國 立 交 通 大 學

電子工程學系 電子研究所碩士班

碩 士 論 文

IEEE 802.16e OFDMA實體層測距技術與數位訊號處理器實現之探討

Study in Ranging Techniques and Associated Digital Signal Processor Implementation for IEEE 802.16e OFDMA PHY

研 究 生：蔡昀澤

指導教授：林大衛 博士

中 華 民 國 九 十 七 年 六 月

IEEE 802.16e OFDMA 實體層測距技術與數位訊號處理器實現之探討

Study in Ranging Techniques and Associated Digital Signal Processor Implementation for IEEE 802.16e OFDMA PHY

研究生：蔡昀澤　　　　Student: Yun-Tze Tsai

指導教授：林大衛 博士　　　　Advisor: Dr. David W. Lin

國 立 交 通 大 學

電子工程學系　電子研究所碩士班

碩士論文

A Thesis

Submitted to Department of Electronics Engineering & Institute of Electronics

College of Electrical and Computer Engineering

National Chiao Tung University

in Partial Fulfillment of the Requirements

for the Degree of

Master of Science

in

Electronics Engineering

June 2008

Hsinchu, Taiwan, Republic of China

中華民國九十七年六月

IEEE 802.16e OFDMA 實體層測距 技術與數位訊號處理器實現之探討

研究生:蔡昀澤　　　　　　　　　　指導教授:林大衛 博士

國立交通大學

電子工程學系 電子研究所碩士班

摘要

本篇論文介紹 IEEE 802.16e 正交分頻多工存取(OFDMA)裡，測距(ranging)的問題、演算法、分析、以及實作方面的議題。

在 WIMAX 的規格裡，測距是一個很重要的程序。初始和週期測距是保證所有動態使用者的訊號能夠同步到達基地台的 2 個重要上行程序，如此上行正交分頻多工存取系統中子載波(subcarrier)的正交性得以維持。其中，週期測距讓行動台能調整傳輸參數以維持和基地台之間的上行通訊。在這篇論文中，我們只討論週期測距的細節及其演算法。然而，其他測距程序，如初始測距、頻寬要求及換手(handover)測距，將不會在此論文中詳談。

在週期測距程序裡，基地台需要偵測出每一個有傳送週期測距碼的使用者之測距碼，並作時間、功率偏移及也可能包括頻率偏移之估計。然後，基地台以廣播的方式回傳測距回報訊息(ranging response message)，其中包含需要做的校正及狀態通知。

我們使用頻域上的方法來完成測距碼的偵測以及時間偏移的估計。我們發現落在 -0.1 到 0.1 之間的頻率偏移並不會對效能造成明顯的影響，因此忽略頻率偏移的估計。我們對演算法作一些簡單的分析，並在可加性白色高斯雜訊通道(AWGN)和多路徑 Rayleigh 衰減通道下做模擬，並觀察其效能。

最後，我們把程式修改成定點運算的版本，將其實作到數位訊號處理器平台上，並盡可能利用一些最佳化技巧來加速測距函式的速度。我們提供了時脈次數(clock cycle)的模擬結果來說明我們的測距作業可以達到即時處理的要求。

Study in Ranging Techniques and Associated Digital Signal Processor Implementation for IEEE 802.16e OFDMA PHY

Student: Yun-Tze Tsai Advisor: Dr. David W. Lin

Department of Electronics Engineering
& Institute of Electronics
National Chiao Tung University

Abstract

In this thesis, we introduce the ranging problems, algorithms, analyses and implementation issues for IEEE 802.16e OFDMA PHY system.

Ranging is one of the significant processes in the mobile WiMAX standard. Initial and periodic ranging are two important uplink processes to ensure that the signals from all active users arrive at the BS synchronously so that the orthogonality among the subcarriers in the uplink of OFDMA systems is maintained. Periodic ranging allows the MS to adjust transmission parameters so that the MS can maintain uplink communication with the BS. In this thesis, we discuss about the details of periodic ranging and algorithms. In fact, there still exist other types of ranging process such as: initial ranging, bandwidth request ranging and handover ranging. However, periodic ranging is the only ranging process being discussed in this thesis.

In the periodic ranging process, the BS is required to detect different received ranging codes and estimate the timing, power and possibly frequency offset for each user that transmits a periodic ranging code. The BS then broadcasts a ranging response message (RNG-RSP) with needed adjustments and a status notification.

We employ a frequency domain method to accomplish the ranging code detection and timing offset estimation. We find that the frequency offset which was within the range of [-0.1,0.1] of subcarrier spacing did not cause an effect on the performance in evidence. Thus, we ignore the estimation of frequency offset. We perform some simple analyses of the ranging algorithm, simulate our ranging system in both AWGN and multipath Rayleigh fading channel and see the performance.

In the end, we modified the program to fixed-point version, implement them on the digital signal processor (DSP) platform and employ some optimization techniques to accelerate functions of ranging as fast as we can. Finally, some clock cycles simulation results were provided to show that the ranging task can achieve real-time requirements.

誌謝

本篇論文的順利完成，首先誠摯地感謝我的指導老師林大衛博士，感謝老師兩年來的細心指導，給予我在課業、研究上的幫助，使我學到了分析問題及解決問題的能力，同時，老師親切隨和的態度，也使我們能勇於發問，能夠勇於面對問題。在此，僅向老師致上最高的感謝之意。

另外要感謝的，是實驗室的洪崑健學長、吳俊榮學長及王海薇學姊。謝謝你們熱心地幫我解決了許多通訊方面相關的疑問，以及程式上的問題。

感謝通訊電子與訊號處理實驗室(commlab)，提供了充足的軟硬體資源，讓我在研究中不虞匱乏。感謝 94 級耀鈞、柏昇、依翎、順成四位學長的指導，以及 95 級光中、婉清、佳楓、威年、紹唐、尚諭等實驗室成員，大家一起努力，讓我的研究生涯充滿歡樂又有所成長。也期待大家畢業之後都能有不錯的發展。

最後，要感謝的是我的家人，他們的支持讓我能夠心無旁騖的從事研究工作。

謝謝所有幫助過我、陪我走過這一段歲月的師長、同儕與家人。謝謝！

誌於 2008.6 新竹交大

昀澤

Contents

List of Figures

List of Tables

Chapter 1

Introduction

The IEEE 802.16 Wireless Metropolitan Area Network (WirelessMAN) standard family provides specifications for an interface for fixed and mobile broadband wireless access systems. Among them, the IEEE 802.16-2004 has been proposed to provide last-mile connectivity to fixed locations by ratio links. There are four air interface specifications in 802.16-2004: WirelessMAN-SC, WirelessMAN-SCa, WirelessMAN-OFDM, and WirelessMAN-OFDMA.

Orthogonal frequency division multiple access (OFDMA) can be considered as OFDM based frequency division multiple access. It has been adopted as important physical layer techmiques in many specifications recently because of its good spectral efficiency, robustness in the multipath propagation environment, and capability to cope with inter-symbol interference (ISI).

An amendment to 802.16-2004, IEEE 802.16e-2005, considers mobility. It provides enhancement specifications to the 802.16-2004 to support mobile stations (MS) moving at vehicular speeds and thereby specify a combined fixed and mobile systems.

Our study is about the ranging techniques for the IEEE 802.16e OFDMA PHY. Ranging is a important process for an MS to acquire or adjust transmission parameters to initialize or maintain uplink communications with the base station (BS). The transmission parameters include timing, power, and possibly frequency offset. In this thesis, we use the term MS and subscriber station (SS) interchangeably.

The following shows the work that we have done.

- Study of ranging-related specifications in IEEE 802.16e.
- Development of the ranging algorithms for transmission and reception of ranging signals.
- Some simple analysis of the performance of ranging algorithms.
- Floating-point simulation in fixed and mobile channels.
- Implementation of the algorithms on Texas Instrument (TI)'s digital signal processor (DSP) employing the Code Composer Studio (CCS). We also use some techniques to accelerate the execution speed of the programs.

This thesis is organized as follows.

- The ranging techniques in the IEEE 802.16e WirelessMAN OFDMA standard are introduced in chapter 2.
- Chapter 3 introduces the ranging algorithms, and provides some analysis and simulation results.
- The DSP platform is introduced in chapter 4.
- Chapter 5 discusses the implementation of the ranging process on DSP platform.
- The conclusion and some future work are given in chapter 6.

Chapter 2

Ranging-Related Specifications in IEEE 802.16e OFDMA

The material in this chapter is largely taken from [1] and [2]. First, we give an overview of IEEE 802.16e OFDMA standard and introduce some basic idea regarding OFDMA. Then, we introduce the ranging technique in the specification. This is the main content of this chapter. Other contents in the specification are not our concern and are ignored in this introduction.

2.1 Introduction to IEEE 802.16e

The IEEE 802.16 standard specifies the air interface and MAC protocol for Wireless MAN. Part of it has been dubbed WiMAX (Worldwide Interoperability for Microwave Access) by an industry group called the WiMAX Forum. The mission of the Forum is to promote and certify compatibility and interoperability of broadband wireless products.

The first 802.16 standard was approved in December 2001. It is a standard for point to multi-point broadband wireless transmission in the 10–66 GHz band, with only a line-of-sight (LOS) capability. It uses a single carrier (SC) physical layer (PHY) technique.

To overcome the disadvantage of the LOS requirement, the 802.16a standard was approved in 2003 to support non-line-of-sight (NLOS) links, operational in both licensed and unlicensed frequency bands from 2 to 11 GHz. It was subsequently integrated with the original 802.16 and 802.16c standards to create the 802.16-2004 standard (initially code-named 802.16d). With such enhancements, the 802.16-2004 standard has been viewed as a promising alternative for providing the last-mile connectivity by radio link. However, the 802.16-2004 specifications were devised primarily for fixed wireless users.

Mobility enhancements are considered in IEEE 802.16e-2005, which was published as an amendment to IEEE 802.16-2004 in February 2006. It specifies four air interfaces: WirelessMAN-SC PHY, WirelessMAN-SCa PHY, WirelessMAN-OFDM PHY, and WirelessMAN-OFDMA PHY. This study is concerned with the OFDMA PHY uplink (UL) in a mobile communication environment.

2.2 Introduction to OFDMA

The basic idea of OFDMA is OFDM based frequency division multiple access (FDMA). In OFDM, a channel is divided into carriers which are used by one user at any time. In OFDMA, each user is provided with a fraction of available number of subcarriers, as shown in Figure 2.1.

OFDMA can also be described as a combination of frequency domain and time domain multiple access, where the resources are partitioned in the time-frequency space, and slots are assigned along the OFDM symbol index as well as OFDM subcarrier index [3].

OFDMA has many advantages. For example, cyclic prefix (CP) can deal with the ISI problem, simple equalization techniques (one-tap equalization) can be used, it provides robustness in the multipath propagation environment, and so on. Due to these many benefits, OFDMA is popular recently and has been adopted as the main physical layer techniques in

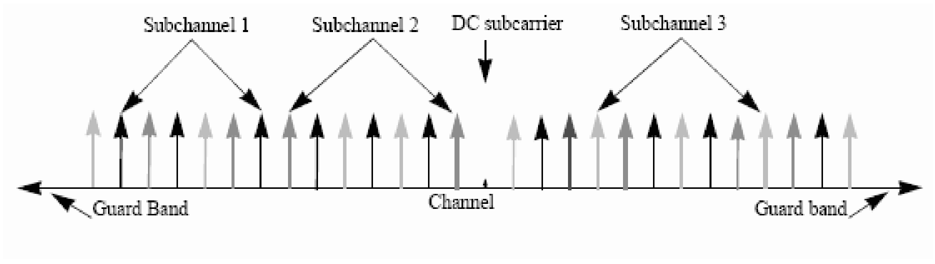


Figure 2.1: OFDMA frequency description (3-channel schematic example, from [1], Figure 214).

the IEEE 802.16e standard.

2.3 Definition of Basic Terms and Parameters [1], [2]

We present some basic concepts and definition of some parameters and basic terms of the 802.16e OFDMA PHY in this section.

2.3.1 Definition of OFDMA Basic Terms

In the OFDMA mode, the active subcarriers are divided into subsets of subcarriers, where each subset is termed a subchannel. The subcarriers forming one subchannel may, but need not be, adjacent. The concept is shown in Figure 2.1.

In the following, we introduce some basic terms in the OFDMA PHY.

Slot

A slot in the OFDMA PHY is a two-dimensional entity spanning both a time and a subchannel dimension. It is the minimum unit for data allocation. For downlink (DL) PUSC (Partial Usage of SubChannels), one slot is one subchannel by two OFDMA symbols. For uplink (UL) PUSC, one slot is one subchannel by three OFDMA symbols.

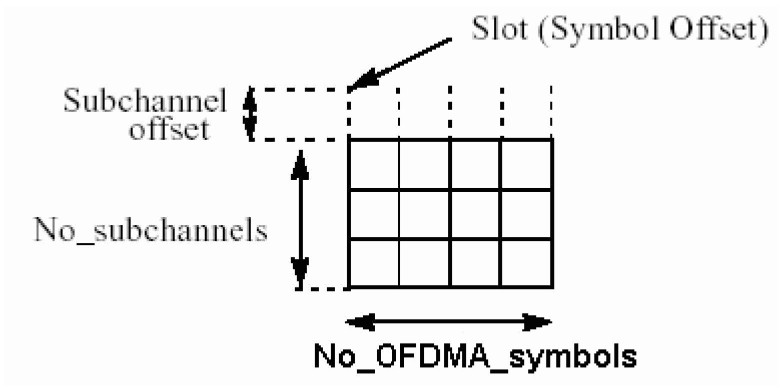


Figure 2.2: Example of the data region which defines the OFDMA allocation (from [1]).

Data Region

In OFDMA, a data region is a two-dimensional allocation of a group of contiguous subchannels, in a group of contiguous OFDMA symbols. All the allocations refer to logical subchannels. A two dimensional allocation may be visualized as a rectangle, such as the 4 $\times$ 3 rectangle shown in Fig. 2.2.

Segment

A segment is a subdivision of the set of available OFDMA subchannels (that may include all available subchannels). One segment is used for deploying a single instance of the MAC.

Permutation Zone

A permutation zone is a number of contiguous OFDMA symbols, in the DL or the UL, that use the same permutation formula. The DL subframe or the UL subframe may contain more than one permutation zone.

2.3.2 OFDMA Symbol Parameters

Primitive Parameters

The following parameters characterize the OFDMA symbols.

- BW: The nominal channel bandwidth.
- N_{used}: Number of used subcarriers (which includes the DC subcarrier).
- n: Sampling factor. Its value is set as follows: For channel bandwidths that are a multiple of 1.75 MHz, $n = 8/7$; else for channel bandwidths that are a multiple of any of 1.25, 1.5, 2 or 2.75 MHz, $n = 28/25$; else for channel bandwidths not otherwise specified, $n = 8/7$.
- G: This is the ratio of CP time to "useful" time, i.e., T_{cp}/T_s.

Derived Parameters

The following parameters are defined in terms of the primitive parameters.

- N_{FFT}: Smallest power of two greater than N_{used}.
- Sampling frequency: $F_s = \lfloor n \cdot BW/8000 \rfloor \times 8000$.
- Subcarrier spacing: $\triangle f = F_s/N_{FFT}$.
- Useful symbol time: $T_b = 1/\triangle f$.
- CP time: $T_g = G \times T_b$.
- OFDMA symbol time: $T_s = T_b + T_g$.
- Sampling time: T_b/N_{FFT}.

2.4 WirelessMAN OFDMA TDD Uplink [1], [2]

Since the ranging process is performed in the uplink, we only introduce some basic idea of the uplink and ignore the description of downlink. We describe the frame structure and subcarrier allocation briefly. For more details we refer the reader to [1] and [2].

2.4.1 Frame Structure

In licensed band, the duplexing method may be either frequency-division duplex (FDD) or time-division duplex (TDD). FDD SSs may be half-duplex FDD (H-FDD). In license-exempt band, the duplexing method should be TDD. The system that we implement is a TDD system.

When implementing a TDD system, the frame structure is built from BS and SS transmissions. Figure 2.3 shows an example. Each frame consists of a DL subframe and a UL subframe. The DL subframe begins with a preamble followed by frame control header (FCH), DL-MAP, UL-MAP and DL bursts. The UL subframe contains a ranging subchannel and UL bursts. In each frame, the transmit/receive transition gap (TTG) and receive/transmit transition gap (RTG) shall be inserted between the DL subframe and UL subframe and at the end of each frame, respectively, to allow transitions between transmission and reception functions.

The parts of the frame that are related to ranging process are the UL-MAP and the ranging subchannel. The relation will be described in later sections.

2.4.2 Subcarrier Allocation

The OFDMA PHY defines four scalable FFT sizes: 2048, 1024, 512, 128. Here we only introduce the 1024-FFT subcarrier allocation which is the most popular choice and is used

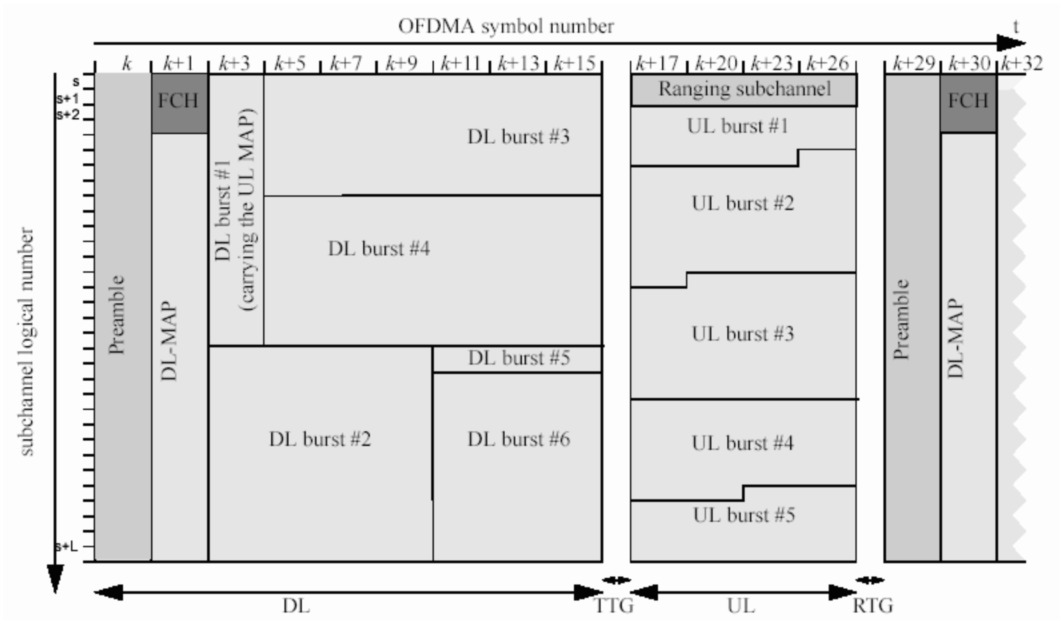


Figure 2.3: Example of an OFDMA frame (with only mandatory zone) in TDD mode (from [2]).

in our study.

An OFDMA symbol consists three types of subcarriers:

- Data subcarriers: For data transmission.
- Pilot subcarriers: For various estimation purposes.
- Null subcarriers: No transmission at all, for guard bands and including the DC subcarrier.

Subtracting the guard tones from N_{FFT}, one obtains the set of used subcarriers N_{used}. For both uplink and downlink, these used subcarriers are allocated to pilot subcarriers and data subcarriers. However, there is a difference between the different possible zones. In PUSC uplink which concern us in this study, each subchannel contains its own pilot subcarriers.

The general OFDMA subcarrier allocation scheme contain data mapping rules, carrier

Table 2.1: OFDMA Uplink Subcarrier Allocation [1], [2]

Parameter	Value	Notes
Number of DC subcarriers	1	Index 512 (counting from 0)
N_{used}	841	Number of all subcarriers used within a symbol
Guard subcarriers:	92,91	Left, right
TilePermutation		Used to allocate tiles to subchannels 11, 19, 12, 32, 33, 9, 30, 7, 4, 2, 13, 8, 17, 23, 27, 5, 15, 34, 22, 14, 21, 1, 0, 24, 3, 26, 29, 31, 20, 25, 16, 10, 6, 28, 18
$N_{subchannels}$	35	
$N_{subcarriers}$	48	
N_{tiles}	210	
Number of subcarriers per tile	4	Number of all subcarriers within a tile
Tiles per subchannel	6	

allocation and pilot modulation, but the detailed procedure is not related to our study. Therefore, we only introduce some basic ideas here.

The uplink supports 35 subchannels in the 1024-FFT PUSC permutation. Each transmission uses 48 data carriers as the minimal block of processing. Each new transmission for the uplink commences with the parameters as given in Table 2.1. A slot in the uplink is composed of one subchannel in three OFDMA symbols. Within each slot, there are 48 data subcarriers and 24 pilot subcarriers. The subchannel is constructed from six uplink tiles, each having four successive active subcarriers with the configuration as illustrated in Figure 2.4.

2.5 Ranging Process [1], [2]

There are four kinds of ranging process: initial ranging, periodic ranging, handover ranging and bandwidth-request ranging. The last two kinds are for purposes that are not of our

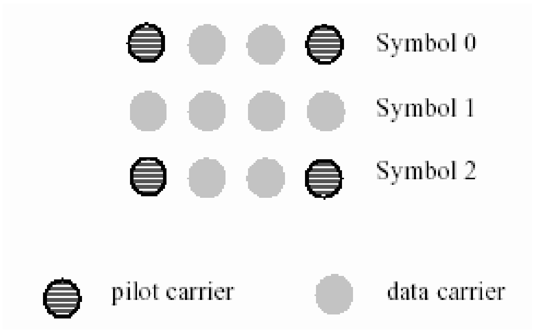


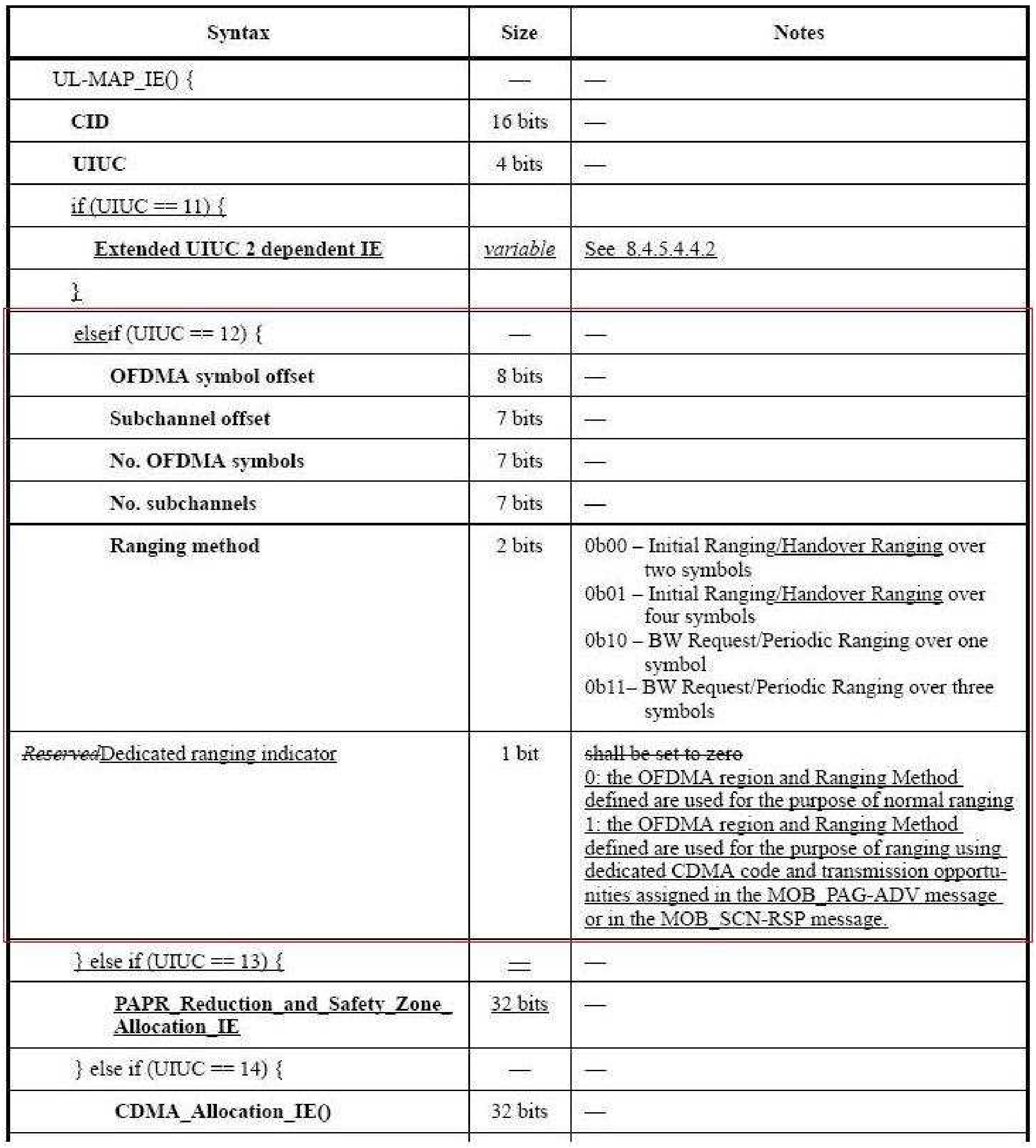
Figure 2.4: Structure of an uplink tile (from [1]).

interest. The first two kinds, initial and periodic ranging, are important uplink processes to ensure that the signals from all active users arrive at the BS synchronously, so that the orthogonality among the subcarriers in the uplink of OFDMA systems is maintained.

Initial ranging allows an MS joining the network to acquire correct transmission parameters, such as time offset and transmission power level, so that the MS can communicate with the BS. Following initial ranging, periodic ranging allows the MS to adjust transmission parameters so that the MS can maintain uplink communication with the BS. In this section, we mainly introduce periodic ranging. Note that initial ranging process is similar to periodic ranging.

During the periodic ranging procedure, the MS shall find an periodic ranging interval and acquire needed parameters from the UL-MAP message, as shown in Figure 2.5. Then, it shall transmit a randomly chosen frequency domain ranging code (CDMA codes) on the ranging channel in a randomly chosen ranging time slot.

Since more than one MS may choose the same time slot, the received ranging signal may include several MSs' ranging information. So, at the receiver side, the BS is required to detect different received ranging codes, estimate the timing offset, power level and possibly frequency offset of each user that transmits a periodic ranging code.



Syntax	Size	Notes
UL-MAP_IE() {	—	—
CID	16 bits	—
UIUC	4 bits	—
if (UIUC == 11) {		
Extended UIUC 2 dependent IE	*variable*	See 8.4.5.4.4.2
}		
elseif (UIUC == 12) {	—	—
OFDMA symbol offset	8 bits	—
Subchannel offset	7 bits	—
No. OFDMA symbols	7 bits	—
No. subchannels	7 bits	—
Ranging method	2 bits	0b00 – Initial Ranging/Handover Ranging over two symbols 0b01 – Initial Ranging/Handover Ranging over four symbols 0b10 – BW Request/Periodic Ranging over one symbol 0b11– BW Request/Periodic Ranging over three symbols
~~*Reserved*~~Dedicated ranging indicator	1 bit	~~shall be set to zero~~ 0: the OFDMA region and Ranging Method defined are used for the purpose of normal ranging 1: the OFDMA region and Ranging Method defined are used for the purpose of ranging using dedicated CDMA code and transmission opportunities assigned in the MOB_PAG-ADV message or in the MOB_SCN-RSP message.
} else if (UIUC == 13) {	—	—
PAPR_Reduction_and_Safety_Zone_Allocation_IE	32 bits	—
} else if (UIUC == 14) {	—	—
CDMA_Allocation_IE()	32 bits	—

Figure 2.5: The ranging region and information indicated in UL-MAP (from [2]).

