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 頻寬檢測與資訊單元處理技術之研究

Study in Bandwidth Detection and Information Element Handling for IEEE 802.16e OFDMA

研 究 生: 蔡婉清

指導教授: 林大衛 博士

中 華 民 國 九 十 七 年 七 月

IEEE 802.16e OFDMA 頻寬檢測與資訊單元處理技術之研究

Study in Bandwidth Detection and Information Element Handling for IEEE 802.16e OFDMA

研 究 生: 蔡婉清　　Student: Wan-Ching 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

July 2008

Hsinchu, Taiwan, Republic of China

中 華 民 國 九 十 七 年 七 月

IEEE 802.16e OFDMA 頻寬檢測與資訊單元處理技術之研究

研究生：蔡婉清　　　　　　　　　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，頻寬檢測和循環自首(cyclic prefix)長度檢測的問題、演算法、以及資訊單元處理實做方面的議題。

當一個行動電話在一開始要進入網路的時候，可能會需要檢測所在的頻寬。在收到的訊號中，可能會包含數個系統而每個系統的頻寬和位置都有待檢測。此篇論文中，我們考慮接收 20 MHz 頻寬的訊號，找出所有這訊號中存在的系統所使用的中心頻率和頻寬。由於在 IEEE 802.16e 中有規範關於頻寬大小與中心頻率放置的限制，我們可以根據規範列出所有可能存在這 20 MHz 的頻寬分佈 (每個系統的頻寬大小與中心頻率)。在所有可能的頻寬分佈的功率頻譜已知的情況下，我們可以找到一個功率頻譜最接近所接收到的訊號的功率頻譜，而這個功率頻譜所屬的頻率分佈就是我們的檢測結果。在這篇論文中，我們提出兩種維度分別去計算功率頻譜的距離，並且分析比較此兩種維度所檢測出的效果和計算複雜度。

完成頻寬檢測後，此行動電話必須檢測它要進入的系統的循環字首的長度。循環字首共有四種長度。考慮到計算複雜度的問題，我們使用循環字首的相關性，找出這四種長度中相關性最高的長度就是我們的檢測結果。

行動電話在完成進入網路的程序並成功進入後，必須在每個下行訊框讀取資料單元，因此我們建議一種低運算複雜度的資訊單元處理的方法並且將他實現與最佳化在 DSP 上。

Study in Bandwidth Detection and Information Element Handling for IEEE 802.16e OFDMA

Student: Wan-Ching Tsai Advisor: Dr. David W. Lin

Department of Electronics Engineering

Institute of Electronics

National Chiao Tung University

Abstract

This thesis introduces the bandwidth detection, cyclic prefix (CP) detection, and information element (IE) handling problems and algorithms, and implementation the issues of IEEE 802.16e OFDMA system.

In DL, the (mobile station) MS receiver may need to detect the bandwidth upon its initial entrance to the network. There are multi types of bandwidth we received from several systems. In this thesis, we consider detecting the bandwidth and the center frequencies of all S-OFDMA systems in the 20 MHz bandwidth that we received. Because there are some rules specify the bandwidth and the permutation of center frequency in IEEE 802.16e [2], we can list all possible bandwidth distributions (bandwidth and center frequency of each system) that fall in 20 MHz. Since the power spectrums of all types of bandwidth distribution are known, we can find the type which has the least distance to the received signal as the decision bandwidth distribution. In this thesis, we propose two types of dimensions to calculate distance separately and analyze the performances and the computational complexities of the two types of dimensions.

After detecting bandwidth, the MS have to detect the CP used in the system. There are four types of CP. Considering the computational complexity, we use the CP complex conjugate correlation method to test which CP result in the largest correlation and that is the determined CP length.

As MS complete the network entry procedure and succeed entrance, it must to read IE every downlink frame, so we proposed an information element handling method that has low computational complexity and implemented and optimized it on DSP.

誌謝

這篇論文能夠順利完成，首先要感謝的人是我的指導教授林大衛老師，在兩年的研究所生涯當中，由於老師的細心指導及在專業領域的博學精深，使得我學習到不少研究的精神與方法，也由於老師親切樂觀的態度，使我在研究的過程中也能保持輕鬆的心情去面對與嘗試解決各種錯誤，享受研究的樂趣。

此外，感謝通訊電子與訊號處理實驗室所有的成員，包含各位師長、同學、學長姐與學弟妹們。特別感謝洪崑健學長、吳俊榮學長總是主動熱心且不保留的給予我在研究過程上許多的指導與建議，也感謝 94 級的依翎學姐不論是在研究還是生活上都給我許多照顧，還有昀澤、光中、佳楓、威年、尚諭、紹唐等同學，因為能和你們共同討論、分享求學的經驗及一路上的相互扶持、，讓這兩年的研究生涯充滿歡樂與回憶。

最後，我要感謝我的家人們，感謝他們一直都在背後支持我，在求學過程中總是不斷的鼓勵我，是我精神上最大的支柱。

在此，將此篇論文獻給所有幫助過我，陪伴我走過這一段歲月的師長，同學，朋友與家人。

蔡婉清

民國九十七年七月 於新竹

Contents

List of Figures

List of Tables

Chapter 1

Introduction

The IEEE 802.16e, of which a subset is commonly known as Mobile WiMAX (Worldwide Interoperability of Microwave Access), is a broadband wireless access (BWA) system that has drawn much attention these days. IEEE 802.16e was originally suggested as an enhancement version of IEEE Std.802.16-2004 to provide mobile station (MS) with mobility at vehicular speed. Therefore, it specifies BWA systems for both fixed and mobile MS simultaneously [1], [2].

One of the most promising modes in the IEEE 802.16e standard is the orthogonal frequency division multiple access (OFDMA) mode, which is generally accepted as a performance efficient multiple access scheme. The Mobile WiMAX system also utilizes the bandwidth scalability, where the FFT size typically increases with the bandwidth. In this thesis, we consider the IEEE 802.16e WirelessMAN OFDMA system with a time-division duplex (TDD) mode, where downlink (DL) and uplink (UL) transmissions are time multiplexed in each TDD frame.

Our study can be divided into two parts. The first part is the bandwidth and cyclic prefix (CP) detection techniques for IEEE 802.16e OFDMA. In a general OFDMA system, when performing initial network entry, the MS may not know the FFT size and the channel bandwidth employed by the BS, so it may need to detect them. After detecting the bandwidth and FFT size, MS should search all possible values of CP until it finds the CP being used by the BS. The literatures about bandwidth detection almost use the transmission time to determine a measured bandwidth (ex. [3], [4]), it is not suitable for the initial network entry. In the present study we consider recognizing the bandwidth structure, that is, appropriate for cognitive radio. We assume the FFT size is proportional to the channel bandwidth, so we just pay attention to bandwidth and CP detection.

The second part is the information element (IE) handling in the digital signal processor (DSP) implementation of an overall system. The principal IE we need to handle is FCH and burst profile in the DL-MAP, that contains the information we have to interpret in order to read the received data. We implement an IE interpreter on Texas Instrument (TI)'s DSP. Moreover, we employ various optimization techniques to accelerate the execution speed of the implemented DSP programs.

This thesis is organized as follows.

1. First introduce the IEEE 802.16e WirelessMAN OFDMA standard in chapter 2.
2. Introduce the DSP implementation platform in chapter 3.
3. Analyze the bandwidth and CP detection problems and present some solutions in chapter 4.
4. Expound IE handling problem and discusses the DSP optimization methods and presents the optimization results in chapter 5.
5. Finally, the conclusion is given in chapter 6, where we also point out some potential future work.

Chapter 2

Overview of the IEEE 802.16e OFDMA Standard

In this chapter, we first introduce some basic concepts regarding orthogonal frequency-division multiplexing (OFDM) and OFDMA. Then we give an overview of the IEEE 802.16e OFDMA standard. For the sake of simplicity, we only introduce the specifications that are useful in our study. Other specifications like channel coding, channel estimation, transmit diversity, etc., are not our concern and are ignored in this introduction. For more details we refer the reader to [1] and [2], from which we take much of the material in this chapter.

2.1 Introduction to OFDM [5]

OFDM is a special case of the multicarrier transmission technique, where a single datastream is transmitted over a number of lower rate subcarriers. It divides the available spectrum into narrower subcarrier bands, and each subcarrier only transmits a portion of the total information.

The orthogonality of OFDM constitutes one major difference from the classical parallel data system, making its use of the available spectrum more efficient. Figure 2.1 shows the difference. As we can see, the subcarriers in an OFDM symbol can be arranged so that the sideband of each subcarrier overlaps but the received symbols still live with-

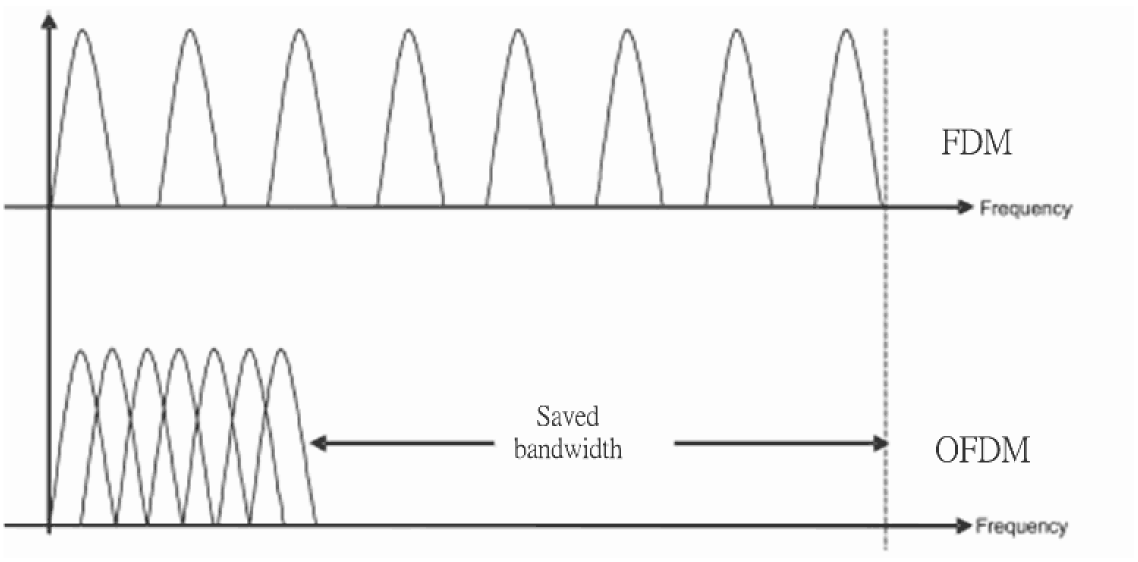


Figure 2.1: Bandwidth efficiency comparison of traditional FDM and OFDM systems (from [5]).

out adjacent-carrier interference. This can be accomplished by using the discrete Fourier transform (DFT) as proposed by Weinstein and Ebert in 1971 [6]. The complexity of DFT, however, was too expensive in the early days. Fortunately, modern advances in very-large-scale integration (VLSI) make it possible to use the fast Fourier transform (FFT) for a more efficient implementation of the DFT. The complexity is reduced from N^2 in DFT to $N \log_2 N$ in FFT.

One of the main reasons to use OFDM is to increase the robustness against frequency selective fading or narrowband interference. An OFDM system may encode data using forward error correction (FEC) coding and distribute them across several subcarriers. If frequency-selective fading causes errors in the reception of few subcarriers, the data bits in those subcarriers are recovered through FEC.

Another reason for choosing OFDM is its ability to handle multipath interference with a small amount of overhead. For a given overall data rate, the increasing number of carriers can reduce the data rate that each individual carriers must convey, and hence lengthen the symbol period. This means that the inter symbol interference (ISI) affects a smaller percentage of each symbol. In order to eliminate the ISI, a guard time (or guard interval) is inserted. The guard time is chosen larger than the expected delay spread, such that multipath components from one symbol cannot interfere with the next symbol. However, if we insert zeros within the guard interval, the orthogonality among subcarriers will no longer exist, which causes serious intercarrier interference (ICI). To preserve the orthogonality among the subcarriers and eliminate ICI, the OFDM symbol should be cyclically extended in the guard time rather than just extended with zero. Figure 2.2 shows how to add cyclic prefix in front of an OFDM symbol. Hence if the maximum multipath delay is smaller than the guard time, there will not be ISI or ICI.

Finally, the advantages and disadvantages are summarized in Table 2.1. The advantages are already discussed above. The disadvantages can be addressed in various ways, but are not the focus of the present study.

2.2 Introduction to OFDMA [7]

OFDMA is a multiple access method based on OFDM signaling that allows simultaneous transmissions to and from multiple users along with the other advantages of OFDM. In OFDM, a channel is divided into carriers which is used by one user at any time. In OFDMA, the carriers are divided into subchannels. Each subchannel has multiple carriers that form one unit in frequency allocation. In this way, the bandwidth can be allocated dynamically to the users according to their needs.

An additional advantage of OFDMA is the following. Due to the large variance in a mobile system's path loss, inter-cell interference is a common issue in mobile wireless systems. An OFDMA system can be designed such that subchannels can be composed

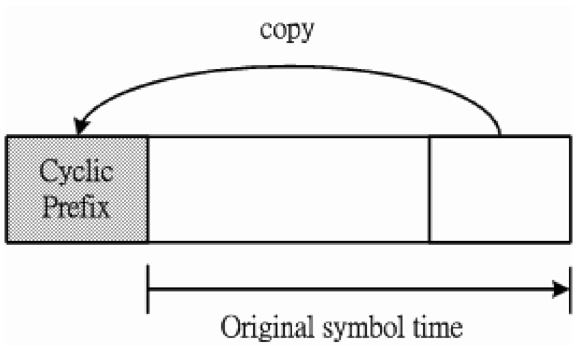


Figure 2.2: The use of cyclic prefix (from [8]).

Table 2.1: OFDM Advantages and Disadvantages [7]

Advantages	Disadvantages
Bandwidth efficiency	Sensitive to frequency offset
Immunity to multipath effect	Sensitive to timing offset
Robust against narrowband interference	Sensitive to phase noise
	Large peak-to-average power ratio

from several distinct permutations of subcarriers. This enables significant reduction in inter-cell interference when the system is not fully loaded, because even on occasions where the same subchannel is used at the same time in two different cells, there is only a partial collision on the active subcarriers. A simple comparison of the subcarrier allocation of OFDM and OFDMA is shown in Fig. 2.3.

In order to support multiple users, the control mechanism becomes more complex. Besides, the OFDMA system has some implementation issues which are more complicated to handle. For example, power control is needed for the uplink to make signals from different users have equal power at the receiver, and all users have to adjust their transmitting time to be aligned.

2.3 Introduction to IEEE 802.16e [2]

The IEEE 802.16 family of standards is officially called WirelessMAN. Part of it is known as WiMAX (Worldwide Interoperability for Microwave Access) by an industry group called the WiMAX Forum. The abjectlies of the Forum are to promote and certify compatibility and interoperability of broadband wireless products.

The first 802.16 standard approved in December 2001 is a standard for point to multipoint 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 line-of-sight (LOS) requirement between transmitters and receivers in the 802.16 standard, the 802.16a standard was approved in 2003 to support nonline-of-sight (NLOS) links, operational in both licensed and unlicensed fre-

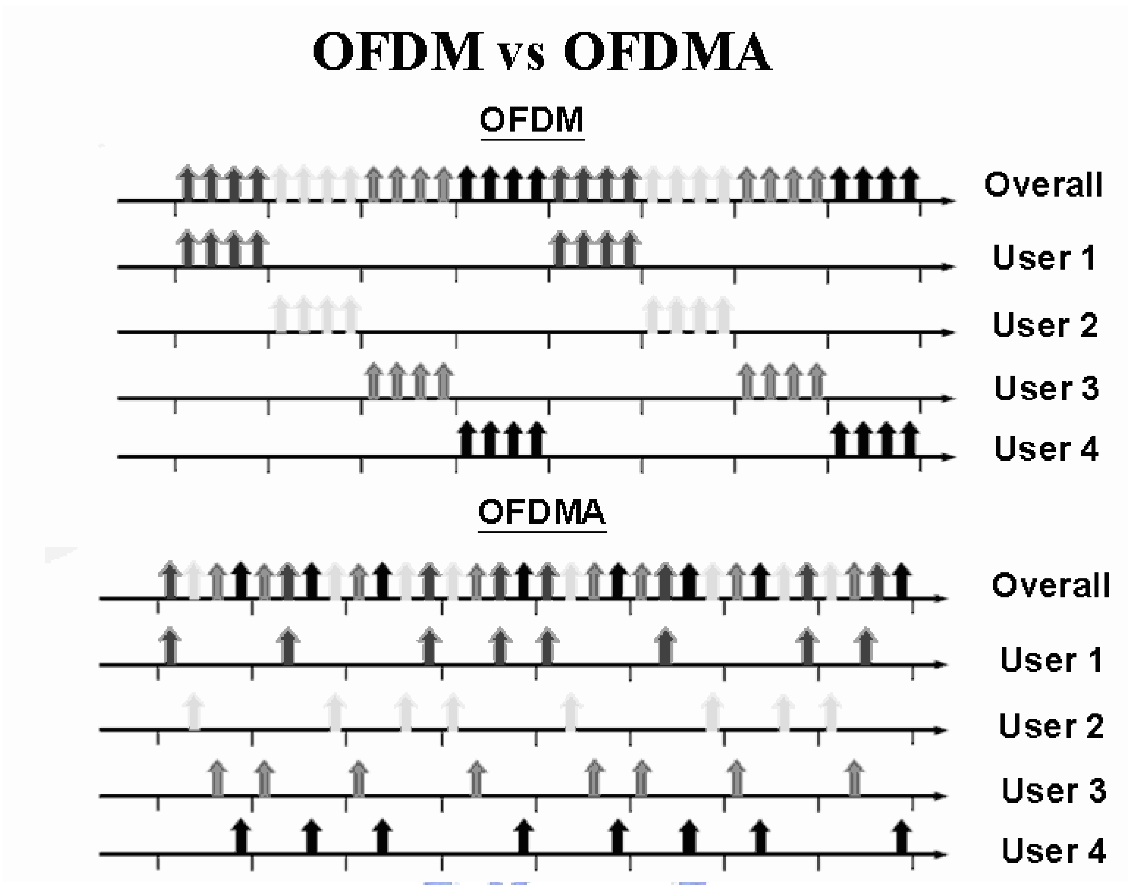


Figure 2.3: Comparison of subcarrier allocatins in OFDM and OFDMA (from [9]).

quency bands from 2 to 11 GHz, and subsequently revised to create the 802.16d standard (now code-named 802.16-2004). 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. The 802.16e task group was subsequently formed with the goal of extending the 802.16-2004 standard to support mobile terminals.

The IEEE 802.16e has been published in Febuary 2006. It specifies four air interfaces: WirelessMAN-SC PHY, WirelessMAN-SCa PHY, WirelessMAN-OFDM PHY, and WirelessMAN-OFDMA PHY. This study is concerned with WirelessMAN-OFDMA PHY in a mobile communication environment.

Some glossary used in the following is as follows. The direction of transmission from the base station (BS) to the subscriber station (SS) is called downlink (DL), and the opposite direction is uplink (UL). The SS is considered synonymous as the mobile station

(MS). It is sometimes termed the user. The BS is a generalized equipment set providing connectivity, management, and control of the SS.

2.3.1 Concept of OFDMA and Definition of Basic Terms

We present some basic concepts and terminology of OFDMA signaling in this subsection. The contents of this subsection have been taken to a large extent from [1] and [2].

The OFDMA PHY mode is based on at least one of the fast fourier transform (FFT) sizes 2048 (backward compatible to IEEE Std 802.16-2004), 1024, 512, and 128 shall be supported. This facilitates support of the various channel bandwidths.

The MS may implement a scanning and search mechanism to detect the DL signal when performing initial network entry, and this may include dynamic detection of the FFT size and the channel bandwidth employed by the BS.

Frequency Domain and Time Domain Descriptions

An OFDMA symbol is made up of subcarriers, the number of which determines the FFT size used. There are several subcarrier types:

- 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.

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 Fig. 2.4.

Three basic types of subchannel organization are defined: partial usage of subchannels (PUSC), full usage of subchannels (FUSC), and adaptive modulation and coding (AMC); among which the PUSC is mandatory and the other two are optional. In PUSC DL, the entire channel bandwidth is divided into three segments to be used separately. The

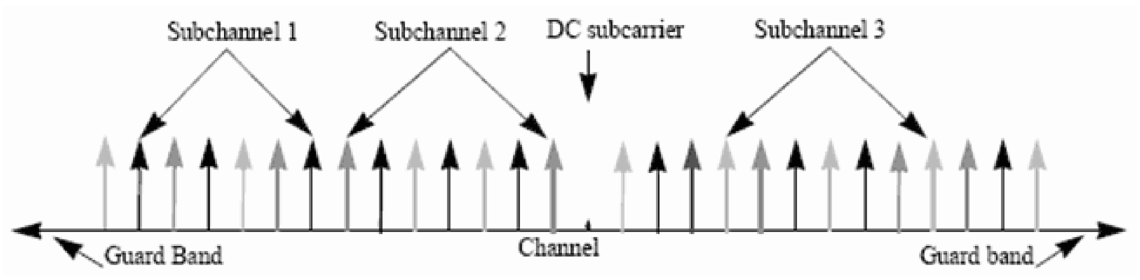


Figure 2.4: OFDMA frequency description (3-channel schematic example, from [1]).

