國 立 交 通 大 學

電子工程學系 電子研究所碩士班

碩 士 論 文

可程式化閘陣列之快速瑞立衰褪通道模擬器

FPGA Channel Simulator of Fast Rayleigh Fading Channel

研 究 生： 余子瀚

指導教授：魏哲和 博士

中 華 民 國 九 十 三 年 六 月

可程式化閘陣列之快速瑞立衰褪通道模擬器

研究生:余子瀚　　　　　　　　　　指導教授:魏哲和 博士

國立交通大學電子工程系　電子研究所碩士班

摘要

在現今的無線通訊系統中，高速的資料傳輸速率以及可靠的通話品質一直是使用者所最關切的。因此，設計一個系統，並且使其能在不同的環境下正常的操作便成爲工程師們的主要任務。爲確保正常運作，系統必須經過不斷的模擬和測試。然而，要在實際的環境裡做測試與模擬，卻是曠日費時，並且花費高昂。一個比較實際的方法是使用通道模擬器去模擬真實世界裡會發生的各種不同的電波傳遞特性。

我們在論文裡模擬的範例系統是以 **OFDMA** 爲基礎的 **IEEE 802.16a**，特別針對使用者在高速移動的環境下所做的快速瑞立 (**Rayleigh**) 衰褪通道模擬，並且分析其複雜度，比較此通道模擬器所能提供之效能。

在這篇論文裡，我們以 Innovative Integration 公司的 Quixote 板來實現通道模擬器。該板上裝置了一個德州儀器公司(Texas Instruments)出品的 TMS320 C6416 數位訊號處理器，以及一個智霖公司(Xilinx) 所出品的 Virtex II 可程式化閘陣列(**FPGA**)。我們使用 **FPGA** 作爲通道模擬器，並且討論三種不同的三角函數產生方法在通道模擬器中的影響。此外，我們實現兩種單一快速 Rayleigh 衰褪通道模擬器，其通道係數產生速度可超過每秒 100 百萬筆。最後，我們實現一個六條快速 Rayleigh 衰褪通道模擬器，其每條通道每秒可產生 12.5 百萬筆通道係數，可用來當作多重路徑通道或是用於多重輸出輸入(MIMO)系統之通道。

FPGA Channel Generator of Fast Rayleigh Fading Channel

Student: Tzu-Han Yu Advisor: Dr. Che-Ho Wei

Department of Electronics Engineering
Institute of Electronics
National Chiao Tung University

Abstract

In today's wireless communication systems, the very issues which users care about the most are high data transmission rate and reliable quality. So, it becomes the most important task for engineers to make the system work in various environments.

In order to ensure the system performance, it requires simulations and tests. However, field testing for a wireless system is very costly and time consuming. A more practical way is to use a channel simulator to emulate the radio propagation characteristic.

In this thesis, we implement the channel generator with a **DSP-FPGA** embedded card-- "Quixote-III" produced by Innovative Integration. One **DSP** chip, TMS320C6416 of TI, and one **FPGA** chip, Virtex II of Xilinx, are placed on the card. We use FPGA to generate the channel coefficients. We also present three methods to implement the triangular function, which is also the most important component in a channel coefficient generator, and their influences on the speed of the simulator will be discussed and compared. Two single-fast Rayleigh fading channels are implemented, the yield rate of channel coefficient reaches 100 M data per second. Finally a 6-ray fast multipath fading channel is implemented, which can be used to simulate a multipath channel environment, and the channel coefficients yield rate reaches 12.5M per second each channel.

The wireless transmission system we simulate is **IEEE 802.16a**, focus on the fast Rayleigh fading channel for high mobility users. We shall analyze the complexity and make comparison.