Upon successfully receiving a periodic ranging code, the BS broadcasts a Ranging Response message (RNG-RSP) that advertises the received periodic ranging code as well as the ranging slot (OFDMA symbol number, subchannel, etc.) where the ranging code has been identified. This information is used by the MS that sent the ranging code to identify the ranging response message that corresponds to its ranging request. The RNG-RSP also contains all needed adjustment and a status notification. The status notification may be success, continue or abort. The MS can know what action should be done next according to the status. We refer the reader to [1] and [2] for more details about the periodic ranging process.

The main subjects of our work are ranging code transmission, ranging code detection and timing offset estimation.

2.6 OFDMA Ranging

The WirelessMAN OFDMA PHY specifies a ranging subchannel and a set of pseudonoise (PN) ranging codes. An example of ranging channel in OFDMA frame structure is specified in Figure 2.3. Subsets of codes should be allocated in the Uplink Channel Descriptor (UCD) Channel Encoding message for the four types of ranging, such that the BS can determine the purpose of the received code by the subset to which the code belongs.

A ranging channel is composed of one or more groups of six adjacent subchannels, using PUSC mode, where the groups are defined starting from the first subchannel. Optionally, a ranging channel can be composed of one or more groups of eight adjacent subchannels, using optional PUSC or adjacent subcarrier permutation mode. We use the former in our study since the system we build is based on the PUSC mode. Subchannels are considered adjacent if they have successive logical subchannel numbers. The indices of the subchannels that compose the ranging channel are specified in the UL-MAP message, as shown in Figure 2.5.

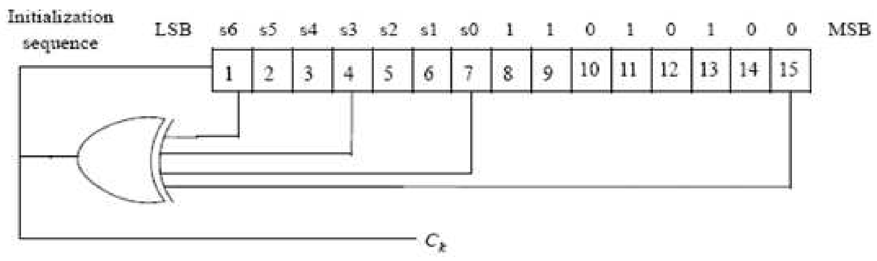


Figure 2.6: PRBS generator for ranging code generation (from [2]).

Users are allowed to collide on the ranging channel. To effect a ranging transmission, each user randomly chooses one ranging code from a bank of specified binary codes. These codes are then BPSK modulated onto the subcarriers in the ranging channel, one bit per subcarrier.

2.6.1 Ranging Codes [1], [2]

The binary ranging codes are the PN codes produced by the PRBS generator described in Figure 2.6, which implements the polynomial generator $1 + x^1 + x^4 + x^7 + x^{15}$. The PRBS generator is initialized by the seed b14..b0 = 0,0,1,0,1,0,1,1,s0,s1,s2,s3,s4,s5,s6 where s6 is the LSB of the PRBS seed, and s6:s0 = UL_PermBase, where s6 is the MSB of UL_PermBase.

The binary ranging codes are the subsequences of the PN sequence appearing at its output C_k. These bits are used to modulate the subcarriers in a group of six adjacent (logically) subchannels, which is the so-called ranging subchannel. The bits are mapped to the subcarriers in increasing frequency order of the subcarriers, such that the lowest indexed bit modulates the subcarrier with the lowest frequency index and the highest indexed bit modulates the subcarrier with the highest frequency index. The index of the lowest numbered subchannel in the six shall be an integer multiple of six.

The length of each ranging code is 144 bits. The first ranging code is obtained by clocking the PN generator 144 times as specified. The next code is produced by taking the output of the 145th to 288th clock instances of the PRBS generator, etc.

There are 256 available codes, numbered 0 to 255. Each BS uses a subgroup of these codes, where the subgroup is defined by a number S, $0 \leq S \leq 255$. The group of codes will be between S and $((S{+}O{+}N{+}M{+}L) \bmod 256)$.

- The first N codes produced are for initial ranging, obtained by clocking the PRBS generator $144 \times (S \bmod 256)$ times to $144 \times ((S + N) \bmod 256) - 1$ times.
- The next M codes produced are for periodic ranging, obtained by clocking the PRBS generator $144 \times ((N + S) \bmod 256)$ times to $144 \times ((N + M + S) \bmod 256) - 1$ times.
- The next L codes produced are for bandwidth (BW) requests, obtained by clocking the PRBS generator $144 \times ((N + M + S) \bmod 256)$ times to $144 \times ((N + M + L + S) \bmod 256) - 1$ times.
- The next O codes produced are for handover ranging, obtained by clocking the PRBS generator $144 \times ((N + M + L + S) \bmod 256)$ times to $144 \times ((N + M + L + O + S) \bmod 256) - 1$ times.

The values of S, N, M, L and O are provided in the UCD Channel Encoding message. Possible values are 0 to 255.

2.6.2 Cross-Correlation of Ranging Codes

As mentioned above, the binary ranging codes are generated by the PRBS generator shown in Figure 2.6. Then the codes are BPSK modulated onto the subcarriers in the ranging channel. We provide some analysis of the cross-correlation of the BPSK modulated ranging codes in this section.

Table 2.2: Cross-correlation Values of Ranging Codes

code	0	1	2	3	4	5	6	7	8	9	10	11	12	13
0	144	-8	6	14	-10	18	20	-10	0	-6	28	-8	0	-10
1		144	2	22	10	10	16	-2	20	14	16	-12	-8	6
2			144	12	16	-8	-6	4	14	16	6	6	-26	-16
3				144	8	4	6	8	6	-8	-14	14	-2	-8
4					144	-4	14	12	2	-12	-2	6	-2	12
5						144	22	0	-6	-4	14	-10	14	24
6							144	-2	12	-6	-4	12	-4	6
7								144	6	0	-2	30	-14	4
8									144	6	4	8	4	-14
9										144	-38	-22	-10	-16
10											144	-8	4	2
11												144	-20	2
12													144	-6
13														144

Since the length of each BPSK modulated code is 144 bits, the correlation values are between -144 and 144. Ideally, the code should be orthogonal and the cross-correlation of the codes should be zero. But this is not the case and we find that the codes do not have good orthogonality. We calculate the cross-correlation values of ranging code set for given UL_PermBase values. Table 2.2 to 2.4 shows some parts of the result for UL_PermBase=0. There are $256 \times 256 \div 2 = 32640$ values since there are a total of 256 codes. Since the total table is too large, we only depict some parts of it.

Figures 2.7, 2.8 and 2.9 show the histograms of code cross-correlation values for UL_PermBase=0, 30 and 60, respectively. We can conclude that the cross-correlation values are mainly between -20 and 20 and their distribution is little affected by different UL_PermBase values.

We also calculate the sample mean and sample variance of the cross correlation values, which are -0.039 and 143.37, respectively.

Table 2.3: Cross-correlation Values of Ranging Codes (Continued)

code	14	15	16	17	18	19	20	21	22	23	24	25	26	27
0	16	-10	8	-4	8	2	-2	-20	-10	14	-2	8	-8	-6
1	-20	22	20	12	-8	-10	-2	12	-6	10	6	16	-8	6
2	-6	-8	6	26	-2	-12	4	6	-8	0	-4	-6	6	0
3	-6	-12	-2	14	-2	-12	8	2	4	-4	4	2	18	-8
4	-2	-12	-2	22	-6	12	-20	-10	-4	-8	-8	10	18	8
5	6	0	6	-2	6	16	-8	6	4	-4	32	-6	-6	4
6	-8	14	4	12	4	10	-2	-4	-2	2	14	12	4	-2
7	-6	-4	-2	6	-18	8	-24	2	-8	-20	-4	6	10	-24
8	-24	2	-4	28	-8	-2	6	8	-18	-6	-14	-4	0	6
9	-14	-8	10	-2	-14	-8	8	-22	-32	8	4	-2	14	-12
10	16	2	4	-8	4	-2	10	-8	14	6	-2	-24	-8	6
11	0	6	-16	12	-16	26	-2	-16	10	-14	10	16	-4	6
12	20	6	0	-16	-16	14	14	0	-6	10	6	-4	4	2
13	-2	4	2	-2	2	20	-12	22	0	-16	8	-2	-2	0
14	144	22	4	-4	0	-18	18	-12	-6	30	6	-20	-12	-2
15		144	2	10	18	8	8	10	12	8	12	6	-14	0
16			144	12	-32	-18	-2	4	-6	-10	10	24	-4	-22
17				144	-24	-6	-22	-8	-6	-14	-6	12	-8	10
18					144	6	14	4	2	-10	-10	-12	-8	6
19						144	-32	10	8	-12	12	14	-2	0
20							144	10	-8	4	-4	-10	18	-12
21								144	6	6	2	4	4	-2
22									144	-4	12	-6	-6	20
23										144	0	-6	10	0
24											144	-2	-6	-4
25												144	4	-2
26													144	-10
27														144

Table 2.4: Cross-correlation Values of Ranging Codes (Continued)

code	28	29	30	31	32	33	34	35	36	37	38	39	40	41
0	12	0	-10	-16	-4	-10	-16	-4	10	10	2	2	-10	4
1	-4	-8	2	-16	0	-6	16	0	2	-10	-6	-2	10	0
2	10	30	4	10	-2	-20	-2	14	20	-4	0	-4	-8	2
3	14	-2	4	-6	-6	12	6	-14	-4	8	4	12	4	-2
4	26	10	-8	-14	6	0	-2	18	4	-4	-12	24	-20	10
5	-6	-42	-28	-2	14	16	14	2	0	8	12	0	12	-14
6	-8	-8	-22	-12	4	6	8	8	-10	-18	2	-10	10	16
7	14	18	12	-10	-10	-4	2	-6	-4	0	-4	-4	-4	-10
8	-4	12	10	-12	-8	-6	-4	-4	14	26	-6	-2	10	-8
9	6	2	-4	-6	6	-8	-14	6	0	4	-4	0	-8	10
10	-24	16	10	24	-8	-2	24	-4	2	10	2	-2	2	12
11	0	16	2	-16	-4	10	0	8	-14	6	-2	-6	14	12
12	8	-12	-10	-4	20	-2	-8	-20	6	6	-6	-10	2	-36
13	10	-6	-24	-18	-22	20	18	6	-4	-16	12	16	4	2
14	8	-16	-2	16	-16	10	12	-40	-2	10	-10	-22	-18	-12
15	2	-26	20	-2	-10	0	2	-10	-16	0	-12	-16	0	-22
16	-12	-8	-6	20	-4	-18	-4	16	-14	-18	-22	-2	-2	12
17	20	12	-6	20	12	-18	-16	0	-2	-10	-6	-6	-10	12
18	4	-12	-6	4	4	-10	-20	0	10	10	18	2	14	-12
19	14	-10	-20	-14	14	0	-14	18	-4	-4	12	16	20	-14
20	-10	10	8	-10	10	-4	2	-10	12	0	-4	-28	-4	6
21	0	4	-10	0	0	2	-4	8	18	-18	10	-2	-14	-8
22	-6	-10	4	-2	-2	12	22	-10	-4	16	12	-4	-4	10
23	6	10	-16	14	-10	16	-6	-6	0	-8	-4	-12	-4	-18
24	-18	-26	-8	2	14	8	10	6	-20	4	-4	-8	12	-2
25	12	-4	-14	0	12	-18	4	-8	-6	-10	-10	6	14	8
26	8	20	-10	-4	-12	-2	0	16	10	-10	-6	6	-10	-4
27	-6	-10	-4	-2	10	-12	6	-26	12	12	0	12	-8	10
28	144	4	2	-12	-4	-14	-8	-8	18	-18	2	-2	-26	-12
29		144	14	0	32	-14	-28	-4	18	-2	-2	-6	-22	28
30			144	-6	6	-16	2	2	-4	12	12	-24	0	-6
31				144	-4	2	-32	4	-2	10	14	2	18	4
32					144	-6	-4	-12	14	-2	6	10	14	0
33						144	-2	18	-16	-4	8	4	-8	2
34							144	4	-18	2	-14	2	14	0
35								144	-10	-22	2	26	-6	8
36									144	8	4	12	20	-2
37										144	-8	8	12	-2
38											144	-12	4	-10

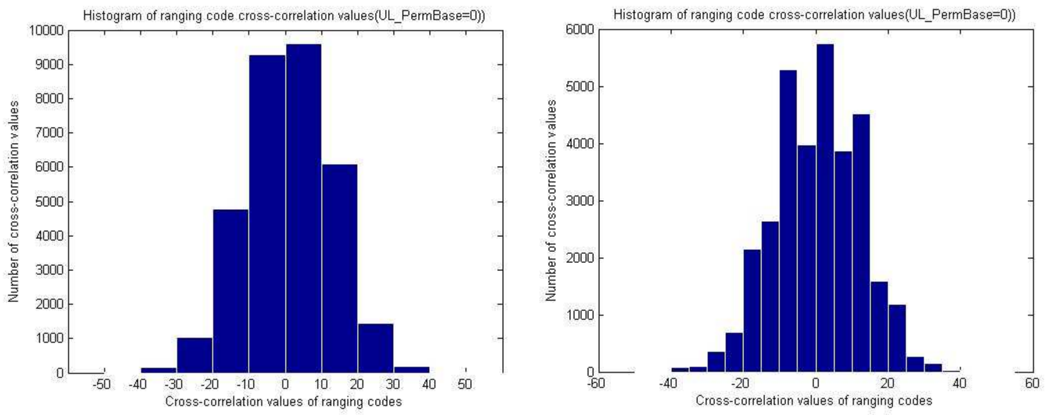


Figure 2.7: Histograms of ranging code cross-correlation values for UL_PermBase=0).

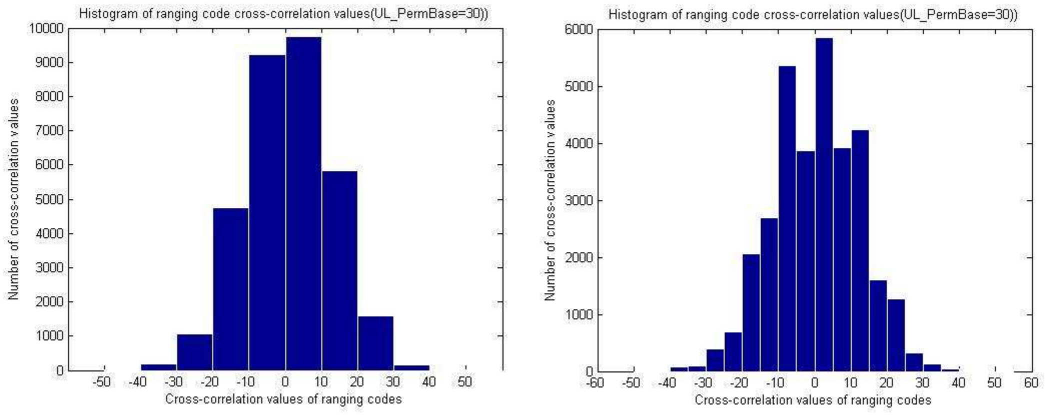
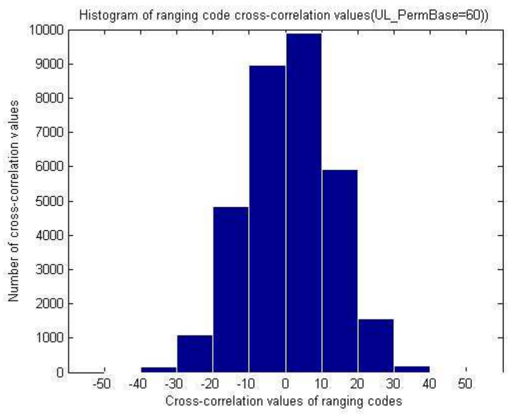


Figure 2.8: Histograms of ranging code cross-correlation values for UL_PermBase=30).



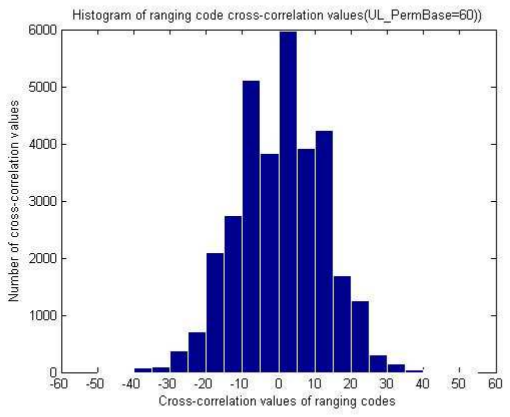


Figure 2.9: Histograms of ranging code cross-correlation values for UL_PermBase=60).

2.6.3 Initial-Ranging and Handover-Ranging Transmissions [1], [2]

The initial ranging codes are used for any MS that wants to synchronize to the system for the first time. Handover ranging codes are used for ranging against a target BS during handover.

An initial-ranging transmission is performed using two or four consecutive symbols. In the former case, The same ranging code is transmitted on the ranging channel during each symbol, with no phase discontinuity between the two symbols. A time-domain illustration is shown in Figure 2.10. In the latter case, the BS can allocate two consecutive initial-ranging/handover-ranging slots, onto which the MS transmits two consecutive initial-ranging/handover-ranging codes where the starting code shall always be a multiple of 2. A time-domain illustration is shown in Figure 2.11.

The choice of the two transmission ways is indicated in the UL-MAP, as shown in Figure 2.5.

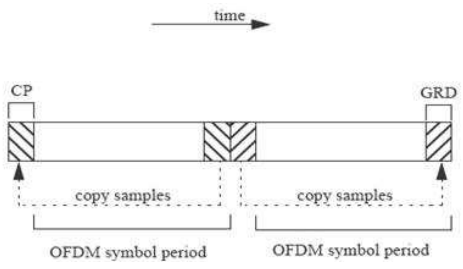


Figure 2.10: Initial-ranging/handover-ranging transmission for OFDMA (from [2]).

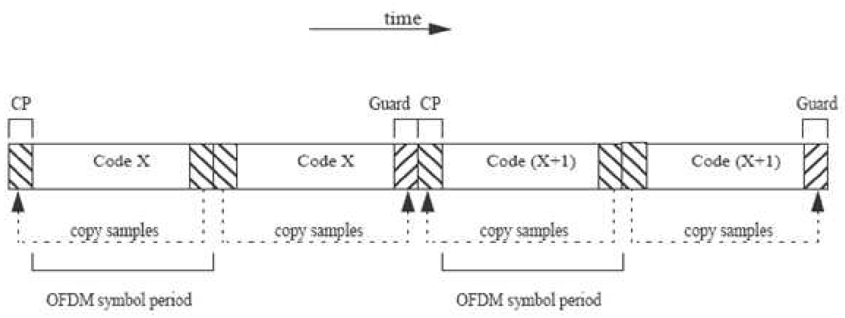


Figure 2.11: Initial-ranging/handover-ranging transmission for OFDMA, using two consecutive initial ranging codes (from [2]).

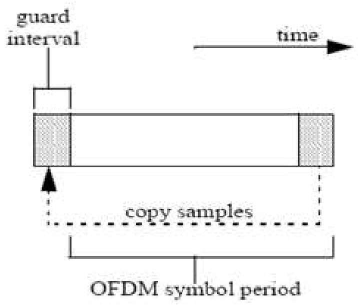


Figure 2.12: Periodic-ranging or bandwidth-request transmission for OFDMA using one code (from [2]).

2.6.4 Periodic Ranging/Bandwidth-Request Ranging Transmissions [1], [2]

Periodic ranging transmissions are sent periodically for system periodic ranging. Bandwidth-request transmissions are for requesting uplink allocations from the BS. These transmissions shall be sent only by an MS that has already synchronized to the system.

To perform either a periodic-ranging or bandwidth-request transmission, the MS can send a transmission in one of two ways: Modulate one ranging code on the ranging subchannel for a period of one OFDMA symbol (a time-domain illustration is shown in Figure 2.12), or modulate three consecutive ranging codes (the starting code shall always be a multiple of 3) on the ranging subchannel for a period of three OFDMA symbols, one code per symbol (a time-domain illustration is shown in Figure 2.13).

The choice of the two transmission ways is indicated in the UL-MAP, as shown in Figure 2.5. Here we choose the former in our study.

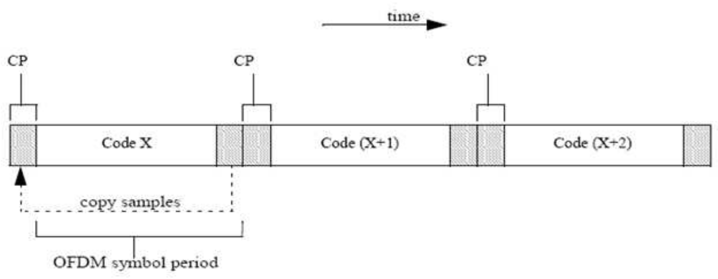


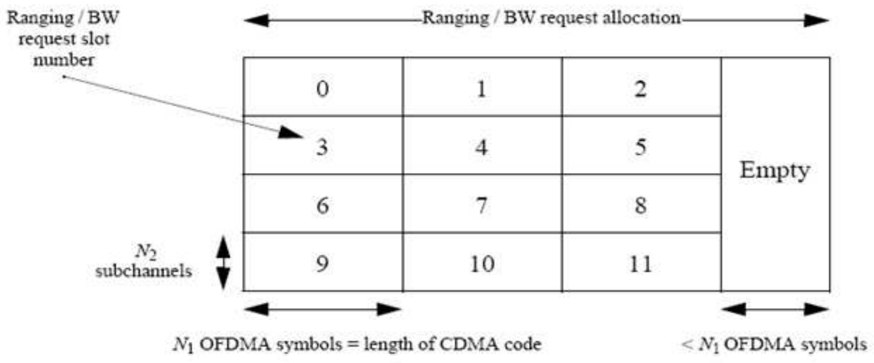
Figure 2.13: Periodic-ranging or bandwidth-request transmission for OFDMA using three consecutive codes (from [2]).

2.6.5 Ranging and BW Request Opportunity Size [1], [2]

For CDMA ranging and BW request, the ranging opportunity size is the number of symbols required to transmit the appropriate ranging or BW request code (1, 2, 3 or 4 symbols), and is denoted N_1. N_2 denotes the number of subchannels required to transmit a ranging code. In each ranging/BW request allocation, the opportunity size (N_1) is fixed and conveyed by the corresponding UL_MAP_IE that defines the allocation.

As shown in Figure 2.14, the ranging allocation is subdivided into slots of N_1 OFDMA symbols by N_2 subchannels, in a time first order, i.e., the first opportunity begins on the first symbol of the first subchannel of the ranging allocation, the next opportunities appear in ascending time order in the same subchannel, until the end of the ranging/BW request allocation (or until there are less than N_1 symbols in the current subchannel), and then the number of subchannel is incremented by N_2. The ranging allocation is not required to be a whole multiple of N_1 symbols, so a gap may be formed (that can be used to mitigate interference between ranging and data transmissions). Each CDMA code will be transmitted at the beginning of the corresponding slot.

In our study, we consider 1-symbol periodic ranging transmission so that $N_1 = 1$. And we consider the ranging channel composed of 6 subchannels, so $N_2 = 2$.



Ranging / BW request slot number
Ranging / BW request allocation
0
1
2
3
4
5
6
7
8
9
10
11
Empty
N_2 subchannels
N_1 OFDMA symbols = length of CDMA code
< N_1 OFDMA symbols

Figure 2.14: Ranging/BW request opportunities (from [2]).

Chapter 3

Ranging Techniques for IEEE 802.16e OFDMA

In this chapter, we discuss the ranging techniques for the IEEE 802.16e system. Some existing initial ranging methods have been described in [4]. Here we describe our ranging algorithm, present some analysis and show some simulation results.

3.1 Ranging Signal Processing

Figure 3.1 shows the overall uplink transceiver system. The ranging signal generator and receiver blocks are our main concern. Ranging operation is performed in the frequency domain, that is, before the transmitter IFFT process and after the receiver FFT process.

The algorithms of initial ranging and periodic ranging are similar, but we focus on periodic ranging here.

Ranging Signal Generator

Figure 3.2 shows the structure of the ranging signal generator. The tasks here are simple and straightforward. The PRBS generator, whose structure is shown in Figure 2.6, generates the binary ranging code according to the ranging information. The ranging information includes

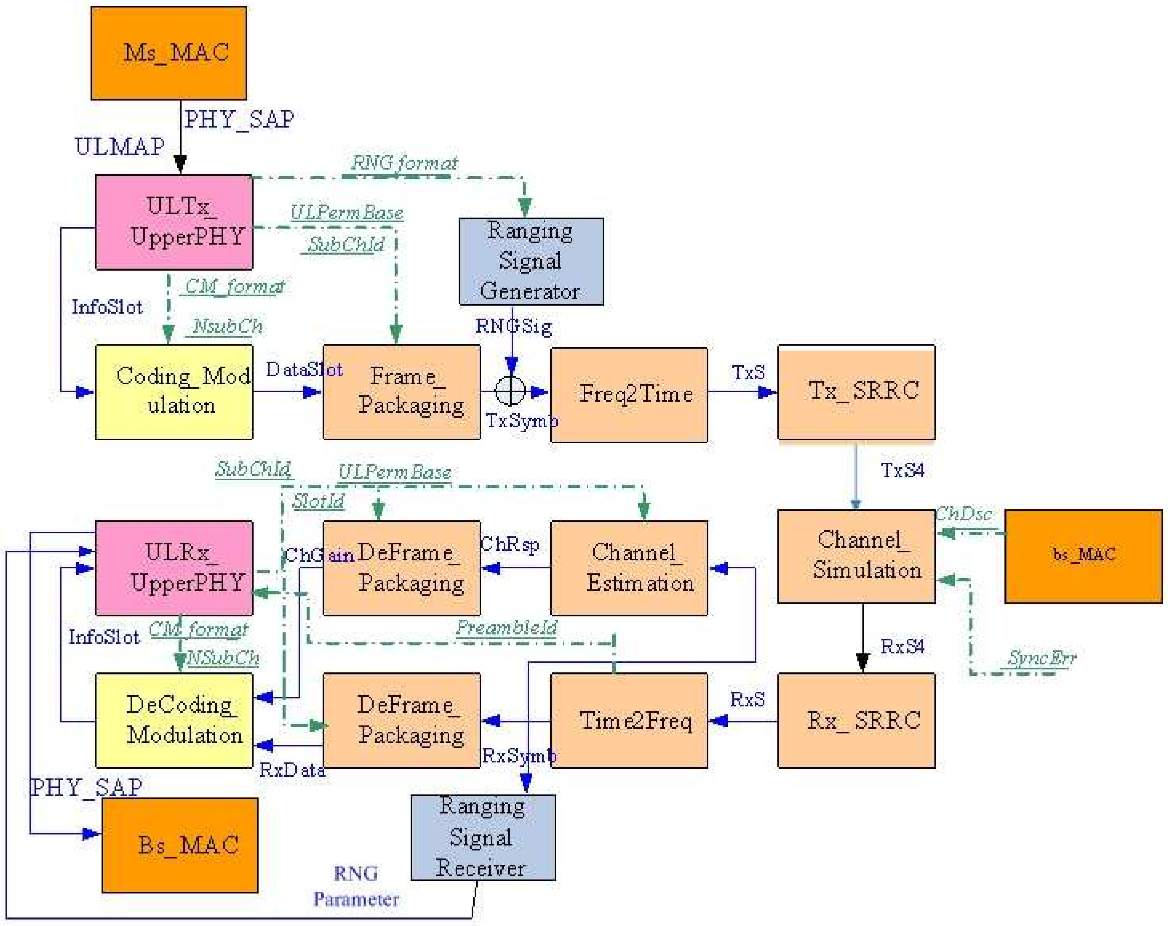


Figure 3.1: Uplink transceiver system.