FUSC is employed only in the DL and it uses the full set of available subcarriers so as to maximize the throughput.

Inverse-Fourier-transformation creates the OFDMA waveform; its time duration is referred to as the useful symbol time T_b. A copy of the last T_g of the useful symbol period, termed cyclic prefix (CP), is used to collect multipaths while maintaining the orthogonality of the tones. Figure 2.5 illustrates the structure.

The transmitter energy increases with the length of the guard time while the receiver energy remains the same (the cyclic extension is discarded), so there is a $10\log(1 - T_g/(T_b + T_g))/\log(10)$ dB loss in E_b/N_0. Using a cyclic extension, the samples required for performing the FFT at the receiver can be taken anywhere over the length of the extended symbol. This provides multipath immunity as well as a tolerance for symbol time synchronization errors.

On initialization, an SS should search all possible values of CP until it finds the CP being used by the BS. The SS shall use the same CP on the UL. Once a specific CP duration has been selected by the BS for operation on the DL, it should not be changed. Changing the CP would force all the SSs to resynchronize to the BS.

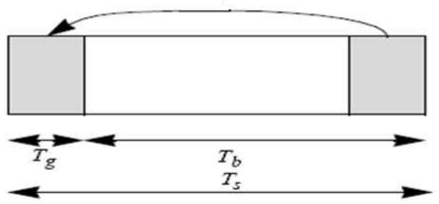


Figure 2.5: OFDMA time description (from [1], Figure 213).

Important Parameters

Four primitive 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.

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}.

Slot and Data Region

The definition of an OFDMA slot depends on the OFDMA symbol structure, which varies for UL and DL, for FUSC and PUSC, and for the distributed subcarrier permutations and the adjacent subcarrier permutation.

- For downlink PUSC using the distributed subcarrier permutation, one slot is one subchannel by two OFDMA symbols.

- For uplink PUSC using either of the distributed subcarrier permutations, one slot is one subchannel by three OFDMA symbols.
- For downlink FUSC and downlink optional FUSC using the distributed subcarrier permutation, one slot is one subchannel by one OFDMA symbol.

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. This two-dimensional allocation may be visualized as a rectangle, such as the 4×3 rectangle shown in Fig. 2.6.

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.

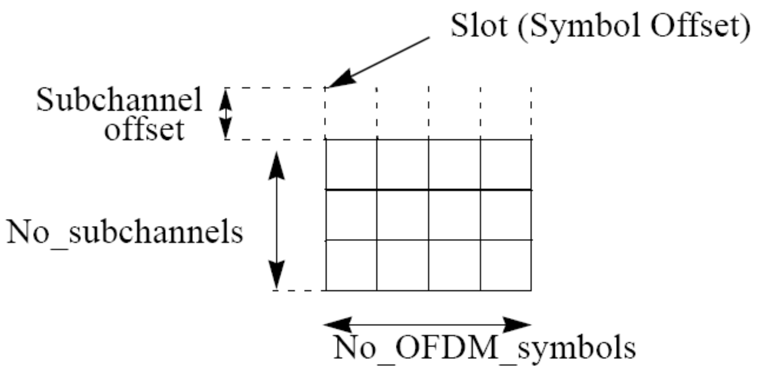


Figure 2.6: Example of the data region which defines the OFDMA allocation (from [1]).

Table 2.2: S-OFDMA Parameters Proposed by WiMAX Forum

Parameters	Values			
System Channel Bandwidth (MHz)	1.25	5	10	20
Sampling Frequency (MHz)	1.4	5.6	11.2	22.4
FFT Size	128	512	1024	2048
Subcarrier Spacing (Δf)	10.94 kHz			
Useful Symbol Time (T_b=1/Δf)	91.4 μsec			
Guard Time (T_g=T_b/8)	11.4 μsec			
OFDMA Symbol Duration (T_s=T_b+T_g)	102.9 μsec			

2.3.2 Scalable OFDMA [10]

One feature of the IEEE 802.16e OFDMA is the selectable FFT size, from 128 to 2048 in multiples of 2, excluding 256 to be used with wirelessMAN-OFDM. This has been termed scalable OFDMA (S-OFDMA). One use of S-OFDMA is that if the channel bandwidths are allocated based on integer power of 2 times a base bandwidth, then one may consider making the FFT size proportional to the allocated bandwidth so that all systems are based on the same subcarrier spacing and the same OFDMA symbol duration, which may simplify system design. For example, Table 2.2 lists some S-OFDMA parameters proposed by the WiMAX Forum [11]. S-OFDMA supports a wide range of bandwidth to flexibly address the need for various spectrum allocation and usage model requirements.

When designing OFDMA wireless systems the optimal choice of the number of subcarriers per channel bandwidth is a tradeoff between protection against multipath, Doppler shift, and design cost/coplexity. Increasing the number of subcarriers leads to better immunity to the ISI caused by multipath; on the other hand it increases the cost and complexity of the system (it leads to higher requirements for signal processing power and power amplifiers with the capability of handling higher peak-to-average power ratios). Having more subcarriers also results in narrower subcarrier spacing and therefore the system becomes more sensitive to Doppler shift and phase noise. Calculations show that the optimum tradeoff for mobile systems is achieved when subcarrier spacing is about 11 kHz

[12] .

2.4 OFDMA Frame Structure [2]

Duplexing Modes

In licensed bands, the duplexing method shall be either frequency-division duplex (FDD) or time-division duplex (TDD). FDD SSs may be half-duplex FDD (H-FDD). In license-exempt bands, the duplexing method shall be TDD.

Point-to-Multipoint (PMP) Frame Structure

When implementing a TDD system, the frame is composed of BS and SS transmissions. Figure 2.7 shows an example. Each frame in the downlink transmission begins with a preamble followed by a DL transmission period and an UL transmission period. In each frame, time gaps, denoted transmit/receive transition gap (TTG) and receive/transmit gap (RTG), are between the downlink and uplink subframes and at the end of each frame, respectively, to allow the BS to turn around.

Subchannel allocation in the downlink may be performed in several ways: PUSC where some of the subchannels are allocated to the transmitter, and FUSC where all subchannels are allocated to the transmitter. The downlink frame shall start in PUSC mode with no transmit diversity. The FCH shall be transmitted using QPSK rate 1/2 with four repetitions using the mandatory coding scheme (i.e., the FCH information will be sent on four subchannels with successive logical subchannel numbers) in a PUSC zone. The FCH contains the DL_Frame_Prefix (see Figure 2.8) which specifies the length of the DL-MAP message that immediately follows the DL_Frame_Prefix and the repetition coding used for the DL-MAP message.

The transitions between modulations and coding take place on slot boundaries in time domain (except in AAS zone, where AAS stands for adaptive antenna system) and on subchannels within an OFDMA symbol in frequency domain. The OFDMA frame may include multiple zones (such as PUSC, FUSC, PUSC with all subchannels, optional FUSC, AMC, TUSC1, and TUSC2, where AMC stands for adaptive modulation and coding and

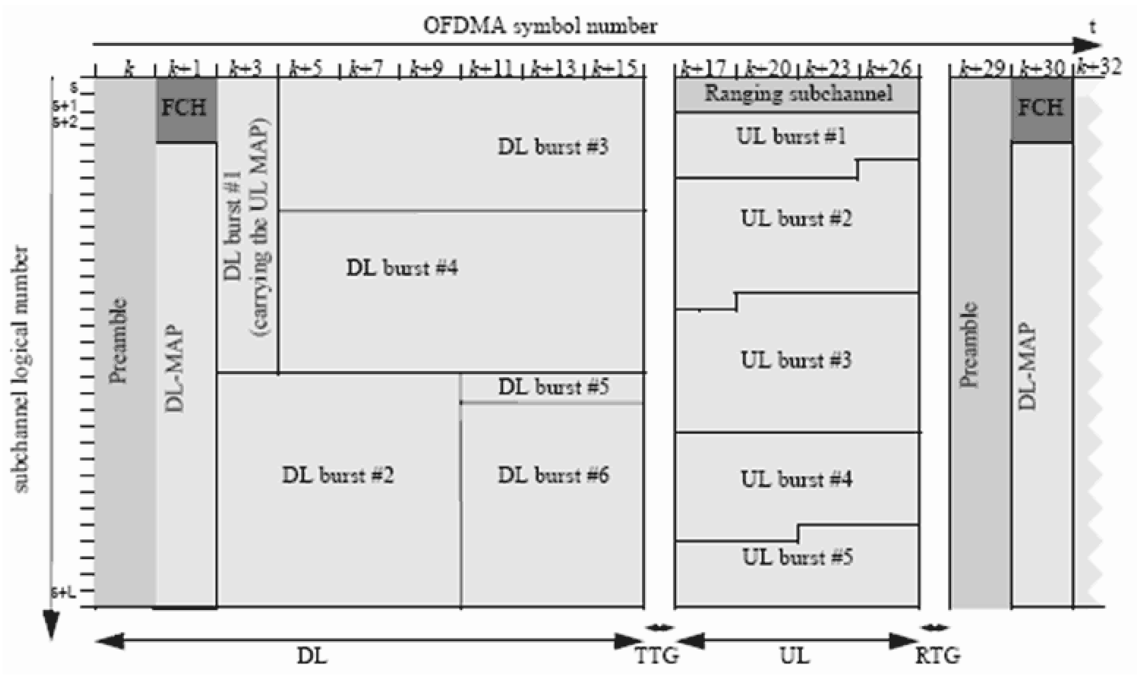


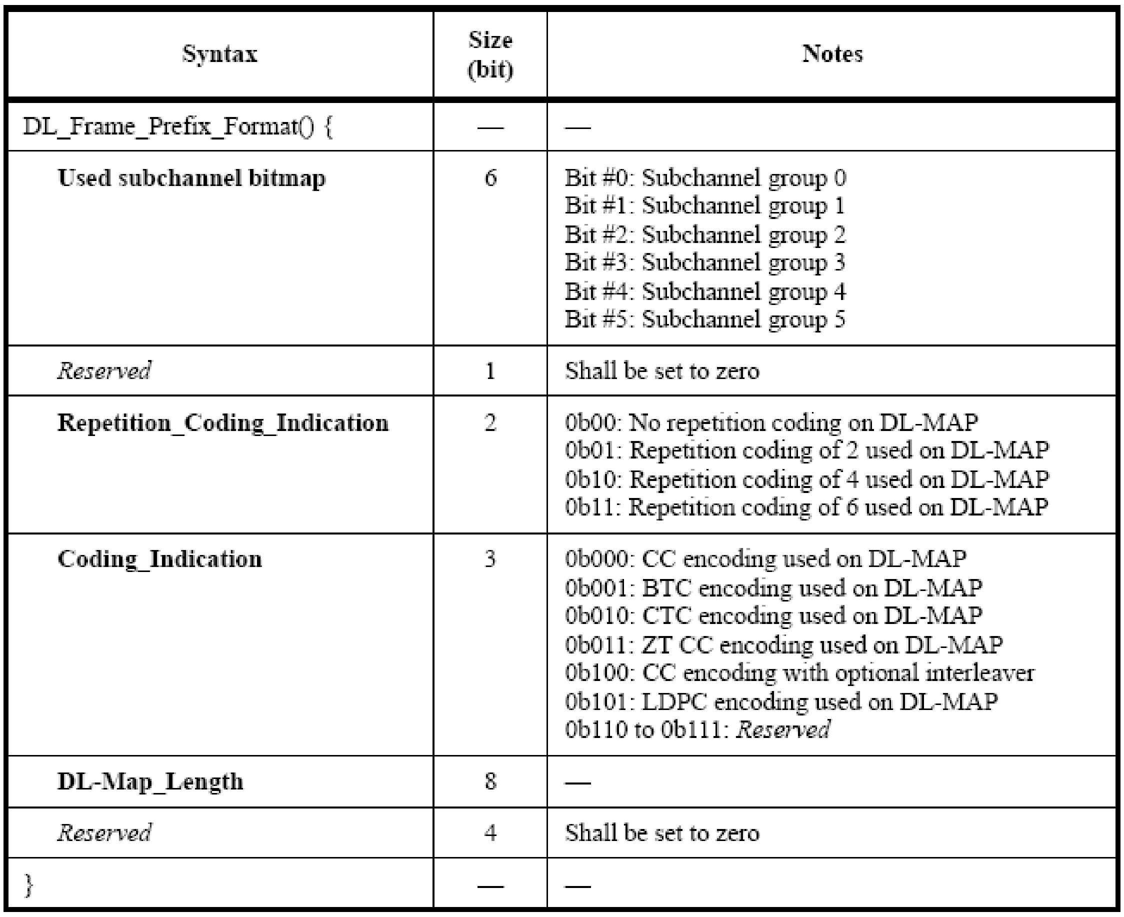
Figure 2.7: Example of an OFDMA frame (with only mandatory zone) in TDD mode (from [2]).

TUSC stands for tile usage of subchannels), the transition between zones is indicated in the DL-MAP. Figure 2.9 depicts an OFDMA frame with multiple zones. The PHY parameters (such as channel state and interference levels) may change from one zone to the next.

The maximum number of downlink zones is 8 in one downlink subframe. For each SS, the maximum number of bursts to decode in one downlink subframe is 64. This includes all bursts without connection identifier (CID) or with CIDs matching the SS's CIDs.

FCH, DL-MAP, and Logical Subchannel Numbering

In PUSC, any segment used shall be allocated at least the same number of subchannels as in subchannel group #0. For FFT sizes other than 128, the first 4 slots in the downlink part of the segment contain the FCH as defined before. These slots contain 48 bits modulated by QPSK with coding rate 1/2 and repetition coding of 4. For FFT-128, the first slot in the downlink part of the segment is dedicated to FCH and repetition is not applied. The basic allocated subchannel sets for segments 0, 1, and 2 are subchannel groups #0, #2, and #4,



Syntax	Size (bit)	Notes
DL_Frame_Prefix_Format() {	—	—
Used subchannel bitmap	6	Bit #0: Subchannel group 0 Bit #1: Subchannel group 1 Bit #2: Subchannel group 2 Bit #3: Subchannel group 3 Bit #4: Subchannel group 4 Bit #5: Subchannel group 5
Reserved	1	Shall be set to zero
Repetition_Coding_Indication	2	0b00: No repetition coding on DL-MAP 0b01: Repetition coding of 2 used on DL-MAP 0b10: Repetition coding of 4 used on DL-MAP 0b11: Repetition coding of 6 used on DL-MAP
Coding_Indication	3	0b000: CC encoding used on DL-MAP 0b001: BTC encoding used on DL-MAP 0b010: CTC encoding used on DL-MAP 0b011: ZT CC encoding used on DL-MAP 0b100: CC encoding with optional interleaver 0b101: LDPC encoding used on DL-MAP 0b110 to 0b111: *Reserved*
DL-Map_Length	8	—
Reserved	4	Shall be set to zero
}	—	—

Figure 2.8: OFDMA DL Frame Prefix format for all FFT sizes except 128 (from [2]).

respectively. Figure 2.10 depicts this structure.

After decoding the DL_Frame_Prefix message within the FCH, the SS has the knowledge of how many and which subchannels are allocated to the PUSC segment. In order to observe the allocation of the subchannels in the downlink as a contiguous allocation block, the subchannels shall be renumbered. The renumbering, for the first PUSC zone, shall start from the FCH subchannels (renumbered to values 0–11), then continue numbering the subchannels in a cyclic manner to the last allocated subchannel and from the first allocated subchannel to the FCH subchannels. Figure 2.11 gives an example of such renumbering for segment 1.

For uplink, in order to observe the allocation of the subchannels as a contiguous allocation block, the subchannels shall be renumbered, and the renumbering shall start from

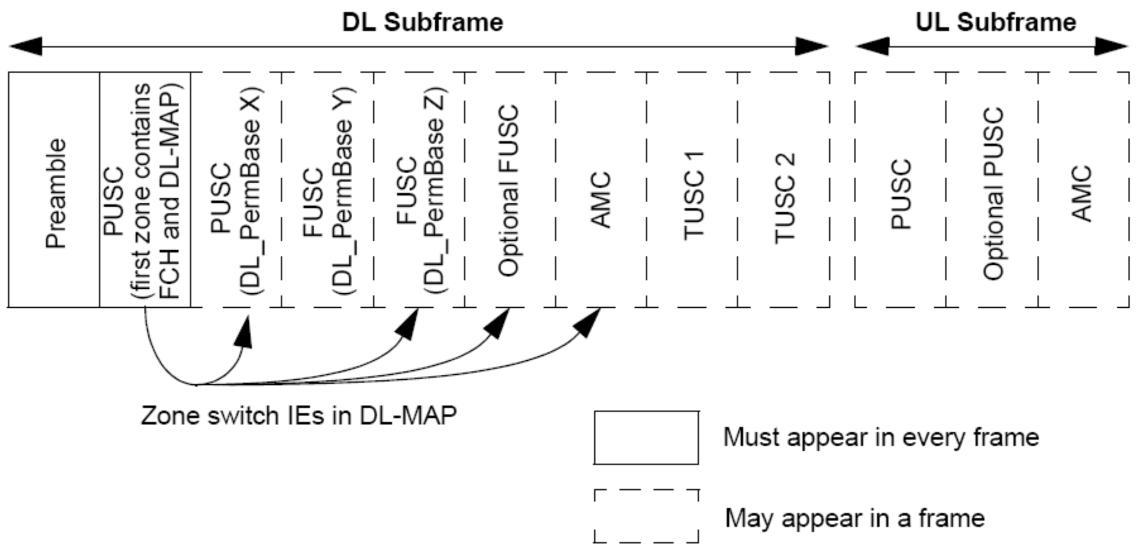


Figure 2.9: Illustration of OFDMA with multiple zones (from [2]).

the lowest numbered allocated subchannel (renumbered to value 0), up to the highest numbered allocated subchannel, skipping nonallocated subchannels.

The DL-MAP of each segment shall be mapped to the slots allocated to the segment in a frequency first order, starting from the slot after the FCH (subchannel 4 in the first symbol, after renumbering), and continuing to the next symbols if necessary. The FCH of segments that have no subchannels allocated (unused segments) will not be transmitted, and the respective slots may be used for transmission of MAP and data of other segments.

2.5 OFDMA Subcarrier Allocation [2]

As mentioned, the OFDMA PHY defines four scalable FFT sizes: 2048, 1024, 512, and 128. For convenience, here we only take the 1024-FFT case for introduction. The subcarriers are divided into three types: null (guard band and DC), pilot, and data. 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. For FUSC and PUSC, in the downlink, the pilot tones are allocated first; what remains are data subcarriers, which are divided into subchannels that are used exclusively for data.

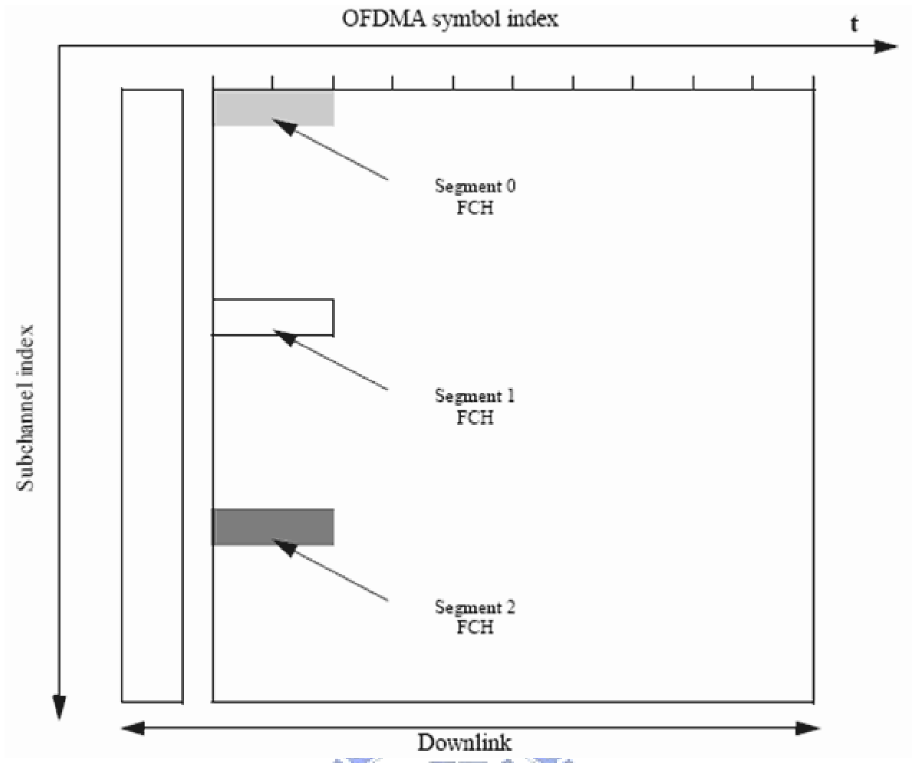


Figure 2.10: FCH subchannel allocation for all 3 segments (from [1]).