致謝

首先，我要感謝我的指導教授魏哲和博士以及林大衛博士。由於他們的細心指導，讓我在兩年的研究生生涯裡，漸漸從一個青澀的大學生，變成了現在能夠獨立思考，並擁有解決問題能力的我。同時在寫作論文的過程中，兩位老師不時的指正我的錯誤，並且指引我往正確的學問之道走去，使我不至於失去方向，更扎實了我做研究的基礎。

實驗室裡的唐之璇學姐，林郁男學長，王士豪學長，吳俊榮學長以及鍾翼州學長也常在我碰上困難的時候給予我指點，讓我不會徬徨無助。另外同級的曾建統同學，李明哲同學，陳盈縈同學，林筱晴同學，蔣宗書同學，黃名彥同學，莊孝強同學，陳俊安同學以及蘇子良同學也都會相互砥礪，研討功課，並且陪我走過這令人懷念的兩年。

最後，我要感謝我的父母親以及我的妹妹，感謝他們一直都在我背後支持我，給予我親情上的支持，讓我可以心無旁騖的完成我的學業。

在此，僅將此篇論文獻給所有幫助過我的人。

Contents

List of Figures

List of Tables

Chapter 1
Introduction

Recent years, with the rapid progress of technology, people start to demand for better quality of life. Among these needs, communication between people is one of the most challenging ones. From the wire-line telecommunication in the 19th century to the latest wireless communication systems in the 21st century, the design constraints and implementation difficulties are getting higher and higher. People ask for high data transmission rate to deliver and receive multimedia information, high mobility support for them to be able to use the service while traveling at high speed.

However, to design and implement such a wireless system requires a lot of simulations and tests. But, field testing for a wireless system is very costly and time consuming, and even though we actually do the field test, the wireless environment changes over time, place and the speed which the subscriber is moving. A better way is to use a channel simulator to emulate the radio propagation characteristic, and simulate the whole system performance.

In this thesis, we will design and implement a channel simulator for a wireless communication system based on **OFDM**, especially focus on high speed moving environment.

The channel simulator will be implemented on a **FPGA** chip. We use a **DSP-FPGA** embedded card as our hardware platform, which contains a digital signal processor (**DSP**) and a field programmable gate array (**FPGA**). (Further details will be introduced in Chapter 4.) We implement two single-fast Rayleigh fading channels and a 6-ray fast Rayleigh fading channel, respectively. We also discuss the triangular function algorithms that influence the performance of channel simulators.

The rest of the thesis will be organized as followed: In Chapter 2, we will introduce the **IEEE 802.16a** specification. And Chapter 3 will introduce our system to be implemented. Then, the hardware information and simulator structure will be discussed in Chapter 4. In this chapter, we also present three methods to implement the triangular function, which is also the most important component in a channel coefficient generator, and their influences on the speed of the simulator will be discussed and compared. The final results and conclusions are given in Chapter 5.

Chapter 2
BWA Communication Standards

2.1 Overview of BWA Communication Standards

BWA (*Broadband wireless access*) is the most challenging segment of the wireless revolution since it has to demonstrate a viable alternative to the cable modem and **DSL** (*Digital Subscriber Lines*) technologies that are strongly entrenched in the last mile access environment. It is also the best way to meet the rising need of rapid multimedia information nowadays [1]. Wireless transmission standards, such as European **DAB** (*digital audio broadcasting*) standard [10], **ETSI** (*European Telecommunications Standard Institute*) **HyperLan/2** [11], **DVB-T** [12], **IEEE 802.11b** [13], **IEEE 802.11g** [14], **IEEE 802.11a** [15], and **IEEE 802.16a** [4] can all be specified as **BWA** communication standards. And we take **IEEE 802.16a** as our example in this thesis.

IEEE 802.16[3] as a standard which specifies the WirelessMan Air Interface for wireless metropolitan networks. It was the first step for IEEE to adapt BWA as a standard specification. IEEE 802.16 addresses the “first mile/last mile” connection in wireless metropolitan are networks and focuses on the efficient use of bandwidth between 10 and 66 GHz and defines a medium access control (**MAC**) layer that supports multiple physical layer specifications customized for frequency band of use [2].

IEEE 802.16a [4] specification enhances the medium access control layer and its operating frequency is between 2 to 11 GHz, mainly used as a fixed point (non-moving) broadband wireless transmission scheme. There are three modes in 802.16a for operators to chose, they are: single carrier mode, **OFDM** mode and **OFDMA** mode (OFDM and OFDMA will be further introduced latter.). And we will take **OFDMA** as our consideration.

As described above, 802.16a mainly supports the stationary transmission, which means that the specification may not be working when users are moving at high speed. So, in this thesis, we try to implement a channel simulator which simulates the channel while users are moving at high speed with a large Doppler shift (up to 2000 Hz) for checking how the mobility rate can 802.16a supports. And this channel simulator can be also used in other systems. For instance, IEEE 802.16's Task Group E is now working on modifying 802.16a to 802.16e which can support high mobility usage. Another group under IEEE PAR 802.20 is also seeking for a novel BWA scheme to support high mobility, too. This channel simulator could be used in these systems as well.