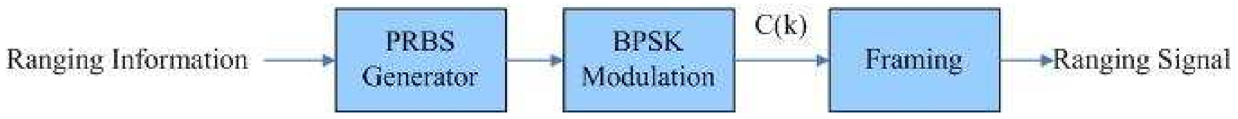


Figure 3.2: Ranging signal generator.

parameters S, N, M, UL_PermBase, subchannel offset and used code number. Then the code is BPSK modulated. Finally, some permutation operation is performed to map the BPSK modulated code to the subcarriers.

Ranging Signal Receiver

The main task here are ranging code detection and timing offset estimation. We do not perform frequency offset estimation in the periodic ranging process. The reason will be described later.

Figure 3.3 depicts the overall structure of the ranging signal receiver system. The inputs to the ranging signal receiver are the ranging information and $R(k)$, which is the received signal after the FFT. The output is a ranging report containing the information of detected codes and estimated timing offsets.

The ranging signal reception is performed on the frequency domain signal $R(k)$. First, the ranging subcarrier selector extracts subcarrier values on which a ranging code may be loaded. Since the length of a ranging code is 144 bits, the length of $S(k)$ is 144 subcarriers. The ranging code generator generates all possible periodic ranging codes. The next step is multiplication of $S(k)$ by each possible ranging code $C_i(k)$. The length of the output $M(k)$ is also 144. After the multiplication, we perform 1024-point IFFT. Since the length of $M(k)$ is only 144, 880 zeros are inserted to the frequency positions before IFFT. The result after IFFT, $U(m)$, consists of 1024 complex samples. Then the norm operation calculates the norm of $U(m)$. Finally, the norm values are used for code detection and timing offset

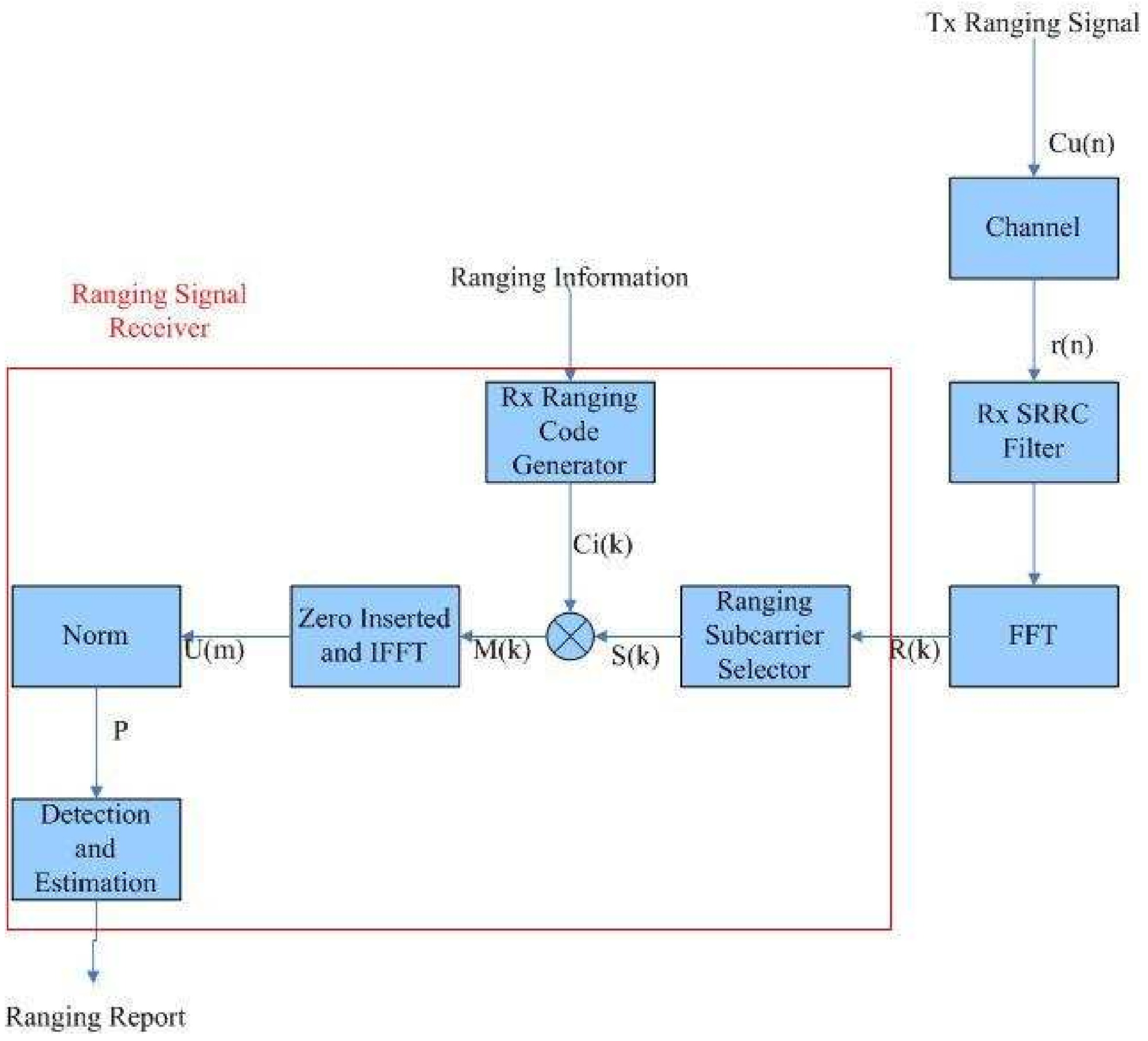


Figure 3.3: Ranging signal receiver.

estimation. The operation is repeated for each possible periodic ranging code.

The above frequency domain ranging reception algorithm is equivalent to time domain cross correlation. The time domain cross correlation needs 1024 $\times$ 1024 complex multiplications. In our algorithm, using IFFT to perform the equivalent operation needs only $1024 \times \log_2 1024$ complex multiplications, which is only 1% of that in time domain cross correlation [5].

The basic concept of our algorithm is shown bellow, assuming presence of additive noise. From DFT theory, we know that a time-domain signal and its DFT are realted by

$$x[((n-d_u))_N] \longleftrightarrow W_N^{kd_u} \cdot X[k]. \tag{3.1}$$

The received time-domain signal after an additive noise channel is:

$$r(n) = C_u(n-d_u) + w_A \cdot w(n). \tag{3.2}$$

Then the frequency-domain signal after FFT is:

$$R(k) = W_N^{kd_u} \cdot C_u(k) + W(k) \tag{3.3}$$

where $W_N = e^{-j2\pi/N}$ (here N=1024), $w_A \cdot w(n)$ is the added Gaussian noise in the time-domain, $W(k)$ is the equivalent noise in the frequency domain and d_u is the delay.

The signal after IFFT is:

$$\begin{aligned} U(m) &= \frac{1}{\sqrt{N}} \sum_k S(k) \cdot C_i(k) \cdot W_N^{-km} \\ &= \frac{1}{\sqrt{N}} \sum_k [W_N^{kd_u} \cdot C_u(k) \cdot C_i(k) \cdot W_N^{-km} + W(k) \cdot C_i(k) \cdot W_N^{-km}]. \end{aligned} \tag{3.4}$$

The results after the norm operation are 1024 norm values for each possible delay m, which may be 0 to 1023 samples. The peak value occurs when $C_i(k) = C_u(k)$ and $m = d_u$. More detailed derivation will be provided later.

The detection and estimation operation contains two steps. First, we decide whether the ranging code is transmitted or not. The decision can be made by two methods. One simple method is setting a single predetermined threshold, h_4. If there are any norm value greater than h_4, we say that this ranging code has been transmitted. We denote this method as method 2. Another method uses three thresholds H, h_1 and h_2, where h_1 is greater than h_2. We calculate the ratio of the mean of norm values greater than h_1 to the mean of norm values smaller than h_2. If this ratio (h_1/h_2) is greater than H, we say that this ranging code has been transmitted [6]. We denote this method as method 1. The choice of these thresholds and comparison of the two methods are left to later sections.

Second, we perform timing offset estimation when a user is detected, that is, we decide the arrival time of the ranging code. By inspection, we can choose the location of the maximum norm value as the estimated timing offset. But this is not proper in multipath channels. Therefore we use another approach. We let the maximum value of the norm be divided by a number (h_t), then we get a new threshold. We choose the first location of norm greater than this threshold as the estimated timing offset.

3.2 Analysis of the Ranging Algorithm

In this section, we provide some performance analysis of ranging method 2. We do not analyze the performance of method 1 since it involves three thresholds and is too complex. We consider additive white Gaussian noise (AWGN) channel and single-path Rayleigh fading channel. We let the channel link power be 0 dB and let d_u denote the delay. We only consider the case that only one user is doing periodic ranging at this ranging slot. In addition, we assume that timing offset estimation is correct and derive the success rate of ranging code detection. Note that there exists other analysis described in [7].

3.2.1 Case of AWGN Channel

The received time-domain signal after the channel is as given in (3.2), where $w_A = \sqrt{\frac{1}{SNR}}$ and

$$w(n) = w_r(n) + j \cdot w_i(n) \tag{3.5}$$

with $w_r(n)$ and $w_r(n)$ both being Gaussian distributed with zero mean and unity variance. So $w_A \cdot w_r(n)$ and $w_A \cdot w_r(n)$ are also Gaussian distributed with zero mean and variance $\frac{1}{SNR}$.

Then the frequency-domain signal after FFT is as in (3.3), where

$$\begin{aligned} W(k) &\equiv W_r(k) + j \cdot W_i(k) \\ &= \sum_n w_A \cdot w(n) \cdot W_N^{kn} \\ &= \sum_n w_A \cdot w_r(n) \cdot W_N^{kn} + j \cdot w_A \cdot w_i(n) \cdot W_N^{kn}. \end{aligned} \tag{3.6}$$

So $W_r(k)$ and $W_i(k)$ are Gaussian distributed with zero mean and variance $\frac{1}{SNR}$.

The result after IFFT is as given in (3.4). When the multiplied ranging code matches the transmitted one, the result after IFFT should have a peak value, denoted D_p. When the multiplied code does not match the transmitted one, the value is denoted I_p. Therefore, the success rate of ranging code detection is

$$P(\text{successful detection}) = P(|D_p| > \sqrt{h_4} \text{ and } |I_p| < \sqrt{h_4}) \approx P(|D_p| > \sqrt{h_4}) \cdot P^{M-1}(|I_p| < \sqrt{h_4}). \tag{3.7}$$

The former probability is the rate of correct detection and the latter one is the product of the rate of correct rejection. Note that we make an approximation that correct detection and correct rejection are independent. In the following, we derive these two probabilities.

Correct Detection

When the multiplied ranging code matches the transmitted one,

$$D_p = \frac{1}{\sqrt{N}} \sum_k |C_u(k)|^2 + C_u(k) \cdot W(k) \equiv R_{Dp} + j \cdot I_{Dp} \tag{3.8}$$

where

$$R_{Dp} = \frac{1}{\sqrt{N}} \{\sum_k |C_u(k)|^2 + \sum_k C_u(k) \cdot W_r(k)\} \tag{3.9}$$

and

$$I_{Dp} = \frac{1}{\sqrt{N}} \sum_k C_u(k) \cdot W_i(k). \tag{3.10}$$

Since $\sum_k |C_u(k)|^2 = 144$ and $W_r(k)$ and $W_i(k)$ are Gaussian distributed with zero mean and variance $\frac{1}{SNR}$, R_{Dp} is Gaussian distributed with mean $\frac{144}{\sqrt{1024}} = 4.5$, and variance $\frac{\sigma_r^2}{1024}$. Similarly, I_{Dp} is Gaussian distributed with zero mean and variance $\frac{\sigma_i^2}{1024}$. Here $\sigma_r^2 = \sigma_i^2 = \frac{144}{2 \cdot SNR}$.

From [8], we know that $R = \sqrt{x^2 + y^2} \sim Ricean(\nu, \sigma)$ if $x \sim N(\nu \cos\theta, \sigma^2)$ and $y \sim N(\nu \sin\theta, \sigma^2)$ are two independent Gaussian distributions and θ is any real number. Here if we let $\theta = 0$, $\nu = 4.5$ and $\sigma = \frac{\sigma_r}{\sqrt{1024}}$, then

$$|D_p| = \sqrt{R_{Dp}^2 + I_{Dp}^2} \sim Ricean(\nu, \sigma) \tag{3.11}$$

with $K = \frac{\nu^2}{2\sigma^2}$. We can further assume $K \gg 1$, which is reasonable in our case, so $|D_p|$ is approximately Gaussian distributed with mean ν and variance σ^2.

Finally, the desired probability can be calculated by

$$P(|D_p| > \sqrt{h_4}) = P(z = \frac{|D_p| - \nu}{\sigma} > \frac{\sqrt{h_4} - 4.5}{\sigma}), \tag{3.12}$$

where the value of σ is determined by SNR.

Correct Rejection

When the multiplied code does not match the transmitted one,

$$\begin{aligned} I_p &\equiv R_{Ip} + j \cdot I_{Ip} \\ &= \frac{1}{\sqrt{N}} \sum_k [C_u(k) \cdot C_i(k) + C_i(k) \cdot W(k)] \\ &= \frac{1}{\sqrt{N}} \sum_k [C_u(k) \cdot C_i(k) + C_i(k) \cdot W_r(k) + C_i(k) \cdot W_i(k)]. \end{aligned} \tag{3.13}$$

The first summation $\sum_k C_u(k) \cdot C_i(k)$, which is the cross correlation of ranging codes, is often nonzero as shown in section 2.6.2. Therefore we need to consider the effect of this cross correlation term. A simple way is to approximate it as a constant, for instance, 10 or -10. Another better approach is to approximate it as a random variable. We take the latter approach. We assume that the cross-correlation term is a Gaussian random variable. Its mean and variance are approximated by the sample mean and variance calculated in section 2.6.2, which are -0.039 and 143.37 respectively. That is, let

$$\sum_k C_u(k) \cdot C_i(k) \sim N(\mu_c, \sigma_c^2) \tag{3.14}$$

where $\mu_c = -0.039$ and $\sigma_c^2 = 143.37$. Then, R_{Ip} is sum of two Gaussian random variable and thus is also Gaussian:

$$\begin{aligned} R_{Ip} &= \frac{1}{\sqrt{N}} \sum_k C_u(k) \cdot C_i(k) + \frac{1}{\sqrt{N}} \sum_k C_i(k) \cdot W_r(k) \\ &\sim N(\mu_1, \sigma_1^2) \\ &\sim N(\frac{\mu_c}{\sqrt{1024}}, \frac{\sigma_c^2 + \sigma_r^2}{1024}) \\ &\sim N(-0.012, \frac{143.37 + \sigma_r^2}{1024}). \end{aligned} \tag{3.15}$$

And I_{Ip} has the same distribution as I_{Dp}.

$$\begin{aligned} I_{Ip} &= \frac{1}{\sqrt{N}} \sum_k C_i(k) \cdot W_i(k) \\ &\sim N(\mu_2, \sigma_2^2) \sim N(0, \frac{\sigma_i^2}{1024}). \end{aligned} \tag{3.16}$$

Here we need the probability density function (PDF) of $|I_p|^2 = R_{Ip}^2 + I_{Ip}^2$, which is the sum of squares of two Gaussian random variables with different variances. There are two approaches to achieve it. The first is to approximate it by a multiple of a chi-squared distribution that has the correct mean and variance [9]. The second is by transformation of random variables. We take the second approach.

Let

$$|I_p| = y_1 + y_2 = R_{Ip}^2 + I_{Ip}^2. \tag{3.17}$$

We calculate the PDFs of y_1 and y_2, respectively. Assume that $\mu_1 \approx 0$. Then

$$\begin{aligned} f(y_1 = R_{Ip}^2) &= \frac{1}{\sqrt{2\pi}\sigma_1} \exp\{-\frac{(-\sqrt{y_1} - \mu_1)^2}{2\sigma_1^2}\} \cdot |\frac{-1}{2\sqrt{y_1}}| \\ &+ \frac{1}{\sqrt{2\pi}\sigma_1} \exp\{-\frac{(\sqrt{y_1} - \mu_1)^2}{2\sigma_1^2}\} \cdot |\frac{1}{2\sqrt{y_1}}| \\ &= \frac{1}{\sqrt{2\pi}\sigma_1\sqrt{y_1}} \exp\{-\frac{y_1}{2\sigma_1^2}\}. \end{aligned} \tag{3.18}$$

$$f(y_2 = I_{Ip}^2) = ... = \frac{1}{\sqrt{2\pi}\sigma_2\sqrt{y_2}} \exp\{-\frac{y_2}{2\sigma_2^2}\}. \tag{3.19}$$

Then, let $y = y_1 + y_2$ and $w = y_1$ and calculate the joint pdf of y and w.

The corresponding Jacobian is given by

$$J = \begin{vmatrix} 1 & -1 \\ 0 & 1 \end{vmatrix} = 1. \tag{3.20}$$

So

$$f(y,w) = f(y_1 = w, y_2 = y - w) \cdot |J| = f(y_1 = w, y_2 = y - w). \tag{3.21}$$

Hence

$$\begin{aligned} f(y) &= \int_0^y f(y,w)dw \\ &= \frac{1}{2\pi\sigma_1\sigma_2}\int_0^y \frac{1}{\sqrt{w}}\exp\{-\frac{w}{2\sigma_1^2}\}\cdot\frac{1}{\sqrt{y-w}}\exp\{-\frac{y-w}{2\sigma_2^2}\}dw. \end{aligned} \tag{3.22}$$

The parameters σ_1 and σ_2 are determined by (3.15) and (3.16).

Finally, the desired probability can be calculated by

$$P(y = |I_p|^2 < h_4) = \int_0^{h_4} f(y)dy. \tag{3.23}$$

Numerical Results

Figure 3.4 show some numerical results of the above analysis compared with the floating-point simulation results.

3.2.2 Case of Single-Path Rayleigh Fading Channel

The time domain received signal after the channel is

$$r(n) = g_F \cdot C_u(n - d_u) + w_A \cdot w(n) \tag{3.24}$$

where w_A and $w(n)$ are the same as that in AWGN case and The fading gain g_F is complex Gaussian with Rayleigh envelope whose PDF is given by

$$f(|g_F| = g) = \frac{g}{\sigma_g^2}\exp\{-\frac{g^2}{2\sigma_g^2}\}, \tag{3.25}$$

with $\sigma_g^2 = 0.4146$.

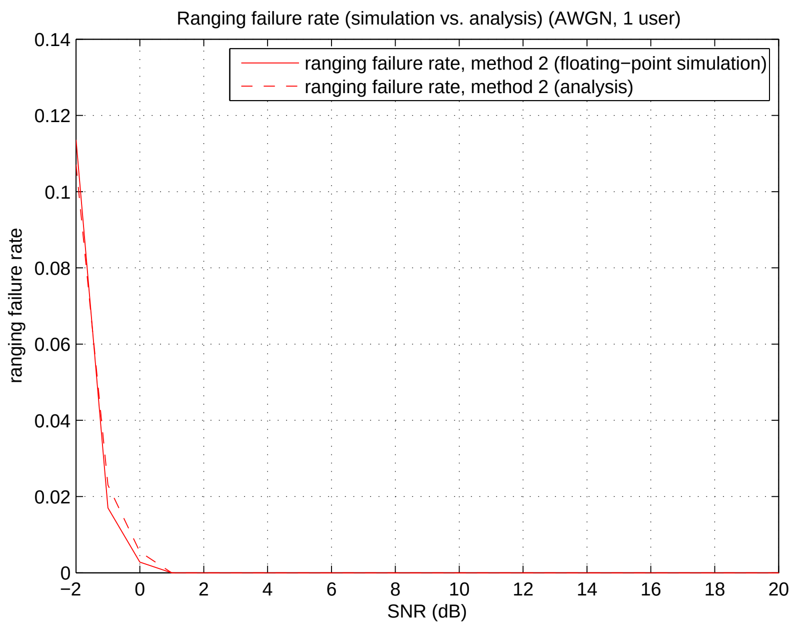


Figure 3.4: Analysis results vs. simulation results in AWGN channel.

The frequency-domain signal after FFT is

$$R(k) = g_F \cdot W_N^{kd_u} \cdot C_u(k) + W(k) \tag{3.26}$$

where $W(k)$ is the same as that in AWGN case.

The result after IFFT is

$$\begin{aligned} U(m) &= \frac{1}{\sqrt{N}} \sum_k S(k) \cdot C_i(k) \cdot W_N^{-km} \\ &= \frac{1}{\sqrt{N}} \sum_k [g_F \cdot W_N^{kd_u} \cdot C_u(k) \cdot C_i(k) \cdot W_N^{-km} + W(k) \cdot C_i(k) \cdot W_N^{-km}]. \end{aligned} \tag{3.27}$$

Similar to the AWGN case, when the multiplied ranging code matches the transmitted one, the result after IFFT should have a peak value, denoted D_p. When the multiplied code does not match the transmitted one, the value is denoted I_p. Therefore, the success rate of ranging code detection given by (3.7) where the first probability is the rate of correct detection and the second one is the product of the rate of correct rejection. In the following, we derive these two probabilities, too.

Correct Detection

When the multiplied ranging code matches the transmitted one,

$$D_p = \frac{1}{\sqrt{N}} \sum_k g_F \cdot |C_u(k)|^2 + C_u(k) \cdot W(k) \equiv R_{Dp} + j \cdot I_{Dp}. \tag{3.28}$$

Since g_F is a random variable, we calculate the PDF of $|D_p|$ by the following

$$f(|D_p| = x) = \int_0^\infty f(x|g)f(g)dg, \tag{3.29}$$

where we let $|g_F| = g$ whose PDF is as given in (3.25).

In the following we find the conditional probability $f(x|g)$ where

$$x = |\frac{1}{\sqrt{N}} \sum_k g_F \cdot |C_u(k)|^2 + C_u(k) \cdot W(k)| = \sqrt{R_{Dp}^2 + I_{Dp}^2} \tag{3.30}$$

with

$$\begin{aligned} R_{Dp} &= \frac{1}{\sqrt{N}} \{\sum_k g_{re} \cdot |C_u(k)|^2 + \sum_k C_u(k) \cdot W_r(k)\} \\ &\sim N(\frac{g_{re} \cdot 144}{\sqrt{1024}}, \frac{\sigma_r^2}{1024}) \end{aligned} \tag{3.31}$$

and

$$\begin{aligned} I_{Dp} &= \frac{1}{\sqrt{N}} \{\sum_k g_{im} \cdot |C_u(k)|^2 + \sum_k C_u(k) \cdot W_i(k)\} \\ &\sim N(\frac{g_{im} \cdot 144}{\sqrt{1024}}, \frac{\sigma_i^2}{1024}). \end{aligned} \tag{3.32}$$

Note that $g_F = g_{re} + jg_{im}$ here.

From [8], if we let $\nu \cos\theta = 4.5g_{re}$ and $\nu \sin\theta = 4.5g_{im}$, we get $\nu = 4.5g$ and $\sigma = \frac{\sigma_r}{\sqrt{1024}}$. Therefore, $x|g \sim Ricean(\nu, \sigma)$. We further assume that $K \gg 1$, so $x|g \sim N(4.5g, \sigma^2)$.

Finally, the desired probability can be calculated by

$$P(|D_p| = x > \sqrt{h_4}) = 1 - \int_0^{\sqrt{h4}} f(x)dx, \tag{3.33}$$

where $f(x)$ is as given by (3.29) and the value of σ is determined by SNR.

Correct Rejection

When the multiplied code does not match the transmitted one,

$$\begin{aligned} I_p &\equiv R_{Ip} + j \cdot I_{Ip} \\ &= \frac{1}{\sqrt{N}} \sum_k g_F \cdot C_u(k) \cdot C_i(k) + C_i(k) \cdot W(k) \\ &= \frac{1}{\sqrt{N}} \sum_k g_F \cdot C_u(k) \cdot C_i(k) + C_i(k) \cdot W_r(k) + C_i(k) \cdot W_i(k) \end{aligned} \tag{3.34}$$

where g_F is complex Gaussian with Rayleigh envelope whose PDF is given by (3.25) and $\sum_k C_u(k) \cdot C_i(k)$ is assumed to be Gaussian distributed with mean $\mu_c = -0.039$ and variance $\sigma_c^2 = 143.37$. Similarly as in the case of correct detection, we can calculate the PDF of $|I_p|$ as

$$f(|I_p| = y) = \int_0^\infty f(y|g) f(g) dg, \tag{3.35}$$

where for the conditional probability $f(y|g)$,

$$y = |\frac{1}{\sqrt{N}} \sum_k g_F \cdot C_u(k) \cdot C_i(k) + C_i(k) \cdot W(k)| = \sqrt{R_{Ip}^2 + I_{Ip}^2}, \tag{3.36}$$

with

$$\begin{aligned} R_{Ip} &= \frac{1}{\sqrt{N}} \{\sum_k g_{re} \cdot C_u(k) \cdot C_i(k) + \sum_k C_i(k) \cdot W_r(k)\} \\ &\sim N(\frac{g_{re} \cdot \mu_c}{\sqrt{1024}}, \frac{\sigma_c^2 + \sigma_r^2}{1024}) \\ &\sim N(-0.0012 g_{re}, \frac{143.37 + \sigma_r^2}{1024}) \\ &\sim N(0, \frac{143.37 + \sigma_r^2}{1024}) \end{aligned} \tag{3.37}$$

and

$$\begin{aligned} I_{Ip} &= \frac{1}{\sqrt{N}}\{\sum_k g_{im}\cdot C_u(k)\cdot C_i(k)+\sum_k C_i(k)\cdot W_i(k)\} \\ &\sim N(\frac{g_{im}\cdot \mu_c}{\sqrt{1024}}, \frac{\sigma_c^2+\sigma_i^2}{1024}) \\ &\sim N(-0.0012 g_{im}, \frac{143.37+\sigma_i^2}{1024}) \\ &\sim N(0, \frac{143.37+\sigma_i^2}{1024}). \end{aligned} \tag{3.38}$$

Again, $g_F = g_{re} + j g_{im}$ here.

So $y|g$ is Rayleigh distributed with $\sigma^2 = \frac{143.37+\sigma_r^2}{1024}$. The PDF is given by

$$f(y|g) = \frac{y}{\sigma^2}\exp\{-\frac{y^2}{2\sigma^2}\}. \tag{3.39}$$

Finally, the desired probability can be calculated by

$$P(y = |I_p| < \sqrt{h_4}) = \int_0^{\sqrt{h_4}}\int_0^{\infty} f(y|g)f(g)dgdy. \tag{3.40}$$

Numerical Results

Figure 3.5 shows some numerical results of the above analysis compared with the floating-point simulation results. The analysis results here is less accurate than the AWGN case since we employ some approximation in the derivation.