Thus, in FUSC, there is one set of common pilot subcarriers, and in PUSC downlink, there is one set of common pilot subcarriers in each major group, but in PUSC uplink, each subchannel contains its own pilot subcarriers.

2.5.1 Downlink

Preamble

Preamble is the first symbol of the downlink transmission. There are three types of preamble carrier-sets, which are defined by allocation of different subcarriers for each one of them. The subcarriers are modulated using a boosted BPSK modulation with a specific pseudo-noise (PN) code. The preamble carrier-sets are defined using

$$PreambleCarrierSet_n = n + 3 \cdot k \tag{2.1}$$

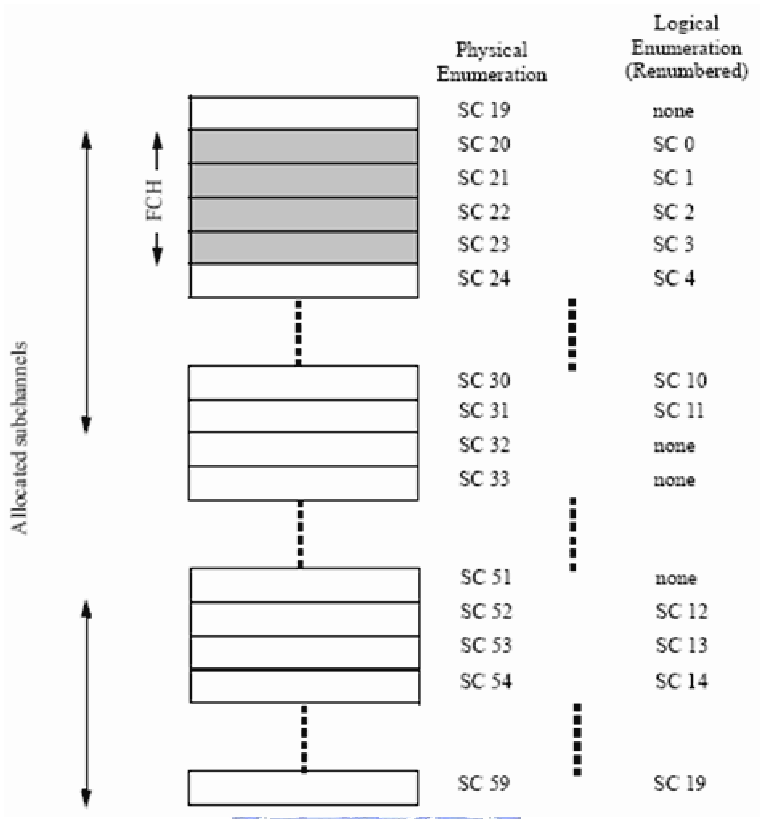


Figure 2.11: Example of DL renumbering the allocated subchannels for segment 1 in PUSC (from [1]).

where:

$PreambleCarrierSet_n$	specifies all subcarriers allocated to the specific preamble,
n	is the number of the preamble carrier-set indexed 0–2,
k	is a running index 0–283.

For the preamble symbol there are 86 guard-band subcarriers on the left side and the right side of the spectrum. Each segment uses a preamble composed of a carrier-set out of the three available carrier-sets in the manner that segment i uses preamble carrier-set i, where $i = 0, 1, 2$. Therefore, each segment eventually modulates each third subcarrier. In the case of segment 0, the DC carrier is zeroed and the corresponding PN number is discarded.

The 114 different PN series modulating the preamble carrier-set are defined in Table 309 of [1] for the 1024-FFT mode. The series modulated depends on the segment used and the IDcell parameter.

Symbol Structure for PUSC

The symbol is first divided into basic clusters and zero carriers are allocated. Pilots and data carriers are allocated within each cluster. Table 310 of [2] summarizes the parameters of the symbol structure of different FFT sizes for PUSC mode. Here we only take the 1024-FFT OFDMA downlink carrier allocation for example, which is summarized in Table 2.3. Fig. 2.12 depicts the DL cluster structure.

Downlink Subchannels Subcarrier Allocation in PUSC

The subcarrier allocation to subchannels is performed using the following procedure:

1. Dividing the subcarriers into the number of clusters ($N_{clusters}$), where the physical clusters contain 14 adjacent subcarriers each (starting from carrier 0). The number of clusters varies with the FFT size.

2. Renumbering the physical clusters into logical clusters using the following formula: $LogicalCluster =$

$$\begin{cases} RenumberingSequence(PhysicalCluster), & \text{first DL zone, or Use All SC indicator} \\ & \text{= 0 in STC\_DL\_Zone\_IE,} \\ RenumberingSequence((PhysicalCluster)+ & \text{otherwise.} \\ \quad 13 \cdot DL\_PermBase) mod N_{clusters}, & \end{cases} \tag{2.2}$$

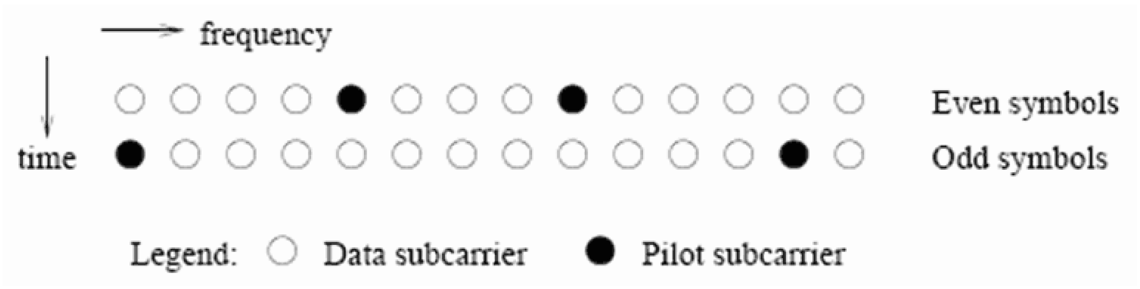


Figure 2.12: DL cluster structure (from [13]).

Table 2.3: 1024-FFT OFDMA DL Carrier Allocation for PUSC (from [2])

Parameter	Value	Comments
Number of DC subcarriers	1	Index 512
Number of guard subcarriers, left	92	
Number of guard subcarriers, right	91	
Number of used subcarriers, N_{used}	841	
Renumbering sequence	6, 48, 37, 21, 31, 40, 42, 56, 32, 47, 30, 33, 54, 18, 10, 15, 50, 51, 58, 46, 23, 45, 16, 57, 39, 35, 7, 55, 25, 59, 53, 11, 22, 38, 28, 19, 17, 3, 27, 12, 29, 26, 5, 41, 49, 44, 9, 8, 1, 13, 36, 14, 43, 2, 20, 24, 52,4, 34, 0	Used to renumber clusters before allocation to subchannels.
Number of subcarriers per cluster	14	
Number of clusters	60	
Number of data subcarriers in each symbol per subchannel	24	
Number of subchannels	30	
Basic permutation sequence 6 (for 6 subchannels)	3, 2, 0, 4, 5, 1	
Basic permutation sequence 4 (for 4 subchannels)	3, 0, 2, 1	

In the first PUSC zone of the downlink (first downlink zone) and in a PUSC zone defined by STC_DL_ZONE_IE() with "use all SC indicator = 0", the default renumbering sequence is used for logical cluster definition. For all other cases DL_PermBase parameter in the STC_DL_Zone_IE() or AAS_DL_IE() shall be used.

3. Allocating logical clusters to groups. The allocation algorithm varies with FFT size. For FFT size = 1024, dividing the clusters into six major groups. Group 0 includes clusters 0–11, group 1 clusters 12–19, group 2 clusters 20–31, group 3 clusters 32–39, group 4 clusters 40–51, and group 5 clusters 52–59. These groups may be allocated to segments; if a segment is used, then at least one group shall be allocated to it. By default group 0 is allocated to sector 0, group 2 to sector 1, and group 4 to sector 2.

4. Allocating subcarriers to subchannels in each major group, which is performed sep-

arately for each OFDMA symbol by first allocating the pilot carriers within each cluster, and then partitioning all remaining data carriers into groups of contiguous subcarriers. Each subchannel consists of one subcarrier from each of these groups. The number of groups is therefore equal to the number of subcarriers per subchannel, and it is denoted $N_{subcarriers}$. The number of the subcarriers in a group is equal to the number of subchannels, and it is denoted $N_{subchannels}$. The number of data subcarriers is thus equal to $N_{subcarriers} \cdot N_{subchannels}$. The parameters vary with FFT sizes. For FFT size = 1024, use the parameters from Table 2.3, with basic permutation sequence 6 for even numbered major groups and basic permutation sequence 4 for odd numbered major groups, to partition the subcarriers into subchannels containing 24 data subcarriers in each symbol. The exact partitioning into subchannels is according to

$$\begin{aligned} subcarrier(k,s) &= N_{subchannels} \cdot n_k \\ &+ \{p_s[n_k \bmod N_{subchannels}] + DL\_PermBase\} \bmod N_{subcahnnels} \end{aligned} \qquad (2.3)$$

where:

$subcarrier(k,s)$	is the subcarrier index of subcarrier k in subchannel s,
s	is the index number of a subchannel, from the set $\{0,\ldots, N_{subcarriers}-1\}$,
n_k	$= (k + 13 \cdot s) \bmod N_{subcarriers}$, where k is the subcarrier-in-subchannel index from the set $\{0,\ldots, N_{subcarriers}-1\}$,
$N_{subchannels}$	is the number of subchannels (for PUSC use number of subchannels in the currently partitioned major group),
$p_s[j]$	is the series obtained by rotating basic permutation sequence cyclically to the left s times,
$DL\_PermBase$	is an integer ranging from 0 to 31, which is set to the preamble IDCell in the first zone and determined by the DL-MAP for other zones.

On initialization, an SS must search for the downlink preamble. After finding the preamble, the user shall know the IDcell used for the data subchannels.

2.5.2 Uplink

The UL follows the DL model, therefore it also supports up to three segments. The UL supports 35 subchannels where each transmission uses 48 data subcarriers as the minimal block of processing. Each new transmission for the uplink commences with the parameters as given in Table 2.4.

Symbol Structure for Subchannel (PUSC)

A slot in the uplink is composed of three OFDMA symbols and one subchannel. Within each slot, there are 48 data subcarriers and 24 fixed-location pilots. The subchannel is constructed from six uplink tiles, each tile has four successive active subcarriers and its configuration is illustrated in Fig. 2.13.

Partitioning of Subcarriers into Subchannels in the Uplink

The usable subcarriers in the allocated frequency band shall be divided into N_{tiles} physical tiles as defined in Fig. 2.13 with parameters from Table 2.4. The allocation of physical tiles to logical tiles in subchannels is performed as

$$Tiles(s,n) = N_{subchannels} \cdot n + \{P_t[(s+n) \bmod N_{subchannels}] + UL\_PermBase\} \bmod N_{subcahnnels} \tag{2.4}$$

Table 2.4: 1024-FFT OFDMA UL Carrier Allocation for PUSC (from [2])

Parameter	Value	Comments
Number of DC subcarriers	1	Index 512
N_{used}	841	
Guard subcarriers: left, right	92,91	
TilePermutation	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	used to allocate tiles to subchannels
$N_{subchannels}$	35	
N_{tiles}	210	
Tile per subchannel	6	

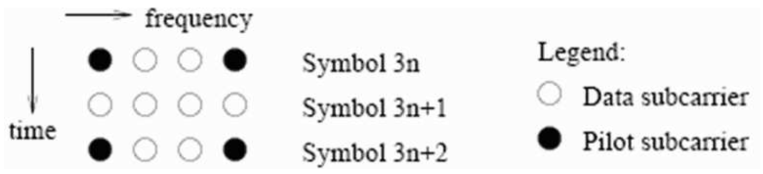


Figure 2.13: Description of an UL tile (from [13]).

where:

$Tiles(s, n)$	is the physical tile index in the FFT with tiles being ordered consecutively from the most negative to the most positive used subcarrier (0 is the starting tile index),
n	is the tile index 0,...,5 in a subchannel,
P_t	is the tile permutation,
$N_{subchannels}$	is the number of subchannels,
s	is the subchannel number in the range $\{0,...,N_{subchannels} - 1\}$,
$UL\_PermBase$	is an integer value in the range 0...69, which is assigned by a management entity.

After mapping the physical tiles in the FFT to logical tiles for each subchannel, the data subcarriers per slot are enumerated by the following process:

1. After allocating the pilot carriers within each tile, indexing of the data subcarriers within each slot is performed starting from the first symbol at the lowest indexed subcarrier of the lowest indexed tile and continuing in an ascending manner through the subcarriers in the same symbol, then going to the next symbol at the lowest indexed data subcarrier, and so on. Data subcarriers are indexed from 0 to 47.

2. The mapping of data onto the subcarriers will follow (2.5), which calculates the subcarrier index (as assigned in item 1) to which the data constellation point is to be mapped:

$$Subcarriers(n, s) = (n + 13 \cdot s) \text{ mod } N_{subcarriers} \tag{2.5}$$

where:

$Subcarriers(n, s)$	is the permutated subcarrier index corresponding to data subcarrier n is subchannel s,
n	is the running index 0,...,47, indicating the data constellation point,
s	is the subchannel number,
$N_{subcarriers}$	is the number of subcarriers per slot.

2.6 Modulation [2]

Subcarrier Randomization

The pseudo random binary sequence (PRBS) generator, as shown in Fig. 2.14, shall be used to produce a sequence w_k. The polynomial for the PRBS generator is $X^{11} + X^9 + 1$. The value of the pilot modulation on subcarrier k shall be derived from w_k.

The initialization vector of the PRBS generator for both uplink and downlink, designated b10..b0, is defined as follows:

b0..b4 = five least significant bits of IDcell as indicated by the frame preamble in the first downlink zone and in the downlink AAS zone with Diversity_Map support, DL_PermBase following STC_DL_Zone_IE() and 5 LSB of DL_PermBase following AAS_DL_IE without Diversity_Map support in the downlink. Five least significant bits of IDcell (as determined by the preamble) in the uplink. For downlink and uplink, b0 is MSB and b4 is LSB, respectively.

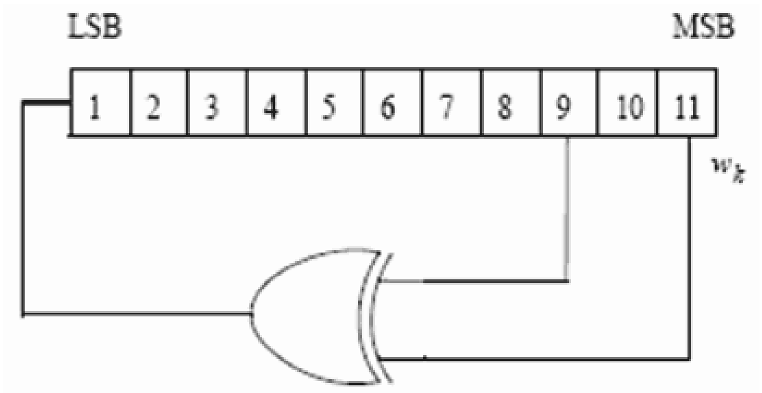


Figure 2.14: PRBS generator for pilot modulation (from [2]).

b5..b6 = set to the segment number + 1 as indicated by the frame preamble in the first downlink zone and in the downlink AAS zone with Diversity_Map support, PRBS_ID as indicated by the STC_DL_Zone_IE or AAS_DL_IE without Diversity_Map support in other downlink zone. 0b11 in the uplink. For downlink and uplink, b5 is MSB and b6 is LSB, respectively.

b7..b10 = 0b1111 (all ones) in the downlink and four LSB of the Frame Number in the uplink. For downlink and uplink, b7 is MSB and b10 is LSB, respectively.

Data Modulation

After the repetition block, the data bits are entered serially to the constellation mapper. Gray-mapped QPSK and 16-QAM shall be supported, whereas the support of 64-QAM is optional.

Pilot Modulation

In all permutations except uplink PUSC and downlink TUSC1, each pilot shall be transmitted with a boosting of 2.5 dB over the average non-boosted power of each data tone. The pilot subcarriers shall be modulated according to

$$\Re\{c_k\} = \tfrac{8}{3}(\tfrac{1}{2} - w_k) \cdot p_k, \; \Im\{c_k\} = 0, \tag{2.6}$$

where p_k is the pilot's polarity for SDMA (spatial division multiple access) allocations in AMC AAS zone, and $p = 1$ otherwise.

Preamble Pilot Modulation

The pilots in the downlink preamble shall be modulated according to

$$\Re\{PreamblePilotModulation\} = 4 \cdot \sqrt{2} \cdot (\frac{1}{2} - w_k), \tag{2.7}$$

$$\Im\{PreamblePilotModulation\} = 0. \tag{2.8}$$

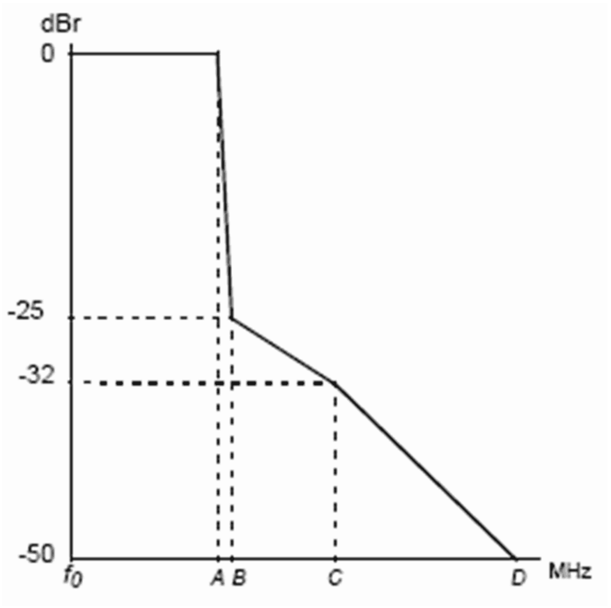


Figure 2.15: Transmit spectral mask for license-exempt operation (from [1]).

Table 2.5: Transmit Spectral Mask for License-Exempt Bands

Bandwidth (MHz)	A	B	C	D
10	9.5	10.9	19.5	29.5
20	4.75	5.45	9.75	14.75

2.7 Transmit Spectral Mask [2]

IEEE 802.16e do not specify the power mask for the licensed bands. Due to requirement of bandwidth-limited transmission, the transmitted spectral density of the transmitted signal shall fall within the spectral mask as shown in Fig. 2.15 and Table 2.5 in license-exempt bands. The measurements shall be made using 100 kHz resolution bandwidth and a 30 kHz video bandwidth. The 0 dBr level is the maximum power allowed by the relevant regulatory body.

Chapter 3

The DSP Implementation Platform

In this chapter, we introduce the DSP platform used in our implementation. We employ the SMT395 DSP module made by Sundance housed on a Sundance PCI-plugin board. The DSP chip on the module is the TMS320C6416T made by Texas Instrument (TI). It also has a Xilinx Virtex II Pro FPGA. We will introduce the DSP card and the DSP chip. In addition, we will also introduce the software development tool, the Code Composer Studio (CCS), and the code development technique.

3.1 The DSP Card [14]

The DSP card used in our implementation is Sundance's SMT395 shown in Fig. 3.1. It houses a 1 GHz 64-bit TMS320C6416T DSP of TI, manufactured on 90 nm technology. The SMT395 is supported by TI's Code Composer Studio and the 3L Diamond real-time operating system (RTOS) to enable multi-DSP systems with minimum programmer effort. It provides a flexible platform for various applications.

Some important features of the SMT395 module are as follows.

- 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 133MHz.
- Two Sundance High-speed Bus (50 MHz, 100 MHz or 200 MHz) ports at 32 bits width.