2.2 Overview of OFDM and OFDMA [5]

OFDM (*Orthogonal Frequency Division Multiplexing*) is usually viewed as a transmission scheme which can be applied in the wireless environment and in the wireline communication system, also.

The history of **OFDM** could be traced back to the 60's, when Chang published his paper on the synthesis of bandlimited signals for multichannel transmission [6]. Chang presented a scheme of transmitting signals simultaneously through a linear bandlimited channel without **ICI (***interchannel interference***)** and **ISI (***intersymbol interference***).** Later on, Salzberg performed an analysis of the performance [7], where he concluded that "the strategy of design an efficient parallel system should concentrate more on reducing crosstalk between adjacent channel than on perfecting the individual channels themselves, since the distortions due to crosstalk tend to dominate".

In 1971, Weinstein and Ebert made a major contribution to **OFDM**, they used **DFT** (*Discrete Fourier Transform*) to perform baseband signal modulation and demodulation [8]. They also added a guard space between the symbols and raised-cosine windowing in the time domain to avoid **ISI** and **ICI**. Another important contribution was due to Peled and Ruiz in 1980 [9]. They came up with the idea of **CP** (*cyclic prefix*) solving the orthogonality problem. Instead of using an empty guard space, they filled the guard space with cyclic extension of the **OFDM** symbol. By doing this, when the **OFDM** signals are transmitted through the channel, effectively simulates a channel performing cyclic convolution, and implies orthogonality over dispersive channels when **CP** is longer than the impulse response of the channel.

Recent years, another **OFDM** based technique**—OFDMA** *(Orthogonal Frequency Division Multiple Access)* has been modified from **OFDM** to get a better bandwidth share idea for multiple users. In **OFDMA**, the carriers are divided into groups which are called subchannels for multiple users to use within one single OFDMA symbol while in **OFDM**, one user is allowed only.

OFDM and **OFDMA** are now widely used in many wireless or wireline transmission standards, such as European **DAB** (*digital audio broadcasting*) standard [10], **ETSI** (*European Telecommunications Standard Institute*) **HyperLan/2**[11], **DVB-T** [12], **IEEE 802.11b** [13], **IEEE 802.11g** [14], **IEEE 802.11a** [15], and **IEEE 802.16a.**

2.2.1 OFDM

The basic idea of **OFDM** is to divide the available spectrum into several subchannels. By making all subchannels narrowband, they experience flat fading, which makes equalization very simple. To obtain a high spectral efficiency the frequency response of the subchannels are overlapping and orthogonal. The orthogonality can be maintained, even though the signal passes through a time-dispersive channel, by introducing a **CP** (*cyclic prefix*). An **OFDM** symbol is shown in Figure 2.1

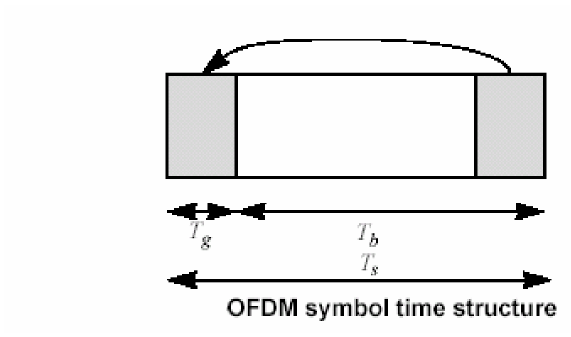


Fig 2.1: The CP is a short copy of the last part of the OFDM symbol

where Tg represents the time length of **CP**, which is a short copy of the last part of the OFDM symbol, and Tb represents the data interval. By introducing **CP**, we can avoid the occurrence of **ISI** and **ICI** at the cost of losses in **SNR (***Signal to Noise Ratio***)** and throughput will be introduced.