3.3 Floating-Point Simulation

In this section, we describe our simulation environment first. Then we discuss the penalty of missed detection and false alarm in ranging signal detection. At the end, we show some simulation results and the effect of carrier frequency offset (CFO).

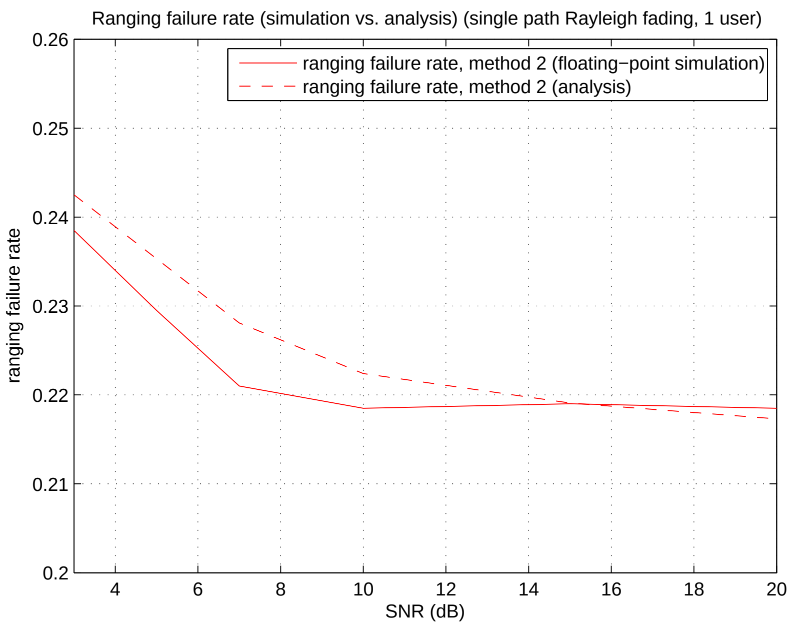


Figure 3.5: Analysis results vs. simulation results in single path Rayleigh fading channel.

3.3.1 System Parameters

The important system parameters used in our study are listed in Table 3.1.

We let parameter $S = 5$, $N = 6$ and $M = 16$ in our simulation. That is, there are a total of sixteen possible periodic ranging codes, which are from the twelfth code to the twenty-seventh code generated by the PRBS generator. We simulate up to the situation of three simultaneous ranging users. The codes used by these three users are the twelfth, the fifteenth and the eighteenth code, respectively. We let UL_Permbase=0. The timing offsets of the three users are 10, 15 and 7 samples, respectively.

3.3.2 Channel Environments

The simulation is done with AWGN and SUI-3 channels with mobile speed being 60 km/hr. More detailed channel environments are described in the following, which are taken from

Table 3.1: System Parameters Used in Our Study

Parameter	Value
System Channel Bandwidth (MHz)	10
Sampling Frequency (MHz)	11.2
FFT Size	1024
Subcarrier Spacing (kHz)	10.94
Useful Symbol Time (μsec)	91.4
Guard Time (μsec)	11.4
OFDMA Symbol Time (μsec)	102.9
G (Ratio of CP Time to "Useful" Time)	1/8
n (Sampling Factor)	28/25

[10] and [11].

Typical models of the wireless communication channel include additive noise and multipath fading. For channel simulation, noise and multipath fading are described as random processes, so they can be algorithmically generated as well as mathematically analyzed.

Gaussian Noise

The simplest kind of channel is AWGN channel, where the received signal is only subject to added noise. A major source of this noise is the thermal noise in the amplifiers which may be modeled as Gaussian with zero mean and constant variance. In computer simulations, random number generators may be used to generate Gaussian noise of given power to obtain a particular SNR.

Slow Fading Channel

In slow fading, multipath propagation may exist, but the channel coefficients do not change significantly over a relatively long transmission period. The channel impulse response over a short time period can be modeled as

$$h(\tau) = \sum_{i=0}^{N-1} \alpha_i e^{j\theta_i} \delta(\tau - \tau_i) \tag{3.41}$$

where N is the number of multipaths, α_i and τ_i are respectively the amplitude and the delay of the ith multipath, and θ_i represents the phase shift associated with path i. These parameters are time-invariant in a short enough time period.

Fast Fading Channel

With sufficiently fast motion of either the transmitter or the receiver, the coefficient of each propagation path becomes time varying. The equivalent baseband channel impulse response can then be better modeled as

$$h(\tau, t) = \sum_{i=0}^{N-1} \alpha_i(t) e^{j\theta_i(t)} \delta(\tau - \tau_i). \tag{3.42}$$

Note that α_i and θ_i are now functions of time. But τ_i is still time-invariant, because the path delays usually change at a much slower rate than the path coefficients. The channel coefficients are often modeled as complex independent stochastic processes. If there is no LOS path between the transmitter and the receiver, then each path may be made of the superposition of many reflected paths, yielding a Rayleigh fading characteristic. A commonly used method to simulate Rayleigh fading is Jakes' fading model, which is a deterministic method for simulating time-correlated Rayleigh fading waveforms. An improvement to Jakes' model is proposed in [12].

Power-Delay Profile Model

For simplicity in analysis and simulation, the delay τ_i in the above two models can be discretized to have a certain easily manageable granularity. This results in a tapped-delay-line model for the channel impulse response, where the spacing between any two taps is

Table 3.2: SUI-3 Channel Model

	Relative delay (μs and sample number)			Average power		
Tap	(μs)	(4×oversampling)	(normal)	(dB)	(normal scale)	(normalized)
1	0	0	0	0	1	0.7061
2	0.4	17	4	-5	0.3162	0.2233
3	0.9	40	10	-10	0.1	0.0706

an integer multiple of the chosen granularity. For convenience, one may excise the initial delay and make $\tau_0 = 0$. Often, it is convenient to normalize the path powers relative to the strongest path. And, often, the first path has the highest average power.

The channel model used here is a modification of the Stanford University Interim (SUI) channel models proposed in [13]. A set of 6 typical channels was selected for the three most common terrain categories that are typical of the continental United States. The scenario of SUI channels are:

- Cell size: 7 km.
- Base station antenna height: 30 m.
- Receiver antenna height: 6 m.
- Base station antenna beamwidth: 120°.
- Receiver antenna beamwidth: omnidirectional (360°) and 30°.
- Vertical polarization only.

We list the SUI-3 channel model in Tables 3.2, which is the one used in our simulation.

3.3.3 Missed Detection and False Alarm Penalty of Ranging Signal Detection

Missed detection means that a ranging code has been transmitted by a certain MS but the BS does not detect it. In this case, the BS will send a RNG-RSP with continue status but without corrections [1], [2]. Then the MS will need to retransmit a ranging signal. It will do some random backoff and adjust the power level of ranging signal up to $P_{Tx\_IR\_MAX}$. The MS will send ranging signal at next ranging opportunity, that is, at the next UL-subframe which is 5 ms later according to our simulation parameters.

The false alarm means that the BS detects a ranging code that has not been transmitted by any MS. In this case, the BS will redundantly estimate wrong transmission parameters (time, power and possibly frequency offset). Then the BS will broadcast a RNG-RSP with corrections, but this message is redundant and will not be used by any MS.

We compare the effect of missed detection and false alarm. The effect of missed detection is retransmission of ranging signal. The main penalties are retransmission efforts for the MS and more latency to complete the ranging process because next ranging opportunity will be 5 ms later. On the other hand, the effect of false alarm is needless parameter estimation and RNG-RSP transmission. The main penalties are redundant efforts for the BS and the wasted downlink bandwidth for RNG-RSP transmission.

The rates of missed detection and false alarm are affected by the choice of the detection threshold (h_4). Since it is difficult to decide which one costs more, we only set the cost of them to be a certain ratio. We calculate the weighted costs and use them to help us determining the detection threshold. In the following, we provide some examples of weighted costs in the one ranging user case under single-path Rayleigh fading channel. Note that the weighted cost is obtained by $missed\ detection\ rate \times r_{md} + false\ alarm\ rate \times r_{fa}$, where we set the ratio of the cost of missed detection to that of false alarm as $r_{md} : r_{fa}$. Figures 3.6 to 3.10

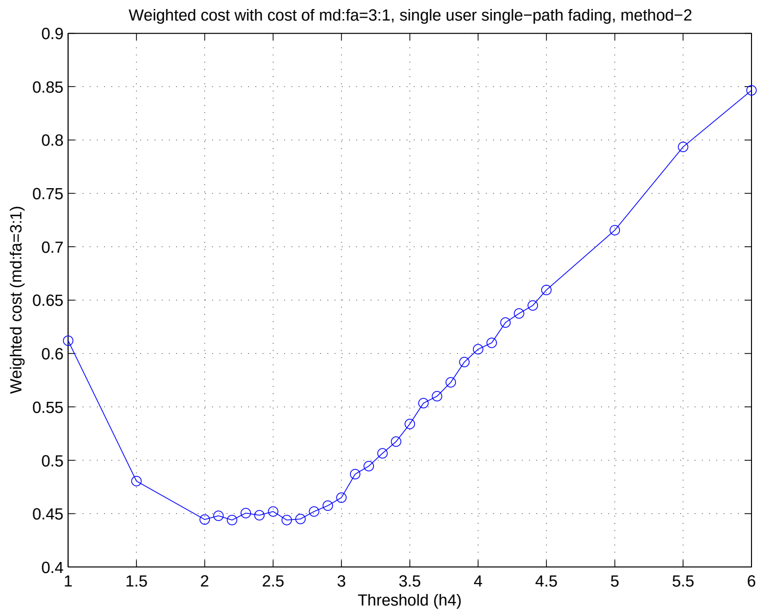


Figure 3.6: Weighted cost for $r_{md} : r_{fa} = 3 : 1$.

shows the results for $r_{md} : r_{fa}$ being 3:1, 2:1, 1:1, 1:2 and 1:3, respectively. We can choose the best threshold as the one that leads to a minimum weighted cost. Note that the weighted costs do not have large sensitivity to h_4.

3.3.4 Simulation Results

Some floating-point simulation results are shown in this section. Note that the signal-to-noise ratio (SNR) used in our simulation means the ratio of the variance of ranging signal samples to that of the noise samples. The detection thresholds h_4, h_1, h_2 and H we have chosen are 4.3, 3, 1.55, 12. The value of h_4 is determined by the weighted-cost method mentioned in previous subsection. The determination of h_1, h_2 and H is also based on the similar idea but is more complex. Thus, we decide them by many times of simulations. In addition, the number of simulation runs are from 2000 to 5000, depending on the simulated case. For simulations with the SUI-3 channel, 2000 runs are enough to observe more than

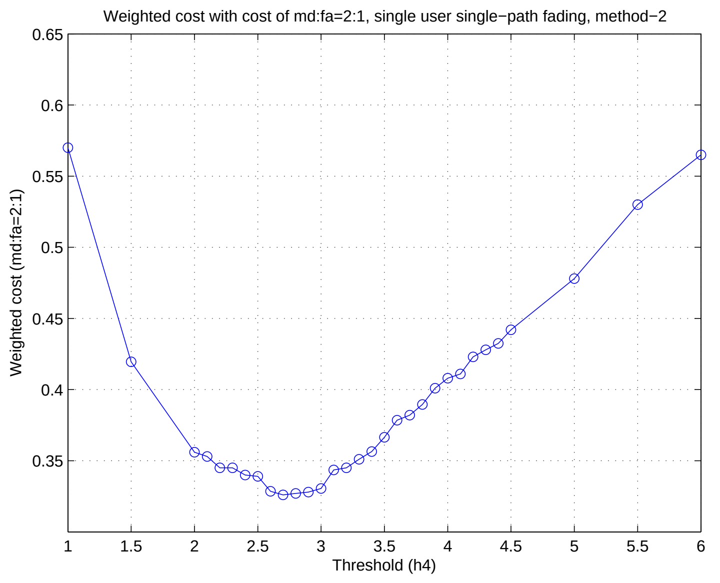


Figure 3.7: Weighted cost for $r_{md} : r_{fa} = 2 : 1$.

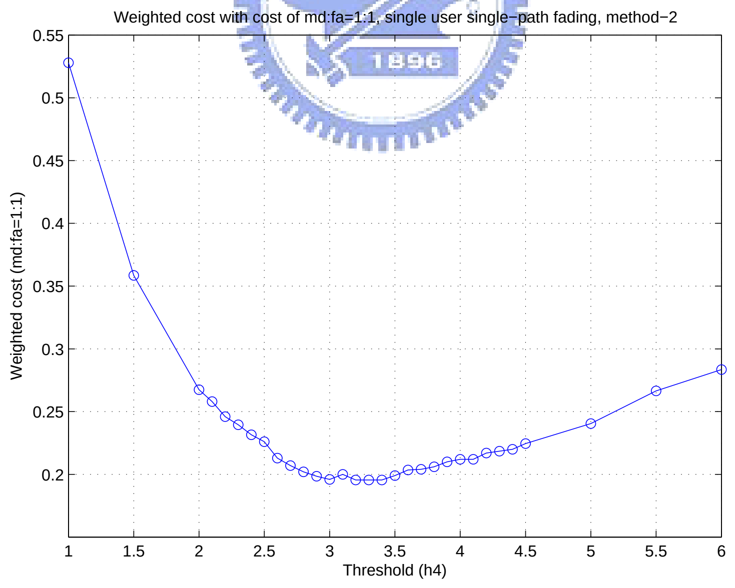


Figure 3.8: Weighted cost for $r_{md} : r_{fa} = 1 : 1$.

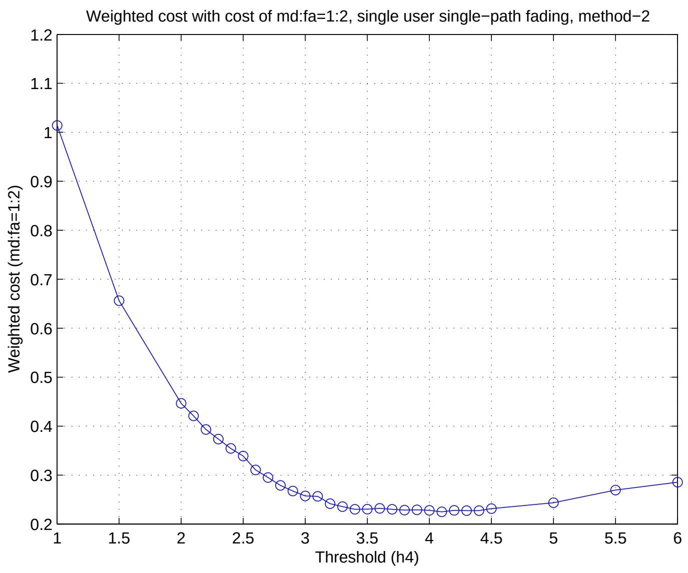


Figure 3.9: Weighted cost for $r_{md} : r_{fa} = 1 : 2$.

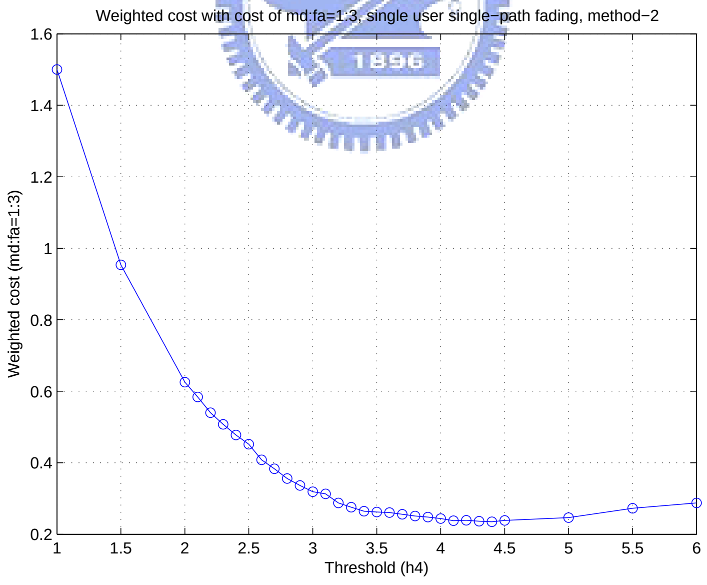


Figure 3.10: Weighted cost for $r_{md} : r_{fa} = 1 : 3$.

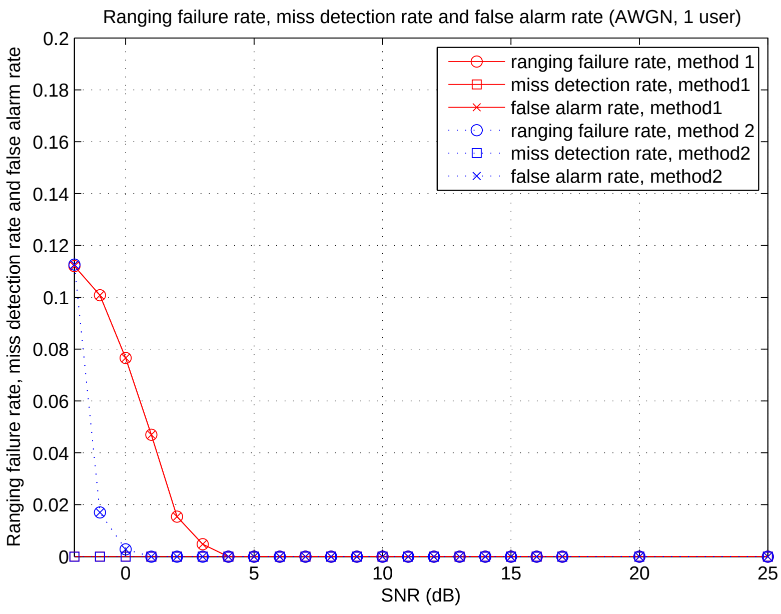


Figure 3.11: Ranging failure rate, miss detection rate and false alarm rate in AWGN channel with one ranging user.

100 detection failures. But for simulations with the AWGN channel, we need up to 5000 runs.

Figures 3.11, 3.12 and 3.13 show the total failure rate, missed detection rate and false alarm rate of ranging code detection with detection method 1 and method 2 in AWGN channel. The total failure rate here means that either missed detection or false alarm occurs. We can see that method 1 obviously has better performance than method 2, especially at low SNR. This is reasonable since method 1 uses the detection metric that is somewhat similar to signal-to-interference ratio (SIR). Another observation is that the main ranging failure comes from false alarm. This is also intuitively reasonable since the peak norm value when the code matches is large in AWGN so we can always detect the desired ranging user. But, noise will cause some false alarm at low SNR.

Figure 3.14 shows the average successful detection rate of each user, which is always very

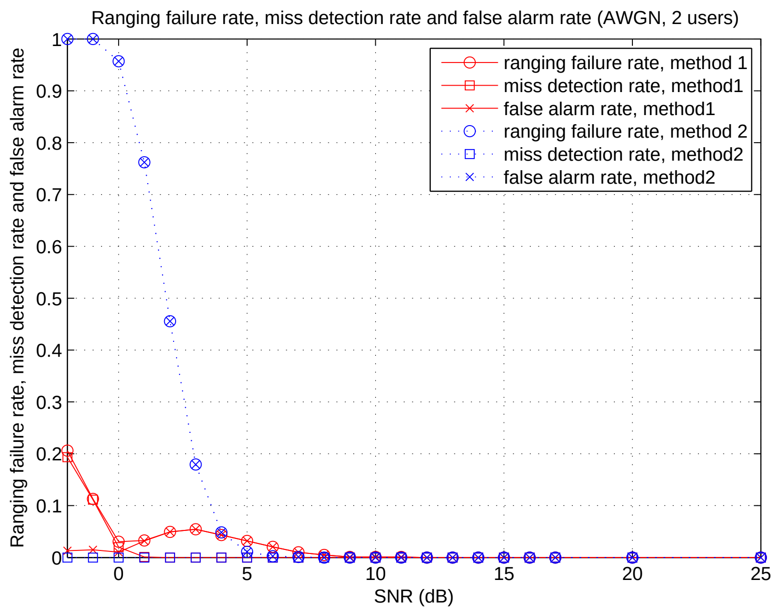


Figure 3.12: Ranging failure rate, miss detection rate and false alarm rate in AWGN channel with two ranging users.

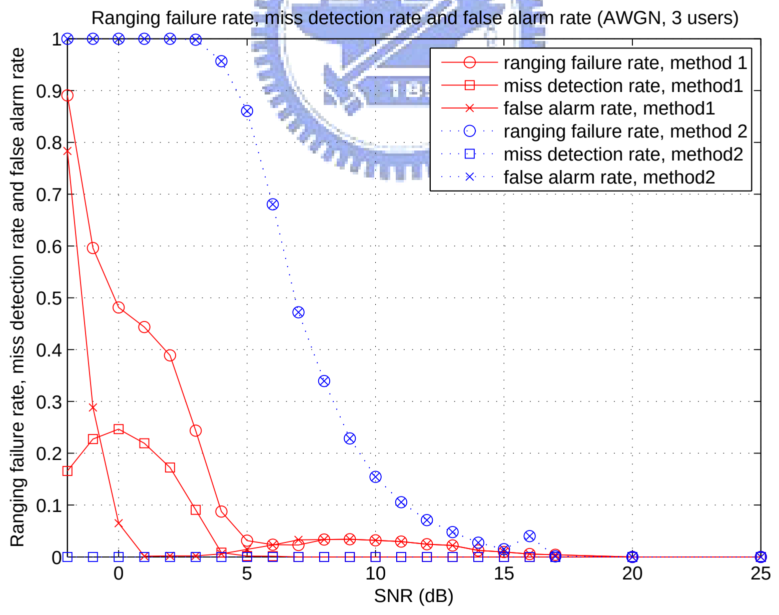


Figure 3.13: Ranging failure rate, miss detection rate and false alarm rate in AWGN channel with three ranging users.

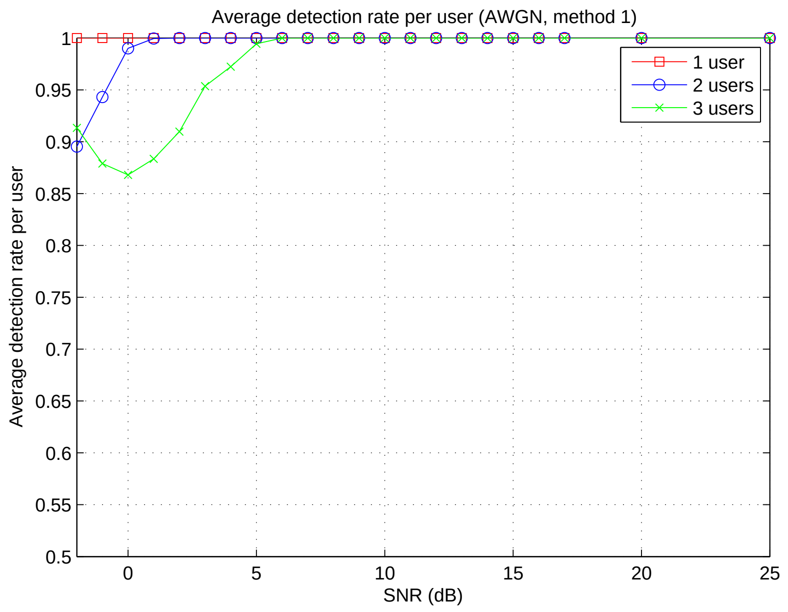


Figure 3.14: Average success rate of each user with method 1 in AWGN channel.

close to 1 here. Figure 3.15 depicts the average root mean square error (RMSE) of timing offset estimation. We see that the estimation is perfect as long as SNR is larger than 5 dB.

The total failure rate, missed detection rate and false alarm rate of ranging code detection with detection method 1 and method 2 in SUI-3 channel are depicted in Figures 3.16, 3.17 and 3.18. As in the AWGN case, method 1 has better performance than method 2. We observe that in a multipath channel like SUI-3, the failure rate is mainly determined by the number of users and less influenced by SNR. As long as the SNR is high enough, there exists a performance floor. That is, the performance does not improve any more when SNR gets larger.

Figure 3.19 shows the average successful detection rate of each user. Note that the product of each user's failure detection rate is the missed detection rate since the transmission of each user is independent. Figure 3.20 depicts the average RMSE of timing offset estimation. The RMSE ranges roughly from 1.9 to 3 samples over the range of SNR considered.

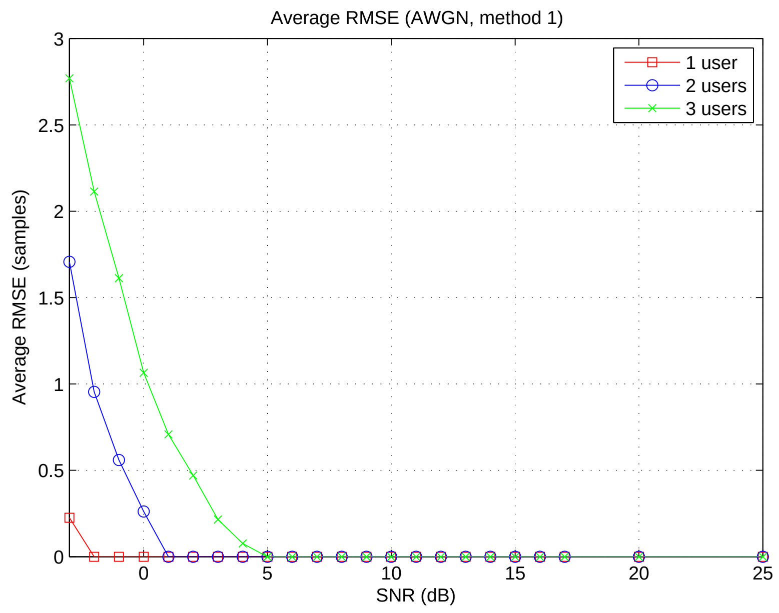


Figure 3.15: RMSE of timing offset estimation in AWGN channel.

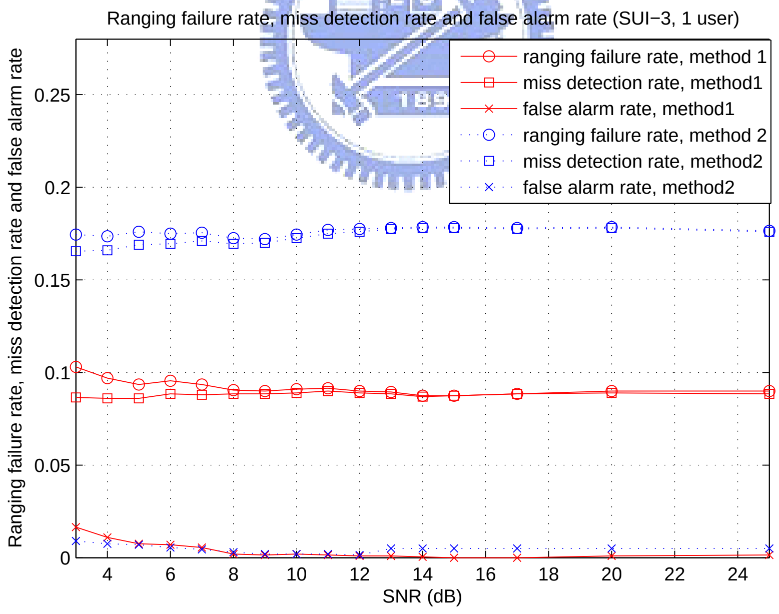


Figure 3.16: Ranging failure rate, miss detection rate and false alarm rate in SUI-3 channel with one ranging user.