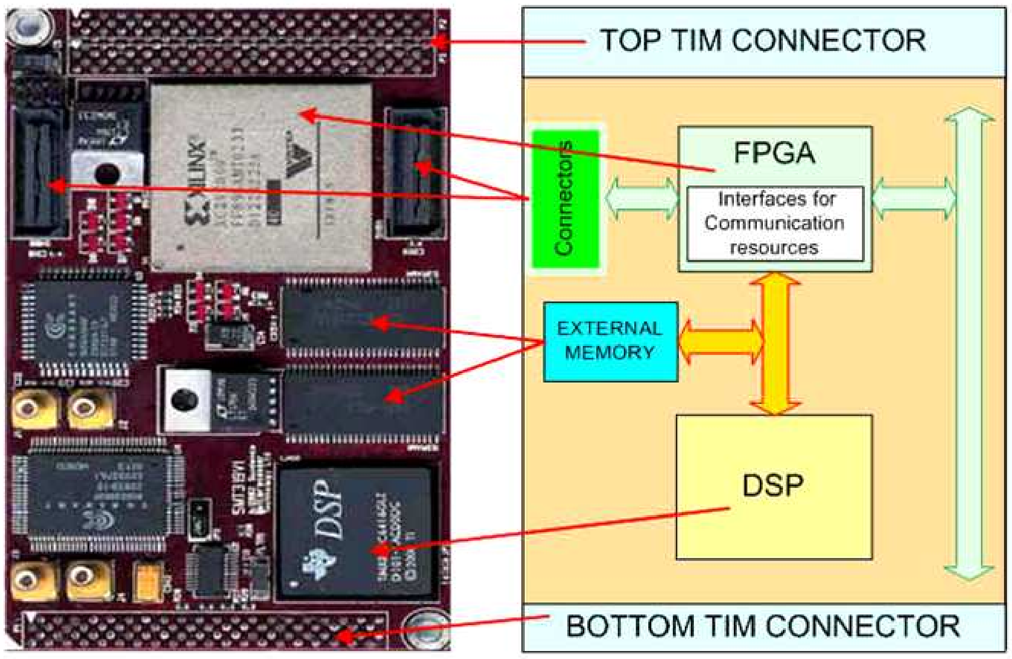


Figure 3.1: Sundance's SMT395 module (from [14])

- Eight 2 Gbit/sec Rocket Serial Links (RSL) for inter-Module communications.
- Six common ports up to 20 MB per second for inter-DSP communication.
- 8 Mbytes flash ROM for configuration and booting.
- JTAG diagnostics port.

3.2 The DSP Chip [15]

The TMS320C6416T DSP is a fixed-point DSP in the TMS320C64x series of the TMS320C6000 DSP platform family. It is based on the advanced VelociTI very-long-instruction-word (VLIW) architecture developed by TI. The functional block and DSP core diagram of the TMS320C64x series is shown in Fig. 3.2.

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

- VLIW CPU with eight functional units, including two multipliers and six arithmetic:
  - Executes up to eight instructions per cycle.
  - Allows designers to develop highly effective RISC-like code for fast development time.
- 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.
- Efficient code execution on independent functional units:
  - Efficient C complier on DSP benchmark suite.
  - Assembly optimizer for fast development and improved parallelization.
- 8/16/32-bit data support, providing efficient memory support for a variety of applications.
- 40-bit arithmetic options add extra precision for vocoders.
- 32x32-bit integer multiply with 32- or 64-bit result.

In the following subsections, three major parts of the TMS320C64x DSP are introduced respectively. They are the central processing unit, memory, and peripherals.

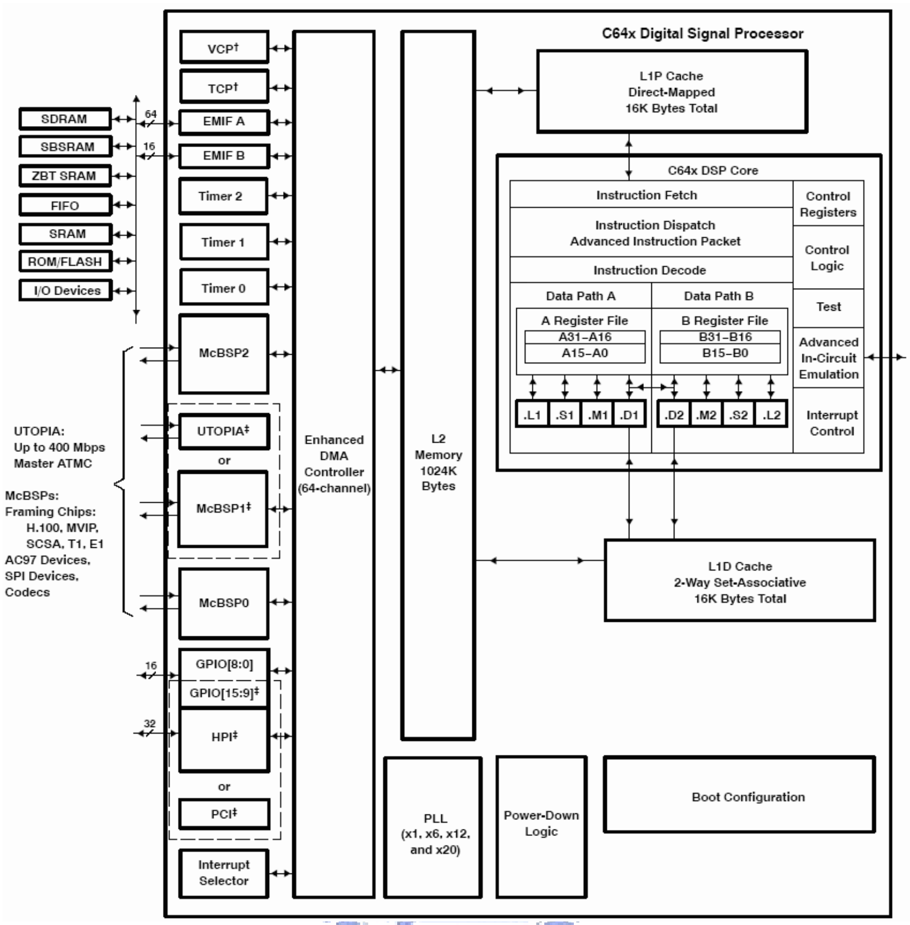


Figure 3.2: Functional block and CPU (DSP core) diagram [16].

3.2.1 Central Processing Unit [15]

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 Fig. 3.2, each of which contains 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 3.1.

Table 3.1: Functional Units and Operations Performed [15]

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

Each register file consists of 32 32-bit registers. Each function unit in the two sets of four functional units reads and writes directly within its own data path. That is, functional units .L1, .S1, .M1, .D1 can only write to register file A. The same holds for register file B. However, two cross-paths (1X and 2X) allow functional units from one data path to access a 32-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 multiply functional units in the same execute packet.

3.2.2 Memory Architecture and Peripherals [15]

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 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 device peripheral. 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.

3.3 TI's Code Development Environment [17]

The Code Composer Studio (CCS) is a key element of the DSP software and development tools from Texas Instruments. The tutorial [18] introduces the key features of CCS and the programmer's guide [19] 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.

3.3.1 Code Composer Studio

The CCS combines the basic code generation tools with a set of debugging and real-time analysis capabilities which supports all phases of the development cycle shown in Fig. 3.3. 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 support.
  - Pre-emptive multi-threading.

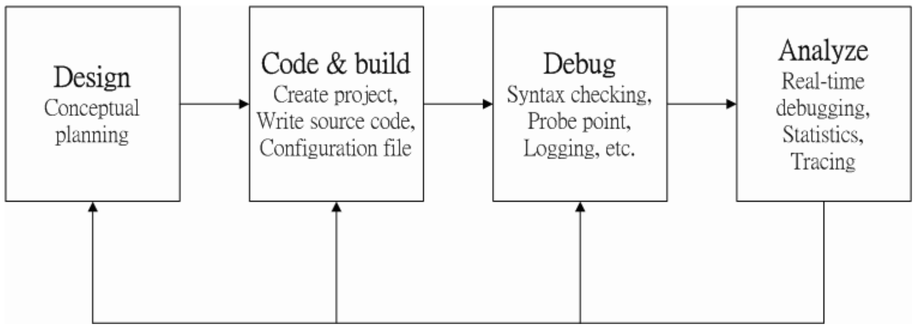

Design
Conceptual planning
Code & build
Create project, Write source code, Configuration file
Debug
Syntax checking, Probe point, Logging, etc.
Analyze
Real-time debugging, Statistics, Tracing

Figure 3.3: Code development cycle [18].

    - Interthread communication.

    - Interupt handing.

- 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 DSP library includes 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. The TMS320C64x digital signal processor library (DSPLIB) provides some routines for:

    - Adaptive filtering.

    - Correlation.

    - FFT.

    - Filtering and convolution.

    - Math.

    - Matrix functions.

    - Miscellaneous.

3.3.2 Code Development Flow [19]

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. Fig. 3.4 illustrates the three phases in the code development flow. Because phase 3 is usually too detailed and time consuming, most of the time we will not go into phase 3 to write linear assembly code unless the software pipelining efficiency is too bad or the resource allocation is too unbalanced.

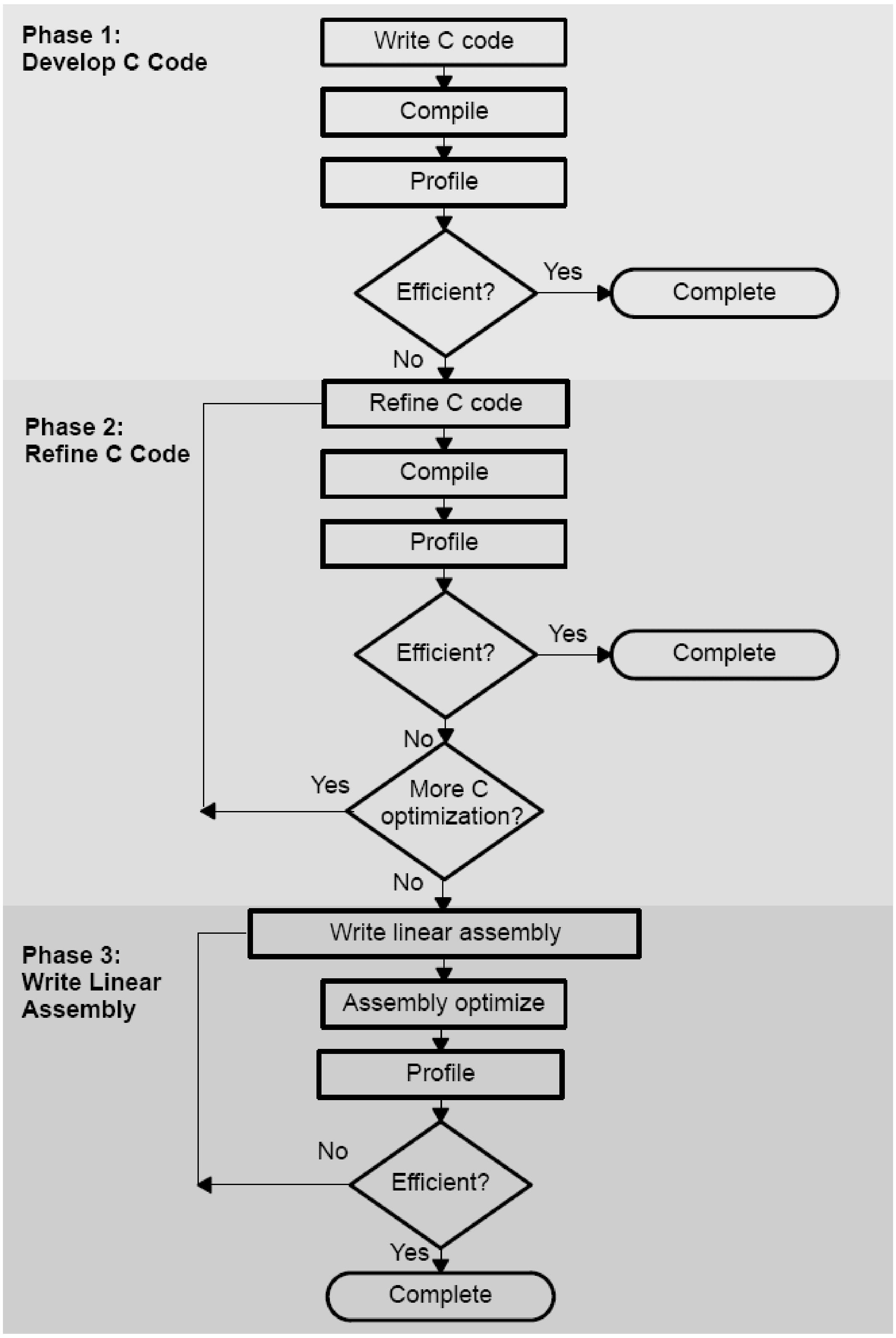


Figure 3.4: Code development flow for C6000 (from [19]).

3.4 Code Optimization on TI DSP Platform

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 Code Composer 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.

3.4.1 Compiler Optimization Options [19]

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 executing performance.

The major compiler options we use are -o3, -k, -pm -op2, -mh<n>, -mw, and -mi.

- -on: The "n" denotes the level of optimization (0, 1, 2, and 3), which controls the type and degree of optimization.
  - -o3: highest level optimization, whose main features are:
    - ∗ Performs software pipelining.

    * Performs loop optimizations, and loop unrolling.
    * Removes all functions that are never called.
    * 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.
- -mh<n>: Allows speculative execution. The appropriate amount of padding, n, must be available in data memory to insure correct execution. This is normally not a problem but must be adhered to.
- -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.

3.4.2 Software Pipelining [20]

Software pipelining is a technique used to schedule instructions from a loop so that multiple iterations of the loop execute in parallel. This is the most important feature we rely on to speed up our system. The compiler always attempts to software-pipeline. Fig. 3.5 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 kernel is known as the pipelined loop prolog, and the area below the kernel the pipelined loop epilog.

But under the conditions listed below, the compiler will not do software pipelining [19]:

- 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.

Usually, we should maximize the number of loops that satisfy the requirements of software pipelining. Software pipelining is a very important technique for optimization; its importance cannot be overemphasized.

3.4.3 Intrinsics [19]

We do not use any intrinsics in our code, but we introduce the concept of this technique here. The C6000 compiler provides intrinsics, which are special functions that map directly to C64x instructions, to optimize C/C++ code quickly. All assembly instructions that are not easily expressed in C/C++ code are supported as intrinsics. A table of TMS320C6000 C/C++ compiler intrinsics can be found in [19].

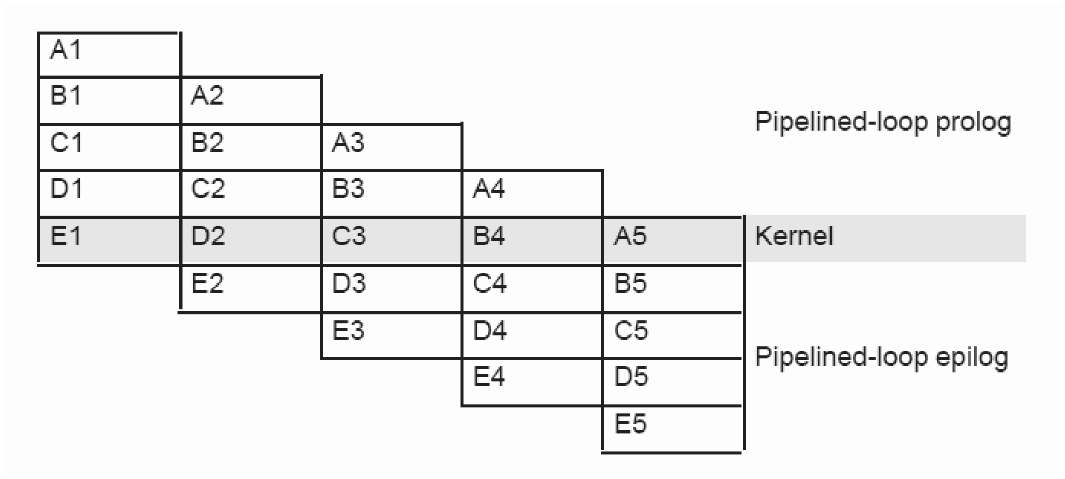


Figure 3.5: Software-pipelined loop (from [15]).

Chapter 4

Channel Bandwidth and Cyclic Prefix Detection for IEEE 802.16e OFDMA

The IEEE 802.16e OFDMA standard is very flexible in choice of bandwidth, FFT size, and CP length [2][p.781]. The FFT size can be 2048, 1024, 512, and 128. The ratio of CP time to useful time can be 1/32, 1/16, 1/8, and 1/4. When performing initial network entry, the MS may implement a scanning and search mechanism to detect the DL signal. This includes dynamic detection of the FFT size and the channel bandwidth employed by the BS. After detecting the bandwidth and the FFT size, MS should search all possible values of CP until it finds the CP being used by the BS. In this chapter, we discuss these detection issues. Note that, in bandwidth detection, we consider recognizing the bandwidth structure. That is appropriated for cognitive radio, not necessarily needed by WiMAX MS.

4.1 System Parameters

The system profile we select is WirelessMAN-OFDMA PHY profile, TDD, and single-input single-output (SISO) operation. The center frequency is 3.5 GHz, and the FFT sizes are 512, 1024, and 2048. The CP lengths are 1/32, 1/16, 1/8, and 1/4 of the FFT size. We consider the PUSC permutation in CP detection simulation and use segment 0 to allocate data subcarriers, but in bandwidth detection simulation and analysis, we consider use of all subchannels with no pilot subcarriers and no preambles for convenience.

The modulation could be QPSK, 16-QAM, or 64-QAM with randomly generated data. Other parameter values are as specified in Table 4.1.

Table 4.1: System Parameters Used in Our Study

Parameters	Values		
System Channel Bandwidth (MHz)	5	10	20
Sampling Frequency (MHz)	5.6	11.2	22.4
FFT Size	512	1024	2048
Subcarrier Spacing (kHz)	10.94		
Useful Symbol Time (μsec)	91.4		

We consider the S-OFDMA system, that is, the FFT size is proportional to the allocated bandwidth. Therefore, we can know the bandwidth employed by BS as the FFT size is detected, and vice versa. So we only look at bandwidth and CP detection techniques in this research.

4.2 Bandwidth Detection

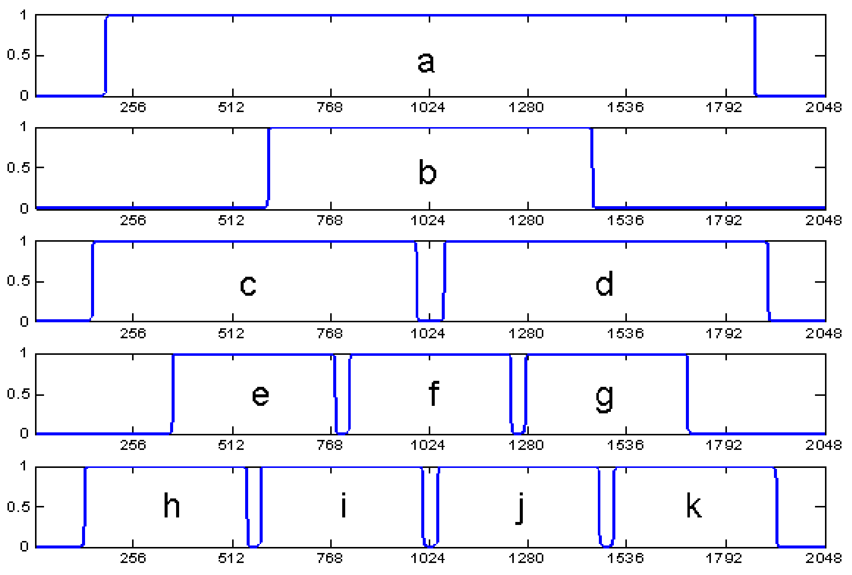
As mentioned above, the MS should detect the channel bandwidth employed by the BS at initial network entry. It may be 5, 10, or 20 MHz. Because of the use of S-OFDMA, the subcarrier spacing is fixed no matter which bandwidth is used. We consider detecting the bandwidth structure in a specific 20 MHz band where multiple systems that use different bandwidths may coexist and the CFO is assumed to be zero. The system profile in IEEE 802.16e [2] specifies some rules for the center frequency of a system of a certain bandwidth. In our research, we assume the center frequencies of systems whose bandwidth is 5 MHz are multiples of 2.5 MHz away from 3.5 GHz, and the center frequencies of systems whose bandwidth is 10 MHz are multiples of 5 MHz away from 3.5 GHz. Figure 4.1 shows all possible bandwidth distributions in the 20 MHz, and detecting which bandwidth distribution is in the 20 MHz is our major work. The following are the steps of the bandwidth detection method that we use:

1. Receive signals with high sampling frequency (22.4 MHz).

2. After each 2048 points, do FFT to convert the signal to the frequency domain.

3. Calculate the power spectrum.

4. Calculate the mean power of 2048 points, and then each point power is divided by



1. a
2. b
3. c
4. d
5. e
6. f
7. g
8. h
9. i
10. j
11. k
12. b+h
13. b+k
14. b+h+k
15. c+d
16. c+g
17. c+j
18. c+k
19. c+j+k
20. d+e
21. d+h
22. d+i
23. d+h+i
24. e+f
25. e+g
26. e+j
27. e+k
28. e+f+g
29. e+f+k
30. e+j+k
31. f+g
32. f+h
33. f+k
34. f+g+h
35. f+h+k
36. g+h
37. g+i
38. g+h+i
39. h+i
40. h+j
41. h+k
42. h+i+j
43. h+i+k
44. h+j+k
45. h+i+j+k
46. i+j
47. i+k
48. i+j+k
49. j+k

Figure 4.1: Possible channel bandwidth distributions with PUSC permutation. All 49 possibilities are listed on the right.

the mean power. This normalizes the final mean power to one.

