcell, numuser, 4) = 0;
            users = users -1;
         end
      end
   end

   %reallocation check
   for numcell = 1:19
      for numuser = 1:user
         if userinfo(numcell, numuser,4) == 1
            reallo = 0; % flag
            cnirdb1 = 0.0;
            dwave = userinfo(numcell, numuser, 3);
            cn = power(10.0, cnedge/10.0) * dwave;
            uwave = 0.0;
            ch = userinfo(numcell, numuser, 6);
            for around = 2:7
               othercell = wrapinfo(numcell, around);
               for other = 1:user
   if userinfo(othercell, other, 4) ...
      == 1 & userinfo(othercell,...
      other, 6) == ch
      userposi(1,1:2) = userinfo...
         (othercell, other, 1:2);
      here = baseinfo(numcell, :);
      there = userposi - baseinfo...
```

```
   (othercell, :) + baseinfo
   (around, :) + baseinfo
   (numcell, :);
uwave = uwave + dist(here,
   there, alpha)*shadow
   (sigma);
   end
      end
   end % around loop
   if uwave == 0
      cnirdb = 10.0*log10(cn);
   else
      cnirdb = 10.0*log10(1/(uwave/
         dwave+1/cn));
   end
   if cnirdb < cnirth
      reallo = 1;
   end

   if reallo == 1
      userinfo(numcell, numuser, 4) = 0;
      users = users -1;
      succeed = 0;
      cnirdb = 0.0;
      for ch = 1:chnum
         available = 1;
         for other = 1:user
            if userinfo(numcell, other, 4)
               == 1 & userinfo(numcell,
               other, 6) == ch
            available = 0;
         end
      end
      if available == 1
         uwave = 0.0;
         for around = 2:7
            othercell = wrapinfo(numcell,
               around);
for other = 1:user
if userinfo(othercell,
   other, 4) == 1 &
   userinfo(othercell,
   other, 6) == ch
   userposi(1,1:2) =
   userinfo(othercell,
   other, 1:2);
   here = baseinfo
      (numcell, :);
   there = userposi -
      baseinfo(othercell,
      :) + baseinfo
      (around, :) + baseinfo
      (numcell, :);
```

```
                uwave = uwave + dist...
                    (here, there, alpha)...
                    *shadow(sigma);
                end
                    end
                        end % around loop
                        if uwave == 0
                            cnirdb = 10.0*log10(cn);
                        else
                            cnirdb = 10.0*log10(1/(uwave/...
                                dwave+1/cn));
                        end
                    else
                        cnirdb = 0.0;
                    end
                    if cnirdb >= cnirth
                        succeed = 1;
                        users = users + 1;
                        userinfo(numcell, numuser, 4) = 1;
                        userinfo(numcell, numuser, 6) = ch;
                        break
                    end
                end % ch loop
                if succeed == 0
                    forcenum = forcenum + 1;
                end
            end % reallo == 1
        end % connected (need to be checked)
    end % user loop
end % cell loop

%new call arrival
for numcell = 1:19
    for numuser = 1:user
        if userinfo(numcell, numuser, 4) == 0 & rand <=...
            lambda*timestep/3600
            callnum = callnum + 1;
            mesh = floor(meshnum.*rand) +1;
            while mesh > meshnum
                mesh = floor(meshnum.*rand) +1;
            end
            userinfo(numcell, numuser, 1:2) = baseinfo...
                (numcell, :) + meshposition(mesh, :);
            succeed = 0; % flag
            cnirdb = 0.0;
            userposi(1,1:2) = userinfo(numcell,...
                numuser, 1:2);
            here = baseinfo(numcell,:);
            there = userposi;
            dwave = dist(here, there, alpha) * shadow...
                (sigma);
            cn = power(10.0, cnedge/10.0) * dwave;
            for ch = 1:chnum
```

```
                available = 1;
                for other = 1:user
                    if userinfo(numcell, other, 4) == 1 &
                        userinfo(numcell, other, 6) == ch
                        available = 0;
                    end
                end
                if available == 1

                    uwave = 0.0;
                    for around = 2:7
                        othercell = wrapinfo(numcell,...
                            around);
for other = 1:user
    if userinfo(othercell, other,....
        4) == 1 & userinfo
        (othercell, other, 6) == ch...
        userposi(1,1:2) = userinfo...
            (othercell, other, 1:2);
        here = baseinfo(numcell, :);
        there = userposi - baseinfo...
            (othercell, :) + baseinfo...
        (around, :) + baseinfo
            (numcell, :);
        uwave = uwave + dist(here,...
            there, alpha)*shadow...
            (sigma);
        end
    end
                    end % around loop
                    if uwave == 0
                        cnirdb = 10.0*log10(cn);
                    else
                        cnirdb = 10.0*log10(1/(uwave/...
                            dwave+1/cn));
                    end
                else
                    cnirdb = 0.0;
                end
                if cnirdb >= cnirth
                    succeed = 1;
                    users = users + 1;
                    userinfo(numcell, numuser, 3) = dwave;
                    userinfo(numcell, numuser, 4) = 1;
                    userinfo(numcell, numuser, 5) =...
                        timenow + holdtime(ht);
                    userinfo(numcell, numuser, 6) = ch;
                    break
                end
            end % ch loop
            if succeed == 0
                blocknum = blocknum + 1;
            end
```

```
         end % new call
      end % user loop
  end % cell loop


  fprintf('%d\t%d\t%d\t%d\t%e\n',parameter,timenow,callnum...
     -callnumold,blocknum-blocknumold,blocknum/callnum);
     check(parameter,timenow/timestep+1) = blocknum/callnum;
     check2(parameter,timenow/timestep+1) = forcenum/...
         (callnum-blocknum);

     timenow = timenow + timestep;
  end %while loop

  %%%%%%%%%%%%%%%%%%%% output part %%%%%%%%%%%%%%%%%%

  output(1,parameter) = callnum;
  output(2,parameter) = blocknum;
  output(3,parameter) = blocknum/callnum;
  output(4,parameter) = forcenum/(callnum-blocknum);
end %parameter loop

fid = fopen('data.txt','w');
fprintf(fid,'UserNumber\t');
fprintf(fid,'%g\t%g\t%g\n', usernum(1,:));
fprintf(fid,'CallNumber\t');
fprintf(fid,'%g\t%g\t%g\n', output(1,:));
fprintf(fid,'BlockNumber\t');
fprintf(fid,'%g\t%g\t%g\n', output(2,:));
fprintf(fid,'BlockingProb. \t');
fprintf(fid,'%g\t%g\t%g\n', output(3,:));
fprintf(fid,'ForcedTerminationProb. \t');
fprintf(fid,'%g\t%g\t%g\n', output(4,:));
fclose(fid);

%********************* end of file ********************
```

Program 7.2

```
% Program 7-2
%
% basest.m
%
% This function sets positions of the base stations.
%
% Programmed by F. Kojima
% Checked by H. Harada
%

function [out] =  basest()
```

```
baseinfo(1, 1) = 0.0;
baseinfo(1, 2) = 0.0;
baseinfo(2, 1) = -0.5*sqrt(3.0);
baseinfo(2, 2) = 1.5;
baseinfo(3, 1) = -sqrt(3.0);
baseinfo(3, 2) = 0.0;
baseinfo(4, 1) = -0.5*sqrt(3.0);
baseinfo(4, 2) = -1.5;
baseinfo(5, 1) = 0.5*sqrt(3.0);
baseinfo(5, 2) = -1.5;
baseinfo(6, 1) = sqrt(3.0);
baseinfo(6, 2) = 0.0;
baseinfo(7, 1) = 0.5*sqrt(3.0);
baseinfo(7, 2) = 1.5;
baseinfo(8, 1) = 0.0;
baseinfo(8, 2) = 3.0;
baseinfo(9, 1) = -sqrt(3.0);
baseinfo(9, 2) = 3.0;
baseinfo(10, 1) = -1.5*sqrt(3.0);
baseinfo(10, 2) = 1.5;
baseinfo(11, 1) = -2.0*sqrt(3.0);
baseinfo(11, 2) = 0.0;
baseinfo(12, 1) = -1.5*sqrt(3.0);
baseinfo(12, 2) = -1.5;
baseinfo(13, 1) = -sqrt(3.0);
baseinfo(13, 2) = -3.0;
baseinfo(14, 1) = 0.0;
baseinfo(14, 2) = -3.0;
baseinfo(15, 1) = sqrt(3.0);
baseinfo(15, 2) = -3.0;
baseinfo(16, 1) = 1.5*sqrt(3.0);
baseinfo(16, 2) = -1.5;
baseinfo(17, 1) = 2.0*sqrt(3.0);
baseinfo(17, 2) = 0.0;
baseinfo(18, 1) = 1.5*sqrt(3.0);
baseinfo(18, 2) = 1.5;
baseinfo(19, 1) = sqrt(3.0);
baseinfo(19, 2) = 3.0;
out = baseinfo;

%******************** end of file ********************
```

Program 7.3

```
% Program 7-3
%
% wrap.m
%
% This function gives informations about cell-wrapping
%
% Programmed by F. Kojima
% Checked by H. Harada
%

function [out] =  wrap(inputmat)

inputmat( 1,1:19) = 1:19;
inputmat( 2,1:19) = [ 2, 9,10, 3, 1, 7, 8,14,13,17,16,...
  11,12, 4, 5, 6,18,19,15];
inputmat( 3,1:19) = [ 3,10,11,12, 4, 1, 2, 9,17,16,15,...
  19,18,13,14, 5, 6, 7, 8];
inputmat( 4,1:19) = [ 4, 3,12,13,14, 5, 1, 2,10,11,19,...
  18,17, 9, 8,15,16, 6, 7];
inputmat( 5,1:19) = [ 5, 1, 4,14,15,16, 6, 7, 2, 3,12,...
  13, 9, 8,19,11,10,17,18];
inputmat( 6,1:19) = [ 6, 7, 1, 5,16,17,18,19, 8, 2, 3,...
  4,14,15,11,10, 9,13,12];
inputmat( 7,1:19) = [ 7, 8, 2, 1, 6,18,19,15,14, 9,10,...
  3, 4, 5,16,17,13,12,11];
inputmat( 8,1:19) = [ 8,14, 9, 2, 7,19,15, 5, 4,13,17,...
  10, 3, 1, 6,18,12,11,16];
inputmat( 9,1:19) = [ 9,13,17,10, 2, 8,14, 4,12,18, 6,...
  16,11, 3, 1, 7,19,15, 5];
inputmat(10,1:19) = [10,17,16,11, 3, 2, 9,13,18, 6, 5,...
  15,19,12, 4, 1, 7, 8,14];
inputmat(11,1:19) = [11,16,15,19,12, 3,10,17, 6, 5,14,...
  8, 7,18,13, 4, 1, 2, 9];
inputmat(12,1:19) = [12,11,19,18,13, 4, 3,10,16,15, 8,...
  7, 6,17, 9,14, 5, 1, 2];
inputmat(13,1:19) = [13,12,18,17, 9,14, 4, 3,11,19, 7,...
  6,16,10, 2, 8,15, 5, 1];
inputmat(14,1:19) = [14, 4,13, 9, 8,15, 5, 1, 3,12,18,...
  17,10, 2, 7,19,11,16, 6];
inputmat(15,1:19) = [15, 5,14, 8,19,11,16, 6, 1, 4,13,...
  9, 2, 7,18,12, 3,10,17];
inputmat(16,1:19) = [16, 6, 5,15,11,10,17,18, 7, 1, 4,...
  14, 8,19,12, 3, 2, 9,13];
inputmat(17,1:19) = [17,18, 6,16,10, 9,13,12,19, 7, 1,...
  5,15,11, 3, 2, 8,14, 4];
inputmat(18,1:19) = [18,19, 7, 6,17,13,12,11,15, 8, 2,...
  1, 5,16,10, 9,14, 4, 3];
inputmat(19,1:19) = [19,15, 8, 7,18,12,11,16, 5,14, 9,...
  2, 1, 6,17,13, 4, 3,10];

out = inputmat;

%******************** end of file ********************
```

Program 7.4

```
% Program 7-4
%
% cellmesh.m
```

```
%
% This function sets meshes in a cell.
%
% Programmed by F. Kojima
% Checked by H. Harada
%

function [meshnum, meshposition] =  cellmesh()

Fineness = 50; % parameter for mesh fineness
k = 1;
j = 0;
ds = sqrt(3.0) / Fineness;
xmesh = -0.5 * sqrt(3.0);
while xmesh 0.5 * sqrt(3.0)
  xmesh = (k - 1) * ds - 0.5 * sqrt(3.0);
  if xmesh 0.0
     ymin = -xmesh/sqrt(3.0) - ds - 1.0;
        % lower line of a cell
     ymax = xmesh/sqrt(3.0) + 1.0;
        % upper line of a cell
  elseif xmesh == 0.0

     ymin = -1.0;
     ymax = 1.0;
  else
     ymin = xmesh/sqrt(3.0) - ds -1.0;
        % lower line of a cell
     ymax = -xmesh/sqrt(3.0) + 1.0;
        % upper line of a cell
  end
  k= k + 1;
  ymesh = ymin;
  while ymesh ymax
     ymesh = ymesh + ds;
     if ymesh  ymax
        break
     end
     j = j + 1;
     posi(j,1) = xmesh;
     posi(j,2) = ymesh;
  end
end
meshnum = j;
meshposition = posi;

%******************** end of file ********************
```

Program 7.5

```
% Program 7-5
%
```

```
% holdtime.m
%
% This function generates holding time
%
% Programmed by F. Kojima
% Checked by H. Harada
%

function [x] =  holdtime(ht)

para = rand;
while para >= 1
  para = rand;
end
x = ht.*(-log(1-para));

%******************** end of file ********************
```

Program 7.6

```
% Program 7-6
%
% shadow.m
%
% This function generates attenuation of shadowing
%
% Programmed by F. Kojima
% Checked by H. Harada
%

function [x] =  shadow(sigma)

anoz = randn;
db = sigma * anoz;
x = power(10.0, 0.1*db);

%******************** end of file *******************
```

Program 7.7

```
% Program 7-7
%
% dist.m
%
% This function generates attenuation due to distance
%
% Programmed by F. Kojima
% Checked by H. Harada
%

function [x] =  dist(a,b,alpha)
```

```
% a,b: position of two points
% size(a) = size(b) = (1,2)

x = sqrt((a-b)*(a-b)');
x = power(x, -1.*alpha);

%********************* end of file *******************
```

Program 7.8

```
% Program 7-8
%
% main.m
%
% Programmed by A. Kanazawa
% Checked by H. Harada
%

clear
%%%%%%%%%%%%%%%% Status initialization
I = 1;       % The cluster size is determined from I and
             % J. (n = I*I + J*J + I*J)
J = 2;
r = 100;     % the radius of the cell[m]
h = 0;       % the height of the BS[m]

D = set_D(I,J,r);
station = stationInit(D);
xbs = real(station);   % The x axis of the BS
ybs = imag(station);   % The y axis of the BS

sigma = 6.5;        % standard deviation of shadowing

alpha = 3.5;        % path loss factor
% margin = 0;       % The parameter for power control

% Characteristics of antenna gain decision for BS
w_HBS = 60;     % [horizontal]: beam width at BS for the
                % target direction [degree]
backg_BS = -100;% [horizontal]: antenna gain at BS for the
                % opposite direction [dB]
w_VBS = 360;    % [vertical]: beam width at BS [degree]

% Characteristics of antenna gain decision for MS
w_HMS = 360;    % [horizontal]: beam width at MS for the
                % target direction [degree]
backg_MS = -100;% [horizontal]: antenna gain at
                % MS for the opposite direction [dB]
w_VMS = 360;    % [vertical]:beam width at MS [degree]

if h == 0,       % In the case of macro cell situation,
  w_VBS = 360; w_VMS = 360;   % the effect of beam tilt
```

```
                              % becomes less.
end

% Antenna gain calculation of each BS
g_HBS = antgain(w_HBS, backg_BS);
g_VBS = antgain(w_VBS, 0);
g_HMS = antgain(w_HMS, backg_MS);
g_VMS = antgain(w_VMS, 0);

%%%%%%%%%%%%%%%% Loop
%-----Initialization of MS positions
N=1000;                 % The number of repeat
for num = 1:N,
  Rx = rand(1,19);      % the random values: [0-1]
  Ry = rand(1,19);      % the random values: [0-1]
  X = r*Rx;
  Y = Ry.* sqrt ( r ^2 - X.^2 );

  tx = 2*((rand(1,19)0.5) -0.5);    % the random values:
                                    % -1 or 1
  ty = 2*((rand(1,19)0.5) -0.5);    % the random values:
                                    % -1 or 1
  x = X.* tx;       % The x axis of the MS when we regard
                    % the position of each BS as (0,0)
  y = Y.* ty;       % The y axis of the MS when we regard
                    % the position of each BS as (0,0)

  x2 = x+xbs.';     % The x axis of the MS when we regard
                    % the position of central BS as (0,0)
  y2 = y+ybs.';     % The y axis of the MS when we regard
      the position of central BS as (0,0)

% The complex expression of MS when we regard the position
% of each BS as (0,0)
  z(1,:) = x + i * y;
% The complex expression of MS when we regard the position
% of central BS as (0,0)
  z(2,:) = x2+ i * y2;

  d(1,:) = abs(z(1,:));   % The distance between BS_i
                          % and MS_i in horizontal axis
  d(2,:) = abs(z(2,:));   % The distance between central
                          % BS and MS_i in horizontal axis
  d2 = sqrt(d.^2 + h^2);  % The distance

  phai(1,:) = angle(z(1,:)); % The angle difference
                             % between BS_i and MS_i [rad]
  phai(2,:) = angle(z(2,:)); % The angle difference
                             % between central BS and
                             % MS_i [rad]
  deg = phai*180/pi;         % the conversion of radian
                             % to degree
```

```
  if h ==0, degH = 90*ones(1,19);
     else
        % the elevation angle between central BS and MS_i
        phaiH = atan(d(2,:)/h);
        % the conversion of radian to degree
        degH = phaiH*180/pi;
     end

%-------- shadowing --------
  for m = 1:19
     g(m) = 10*log10(shadow(sigma));
  end

%-------- propagation loss --------
  Loss(1,:) = 10 * log10(d2(1,:).^alpha);
     % The propagation loss from MS_i to BS_i [dB]
  Loss(2,:) = 10 * log10(d2(2,:).^alpha);
     % The propagation loss from MS_i to BS_0 [dB]
  Loss_max = 10 * log10(r.^alpha);
     % The propagation loss from the cell boundary to
     % BS [dB]

% Free space loss
% wl = 0.1;
% Loss(1,:) = 10 * log10((4*pi*d2(1,:)/wl).^2);
  % The propagation loss from MS_i to BS_i [dB]
% Loss(2,:) = 10 * log10((4*pi*d2(2,:)/wl).^2);
  % The propagation loss from MS_i to BS_0 [dB]
% Loss_max = 10 * log10((4*pi*r/wl).^2);
  % The propagation loss from the cell boundary to BS [dB]

%----- Transmission power level of each MS [dB] -----

  Ptm_0= Loss_max*ones(1,19);      % no power control
% Ptm_0= Loss(1,:) + margin;       % power control
                                   % (with margin [dB])

%--Calculation of antenna gain for the target direction

% the angle difference between the MS_0 and MS_i from
% central BS
  deg_B = deg(2,1)-deg(2,:);
% the angle difference between the BS_0 and BS_i from MS_i
  deg_M = deg(1,:)-deg(2,:);

  degHBS = mod(round(deg_B),360);
  degHMS = mod(round(deg_M),360);
  degVBS = round(degH-degH(1)); % the angle difference
     % in vertical direction between MSs and central BS
  degVMS = degVBS;     % the angle difference in
     % vertical direction between MSs and central BS
```

```
 %----- Calculation of CIR at centered BS -----
 %Control
 CIdB_a= Ptm_0(1:19)+g_HBS(degHBS(1:19)+1) + g_VBS...
   (degVBS(1:19)+1) + g_HMS(degHMS(1:19)+1) + g_VMS...
   (degVMS(1:19)+1)- Loss(2,1:19)-g(1:19);
   % Received level at central BS (beam forming)
 CIw_a = 10 .^ ( CIdB_a ./ 10 );      % dB → W
 isum_a = sum( CIw_a(2:19));
 CIR_a(num) =   CIw_a(1) / isum_a;
 %No Control
 CIdB_o= Ptm_0(1:19)- Loss(2,1:19)-g(1:19);
   % Received level at centered BS (Omni)
 CIw_o= 10 .^ ( CIdB_o ./ 10 );       % dB → W
 isum_o = sum( CIw_o(2:19));
 CIR_o(num) =  CIw_o(1) / isum_o;

 %----- Calculation of CIR under various w_HBS
 %ii = 1;
 %for w_HBS2=30:10:180,
 %  g_HBS2 = antgain(w_HBS2, backg_BS);
 %  CIdB_a2= Ptm_0(1:19)+g_HBS2(degHBS(1:19)+1) + ...
 %  g_VBS(degVBS(1:19)+1) + g_HMS(degHMS(1:19)+1) + ...
 %  g_VMS(degVMS(1:19)+1)- Loss(2,1:19)-g(1:19);
    % Received level at central BS (beam)
 %  CIw_a2 = 10 .^ ( CIdB_a2 ./ 10 );   % dB → W
 %  ciw_a2 = sum( CIw_a2(2:19));
 %  CIR_a2(num,ii) =   CIw_a2(1) / ciw_a2;
 %  ii = ii+1;
 %end

end
%---- statistics

CA = 10 * log10(sum(CIR_a)/N);
CO = 10 * log10(sum(CIR_o)/N);

%---- result
CA-CO                          % Improvement

%---- Calculation of CIR under various w_HBS
% CA2= 10 * log10(sum(CIR_a2)/N);
% CA2-CO
% plot(30:10:/20,CA2-CO)

%******************** End of file ********************
```

Program 7.9

```
% Program 7-9
%
% set_D.m
%
```

```
% Programmed by A. Kanazawa
% Checked by H. Harada
%

function D = set_D(x,y,R)

% The funstion to determine the distance between BSs
i_R = sqrt(3) * R;
j_R = i_R / sqrt(2);

D2 = 0+0j;
for k = 1:x
  D2 =  D2 + i_R;
end

for k = 1:y
  D2 = D2 + j_R + j * j_R;
end

D = abs(D2);