A basic **OFDM** system diagram is given in Figure 2.2

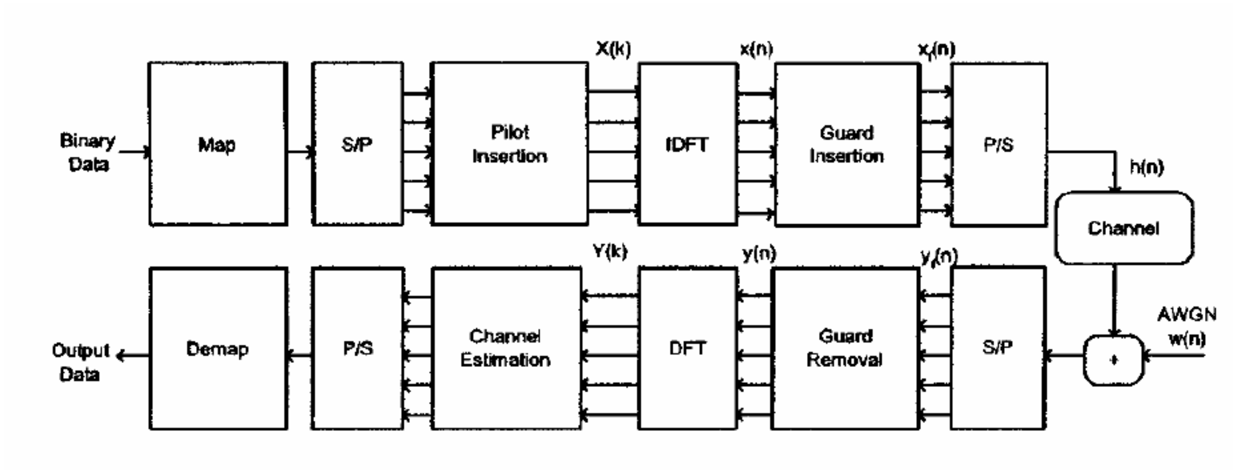


Fig 2.2: A basic view of a digital baseband **OFDM** system

From Fig 2.2, we can see how an **OFDM** symbol is formed, transmitted and received. First of all, binary data from the source are mapped (may be in **BPSK**, **QAM, 16-QAM,** or **64-QAM** form), then separated from serial sequence to parallel sequence. After that, we insert pilot carriers which will be used when doing channel estimation, then, perform **IDFT** (*Inverse Discrete Fourier Transform*). Finally, we add **CP**, an **OFDM** symbol is formed. Through the channel, the signal will be transmitted to the receiver and demodulated with the counter steps as what had been done in the transmitter end. Thus, an **OFDM** transmission process is completed.

The continuous-time **OFDM** system has always considered as the ideal OFDM system. However, in most of the case, we implement such a system in the discrete-time form. But, we still like to introduce **OFDM** from the viewpoint of continuous-time **OFDM** system.

Consider an **OFDM** system consisting of N subcarriers with a bandwidth of W Hz. The symbol time is T seconds long, composed of Ts and Tg, where Ts= N/W seconds which represents the real data interval, and Tg seconds is the time interval of **CP.** The transmitted signal for the kth subcarrier could be expressed as

$$g_k(t) = \begin{cases} \frac{1}{\sqrt{T_s}} e^{j2\pi \frac{W}{n} k(t-T_g)} & \text{if } \; t \in [0, T\,]; \\ 0 & \text{otherwise ;} \end{cases} \qquad (2.1)$$

where T=Ts +Tg . Note that $g_k(t) = g_k(t+T_s)$ when t is within the guard interval [0, Tg]. We also notice that $g_k(t)$ is a rectangular pulse modulated by the carrier frequency K$\frac{W}{N}$. The transmitted baseband signal s(t) for the **OFDM** symbol l can be obtained by summing all modulated subchannel signals as

$$S_l(t) = \sum_{k=0}^{N-1} x_{k,l}\, g_k(t - lT) \qquad (2.2)$$

where $x_{0,l}, x_{1,l}, \ldots\ldots, x_{n-1,l}$ are complex numbers from a set of signal constellation points, which represent the transmitted data. From (2.1) and (2.2), we can see that $s_l(t) = 0$ for $t \notin [lT, (l+1)T]$. This way, the baseband signal s(t) can be expressed as:

$$s(t) = \sum_{t=-\infty}^{\infty} s_l(t) = \sum_{t=-\infty}^{\infty} \sum_{k=0}^{N-1} x_{k,l}\, g_k(t - lT) \qquad (2.3)$$

We assume that the impulse response $h(\tau;t)$ is shorter than the length of **CP**, then, the received signal $r(t)$ will be

$$r(t) = (h * s)(t) + n(t)$$

$$= \int_0^{T_g} (h;t)\, s(t-\tau)\, d\tau + n(t), \tag{2.4}$$

where n(t) is additive, white, and complex **Gaussian noise**.