5. Decide bandwidth distribution.

4.2.1 Interference Analysis

Ideally, we should like the result of step 3 to be something like that depicted in Figure 4.1. However, even though the channel is perfect, we cannot get the perfect power spectrum as shown in Figure 4.1. Interferences comes from two sources. One is the starting point of each 2048 points may not fall in the CP or the first point in the useful time, that is, the 2048 points may come from two adjacent OFDMA symbols. The proportions of the two adjacent symbols depend on the CP length. The shorter the CP length the higher the probability the 2048 points include two symbols. Figure 4.2 shows one example and Figure 4.3 shows the resulting power spectrum. The other reason has to do with the

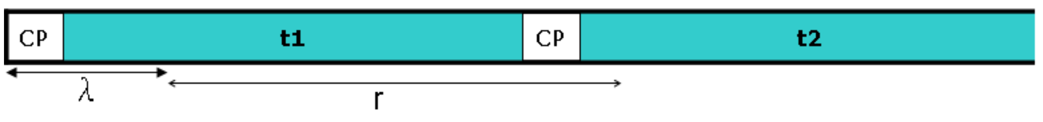


Figure 4.2: The section of samples r (i.e., the 2048 points we get) may straddle over two adjacent symbols.

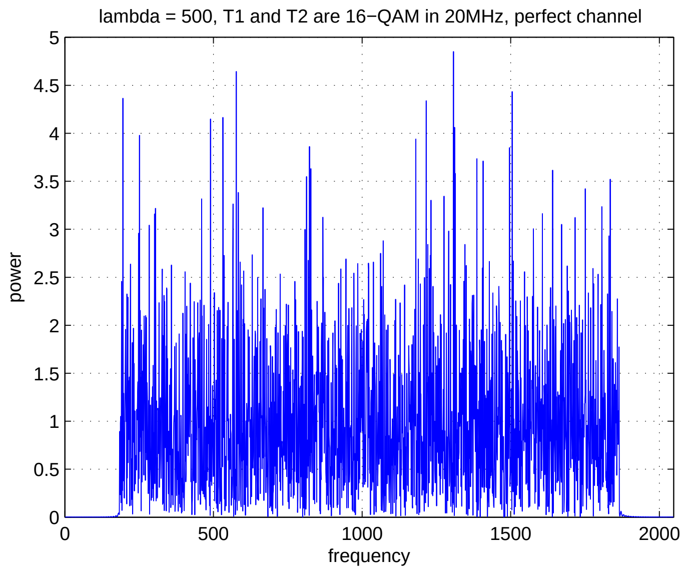


Figure 4.3: Power spectrum of 2048 points of one 20 MHz bandwidth signal with QPSK modulation in perfect channel ($\lambda = 500$).

relation between the center frequency of each system and the sampling rate. The center frequency of each system is a multiple of 2.5 MHz away from 3.5 GHz, but 2.5 MHz is not an integer multiple of the subcarrier spacing (10937.5 Hz). Therefore, when we transform 2048 points of samples of signals, say, having a 3.5 GHz center frequency into the frequency domain, the fractional subcarrier offset with signals that have center frequencies at other than 3.5 GHz will cause interference. Figure 4.4 illustrates this effect.

Consider the case of a single 20 MHz system with center frequency at 3.5 GHz and with 1/8 CP. Let the FFT start point be at the λth point of one OFDMA symbol. That is, the 2048 points we get include parts of two adjacent symbols where the first $(2048 + CPlength) - \lambda$ points are from the first symbol and the last $\lambda - CPlength$ points are from the second symbol. (See Figure 4.2.) Let $r(n)$ denote the signal samples and let the two adjacent symbols (without CP) be denoted $t1(n)$ and $t2(n)$, where $n \in \aleph$ and $0 \leq n \leq 2047$. And let their frequency spectra be denoted $R(k)$, $T1(k)$, and $T2(k)$, respectively, where $k \in \aleph$ and $0 \leq k \leq 2047$. Because the first $(2048 + CPlength) - \lambda$ points of r are the last $(2048 + CPlength) - \lambda$ points of $t1$, the first part of r is $t1$ left circular shifted by $\lambda - CPlength$ points or, equivalently, right circular shifted by

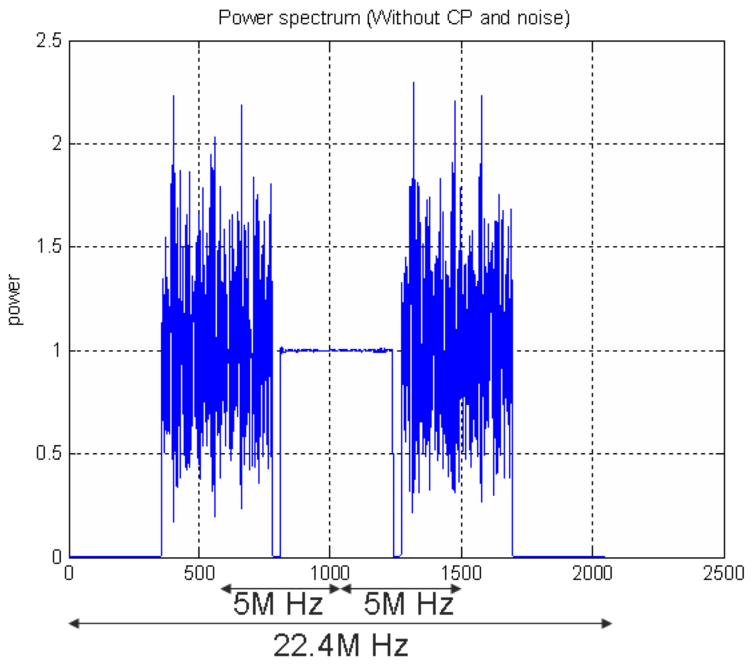


Figure 4.4: Power spectrum of 2048 points of three 5 MHz bandwidth systems with QPSK modulation in perfect channel. All symbols are without CP and received with no offset.

$2048 + CPlength - \lambda$ points. In the same way, the latter part of r is $t2$ left circular shifted by $\lambda - 2CPlength$ points, or right circular shifted by $2CPlength - \lambda$ points. We have

$$R = T1_{\theta_1} \otimes W1 + T2_{\theta_2} \otimes W2 \tag{4.1}$$

where

$$T1_{\theta_1}(k) = e^{i2\pi k\theta_1} \times T1(k), \qquad T2_{\theta_2}(k) = e^{i2\pi k\theta_2} \times T2(k),$$

$$\theta_1 = -[(2048 + CPlength) - \lambda], \qquad \theta_2 = -[2 \times CPlength - \lambda],$$

$$W1(k) = \frac{1}{2048} \sum_{n=0}^{(2048+CPlength)-\lambda-1} e^{-i\frac{2\pi kn}{2048}}, \qquad W2(k) = \frac{1}{2048} \sum_{n=(2048+CPlength)-\lambda}^{2047} e^{-i\frac{2\pi kn}{2048}}.$$

The means and the variances of $T1_{\theta_1}$ and $T2_{\theta_2}$ can be calculated as follows:

$$\overline{T1_{\theta_1}} = e^{i2\pi k\theta_1}\overline{T1(k)} = 0, \qquad \overline{T2_{\theta_2}(k)} = e^{i2\pi k\theta_2}\overline{T2(k)} = 0. \tag{4.2}$$

Since $\overline{\Re\{T1(k)\}\Im\{T1(k)\}} = 0 = \overline{\Re\{T1(k)\}}\,\overline{\Im\{T1(k)\}}$, $\Re\{T1(k)\}$ and $\Im\{T1(k)\}$ are uncorrelated. Thus

$$Var[\Re\{T1_{\theta_1}(k)\}] = \cos^2(2\pi k\theta_1)Var[\Re\{T1(k)\}] + \sin^2(2\pi k\theta_1)Var[\Im\{T1(k)\}]. \tag{4.3}$$

Now since $T1(k)$ are QPSK, 16-QAM, or 64-QAM data, $Var[\Re\{T1(k)\}] = 1/2$, and $Var[\Im\{T1(k)\}] = 1/2$. Hence

$$Var[\Re\{T1_{\theta_1}(\theta)\}] = \frac{1}{2}\cos^2(2\pi k\theta_1) + \frac{1}{2}\sin^2(2\pi k\theta_1) = \frac{1}{2}. \tag{4.4}$$

Similarly,

$$\begin{aligned} Var[\Im\{T1_{\theta_1}(k)\}] &= \cos^2(2\pi k\theta_1)Var[\Im\{T1(k)\}] + \sin^2(2\pi k\theta_1)Var[\Re\{T1(k)\}] \\ &= \frac{1}{2}\cos^2(2\pi k\theta_1) + \frac{1}{2}\sin^2(2\pi k\theta_1) = \frac{1}{2}. \end{aligned} \tag{4.5}$$

In the same way, $Var[\Re\{T2_{\theta_2}(k)\}] = \frac{1}{2}$ and $Var[\Im\{T2_{\theta_2}(k)\}] = \frac{1}{2}$. Therefore, the means and variances of $T1_{\theta_1}$ and $T2_{\theta_2}$ are independent of θ_1 and θ_2 and are also independent of λ. So, we can find the mean and variance of $|R(k)|^2$ in terms of $W1$ and $W2$ as follows. Note first that

$$\begin{aligned} Var[\Re\{R\}] &= Var[\Re\{W1\} \otimes \Re\{T1_{\theta_1}\}] + Var[\Re\{W2\} \otimes \Re\{T2_{\theta_2}\}] \\ &\quad + Var[\Im\{W1\} \otimes \Im\{T1_{\theta_1}\}] + Var[\Im\{W2\} \otimes \Im\{T2_{\theta_2}\}]. \\ Var[\Im\{R\}] &= Var[\Re\{W1\} \otimes \Im\{T1_{\theta_1}\}] + Var[\Re\{W2\} \otimes \Im\{T2_{\theta_2}\}] \\ &\quad + Var[\Im\{W1\} \otimes \Re\{T1_{\theta_1}\}] + Var[\Im\{W2\} \otimes Re\{T2_{\theta_2}\}]. \end{aligned} \tag{4.6}$$

Then

$$\overline{|R(k)|^2} = \overline{\Re\{R(k)\}^2 + \Im\{R(k)\}^2} = Var[\Re\{R(k)\}] + Var[\Im\{R(k)\}], \tag{4.7}$$

$$\begin{aligned} Var[|R(k)|^2] &= Var[\Re\{R(k)\}^2 + \Im\{R(k)\}^2] \\ &= Var[\Re\{R(k)\}^2] + Var[\Im\{R(k)\}^2] \\ &\qquad \text{(since } \Re\{R(k)\}^2 \text{ and } \Im\{R(k)\}^2 \text{ independent)} \\ &= \overline{\Re\{R(k)\}^4} - \overline{\Re\{R(k)\}^2}^2 + \overline{\Im\{R(k)\}^4} - \overline{\Im\{R(k)\}^2}^2 \\ &= 2Var[\Re\{R(k)\}]^2 + 2Var[\Im\{R(k)\}]^2 \\ &\qquad \text{(assuming } \Re\{R(k)\} \text{ and } \Im\{R(k)\} \text{ normal distributed)} \end{aligned} \tag{4.8}$$

where recall that for a normal random variable $X \sim (\mu, \sigma^2)$, we have $\overline{X^4} = 3\sigma^4 + 6\sigma^2\mu^2 + \mu^4$.

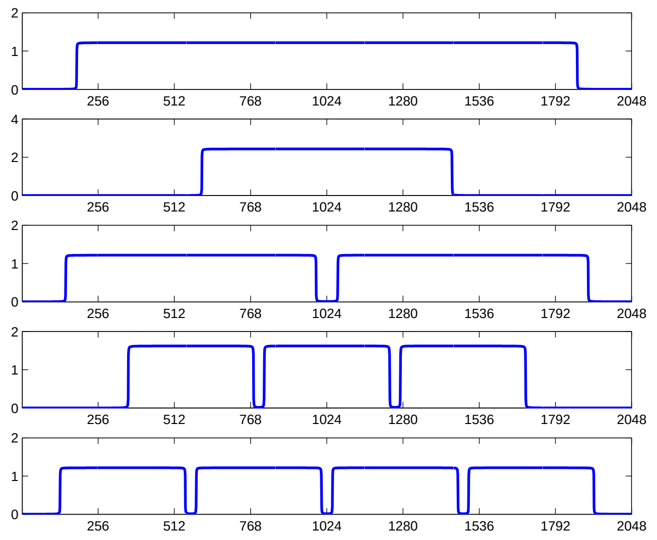


Figure 4.5: The same bandwidth distributions shown in Figure 4.1 after power normalization.

4.2.2 Bandwidth Decision Method

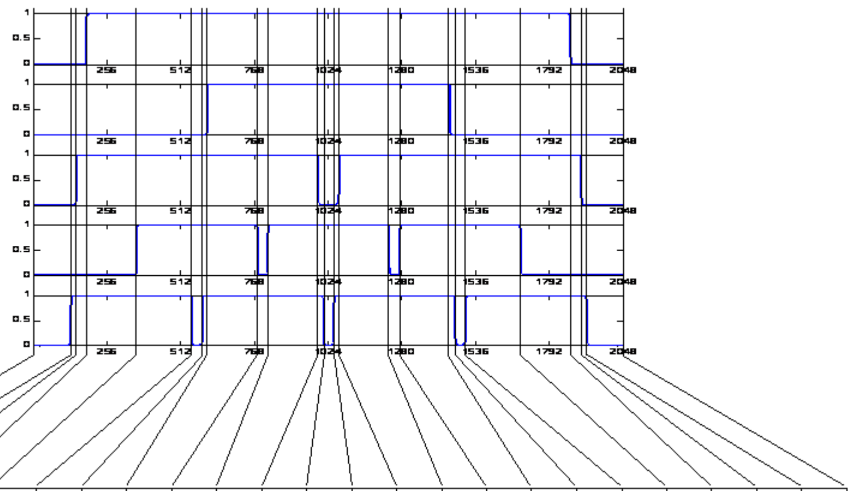
Because the number of all possible bandwidth distributions we have to detect is limited (as Figure 4.1 shows), we can make use of the concept of multiple hypotheses testing to detect them. First, we use (4.7) to calculate the mean of each point of all possible bandwidth distributions. In order to apply to the fading condition, we normalize the power to make the mean power of the 2048 points of the bandwidth distributions equal to one. After we have received 2048 points and calculated the final normalized power spectrum, compute the distance between the received power spectrum and that of all possible bandwidth distributions. Here, the distance between A and B means the sum of the squared differences between each point of A and B. Then, bandwidth distribution that has the smallest distance is our determination. Figure 4.1 shows the mean of each point of some bandwidth distributions, and Figure 4.5 shows that after power normalization. Due to the fact that some points of the bandwidth distributions are almost the same, we present two ways to calculate the distance. The main different of the two way is the number of points to calculate distance. Here we call the number of points to calculate distance as the number of dimensions of distance. So one of the two way is just calculates the distance in 2048 dimensions (2048D), and the other reduces the dimension to 23 (23D). The 23D structure is as shown in Figure 4.6, where we combine the points that are adjacent and have the same value no matter which distribution it belongs to, resulting in 23 segments. In each segment, we calculate the mean of all spectrum points in it to transfer the received 2048 points and the 2048 points of each type of bandwidth distribution to 23D. And then, like 2048D, we calculate the distance with new 23D, and choose the lowest distance as the determination. We analyze the performance of these two methods in the following.

Detection Error Probability Analysis

Consider two arbitrary bandwidth distributions A and B, and the known means of power of each points of them are $\overline{|A|^2}$ and $\overline{|B|^2}$, where $\overline{|A|^2}$ and $\overline{|B|^2}$ can be calculate by (4.7). Assume that the bandwidth distribution of the signal that we received is A. If we detect A, then that means the distance between the power of received signal and $\overline{|A|^2}$ is lower than the distance between the power of received signal and $\overline{|B|^2}$. Conversely, if we detect B, then that means some added noise has rendered the power of received signal closer to $\overline{|B|^2}$ than $\overline{|A|^2}$. Therefore, the distance between $\overline{|A|^2}$ and $\overline{|B|^2}$, and the variance of the distance between received signal and $\overline{|A|^2}$ are the key to analyze the detect error rate. Since $\overline{|A|^2}$ and $\overline{|B|^2}$ are the given value, we can calculate $dist(\overline{|A|^2}, \overline{|B|^2})^2$ easily. Following, we compute the variance of distance between any received signal and its bandwidth distribution type. For convenience, we calculate the squared distance instead of the distance.

For 2048D, let $R_A(k)$, $k = 0, ..., 2047$, denote the 2048 samples of the signal which original bandwidth distribution is A. The variance of the squared distance between R_A and $\overline{|A|^2}$ are

$$\begin{aligned} Var[dist(R_A, \overline{|A|^2})^2] &= \sum_{k=0}^{2047} Var[(|R_A(k)|^2 - \overline{|A(k)|^2})^2] \\ &= \sum_{k=0}^{2047} \{\overline{(|R_A(k)|^2 - \overline{|A(k)|^2})^4} - \overline{[|R_A(k)|^2 - \overline{|A(k)|^2}]^2}^2\} \\ &= \sum_{k=0}^{2047} \{20\overline{|R_A(k)|^2}^4 - 16\overline{|A(k)|^2}\,\overline{|R_A(k)|^2}^3 + 4\overline{|A(k)|^2}^2\overline{|R_A(k)|^2}^2\} \\ &\quad \text{(since previous assuming } R(k) \text{ normal distributed,} \\ &\quad |R_A(k)|^2 \text{ exponential distributed)} \end{aligned} \tag{4.9}$$



K	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23
Points	128	19	37	173	192	36	19	174	36	174	18	37	18	174	36	174	19	36	192	173	37	19	127

num: 0 128 147 184 357 549 585 604 778 814 988 1006 1043 1061 1235 1271 1445 1464 1500 1692 1865 1902 1921 2048

Figure 4.6: Each segment in the 23D structure calculates the mean of all spectrum points belonging to it (PUSC permutation).

where recall that for a exponential random variable $X \sim Ex(\lambda)$, we have $\overline{X} = \frac{1}{\lambda}$, $\overline{X^2} = \frac{2}{\lambda^2}$, $\overline{X^3} = \frac{6}{\lambda^3}$, $\overline{X^4} = \frac{24}{\lambda^4}$.

For 23D, let $|R'_A(k)|^2$ $k = 0, ..., 22$, denote the 23-dimension averaged signals of $|R_A|^2$ (see Figure 4.6), where $|R'_A(k)|^2 = \sum_{t=num(k-1)}^{num(k)-1} \frac{1}{num(k)-num(k-1)} |R_A(t)|^2$. Since $R_A(k)$, $k = 0, ..., 2047$ are independent,

$$\overline{|R'_A(k)|^2} = \sum_{t=num(k-1)}^{num(k)-1} \frac{1}{num(k) - num(k-1)} \overline{|R_A(t)|^2}, \tag{4.10}$$

$$Var[|R'_A(k)|^2] = \sum_{t=num(k-1)}^{num(k)-1} \frac{1}{\{num(k) - num(k-1)\}^2} Var[|R_A(t)|^2]. \tag{4.11}$$

Similar to 2048D, the variance of the squared distance between R'_A and $\overline{|A|^2}$

$$\begin{aligned} Var[dist(R'_A, \overline{|A|^2})^2] &= \sum_{k=0}^{22} Var[(|R'_A(k)|^2 - \overline{|A'(k)|^2})^2] \\ &= \sum_{k=0}^{22} \{\overline{(|R'_A(k)|^2 - \overline{|A'(k)|^2})^4} - \overline{[|R'_A(k)|^2 - \overline{|A'(k)|^2}]^2}^2\} \\ &= \sum_{k=0}^{22} \{20\overline{|R'_A(k)|^2}^4 - 16\overline{|A'(k)|^2}\,\overline{|R'_A(k)|^2}^3 + 4\overline{|A'(k)|^2}^2\overline{|R'_A(k)|^2}^2\} \end{aligned} \tag{4.12}$$

Theoretical Performance Analysis of 23D and 2048D

As preceding analyzed, the performance of bandwidth detection depends on the mean and variance of distance between the power of received signal and different bandwidth distribution.

We already known the squared power of bandwidth distributions is exponential distribution, but we cannot find the probability distribution of the squared distance of the squared power of bandwidth distributions. So we try using the Q function at this stage to calculate the probability that original A but detect B is

$$P_e(B|A) = Q(\frac{dist(\overline{|A|^2}, \overline{|B|^2})^2}{\sqrt{Var[dist(|R_A|^2, \overline{|A|^2})^2]}}), \tag{4.13}$$

where $Q(x) = \frac{1}{2\pi}\int_x^\infty e^{-\frac{t^2}{2}}dt$. Then, the totally error probability of original A_k is

$$P_e = 1 - \prod_{i=1, i\neq k}^{49} \{1 - P_e(A_i|A_k)\}, \tag{4.14}$$

where A_i, i = 1–49 and $i \neq k$ are the 48 types of bandwidth distribution other than A_k. Consider the AWGN channel case. We just add the variance of noise to $Var[\Re\{R\}]$ and $Var[\Im\{R\}]$, and use the new $\overline{|R|^2}$ to compute as above. So we can calculate the theoretical performance of 2048D and 23D in AWGN channel as Figure 4.7 shows.