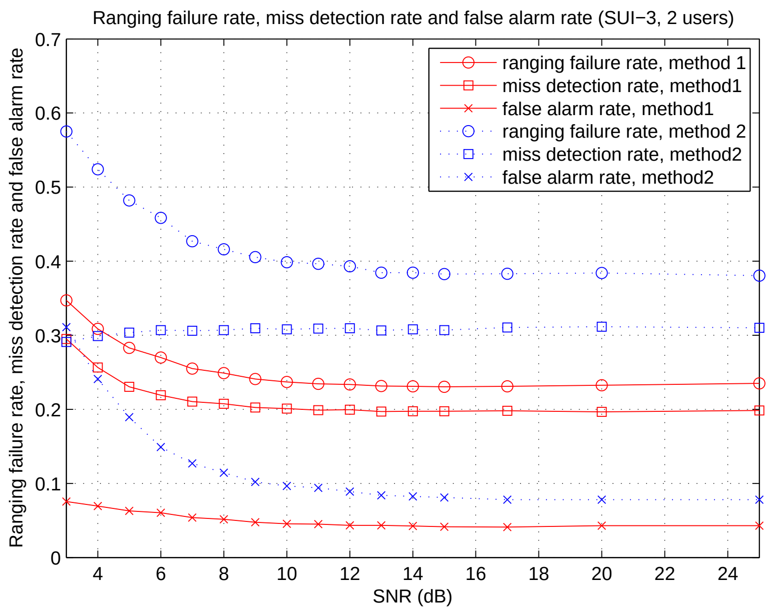


Figure 3.17: Ranging failure rate, miss detection rate and false alarm rate in SUI-3 channel with two ranging users.

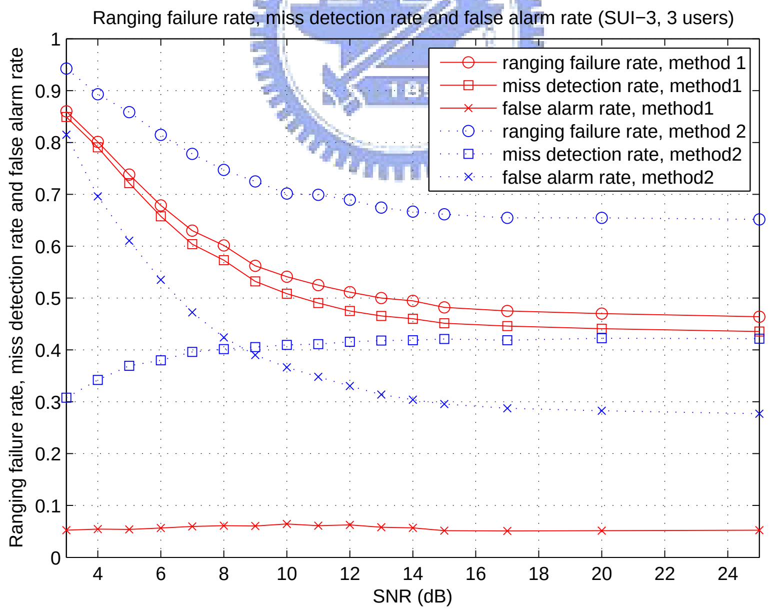


Figure 3.18: Ranging failure rate, miss detection rate and false alarm rate in SUI-3 channel with three ranging users.

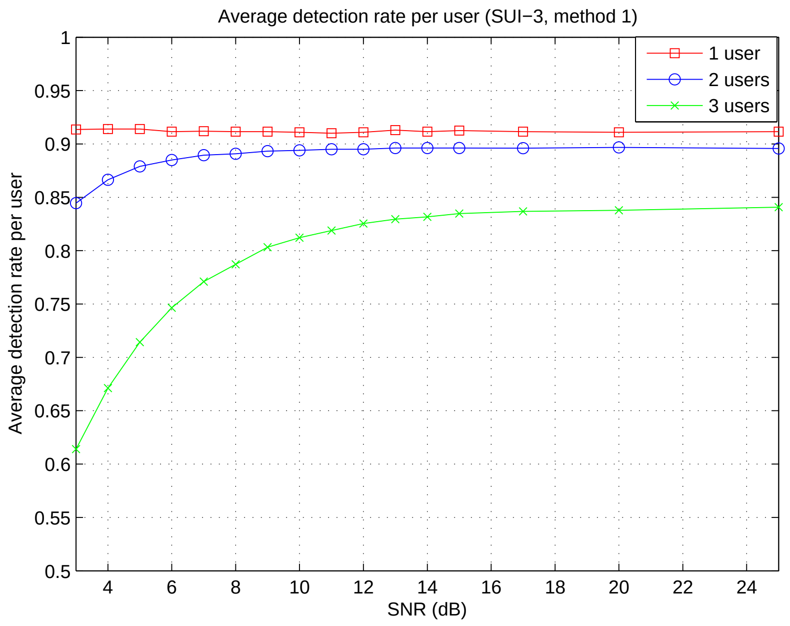


Figure 3.19: Average success rate of each user with method 1 in SUI-3 channel.

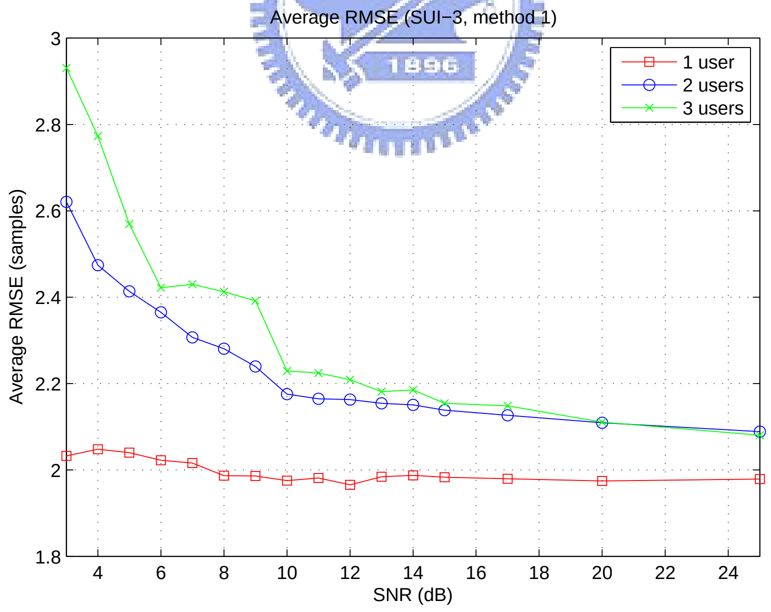


Figure 3.20: RMSE of timing offset estimation in SUI-3 channel.

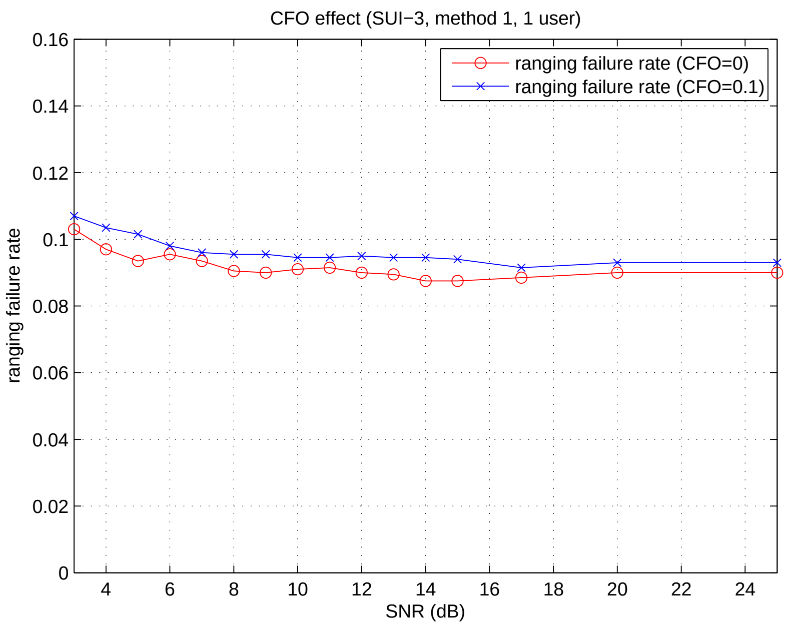


Figure 3.21: Comparison of the failure detection rates under no CFO and CFO=0.1 in SUI-3 channel with one ranging user.

3.3.5 Effect of CFO

We do not perform CFO estimation in periodic ranging process for two reasons. First, the carrier frequency should be accurate enough when doing periodic ranging. Second, the following simulation shows that CFO has little effect on the performance of ranging.

In our system, the frequency offset value is within the range of $[-0.1, 0.1]$ of subcarrier spacing, where the speed is 120 km/hr and the central frequency is 3.5 GHz. Figure 3.21 compares the failure detection rates under no CFO and CFO=0.1 in the case of one ranging user. Figure 3.22 compares the RMSE under the same conditions. Further, Figure 3.23 compares the average detection rate in the case of three ranging users. Figure 3.24 compares the RMSE. All simulations are done with the SUI-3 channel. Therefore, we ignore CFO estimation since it improves the performance of ranging process little.

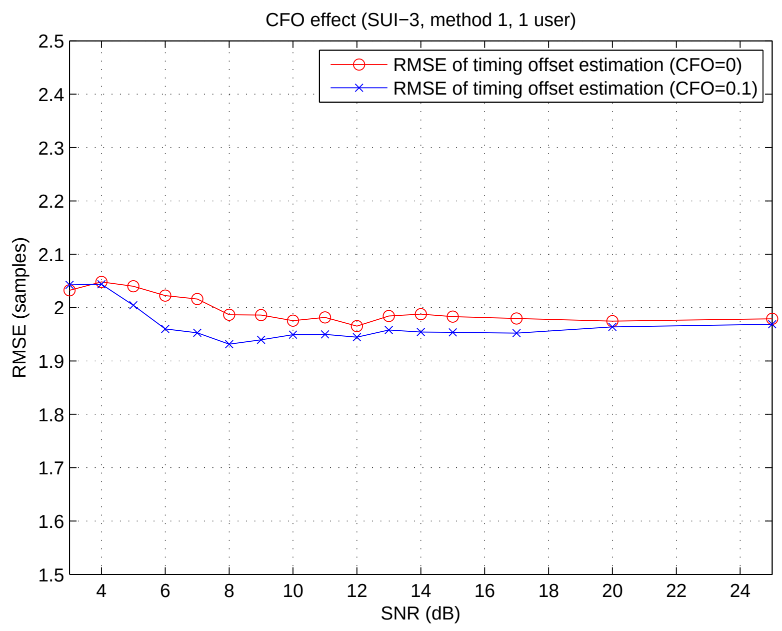


Figure 3.22: Comparison of RMSE of timing offset estimation under no CFO and CFO=0.1 in SUI-3 channel with one ranging user.

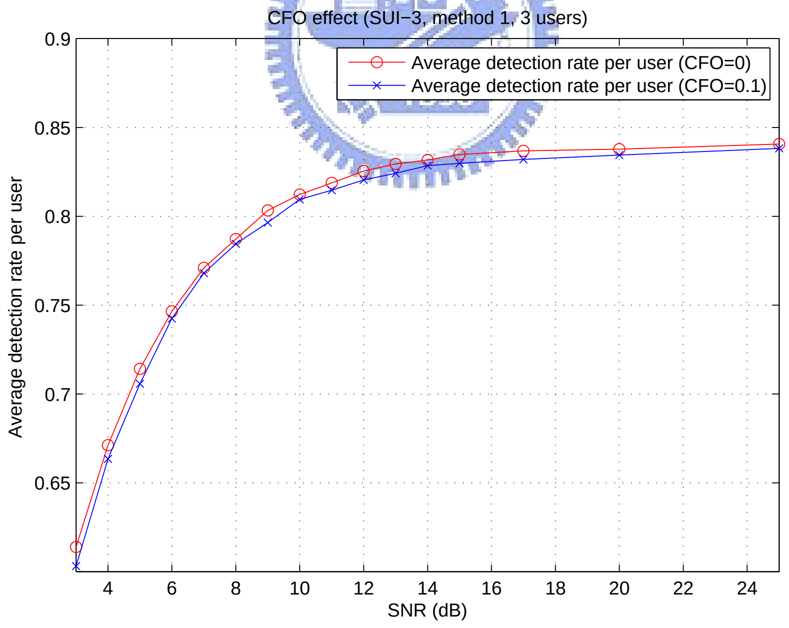


Figure 3.23: Comparison of average detection rates under no CFO and CFO=0.1 in SUI-3 channel with three ranging users.

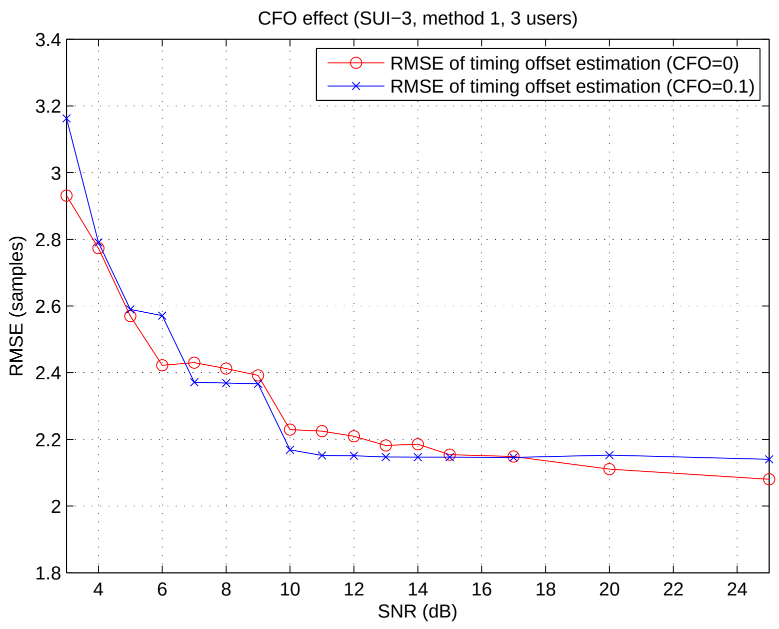


Figure 3.24: Comparison of RMSE of timing offset estimation under no CFO and CFO=0.1 in SUI-3 channel with three ranging users.

3.3.6 Missed Detection and False Alarm Penalty under Our Algorithm

In the end of this section, we discuss the missed detection and false Alarm penalty under our algorithm. This provides another view of the performance. We consider two cases for example.

For AWGN channel with 3 ranging users and SNR=3 dB, the average detection success rate is 0.9535. We can calculate average required number of ranging transmissions:

$$1 \times 0.9535 + 2 \times (1-0.9535) \times 0.9535 + 3 \times (1-0.9535)^2 \times 0.9535 + ... = 1.0488. \quad (3.43)$$

Therefore, average required number of retransmissions is 0.0488, where each retransmission causes increased latency which is 5 ms in our system. On the other hand, the false alarm rate is 0.002 here. The main false alarm penalty is wasted DL bandwidth but it is hard to quantify it.

Consider another case with worse channel condition. For SUI-3 channel with 3 ranging users and SNR=5 dB, the average detection success rate is 0.7142. Thus the average required number of ranging transmissions is

$$1 \times 0.7142 + 2 \times (1 - 0.7142) \times 0.7142 + 3 \times (1 - 0.7142)^2 \times 0.7142 ... = 1.4002. \quad (3.44)$$

Average required number of retransmissions is 0.4002. The false alarm rate is 0.054 now. The performance is acceptable even under bad channel conditions.

Chapter 4

DSP Implementation Environment

In this chapter, we introduce the DSP platform used in our implementation, which is a Sundance's product. In addition to hardware introduction, the software development tools and code development techniques are also introduced.

4.1 DSP Implementation Platform

A typical DSP application using Sundance equipment is made up of a host personal computer (PC) holding one or more "carrier boards," each supporting one or more processing elements (DSPs, I/O devices, or FPGAs) known as the Texas Instruments Module (TIM) [14].

In our DSP application, the carrier board (SMT310Q) plugs into a host PC using a PCI slot and does not run any program; it simply holds the TIMs (SMT395), providing them with power and a means of communicating with the rest of the world.

4.1.1 The Carrier Board (SMT310Q) [15]

The SMT310Q is a quad site module carrier developed to provide access to TIM modules over the PCI bus running at 33 MHz with a 32-bit data bus. As shown in Figure 4.1, the main connection to the half-length PCI bus is via the module's global bus. A single ComPort

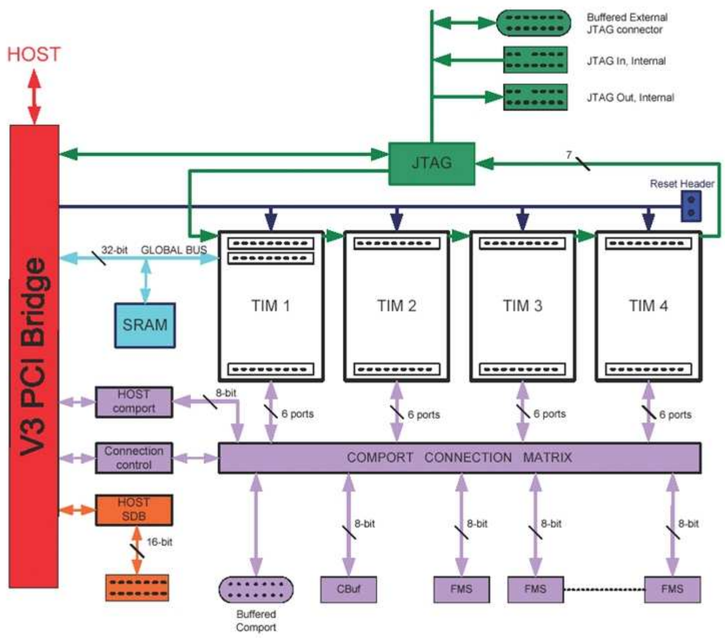


Figure 4.1: The SMT310Q carrier board (from [15]).

is also mapped to the PCI bus providing support for application boot and data transfer. 1 Mbyte of SRAM is mapped on to the global bus and can be accessed by the module as a global resource or by the PCI bridge. A 3.3 volt supply is available at the fixing pillars for the module. This supply is taken from the PCI edge connector.

4.1.2 The Texas Instruments Module (SMT395)

The TIMs used in our implementation are Sundance's SMT395, as shown in Figure 4.2. It is based on the 1 GHz 64-bit TMS320C6416T fixed-point DSP, manufactured on 90 nm wafer technology. The SMT395 is supported by the TI Code Composer Studio and 3L Diamond RTOS (real-time operating system) to enable full multi-DSP systems with minimum efforts by the programmers [15].

Some important features of the SMT395 module are as follows.

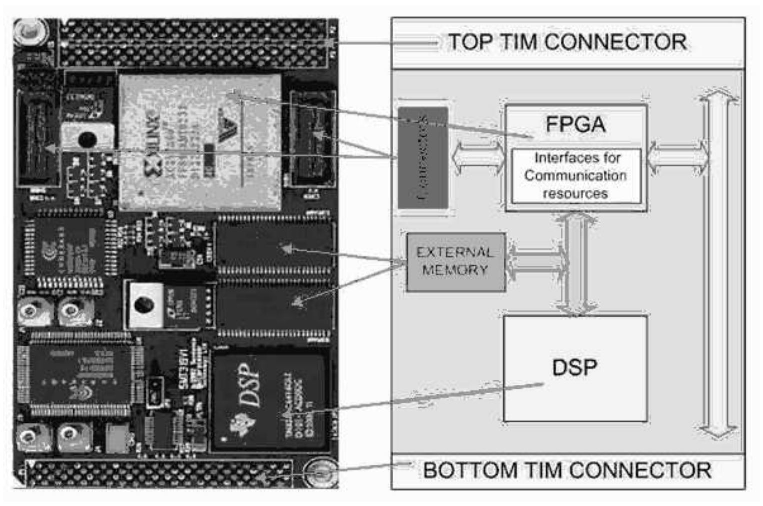


Figure 4.2: The SMT395 module (from [14]).

- 1 GHz TMS320C6416T fixed point DSP with 8000 MIPS peak DSP performance.
- Xilinx Virtex II Pro FPGA XC2VP30-6 in FF896 package.
- 256 Mbytes of SDRAM at 133 MHz.
- Two Sundance High-speed Bus (50 MHz, 100 MHz or 200 MHz) ports at 32 bits width.
- Eight 2 Gbit/sec Rocket Serial Links (RSL) for inter-Module communication.
- Six common ports up to 20 MB per second for inter-DSP communication.
- 8 Mbytes flash ROM for configuration and booting.
- JTAG diagnostics port.

4.1.3 The DSP Chip [16]

The TMS320C6416T DSP is a fixed-point DSP in the TMS320C64x series of the TMS320C6000 DSP platform family. It is based on the VelociTI very-long-instruction-word (VLIW) archi-

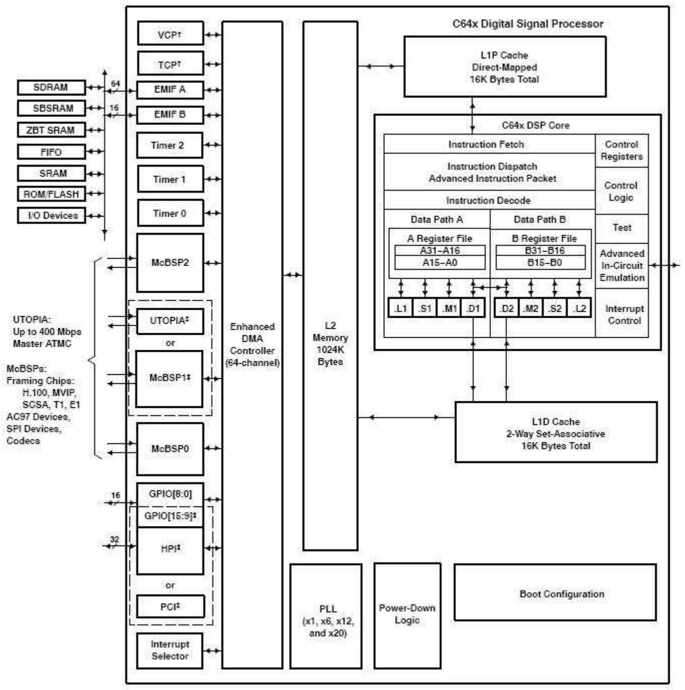


Figure 4.3: Functional block and CPU diagram of the DSP [17].

tecture developed by TI. The functional block and DSP core diagram of TMS320C64x series is shown in Fig. 4.3.

The C6000 core CPU consists of 64 general-purpose 32-bits registers and eight function units. Features of C6000 devices include the following:

- VLIW CPU with eight functional units, including two multipliers and six arithmetic logic units:
  - Executes up to eight instructions per cycle.
  - Allows designers to develop highly effective RISC-like code for fast development time.
- Efficient code execution on independent functional units:

  - Efficient C complier on DSP benchmark suite.
  - Assembly optimizer for fast development and improved parallelization.

- Instruction packing:
  - Gives code size equivalence for eight instructions executed serially or in parallel.
  - Reduces code size, program fetches, and power consumption.

- Conditional execution of all instructions:
  - Reduces costly branching.
  - Increases parallelism for higher sustained performance.

- 8/16/32-bit data support, providing efficient memory support for a variety of applications.

- 40-bit arithmetic options add extra precision for vocoders.

- 32×32-bit integer multiply with 32- or 64-bit result.

Central Processing Unit [17]

The C64x DSP core contains 64 32-bit general purpose registers, program fetch unit, instruction decode unit, two data paths each with four function units, control register, control logic, advanced instruction packing, test unit, emulation logic and interrupt logic. The program fetch, instruction fetch, and instruction decode units can arrange eight 32-bit instructions to the eight function units every CPU clock cycle.

The processing of instructions occurs in each of the two data paths (A and B) shown in Figure 4.3, each of which containing four functional units and one register file. The four functional units are: one unit for multiplier operations (.M), one for arithmetic and logic operations (.L), one for branch, byte shifts, and arithmetic operations (.S), and one for linear and circular address calculation to load and store with external memory operations (.D). The details of the functional units are described in Table 4.1.

Each register file consists of 32 32-bit registers. Each of the four functional units reads and writes directly within its own data path. That is, the functional units .L1, .S1, .M1, .D1 can only write to register file A. The same condition occurs in register file B. However, two cross-paths (1X and 2X) allow functional units from one data path to access a 32-bit operand from the opposite-side register file. The cross path 1X allows data path A to read their source from register file B. The cross path 2X allows data path B to read their source from register file A. In the C64x, CPU pipelines data-cross-path accesses over multiple clock cycles. This allows the same register to be used as a data-cross-path operand by multiple functional units in the same execute packet.

Memory Architecture and Peripherals

The C64x is a two-level cache-based architecture. The level 1 cache is separated into program and data spaces. The level 1 program cache (L1P) is a 128 Kbit direct mapped cache and the level 1 data cache (L1D) is a 128 Kbit 2-way set-associative mapped cache. The level 2 (L2) memory consists of 8 Mbytes memory space for cache (up to 256 Kbytes) and unified mapped memory.

The external memory interface (EMIF) provides interfaces for the DSP core and external memory, such as synchronous-burst SRAM (SBSRAM), synchronous DRAM (SRAM), SDRAM, FIFO and asynchronous memories (SRAM and EPROM). The EMIF also provides 64-bit-wide (EMIFA) and 16-bit-wide (EMIFB) memory read capability.

The C64x contains some peripherals such as enhanced direct-memory-access (EDMA), host-port interface (HPI), PCI, three multichannel buffered serial ports (McBSPs), three

Table 4.1: Functional Units and Operations Performed [18]

Parameter	Value
.L unit(.L1, .L2)	32/40-bit arithmetic and compare operations 32-bit logical operations Leftmost 1 or 0 counting for 32 bits Normalization count for 32 and 40 bits Byte shifts Data packing/unpacking 5-bit constant generation Dual 16-bit and Quad 8-bit arithmetic operations Dual 16-bit and Quad 8-bit min/max operations
.S unit (.S1, .S2)	32-bit arithmetic operations 32/40-bit shifts and 32-bit bit-field operations 32-bit logical operations Branches Constant generation Register transfers to/from control register file (.S2 only) Byte shifts Data packing/unpacking Dual 16-bit and Quad 8-bit compare operations Dual 16-bit and Quad 8-bit saturated arithmetic operations
.M unit (.M1, .M2)	16 x 16 multiply operations 16 x 32 multiply operations Dual 16 x 16 and Quad 8 x 8 multiply operations Dual 16 x 16 multiply with add/subtract operations Quad 8 x 8 multiply with add operations Bit expansion Bit interleaving/de-interleaving Variable shift operations Rotation Galois Field Multiply
.D unit (.D1, .D2)	32-bit add, subtract, linear and circular address calculation Loads and stores with 5-bit constant offset Loads and stores with 15-bit constant offset(.D2 only) Loads and stores doubles words with 5-bit constant Loads and store non-aligned words and double words 5-bit constant generation 32-bit logical operations

32-bit general-purpose timers and sixteen general-purpose I/O pins. The EDMA controller handles all data transfers between the level-two (L2) cache/memory and the peripheral device. The C64x has 64 independent channels. The HPI is a 32-/16-bit wide parallel port through which a host processor can directly access the CPUs memory space. The PCI port supports connection of the DSP to a PCI host via the integrated PCI master/slave bus interface.