%********************** end of file ********************
```

Program 7.10

```
% Program 7-10
%
% stationInit.m
%
% Programmed by A. Kanazawa
% Checked by H. Harada
%

function STATION = stationInit(d)

% The function to determine the position of BS
STATION = zeros(19,1);

for i = 0:5
  STATION(i + 2, 1) = d * cos(i * pi / 3.0) + j * d...
     * sin(i * pi / 3.0);
end

for i = 0:11
  STATION(i + 8, 1) = 2 * d * cos(i * pi / 6.0) + j * 2...
     * d * sin(i * pi / 6.0);
end

%********************** end of file*******************
```

Program 7.11

```
% Program 7-11
%
% antgain.m
%
% Programmed by A. Kanazawa
% Checked by H. Harada
%

function gain = antgain(width, back)

theta = [0:359];
gain = zeros(1,360);

if width ~= 360,

   n = log10(sqrt(1/2))./log10(cos(width*pi/360));
       % the coefficient n
   gain(1:89) = 10*log10(cos(theta(1:89)*pi/180).^n);
       % [dB] Antenna gain
   gain(272:360) = 10*log10(cos(theta(272:360)*pi/180).^n);
       % [dB] Antenna gain

   if back ~= 0,     % definition of antenna gain for
                     % horizontal direction[dB]
       gain(90:271) = back;              %[dB]
   end
end