However, the above analysis is assumes usage of all subchannels and no pilot subcarrier. If we want to apply this method to a true 802.16e OFDMA system, we need to modify the means of all possible bandwidth distributions and the resulting formula is still similarly to (4.7). We can even collect several 2048 points and average them in one detection. If we collect n times 2048 points and average, the variance in squared distance will be divided by n and the detection error rate would decrease.

In this section, we have considered the bandwidth structure in 20 MHz. There may be systems that straddle two 20 MHz bandwidths, which follow the rules for the center frequency of a system of a certain bandwidth specified by the system profile in IEEE 802.16e [2]. If we want to consider this condition, we have to calculate the mean power spectrum of the straddling bandwidth distribution and add the straddle condition to all possible bandwidth distributions. This would result in higher error rate and computa-

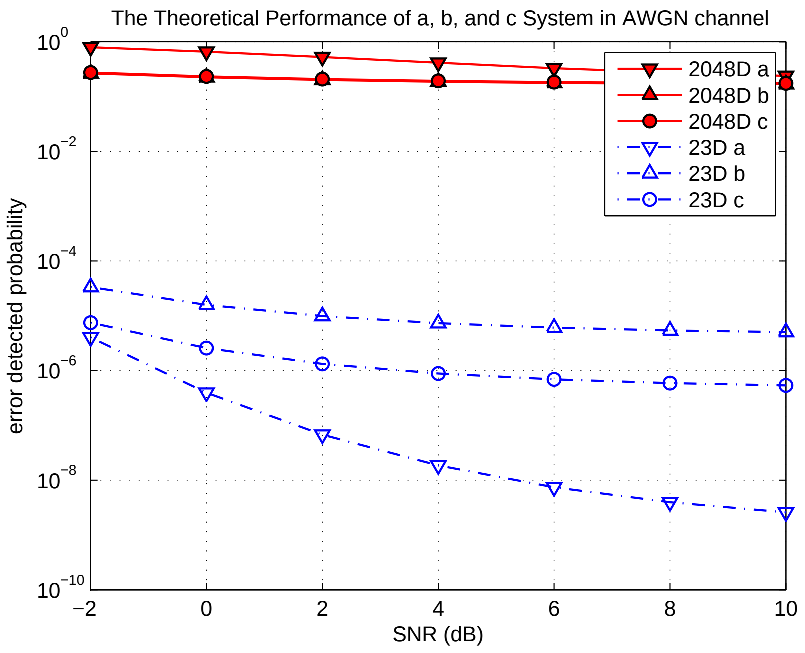


Figure 4.7: The theoretical performance of the detection of type a, b, and c in AWGN channel.

tional complexity, because the number of distances needed to calculate and compare is increased.

4.3 CP Detection

After detecting the bandwidth and the FFT size, when performing initial network entry, the MS should search all possible values of CP length until it finds the CP being used by the BS. That is, we have known the FFT size and only have to detect four types of CP length. In our research, we compare some CP decision methods that are also initial timing offset estimation methods and are completely blind relying solely on the repetitive signal structure of OFDMA symbols. This appears simplest and suitable for use in IEEE 802.16e OFDMA. Finally, we choose one to use in our program. Note that here we need to detect CP for just one system, and the start time of this signal has limited inaccuracy.

4.3.1 CP Decision Method

Preamble Complex Conjugate Correlation [23]

In an OFDMA system, the first symbol of a frame is preamble, whose data values in the frequency domain are real. That makes the preamble complex conjugate symmetric in the time domain. Denote N by the FFT size and let r be the received preamble signal in the time domain. The conjugate symmetric correlation can be written as

$$X_{cs}(n) = \sum_{i=1}^{\frac{N}{2}-1} r(n+\frac{N}{2}-i) \times r(n+\frac{N}{2}+i). \tag{4.15}$$

We can find

$$\widehat{n} = \arg \max_{-CP<n-CP<CP} X_{cs}(n), \tag{4.16}$$

where $\widehat{n}$ is the start of the symbol (without CP). Since our aim is to detect CP length, we find the signal length before the start of the symbol, and that length is our determined CP length.

The advantage of this method is that we can use only one symbol (preamble) to detect, but the disadvantage is that we have to calculate $N/2$ multiplications for every conjugate symmetric correlation, which results in an overall complexity $O(Nn)$. To reduce the complexity, we consider the following detection method.

CP Complex Conjugate Correlation [24]

The simplest and appropriate CP detection method for IEEE 802.16e OFDMA is blind estimation based on CP structure. Since there is a CP in front of each OFDMA symbol, we test four CP lengths to do CP complex conjugate correlation and find the mean of the correlation values of several OFDMA symbols. The CP length that has the largest correlation value is the determined value. Therefore, let

$$\hat{\tau} = \arg\max\{|\lambda(\tau)|^2\} \tag{4.17}$$

where

$$\lambda(\tau) = \frac{1}{c} \sum_{k=\tau}^{\tau+c-1} r(k) r^*(k+N),$$

with c being the testing CP length. There are some design parameters in this method. We can design the number of correlation symbols. The more symbols we use in the detection, the better accuracy we obtain, but the more time it costs. So the decision of this parameter should be considered with the detection error penalty.

Consider the computational complexity of this method. Because the calculation of CP correlation can use the sliding window technique to reduce the computational complexity, it results in a complexity of $O(\tau)$.

Performance Analysis

In this section, we analyze the performance of the CP complex conjugate correlation detection method that we employ. Let the number of symbols that we collect to do correlation be L. Than, when we are testing the right CP length, $c|\lambda(\tau)|^2$ would be equal to L^2 in perfect channel. But the signal samples that we do correlation with wrong a CP length may overlap those that we do correlation with the right CP length, which results in the condition that when we test the wrong CP length, the expected value of $|\lambda(\tau)|^2$ would not be equal to zero, although the correlation with wrong CP length are low. Figure 4.8 shows an example where the real CP is 1/8, so the signals in the gray block are all the same signals (ignore the noise and CFO), then, $|\lambda(\tau)|^2$ of the first symbol with CP 1/4 will result in $(1/2)^2$, and $|\lambda(\tau)|^2$ of first symbol with CP 1/8, 1/16 or 1/32 will all result in 1. We can easily calculate that how many symbols they may overlap by finding the lowest common multiple. Figure 4.9 depicts the data. Note that, we lists the number of symbols that are not repeat to the start condition, that is the length we listed is the lowest common multiple of real CP symbol and testing CP symbol. Let the number of the No. of testing CP symbols that are not more than L be M and the correlation value due to overlap one time be O. Than, the expected value of the wrong CP correlation value is $(M \times O)^2$.

4.4 Simulation Results

The system parameters for our simulation are defined in Table 4.1 and chap. 4. The simulation platform is Matlab. We only use 16-QAM modulation in the simulation for simplicity. Note that, in bandwidth detection error, the MS will fail to find the preamble.

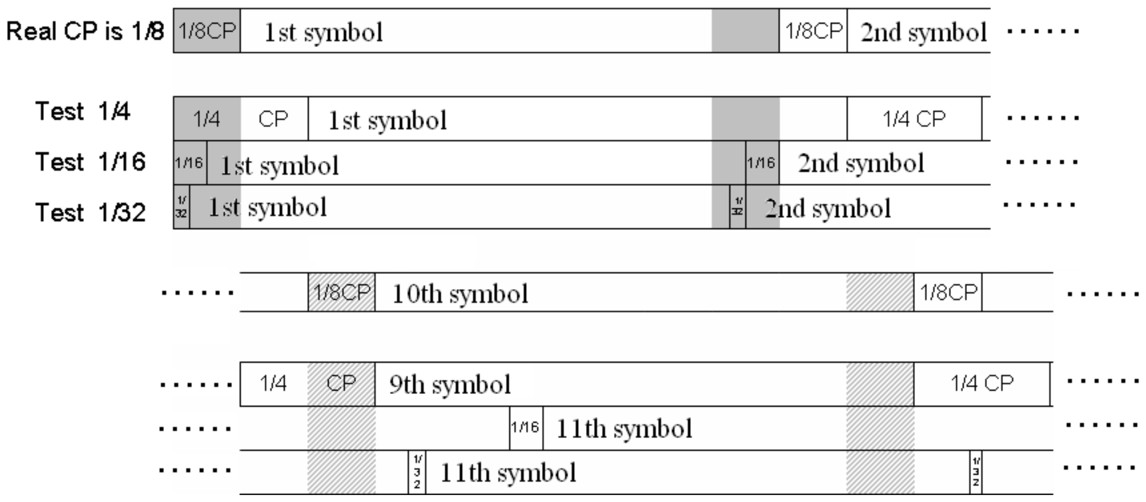


Figure 4.8: An example of CP overlap between different CP lengths.

real CP / testing CP	1/4		1/8		1/16		1/32	
1/4			1,9	1/2	1,6,12,17	1/4	1,6,12,18,23,29,35,40	1/8
1/8	1,10	1			1,17	1/2	1,11	1/4
1/16	1,7,14,20	1	1,18	1			1,33	1/2
1/32	1,5,10,15,19,24,29,33	1	1,12	1	1,34	1		

No. of testing CP symbols which CP overlap to real CP

The experience of testing CP correlation due to one times of the CP overlap

Figure 4.9: Index number of testing CP symbols with CP overlap to real CP and the correlation value that results.

Therefore the penalty for detection errors is the need to detect one more time, or to waste one more frame. So we just simulate the error rate to 10^{-4}. The mobile speed is from 0 to 240 km/hr. The signal-to-noise ratio (SNR) used in our simulations means the ratio of the variance of the received signal samples to that of the noise samples and the CFO is zero in bandwidth detection simulation.

4.4.1 Bandwidth Detection

The bandwidth detection simulation flow is as shown in Figure 4.10. First, we randomly generate data and put them in subcarriers. Second, we transform data to time domain and add CP. Third, resample the signals to 20 MHz. Besides the Nfft and CP, the three steps are all the same to all systems. Fourth, we observe the rule of center frequency allocation to shift the systems in frequency domain. Due to the center frequency allocation is specified, the number of systems are not more than four. Finally, the signals of all systems pass through the channel and we receive the summation of them at the receiver. Note that, in the flow the resampling is a Matlab function that can resample the signals at another sampling rate. The resample function is based on zero frequency and expand the bandwidth to both sides of zero, so we have to shift the signal frequency to between $-N_{fft}/2$ and $N_{fft}/2-1$ before resampling. After resampling we shift the frequency back to 1 and N_{fft}. In addition, the timing offset and CP length of each system are assigned randomly in 0 – $2047 + CPlength$ samples and four types.

Since the earlier analysis is based on using of all subchannels, no pilot subcarriers, no preamble, and every 2048 points execute one test, Figures 4.11, 4.12 and 4.13 show the bandwidth detection of one 20 MHz system simulation result in the above conditions with AWGN, SinglePath, SUI-3, and SUI-4 channels. We can find that in most cases the performance of 2048D is better than 23D. Besides, the performance of 2048D is much better than the theoretical result that we desired above. The discrepancy may result from that we have used Q function to calculate $dist^2$ but actually $dist^2$ is not normal distributed. Figures 4.14 and 4.15 depict the histogram of $dist(a, \overline{a})^2$ with 2048D and 23D that we get by experiment 10^4 times in AWGN channel with SNR 0dB. Since the points in 23D is the mean of several points, the squared distance distribution at 23D is

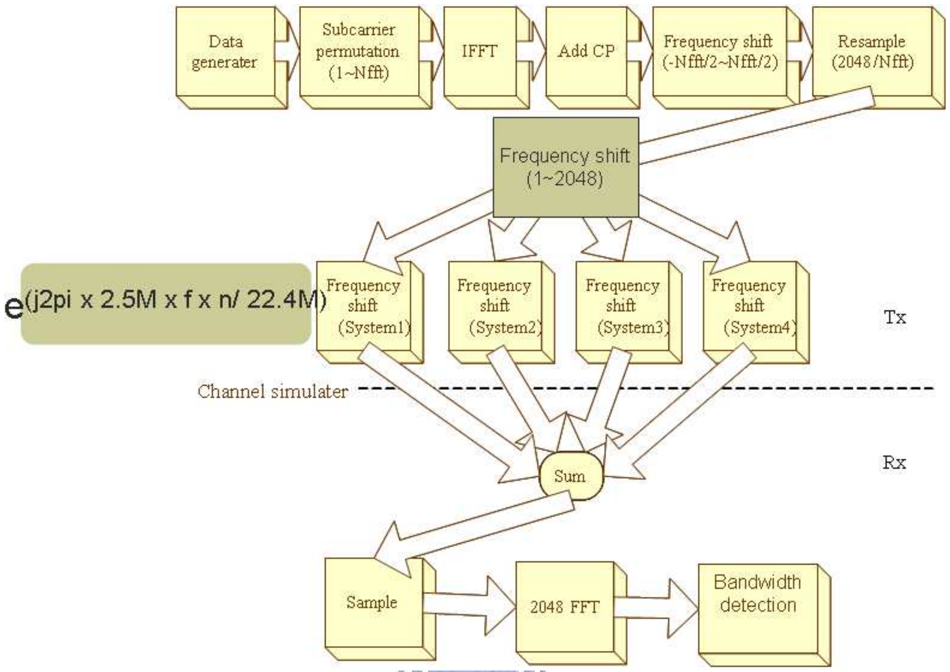

Data generater
Subcarrier permutation (1~Nfft)
IFFT
Add CP
Frequency shift (-Nfft/2~Nfft/2)
Resample (2048/Nfft)
Frequency shift (1~2048)
$e^{(j2pi \times 2.5M \times f \times n/22.4M)}$
Frequency shift (System1)
Frequency shift (System2)
Frequency shift (System3)
Frequency shift (System4)
Tx
Channel simulater
Rx
Sum
Sample
2048 FFT
Bandwidth detection

Figure 4.10: Flow diagram of bandwidth detection simulation.

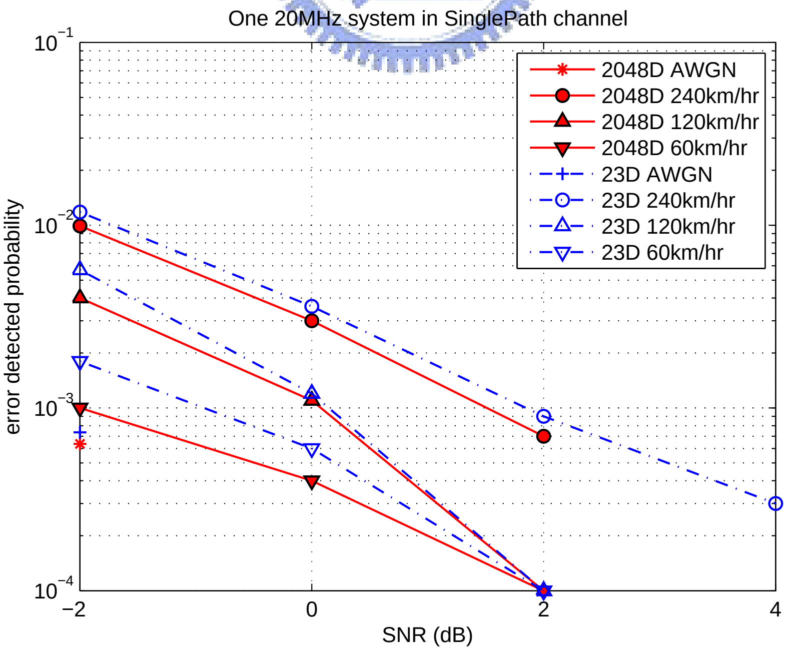


Figure 4.11: Detection error rates of one 20 MHz system in SinglePath and AWGN channels.

more like normal distribution than at 2048D, this may result in the theoretical value is more close to experimental value in 23D than that in 2048D. We can find, moreover, the performance in SUI-4 is better than in SUI-3. It may result from the coherent bandwidth $f_0 = 1/T_m$, where T_m is the maximum access delay. Since the T_m of SUI-4 is larger, the coherent bandwidth in SUI-4 is smaller. Therefore the spaced-frequency correlation function of SUI-4 channel is more stable and suitable for our bandwidth detection method. Figures 4.16 and 4.17 show the spaced-frequency correlation functions of SUI-3 and SUI-4, respectively. In Figure 4.18, we show the pie chart of the bandwidth detection error distribution in moving speed 120 km/hr, SNR 0 dB, SUI-3 channel. As may be expected the more similar the bandwidth distribution is, the higher the probability to be detected.

Figure 4.19 shows the bandwidth distribution detection result in SUI-3 channel of one system with 10MHz bandwidth and the center frequency is 3.5 GHz and (3.5G − 5M) Hz, that bandwidth distributions are "b" and "c". Because the central frequency shift 5 MHz is not an integer multiple of the subcarrier spacing (10937.5 Hz), the power spectrum would exhibits some interference like Figure 4.4 show, which results in that the detection performance of this system is worse than the system whose center frequency is 3.5 GHz in 2048D. Since the distributions "b" and "c" are very close to "i+j" and "h+i", the error rate drop very slowly when the SNR is larger than 2 dB in the 2048D method. However in 23D, the performance of "c" is better than "b" and the error rate drops when the SNR becomes larger. This may result from that the 23D method takes the average of some points to calculate the distance, so that the influence of frequency shift of fractional samples is less and performance of "c" is better than "b" in theoretical performance.

Now we consider the computational complexity of the 2048D and the 23D methods. Let S denote the number multiple to 2048 points, and we use $S \times 2048$ points to execute one detection. For 2048D, we calculate the power of each points results in $S \times 2048 \times 2$ multiplications, and average the power of S symbols result in 2048 multiplications (if $S = 1$, ignore it). The distance between received power spectrum and 49 types power spectrum results in 2048×49 multiplications. So totally $S \times 2048 \times 2 + 2048(S > 1) + 2048 \times 49 = 4096S + 2048(S > 1) + 100352$ multiplications. For 23D, we calculate

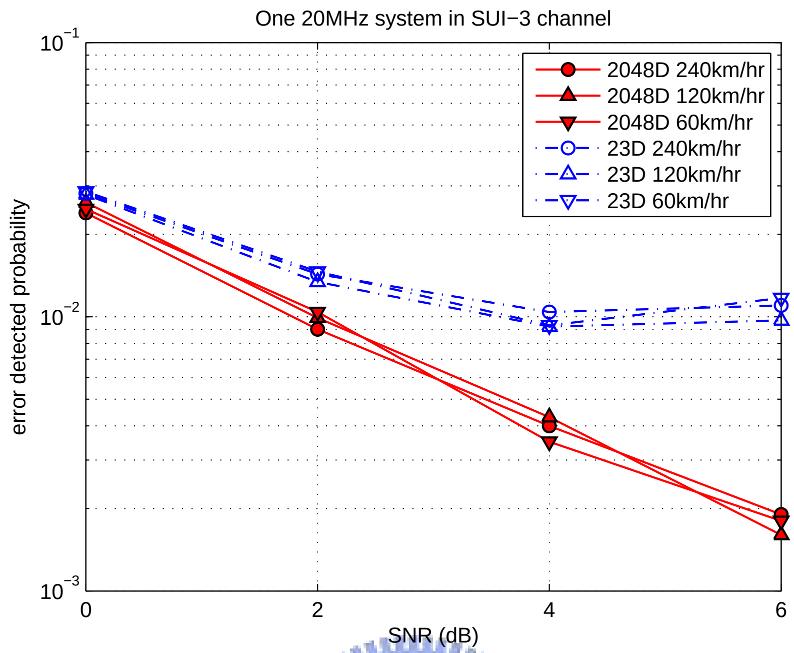


Figure 4.12: Detection error rates of one 20 MHz system in SUI-3 channel.

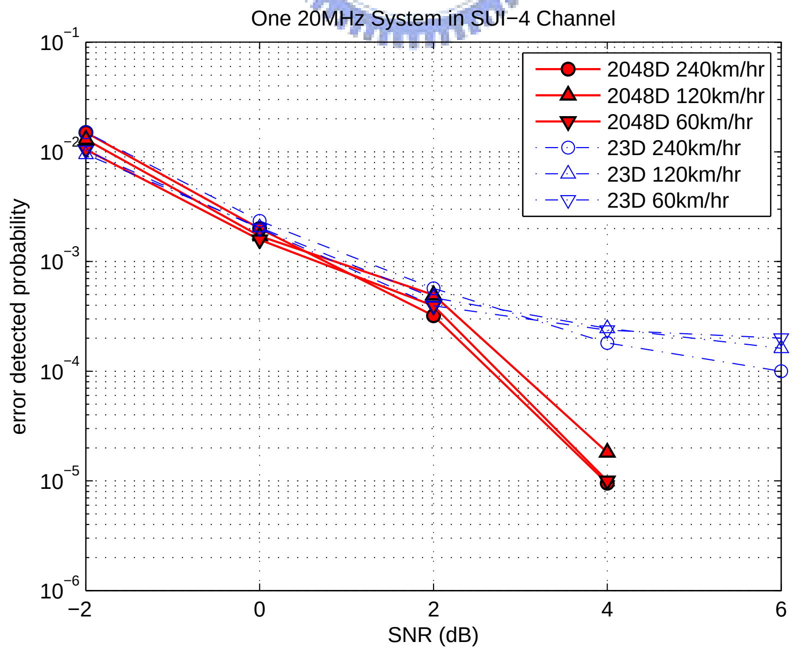


Figure 4.13: Detection error rates of one 20 MHz system in SUI-4 channel.