4.2 Code Development Environment

In our study, we use the software development tool Code Composer Studio (CCS). The CCS is a key element of the DSP software and development tools from Texas Instruments. The tutorial [19] introduces the key features of CCS and the programmer's guide [20] gives a reference for programming TMS320C6000 DSP devices. A programmer needs to be familiar with coding development flow and CCS for building a new project on the DSP platform efficiently.

4.2.1 Code Composer Studio [19]

Code Composer Studio is a Texas Instruments product. It provides the compiler, assembler, and linker that we need to be able to generate programs that will execute on DSP TIMs. It also provides software for debugging programs by watching and interacting with their execution.

Some main features of the CCS are:

- Real-time analysis.
- Source code debugger common interface for both simulator and emulator targets:
  - C/C++/assembly language support.

  - Simple breakpoints.
  - Advanced watch window.
  - Symbol browser.

- DSP/BIOS host tooling support (configure, real-time analysis and debug).

- Data transfer for real time data exchange between host and target.

- Profiler to understand code performance.

- DSP/BIOS support:
  - Pre-emptive multi-threading.
  - Interthread communication.
  - Interupt handling.

- Chip Support Libraries (CSL) to simplify device configuration. CSL provides C-program functions to configure and control on-chip peripherals.

- DSP libraries for optimum DSP functionality. The libraries include many C-callable, assembly-optimized, general-purpose signal-processing and image/video processing routines. These routines are typically used in computationally intensive real-time applications where optimal execution speed is critical.

4.2.2 Code Development Flow [20]

The recommended code development flow involves utilizing the C6000 code generation tools to aid in optimization rather than forcing the programmer to code by hand in assembly. Hence the programmer may let the compiler do all the laborious work of instruction selection, parallelizing, pipelining, and register allocation. This simplifies the maintenance of the code, as everything resides in a C framework that is simple to maintain, support, and upgrade. Figure 4.4 illustrates the three phases in the code development flow. Because phase 3 is usually too detailed and time consuming, most of the time we should not need to go into phase 3 to write linear assembly code unless the software pipelining efficiency is too bad or the resource allocation is too unbalanced.

4.3 Code Optimization

In this section, we describe several methods that can accelerate our code and reduce the execution time on the C64x DSP. First, we introduce two techniques that can be used to analyze the performance of specific code regions:

- Use the clock( ) and printf( ) functions in C/C++ to time and display the performance of specific code regions. Use the stand-alone simulator (load6x) to run the code for this purpose.
- Use the profile mode of the stand-alone simulator. This can be done by compiling the code with the -mg option and executing load6x with the -g option. Then enable the clock and use profile points and the RUN command in the CCS debugger to track the number of CPU clock cycles consumed by a particular section of code. Use View Statistics to view the number of cycles consumed.

Usually, we use the second technique above to analyze the C code performance. The feedback of the optimization result can be obtained with the -mw option. It shows some important results of the assembly optimizer for each code section. We take these results into consideration in improving the computational speed of certain loops in our program.

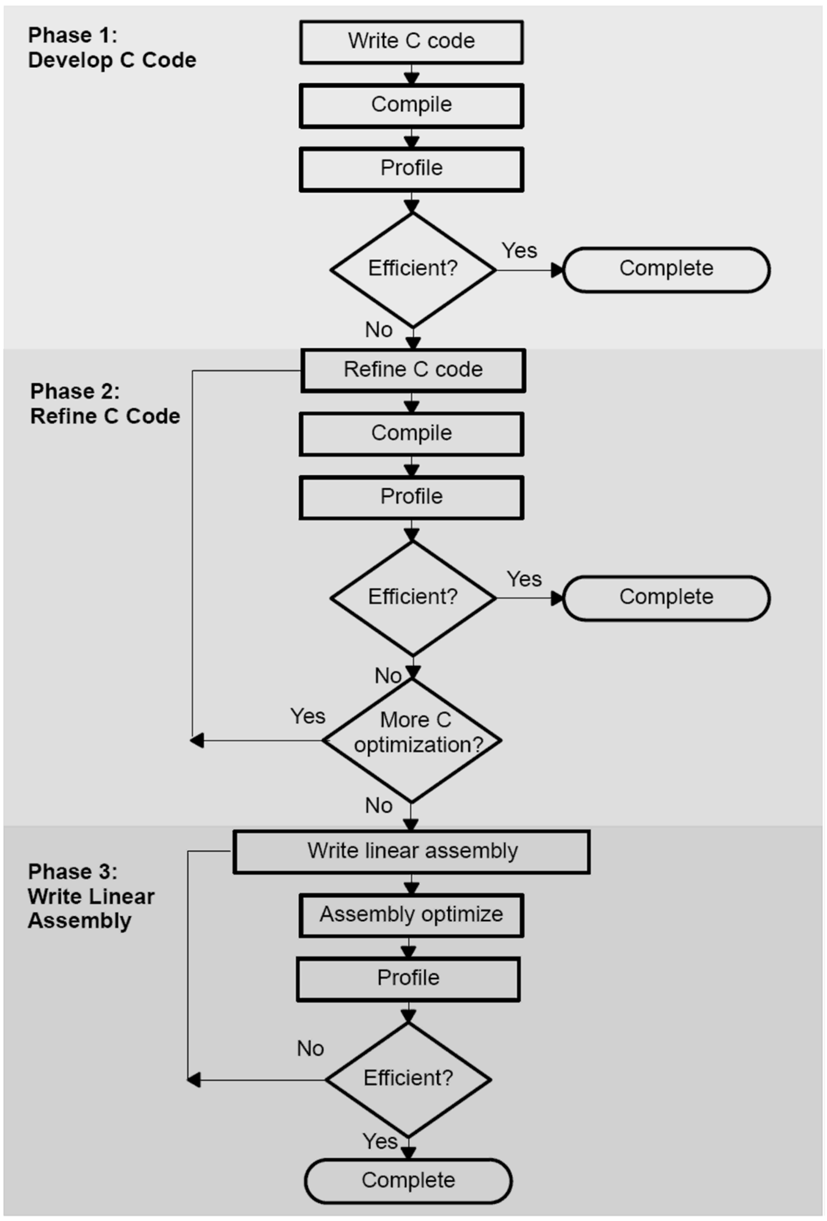


Figure 4.4: Code development flow for TI C6000 DSP (from [20]).

4.3.1 Compiler Optimization Options [20]

In this subsection, we introduce the compiler options that control the operation of the compiler. The CCS compiler offers high-level language support by transforming C/C++ code into more efficient assembly language source code. The compiler options can be used to optimize the code size or the execution performance.

Some compiler options that are more important in our code development are:

- -o0:
  - Performs control-flow-graph simplification.
  - Allocates variables to registers.
  - Performs loop rotation.
  - Eliminates unused code.
  - Simplifies expressions and statements.
  - Expands calls to functions declared inline.
- -o1. Performs all -o0 optimization, and:
  - Performs local copy/constant propagation.
  - Removes unused assignments.
  - Eliminates local common expressions.
- -o2. Performs all -o1 optimizations, and:
  - Performs software pipelining.
  - Performs loop optimizations.
  - Eliminates global common subexpressions.

  - Eliminates global unused assignments.
  - Converts array references in loops to incremented pointer form.
  - Performs loop unrolling.

- -o3. Performs all -o2 optimizations, and:
  - Removes all functions that are never called.
  - Simplifies functions with return values that are never used.
  - Inline calls to small functions.
  - Reorders function declarations so that the attributes of called functions are known when the caller is optimized.
  - Propagates arguments into function bodies when all calls pass the same value in the same argument position.
  - Identifies file-level variable characteristics.

- -k: Keep the assembly file to analyze the compiler feedback.

- -pm -op2: In the CCS compiler option, -pm and -op2 are combined into one option.
  - -pm: Gives the compiler global access to the whole program or module and allows it to be more aggressive in ruling out dependencies.
  - -op2: Specifies that the module contains no functions or variables that are called or modified from outside the source code provided to the compiler. This improves variable analysis and allowed assumptions.

- -mw: Produce additional compiler feedback. This option has no performance or code size impact.

- -mi: Describes the interrupt threshold to the compiler. If the compiler knows that no interrupts will occur in the code, it can avoid enabling and disabling interrupts before and after software-pipelined loops for improvement in code size and performance. In addition, there is potential for performance improvement where interrupt registers may be utilized in high register pressure loops.

4.3.2 Software Pipelining [21]

Software pipelining is a technique used to schedule instructions from a loop so that multiple iterations of the loop can be executed in parallel. Figure 4.5 illustrates a software pipelined loop. The stages of the loop are represented by A, B, C, D and E. In this figure, a maximum of five iterations of the loop can execute at one time. The shaded area represents the loop kernel. In the loop kernel, all five stages execute in parallel. The area above the is known as the pipelined loop prolog, and the area below the kernel is known as the pipelined loop epilog.

But under the conditions listed below, the compiler will not do software pipelining:

- If a register value lives too long, the code is not software-pipelined.
- If a loop has complex condition code within the body that requires more than five condition registers, the loop is not software pipelined.
- A software-pipelined loop cannot contain function calls, including code that calls the run-time support routines.
- In a sequence of nested loops, the innermost loop is the only one that can be software-pipelined.
- If a loop contains conditional break, it is not software-pipelined.

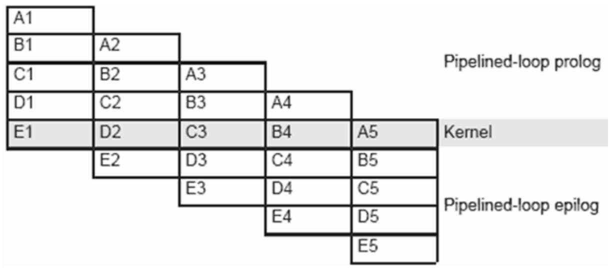


Figure 4.5: Software-pipelined loop (from [22]).

4.3.3 Loop Unrolling

Loop unrolling can be used to make programs more suitable for parallel processing. The idea of loop unrolling is to save time by reducing the number of overhead instructions that the computer has to execute in a loop, thus improving the cache hit rate and reducing branching. To achieve this, the instructions that are called in multiple iterations of the loop are combined into a single iteration.

Chapter 5

Fixed-Point DSP Implementation

We have introduced the ranging techniques for IEEE 802.16e OFDMA and the DSP implementation environment. Now we consider the fixed-point implementation of the ranging techniques on DSP in this chapter.

5.1 Fixed-Point Implementation

The contents of the first two paragraphs are taken from [11].

It is convenient to employ floating-point computation to verify the performance of the algorithm during algorithm development. But practical transceiver implementations normally use fixed-point computation for power, speed, and hardware cost reasons. The DSP employed in this work, TI's TMS320C6416T, is also of the fixed-point category, which can perform fixed-point computations more efficiently than floating-point ones. We consider fixed-point DSP implementation in this work. In addition, we try to make the resulting program run fast by making efficient use of the DSP's features.

The C6416T CPU contains 8 parallel 32-bit function units, two of which are multipliers and the remaining six can do a number of arithmetic, logic, and memory access operations. There is also flexibility in arranging the data so that each function unit can do double 16-bit

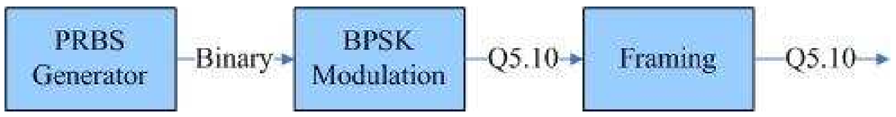


Figure 5.1: Fixed-point data formats used at different points in the ranging signal generator.

or quadruple 8-bit operations. Running at 1 GHz, the peak performance is 8000 MIPS. For many transmission systems, 32-bit computations are an overkill and 8-bit computations do not provide the necessary accuracy. This appears to be the case with the present system, too. Hence we choose to use the 16-bit data type mostly, with careful selection of dynamic range of the data at different points in the function chain. Simulation results confirm that this is an appropriate choice. In fact, a TI document also suggests use of the short data type (16-bit) for fixed-point multiplication inputs whenever possible [20].

The ranging system is one part of the UL system shown in Figure 3.1. Since the other parts of the UL system employ $Q5.10$, we also use $Q5.10$ in most parts of the ranging system except the last two parts in ranging signal system. We use $Q6.9$ in the norm and detection and estimation blocks. The data formats are shown in Figs. 5.1 and 5.2 for the ranging signal generator and receiver, respectively. Note that $Qx.y$ means there are x bits before the binary points and y bits after. In every case, $x + y = 15$ because the sign takes one bit. We discuss the details of some blocks of the ranging system in the following subsections.

5.1.1 PRBS Generator

We use a lookup table in the implementation of the PRBS generator. As long as UL_PermBase does not change, we construct a table of initialization sequences and generate codes by table-lookup method. That is, the table contains the initialization sequences of each code which is 144 bits long. So we only need to clock the PRBS generator 144 times on average to generate a code. When UL_PermBase changes, we reconstruct the table.

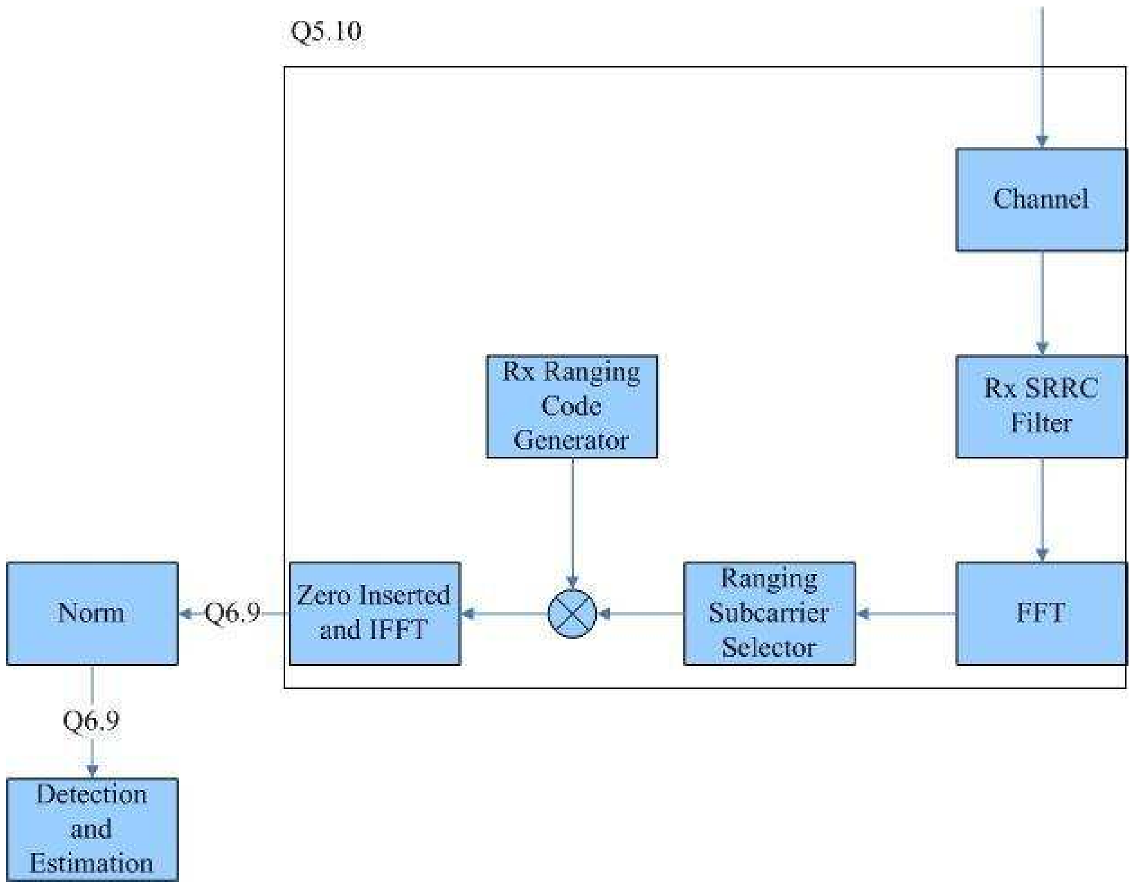


Figure 5.2: Fixed-point data formats used at different points in the ranging signal receiver.

We further use an approach to generate each code with less complexity. More details are given in a later section.

5.1.2 Norm and Detection and Estimation

For detection method 1, we need to calculate the average of two norms. This includes a summation operation and we find that overflow may occur in some cases when we use $Q5.10$. Therefore, we employ $Q6.9$ here.

5.2 DSP Optimization

In this section, we introduce some optimization techniques. Besides utilizing the compiler options mentioned before, we tune the program for compiler to software-pipeline the loop automatically. Moreover, we unroll the loop by 4 times to speed up some sections where the compiler cannot software-pipeline. Fig. 5.3 is an example showing the unrolling of some loops for ranging subcarrier selection and multiplication function. Figures 5.4 and 5.5 are a part of the assembly code for this function. Figure 5.6 is the corresponding software-pipeline information. Some other optimizations such as employing the TI C64x DSP library is shown in the following subsections.

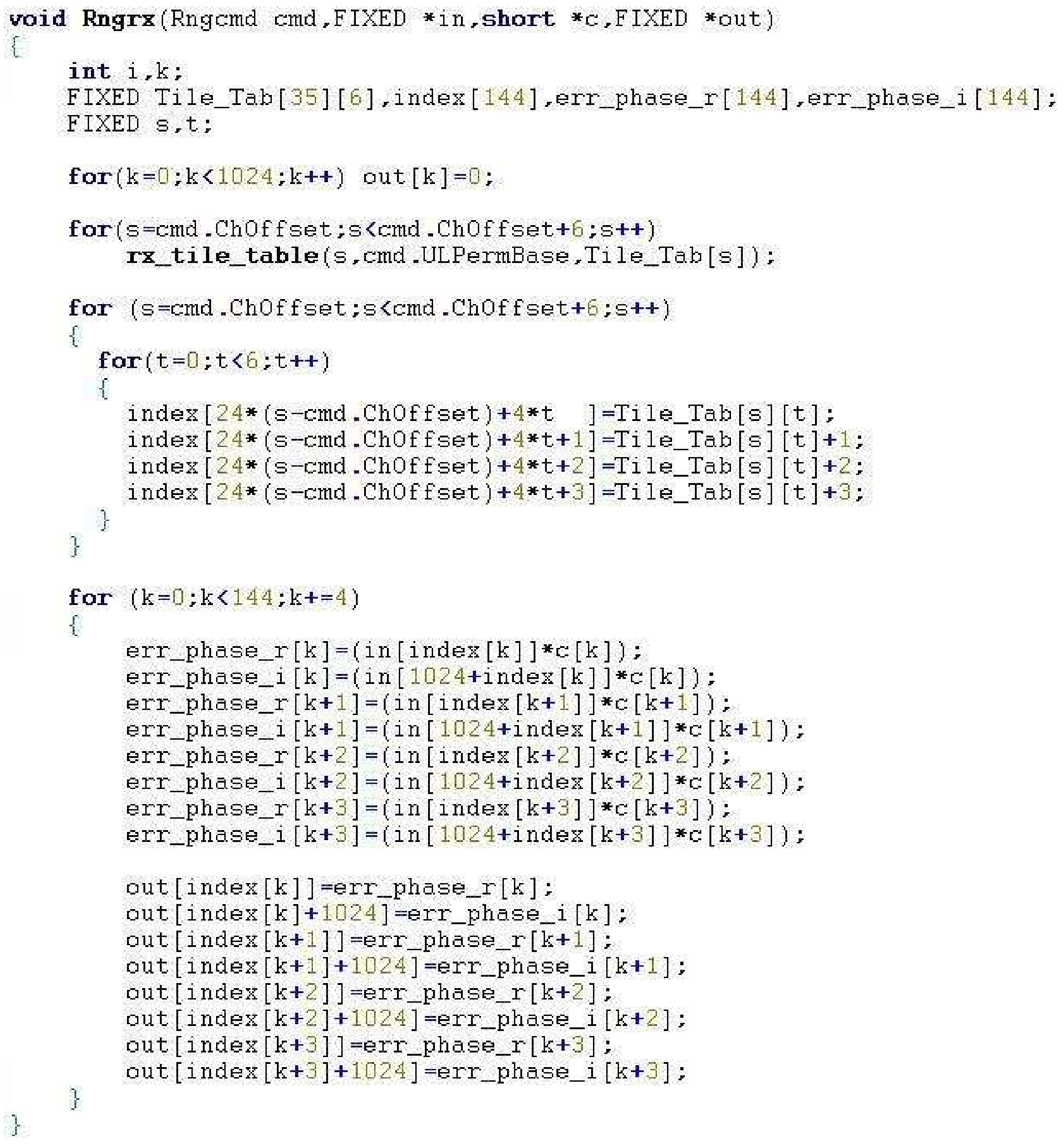
5.2.1 PRBS Generator

The PRBS generator is shown in Figure 2.6. In the following we discuss a straightforward method and a more sophisticated method to generate a 144-bit code.

Straightforward Method

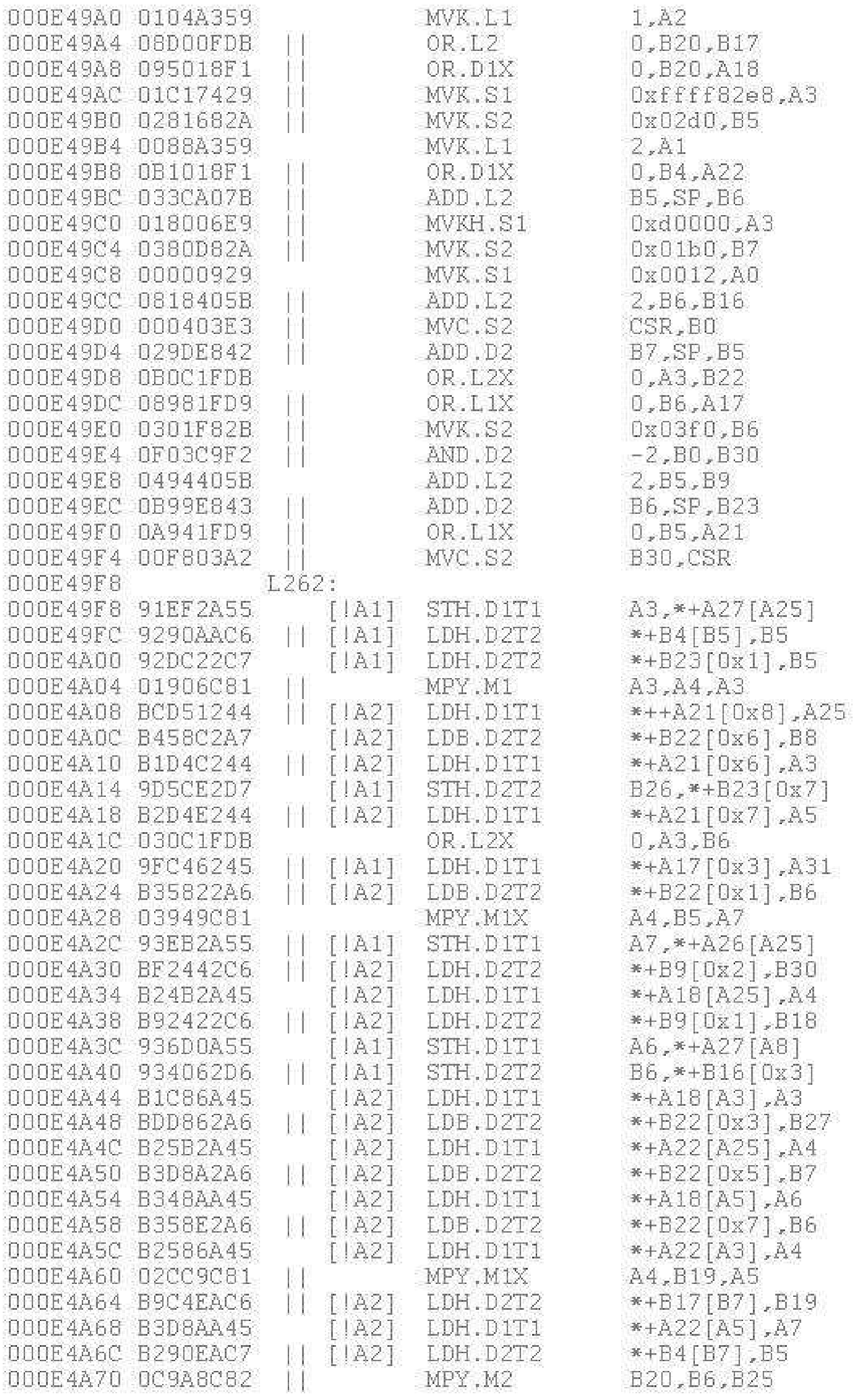
We denote the 15-bit initialization sequence as S_i, where i is the index of clock cycles, as

$$S_i = [S(0)_i \;\; S(1)_i \;\; \dots \;\; S(14)_i]^T. \tag{5.1}$$



```c
void Rngrx(Rngcmd cmd,FIXED *in,short *c,FIXED *out)
{
    int i,k;
    FIXED Tile_Tab[35][6],index[144],err_phase_r[144],err_phase_i[144];
    FIXED s,t;

    for(k=0;k<1024;k++) out[k]=0;

    for(s=cmd.ChOffset;s<cmd.ChOffset+6;s++)
        rx_tile_table(s,cmd.ULPermBase,Tile_Tab[s]);

    for (s=cmd.ChOffset;s<cmd.ChOffset+6;s++)
    {
      for(t=0;t<6;t++)
      {
        index[24*(s-cmd.ChOffset)+4*t  ]=Tile_Tab[s][t];
        index[24*(s-cmd.ChOffset)+4*t+1]=Tile_Tab[s][t]+1;
        index[24*(s-cmd.ChOffset)+4*t+2]=Tile_Tab[s][t]+2;
        index[24*(s-cmd.ChOffset)+4*t+3]=Tile_Tab[s][t]+3;
      }
    }

    for (k=0;k<144;k+=4)
    {
        err_phase_r[k]=(in[index[k]]*c[k]);
        err_phase_i[k]=(in[1024+index[k]]*c[k]);
        err_phase_r[k+1]=(in[index[k+1]]*c[k+1]);
        err_phase_i[k+1]=(in[1024+index[k+1]]*c[k+1]);
        err_phase_r[k+2]=(in[index[k+2]]*c[k+2]);
        err_phase_i[k+2]=(in[1024+index[k+2]]*c[k+2]);
        err_phase_r[k+3]=(in[index[k+3]]*c[k+3]);
        err_phase_i[k+3]=(in[1024+index[k+3]]*c[k+3]);

        out[index[k]]=err_phase_r[k];
        out[index[k]+1024]=err_phase_i[k];
        out[index[k+1]]=err_phase_r[k+1];
        out[index[k+1]+1024]=err_phase_i[k+1];
        out[index[k+2]]=err_phase_r[k+2];
        out[index[k+2]+1024]=err_phase_i[k+2];
        out[index[k+3]]=err_phase_r[k+3];
        out[index[k+3]+1024]=err_phase_i[k+3];
    }
}
```

Figure 5.3: A part of C code for ranging subcarrier selection and multiplication function.