%********************** end of file*********************
```

8

Software Radio Communication Systems

8.1 Introduction

As the number of systems that users can employ increased, there have been increasing demands for the coexistence of several mobile telecommunication services (e.g., GSM, IS-54, IS-136, IS-95, Japanese PDC or PHS, or the IMT-2000 system). This is because there are a lot of air interfaces in the world, and most mobile phone users disapprove of carrying more than one mobile terminal.

Readers of this book should already know that any communication system—whether for radio transmission schemes or multiple access schemes—can be written in programming languages such as MATLAB or C. These computer simulation languages have a good relationship with software languages that configure DSPH such as DSP and/or FPGAs and/or ASIC. Typical software languages for DSPH are VHDL and Verilog-HDL. DSPH has been utilized to configure the mobile terminals and base stations of mobile communication systems. The DSPH is general-purpose, and the configuration can be programmed by downloading *DSP software* (DSPS). This means that users can download DSPS describing the desired elemental components into the DSPH of only one terminal. This is the basic concept of a software radio communication system.

If software radio is realized in the near future, mobile terminal manufacturers will be able to make products that are independent of the specifications of a particular telecommunication device. This means that users can select providers as they like (e.g., if a user finds that the capacity of one provider company is full in a cellular zone, he or she can select another provider in the same cellular zone). There are benefits from the point of an ecological view as well; software radio reduces the amount of hardware, which, in turn, reduces the amount of industrial waste. Moreover, from a cost-saving viewpoint of the manufacturer, instead of selling the mobile terminal, the manufacturer will only have to sell the software for the implementation of DSPH in the mobile terminal.

Research into radio communication systems based on DSPH started around the beginning of the 1990s. However, these early studies did not examine DSP-based radio communication systems from the viewpoint of reconfigurable radio communication systems. As far as we know, the first paper to coin the phrase "software radio" is that of Mitola [1]. Mitola described the fundamental and functional architecture of software radio. Moreover, one prototype, the military software radio SPEAKeasy was introduced in [2]. SPEAKeasy can use several voice and data military services; its frequency band is 2 MHz–2 GHz.

Most research into realizing the ultimate communication system has focused on (1) the architecture of the DSPH, (2) the configuration of the analog signal processing hardware, (3) the method of downloading software to the hardware, and (4) the method of application of software defined radio [3–7].

This chapter introduces the software radio communication system as a future application of the software programming method studied in this book. The concept of software radio is one that bridges software programming and real hardware implementation. The chapter is organized as follows: Section 8.2 defines what is meant by a software radio communication system. Section 8.3 lists the advantages of a software radio communication system, and Section 8.4 lists the impediments to realizing a practical system. Section 8.5 describes software radio communication system technologies. Section 8.6 reviews current software radio research, and Section 8.7 discusses future applications of a software radio–based communication system. Section 8.8 concludes the chapter.

8.2 Definition of Software Radio Communication System

Figure 8.1 shows the basic configuration of the software radio of [1]. The radio consists of eight units: (1) the antenna unit, (2) the *radio frequency signal processing unit* (RFU), (3) the *intermediate frequency signal processing unit* (IFU), (4) the *analog-to-digital conversion* (ADC) and *digital-to-analog conversion* (DAC) unit (ADC/DAC), (5) the *baseband signal processing unit* (BBU), (6) *transmission control unit* (TCU), (7) the *input/output* (I/O) *processing unit* (IOU), and (8) the end-to-end *timing processing unit* (TPU). Each unit will be described in detail.

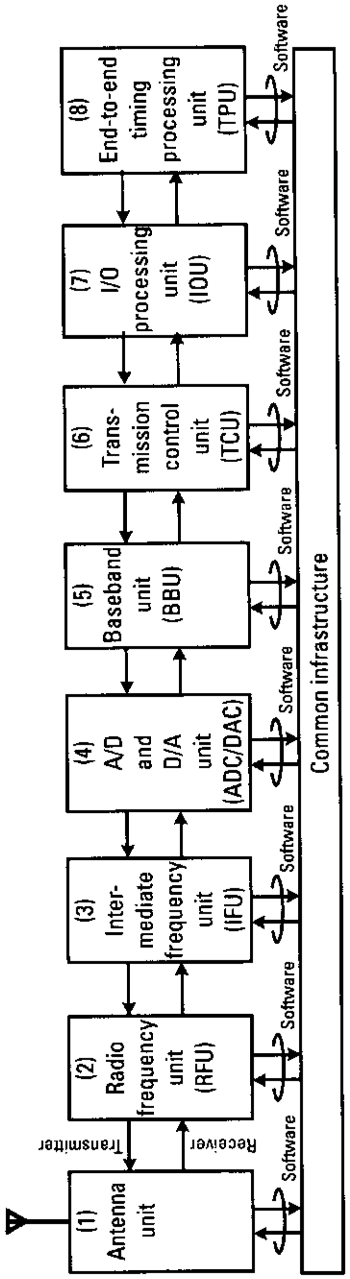

(1) Antenna unit
Transmitter
Receiver
(2) Radio frequency unit (RFU)
(3) Inter-mediate frequency unit (IFU)
(4) A/D and D/A unit (ADC/DAC)
(5) Baseband unit (BBU)
(6) Trans-mission control unit (TCU)
(7) I/O processing unit (IOU)
(8) End-to-end timing processing unit (TPU)
Software
Software
Software
Software
Software
Software
Software
Software
Common infrastructure

Figure 8.1 Basic configuration of software radio.

8.2.1 Antenna Unit

An omnidirectional, low-loss, and broadband antenna is required because it can be used by a variety of wireless communication systems. Moreover, signal processing technology based on array antennas makes it possible to select the performance of the software radio according to the surroundings and to perform optimum selection of the algorithm by using software radio technology. Such an antenna is called a smart antenna or software antenna [8–10]. The software antenna is capable of SDMA, in which the antenna configures the beam in the direction of selected users. Multiple access is achieved by changing the direction of the antenna, or by interference cancellation, in which the antenna configures its direction to the desired user or allocates null points to the direction of undesired users or signals.

8.2.2 Radio Frequency Signal Processing Unit (RFU)

In the transmitter's RFU, the signals coming from the *intermediate frequency unit* (IFU) or *baseband unit* (BBU) are upconverted to the *radio frequency* (RF) band signals, amplified, and transmitted to the antenna unit. At the receiver, the signals received by the antenna unit are amplified to a constant level that is suitable for signal processing and directly downconverted to a lower frequency band such as IF band or baseband. The signal processing is done by an analog circuit. The linearity or efficiency of the RF amplifier and the conversion method to the lower frequency band at the receiver are main discussion points.

8.2.3 Intermediate Frequency Signal Processing Unit (IFU)

In this unit, the signals from the ADC/DAC unit are upconverted to the IF band signal, amplified, and transferred to the RFU of the transmitter. At the receiver, the signals from the RFU unit are amplified to an adequate level for signal processing in the IFU and directly downconverted to a suitable frequency for the ADC/DAC unit or baseband unit. When the signals of several systems are received at the receiver, the required frequency band must be selected by using a filter.

8.2.4 Analog-to-Digital Conversion (ADC) and Digital-to-Analog Conversion (DAC) Unit

In this unit, the digital signal from the baseband unit is converted to an analog signal by using a DAC, and the converted signal is transferred to the upper frequency band unit (IFU or RFU). At the receiver, the signals from the IFU or RFU are amplified to an adequate level for ADC. The stabilized signal is then sampled by ADC and converted to a digital signal. In this unit, the sampling method is a key technology. Section 8.5.2 describes the three main sampling methods.

8.2.5 Baseband Signal Processing Unit (BBU)

In this unit, data is digitally modulated and transferred to the ADC/DAC unit of the transmitter. Transmitted data is recovered by using the sampled signal from the ADC/DAC unit and digital signal processing at the receiver. The basic configuration of the BBU is shown in Figure 8.2 [2]. In the BBU of the transmitter, frame, coding, mapping and modulation, and transmission filter blocks are the key blocks. On the other hand, in the BBU of the receiver, receiver filter, code and symbol timing, sampling rate conversion (resample), demapping and demodulation and decoding blocks are the key blocks. Moreover, in the BBU of the receiver, the fading compensation (equalization) block and the interference cancellation block for eliminating undesired signals are present. In most cases, the BBU is configured by several DSPHs such as DSP, FPGA, or ASIC. All component blocks are described using DSPS written in Verilog-HDL or VHDL and compiled. The BBU's configuration can be changed by changing the DSPS.

8.2.6 Transmission Control Unit (TCU)

In this unit, the input bit stream format for the BBU is configured at the transmitter by adjusting the transmission protocol of the *media access control* (MAC) layer, and at the receiver, the detected data from the BBU is checked according to the data format of the transmission protocol of the MAC layer. If the number of bit errors in the detected data is large, retransmission is requested. In addition to this transmission control, this unit can manage cryptography. In most cases, TCU can be configured by a range of DSPHs, and all the component blocks can also be described by using DSPS. By changing DSPS, TCU can configure the transmission protocol as a user likes.

8.2.7 Input/Output (I/O) Processing Unit

In the mobile station, all information data comes from a handset, PDA terminal, or personal computer, and all received data comes back to these terminals or computers. The I/O and timing are managed so as to connect with the external terminals flexibly.

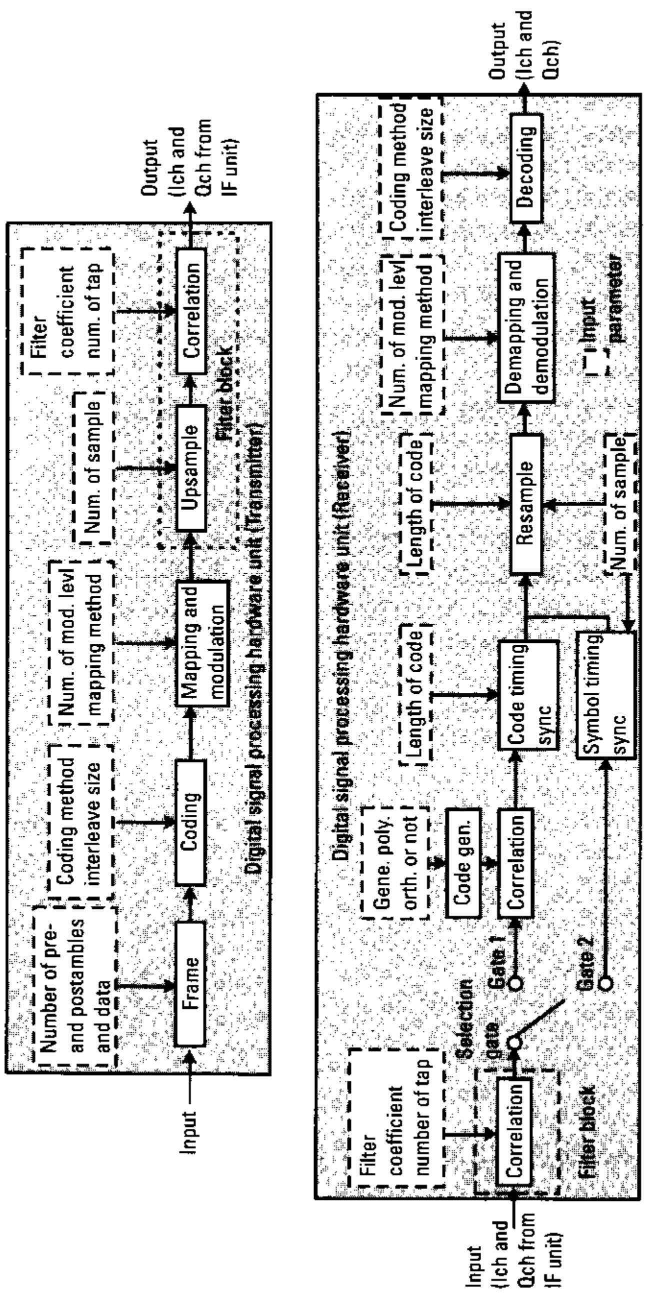

Input
Number of pre- and postambles and data
Frame
Coding method interleave size
Coding
Num. of mod. levl mapping method
Mapping and modulation
Num. of sample
Upsample
Filter coefficient num. of tap
Correlation
Filter block
Digital signal processing hardware unit (Transmitter)
Output (Ich and Qch from IF unit)
Input (Ich and Qch from IF unit)
Filter coefficient number of tap
Correlation
Filter block
Selection gate
Gate 1
Gate 2
Gene. poly. orth. or not
Code gen.
Correlation
Length of code
Code timing sync
Symbol timing sync
Length of code
Resample
Num. of sample
Num. of mod. levl mapping method
Demapping and demodulation
Input parameter
Coding method interleave size
Decoding
Digital signal processing hardware unit (Receiver)
Output (Ich and Qch)

Figure 8.2 Basic BBU configuration.

8.2.8 End-to-End Timing Processing Unit

This unit controls the transmission delay between the transmitter and receiver. For example, the transmission delay for voice must be within 150 ms.

In most software radio systems, several software programs, which describe all the telecommunications components in DSPS language, are used to configure the components on the DSPH. This software can be readily changed to suit the requirements of a particular system. Such a software radio system is called a full-download-type software radio system. Figure 8.3 shows the configuration of a full-download-type software radio system. The system has an RFU, IFU, and BBU as in the basic system. Moreover, it has a TX module that is related to the transmitter and an RX module that is related to the receiver.

Implementing a specific telecommunication system with a full-download-type software radio system, requires first of all, all necessary DSPS to be downloaded to the BBUs before starting communication. DSPS blocks, including frame block, encoder block, mapping and modulation block, and filter block (Figure 8.2) are downloaded to the BBU of the TX module. DSPS blocks, including filter block, equalizer block, detector, and decoder

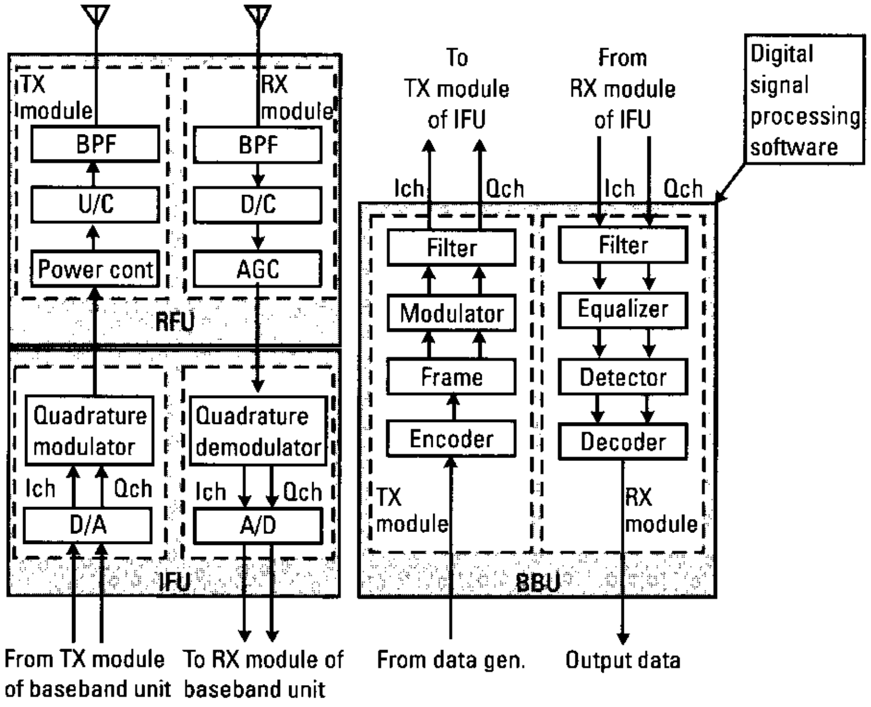


Figure 8.3 Configuration of a full-download-type software radio system.

block (Figure 8.2), are also described in DSPS and downloaded to the BBU of the RX module. After the software has been downloaded, the configuration check program is executed. The BBU then configures the required baseband modulation and demodulation circuit. Then, transmitted data can then be fed into the BBU of the TX module.

In the BBU of TX module, this data is formatted into frames, modulated, and converted into two signals, in-phase channel (Ich) and quadrature-phase channel (Qch) signals, by the DSP blocks mentioned above. These signals are fed into the IFU of the TX module.

In the IFU of the TX module, the digitally modulated Ich and Qch signals are converted from digital to analog by a D/A converter block. These signals are quadrature-modulated on the IF band and sent to the RFU of the TX module. In the RFU of the TX module, the quadrature-modulated signal is upconverted to the RF band by the power control part before being transmitted.

To receive the RF signal, first of all, the received signal is fed into the RFU of the RX module. Here, the received RF data is bandpass-filtered, eliminating spurious signals, and down-converted to the IF band. The *automatic gain control* (AGC) block keeps the power of the downconverted signal at a constant level. This power-controlled signal is fed into the IFU of the RX module.

In the IFU of the RX module, the received signal from the RFU is split into Ich and Qch signals by using a quadrature demodulator block. The A/D converter block then oversamples and transfers them into the BBU of the RX module.

In the BBU of the RX module, all telecommunications component blocks have been implemented into the DSPH before starting communication, and the configuration of what has been checked by a test program. The oversampled Ich and Qch are filtered and equalized by a customized method and detected and decoded by using the filter, equalizer, and decoder blocks in the BBU of RX module.

8.3 Advantages of the Software Radio Communication System

In full-download-type software radio, the system can be changed on demand by changing software. Needless to say, there are many gains for not only operators and service providers but also for government and commercial customers.

In particular, for commercial operators and service providers, the advantages are listed as follows:

- Global roaming services;
- Upgradeable terminals;

- New services added without having to change the terminal;
- Bugs fixed without the need to recall the product;
- Versatile software (i.e., wireless communication software that can be installed in other electrical products as well as in the mobile terminal). This means that not only mobile users but also users of other devices sometimes can access software radio providers.

For the government, the advantages are listed as follows:

- Global roaming services can be offered to customers.
- From an ecological viewpoint, software radio eliminates hardware, because it needs only one mobile terminal. This means that the amount of industrial waste can be reduced.
- It should be easy to certify terminals according to radio law. Ultimately, only software may have to be certified.

For commercial customers, the advantages are the following:

- Unlimited global roaming;
- One terminal for many services;
- New services provided without the need to upgrade hardware;
- New services added to the terminal without changing the terminal;
- Bugs fixed without the need to recall the product.

8.4 Problems in the Software Radio Communication System

As shown in Section 8.3, software radio has many advantages. However, software radio technology must overcome the following problems.

- The volume of software downloaded to the DSPH increases as the contents of the required telecommunications component blocks become more complicated. As a result, the download time is lengthened. In addition, the redundancy code for coding must be added to the download software when the concealment of the download program or the robustness for all jamming signals or fading is considered. In this case, the download time becomes even longer [11, 12].
- The period for the configuration check of the DSPH also increases as the contents of the required telecommunications component blocks

become more complicated. The problem also affects the stability of the operating characteristic of the DSPH when there is not a sufficient period of time for the configuration check [11, 12].

- In the downloaded software, there are often several component blocks containing manufacturer-specific know-how (e.g., the optimization method or calculation algorithm for some special blocks). There is the possibility that this know-how may leak out when the software is downloaded. There is also the possibility that it may be tampered with [11, 12].

In addition to the above problems, if software radio controls RF and IFU as well as the baseband unit, the following problems must be considered.

- By using software radio, a user has only to download the software describing the elemental components to the hardware for realizing a particular communication system. However, if several communication systems operating on different frequency bands are integrated into one mobile communication hardware by software radio technology, several antenna units, RFUs, IFUs, and ADC/DAC units are needed in the mobile communication hardware. Moreover, a user may want to use several systems simultaneously [13], in which case several ADC/DAC units, baseband units, transmission control units, I/O processing units, and end-to-end timing processing units must be prepared. The method for managing several units must be considered.
- If several communication systems operating on different frequency bands are integrated into one mobile communication hardware by software radio technology, a broadband antenna must be used. However, since the bandwidth of an antenna is limited, several antennas are needed. The placement of these antennas in the terminal is an important consideration.

Moreover, from the viewpoint of the entire system, the following questions arise.

- Which services are to be integrated in one software radio terminal? Say that a multimode terminal can easily be realized by developing a specific ASIC for each communication system and planning these ASICs on one circuit board. In this case, software radio technology is unnecessary. Therefore, software radio technology should be targeted to application fields in which many communication systems are required.

- How and in which field is the software radio technology socially recognized? There are many applications for software radio technology. These applications must be arranged.
- How and in which field is standardization to be done? How should the software be protected against viruses or hackers? In particular, software radio's capability of international roaming by changing software instead of changing terminal becomes a serious threat, if a virus is included that can reconfigure a wireless communication system. A simple example is the reconfigured terminal in which the transmitter no longer matches the receiver. Moreover, a virus altering the transmission power to a much higher level than the regulation threatens to shut down all wireless communication systems in the world. Reference [14] describes "wireless terrorism." In wireless terrorism, all users of software radio effectively become potential terrorists. To prevent such an incident from occurring, standardization must be done. In addition, the organizations and radio that certify software radio terminals must be developed. Software radio thus presents a new paradigm in radio law.

8.5 Technology of a Software Radio Communication System

8.5.1 Direct Conversion Technology

Suppose that several communication systems operating on different frequency bands are integrated into one mobile communication hardware by software radio technology and the difference in frequency is quite large. In this case, several broadband antennas and several RFUs that are matched with these different frequencies must be prepared at the receiver. However, if the difference in the frequency between the integrated communication systems is small, one antenna can cover the frequency band. However, if the RFU and IFU are heterodyne types as shown in Figure 8.4(a), several RFUs are needed. In this case, if new systems are serviced in the same frequency band, new RFUs must be prepared. Consequently, the size of the analog circuit must be reduced.

The direct conversion method can solve this problem [15, 16]. Figure 8.4(b) shows the basic configuration, and Figure 8.5 illustrates the procedure. First, the target center frequency and target bandwidth are selected. By using a quadrature demodulator and an image rejection filter, the bandwidth can be converted from the RF band to the baseband directly. [See Figure 8.5(a) and 8.5(b).] Then, by using an ADC, the received signal can be sampled at a rate of twice the target bandwidth. [See Figure 8.5(c).] The required channel for the

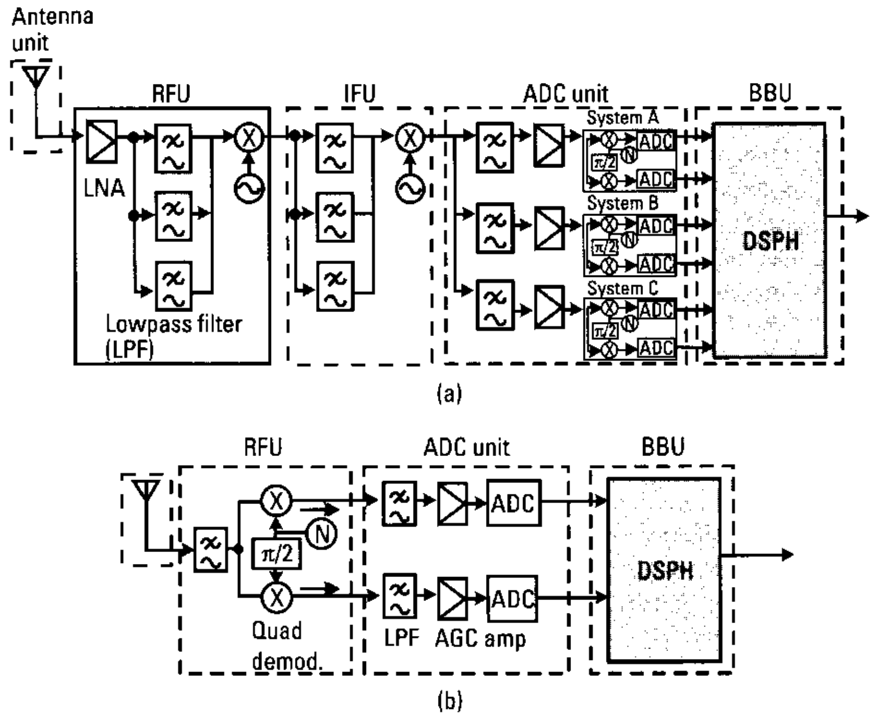


Figure 8.4 Direct conversion method: (a) heterodyne-type construction and (b) direct-conversion-type construction.

required communication system can be filtered digitally from the sampled data, and the signal is recovered by quadrature demodulation in the baseband unit. By using the direct conversion method, the number of RFUs can be reduced as shown in Figure 8.4. However, the dynamic range is reduced whenever a high received-signal-level input enters in the target bandwidth. Furthermore, there are issues of second-order distortion caused by the nonlinearity of the mixer and dc offset after converting from RF to baseband. For the direct conversion, many papers and trials [15–17] considered the technique to apply for the communication systems based on software radio. In the near future, broadband and multiband direct converters will be utilized.

8.5.2 Sampling Method

To realize a multimode receiver with software radio technology, sampling is done in either the RFU, IFU, or BBU. In this section, these samplings are respectively called RF sampling, IF sampling, and BB sampling. Figure 8.6

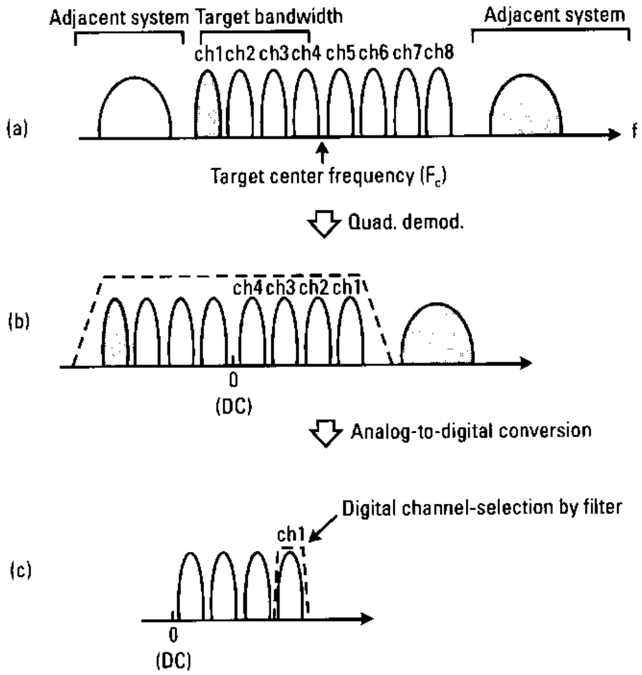

Adjacent system
Target bandwidth
Adjacent system
ch1ch2ch3ch4 ch5ch6 ch7ch8
(a)
f
Target center frequency (F$_c$)
Quad. demod.
ch4ch3ch2ch1
(b)
0
(DC)
Analog-to-digital conversion
Digital channel-selection by filter
ch1
(c)
0
(DC)

Figure 8.5 Procedure of direct conversion.

shows these sampling methods. In the case of BB sampling, all computation can be used for baseband signal processing, because frequency conversion and other necessary steps have been finished in the analog signal processing section in the RFU and IFU. However, in the case of IF and RF sampling, the load of the signal processing section becomes heavy and must be reduced [18–20]. As for the sampling methods, we can categorize them as: (1) Nyquist sampling, (2) oversampling, or (3) undersampling.

8.5.2.1 Nyquist Sampling

The sampling frequency f_s is twice as high as the maximum frequency of the input signal. Figure 8.7 shows the method of Nyquist sampling. However, this sampling method needs an anti-aliasing filter with steep attenuation characteristics.

8.5.2.2 Oversampling

The sampling frequency f_s is higher than the Nyquist sampling frequency. Figure 8.8 shows this method. In comparison with Nyquist sampling, the cutoff of the anti-aliasing filter does not have to be steep.

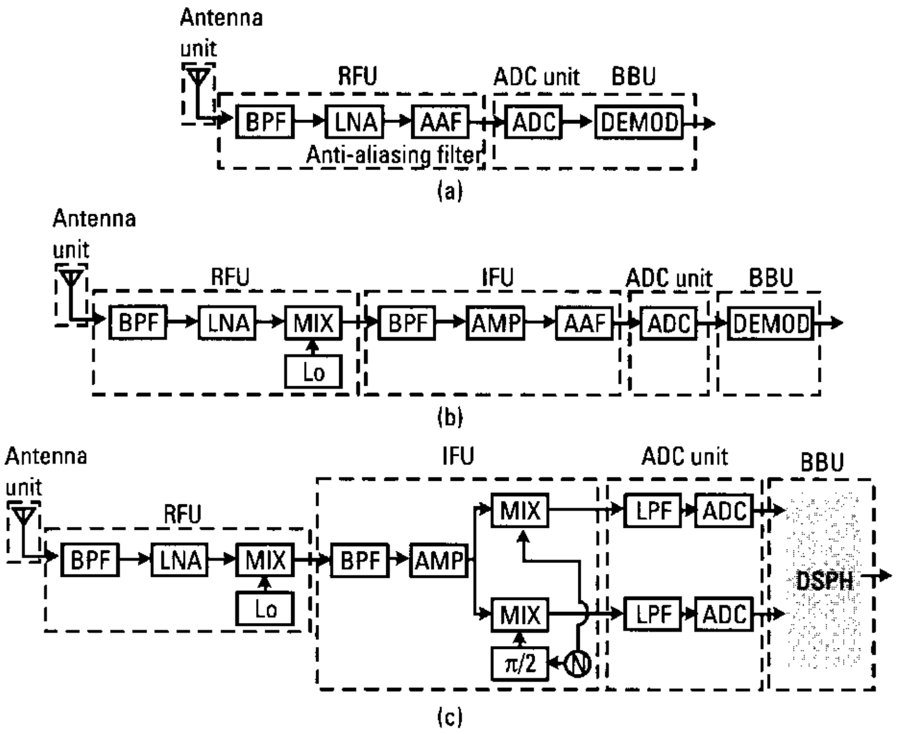


Figure 8.6 Sampling methods: (a) RF sampling, (b) IF sampling, and (c) baseband sampling.

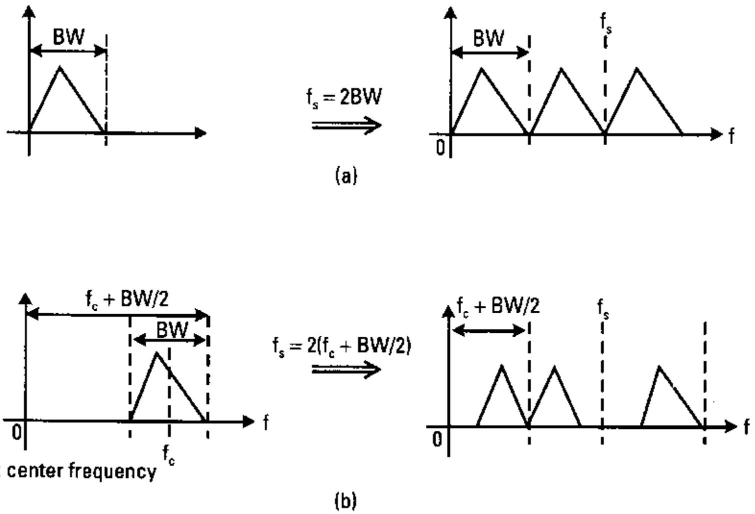


Figure 8.7 Nyquist sampling: (a) baseband signal and (b) band-limited signal.

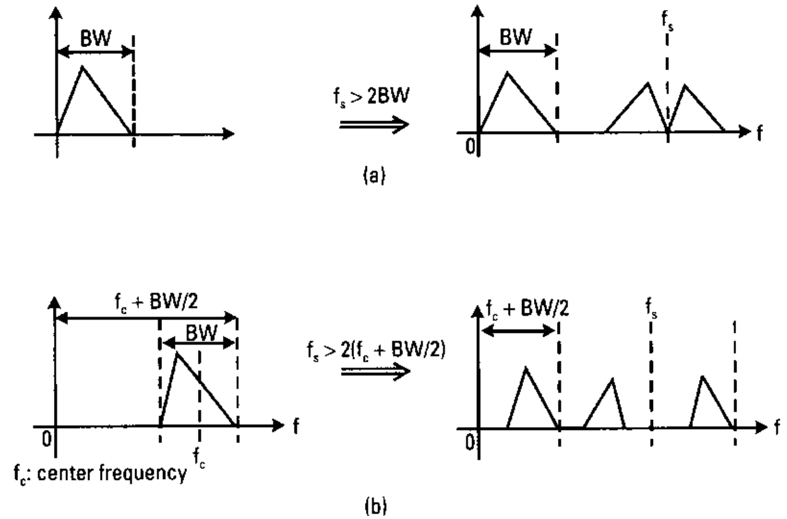


Figure 8.8 Oversampling: (a) baseband signal and (b) band-limited signal.

8.5.2.3 Undersampling

The sampling frequency f_s is longer than the Nyquist sampling frequency. By using the Nyquist theorem, if the sampling frequency is lower than the Nyquist sampling frequency, aliasing occurs. However, in this case, the information bandwidth of the RF and IF signal is band-limited and much lower than the RF and IF carrier signal. Therefore, the influence of the aliasing is reduced. However, since the sampling frequency is lower than the Nyquist sampling frequency, the relationship between the band-limited input signal to the sampler and sampling frequency is very important. When the sampling frequency is defined as f_s and the lowest and highest frequencies of the band-limited input signal are defined as f_L and f_u, and the required sampling frequency is given by [20]:

$$\frac{2f_u}{n} \le f_s \le \frac{2f_L}{n-1} \tag{8.1}$$

where n is an integer satisfying the following condition

$$1 \le n \le \lfloor f_u/(f_u - f_L) \rfloor \tag{8.2}$$

$\lfloor \bullet \rfloor$ means maximum integer of •.

Undersampling is shown in Figure 8.9. Frequency conversion can be done at the same time by performing undersampling when an information signal near the baseband is selected. As shown in the above, the undersampling has advantages. However, it is difficult to keep the sampling frequency within the area defined by (8.1), and sometimes the clock fluctuates because of jitter. Therefore, aliasing still occurs. To reduce the occurrence of aliasing, an *anti-aliasing filter* (AAF) is used. The design of such a filter is a future discussion topic.

8.5.3 Sampling Rate Conversion

If several communication systems operating on different frequency bands are integrated into one mobile communication hardware by software radio technology, their symbol rates are completely different. If the sampling rate is fixed, symbol timing errors often occur. This is because the value that divided the sampling rate by symbol rate is not an integer. Consequently, timing jitter or slipping occurs. Figure 8.10 shows the behavior of symbol timing when the sampling rate is integer or noninteger times for the received signal's symbol rate.

Sample rate conversion is used to prevent symbol timing errors. Figure 8.11 illustrates this simple method [21], and Figure 8.12 shows the transition of the frequency spectrum during sampling rate conversion. For the input signal, the first sampling is performed at the sampling rate of T_1. $x(kT_1) : k = 1, 2, \ldots$ is the sampled data, and its spectrum is shown in Figure 8.12(b). $x(kT_1)$ is converted to $y(mT_2) : m = 1, 2, \ldots$, at the sampling rate of T_2. To do so, the sampled data is reconstructed to analog signal by using DAC and lowpass filter with the transfer function in Figure 8.12(c) and an analog signal $y(t)$ is obtained. The spectrum of $y(t)$ is shown in Figure 8.12(d). The analog signal $y(t)$ is then resampled at the rate of T_2 and finally, we can obtain rate converted signal $y(mT_2) : m = 1, 2, \ldots$. The spectrum of the analog signal $y(t)$,

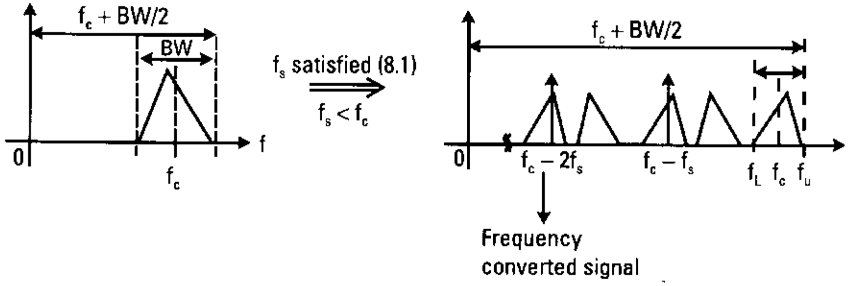


Figure 8.9 Undersampling.

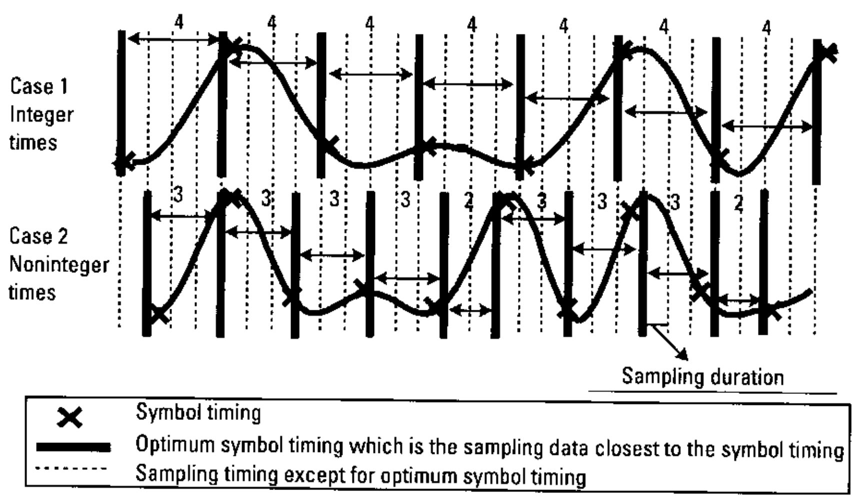

4
Case 1
Integer
times
3
2
Case 2
Noninteger
times
Sampling duration
Symbol timing
Optimum symbol timing which is the sampling data closest to the symbol timing
Sampling timing except for optimum symbol timing

Figure 8.10 Behavior of symbol timing when the sampling rate is integer or noninteger times for the received signal's symbol/chip rate.

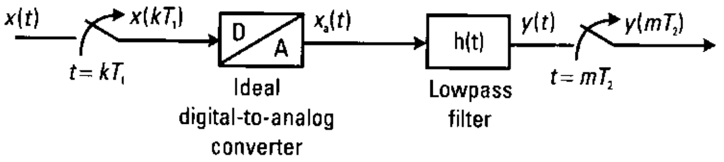


Figure 8.11 Sampling rate conversion method.

which has been converted from $y(mT_2) : m = 1, 2, \ldots$ by using an ideal DAC, is shown in Figure 8.12(e). If the impulse response of the lowpass filter is ideal, $y(t) = x(t)$. However, the configuration of a digital filter is time-varying and not ideal; therefore, the image signals do not cancel perfectly during the lowpass filter process. Thus, aliasing is included in the spectrum of $y_a(t)$. Reduction of this aliasing is a research topic in sampling rate conversion [21].

By using the sampling rate conversion technique, we can adaptively change the sampling rate. However, the sampling rate conversion is performed by analog circuits that include ADCs, a DAC, and a lowpass filter. Therefore, if the number of systems used in the software radio increases, several analog devices perform rate conversion circuits are required. Increasing the number of analog circuits is not practical because the terminal must be small. Therefore, adaptive symbol timing methods, which find adequate symbol timings for several systems under a fixed sampling rate on the baseband unit, must be considered. One of these methods [22] is shown as follows.

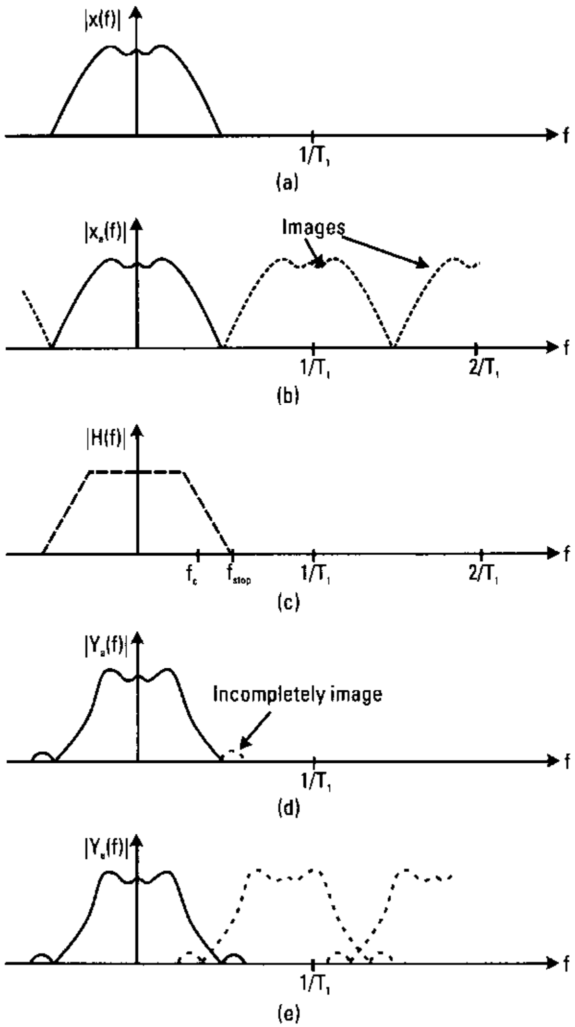


Figure 8.12 Transmission of frequency spectrum during sampling rate conversion: (a) before sampling; (b) after sampling; (c) transfer function of reconstruction filter; (d) after reconstruction; and (e) after resampling.

The method is performed on the baseband unit after sampling the received signal with a fixed noninteger time clock rate for the symbol rate. As shown in Figure 8.10, when the received data is sampled at noninteger times sampling rate for the symbol rate, the sampling interval of optimum symbol timing [$x(0)$, $x(1)$, ..., $x(i)$, ...], which shows the optimum sample timing closest to each symbol timing, can be found from (8.3) and (8.4):

$$x(i) = \text{round}\left(\frac{C}{S} + O(i-1)\right) \tag{8.3}$$

and

$$\begin{aligned} O(i) &= \frac{C}{S} + O(i-1) - \text{round}\left(\frac{C}{S} + O(i-1)\right) \\ &= \frac{C}{S} + O(i-1) - x(i) \qquad (i \geq 1) \end{aligned} \tag{8.4}$$

where $O(i)$, C, S, and round (X) show the sampling offset of optimum sampling timing for each symbol timing, the sampling rate, the symbol/chip rate, and rounds for the elements of X to the nearest integer, respectively. $O(0)$ is the sampling offset for the first symbol timing. From (8.3) and (8.4), the value of $x(i)$ is limited to two cases:

$$x(i) = \begin{cases} A = \text{round}(C/S) \\ B = \begin{cases} A + 1 : C/S \text{ is larger than round}(C/S) \\ A - 1 : C/S \text{ is smaller than round}(C/S) \end{cases} \end{cases} \tag{8.5}$$

The proof is shown in [22]. Hence, to search the symbol timing caused by the cycle slip shown in Figure 8.10, one needs to know all patterns of the two values (A and B), and to search for the suitable patterns having the sampling interval of the optimum symbol timing. However, the total number of the state patterns increases with the square of the number of observed symbols. This makes for heavy loads on the baseband unit, and thus the total number of state patterns should be reduced. The state patterns have three important features:

1. The maximum number of repetitions of A in a state pattern, M, is the maximum integer smaller than $1/|T|$, where T denotes the difference between the number of oversamples and the rounded value of the number of oversamples to the nearest integer [= C/S − round(C/S)].
2. The minimum number of repetitions of A, N, is smaller than M by 1, namely, $N = M - 1$.
3. B is never repeated in a state pattern.

The proof is also in [22]. By utilizing these features, total number of state patterns can be drastically reduced.

Figure 8.13 shows the state pattern generator. The information of target communication system is given to calculate *A, B, M,* and *N.* The given information is the sampling rate, symbol/chip rate, and the number of observation symbols. The results, *A, B, M,* and *N,* are fed into the state pattern generator. Figure 8.14 shows the flowchart of state pattern generator. The conditions utilized in this generator are summarized as follows:

- The repeat of repetition *A* is no more than *M.*
- *B*'s neighbor is always *A.*
- The number of repetitions of *A* between two *B*s is more than *N.*

These conditions include all the conditions (–0.5 < $O(0)$ < 0.5) for the first sampling offset $O(0)$ in the initial acquisition. By following the procedure in the flowchart, the number of state patterns can be reduced.

The symbol timing synchronization method is based on the state patterns generated by the procedure described. Figure 8.15 shows its flow. The received signal is filtered by the pulse-shaping filter. The synchronization timing estimation block uses a maximum likelihood estimation to determine an adequate state pattern having the maximum of signal power. The sampling data, which the selected state pattern shows, is output as the optimum symbol timing to each demodulation block. This synchronization method adaptively searches the optimum symbol timing for the received data sampled at *integer or noninteger times*[3] sampling rate for the symbol rate.

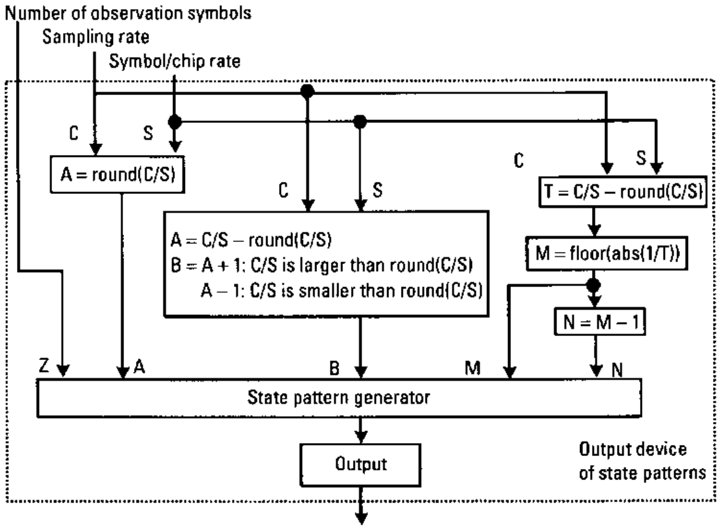


Figure 8.13 State pattern generator.

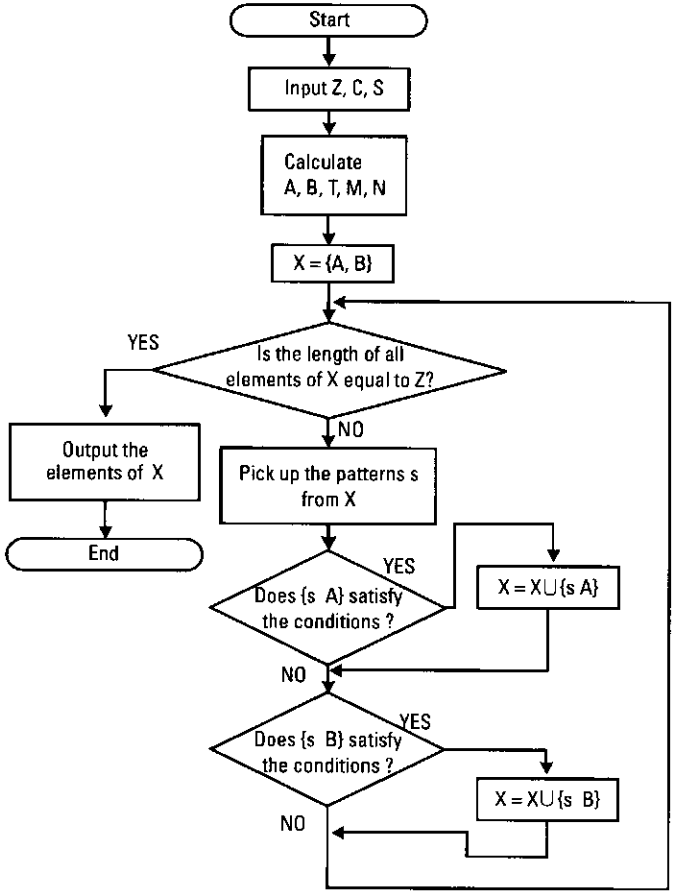

Start
Input Z, C, S
Calculate A, B, T, M, N
X = {A, B}
YES
Is the length of all elements of X equal to Z?
NO
Output the elements of X
End
Pick up the patterns s from X
YES
Does {s A} satisfy the conditions ?
X = XU{s A}
NO
YES
Does {s B} satisfy the conditions ?
X = XU{s B}
NO

Figure 8.14 Flowchart for state pattern generator.

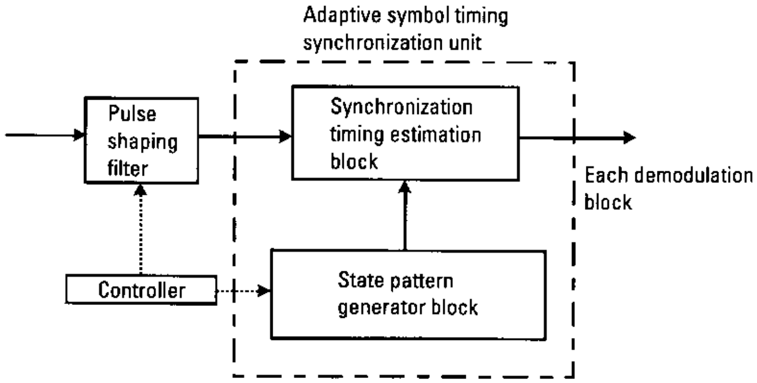

Adaptive symbol timing synchronization unit
Pulse shaping filter
Synchronization timing estimation block
Each demodulation block
Controller
State pattern generator block

Figure 8.15 Flow of signal synchronization processing.

As an example of generating a state pattern, consider three systems, GPS, *electric toll collection* (ETC), and PHS, which are integrated as a multimode and multiservice system for future intelligent transport systems. Table 8.1 shows the parameters of each system.

The state patterns generated for ETC and PHS signals when the sampling rate is 32 times the clock rate of the GPS system will be shown as follows. Because the chip rate of the GPS system is the highest symbol/chip rate of the three systems and the number of observation symbols is 10, the state pattern characteristics for the ETC signal are:

- S (symbol rate) = 1,000,000
- C (sampling rate) = 32,736,000
- A (sampling interval 1) = 33
- B (sampling interval 2) = 32
- T (sampling offset) = −0.2640
- M (maximum repeat number of A) = 3
- N (minimum repeat number of A) = 2
- Z (number of observation symbols) = 10

The state pattern is:

```
 1.  A  A  A  B  A  A  A  B  A  A
 2.  A  A  A  B  A  A  B  A  A  A
 3.  A  A  A  B  A  A  B  A  A  B
 4.  A  A  B  A  A  A  B  A  A  A
 5.  A  A  B  A  A  A  B  A  A  B
 6.  A  A  B  A  A  B  A  A  A  B
 7.  A  A  B  A  A  B  A  A  B  A
 8.  A  B  A  A  A  B  A  A  A  B
 9.  A  B  A  A  A  B  A  A  B  A
10.  A  B  A  A  B  A  A  A  B  A
11.  A  B  A  A  B  A  A  B  A  A
12.  B  A  A  A  B  A  A  A  B  A
13.  B  A  A  A  B  A  A  B  A  A
14.  B  A  A  B  A  A  A  B  A  A
15.  B  A  A  B  A  A  B  A  A  A
16.  B  A  A  B  A  A  B  A  A  B
```

The state pattern characteristics for the PHS signal are:

- S (symbol rate) = 192,000
- C (sampling rate) = 3,2736,000/5 (after decimation of every 5 samples)
- A (sampling interval 1) = 34
- B (sampling interval 2) = 35
- T (sampling offset) = 0.1000 … 0.1
- M (maximum repeat number of A) = 9
- N (minimum repeat number of A) = 8
- Z (number of observation symbols) = 10

State pattern is shown as follows:

```
 1.  B  A  A  A  A  A  A  A  A  B
 2.  B  A  A  A  A  A  A  A  A  A
 3.  A  B  A  A  A  A  A  A  A  A
 4.  A  A  B  A  A  A  A  A  A  A
 5.  A  A  A  B  A  A  A  A  A  A
 6.  A  A  A  A  B  A  A  A  A  A
 7.  A  A  A  A  A  B  A  A  A  A
 8.  A  A  A  A  A  A  B  A  A  A
 9.  A  A  A  A  A  A  A  B  A  A
10.  A  A  A  A  A  A  A  A  B  A
11.  A  A  A  A  A  A  A  A  A  B
```

Table 8.1
Simulation Models

System	Frequency	Modulation	Data Rate	Bandwidth
PHS	1.9-GHz band	π/4DQPSK TDMA-TDD	384 Kbps	300 kHz/1 channel
GPS	1.5-GHz band	BPSK+ DS-SS	50 bps	1.023 MHz
ETC	5.8-GHz	ASK (Manchester code)	1,000 Kbps	Less than 8 MHz

These results demonstrate that the total number of state patterns for the ETC and PHS could be reduced from 2^{10} patterns to 16 and 11 patterns, respectively. The fewer patterns combined with the symbol timing synchronization method based on maximum likelihood estimation mean that the optimum symbol timing can be obtained under a fixed sampling rate.

8.5.4 DSPH Architecture

In the ITS in which many systems intermingle, a *multimode software radio communication* (MSR) system using software radio technology can make efficient use of one terminal. The basic system configuration is the full-download type (Figure 8.3).

As of this writing, MSR research on individual software components and prototypes is progressing (see Section 8.6). However, software radio has been difficult to implement in the case of simultaneous use of more than one communication system, for example, the ITS. Table 8.2 lists the ITS communication services intended for use in cars in Japan. The services can be divided into three categories [23]:

1. Services that are always necessary when driving a car;
2. Services that are absolutely necessary at all times;
3. Services that are chosen by the user.

Priority is given to the systems shown in Table 8.2 by using the above classification scheme. A software radio system for ITS may be able to clarify the needs of users accessing several services simultaneously. However, as of this writing, research and development efforts have focused on users selecting only one of several software radio communication systems. Future ITS systems will provide more than one service and thus be a *multimode and multiservice software radio communication system* (MMSR) [13].

The MMSR system has four subsystems [13]. The signal processing is done under the condition that the signals of each subsystem of the MMSR system are digital.

1. *Subsystem 1:* The DSPH of the BBU performs digital signal processing for the manifold systems simultaneously by installing hardware components for the software radio communication system.
2. *Subsystem 2:* DSPH of the BBU perform digital signal processing for the manifold systems simultaneously by TDM or TDMA.

Table 8.2
Communication Systems Used in the ITS System

Category	Control			Broadcasting			
System	GPS	VICS	ETC	Radio Broadcasting		TV	Satellite
Frequency	1.5 GHz	FM band	5.8 GHz	0.5–1.6 MHz	76–90 MHz	90–222, 470–770 MHz	11.7–12.0 MHz
Modulation	BPSK + DS-SS	LMSK	ASK	AM	FM	VSB-AM FM	FM PSK
Data rate	50 bps	16 Kbps	1,024 Kbps	Analog	Analog	Analog	Analog
Bandwidth	1.023 MHz	100 kHz	8 MHz	15 kHz/ channel	200 kHz	6 MHz	27 MHz
Priority	1	3	3	3	3	2	2

3. *Subsystem 3:* The method in which DSPH of the BBU perform digital signal processing for the manifold systems simultaneously by code division multiplexing or code division multiple access.
4. *Subsystem 4:* The DSPH of the BBU performs digital signal processing for the manifold systems simultaneously by the random access protocol.

The transmitter and receiver of (1) are shown in Figures 8.16(a) and 8.17(a), respectively. There are several DSPHs for which the particular demand of the user is transmitted from the outside controller. Several DSPHs can be integrated into one DSPH that is also controlled from an outside controller [13]. Since each digital signal processing flow is separated, this MMSR system does not cause interference between services. However, several DSPHs must be prepared, and there is redundant signal processing. As a result, the scale of the hardware may increase with the number of integrated systems.

The transmitter and receiver of (2) are shown in Figures 8.16(b) and 8.17(b), respectively. One DSPH does the signal processing for the communication services by time division–based multiple access to eliminate the redundant signal processing, which is a problem in subsystem (1). There are two time-division–based multiple access methods: TDM, which allocates a fixed time slot to the signal processing for the specified communication services, and TDMA, which allocates a time slot to the signal processing for communication services on demand.

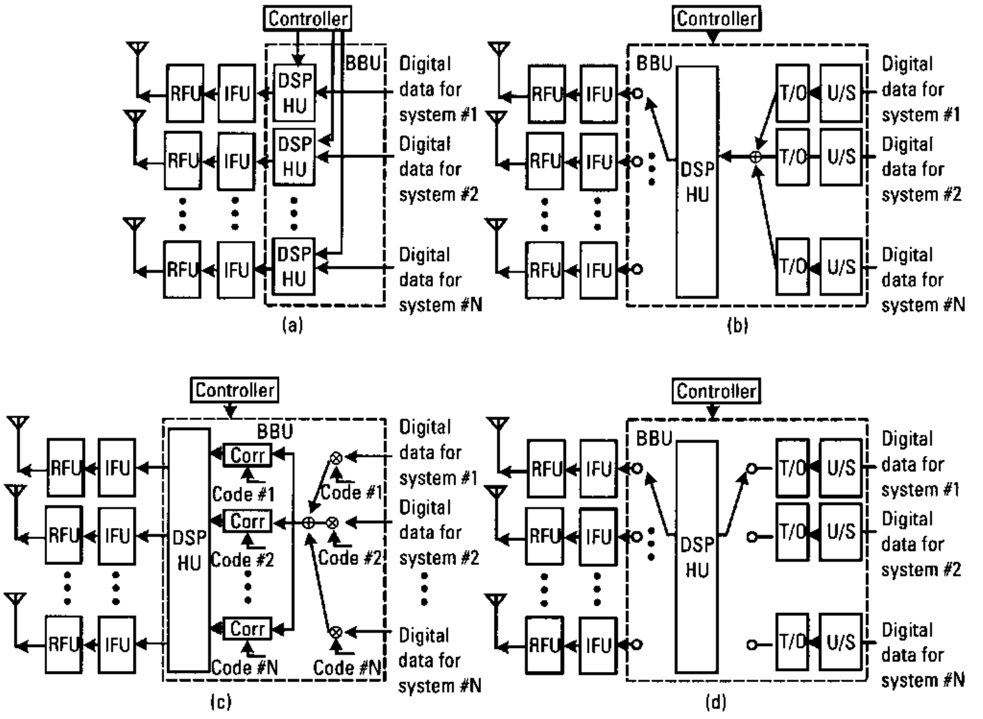

Controller
BBU
RFU
IFU
DSP HU
Digital data for system #1
Digital data for system #2
Digital data for system #N
(a)
T/O
U/S
(b)
Corr
Code #1
Code #2
Code #N
(c)
(d)

Figure 8.16 Configuration of MMSR system's transmitter (T/O: time offset, corr: correlator): (a) separate type, (b) TDM- or TDMA-based, (c) CDM- or CDMA-based, and (d) random access type.

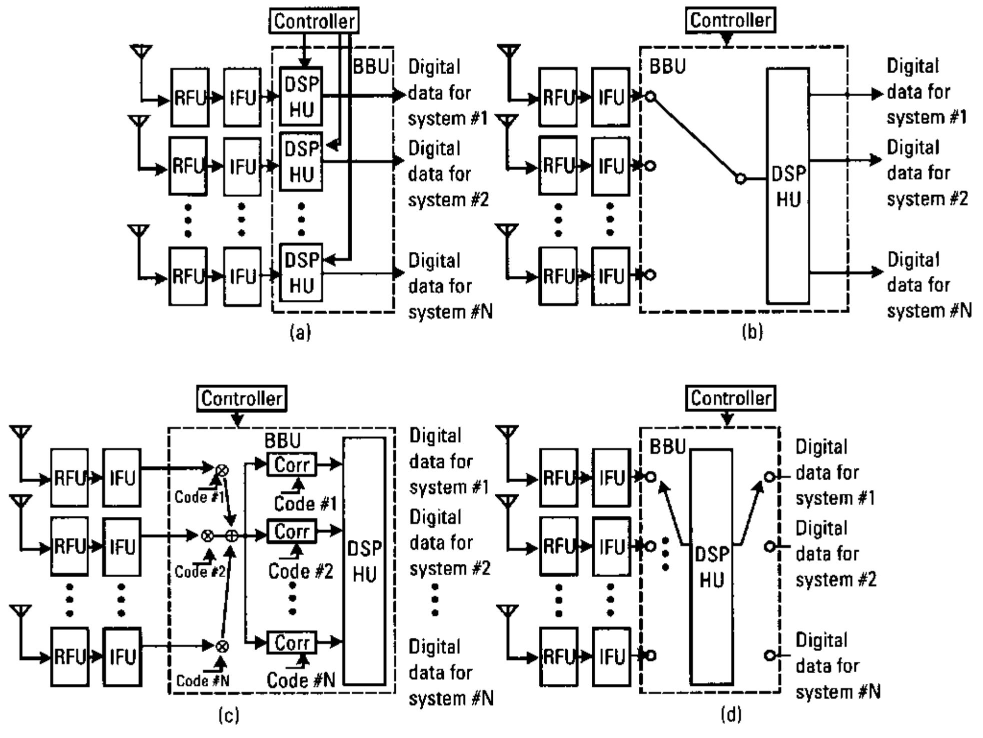

Controller
BBU
RFU
IFU
DSP HU
Digital data for system #1
Digital data for system #2
Digital data for system #N
(a)
(b)
Corr
Code #1
Code #2
Code #N
(c)
(d)

Figure 8.17 Configuration of the MMSR system's receiver (T/O: time offset, corr: correlator) for multimode and multiservice software radio system: (a) separate type, (b) TDM- or TDMA-based with multisampler based common A/D, (c) CDM- or CDMA-based with separated A/D, and (d) random access type.

TDM can realize an efficient MMSR system if the slot timing is fixed. However, the number of time slots for digital signal processing increases as the number of integrated systems increases. Moreover, if we allocate one time slot to a communication service with priority (3), the time slot is mostly unused. TDMA does not have this problem but needs management techniques such as time-domain interruption and so on, which have not been studied in depth.

The transmitter and receiver of (3) are shown in Figures 8.16(c) and 8.17(c), respectively. To eliminate the redundant signal processing of the DSPHs of (a) and to avoid the synchronous signal processing of the TDM or TDMA technique of (2), the code is superposed for the digital data of each system, which is obtained before D/A conversion in the TX module of the BBU and after A/D conversion in the BBU of the RX module. The superposed digital data is multiplexed using CDM. To request a specific service, the user obtains the information data of this service by using its spreading code. The code used in this case could be either a cryptogram or a spreading code. The merit of this system is that the code determines the type of service. However, the superposition of the code means that the clock speed increases. The system, therefore, needs a high-speed digital signal processor.

The transmitter and receiver of (4) are shown in Figures 8.16(d) and 8.17(d), respectively. This configuration is basically an application (2), where a random access protocol is used instead of TDMA and TDM. Moreover, except for the MMSR system (1–4), we can utilize hybrid systems, in which system (2) is adopted for the configuration of transmitter and system (3) is adopted for the configuration of receiver.

The MMSR is useful for the integration of "multiple systems." However, as for the software radio technology, we can make it more efficient by using a new concept for the configuration of BBU.

8.5.5 Parameter-Controlled Software Radio Technology

In the conventional full-download-type software radio technology, the users themselves can change the system configuration. However, the following problems arise (see Section 8.3):

- The volume of software downloaded into the DSPH increases as the contents of the required telecommunications component blocks become more complicated.
- The period for the configuration check of the DSPH also increases as the contents of the required telecommunications component blocks become more complicated.

- In the download software, there are often several component blocks that are related to the know-how of the manufacturer (e.g., the optimization method or calculation algorithm for some special blocks). This knowledge may leak out when the software is downloaded. In addition, there is the possibility of alterations being made by other people.

The software radio technology of [11, 12] is related only to the BBU of the TX and RX modules. The RFU and IFU are prepared individually or as an integrated RFU and IFU. The BBU is shown in Figure 8.18. It contains basic telecommunications component blocks like the encoder, frame, modulator, filter blocks of transmitter and equalizer, detector, and decoder blocks of receiver that have already been implemented in the DSPH. The functions of the telecommunication blocks are programmable and can be easily changed by downloading the external parameters. To change the configuration of the digital filter of the transmitter, for example, the user sends only the coefficient of the required digital filter to the filter block of the BBU. The use of external parameters lets the BBU become a general-purpose trans- mitter and receiver, and the resulting software radio system is called a parameter-controlled software radio system. Figure 8.2 shows the configuration of BBU and the parameters required for parameter-controlled software radio.

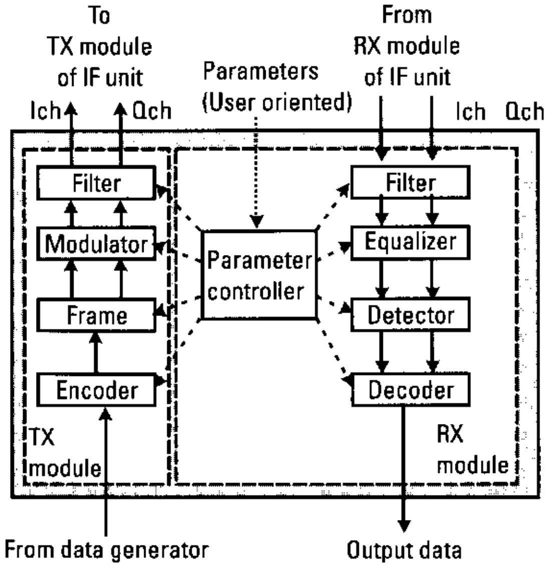


Figure 8.18 Configuration of BBU in a parameter-controlled software radio technology.

The proposed system is realized by downloading external parameters, which are small and important pieces of information for the realization of a particular telecommunication system. Therefore, important blocks must be configured in the DSPH in advance. Stable performance can be obtained, and the configuration check period can be reduced in comparison with full-download-type software radio technology. Moreover, manufacturer-specific know-how, such as the optimization methods for the specified telecommunications components, never leak out, because only a small volume of software is downloaded. Moreover, since the volume of download software to the DSPH is quite small, several strong coding methods can be used to download the software. Consequently, the software radio communication system is secure from copying or tampering.

An experimental prototype of the parameter-controlled type has been developed and its transmission performance evaluated [12, 23]. Figures 8.19 and 8.20 show the system configuration of the experimental prototype and its appearance. The system parameters are summarized in Table 8.3. The experimental prototype can make use of three real telecommunications systems as service models for ITS, PHS, GPS, and ETC systems. MMSR allows for simultaneous services (i.e., PHS and ETC or GPS and ETC). This is the first prototype of an MMSR system for ITS in the world [12, 23]. Moreover, in user mode, the user can freely choose between modulation schemes of GMSK, $\pi/4$QPSK, BPSK and QPSK. The PHS and GPS systems use an integrated antenna because the frequency utilized in GPS (1.5-GHz band) is quite similar to PHS (1.9-GHz band). The external parameters are supplied from a notebook-type computer loaded with management software connected to the experimental prototype by a 10Base-T Ethernet cable.

To realize the PHS system in Figure 8.19, its external parameters are sent through the 10Base-T Ethernet cable from a notebook-type computer and stored in the CPU unit. The CPU unit then gives the parameters to the BBU, IFU, and RFU. A similar technique can be utilized for the GPS and ETC systems.

The following compares full-download-type software radio with parameter-controlled software radio, the download program volume and length of download time. The controlled parameter uses the FPGA by the Altera Corporation in the conversion, FPGA units, and DSP. As the first results, the volume of software that can be utilized in the conversion, FPGA and DSP units. For the FPGA-based units, the software volume is given as the required number of gates (gate) as in Table 8.4. For the DSP-based unit, the software volume is given as the volume of programmed code (byte) as in Table 8.5. Table 8.6 shows the average configuration time in the user mode for the parameter-controlled software radio system.

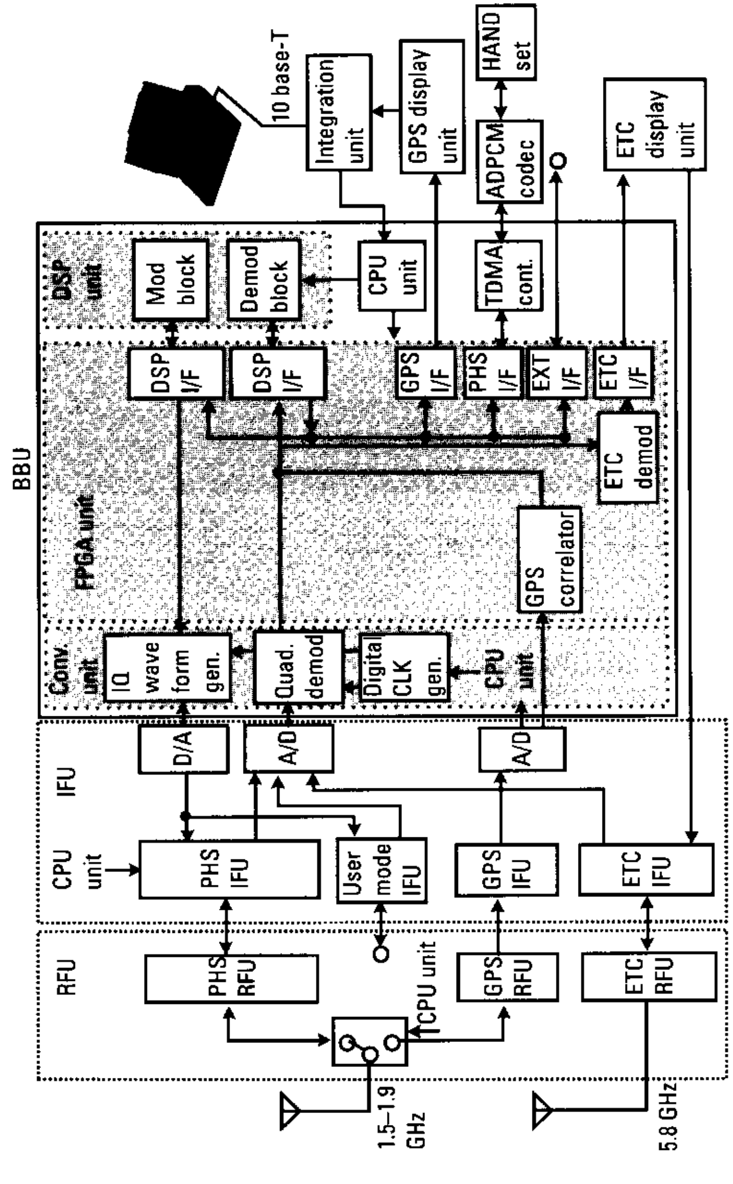


Figure 8.19 System configuration of the experimental prototype.

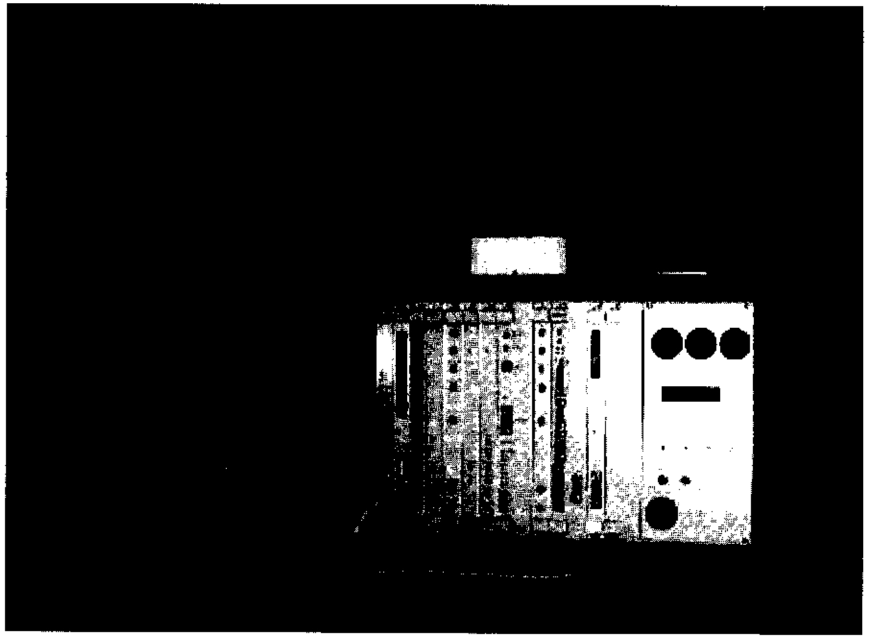


Figure 8.20 Appearance of the experimental prototype.

Table 8.3
System Parameters of the Experimental Prototype

Services	Modulation	Frequency	Data Rate
(Service mode) PHS, ETC, GPS (User mode) BPSK, QPSK, $\pi/4$ QPSK,GMSK	PHS—$\pi/4$ DQPSK ETC— ASK GPS—BPSK+ SS User Mode—BPSK, QPSK, $\pi/4$ QPSK, GMSK	PHS—1.9 GHz ETC—5.8 GHz GPS—1.5 GHz User Mode—IF band	PHS—384 Kbps (carrier bit rate) ETC—1,024 Kbps: 2,048 kbaud (data rate) GPS—50 bps (data rate): 1.023 Mcps (chip rate) User mode BPSK, QPSK $\pi/4$ QPSK—384 Kbps (carrier bit rate) GMSK—270.833 Kbps (carrier bit rate)

Table 8.4 shows the required number of gates for FPGAs of each function block and hardware device. The marked data (*) is shared by several systems. For example, the TX filter block is shared with the user mode and PHS mode, the size of the FPGA is 1,013 gates, and it is programmed in EPM7064(1).

Table 8.4
Volume of Software for FPGA-Based Units in Experimental Prototype

Unit	**Conv. Unit (gate)**		**FPGA Unit (gate)**						**FPGA Total Gate**
Device (Number)	EPM 7064 (1)	EPM 9400 (1)	EPM 7064 (1)	EPF 10K20 (1)	EPM 7128 (1)	EPM 7128 (2)	EPM 9400 (2)	EPM 7192 (1)	EPF 10K250 (1)(2)(3)	
Max. Gate	1,250	8,000	1,250	20,000	2,500	2,500	8,000	3,750	250,000 × 3	
User Mode	*1,013	*2,176			*725	*826	*2,520	*776		8,036
PHS	*1,013	*2,176	113	7,200	*725	*413	*2,520	*388		14,548
GPS							560	86	547,500	548,146
ETC		1,088								1,088
Total	1,013	3,264	113	7,200	725	826	3,080	862	547,500	564,583
Purpose	TX Filter (1013 shared)	RX Filter (2176, shared) ETC Demod +ETC I/F(1088)	PHS/IF	(7313)	DSP I/F TX (725 shared)	DSP I/F TX (413 shared) EXT I/F (413)	DSP I/F RX (2,520 shared) GPS I/F (560)	DSP I/F RX (388 shared) EXT I/F (388) GPS I/F (86)	GPS Correlator	

*Means the software which is shared for the realization of some systems.

Table 8.5
Volume of Software for Each System in the Experimental Prototype

Device	Conv. & FPGA Unit (gate)	DSP Unit (byte)			Parameter (byte)
		TX	RX	Total	
User Mode	8036	13357*	40509*	53866	1660
PHS	14548	9586*	13340*	22926	1616
GPS	548146	Not used	13340*	13340	4
ETC	1088	Not used	Not used	0	570
Total	564583	13857	40509	53866	

*Means the software which is shared for the realization of some systems.

Table 8.6
Average Configuration Time in User Mode for Parameter-Controlled Software Radio System

BPSK –> QPSK	37.5 ms	BPSK –> GMSK	30.5 ms	QPSK π/4 –> QPSK	30 ms
QPSK –> BPSK	32.5 ms	GMSK –> BPSK	35.25 ms	π/4 QPSK –> QPSK	35 ms
BPSK –> π/4 QPSK	32.5 ms	QPSK –> GMSK	30 ms	π/4 QPSK –> GMSK	30 ms
π/4 QPSK –> BPSK	35 ms	GMSK –> QPSK	40.5 ms	GMSK –> π/4 QPSK	35 ms
Full download user mode		3,410 ms			

Full-download-type software radio requires 8,036, 14,548, 548,146 and 1,088 gates user mode, PHS, GPS, and ETC systems, respectively, whereas the prototype requires 564,583 gates to be preprogrammed in the FPGA based units. The volume of software for the DSP unit in the prototype and in individual user mode, PHS, GPS and ETC systems can be investigated with the help of volumes in Table 8.5. The volumes of parameters are shown for the prototype. If we assume that only the software for DSP unit is downloaded, the prototype downloads only about 1/15 (for PHS) and 1/30 (for user mode) in comparison with the case that all DSP software is downloaded to the DSP unit for the individual realization of each system. Moreover, Table 8.5 reveals that 22,926 bytes of DSP software are needed in order to realize the PHS system in full-download-type software radio system, whereas the prototype uses 53,866 bytes to realize all systems. The prototype does use redundant programs for some systems, but this redundancy reduces the software download time and provides stable performance when reconfiguring the systems.

Comparing the required scale of DSPH for the realization of both software radio systems (see Table 8.5), the prototype needs 564,583 gates, whereas the full-download-type needs the maximum number of gates for each system (for GPS, this amounts to 548,146 gates). In other words, the volume of FPGAs in the prototype is 1.02 times that of the full-download-type. On the other hand, as for DSP unit, programs for user mode are needed in the prototype, because the other systems utilize some parts of these programs. Therefore, both software radio systems need to have DSP components that can include the user mode programs.

Table 8.6 compares the download time for user mode. We assume GMSK, π/4QPSK, BPSK, or QPSK can be utilized. The average download time of a prototype is only about 1/100 that of the full-download-type type. In addition, the average download time for the realization of user mode is independent of modulation scheme and almost the same as that of the parameter-controlled software radio system. This is because all modulation schemes of user mode use similar download software.

The BER performance of the experimental prototype in the case of user mode is shown in Figure 8.21. Theoretical BER value versus E_b/N_0 of all systems is included. In this case, since coherent detection for all systems is used, BPSK, QPSK and π/4 QPSK have equivalent BER versus E_b/N_0. It is clear that the experimental prototype's BER performance was 1 dB of the theoretical BER performance for all modulation schemes.

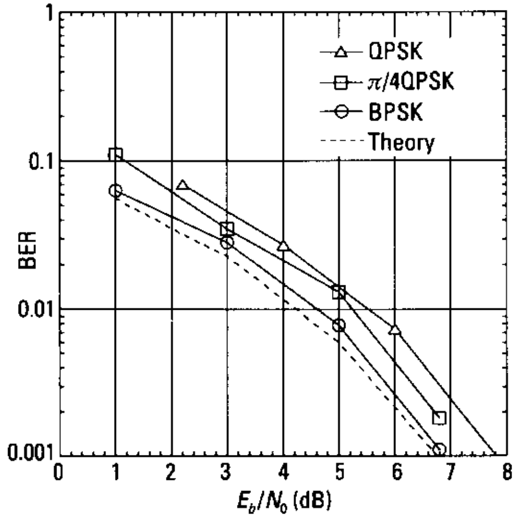


Figure 8.21 BER performance of the experimental prototype in the case of user mode.

8.5.6 Simultaneous Signal-Processing Method for Several Systems

The common sampling processing method, which distributes a system clock to all the target systems equally by a sampler, is effective to realize a scale reduction of the MMSR system. However, the method should be a redundant sampling processing for systems with low symbol rates when the difference in the target system's symbol rate is quite large, since the system clock speed must be enough for the maximum symbol/chip rate in all target systems.