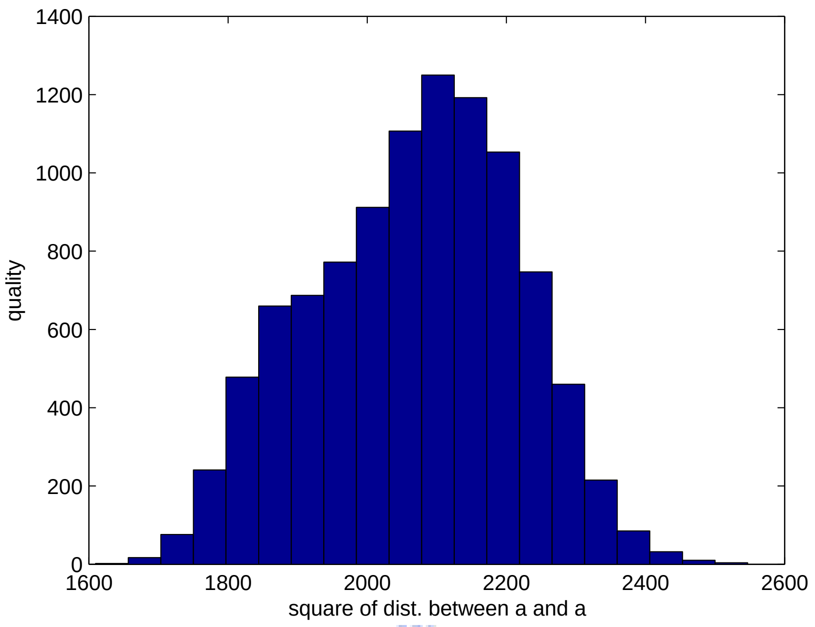


Figure 4.14: The histogram of the squared of distance between “a” system and mean power of “a” in 2048D.

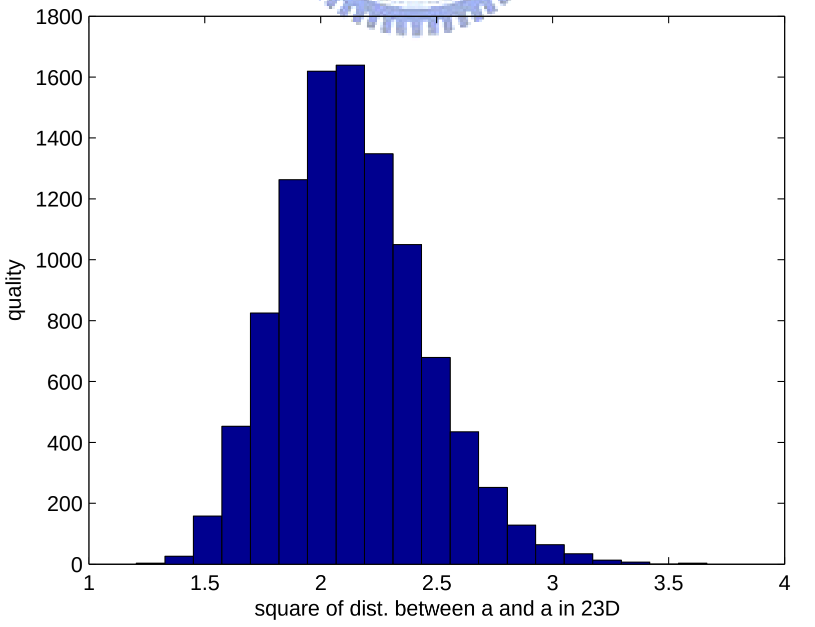


Figure 4.15: The histogram of the square of distance between “a” system and mean power of “a” in 23D.

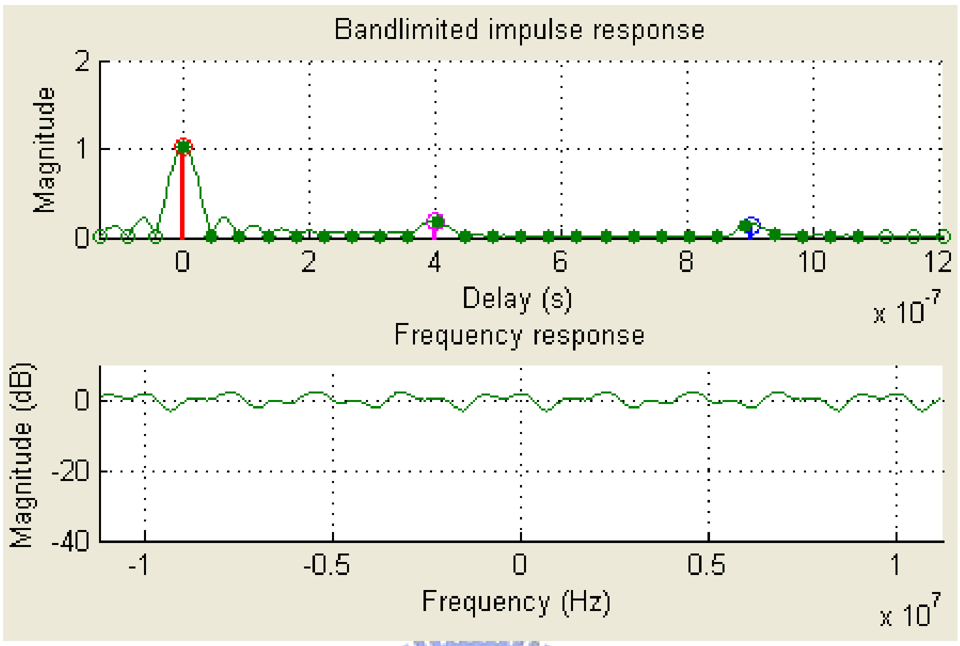


Figure 4.16: The spaced-frequency correlation function of the SUI-3 channel.

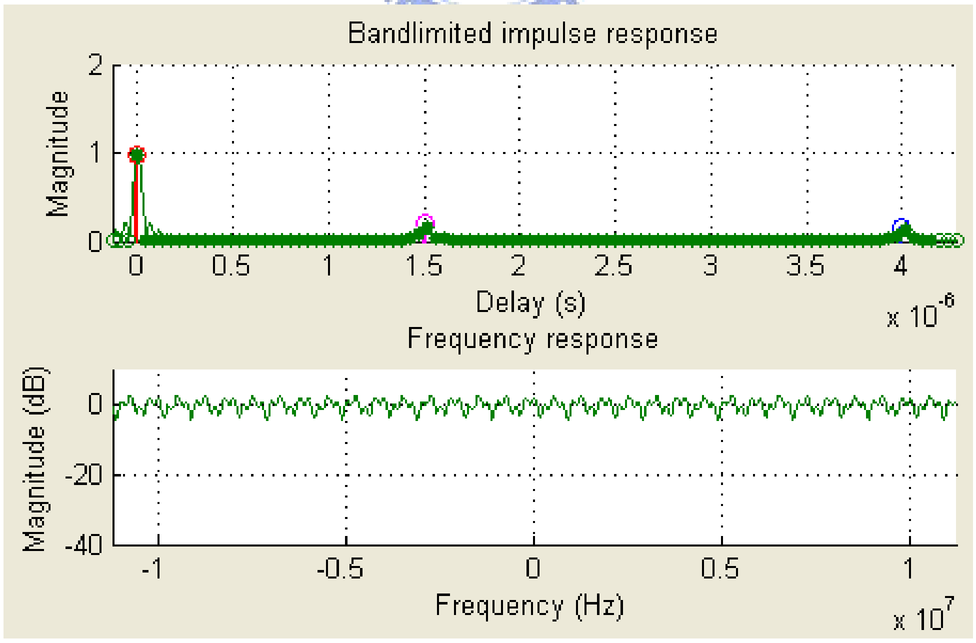


Figure 4.17: The spaced-frequency correlation function of the SUI-4 channel.

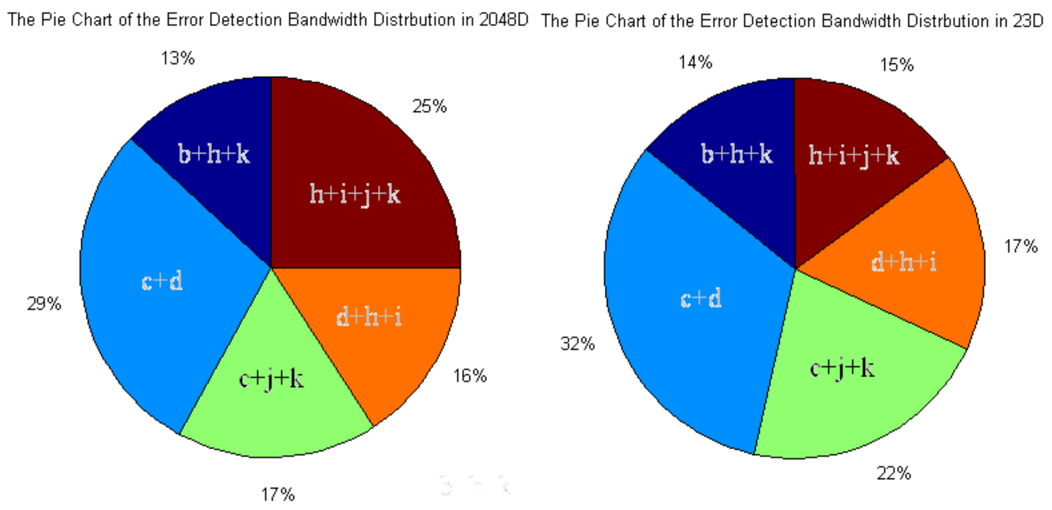


Figure 4.18: Distribution of bandwidth detection errors at SNR 0 dB, SUI-3 channel.

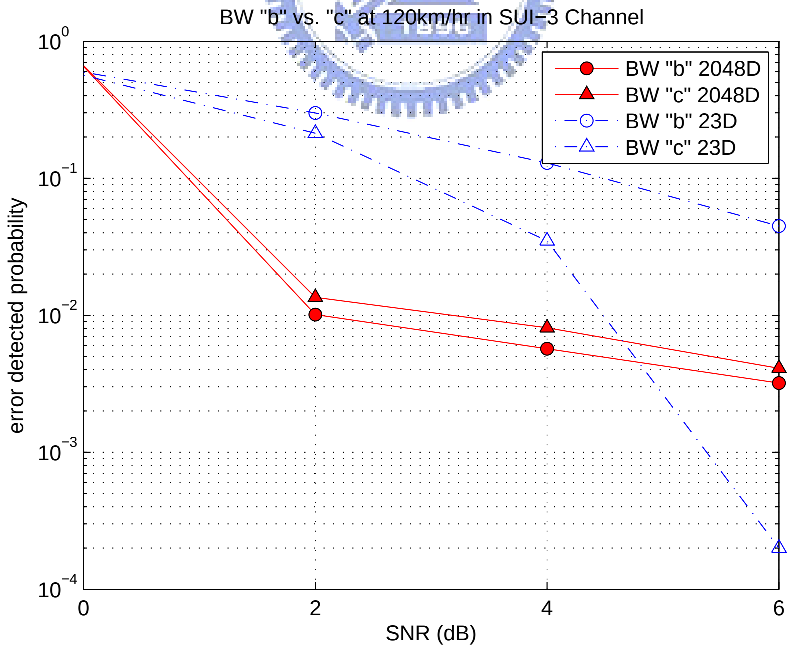


Figure 4.19: One 10 MHz center frequency 3.5 GHz system vs. one 10 MHz center frequency 3.5 GHz − 5 MHz.

the power of each points as 2048D are totally $S \times 2048 \times 2 + 2048$ multiplications, and transfer to 23D need 23 multiplications. The distance between received power spectrum and 49 types power spectrum results in 23×49 multiplications. So totally $S \times 2048 \times 2 + 2048(S > 1) + 23 + 23 \times 49 = 4096S + 2048(S > 1) + 1150$ multiplications. The computational complexity of 23D is much lower than 2048D.

4.4.2 CP Detection

The system of CP detected simulation is as mentioned above and the PUSC permutation with using segment 0 to allocate data subcarrier. In this subsection, we depict the performance and computational complexity of the CP complex conjugate correlation method. The range of timing offset is from -15 to CP randomly assigned. Figure 4.20 shows the CP detection error rate of 1/8 and 1/16 CP system in AWGN channel and summation the correlation of $L = 5$, 6, or 7 symbols every one test. Because the fewer number of correlated points, the larger variance of correlation value resulted. Since the correlation length of 1/16 CP is shorter, the variance of CP correlation value of original 1/16 CP is larger. so the error rate is higher than original 1/4 CP. In Figure 4.21, we show the pie chart of type of error detected CP when original 1/8 CP, $L = 5$, and SNR 4 dB. Since the preceding section has analyzed the expected value of CP correlation of, result from CP overlap, 1/4, 1/8, 1/16, and 1/32 CP are 0.5, 5, 1, and 1 when $L = 5$ and original 1/8 CP. Therefore it has the highest probability of detected error to 1/32 CP, and the second is 1/16 CP. Because there is long correlated length and less expected value in 1/4 CP, it is almost not to be error detected.

Now we consider the computational complexity of this method. Let the search range of timing offset be γ. Since we need to do correlation with each type of CP and each correlation use 2 multiplications, that result in $2 \times (1/4 + 1/8 + 1/16 + 1/32) \times N_{fft} \times L$ multiplications. After every symbol correlation, it use one multiplications to average the correlation with different test CP length, that result in $L \times 4$ multiplications. Because the usage of sliding windows, the number of multiplication related to γ is $(\gamma - 1) \times 3$ (3 include 2 for correlation and 1 for average). So the total number of multiplications is $2(1/4 + 1/8 + 1/16 + 1/32) \times N_{fft} \times L + 4L + 3(\gamma - 1)$. Note that, since the CP lengths

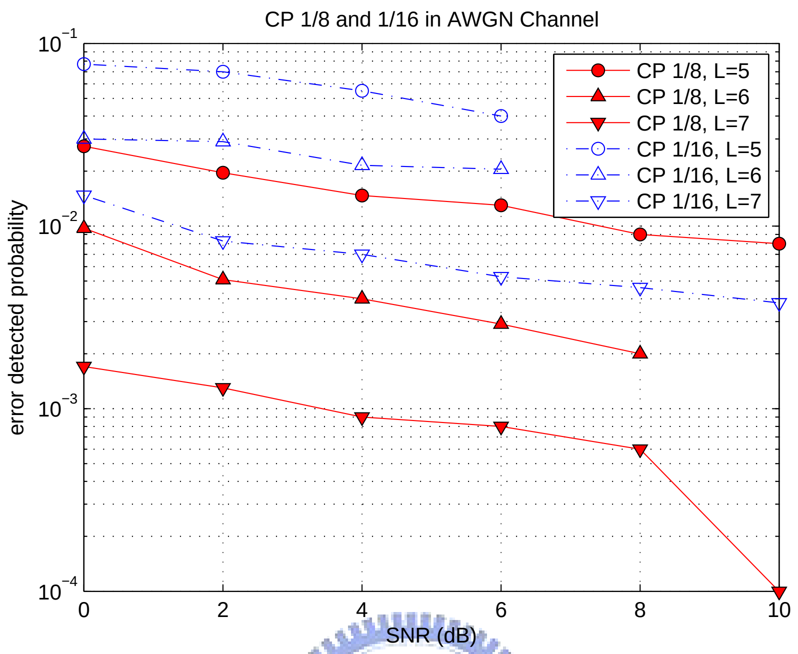


Figure 4.20: The CP detection error rate in AWGN channel.

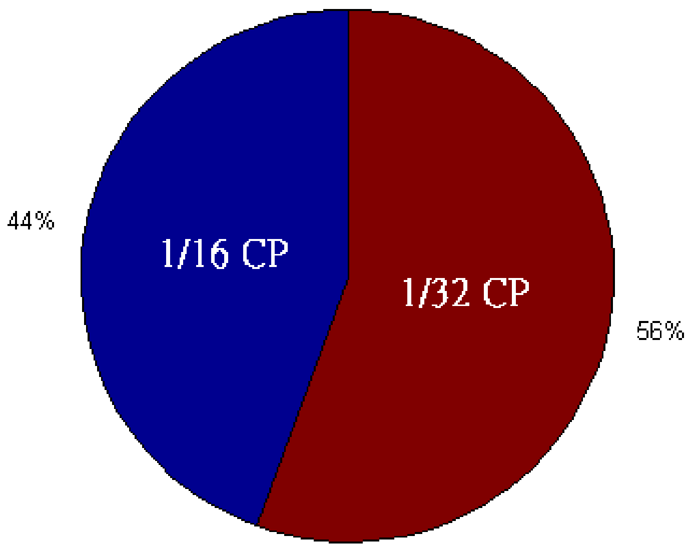


Figure 4.21: The types of error detect CP when original 1/8 CP, $L = 5$, and SNR 4 dB.

are all integer powers of 2, the $4L$ multiplications can be ignored.

Chapter 5

Information Element Handling and DSP Implementation

After finding the bandwidth and CP, as MS performing the initial network entry. The next procedure is to read the data in the signal. See Figure 2.7, the frame structure is composed of preamble, FCH, DL-MAP, and bursts. The permutation rules of burst are defined by MAC, and the related information (contains subchannel offset, number of subchannels, OFDMA symbol offset, and number of OFDMA symbols of each burst) are wrote in the DL-MAP. The front of DL-MAP is FCH that contains the information of DL-MAP (Figure 2.8 shows the FCH format). So in the DL system, MS must read the FCH first and find the DL-MAP base on the repetition coding indication read in FCH and then find the burst and read data.

5.1 Information Element Handling

Because MS must expand the information element every time it received a frame. To improve the information element expanding complexity is important. Here we consider the implementation and optimization of information element handling techniques on DSP.

Following we introduce the optimization techniques used in the system. Besides uti-

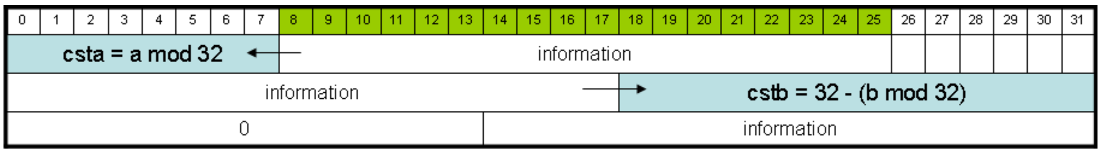
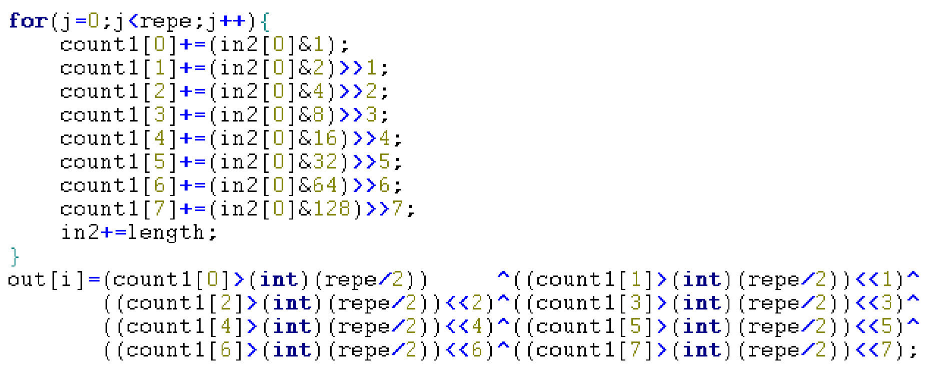


Figure 5.1: The procedure of read the information from 32bits aligned IE.

lizing the compiler options we mentioned before, we adjust the program for compiler to software-pipeline the loop automatically and use the function that are TMS320C6000 C/C++ Compiler Intrinsics. Since the DL-MAP stream contains several type of IE and each information element length is different even variable. We suppose that the DL-MAP stream is 32 bits align, that reduce the probability of information element be separate to different integer. So we transfer the DL-MAP stream to 32 bits align and then use function uint _ extu(src2,csta,cstb), extracts the specified field in src2, zero-extended to 32 bits. The extract is performed by a shift left followed by a unsigned shift right; csta and cstb are the shift left and shift right amounts, respectively. Let the input signal, R, are 32 bits align (if not transfer them), and the information we want to read is among the ath bit to the bth bit. The output of _extu(R,(a mod 32),(32-(b mod 32))) is the information we desired. Figure 5.1 depicts this method. We can use this method to handle all information in FCH and DL-MAP. Furthermore, we also make use of several useful compiler options supported by TI's compiler. As mentioned in chapter 3, the compiler options can be used to optimize code size or execution performance. The major compiler options we utilize are -o3, -k, and -pm -op2.