The **OFDM** receiver consists of a filter bank, matched to the last part [Tg, T] of the transmitted waveform $g_k(t)$, that is,

$$u_k = \begin{cases} g_k^*(T-t) & t \in [0, T-T_g] \\ 0 & \text{otherwise ;} \end{cases} \tag{2.5}$$

It means that **CP** is removed in the received signal, and the signal is clean without **ISI**. Now we can omit the time index *l* when calculating the sampled output at the kth matched filter. Through (2.3), (2.4) and (2.5), we find that

$$y_k = (\tau * u_k)(t)\,|_{t=T}$$

$$= \int_{-\infty}^{\infty} r(t) u_k(T-t)\, dt$$

$$= \int_{T_g}^{T} \left(\int_0^{T_g} h(\tau;t)\, s(t-\tau)\, d\tau + n(t) \right) g_k^*(t)\, dt$$

$$= \int_{T_g}^{T} \left(\int_0^{Tg} h(\tau;t) \left[\sum_{k'=0}^{N-1} x_{k'} g_{k'}(t-\tau) \right] dt \right) g_k^*(t)\, dt + \int_{T_g}^{T} n(t)\, g_k^*(t) dt \tag{2.6}$$

Assuming that the channel response is static over one **OFDM** symbol interval and is denoted as $h(\tau)$, then

$$y_k = \sum_{k'=0}^{N-1} x_{k'} \int_{T_g}^{T} \left(\int_0^{T_g} h(\tau) g_{k'}(t-\tau) \right) g_k^*(t) dt + \int_{T_g}^{T} n(t) g_k^*(t)\, dt \tag{2.7}$$

From (2.7), we can see that $T_g < t < T$ and $0 < \tau < T$, then $0 < t-\tau < T$. Therefore, by substituting (2.1) to (2.7), the inner integral of (2.7) becomes

$$\int_0^{T_g} h(\tau) g_{k'}(t-\tau)\, d\tau = \int_0^{T_g} h(\tau) \frac{e^{j2\pi k'(t-\tau-T_g)w/n}}{\sqrt{T-T_g}}\, d\tau$$

$$= \frac{e^{j2\pi k'(t-T_g)W/N}}{\sqrt{T-T_g}} \int_0^{T_g} h(\tau) e^{-j2\pi k'\tau W/N}\, d\tau, \quad T_g < t < T \tag{2.8}$$

Furthermore, the integral in (2.8) is the sampled frequency response of the channel at the kth subcarrier frequency f=k'W/N:

$$h_{k'} = H\left(K'\frac{W}{N}\right) = \int_0^{T_g} h(\tau)e^{-j2\pi k'\tau W/N}d\tau, \qquad (2.9)$$

where H(f) is the **Fourier transform** of $h(\tau)$. The output from the receiver filter bank can be rewritten as

$$y_k = \sum_{k'=0}^{N-1} x_{k'}\int_{T_g}^{T}\frac{e^{j2\pi k'\tau W/N}}{\sqrt{T-T_g}}h_{k'}g_k^*(t)\,dt + \int_{T_g}^{T} n(t)g_k^*(t)\,dt$$

$$= \sum_{k'=0}^{N-1} x_{k'}h_{k'}\int_{T_g}^{T} g_{k'}(t)g_k^*(t)\,dt + n_k \qquad (2.10)$$

where $n_k = \int_{T_g}^{T} n(t)g_k^*(t)\,dt$. Since the transmitter filters $g_k(t)$ are orthogonal,

$$\int_{T_g}^{T} g_{k'}(t)g_k^*(t) = \int_{T_g}^{T}\frac{e^{j2\pi k'(t-T_g)W/N}}{\sqrt{T-T_g}}\frac{e^{-j2\pi k(t-T_g)W/N}}{\sqrt{T-T_g}}dt = \delta[k-k'], \qquad (2.11)$$

where $\delta[k]$ is the **Kronecker delta function** [16]. Now, we can simplify (2.10) as

$$y_k = h_k x_k + n_k \qquad (2.12)$$

where n_k is additive white Gaussian noise.