To avoid this problem, a common sampling processing has been proposed [24]. Here, the system clock is determined according to the target system's clock speed and is distributed to all the target systems. This method should be able to reduce the system clock when the target system's clock speeds are quite different from each other. Figure 8.22(a) illustrates an example of the proposed sampling procedure for the two systems (A and B) whose clocks (S_a and S_b) are quite different from each other ($S_a >> S_b$). The sampling clock allocated for A is suspended during a constant interval, and, during this time, another sampling for B is performed, as shown in Figure 8.22(b). System A is then interpolated the value of 0 as the sampling data. This procedure results in an efficient common sampling processing, even if the difference in the target system's clock speed is quite large. Now let us consider a modification to this method that is good for several systems. Figure 8.23 illustrates the flow of the common sampling method in the MMSR system. The user's request for communication systems is sent to the block, which generates a list of the target systems. The number of target services N; the target service's symbol/chip rate $J\{\ni(S_1, S_2, \ldots, S_i, \ldots, S_N)\}$, which have been arranged in decreasing order; the minimum suspended interval $T_{\min}\{(T_{\min(1)}, T_{\min(2)}, \ldots, T_{\min(i)}, \ldots, T_{\min(N)})\}$ of the sampling processing for each target system; and the necessary number of oversamples P [samples/symbol] for the system of maximum symbol rates in all the target systems are included in the lists of the target systems. These lists are then fed into the system clock allocation unit. The configuration is shown in Figure 8.24, and the system clock allocation list L for all the target systems is generated by the system clock allocation algorithm. In addition, the system clock is determined by N_L, which denotes the total number of the systems directly allocated to the system clock. Finally, each sampling clock for all the target systems are generated by the sampling clock generator, and each piece of received data is digitally oversampled by a multisampler, and sent to the digital processing parts.

Next, let us describe the system clock allocation algorithm for generating L. Here, we will assume that master-system denotes the system that can cover

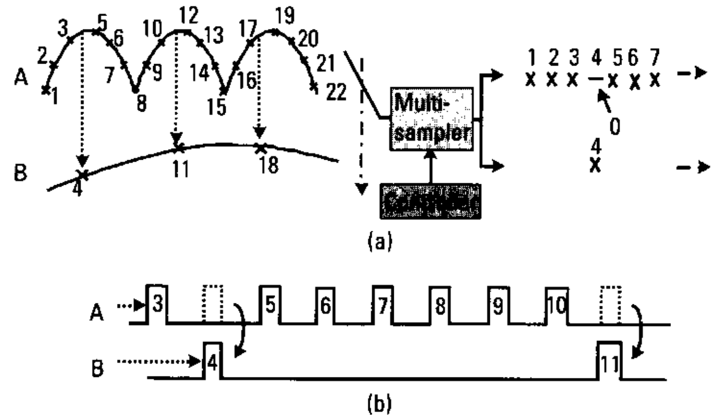


Figure 8.22 (a) Example of the proposed sampling method and (b) sampling clock used in systems A and B.

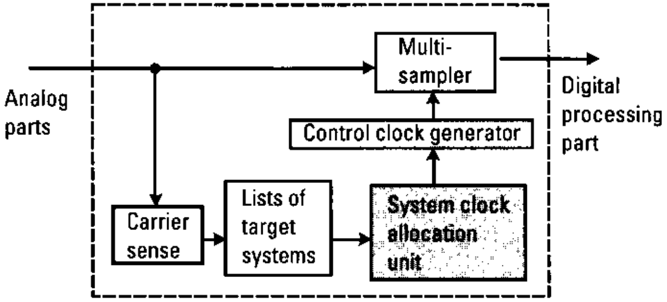


Figure 8.23 Flow of common sampling processing method.

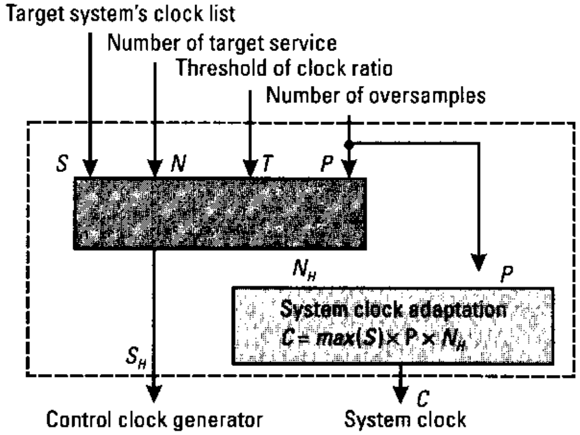


Figure 8.24 System clock allocation unit.

the sampling processing time of a lower symbol/chip rate's system during the minimum suspended interval of sampling processing and that slave-system denotes the system that robs the processing time of the master-system.

In this algorithm, the higher symbol/chip rate $S_j \{j = i - 1, \ldots, 1\}$ for $S_i\{i = N, N - 1, \ldots, 2\}$ in J is searched by order of the following allocation methods.

I. *The processing time of a master-system covers the processing time of a slave-system.*

To utilize method (I) without affecting the later processing of the master-system Π_j, enough sampling clock for the slave-system Π_i should be given within the minimum suspended interval $T_{min(j)}$ of the master-system Π_j, where Π_x denotes the system name according to the xth symbol/chip rate S_x in J. This means that the ratio of the symbol/chip rate (S_j) of the master-system Π_j and the symbol/chip rate (S_i) of the slave-system Π_i must be larger than $T_{min(j)}$. Therefore, the estimation, for whether this method (I) can be utilized or not, is:

$$\left({}^{S_j}\!/_{S_i} \right) \geq T_{\min(j)} \tag{8.6}$$

The system clock allocation list L is then generated as follows:

$$L = \begin{Bmatrix} \Pi_j \\ \Pi_i\left(\Pi_j\right) \end{Bmatrix}$$

where each rank of L, and $\Pi_i(\Pi_j)$ denote the class between master-system and slave-system, and the system Π_i is the slave of the system Π_j, respectively. Specifically, the system in the first rank of L shows the system directly allocated the system clock. Hence, the system clock C [samples/second] needs to be at least as follows:

$$C\,[\text{samples/second}] \geq S_j[\text{symbols/second}] \times P\,[\text{samples/symbol}]$$

An example utilizing the system (I) for three systems, whose symbol rates (S_a, S_b, and S_c) are quite different from each other ($S_a >> S_b >> S_c$) is shown in Figure 8.25.

II. *The processing time of a master-system covers the processing time of several slave-systems.*

To utilize method (II) without affecting the latėr processing of the master-system Π_j, enough processing time for all slave-systems (Π_i, ...,

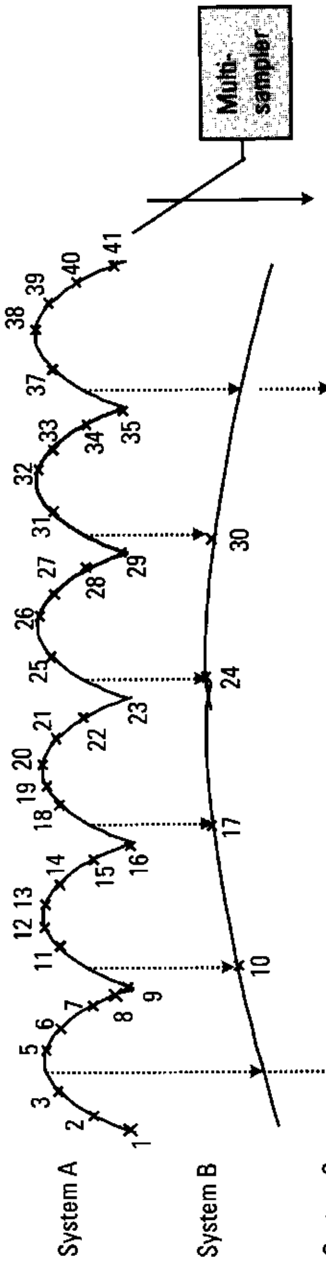


Figure 8.25 Example of (I) ($S_a >> S_b >> S_c$).

Π_k, …, Π_l) should be given within the minimum suspended interval $T_{\min(j)}$ of master-system Π_j. In this case, the proposed method utilizes the method that a master-system distributes the processing time, which is determined by the maximum symbol/chip rate (S_i) in the slave-systems uniformly for all the slave-systems. Therefore, the estimation, whether this method (II) can be utilized or not, is written as follows:

$$\left(S_j \middle/ (S_i \times n) \right) \geq T_{\min(j)} \tag{8.7}$$

where $n(= l - i + 1)$ denotes the number of the slave-systems. Then, the system clock allocation list L is generated as follows:

$$L = \begin{Bmatrix} \Pi_j & \\ \Pi_i(\Pi_j), \ldots, \Pi_k(\Pi_j), \ldots, \Pi_l(\Pi_j) \end{Bmatrix}$$

where the information of the second rank denotes that several slave-systems exist for a master-system. In this case, the system clock C[samples/second] is determined on just the basis of the symbol rate S_j by the same calculation as the case of (I). An example utilizing the method (II) for three systems is shown in Figure 8.26.

III. *The processing time of several master-systems cover the processing time of a slave-system.*

To utilize the method (III) without affecting the later processing of all the master-systems (Π_l, …, Π_m, …, Π_j), the proposed method utilizes the method that each master-system (Π_l, …, Π_M, …, Π_j) uniformly covers the necessary processing time for the slave-system Πj. Therefore, the minimum symbol/chip rate's system Π_i in all the master-systems (Π_l, …, Π_m, …, Π_j) must then cover the processing time, which covers equally with the other master-systems during the minimum suspended interval $T_{\min(j)}$. Hence, the estimation, whether this method (III) can be utilized or not, is written as follows:

$$\left((S_j \times t) \middle/ S_i \right) \geq T_{\min(j)} \tag{8.8}$$

where $t(= j - l + 1)$ denotes the total number of the master-systems. Then, the system clock allocation list L is generated as follows:

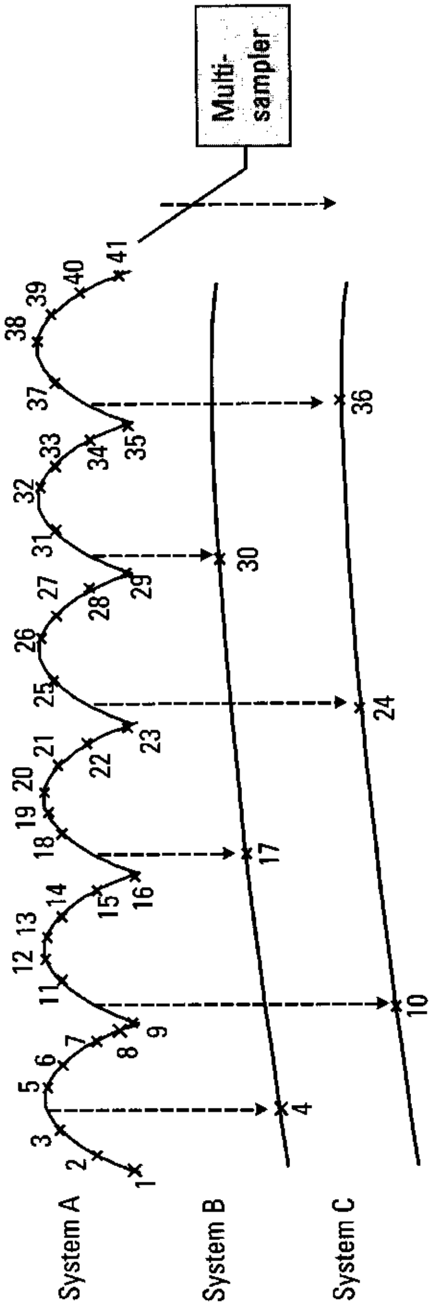


Figure 8.26 Example of (II) ($S_a > S_b$, S_c).

$$L = \left\{ \begin{matrix} \Pi_l, \ldots, \Pi_m, \ldots, \Pi_j \\ \Pi_i \left(\Pi_l, \ldots, \Pi_m, \ldots, \Pi_j \right) \end{matrix} \right\}$$

where the information of the second rank denotes that several master-systems exist for a slave-system. In this case, the system clock C[sample/second] is determined on the maximum symbol/chip rate S_l in all the master-systems and is distributed uniformly for all the master-systems. Hence, the system clock C[samples/second] needs to be at least as follows:

$$C\text{[samples/second]} \geq S_l\text{[symbols/second]} \times t(= N_L) \times P\text{[samples/symbol]}$$

An example utilizing the method (III) for three systems is shown in Figure 8.27.

With an eye on the development of an integrated receiver for a future ITS services with GPS, ETC, PHS, VICS, and PDC, the effectiveness of the proposed method has been evaluated in terms of BER performance in a AWGN and one-path Rayleigh fading environment by computer simulation [24]. The proposed method was shown to be sufficiently robust under AWGN and Rayleigh fading environments.

8.5.7 Software Radio Devices

This section describes three key devices to configure software radio: ADC, the DSP, and the FPGA.

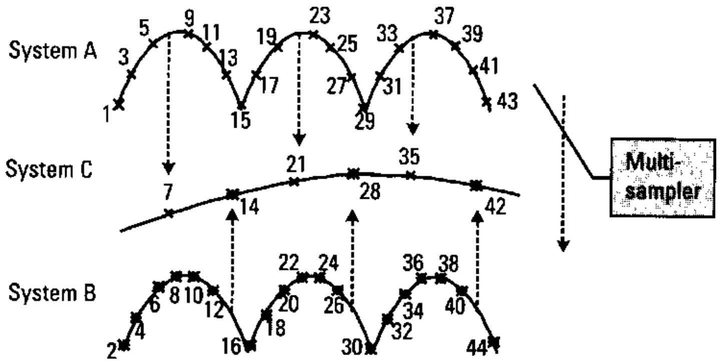


Figure 8.27 Example of (III) (S_a, $S_b >> S_c$).

8.5.7.1 Analog-to-Digital Converter (ADC)

The ADC is a gateway between the analog signal and digital signal. Users always favor ADCs with ever higher speeds, higher resolutions, and cheaper prices. Table 8.7 shows ADCs that have sampling rates over 100 MHz. Sampling rates continue to increase. Figure 8.28 shows this trend [25]. After 2000, ADCs are expected to be high-speed and high-resolution ADC.

8.5.7.2 Digital Signal Processor (DSP)

DSP is a CPU that can accelerate the multiply and accumulate calculations (MACs). Its main advantage relates to software programming. To configure the DSP, a high-level computer language such as C and assembly language can be used. It is possible to reconfigure the software by reading the program at a memory address, in which the required software must be stored.

However, the DSP has a problem regarding its processing speed. Because DSP program is usually sequential and cannot be paralleled, the processing speed depends on the architecture of the DSP. Figure 8.29 shows the expected trend in DSP technology. If the DSP's processing speed is more than 14 Gflops, it can be used for real-time mobile communication. After 2000, a real-time mobile communication terminal based on a DSP circuit may become possible.

Table 8.7
List of Remarkable ADCs

Name	Manufacturer	Resolution (bit)	Sampling Rate (MHz)
AD9433	Analog Devices	12	105
AD 9214	Analog Devices	10	208
AD 9410	Analog Devices	10	105
AD 9432	Analog Devices	12	200
AD 9054A	Analog Devices	8	100
AD 9288	Analog Devices	8	100
AD 9071	Analog Devices	10	100
TDC1029	TRW	6	100
SPT7760A	Sig Proc Tech	8	1,000
HADC77100	Sig Proc Tech	8	150
CXA1176K	Sony	8	250
CXA1276K	Sony	8	400

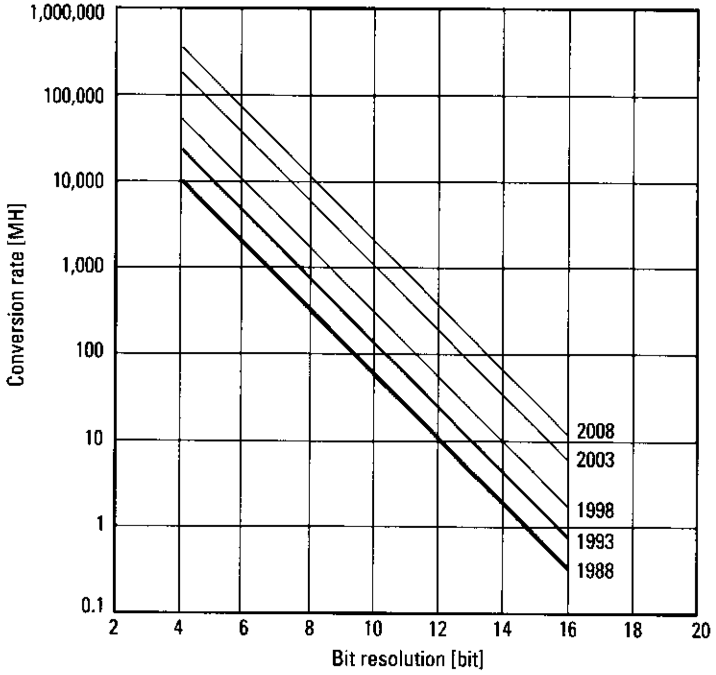


Figure 8.28 Future prospect of ADC.

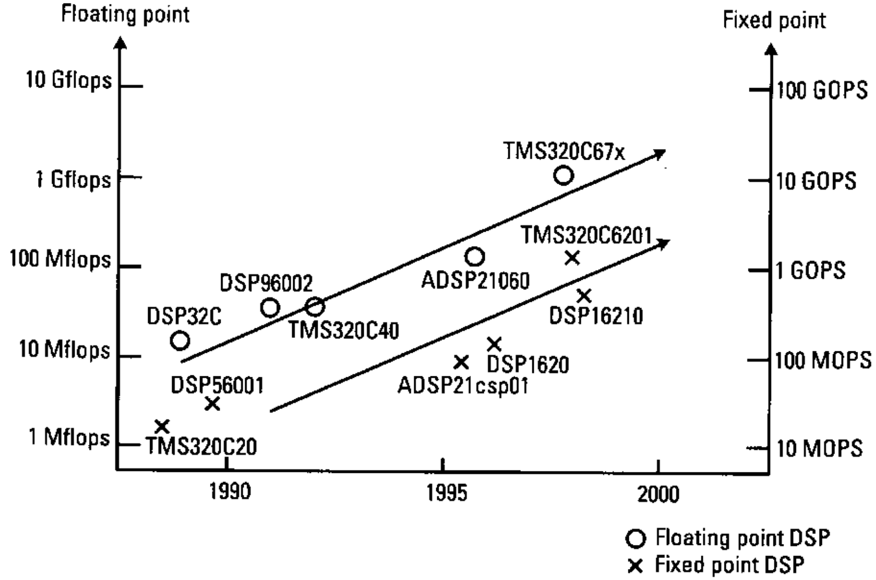


Figure 8.29 Future prospects for the digital signal processor.

8.5.7.3 Field-Programmable Gate Array (FPGA) [26]

An FPGA is an array of gates with a programmable interconnect and logic function that can be defined after manufacture. It consists of an array of blocks, each block containing logic blocks. A typical FPGA architecture is shown in Figure 8.30. An FPGA logic block usually contains *lookup tables* (LUTs) of *n* inputs. Inputs to the logic block are connected to either the LUT input or flip-flop input ports. Outputs from LUTs are connected to either the output ports of the logic block or flip-flop input ports. By adding multiplexers, various combinations of input signals can be chosen. LUTs are realized by using *static RAM* (SRAM). By setting data contents at all addresses of static RAM, LUTs can implement any Boolean function. Moreover, flip-flops are used to store data such as state information of finite state machine. Figure 8.31 shows an example of the FPGA logic block's structure [26].

VHDL and Verilog-HDL are the programming languages for designing FPGAs. Their software can be reconfigured by changing the data in the SRAM. In comparison with DSP, FPGAs can be put in parallel for high-speed signal processing. However, programming in VHDL or Verilog-HDL requires knowledge of the hardware architecture of the target device. Work is progressing on converters between such hardware description functions and high-level

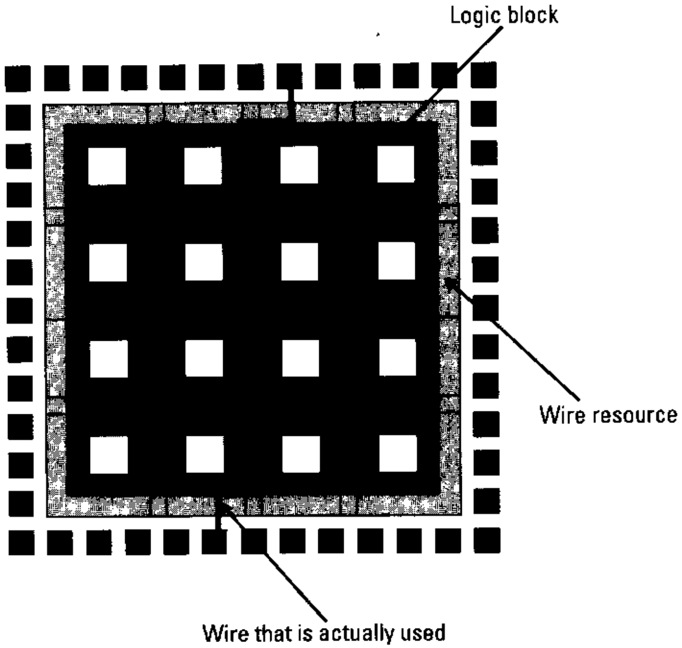


Figure 8.30 A typical FPGA architecture [26].

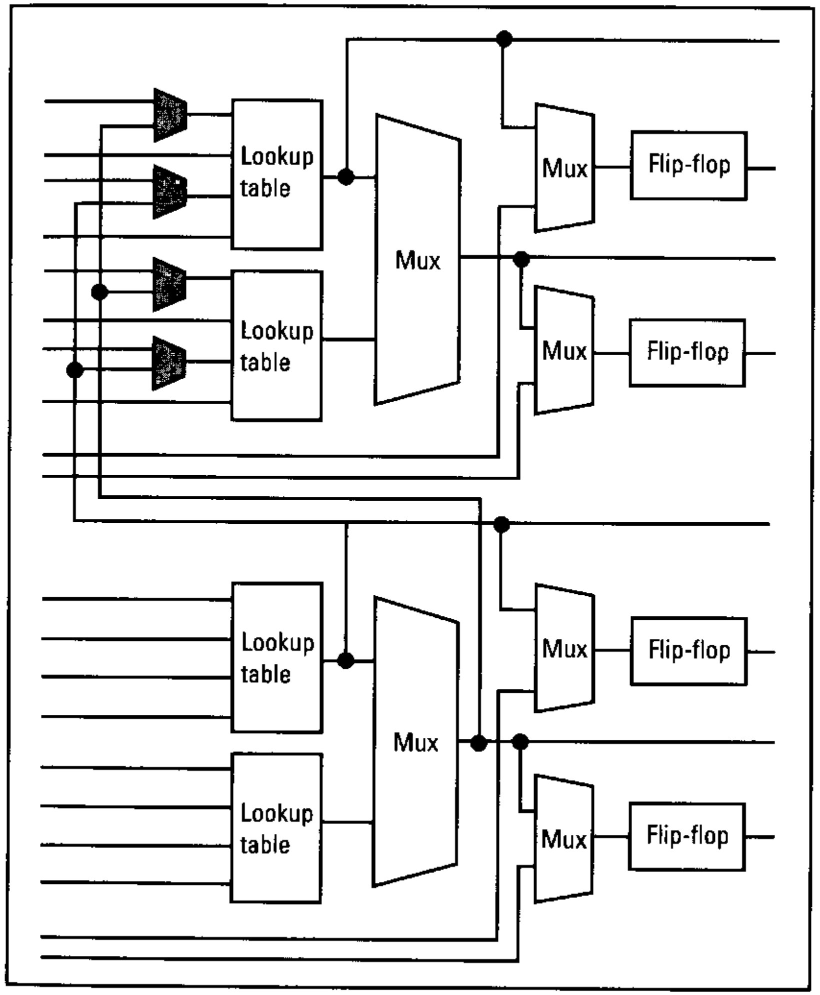


Figure 8.31 An example of an FPGA logic block structure [26].

computer languages, such as C language. Moreover, the FPGAs now exceed 1 million gates [27]. In the future, all mobile communication systems may be programmed using only one FPGA.

8.6 Software Radio Communication System Projects

This section summarizes the projects related to software radio communication systems in the United States, Europe, and Japan.

8.6.1 United States

Research and development on software radio in the United States began with military communication systems. The best-known system is the "SPEAKeasy" project of 1994 [2], which integrates many communication systems for military use into one terminal. Its main objective is to communicate between terminals having distinctive modulation schemes. The concept is called "voice bridge." The system concept is shown in Figure 8.32. All components are arranged as functional modules, and *common system equipment* (CSE) is utilized to manage these functional modules. A prototype was developed, and Table 8.8 shows the baseline waveform compatibility, in SPEAKeasy.

After the success of the SPEAKeasy project, research and development on software radio accelerated. Many projects are now in the works, including the SDR forum and the GloMo project.

8.6.1.1 Software-Defined Radio Forum (Former MMITS Forum) [28]

The members of project SPEAKeasy established a standardization organization for software radio in 1996. The forum discusses the following standard points: (1) standardization of the interfaces between integrated modules or between the integrated modules and common infrastructure shown in Figure 8.33 [29] and (2) standardization of the procedure of software download and its technology. The forum has two committees: a technical committee and a market committee. In the technical committee, four working groups exist: (1) the base station and antenna work group, (2) the download work group, (3) the handheld work group, and (4) the mobile work group. More than 80 institutes and companies worldwide are currently members of the forum. The forum has published technical documents describing the basic architecture, the notation method of the *application programmable interface* (API), and software download methods (URL: http://www.sdrforum.org).

8.6.1.2 GloMo (Global Mobile Information Systems) Project [30]

The main objective is to enable mobile users to access the full range of services available in the defense information infrastructure. In particular, this program will enable the use of application environments such as the World Wide Web, video servers, videoconferencing, whiteboarding, electronic mail, and voice communications, despite the sporadic connectivity characteristics of wireless mobile communications [31]. This project is sponsored by the *Defense Advanced Reseach Projects Agency* (DARPA) and is organized by a number of universities and industries in the United States. Table 8.9 lists the research titles and research organizations [30]. (URL: http://www.darpa.mil/ito/glomo/index.html.)

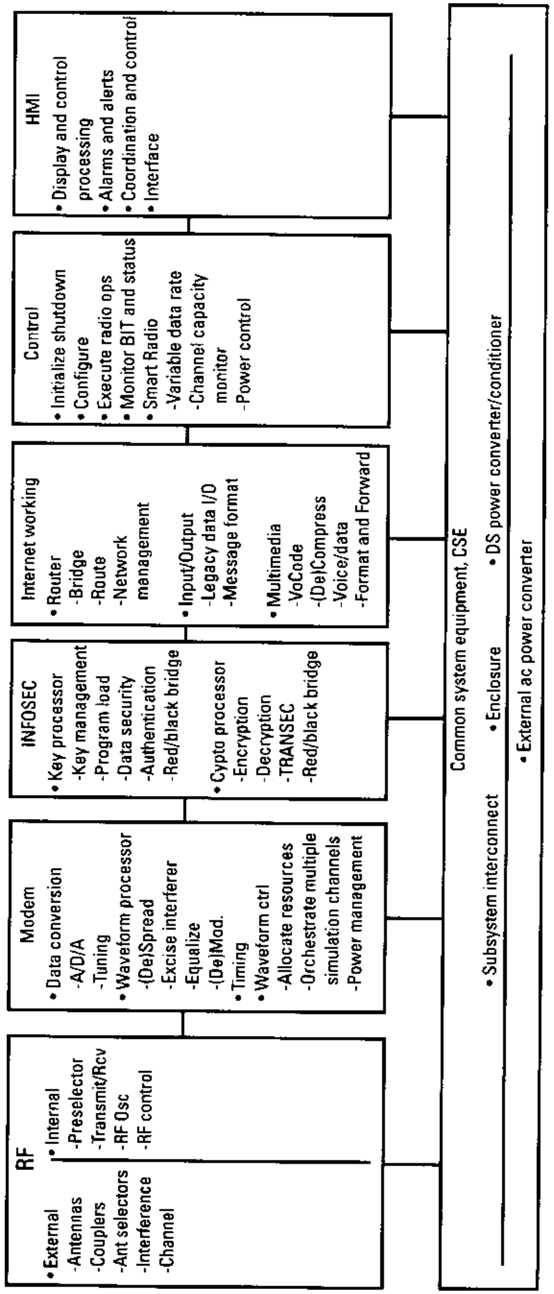


Figure 8.32 SPEAKeasy system architecture and functional allocations.

Table 8.8
Baseline Waveform Compatibility

Waveform	RF Band	Synopsis
STAJ	HF	Frequency hopping data and voice communications
HF modem (MIL-STD-188-110A)	HF	Multirate data communications
Automatic Link Establishment (MIL-STD-188-141A, App. A)	HF	Automatic setup and monitoring of HF links, including channel scanning, automatic answering, and link quality analysis
PACER BOUNCE	HF	Classic analog HF communications
SINCGARS	VHF	Frequency hopping voice and data communications
HAVEQUICK I/II	UHF	Frequency hopping voice and data communications

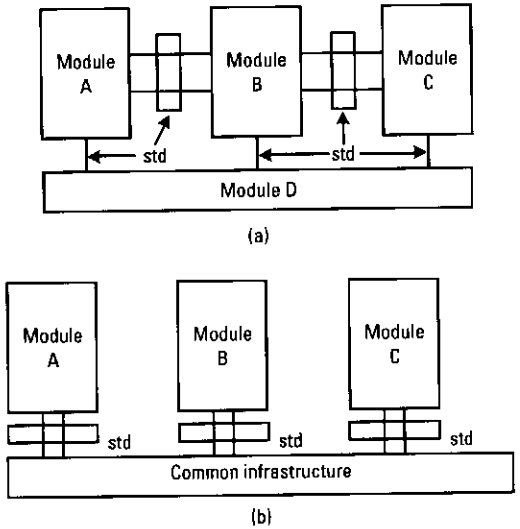


Figure 8.33 Standardization point discussed in SDR forum: (a) module-to-module and (b) module-to-bus standards.

8.6.2 Europe

In Europe, the first International Workshop on Software Radios was held in Brussels, Belgium, in May 1997. The workshop was organized by the *European Commission* (EC) DG XIII-B in the *Advanced Communications Technologies and Services* (ACTS) program. The ACTS program supports some projects related to software radio communication systems [32, 33], described as follows.

Table 8.9
Research Activity of GloMo

Organization	Research Title
BBN Technologies	Support for distributed real-time multimedia applications in *mobile wireless networks* (MMWN)
BBN Technologies	*Density and asymmetry adaptive wireless networking* (DAWN)
Carnegie Mellon University	Mobile data access
Carnegie Mellon University	Pyxis: Middleware for distributed multimedia programming
CTA Incorporated	Integration of DBS into digital battlefield
ITT Aerospace Communications Division	High-reliability all-informed voice service
Massachusetts Institute of Technology (MIT)	Spectrum wave
Massachusetts Institute of Technology	Information survivability technology and experiments for GloMo
PLATINUM Technology, Inc.	Mobile computing support for database applications
Raytheon Systems Company	*Advanced signal processing & networking* (ASPEN)
Rutgers University	Dataman project-information services for low-powered wireless mobile clients
Rutgers University	NIMBLE: Many-time, many-where communication support for information systems in highly mobile and wireless environments
Science Applications International Corporation (SAIC)	*Simulation and evaluation of adaptive mobile large-scale networks systems* (SEAM-LSS)
SRI International	*System engineering, integration* (SE&I) and coordination for GloMo
SRI International	Advanced secure wireless integrated networks
Stanford University	Low-power distributed mobile networks
Stanford University	Reconfigurable multimode, multiband information transfer systems
University of Texas at Dallas	Generic control channel mechanism
University of Calfornia, Berkeley	Toward a wireless overlay internetworking architecture
University of Calfiornia, Los Angeles (UCLA)	Transparent virtual mobile environment (Travler)
University of California, L\os Angeles	Handheld untethered nodes for high performance wireless network systems
University of California, Los Angeles	DOMAINS: Design of mobile adaptive networks using simulation and agent technology
University of California, Santa Cruz	WINGs for the Internet
University of California, Santa Cruz	SPARROW
University of Kansas	*Rapidly deployable radio network* (RDRN): phase II
University of Texas Pan American	Reconfigurable antennas for high-data-rate untethered nodes
USC Information Sciences Institute	MOSIS: An innovative prototyping service
USC Information Sciences Institute	*GloMo university modular packaging system* (GUMPS)
Virginia Tech	Software radio using reconfigurable computing