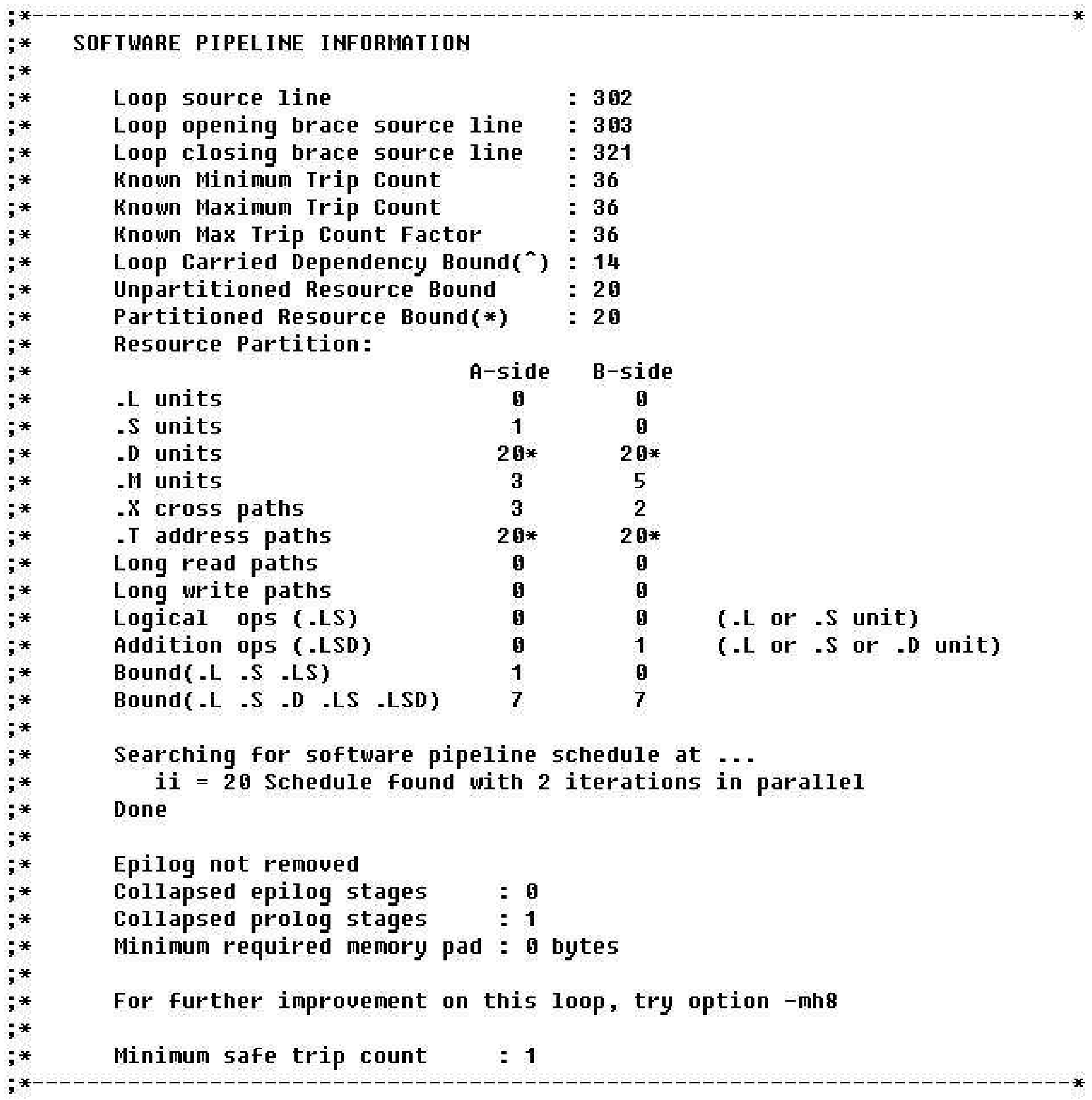


```
000E49A0 0104A359             MVK.L1      1,A2
000E49A4 08D00FDB  ||         OR.L2       0,B20,B17
000E49A8 095018F1  ||         OR.D1X      0,B20,A18
000E49AC 01C17429  ||         MVK.S1      0xffff82e8,A3
000E49B0 0281682A  ||         MVK.S2      0x02d0,B5
000E49B4 0088A359             MVK.L1      2,A1
000E49B8 0B1018F1  ||         OR.D1X      0,B4,A22
000E49BC 033CA07B  ||         ADD.L2      B5,SP,B6
000E49C0 018006E9  ||         MVKH.S1     0xd0000,A3
000E49C4 0380D82A  ||         MVK.S2      0x01b0,B7
000E49C8 00000929             MVK.S1      0x0012,A0
000E49CC 0818405B  ||         ADD.L2      2,B6,B16
000E49D0 000403E3  ||         MVC.S2      CSR,B0
000E49D4 029DE842  ||         ADD.D2      B7,SP,B5
000E49D8 0B0C1FDB             OR.L2X      0,A3,B22
000E49DC 08981FD9  ||         OR.L1X      0,B6,A17
000E49E0 0301F82B  ||         MVK.S2      0x03f0,B6
000E49E4 0F03C9F2  ||         AND.D2      -2,B0,B30
000E49E8 0494405B             ADD.L2      2,B5,B9
000E49EC 0B99E843  ||         ADD.D2      B6,SP,B23
000E49F0 0A941FD9  ||         OR.L1X      0,B5,A21
000E49F4 00F803A2  ||         MVC.S2      B30,CSR
000E49F8          L262:
000E49F8 91EF2A55     [!A1]   STH.D1T1    A3,*+A27[A25]
000E49FC 9290AAC6  || [!A1]   LDH.D2T2    *+B4[B5],B5
000E4A00 92DC22C7     [!A1]   LDH.D2T2    *+B23[0x1],B5
000E4A04 01906C81  ||         MPY.M1      A3,A4,A3
000E4A08 BCD51244  || [!A2]   LDH.D1T1    *++A21[0x8],A25
000E4A0C B458C2A7     [!A2]   LDB.D2T2    *+B22[0x6],B8
000E4A10 B1D4C244  || [!A2]   LDH.D1T1    *+A21[0x6],A3
000E4A14 9D5CE2D7     [!A1]   STH.D2T2    B26,*+B23[0x7]
000E4A18 B2D4E244  || [!A2]   LDH.D1T1    *+A21[0x7],A5
000E4A1C 030C1FDB             OR.L2X      0,A3,B6
000E4A20 9FC46245  || [!A1]   LDH.D1T1    *+A17[0x3],A31
000E4A24 B35822A6  || [!A2]   LDB.D2T2    *+B22[0x1],B6
000E4A28 03949C81             MPY.M1X     A4,B5,A7
000E4A2C 93EB2A55  || [!A1]   STH.D1T1    A7,*+A26[A25]
000E4A30 BF2442C6  || [!A2]   LDH.D2T2    *+B9[0x2],B30
000E4A34 B24B2A45     [!A2]   LDH.D1T1    *+A18[A25],A4
000E4A38 B92422C6  || [!A2]   LDH.D2T2    *+B9[0x1],B18
000E4A3C 936D0A55     [!A1]   STH.D1T1    A6,*+A27[A8]
000E4A40 934062D6  || [!A1]   STH.D2T2    B6,*+B16[0x3]
000E4A44 B1C86A45     [!A2]   LDH.D1T1    *+A18[A3],A3
000E4A48 BDD862A6  || [!A2]   LDB.D2T2    *+B22[0x3],B27
000E4A4C B25B2A45     [!A2]   LDH.D1T1    *+A22[A25],A4
000E4A50 B3D8A2A6  || [!A2]   LDB.D2T2    *+B22[0x5],B7
000E4A54 B348AA45     [!A2]   LDH.D1T1    *+A18[A5],A6
000E4A58 B358E2A6  || [!A2]   LDB.D2T2    *+B22[0x7],B6
000E4A5C B2586A45     [!A2]   LDH.D1T1    *+A22[A3],A4
000E4A60 02CC9C81  ||         MPY.M1X     A4,B19,A5
000E4A64 B9C4EAC6  || [!A2]   LDH.D2T2    *+B17[B7],B19
000E4A68 B3D8AA45     [!A2]   LDH.D1T1    *+A22[A5],A7
000E4A6C B290EAC7  || [!A2]   LDH.D2T2    *+B4[B7],B5
000E4A70 0C9A8C82  ||         MPY.M2      B20,B6,B25
```

Figure 5.4: A part of the assembly code for ranging subcarrier selection and multiplication function (1/2).

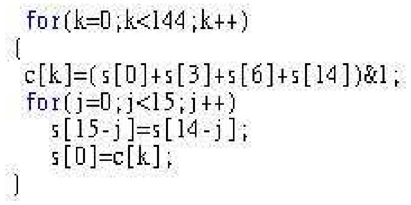
```
000E4A74 9C5CC2C7      [!A1]  LDH.D2T2        *+B23[0x6],B24
000E4A78 92448245   || [!A1]  LDH.D1T1        *+A17[0x4],A4
000E4A7C 01A07C81   ||        MPY.M1X         A3,B8,A3
000E4A80 0A1AAC82   ||        MPY.M2          B21,B6,B20
000E4A84 92D44045      [!A1]  LDH.D1T1        *-A21[0x2],A5
000E4A88 B413CAC6   || [!A2]  LDH.D2T2        *+B4[B30],B8
000E4A8C 92E90A57      [!A1]  STH.D1T2        B5,*+A26[A8]
000E4A90 93DC82D5   || [!A1]  STH.D2T1        A7,*+B23[0x4]
000E4A94 0A927C82   ||        MPY.M2X         B19,A4,B21
000E4A98 995C82C7      [!A1]  LDH.D2T2        *+B23[0x4],B18
000E4A9C B4D44245   || [!A2]  LDH.D1T1        *+A21[0x2],A9
000E4AA0 0E111C82   ||        MPY.M2X         B8,A4,B28
000E4AA4 945CE2C7      [!A1]  LDH.D2T2        *+B23[0x7],B8
000E4AA8 B4542244   || [!A2]  LDH.D1T1        *+A21[0x1],A8
000E4AAC B3924AC7      [!A2]  LDH.D2T2        *+B4[B18],B7
000E4AB0 B2C51254   || [!A2]  STH.D1T1        A5,*++A17[0x8]
000E4AB4 9E6D2A55      [!A1]  STH.D1T1        A28,*+A27[A9]
000E4AB8 B9C64AC7   || [!A2]  LDH.D2T2        *+B17[B18],B19
000E4ABC 0318DC81   ||        MPY.M1X         A6,B6,A6
000E4AC0 0D1CDC82   ||        MPY.M2X         B6,A7,B26
000E4AC4 9EE92A55      [!A1]  STH.D1T1        A29,*+A26[A9]
000E4AC8 BA5842A7   || [!A2]  LDB.D2T2        *+B22[0x2],B20
000E4ACC 0FED0C82   ||        MPY.M2          B8,B27,B31
000E4AD0 9FEFCA55      [!A1]  STH.D1T1        A31,*+A27[A30]
000E4AD4 031E6C83   ||        MPY.M2          B19,B7,B6
000E4AD8 09A512C6   ||        LDH.D2T2        *++B9[0x8],B19
000E4ADC B1C4C255      [!A2]  STH.D1T1        A3,*+A17[0x6]
000E4AE0 BA4112D7   || [!A2]  STH.D2T2        B20,*++B16[0x8]
000E4AE4 029CAC82   ||        MPY.M2          B5,B7,B5
000E4AE8 9EEBCA57      [!A1]  STH.D1T2        B29,*+A26[A30]
000E4AEC B1C7CAC4   || [!A2]  LDH.D2T1        *+B17[B30],A3
000E4AF0 926E0A55      [!A1]  STH.D1T1        A4,*+A27[A16]
000E4AF4 B34082D6   || [!A2]  STH.D2T2        B6,*+B16[0x4]
000E4AF8 B8548245      [!A2]  LDH.D1T1        *+A21[0x4],A16
000E4AFC 03526C83   ||        MPY.M2          B19,B20,B6
000E4B00 03A482C6   ||        LDH.D2T2        *+B9[0x4],B7
000E4B04 99DCA2C7      [!A1]  LDH.D2T2        *+B23[0x5],B19
000E4B08 B1C40244   || [!A2]  LDH.D1T1        *+A17[0x0],A3
000E4B0C B344E255      [!A2]  STH.D1T1        A6,*+A17[0x7]
000E4B10 0AC66AC6   ||        LDH.D2T2        *+B17[B19],B21
000E4B14 BC44A245      [!A2]  LDH.D1T1        *+A17[0x5],A24
000E4B18 026C7C81   ||        MPY.M1X         A3,B27,A4
000E4B1C 0A126AC6   ||        LDH.D2T2        *+B4[B19],B20
000E4B20 996A0A57      [!A1]  STH.D1T2        B18,*+A26[A16]
000E4B24 B25882A5   || [!A2]  LDB.D2T1        *+B22[0x4],A4
000E4B28 0950EC82   ||        MPY.M2          B7,B20,B18
000E4B2C 9C6E8A55      [!A1]  STH.D1T1        A24,*+A27[A20]
000E4B30 BADD12D6   || [!A2]  STH.D2T2        B21,*++B23[0x8]
000E4B34 99EA8A57      [!A1]  STH.D1T2        B19,*+A26[A20]
000E4B38 B3DC02C5   || [!A2]  LDH.D2T1        *+B23[0x0],A7
000E4B3C 09901FDA   ||        OR.L2X          0,A4,B19
000E4B40 B3442245      [!A2]  LDH.D1T1        *+A17[0x1],A6
000E4B44 BFDC62D6   || [!A2]  STH.D2T2        B31,*+B23[0x3]
000E4B48 B95C42D7      [!A2]  STH.D2T2        B18,*+B23[0x2]
```

Figure 5.5: A part of the assembly code for ranging subcarrier selection and multiplication function (2/2).



```
;*----------------------------------------------------------------------------*
;*   SOFTWARE PIPELINE INFORMATION
;*
;*      Loop source line                 : 302
;*      Loop opening brace source line   : 303
;*      Loop closing brace source line   : 321
;*      Known Minimum Trip Count         : 36
;*      Known Maximum Trip Count         : 36
;*      Known Max Trip Count Factor      : 36
;*      Loop Carried Dependency Bound(^) : 14
;*      Unpartitioned Resource Bound     : 20
;*      Partitioned Resource Bound(*)    : 20
;*      Resource Partition:
;*                                A-side   B-side
;*      .L units                     0        0
;*      .S units                     1        0
;*      .D units                    20*      20*
;*      .M units                     3        5
;*      .X cross paths               3        2
;*      .T address paths            20*      20*
;*      Long read paths              0        0
;*      Long write paths             0        0
;*      Logical  ops (.LS)           0        0     (.L or .S unit)
;*      Addition ops (.LSD)          0        1     (.L or .S or .D unit)
;*      Bound(.L .S .LS)             1        0
;*      Bound(.L .S .D .LS .LSD)     7        7
;*
;*      Searching for software pipeline schedule at ...
;*         ii = 20 Schedule found with 2 iterations in parallel
;*      Done
;*
;*      Epilog not removed
;*      Collapsed epilog stages     : 0
;*      Collapsed prolog stages     : 1
;*      Minimum required memory pad : 0 bytes
;*
;*      For further improvement on this loop, try option -mh8
;*
;*      Minimum safe trip count     : 1
;*----------------------------------------------------------------------------*
```

Figure 5.6: Software-pipeline information for ranging subcarrier selection and multiplication function.



```
 for(k=0;k<144;k++)
{
 c[k]=(s[0]+s[3]+s[6]+s[14])&1;
 for(j=0;j<15;j++)
   s[15-j]=s[14-j];
   s[0]=c[k];
}
```

Figure 5.7: C code for the straightforward method.

And let

$$A = \begin{bmatrix} 1&0&0&1&0&0&1&0&0&0&0&0&0&0&1 \\ 1&0&0&0&0&0&0&0&0&0&0&0&0&0&0 \\ 0&1&0&0&0&0&0&0&0&0&0&0&0&0&0 \\ 0&0&1&0&0&0&0&0&0&0&0&0&0&0&0 \\ 0&0&0&1&0&0&0&0&0&0&0&0&0&0&0 \\ 0&0&0&0&1&0&0&0&0&0&0&0&0&0&0 \\ 0&0&0&0&0&1&0&0&0&0&0&0&0&0&0 \\ 0&0&0&0&0&0&1&0&0&0&0&0&0&0&0 \\ 0&0&0&0&0&0&0&1&0&0&0&0&0&0&0 \\ 0&0&0&0&0&0&0&0&1&0&0&0&0&0&0 \\ 0&0&0&0&0&0&0&0&0&1&0&0&0&0&0 \\ 0&0&0&0&0&0&0&0&0&0&1&0&0&0&0 \\ 0&0&0&0&0&0&0&0&0&0&0&1&0&0&0 \\ 0&0&0&0&0&0&0&0&0&0&0&0&1&0&0 \\ 0&0&0&0&0&0&0&0&0&0&0&0&0&1&0 \end{bmatrix}. \tag{5.2}$$

Then we have

$$S_{i+1} = A \cdot S_i. \tag{5.3}$$

We generate a code by clocking the PRBS generator 144 times. The corresponding c code is shown in Figure 5.7. It needs 432 additions, 144 ands and 2304 moves.

Modified Method 1

An alternative approach similar to the lookahead methods mentioned in [23] and [24] is precalculating A^{12}. Note that in the matrix multiplication here, the bit-wise addition is simply an exclusive-OR operation.

$$A^{12} = \begin{bmatrix} 0&1&0&0&0&1&1&0&1&0&0&1&1&0&0 \\ 0&0&1&0&0&0&1&1&0&1&0&0&1&1&0 \\ 0&0&0&1&0&0&0&1&1&0&1&0&0&1&1 \\ 1&1&0&0&0&0&0&1&1&1&0&1&0&0&1 \\ 1&0&1&0&1&0&0&1&1&1&1&0&1&0&0 \\ 0&1&0&1&0&1&0&0&1&1&1&1&0&1&0 \\ 0&0&1&0&1&0&1&0&0&1&1&1&1&0&1 \\ 1&1&0&1&1&1&0&0&0&0&1&1&1&1&0 \\ 0&1&1&0&1&1&1&0&0&0&0&1&1&1&1 \\ 1&1&1&1&1&1&1&0&0&0&0&0&1&1&1 \\ 1&0&1&1&0&1&1&0&0&0&0&0&0&1&1 \\ 1&0&0&1&0&0&1&0&0&0&0&0&0&0&1 \\ 1&0&0&0&0&0&0&0&0&0&0&0&0&0&0 \\ 0&1&0&0&0&0&0&0&0&0&0&0&0&0&0 \\ 0&0&1&0&0&0&0&0&0&0&0&0&0&0&0 \end{bmatrix}. \tag{5.4}$$

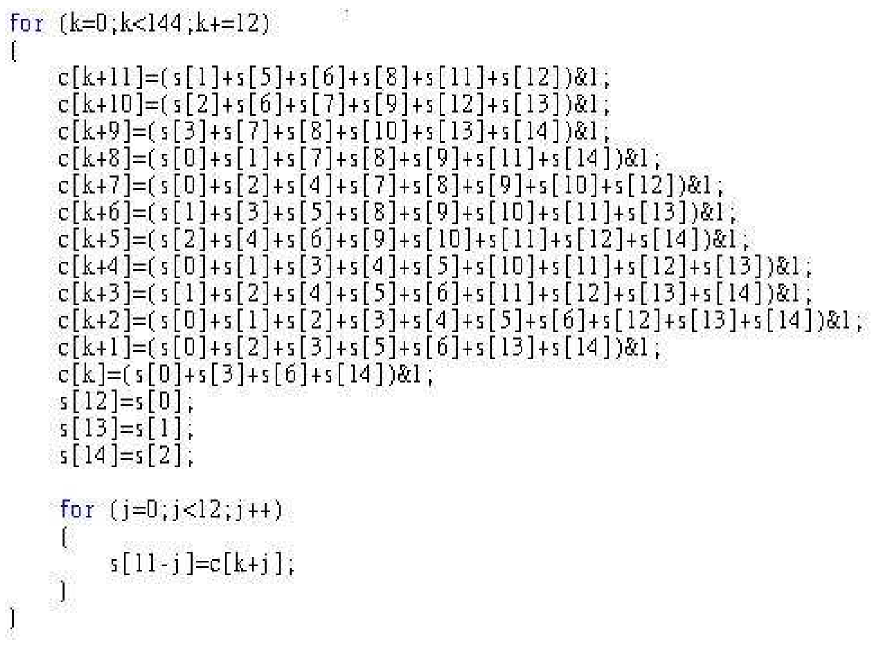
Then we have

$$S_{i+12} = A^{12} \cdot S_i. \tag{5.5}$$

A code is generated by clocking 12 times. The for-loop now has 12 iterations only. The corresponding C code is shown in Figure 5.8. It needs 912 additions, 144 ands and 324 moves.

Modified Method 2

We find that the above modified method does not improve the latency. So, we precalculate A^6 here rather than A_{12}.



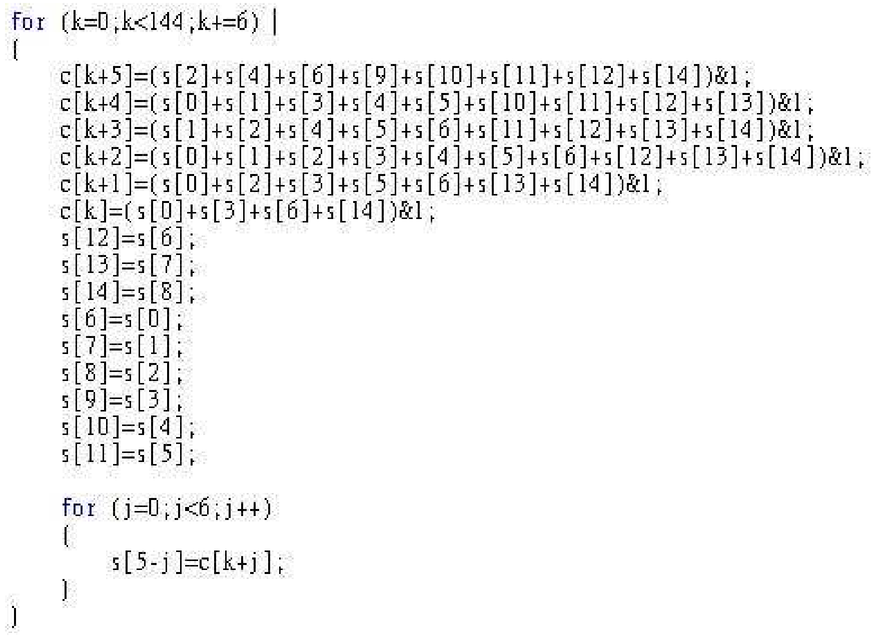
```
for (k=0;k<144;k+=12)
{
    c[k+11]=(s[1]+s[5]+s[6]+s[8]+s[11]+s[12])&1;
    c[k+10]=(s[2]+s[6]+s[7]+s[9]+s[12]+s[13])&1;
    c[k+9]=(s[3]+s[7]+s[8]+s[10]+s[13]+s[14])&1;
    c[k+8]=(s[0]+s[1]+s[7]+s[8]+s[9]+s[11]+s[14])&1;
    c[k+7]=(s[0]+s[2]+s[4]+s[7]+s[8]+s[9]+s[10]+s[12])&1;
    c[k+6]=(s[1]+s[3]+s[5]+s[8]+s[9]+s[10]+s[11]+s[13])&1;
    c[k+5]=(s[2]+s[4]+s[6]+s[9]+s[10]+s[11]+s[12]+s[14])&1;
    c[k+4]=(s[0]+s[1]+s[3]+s[4]+s[5]+s[10]+s[11]+s[12]+s[13])&1;
    c[k+3]=(s[1]+s[2]+s[4]+s[5]+s[6]+s[11]+s[12]+s[13]+s[14])&1;
    c[k+2]=(s[0]+s[1]+s[2]+s[3]+s[4]+s[5]+s[6]+s[12]+s[13]+s[14])&1;
    c[k+1]=(s[0]+s[2]+s[3]+s[5]+s[6]+s[13]+s[14])&1;
    c[k]=(s[0]+s[3]+s[6]+s[14])&1;
    s[12]=s[0];
    s[13]=s[1];
    s[14]=s[2];

    for (j=0;j<12;j++)
    {
        s[11-j]=c[k+j];
    }
}
```

Figure 5.8: The c code for modified method 1.

$$A^6 = \begin{bmatrix} 0&0&1&0&1&0&1&0&0&1&1&1&1&0&1\\ 1&1&0&1&1&1&0&0&0&0&1&1&1&1&0\\ 0&1&1&0&1&1&1&0&0&0&0&1&1&1&1\\ 1&1&1&1&1&1&1&0&0&0&0&0&1&1&1\\ 1&0&1&1&0&1&1&0&0&0&0&0&0&1&1\\ 1&0&0&1&0&0&1&0&0&0&0&0&0&0&1\\ 1&0&0&0&0&0&0&0&0&0&0&0&0&0&0\\ 0&1&0&0&0&0&0&0&0&0&0&0&0&0&0\\ 0&0&1&0&0&0&0&0&0&0&0&0&0&0&0\\ 0&0&0&1&0&0&0&0&0&0&0&0&0&0&0\\ 0&0&0&0&1&0&0&0&0&0&0&0&0&0&0\\ 0&0&0&0&0&1&0&0&0&0&0&0&0&0&0\\ 0&0&0&0&0&0&1&0&0&0&0&0&0&0&0\\ 0&0&0&0&0&0&0&1&0&0&0&0&0&0&0\\ 0&0&0&0&0&0&0&0&1&0&0&0&0&0&0 \end{bmatrix}. \tag{5.6}$$

Then we have

$$S_{i+6} = A^6 \cdot S_i. \tag{5.7}$$



```
for (k=0;k<144;k+=6)
{
    c[k+5]=(s[2]+s[4]+s[6]+s[9]+s[10]+s[11]+s[12]+s[14])&1;
    c[k+4]=(s[0]+s[1]+s[3]+s[4]+s[5]+s[10]+s[11]+s[12]+s[13])&1;
    c[k+3]=(s[1]+s[2]+s[4]+s[5]+s[6]+s[11]+s[12]+s[13]+s[14])&1;
    c[k+2]=(s[0]+s[1]+s[2]+s[3]+s[4]+s[5]+s[6]+s[12]+s[13]+s[14])&1;
    c[k+1]=(s[0]+s[2]+s[3]+s[5]+s[6]+s[13]+s[14])&1;
    c[k]=(s[0]+s[3]+s[6]+s[14])&1;
    s[12]=s[6];
    s[13]=s[7];
    s[14]=s[8];
    s[6]=s[0];
    s[7]=s[1];
    s[8]=s[2];
    s[9]=s[3];
    s[10]=s[4];
    s[11]=s[5];

    for (j=0;j<6;j++)
    {
        s[5-j]=c[k+j];
    }
}
```

Figure 5.9: C code for the modified method 2.

A code is generated by clocking 24 times. The for-loop now has 24 iterations. The corresponding C code is shown in Figure 5.9. It needs 984 additions, 144 ands and 504 moves.

CCS Cycles Comparison

The following shows the cycles comparison. We use the TI Code Composer Studio (CCS) to simulate the execution cycles. Interestingly, modified method 1 does not outperform the straightforward method. But modified method 2 achieves 54.3% reduction of cycles. Therefore, we employ modified method 2 in our implementation.

5.2.2 IFFT and FFT

The DSPLIB contains FFT functions employing 32 × 32-bit and 16×16-bit multiplications. The former has higher computational complexity. We resolve to use the latter.