By re-introducing time index *l*, we may now view the **OFDM** system as a set of parallel channels, just as Figure 2.3.

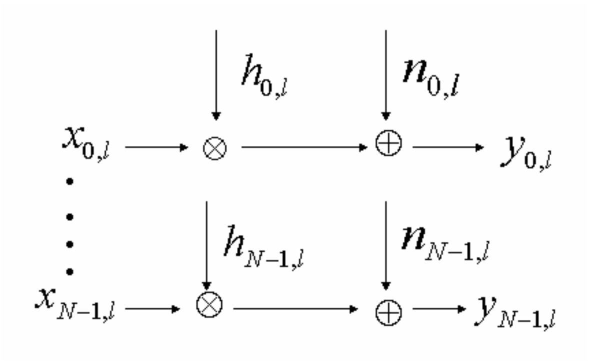


Fig 2.3: **OFDM** system as a set of parallel channels

2.2.2 OFDMA

The basic idea of **OFDMA** is an **OFDM** based frequency division multiple access. The main structure and transmission technique are very alike. But, in **OFDMA**, the carriers are divided into groups which are called subchannels, and the number of subcarriers for each subchannel depends on the system design. Users are allowed to use one or more subchannels on their demand, which means that OFDMA support multiple users within one OFDMA symbol. While in OFDM structure, only one single user is allowed. So, by doing subchannelizing the symbol in the frequency domain, the bandwidth can be allocated dynamically and efficiently. That is the major difference between OFDM and OFDMA.

Due to the large variance of transmission path loss in the wireless environment, the intercell interference is one of the impairments in wireless communication. In OFDMA system, the subchannels can be composed from several distinct permutations of subcarrirers, and this enables significant reduction in the intercell interference, if the system is not fully loaded. This is another advantage for OFDMA.

As what had been described above, OFDMA is more complex than OFDM since it supports multiple users, and introduces the idea of subchannels, which make it even harder for implementation. We make a further description in the next part, how IEEE 802.16a formulate the OFDMA structure and make it work perfectly.

2.3 Physical Layer of IEEE 802.16a OFDMA Mode

In IEEE 802.16 specification, it formulates wide range of aspects in the whole system, but we only underline some of the **PHY** (*Physical Layer*) for the **OFDMA** mode in this thesis, since **PHY** concerns the channel simulator the most. Other detailed information could be found in the 802.16a specification [4].

2.3.1 Carrier Allocation for Downlink

The FFT size in the 802.16a **OFDMA** mode for downlink is 2048, which means that there are 2048 subcarriers within a channel. Among these subcarriers, they are mainly divided into tree groups: data carriers used to carry the information, pilot carriers used for channel response estimation and various synchronization purposes, and null carriers, including guard bands and DC carrier, carrying no useful information at all.

The frequency description is shown in Figure 2.4.