- *Future Radio Wideband Multiple Access Systems* (FIRST): FIRST's main objective is a feasibility study and the demonstration of intelligent multimode terminals for second- and third-generation mobile systems such as GSM 1800 and TD-CDMA. Its projects include concept papers [34, 35]. A DSP-based software radio demonstrator, for the software radio algorithm of the GSM 1800 and TD-CDMA air interfaces, has also been developed [36]. Its URL is http://www.era.co.uk/first/First.htm.

- *Software Radio Technologies* (SORT): SORT's main objective is to establish the requirements for adaptive radio access including the definition of a functional architecture and digital signal processing and the development of a real-time hardware demonstrator consisting of an anti-aliasing filter and ADC and functional blocks for channelization and sample rate adaptation (e.g., GSM and WCDMA). SORT's aim is to obtain the minimum standardization functions for the software air interface. Its URL is http://www.ifn.et.tu-dresden.de/sort/welcome.html.

- *Smart Universal Beamforming* (SUNBEAM): SUNBEAM's main project objective is the development of innovative base station array processing architectures and algorithms for UMTS that are sufficiently flexible to support a range of architectures and software radio techniques. Its URL is http://www.project-sunbeam.org. A related EU project, the Esprit program, focuses on projects related to software radio communication systems.

- *Software Libraries for Advanced Terminal Solutions* (SLATS): SLAT's main objective is developing the software libraries for software radio applications of both second- and third-generation systems (for a range of platforms). The project consists of phase A, definition of requirements for libraries, and phase B, development of the libraries. Its URL is http://www.csem.ch/systemeng/slats.
- *Programmable Multimode Radio for Multimedia Wireless Terminals* (PROMURA): PROMURA's main objective is to develop a prototype of a programmable RF System that will support TDMA and CDMA wideband-RF architecture and to develop H/W blocks for WB-radio by using the BICMOS/Bipolar SiGe technology.
- *Mobile Multimedia Access Using Intelligent Agents* (M3A): The M3A project aims to bind the three different kinds of networks (e.g., GSM, DAB, and HiperLAN) together and make them work transparently with a browser. It will provide UMTS-like services and help in gaining experience and understanding of user reactions. It will also exploit the network and terminal technology of the existing systems and agent technology, to produce a compact terminal.

These projects were completed by March 2000. A new project, the *Information Society Technologies* (IST) program, has started. IST has at least eight projects that are related to software radio. Reference [33] is very instructive, and Table 8.10 shows a list of projects.

8.6.3 Japan

In Japan, demand for software radio was first evident in 1995. The Japanese government recognized the importance of software radio for wireless systems [37]. In 1996, a study group was organized by ARIB with the support of the Ministry of Post and Telecommunications of Japan and a final report was published in 1999 after 3 years of study. This study group was open to a limited number of people. To provide open discussion on software radio and its technologies, a software radio study group was organized by the *Institute of Electronics, Information and Communication Engineers* (IEICE) and has been ongoing since December 1998 [38]. The following is a brief introduction on the latest activities of these two study groups.

- Research and investigation group of software radio in ARIB [37]: The study group's activities date from 1996 and lasted 3 years. Its main objectives were to study the feasibility of a universal terminal applicable for multiband, multimode, and multirate wireless signals and the

Table 8.10
Projects Related to Software Radio Technology in the IST Project [33]

CAST	Aims at laying the foundations for intelligent and adaptable configuration of the physical layer in wireless communications links, enabling users to access customized services over networks operating with different radio standards across different frequency bands. Targets implementation of the protocol stack by proposing a three-layer (management, procedural, physical) reconfigurable architecture to provide the interface between the application and the underlying physical layer of the terminal processing platform; will build a validation platform to assess the operation of the proposed architecture for the delivery of selected user services URL: http://cast5.freeserve.co.uk
DRIVE	Aims at developing methods for dynamic frequency allocation and for coexistence of different radio technologies (dynamic radio) to increase spectrum efficiency and reach; will develop an IPv6-based multiradio infrastructure to ensure optimized interworking of cellular and broadcast networks for the provision of adaptive high-quality multimedia services in vehicular environments URL: http://www.comnets.rwth-aachen.de/drive
MOBIVAS	Aims at developing architectural approaches for integrated software platforms and systems, adaptable to different network services and technologies, that will open new opportunities for advanced *value-added service* (VAS) providers, and at developing innovative and modular network components for seamless and efficient service provision, enabling downloadable software-defined radio VAS; will combine wireless technology with CORBA/TINA principles and with sophisticated QoS mechanisms URL: http://www.mobivas.ccrle.nec.de
PASTORAL	Aims at developing a reconfigurable, real-time platform for third-generation mobile terminal baseband development, using FPGA devices developed through a new cosimulation, codesign methodology that allows for an accelerated design cycle; also looks into downloading applications and protocols over the air for reconfiguration URL: http://www.ist-pastoral.org
SATURN	Looks into adaptive/smart antenna techniques, on both the terminal and the base station, for outdoor (UMTS) and for local/campus area wireless networks (HiperLAN), aiming to promote high-bit rate wireless services as well as to provide enhanced location information for location-based services URL: http://www.ist-saturn.org
SODERA	Aims at defining and validating the feasibility of the RF architecture best suited for reconfigurable radio taking into consideration the terminal constrains of low consumption, low cost, and small form factor, as well as at studying the optimum partitioning between different technologies (e.g., BICMOS-SiGe, SOI, and Micro-Machining); advanced RF libraries will be developed to validate this approach URL: http://www.ist-sodera.org

Table 8.10 (continued)

TRUST	Starting from the user requirements from the perspective of the terminal, investigates enabling technologies such as analog signal processing, adaptive baseband processing, novel transceiver algorithms, and smart power management; investigates also system aspects like spectrum sharing techniques, multimode monitoring, intelligent mode switching, and software download including security issues URL: http://www.ist-trust.org
WIND-FLEX	Investigates high-bit rate adaptive modem architecture, configurable in real time, for indoor single-hop, ad hoc networks, concentrating on algorithms, protocols, and RF/IF subsystems URL: http://www.vtt.fi/ele/research/els/projects/windflex.htm

feasibility of monitoring and surveillance of illegal multiband, multimode, and multirate wireless signals. In conjunction with the above objectives, the study group investigated the following related topics: (1) suitable software architecture for software radio receivers, (2) organization of software radio receivers, (3) future trends in application markets for software radio receivers, (4) survey of R&D trends of SDR, (5) standardization items for SDR receivers, and (6) an evaluation method for the software radio receivers. The study group also evaluated the software radio receiver by developing a prototype for wireless signal monitoring [18].

- Software radio study group in IEICE [38]: The study group's activities date from December 1998. The study group belongs to the IEICE Communication Society. The following subjects are discussed: (1) theory of software radio, (2) software and hardware technology, (3) applications, (4) research on APIs, (5) standardization, and (6) information exchange and cooperation with active organizations in other countries [38]. As of this writing, there have been six technical meetings. Around 60 papers on the above topics were presented in the technical committee.

Moreover, prototypes from several research institutes and companies have been produced. The following describes these prototypes. This information was collected until October 2000.

- ARIB [18]: The ARIB study group developed a prototype software radio receiver for wireless signal monitoring in 1998 and 1999. The configuration of the 1999 prototype is shown in Table 8.11. The prototype can change its demodulation method by changing its software.

Table 8.11
System Parameters of the Prototype by ARIB [18]

Frequency band	27 MHz, 900 MHz, 2 GHz
Antenna array	Six-element circular array
Modulation method	BPSK, GMSK, FM, AM
Bit rate	8, 16, 32 Kbps
Direction of arrival estimation algorithm	MUSIC [39]
Interference wave suppression	DCMP [40]

The prototype can accommodate an array antenna with six elements, and the direction of arrival of the required wave is estimated by using the *multiple signal classification* (MUSIC) algorithm [39]. In addition, the prototype can suppress undesired signals by using the *directionally constrained minimization of power* (DCMP) algorithm [40].

- Communications Research Laboratory (CRL), M.P.T. [12, 23]: CRL developed a prototype software radio transceiver for an intelligent transport system in 1999 [23]. The configuration and appearance of the prototype are shown in Table 8.3 and Figure 8.20, respectively. The prototype is triple mode (Japanese PHS, ETC, and GPS) by using a menu shown in Figure 8.34. The prototype is the first of its kind for the intelligent transport systems and uses a PC-based management software. It is shown in Table 8.3 and Figure 8.20. The prototype includes the parameter-controlled software radio [11] and MMSR [13] concepts described in Section 8.4.

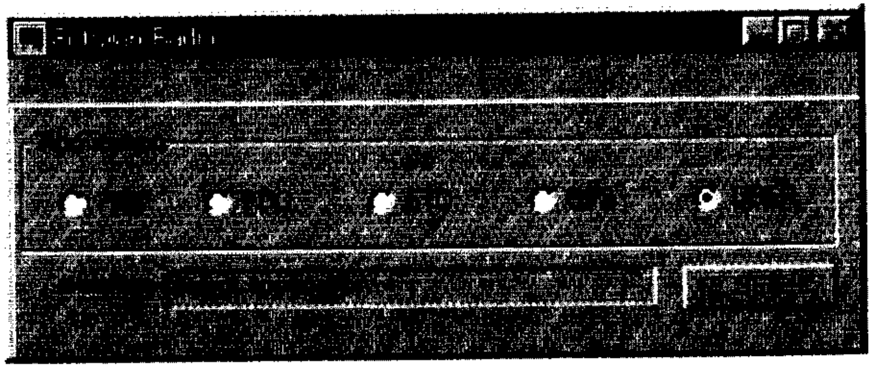


Figure 8.34 Menu window of management software.

- NTT: NTT developed a prototype software radio transceiver for an advanced mobile communication system in 1999 [41, 42]. The configuration of the prototype is shown in Table 8.12. The prototype includes three-array antennas and offers a RCR STD-28 based standard (PHS) protocol. Moreover, the prototype can switch its communication scheme between TDMA/TDD and FDMA/TDD and perform multimode on π/4QPSK and 16-QAM. In addition, the prototype can connect with an ISDN network.
- TOSHIBA: TOSHIBA developed a prototype software radio transceiver for an advanced mobile communication system in 2000 [17]. The configuration of the prototype is shown in Table 8.13. The prototype adopts direct conversion technique of Section 8.5.4 at the receiver as a frequency conversion method from RF band to baseband.
- NEC: NEC developed a prototype software radio receiver for wireless signal monitoring in 1998 [43]. The configuration of the prototype is shown in Table 8.14. The prototype uses the undersampling technique of Section 8.5.4 as a frequency conversion method from the IF band to baseband. It can use many modulation schemes such as AM, FM, MSK, GMSK, BPSK, QPSK, π/4QPSK, 8PSK and 16-QAM by changing its software. A modulation recognition algorithm is also included.

Table 8.12
System Parameters of the Prototype by NTT [41, 42]

RF band	2.45 GHz
Bandwidth	13 MHz
Frequency conversion method	Double superheterodyne
IF	2/5/39 MHz
Number of RF/IF systems	3
Access protocol	TD MA/TDD FDMA/TDD
Modulation scheme	π/4-shift QPSIC, 16-QAM
Communication rate	96/384 Kbps
External interface	ISDM (base station) Voice, bearer (personal station)

Table 8.13
System Parameters of the Prototype by TOSHIBA [17]

RF band	1.5/1.9 GHz
Bandwidth	10 MHz
Frequency conversion method	Direct conversion
Modulation scheme	Maximum 384 Kbps
Encode	Differential code

- Toyo Communication Equipment and Tohoku Electronics Power Company [44]: Toyo Comm. developed a prototype software radio transceiver for an intelligent base station in 1999 [44]. The configuration of the prototype is shown in Table 8.15. The prototype has

Table 8.14
System Parameters of the Prototype by NEC [43]

Modulation schemes	AM, FM, FSK, MSK, GMSK, BPSK, QPSK, π/4QPSK, 8PSK, 16-QAM
Symbol rate	Maximum 10 Msymbol/second
Speech codec	32-Kbps ADPCM, CVSD-PM
Sampling method	IF undersampling

Table 8.15
System Parameters of the Prototype by Toyo Comm. [44]

Modulation schemes	FM, FSK, BPSK, QPSK, π/4 QPSK
RF frequency	370–380 MHz
Transmitter bandwidth	1.25 MHz
Receiver bandwidth	650 kHz
ADC	12 bits/40 MHz
DAC	12 bits/40 MHz

Table 8.16
System Parameters of the Prototype by Hitachi Kokusai Electronic Inc. [19]

Modulation scheme	FSK, BPSK, QPSK, 16-QAM, AM, FM, SSB (USB, LSB)
IF frequency	455 kHz–100 MHz
Transmission speed	50 bps–64 Kbps
ADC	12 bit/125 Msps 6 bit/30 Msps
DAC	14 bit/125 Msps 16 bit/30 Msps

software download technology. When downloading software, the prototype can still communicate.

- Hitachi Kokusai Electronic Inc. [19]: Hitachi Kokusai Electronic Inc. developed a prototype software radio transceiver for an advanced mobile communication system in 2000 [19]. The configuration of the prototype is shown in Table 8.16. The prototype has a custom-made DSPH and supports various modulation schemes such as FSK, BPSK, QPSK, 16-QAM, AM, FM, and SSB, which can be used to communicate between transmitter and receiver in full duplex operation.

Other companies have started to develop software radio systems. We can expect to see more prototypes in the near future.

8.7 Future Applications of Software Radio Communication Systems

8.7.1 Future Telecommunications Applications

- *Mobile communication terminal:* As mentioned above, a software radio technology–based mobile communication terminal allows users to select a system by changing the software of the DSPH. Moreover, users can select the provider company as they like. This will reduce the number of hardware products and industrial waste. These points were mentioned in Section 8.1. However, to implement them will require miniaturization and reduction in the low power consumption and cost of the hardware as well as reduction in software download time. An efficient system selection procedure is also required.
- *Mobile communication base station:* The base station that has software radio technology would make it easy to implement new communication systems and fix bugs. In addition, base stations could use more than one algorithm for fading compensation. Moreover, by using an adaptive array antenna, the antenna beam shape could also be controlled with software. To realize this application, we must make a rule for programming software that configures components for the mobile communication system to change the program easily and to make the connection with other components. Moreover, a fast ADC or digital signal processor is needed to perform real-time transmission.
- *Broadband radio access system:* Software radio technology can enhance the flexibility of a broadband radio access system, which operates between base stations and buildings. For example, software can be used for components such as distortion compensators and interference suppressors. The modulation scheme also can be changed, and the best components selected. Using an adaptive array antenna would allow the beam shape to be controlled with software. To do so, we must make a rule to make each module by software, specially, a rule for the connections between modules. A high-speed ADC or digital signal processor must also be used to perform real-time transmission.

- *ITS:* The communication systems utilized in the ITS are shown in Table 8.2. They fall in three categories: communication-based systems, control-based systems, and broadcasting-based systems. Representative examples of communication-based systems are PDC, PHS, and digital cellular using CDMA. The third-generation mobile communication system based on CDMA: IMT-2000 is expected in the near future. The GPS system, the VICS system, and radar are representative examples of control-based systems. The advanced cruise-*assist highway system* (AHS), which provides information on traffic accidents, road obstructions, and meteorological phenomena, and the ETC system will be mounted in cars. Representative examples of broadcasting-based systems are radio broadcasting systems, analog-television broadcasting systems and satellite broadcasting systems, and digital television systems. As mentioned in the above, the number of communication systems appearing in the ITS system will likely increase. This means that space saving and integration of the system components must be chief concerns if so many services are to be provided in the limited space of a car. Software radio technology holds the prospect of these services being provided on only one small terminal. Figure 8.35 shows the potential application. The method of introducing software radio technology is the same as shown in applications (1)–(3). In the ITS, the number of services that can be used in a car is more than 10. Therefore, we urge the implementation of software radio technology as soon as possible.

8.7.2 Broadcasting Applications

Software radio technology can realize automatic switching between terrestrial wave/base station/circuit switch/cable television. Moreover, it is easy to introduce new services on existing hardware if only the software needs to be changed. Moreover, receivers compatible with the standards of many countries can be realized. By using receivers, it is easy to promote international distribution of products. Such a multimode, multistandard receiver must have a

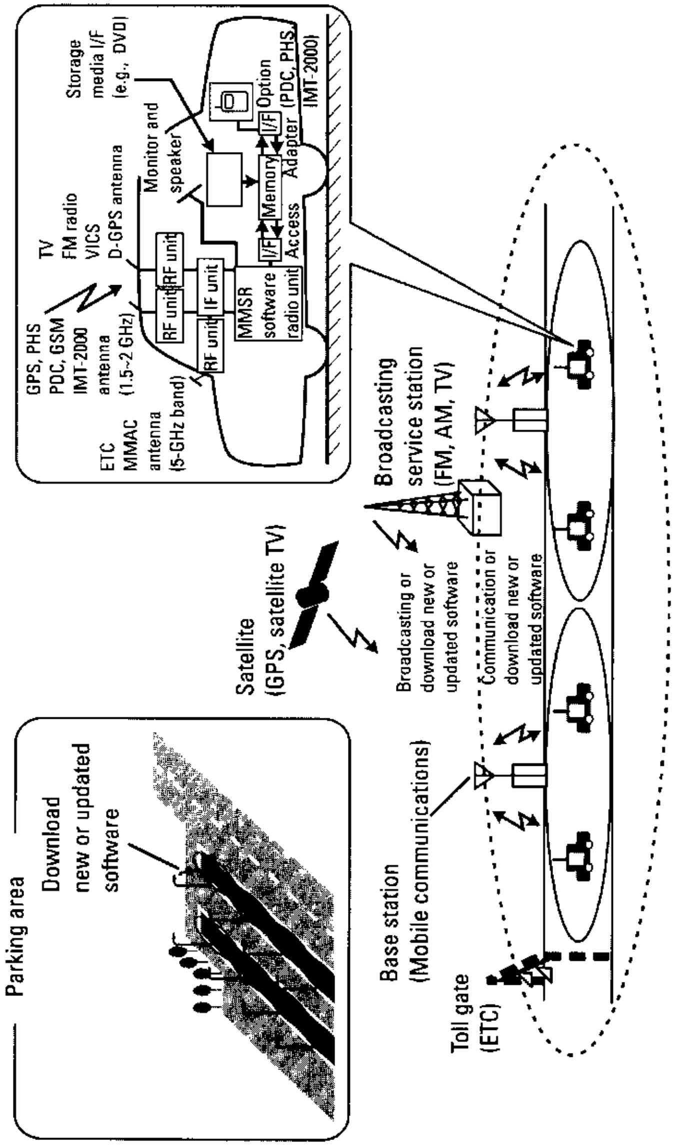

Parking area
Download new or updated software
Satellite (GPS, satellite TV)
Broadcasting or download new or updated software
Broadcasting service station (FM, AM, TV)
Communication or download new or updated software
Base station (Mobile communications)
Toll gate (ETC)
ETC MMAC antenna (5-GHz band)