Table 5.1: Code Size and Execution Cycles of PRBS Generation of A Code

Method	Code size (Bytes)	Clock cycles
Straightforward method	1920	2888
Modified method 1	2368	2943
Modified method 2	2496	1321

The function `DSP_fft16x16r()` computes a complex forward mixed radix FFT with scaling, rounding and digit reversal. The input data $x[]$ and the coefficients $w[]$ are arrays of complex numbers, with the numbers stored in interleaved 16-bit real and imaginary parts. The output data are returned in a separate array $y[]$ in normal order, also complex with interleaved 16-bit real and imaginary parts. The code uses a special ordering of FFT coefficients (also called twiddle factors). These twiddle factors are generated by using the function `tw_fft16x16()` provided by TI.

5.2.3 Fixed-Point Simulation Results

We present some simulation results in this section. All simulation parameters and environments are similar to those in Section 3.3, but here we have fixed-point implementation. For comparison, floating-point simulation results are also presented together with the fixed-point results.

Figures 5.10, 5.11 and 5.12 show the total failure rate, missed detection rate and false alarm rate of ranging code detection with detection method 1 and method 2 in AWGN channel. We can see that method 1 obviously has better performance than method 2, especially at low SNR.

Figure 5.13 shows the average successful detection rate of each user, which is always very close to 1 here. Figure 5.14 depicts the average root mean square error (RMSE) of timing

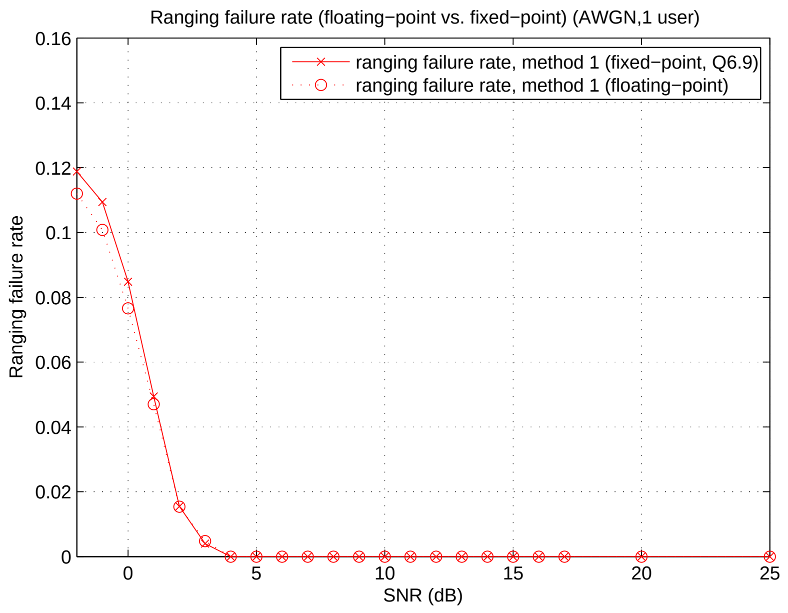


Figure 5.10: Ranging failure rate of method 1 in AWGN channel with one ranging user.

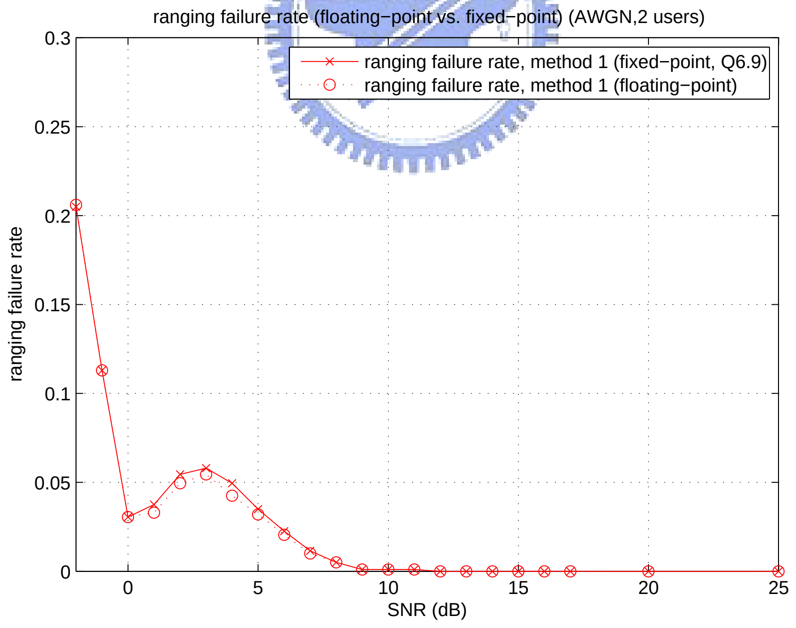


Figure 5.11: Ranging failure rate of method 1 in AWGN channel with two ranging users.

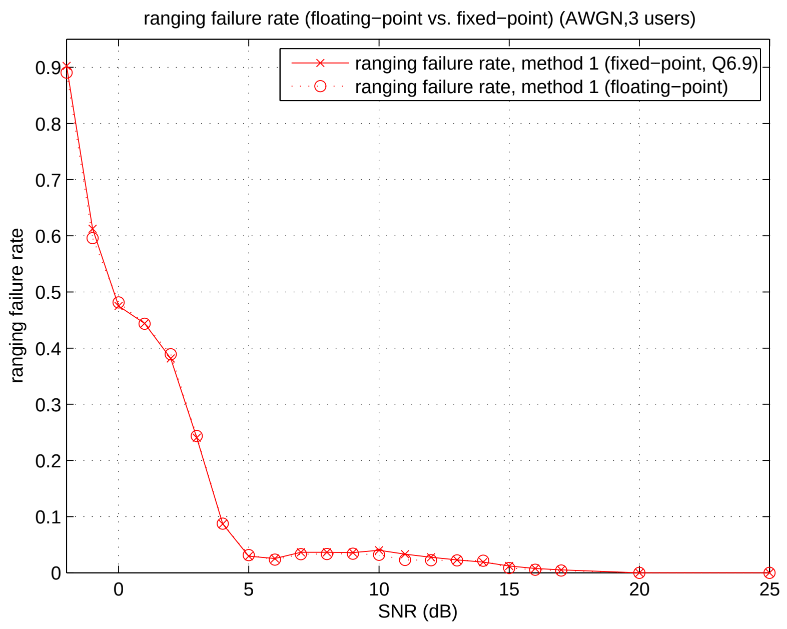


Figure 5.12: Ranging failure rate of method 1 in AWGN channel with three ranging users.

offset estimation. We see that the estimation is perfect as long as SNR is larger than 5dB.

The total failure rate, missed detection rate and false alarm rate of ranging code detection with detection method 1 and method 2 in the SUI-3 channel are depicted in Figures 5.15, 5.16 and 5.17.

Figure 5.18 shows the average successful detection rate of each user. Note that the product of all users' detection failure rate is the missed detection rate since the transmission of each user is independent. Figure 5.19 depicts the average RMSE of timing offset estimation. The RMSE are roughly between 1.9 and 3 samples over the range of SNR values considered.

We can see that the fixed-point simulation results are all very close to those of floating-point simulation.

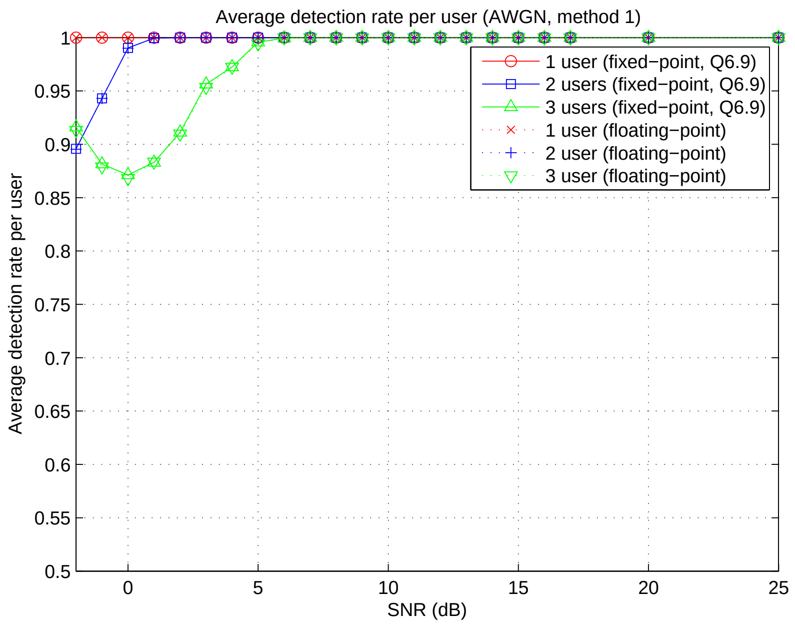


Figure 5.13: Average success rate of each user with method 1 in AWGN channel.

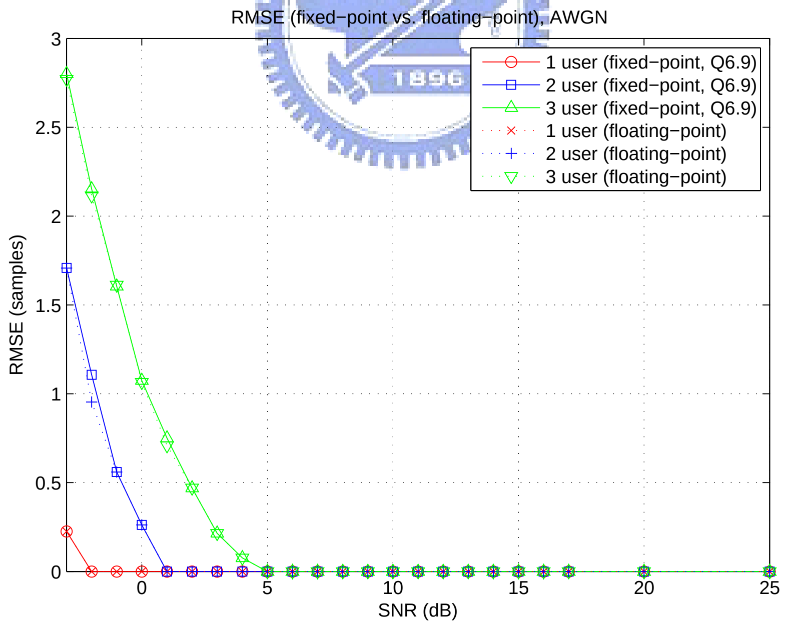


Figure 5.14: RMSE of timing offset estimation in AWGN channel.

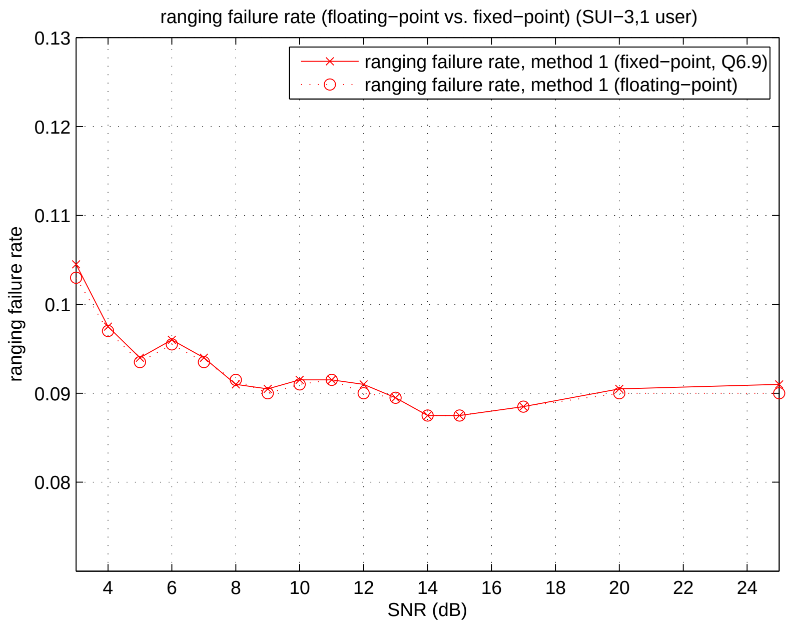


Figure 5.15: Ranging failure rate of method 1 in SUI-3 channel with one ranging user.

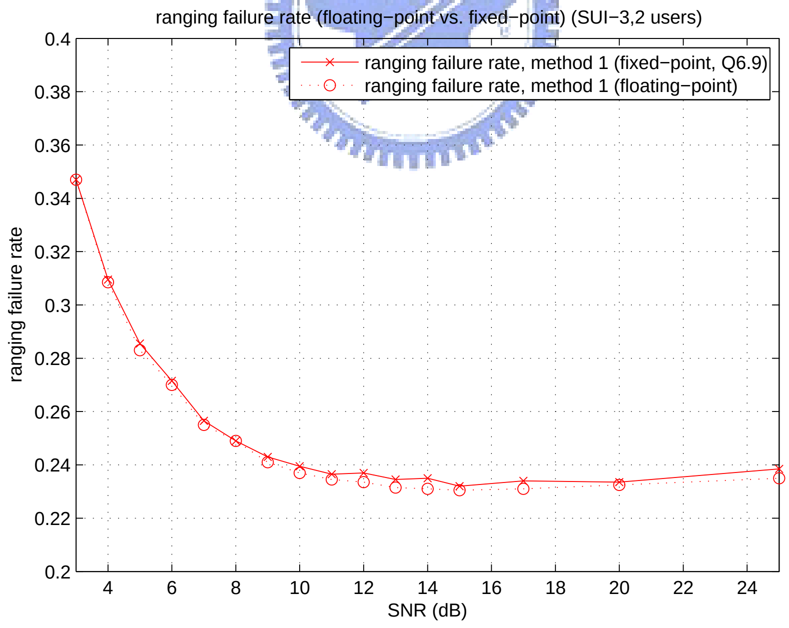


Figure 5.16: Ranging failure rate of method 1 in SUI-3 channel with two ranging users.

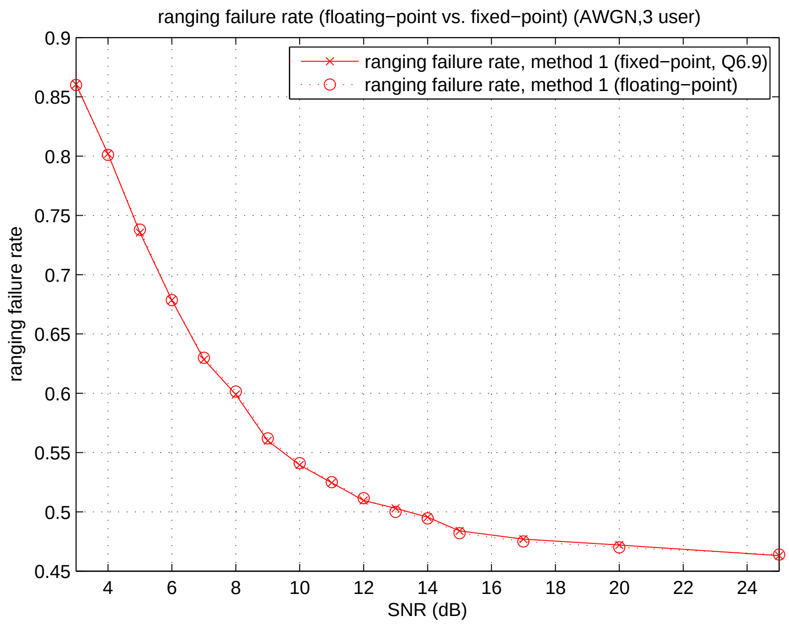


Figure 5.17: Ranging failure rate of method 1 in SUI-3 channel with three ranging users.

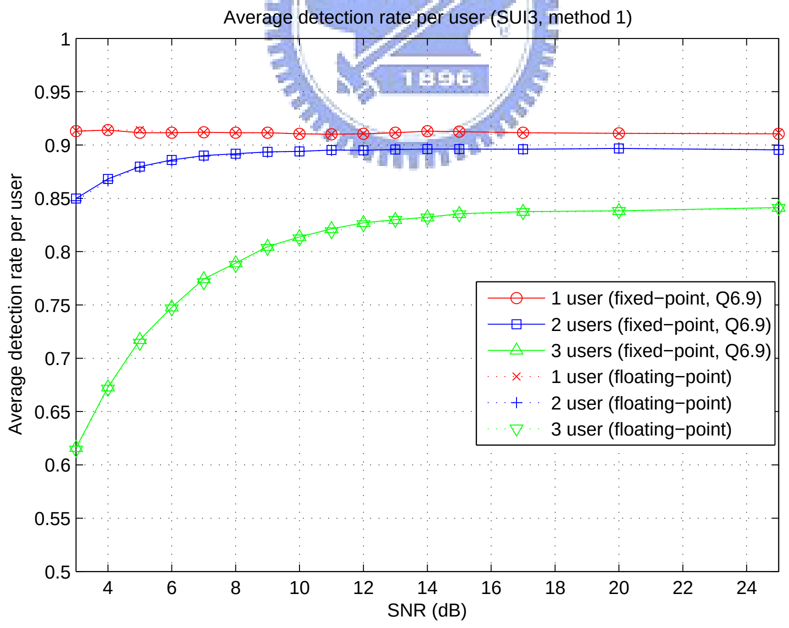


Figure 5.18: Average success rate of each user with method 1 in SUI-3 channel.

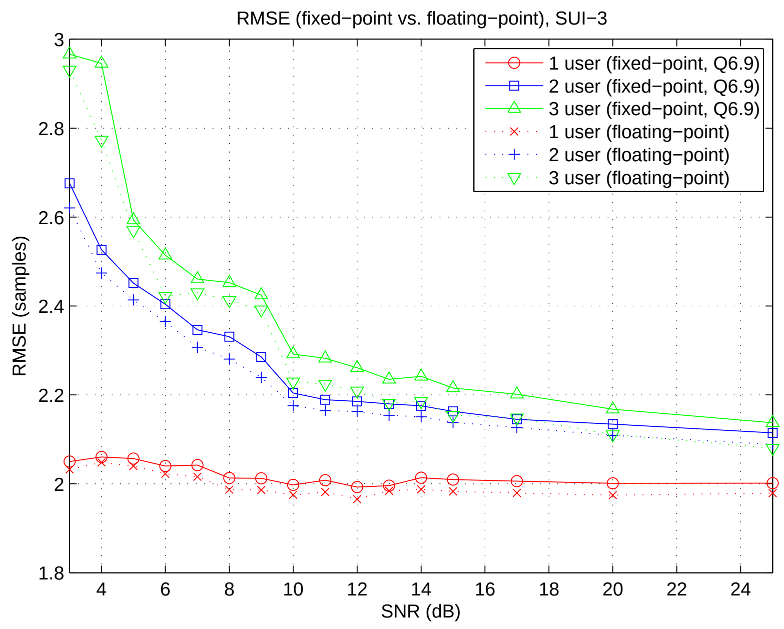


Figure 5.19: RMSE of timing offset estimation in SUI-3 channel.

5.3 DSP Optimization Results

In our system, the clock frequency of DSP is 1 GHz and one frame duration is 5 ms with FFT size equal to 1024 and bandwidth equal to 10 MHz. Therefore, the available execution clock cycles are 5,000,000 per UL subframe.

Table 5.2 shows the number of clock cycles for each function used in the ranging process, where "load" gives the fraction of a DSP's real-time computing power. Note that in programming terms, an "optimized" program does not mean that a program has been made ultimately efficient without any possibility of further improvement. It merely means that suitable programming techniques have been used in writing the program to make it reasonably efficient compared to one without using such techniques. For comparison, we show clock cycles for both detection method 1 and method 2 here.

Note that method 1 and method 2 only differ in the detection and estimation function.

Table 5.2: DSP Optimization Results

	Function	Avg. Clock Cycles		DSP Computational Load (1 UL subframe / 5ms)	
		method 1	method 2	method 1	method 2
Ranging Signal Generator	PRBS generator BPSK modulation	2676			
	Framing	3547			
Total	Rng_Tx	6357		0.00127	
Ranging Signal Receiver	PRBS generator BPSK modulation	2882			
	Multiplication	4836			
	IFFT	12190			
	Norm	1241			
	Detection and estimation	16459	12206		
Total	Rng_Rx	43315	38691	0.00866	0.00774

Since the operation in the ranging signal generator is simple, it needs only 0.00127 DSP chips' processing power to generate one ranging signal.

The task in the ranging signal receiver is more complex. The total requirement for ranging signal reception using method 1 is 0.00866 DSP chips' processing power, while that for reception using method 2 is 0.00774 DSP chips' processing power. Note that this is required power to process one possible ranging code. Since we need to cope with all possible M codes, the processing power for overall ranging signal reception are $0.00866 \times M$ and $0.00774 \times M$ respectively for method 1 and method 2.

The processing power for method 1 is 11.95% more than that for method 2. However, we have observed that method 1 significantly outperforms method 2 in performance. Therefore, method 1 is a better choice. Note that we only analyze the performance of method 2 because the analysis of method 1 is much more complex.

Chapter 6

Conclusion and Future Work

6.1 Conclusion

In this thesis, we first presented the ranging techniques for IEEE 802.16e OFDMA PHY, analyzed and verified them by floating-point simulation. Second, we modified them to fixed-point version and compare the difference of performance between floating-point and fixed-point versions. Finally, we implemented them on TI's digital signal processor.

We mainly discussed the periodic ranging process and presented several analysis and simulations. We employ the frequency domain reception algorithm because it is less complex. We find that the frequency offset which was within the range of [-0.1,0.1] did not cause an effect on the performance in evidence. Thus, we ignore the estimation of frequency offset and focus on the tasks of ranging code detection and timing offset estimation. We provide some simple analysis of our algorithm. The nonzero cross-correlation values of ranging codes had some effect on the analysis and we took them into account. Then, we considered two code detection methods [6], did some floating-point simulation and concluded that method-1 is a better choice. In addition, we discussed the costs of missed detection and false alarm and used them to determine the detection thresholds. Under our algorithm and detection thresholds, the average required numbers of ranging retransmission are 0.0488 and 0.4002 for AWGN channel with 3 ranging users and SNR=3 dB and SUI-3 channel with 3 ranging users and SNR=5 dB, respectively. Each retransmission causes increased latency which is 5 ms in our system.

Finally, we modified the program to fixed-point version. Simulation results showed that all curves of fixed-point simulations are very close to that of floating-point simulations. Then we employed some optimization techniques to accelerate functions of ranging as fast as we can. Finally, some clock cycles simulation results were provided to show that the ranging task can achieve real-time requirements.

6.2 Future Work

There are several possible extension for our study:

- In this thesis, we only discussed periodic ranging. It is possible to includes bandwidth request and handover ranging into our study.
- We only provided analysis of successful detection rate of the algorithm. Some analysis of the performance of timing offset estimation can also be considered.
- The analysis were done for single-user case. More complex analysis in multi-user case can be further considered.
- Analysis of detection method 1.
- Design of better ranging methods.
- Try to develop frequency offset estimation algorithms. It may improve the ranging performance although the improvement is small.

Bibliography

[1] IEEE Std 802.16-2004, *IEEE Standard for Local and Metropolitan Area Networks — Part 16: Air Interface for Fixed Broadband Wireless Access Systems.* New York: IEEE, June. 2004.

[2] IEEE Std 802.16e-2005, *IEEE Standard for Local and Metropolitan Area Networks — Part 16: Air Interface for Fixed Broadband Wireless Access Systems. Amendment 2: Physical and Medium Access Control Layers for Combined Fixed and Mobile Operation in Licensed Bands and Corrigendum 1.* New York: IEEE, Feb. 2006.

[3] OFDMA Introduction on Wikepedia: http://en.wikipedia.org/ofdma

[4] Yue Zhou, Zhaoyang Zhang, and Xiangwei Zhou, "OFDMA initial ranging for IEEE 802.16e based on time-domain and frequency-domain approaches," in *Int. Conf. Commun. Technology,* pp.1–5, Nov. 2006.

[5] Soon-Seng Teo, "IEEE 802.16e OFDMA TDD ranging process and uplink transceiver integration on DSP platform with real-time operating system," M.S. thesis, Dept. of Electronics Eng., National Chiao Tung Univer- sity, Hsinchu, Taiwan, R.O.C., Jan 2008.

[6] Jae-Hyok Lee, Evgeny Gontcharov, Jae-Ho Jeon, and Seung-Joo Maeng, "Apparatus and method for detecting user in a communication system," United States patent application, pub. no. US2007/0002959 A1, Jan. 4, 2007.

[7] D, H, Lee and Hiroyuki Morikawa, “Analysis of ranging process in IEEE 802.16e wireless access systems,” in *Proceedings of the International Workshop on Mobility Management and Wireless Access,* pp.172–179, Oct. 2006.

[8] Rician Distribution Introduction on Wikepedia: http://en.wikipedia.org/wiki/Rician_distribution

[9] Message on the R-help Mailing List: https://stat.ethz.ch/pipermail/r-help/2003-September/039394.html

[10] Guo-Wei Ji, “Research in synchronization techniques and DSP implemantation for IEEE 802.16e OFDMA,” M.S. thesis, Dept. of Electronics Eng., National Chiao Tung University, Hsinchu, Taiwan, R.O.C., June 2006.

[11] Yao-Chun Liu, “Research in and DSP implemantation of synchronization techniques for IEEE 802.16e,” M.S. thesis, Dept. of Electronics Eng., National Chiao Tung University, Hsinchu, Taiwan, R.O.C., June 2007.

[12] P. Dent, G. E. Bottomley, and T. Croft, “Jakes’ fading model revisited,” *Electron. Lett.*, vol. 29, no. 13, pp. 1162–1163, June 1993.

[13] V. Erceg *et al.*, “Channel models for fixed wireless applications,” IEEE 802.16.3c-01/29r4, July 2001.

[14] Sundance, Sundance.chm, Apr. 2006.

[15] Sundance home page: http://www.sundance.com/default.asp

[16] Texas Instruments, *TMS320C6000 CPU and Instruction Set Reference Guide.* Lit. no. SPRU189F, Oct. 2000.

[17] Texas Instruments, *TMS320C6414T, TMS320C6415T, TMS320C6416T Fixed-Point Digital Signal Processors.* Lit. no. SPRS226A, Mar. 2004.

[18] Texas Instruments, *TMS320C6000 CPU and Instruction Set.* Lit. number SPRU189F, Oct. 2000.

[19] Texas Instruments, *TMS320C6000 Code Composer Studio Tutorial.* Lit. no. SPRU301CI, Feb. 2000.

[20] Texas Instruments, *TMS320C6000 Programmer's Guide.* Lit. no. SPRU198I, Mar. 2006.

[21] Texas Instrument, *TMS320C6000 Optimizing Compiler User Guide.* Lit. no. SPRU187K, Oct. 2002.

[22] Texas Instruments, *TMS320C6000 CPU and Instruction Set Reference Guide.* Lit. no. SPRU189F, Oct. 2000.

[23] Carry Look-ahead Adder Introduction on Wikepedia: http://en.wikipedia.org/wiki/Carry_look-ahead_adder

[24] K. K. Parhi, *VLSI Digital Signal Processing Systems: Design and Implementation,* Wiley, New York, 1999.

作者簡歷

姓名：蔡昀澤 (Yun-Tze Tsai)

生日：1984 年 10 月 23 日

出生地：高雄市

學歷：交通大學電子工程系學士(2002.9~2006.6)

交通大學電子研究所碩士(2006.9~2008.6)

研究領域：通訊系統及數位訊號處理

論文題目：IEEE 802.16e OFDMA 實體層測距技術與數位訊號處理器實現之探討

(Study in and Ranging Techniques and Associated Digital Signal Processor Implementation for IEEE 802.16e OFDMA PHY)