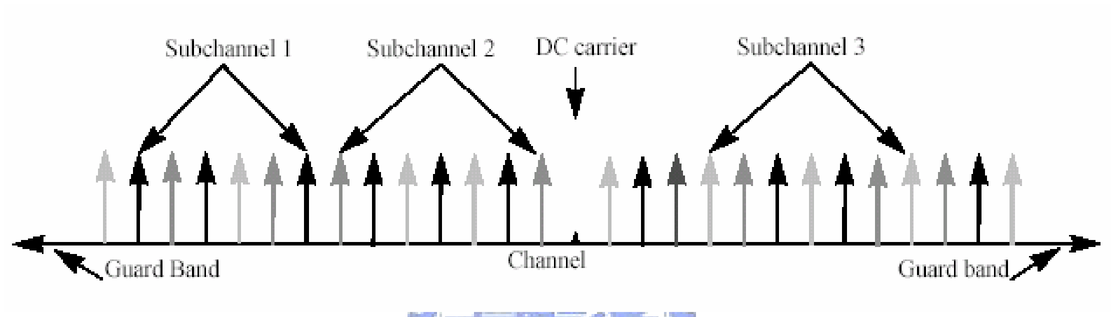
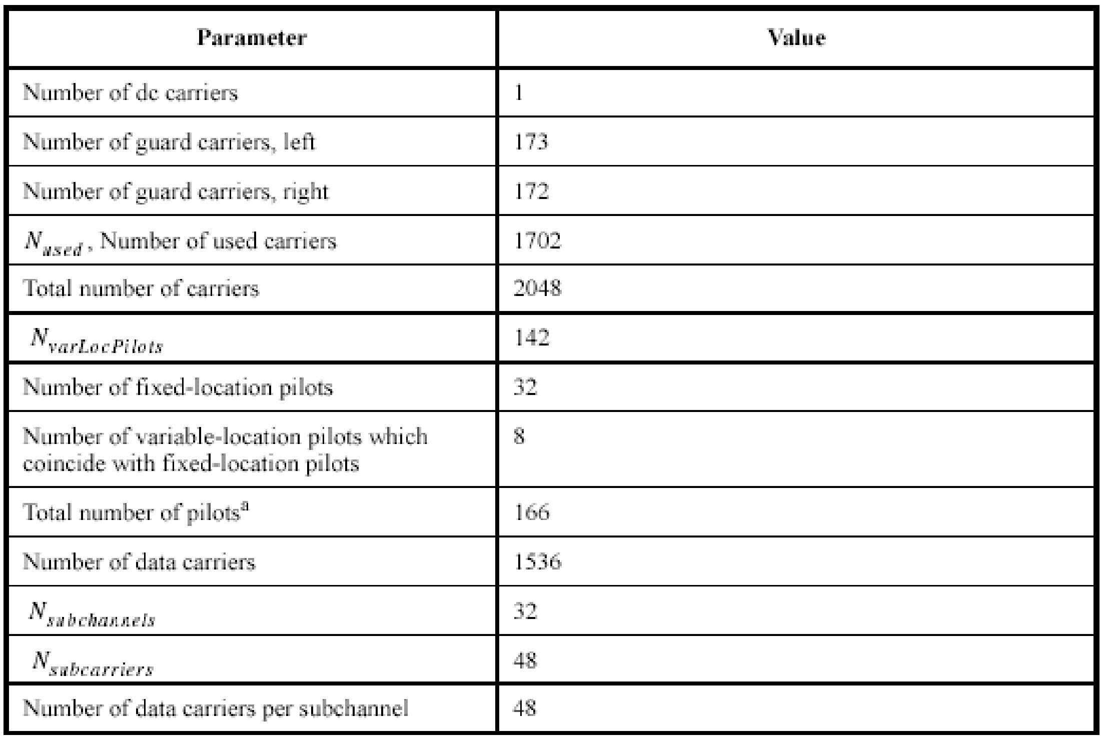


Fig. 2.4: OFDMA frequency description (3 subchannel example)

As we can see in Figure 2.4, among those 2048 subcarriers, there are 1702 used subcarriers, composed of 1536 data carriers and 166 pilot carriers. The rest 346 subcarriers are unused subcarriers as the guard band distributed on the edges of the symbol, and one DC carrier right in the middle of the OFDMA symbol. In, IEEE 802.16a those 1702 used subcarriers are divided into 32 subchannels, each subchaneel consists of 48 data carriers and 2 fixed pilot carriers and 3 variable location pilot carriers. The pilot location changing rule follows some permutation formula which will be described latter. The carrier allocation for OFDMA in 802.16a is listed in Table 2.1.



Parameter	Value
Number of dc carriers	1
Number of guard carriers, left	173
Number of guard carriers, right	172
N_{used}, Number of used carriers	1702
Total number of carriers	2048
$N_{varLocPilots}$	142
Number of fixed-location pilots	32
Number of variable-location pilots which coincide with fixed-location pilots	8
Total number of pilots[a]	166
Number of data carriers	1536
$N_{subchannels}$	32
$N_{subcarriers}$	48
Number of data carriers per subchannel	48

Table 2.1: OFDMA carrier allocation for downlink

As what had mentioned above, there are variable location pilot carriers and fixed location pilot carriers in the downlink OFDMA symbol. The carrier indices of the fixed location pilots never change. The variable location pilots shift their location every symbol, repeating every 4 symbols, according to the formula:

$$\mathrm{var}\,LocPilot = 3L + 12P_k \qquad L \in 0,1,2,3 \qquad P_k \in \{0,1,2......141\} \quad (2.13)$$

where $\mathrm{var}\,LocPilot$ is the carrier index of variable location pilot . The detailed illustration is given in Figure 2.5

After mapping the pilot carriers, the remainders of the used carriers are the data subchannels. Note that since the variable location pilots change location in each symbol, repeating every fourth symbol, the locations of the carriers in the data shall change also.