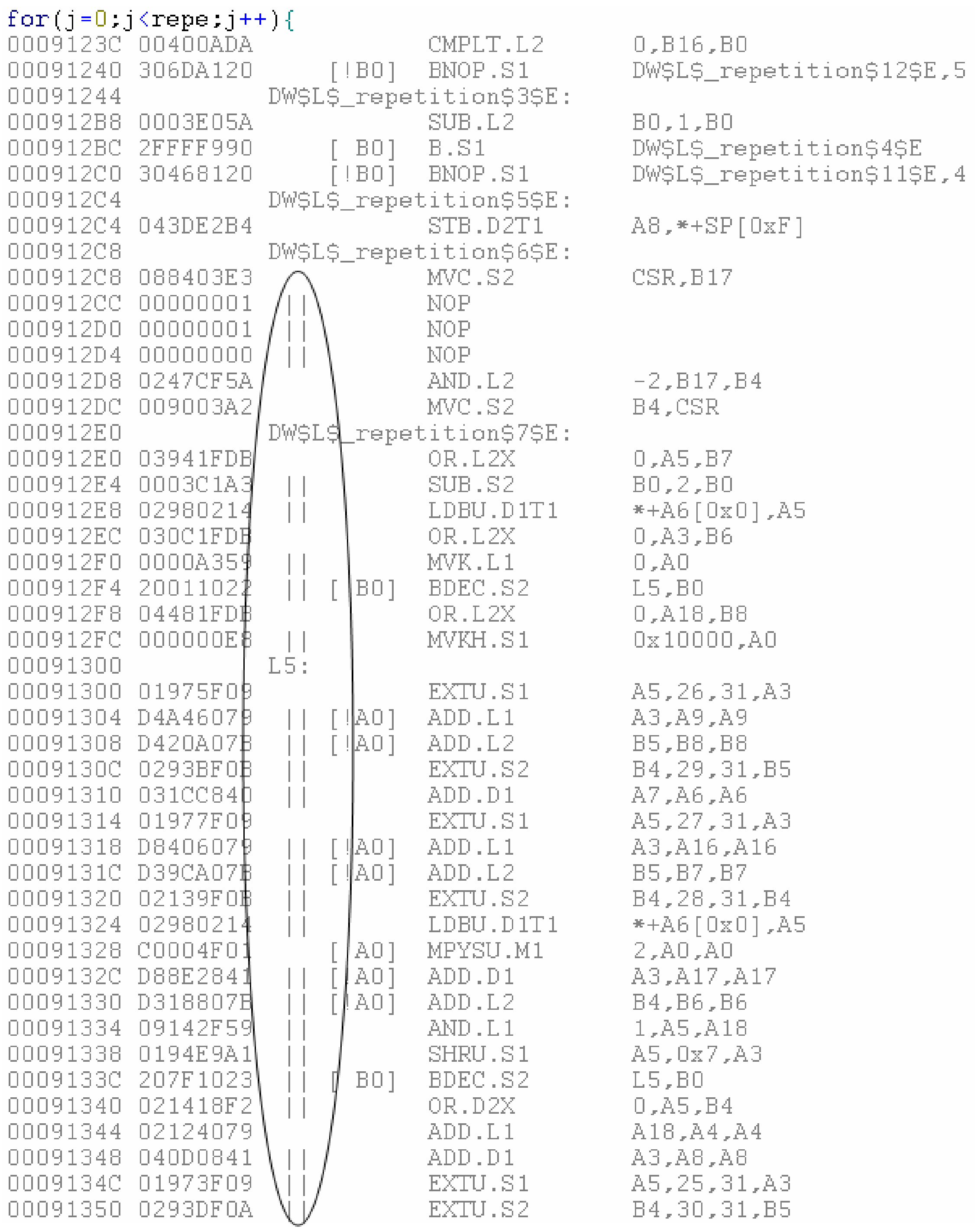
5.2 The Optimization Result

Figure 5.2 shows the method we used to decode the repetition coding and Figure 5.3 shows some assembly code depict the optimization result. Figure 5.4 and Figure 5.5 shows the byte to words c code and part of assembly code.



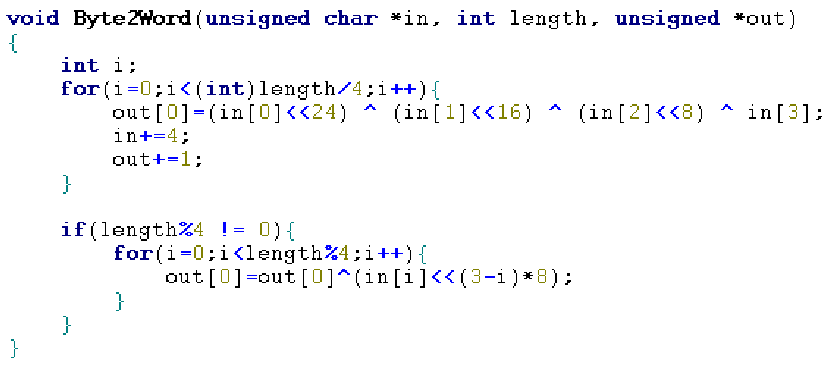
```
for(j=0;j<repe;j++){
    count1[0]+=(in2[0]&1);
    count1[1]+=(in2[0]&2)>>1;
    count1[2]+=(in2[0]&4)>>2;
    count1[3]+=(in2[0]&8)>>3;
    count1[4]+=(in2[0]&16)>>4;
    count1[5]+=(in2[0]&32)>>5;
    count1[6]+=(in2[0]&64)>>6;
    count1[7]+=(in2[0]&128)>>7;
    in2+=length;
}
out[i]=(count1[0]>(int)(repe/2))     ^((count1[1]>(int)(repe/2))<<1)^
        ((count1[2]>(int)(repe/2))<<2)^((count1[3]>(int)(repe/2))<<3)^
        ((count1[4]>(int)(repe/2))<<4)^((count1[5]>(int)(repe/2))<<5)^
        ((count1[6]>(int)(repe/2))<<6)^((count1[7]>(int)(repe/2))<<7);
```

Figure 5.2: A part of C code for decode repetition coding.



```
for(j=0;j<repe;j++){
0009123C 00400ADA                CMPLT.L2       0,B16,B0
00091240 306DA120         [!B0]  BNOP.S1        DW$L$_repetition$12$E,5
00091244          DW$L$_repetition$3$E:
000912B8 0003E05A                SUB.L2         B0,1,B0
000912BC 2FFFF990         [ B0]  B.S1           DW$L$_repetition$4$E
000912C0 30468120         [!B0]  BNOP.S1        DW$L$_repetition$11$E,4
000912C4          DW$L$_repetition$5$E:
000912C4 043DE2B4                STB.D2T1       A8,*+SP[0xF]
000912C8          DW$L$_repetition$6$E:
000912C8 088403E3                MVC.S2         CSR,B17
000912CC 00000001   ||           NOP
000912D0 00000001   ||           NOP
000912D4 00000000   ||           NOP
000912D8 0247CF5A                AND.L2         -2,B17,B4
000912DC 009003A2                MVC.S2         B4,CSR
000912E0          DW$L$_repetition$7$E:
000912E0 03941FDB                OR.L2X         0,A5,B7
000912E4 0003C1A3   ||           SUB.S2         B0,2,B0
000912E8 02980214   ||           LDBU.D1T1      *+A6[0x0],A5
000912EC 030C1FDB                OR.L2X         0,A3,B6
000912F0 0000A359   ||           MVK.L1         0,A0
000912F4 20011022   ||    [ B0]  BDEC.S2        L5,B0
000912F8 04481FDB                OR.L2X         0,A18,B8
000912FC 000000E8   ||           MVKH.S1        0x10000,A0
00091300          L5:
00091300 01975F09                EXTU.S1        A5,26,31,A3
00091304 D4A46079   ||    [!A0]  ADD.L1         A3,A9,A9
00091308 D420A07B   ||    [!A0]  ADD.L2         B5,B8,B8
0009130C 0293BF0B   ||           EXTU.S2        B4,29,31,B5
00091310 031CC840   ||           ADD.D1         A7,A6,A6
00091314 01977F09                EXTU.S1        A5,27,31,A3
00091318 D8406079   ||    [!A0]  ADD.L1         A3,A16,A16
0009131C D39CA07B   ||    [!A0]  ADD.L2         B5,B7,B7
00091320 02139F0B   ||           EXTU.S2        B4,28,31,B4
00091324 02980214   ||           LDBU.D1T1      *+A6[0x0],A5
00091328 C0004F01         [ A0]  MPYSU.M1       2,A0,A0
0009132C D88E2841   ||    [ A0]  ADD.D1         A3,A17,A17
00091330 D318807B   ||    [!A0]  ADD.L2         B4,B6,B6
00091334 09142F59   ||           AND.L1         1,A5,A18
00091338 0194E9A1   ||           SHRU.S1        A5,0x7,A3
0009133C 207F1023   ||    [ B0]  BDEC.S2        L5,B0
00091340 021418F2   ||           OR.D2X         0,A5,B4
00091344 02124079                ADD.L1         A18,A4,A4
00091348 040D0841   ||           ADD.D1         A3,A8,A8
0009134C 01973F09   ||           EXTU.S1        A5,25,31,A3
00091350 0293DF0A   ||           EXTU.S2        B4,30,31,B5
```

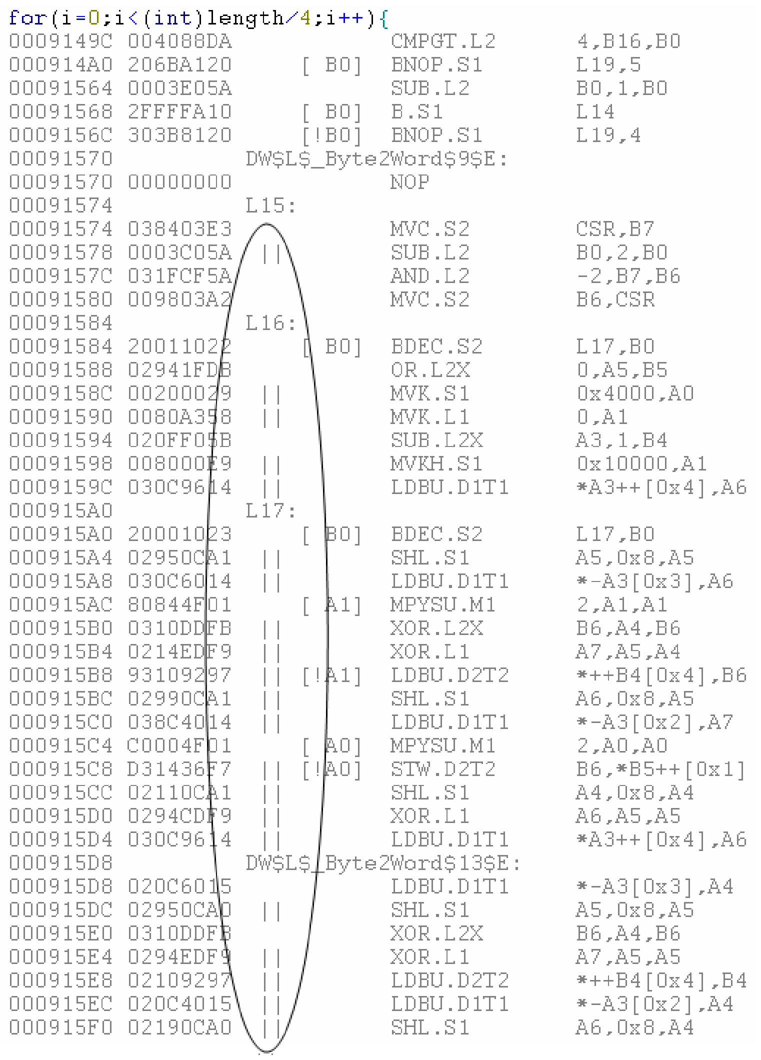
Figure 5.3: A part of assembly code for decode repetition coding. The adjacent parallel lines in the ellipse means which instructions are parallel.



```c
void Byte2Word(unsigned char *in, int length, unsigned *out)
{
    int i;
    for(i=0;i<(int)length/4;i++){
        out[0]=(in[0]<<24) ^ (in[1]<<16) ^ (in[2]<<8) ^ in[3];
        in+=4;
        out+=1;
    }

    if(length%4 != 0){
        for(i=0;i<length%4;i++){
            out[0]=out[0]^(in[i]<<(3-i)*8);
        }
    }
}
```

Figure 5.4: A part of C code for byte align transfer to 32 bits align.



```
for(i=0;i<(int)length/4;i++){
0009149C 004088DA              CMPGT.L2      4,B16,B0
000914A0 206BA120      [ B0]   BNOP.S1       L19,5
00091564 0003E05A              SUB.L2        B0,1,B0
00091568 2FFFFA10      [ B0]   B.S1          L14
0009156C 303B8120      [!B0]   BNOP.S1       L19,4
00091570            DW$L$_Byte2Word$9$E:
00091570 00000000              NOP
00091574            L15:
00091574 038403E3              MVC.S2        CSR,B7
00091578 0003C05A   ||         SUB.L2        B0,2,B0
0009157C 031FCF5A              AND.L2        -2,B7,B6
00091580 009803A2              MVC.S2        B6,CSR
00091584            L16:
00091584 20011022      [ B0]   BDEC.S2       L17,B0
00091588 02941FDB              OR.L2X        0,A5,B5
0009158C 00200029   ||         MVK.S1        0x4000,A0
00091590 0080A358   ||         MVK.L1        0,A1
00091594 020FF05B              SUB.L2X       A3,1,B4
00091598 008000E9   ||         MVKH.S1       0x10000,A1
0009159C 030C9614   ||         LDBU.D1T1     *A3++[0x4],A6
000915A0            L17:
000915A0 20001023      [ B0]   BDEC.S2       L17,B0
000915A4 02950CA1   ||         SHL.S1        A5,0x8,A5
000915A8 030C6014   ||         LDBU.D1T1     *-A3[0x3],A6
000915AC 80844F01      [ A1]   MPYSU.M1      2,A1,A1
000915B0 0310DDFB   ||         XOR.L2X       B6,A4,B6
000915B4 0214EDF9   ||         XOR.L1        A7,A5,A4
000915B8 93109297   || [!A1]   LDBU.D2T2     *++B4[0x4],B6
000915BC 02990CA1   ||         SHL.S1        A6,0x8,A5
000915C0 038C4014   ||         LDBU.D1T1     *-A3[0x2],A7
000915C4 C0004F01      [ A0]   MPYSU.M1      2,A0,A0
000915C8 D31436F7   || [!A0]   STW.D2T2      B6,*B5++[0x1]
000915CC 02110CA1   ||         SHL.S1        A4,0x8,A4
000915D0 0294CDF9   ||         XOR.L1        A6,A5,A5
000915D4 030C9614   ||         LDBU.D1T1     *A3++[0x4],A6
000915D8            DW$L$_Byte2Word$13$E:
000915D8 020C6015              LDBU.D1T1     *-A3[0x3],A4
000915DC 02950CA0   ||         SHL.S1        A5,0x8,A5
000915E0 0310DDFB              XOR.L2X       B6,A4,B6
000915E4 0294EDF9   ||         XOR.L1        A7,A5,A5
000915E8 02109297   ||         LDBU.D2T2     *++B4[0x4],B4
000915EC 020C4015   ||         LDBU.D1T1     *-A3[0x2],A4
000915F0 02190CA0   ||         SHL.S1        A6,0x8,A4
```

Figure 5.5: A part of assembly code for byte align transfer to 32 bits align. The adjacent parallel lines in the ellipse means which instructions are parallel.

Chapter 6

Conclusion and Future Work

6.1 Conclusion

In this thesis, we first presented bandwidth and CP detection method of the IEEE 802.16e OFDMA TDD system that should be executed at initial network entry, and verified them through Matlab computation. Second, we presented the information element handling techniques that MS should practice at every downlink frame after network entry , and we implemented it on TIs C6416T digital signal processor.

In bandwidth detection, we proposed a method to detect 49 types of bandwidth distribution in 20 MHz range and S-OFDMA system with no CFO. First we compute the mean power of each point at frequency domain in each type. Then, considering 2048D and 23D to compute the distance between the power spectrum of received signal and the computed mean power of each type. Finally, finding the type that result in the smallest distance is the determined bandwidth distribution. Simulation result shows the 2048D method is usually better than 23D method and in SUI-4 channel can get better performance than in SUI-3 channel. But the computational complexity of 23D is much lower than 2048D. The error rate in system "a", "b", and "c" are all less than 0.01 when SNR larger than 4dB at SUI-3.

After detecting bandwidth, the MS has known the bandwidth and FFT size in the system it enters and has to detect the CP. Considering the compute complexity, we use CP complex conjugate correlation to test four possible CP length, and choose the CP that result in the largest correlation. Simulation results show the shorter the original CP length, the larger detection error rate result. And the CP length which is short has higher probability to be error detected.

As MS complete the network entry, it must to read IE every downlink frame, we proposed an information element handling method that has low compute complexity and DSP implementation and optimization of the method employing the Code Composer Studio (CCS)

6.2 Future Work

There are several possible extensions for our research:

- Consider to detect the bandwidth and FFT size that are not S-OFDMA system. That is the Bandwidth is not proportional to FFT size.
- Analyze the performance of the bandwidth detection method. In this thesis assuming the distance normal distributed is not suitable.
- Design bandwidth detection algorithm from network entry point of view for WiMAX MS.
- Try to implement the CP and bandwidth detection method on DSP since we only implement the information element handling in this thesis.

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 and IEEE Std 802.16-2004/Cor1-2005, *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. 28, 2006.

[3] A. S. Florschuetz,*"Method and system of automatic bandwidth detection."* USPTO, Patent no. 6601009, Jul. 29, 2003.

[4] T. B. Brown, D. Val, A. E. Klemets,*"Fast dynamic measurement of bandwidth in a TCP network"* USPTO, Patent no. 7266613, Sep 4, 2007.

[5] R. van Nee and R. Prasad, *OFDM for Wireless Multimedia Communications.* Boston: Artech House, 2000.

[6] S. B. Weinstein and P. M. Ebert, "Data transmission by frequency-division multiplexing using the discrete Fourier transform," *IEEE Trans. Commun. Technol.*, vol. COM-19, pp. 628–634, Oct. 1971.

[7] Y.-C. Liu, "Research in and DSP Implementation of Synchronization Techniques for IEEE 802.16e," M.S. thesis, Department of Electronics Engineering, National Chiao Tung University, Hsinchu, Taiwan, R.O.C., June 2007.

[8] C.-C. Tung, “IEEE 802.16a OFDMA TDD uplink transceiver system integration and optimization on DSP platform,” M.S. thesis, Department of Electronics Engineering, National Chiao Tung University, Hsinchu, Taiwan, R.O.C., June 2005.

[9] J. Puthenkulam, and M. Goldhammer, “802.16 overview and coexistence aspects,” http://grouper.ieee.org/groups/802/secmail/ppt00009.ppt.

[10] V. Bykovnikov, “The advantages of SOFDMA for WiMAX,” http://mail.com.nthu.edu.tw/ jmwu/LAB/SOFDMA-for-WiMAX.pdf.

[11] WiMAX Forum, “Mobile WiMAX — Part 1: A technical overview and performance evalution,” June 2006, http://www.wimaxforum.org/news/downloads/Mobile_WiMAX_Part1_Overview_and _Performance.pdf

[12] H. Yaghoobi, “Scalable OFDMA physical layer in IEEE 802.16 WirelessMAN,” *Intel Technology Journal*, vol. 8, pp. 201–212, Aug 2004.

[13] K.-C. Hung, D. W. Lin, Y.-T. Lee, and K. Loa, “Wireless MAN physical layer specifications: signal processing perspective,” in Yan Zhang and Hsiao-Hwa Chen, eds., *Mobile WiMax: Toward Broadband Wireless Metropolitan Area Networks.* CRC Press, Dec. 2007, ch. 3.

[14] Sundance home page: http://www.sundance.com

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

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

[17] Texas Instruments, *Code Composer Studio User’s Guide.* Lit. no. SPRU328B, Feb. 2000.

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

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

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

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

[22] V. Erceg italic., "Channel models for fixed wireless applications," IEEE 802.16.3c-01/29r4, July 2001.

[23] T. Bhatt, V. Sundaramurthy, J. Zhang and D. McCain, "Initial Synchronization for 802.16e Downlink," *Signals, Systems and Computers, 2006. ACSSC '06. Fortieth Asilomar Conf. on,* pp. 701-706, 2006.

[24] J.-C. Lin, "Maximum-likelihood frame timing instant and frequency offset estimation for OFDM communication over a fast Rayleigh-fading channel," *IEEE Trans. Vehicular Technology,* vol. 52, no. 4, pp. 1049–1062, July 2003.

作者簡歷

學生蔡婉清，民國七十三年四月出生於台灣台北市。民國九十五年六月畢業於國立交通大學電子工程學系，並於同年九月進入國立交通大學電子研究所就讀，從事 OFDMA 通訊系統方面相關研究。民國九十七年七月取得碩士學位，碩士論文題目為『IEEE 802.16e OFDMA 頻寬檢測與資訊單元處理技術之研究』。