GPS, PHS PDC, GSM IMT-2000 antenna (1.5~2 GHz)
TV FM radio VICS D-GPS antenna
Monitor and speaker
Storage media I/F (e.g., DVD)
RF unit
RF unit
RF unit
IF unit
MMSR software radio unit
I/F
Memory
I/F
Access
Option Adapter (PDC, PHS, IMT-2000)

Figure 8.35 Applications of software radio for future ITS system.

broadband antenna and software describing the components of the particular communication system, and this means that the connectivity between components must be increased.

8.7.3 For Private Network

Nowadays, many private networks exist for educational or office or community or ITS or emergency and commercial use (Figure 8.36). Current terminals for these networks do not have any connectivity with each other. Software radio can be used to create a universal terminal that works on any private network. This means that huge networks can be configured. In the near future, these huge networks may oppose the conventional public network.

8.7.4 Certification Method of Software Radio

One problem facing the practical implementation of software radio is wireless terrorism [14]. The potential types of damage are shown in Figure 8.37. In some cases, downloaded data may be illegally altered or have a computer virus. Caused by the altered software or virus software, base stations may be destroyed or shut down. Moreover, the kinds of download software are different between transmitters and receivers, and as one of the results, the terminal radiates high power signal suddenly. To oppose such wireless terrorism, a new certification method for the systems used in software radio must be required in the certification organizations.

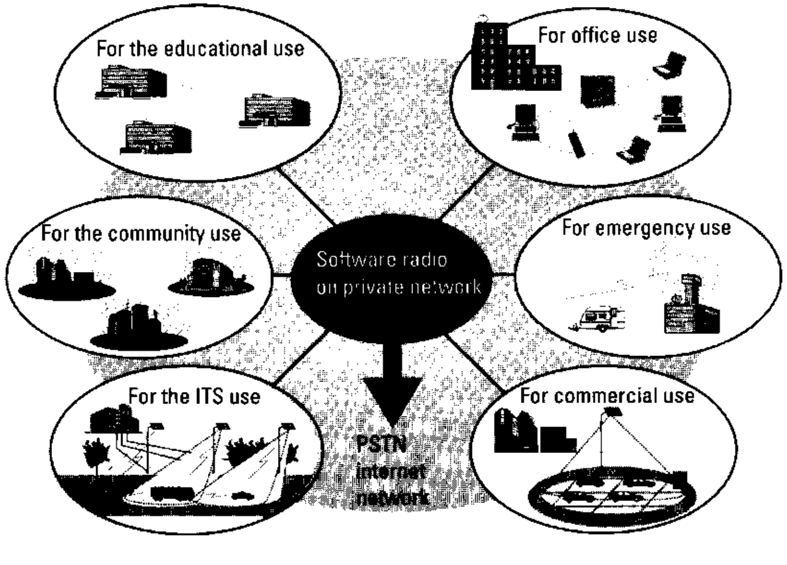


Figure 8.36 Applications of software radio of private network.

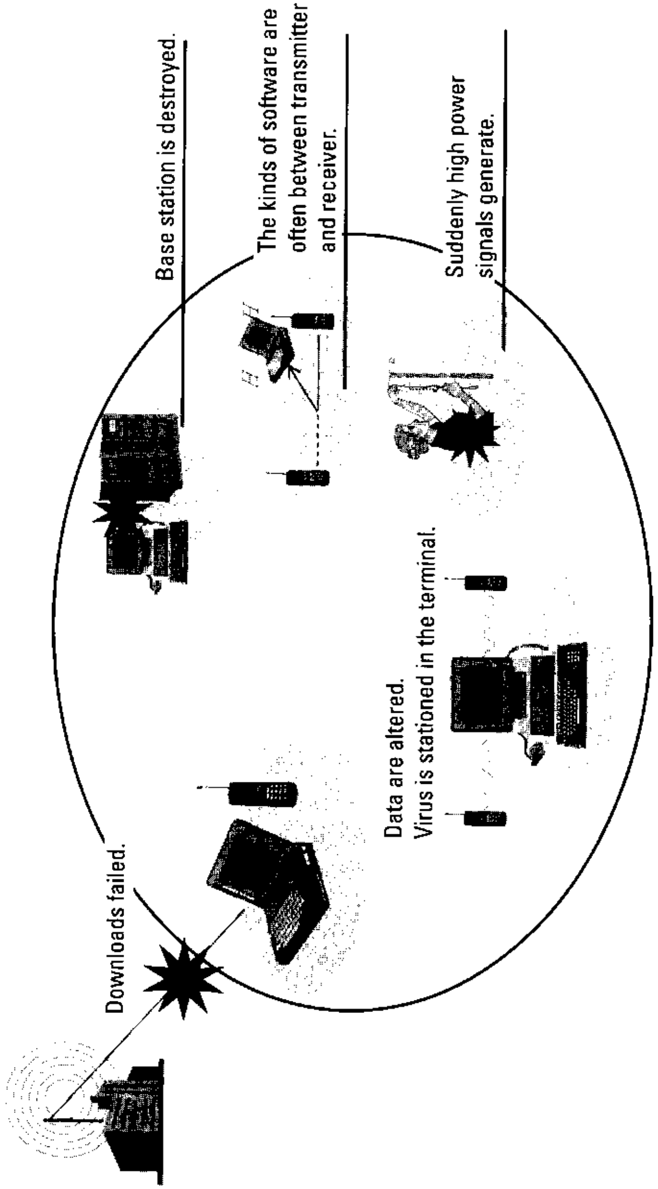

Base station is destroyed.
The kinds of software are often between transmitter and receiver.
Suddenly high power signals generate.
Data are altered.
Virus is stationed in the terminal.
Downloads failed.

Figure 8.37 Wireless terrorism.

An example of certification is shown in Figure 8.38. The certification method has two stages. In the first stage, the certification organization checks the relationship between input data and output data by changing the software that configures several systems. In this case, for each system, the transmission power, transmission bandwidth and electric power leakage in the adjacent frequency band are measured and evaluated by comparing the radio law of each country.

If the software is well worked, the certification organization gives a certification password to the software. The certification password is used in the handshake between the DSPH and memory before downloading the software to the DSPH. During the handshake, if the password is certificated, the software is downloaded from memory to the DSPH, otherwise the software radio

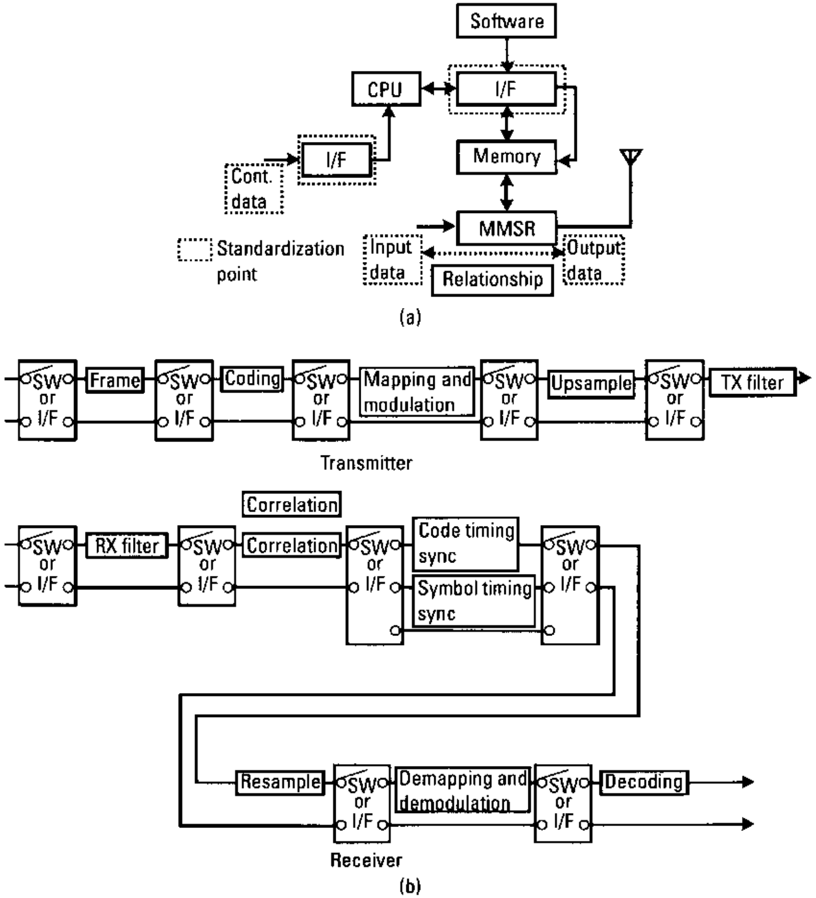


Figure 8.38 Certification method of software radio equipments: (a) first stage and (b) second stage.

equipment issues a warning; in the worst case, the equipment must destroy itself. The public key cryptography technique is a candidate for the certification password. Moreover, the download protocol between the memory and the DSPH must be standardized and the protocol must be certificated.

The second stage of certification involves the modules that configure software radio. To certify each component, the configuration of software radio must be changed. An example of configuration is shown in Figure 8.38(b). Here, switching or interface modules are inserted between functional modules to configure the software radio equipment. These switching or interface modules can be controlled from outside of the DSPH. By switching modules, each functional module is certificated. The switching or interface module must be standardized for such a certification procedure to work.

8.8 Conclusions

One of future applications of the software programming method studied in this book is software radio communication systems. The concept of software radio bridges the software programming studied in this book and hardware implementation.

This chapter describes the software radio communication system, its advantages, and its problems. The technologies for realizing a software radio communication system and its future applications are also discussed. This technology will likely become key to the success of fourth-generation mobile communication systems, because nowadays there are many systems in the world and these systems must be integrated. The issues discussed in this book will be useful to integrate them.

References

[1] Mitila, J., "The Software Radio Architecture," *IEEE Commun. Mag.*, May 1995, pp. 26–38.

[2] Lackey, R. J., and D. W. Upmal, "SPEAKeasy: The Military Software Radio," *IEEE Commun. Mag.*, May 1995, pp. 56–61.

[3] European Commission DG XIII-B: *Proc. of Software Radio Workshop*, May 1997.

[4] Special Issue on the Globalization of Software Radio, *IEEE Commun. Mag.*, February 1999.

[5] Special Issue on Software Radio, *IEEE Personal Communications*, August 1999.

[6] Special Issue on Software and DSP in Radio, *IEEE Commun. Mag.*, August 2000.

[7] Special Issue on Software-Defined Radio and Its Technologies, *IEICE Trans. on Commun.*, June 2000.

[8] Kohno, R., "Structures and Theories of Software Antennas for Software Defined Radio," *IEICE Trans. Commun.*, Vol. E83-B, No. 6, June 2000, pp. 1189–1199.

[9] Karasawa, Y., "Algorithm Diversity in a Software Antenna," *IEICE Trans. Commun.*, Vol. E83-B, No. 6, June 2000, pp. 1229–1236.

[10] Mangum, C., "Market Opportunities II: The MMITS Forum Market Forecast Study," *1st International Software Radio Workshop*, Rhodes, Greece, June 1998, pp. 15–24.

[11] Harada, H., and M. Fujise, "Multimode Software Radio System by Parameter Controlled and Telecommunication Toolbox Embedded Digital Signal Processing Chipset," *Proc. of 1998 ACTS Mobile Communications Summit*, June 1998, pp. 115–120.

[12] Harada, H., Y. Kamio, and M. Fujise, "Multimode Software Radio System by Parameter Controlled and Telecommunication Component Block Embedded Digital Signal Processing Hardware," *IEICE Trans. Commun.*, Vol. E83-B, No. 6, June 2000.

[13] Harada, H., "A Proposal of Multimode & Multiservice Software Radio Communication Systems for Future Intelligent Telecommunication Systems," *Proc. of International Symposium on Wireless Personal Multimedia Communications (WPMC'99)*, September 1999, pp. 301–304.

[14] Harada, H., "Wireless Terrorism," *Denpa-shinbun* newspaper, May 30, 2000.

[15] Tsurumi, H., and Y. Suzuki, "Broadband RF Stage Architechture for Software-Defined Radio in Handheld Terminal Applications," *IEEE Commun. Mag.*, Vol. 37, No. 2, February 1999, pp. 90–95.

[16] Tsurumi, H., et al., "Broadband and Flexible Receiver Architecture for Software-Defined Radio Terminal Using Direct Conversion and Low-IF Principle," *IEICE Trans. Commun.*, Vol. E83-B, No. 6, June 2000, pp. 1246–1253.

[17] Yoshida, H., et al., *A Software-Defined Radio Using Direct Conversion Principle*, Technical Report of IEICE, SR00-09, April 2000, pp. 59–66.

[18] Yokoi, T., et al., "Software Receiver Technology and Its Applications," *IEICE Trans. Commun.*, Vol. E38-B, No. 6, June 2000, pp. 1261–1268.

[19] Ide, T., et al., *Development and Evaluation of Software Radio Prototype*, IEICE Technical Report, SR00-23, October 2000 (in Japanese).

[20] Vanghan, R. G., N. L. Scott, and D. R. White, "The Theory of Bandpass Sampling, " *IEEE Trans. Signal Proc.*, Vol. 39, No. 9, September 1991, pp. 1973–1984.

[21] Hentschel, T., and G. Fettweis, "Sample Rate Conversion for Software Radio," *IEEE Commun. Mag.*, Vol. 38, No. 8, August 2000, pp. 142–150.

[22] Sawai, R., et al., "An Adaptive Symbol Timing Synchronization Method Multimode & Multiservice Software Radio Communication System," *IEICE Trans. on Communications*, Vol. E84-B, No. 7, July 2001, pp. 1885–1896.

[23] Harada, H., Y. Kamio, and M. Fujise, *A New Multimode and Multiservice Software Radio Communication System for Future Intelligent Transport Systems*, Technical Report of IEICE, SR99-21, September 1999 (in Japanese)

[24] Sawai, R., et al., "A Feasibility of an Adaptive Sampling Processing Method for Software-Defined Radio," *Proc. of 2001 IEEE Vehicular Technology Conference Spring (VTC 2001-Spring)*, Rhodes, Greece, May 2001.

[25] Walden, R. H., " Performance Trends for Analog-to-Digital Converters," *IEEE Commun. Mag.*, Vol. 37, No. 2, February 1999, pp. 90–95.

[26] Cunmings, M., and S. Haruyama, "FPGA in the Software Radio," *IEEE Commun. Mag.*, Vol. 37, No. 2, February 1999, pp. 108–112.

[27] http://www.altera.com/.

[28] http://www.sdrforum.org.

[29] Cummings, M., J. Hoffmeyer, and S. Blust, "Modular Multifunctional Information Transfer System Forum," *1st Software Radio Workshop*, Brussels, Belgium, May 1997.

[30] http://www.darpa.mil/ito/research/glomo/index.html.

[31] http://www.darpa.mil/ito/research/glomo/projects.html.

[32] Tuttlebee, W., "Software Radio: Developments in Europe," *1st International Software Radio Workshop*, Rhodes, Greece, June 1998, pp. 49–61.

[33] Pereira, J. M., "Redefining Software (Defined) Radio: Reconfigurable Radio Systems and Networks," *IEICE Trans. Commun.*, Vol. E83-B, No. 6, June 2000, pp. 1174–1182.

[34] Erben, H., and P. Crichton, "Software Radio Architecture for UMTS," *Proc. of 1997 ACTS Mobile Communications Summit*, Aalborg, Denmark, October 1997.

[35] Kenington, P. B., D. W. Benett, and R. J. Wilkinson, "RF Transceiver Architechtures for Software Radio Design," *Proc. of 1997 ACTS Mobile Communications Summit*, Aalborg, Denmark, October 1997.

[36] Tayler, C., "A DSP System for a Software Radio Testbed," *Proc. of 1997 ACTS Mobile Communications Summit*, Rhodes, Greece, June 1998.

[37] Araki, K., "Prehistory of the SDR Studies in Japan—A Role of ARIB Study Group," *IEICE Trans. Commun.*, Vol. E83-B, No. 6, June 2000, pp. 1183–1188.

[38] Kohno, R., et al., "Overview of Japanese Activities in Software-Defined Radio," *12th Tyrrhenian International Workshop on Digital Communications*, Elba, Italy, September 2000.

[39] Schmidt, R. O., "Multiple Emitter Location and Signal Parameter Estimation," *IEEE Trans. Antennas and Propagation*, Vol. AP-34, No. 3, March 1986, pp. 276–280.

[40] Takao, K., M. Fujita, and T. Nishi, "An Adaptive Array Under Directional Constraint," *IEEE Trans. Antennas and Propagation*, Vol. AP-24, No. 5, May 1976, pp. 662–669.

[41] Uehara, K., et al., *Software Radio Base and Personal Station Prototype (1)*, Technical Report of IEICE, SR99-10, September 1999, (in Japanese), pp. 1–8.

[42] Suzuki, Y., et al., "Software Radio Base and Personal Station Prototype," *IEICE Trans. Commun.*, Vol. E38-B, No. 6, June 2000, pp. 1261–1268.

[43] Ishii, H., T. Yamamoto, and T. Higuchi, *Development of Software Radio for Radio Monitoring System*, IEICE Technical Report, RCS98-71, July 1998 (in Japanese).

[44] Yokoi, A., et al., *Development of Software Radio and Performance Measurement*, IEICE Technical Report, SR99-2, June 1999 (in Japanese).

About the Authors

Hiroshi Harada received his M.E. and Ph.D. from Osaka University in Osaka, Japan, in 1994 and 1995, respectively. His Ph.D. research involved system configuration methods for the radio over fiber (RoF)–based wireless communication system. In 1997 Dr. Harada was also a postdoctoral fellow at Delft University of Technology (DUT), the Netherlands, where he was engaged in the research of OFDM-based mobile communication systems, especially radio transmission techniques, and digital-signal-processing-based wireless telecommunications systems.

Since 1995, he has worked for the Communications Research Laboratory (CRL), Ministry of Posts and Telecommunications, Japan, where his research involved the areas of high-speed mobile radio trans- mission techniques by using parallel transmission, such as multicode and multicarrier-based transmission. At CRL he is currently a senior researcher.

Dr. Harada's current research interests include digital-signal-processing-based wireless telecommunications systems, such as software radio, advanced multimedia mobile access communication (MMAC) systems, and intelligent transport systems (ITS) on millimeter-wave as well as microwave bands.

As for his technical works, he developed a system that realized high-speed transmission with a rate of 4.608 Mbps by using code division multiplexing (CDM) radio transmission technology under high mobility environment with a vehicle speed of 80 km/hr on a 5-GHz band for the application of ITS in 1998. He also developed a software-defined radio terminal that is suitable for ITS application and can realize GPS, ETC, and the Japanese personal handy-phone system by changing software for digital-signal-processing hardware. This is the first software radio terminal specified for ITS application in the world.

In 2000, Dr. Harada developed a mobile communication system with a rate of several millibits per second under a mobile communication environment with an average vehicle speed of 30 km/hr on the 37-GHz millimeter-wave band by integrating RoF technology and the CDM radio transmission scheme in a system.

He is a member of the IEICE and IEEE, and he has also been a member of many standardization committees and a study group managed by the Association of Radio Industries and Businesses and the Telecom Engineering Center in the field of MMAC, advanced dedicated short range communication systems in ITS, and a software-defined radio system.

Dr. Harada has received the Young Engineer Award of the Institute of Electronics, Information, and Communication Engineers of Japan, and the Excellent Paper Award of the Third International Symposium on Wireless Personal Multimedia Communications (WPMC) in 1999 and 2000, respectively. He holds several patents and has published more than 100 papers, including ones on digital-signal-processing-based mobile communication and RoF-based wireless communication systems.

Ramjee Prasad received his B.Sc. (Eng.) from the Bihar Institute of Technology, Sindri, India, and his M.Sc. (Eng.) and Ph.D. from the Birla Institute of Technology (BIT), Ranchi, India, in 1968, 1970, and 1979, respectively. He joined BIT as a senior research fellow in 1970 and became an associate professor in 1980. While he was with BIT, he supervised a number of research projects in the area of microwave and plasma engineering. From 1983 to 1988, he was with the University of Dar es Salaam (UDSM), Tanzania, where he became a professor of telecommunications in the Department of Electrical Engineering in 1986. At UDSM, he was responsible for the collaborative project Satellite Communications for Rural Zones with Eindhoven University of Technology, the Netherlands. From February 1988 through May 1999, he was with the Telecommunications and Traffic Control Systems Group at DUT, where he was actively involved in the area of wireless personal and multimedia communications. He was the founding head and program director of the Center for Wireless and Personal Communications of the International Research Center for Telecommunications–Transmission and Radar. Since June 1999, Dr. Prasad has been with Aalborg University as the codirector of the Center for PersonKommunikation (CPK) and holds the chair of wireless information and multimedia communications. He was involved in the European ACTS project FRAMES (Future Radio Wideband Multiple Access Systems) as a DUT project leader. He is a project leader of several international industrially-funded projects. He has published more than 300 technical papers, contributed to several books, and has authored, coauthored, and edited nine books: *CDMA for Wireless Personal Communications; Universal Wireless Personal Communications; Wideband CDMA for Third Generation Mobile Communications; OFDM for Wireless Multimedia Communications; Third Generation Mobile Communication Systems; WCDMA: Towards IP Mobility and Mobile Internet; Towards a Global 3G System: Advanced Mobile Communications in Europe, Volumes 1 & 2;* and *IP/ATM Mobile Satellite Networks*, all published by Artech House. His current research interests lie in wireless networks, packet communications, multiple-access protocols, advanced radio techniques, and multimedia communications.

Dr. Prasad has served as a member of the advisory and program committees of several IEEE international conferences. He has also presented keynote speeches and delivered papers and tutorials on WPMC at various universities, technical institutions, and IEEE conferences. He was also a member of the European Cooperation in the Scientific and Technical (COST-231) research project dealing with the evolution of land mobile radio (including personal) communications as an expert for the Netherlands, and he was a member of the COST-259 project. He was the founder and chairman of the IEEE Vehicular Technology/Communications Society Joint Chapter, Benelux Section, and is now the honorary chairman. In addition, Dr. Prasad is the founder of the IEEE Symposium on Communications and Vehicular Technology (SCVT) in the Benelux, and he was the symposium chairman of SCVT'93.

In addition, Dr. Prasad is the coordinating editor and editor-in-chief of the *Kluwer International Journal on Wireless Personal Communications* and a member of the editorial board of other international journals, including the *IEEE Communications Magazine* and *IEE Electronics Communication Engineering Journal*. He was the technical program chairman of the PIMRC'94 International Symposium held at The Hague, the Netherlands, from September 19–23, 1994, and also of the Third Communication Theory Mini-Conference in conjunction with GLOBECOM'94 held in San Francisco, California, from November 27–30, 1994. He was the conference chairman of the Fiftieth IEEE Vehicular Technology Conference and the steering committee chairman of the Second International Symposium WPMC, both held in Amsterdam, the Netherlands, from September 19–23, 1999. He was the general chairman of WPMC'01, which was held in Aalborg, Denmark, from September 9–12, 2001.

Dr. Prasad is also the founding chairman of the European Center of Excellence in Telecommunications, known as HERMES. He is a fellow of IEE, a fellow of IETE, a senior member of IEEE, a member of The Netherlands Electronics and Radio Society, and a member of the Engineering Society in Denmark.

Index