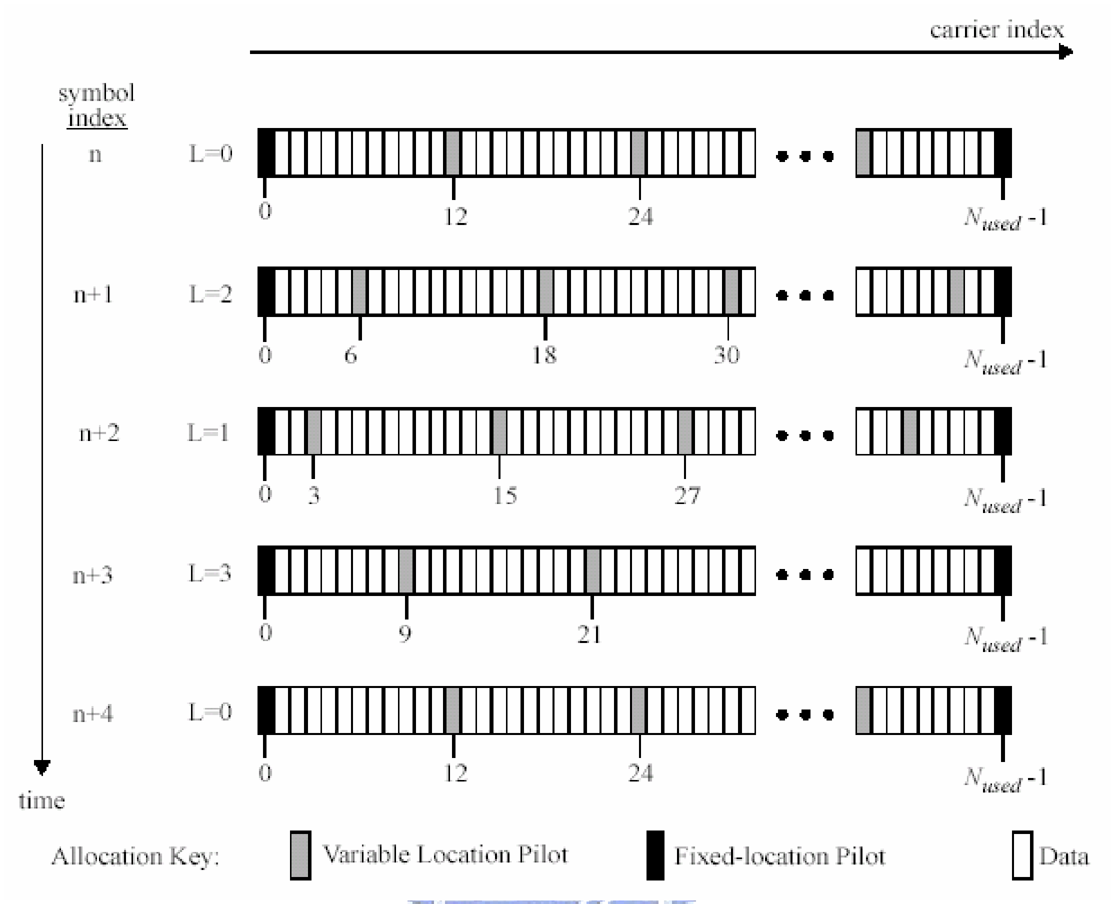


Fig 2.5: Carrier allocation in The OFDMA DL

The exact partitioning into subchennels is done according to (2.14) [1], called a permutation formula.

$$carrier(n,s) = N_{subchannels} \bullet n + \left\{ p_s\left[n_{\mathrm{mod}(N_{subchannels})} \right] + ID_{cell} \bullet ceil\left[(n+1)/N_{subchannels} \right] \right\}_{\mathrm{mod}(N_{subchannels})} \tag{2.14}$$

where

$carrier(n,s)$ is the carrier index of carrier n in subchannels

s is the index number of a subchannel, from the set $[0,1......N_{subchannels}-1]$,

n is the carrier-in-subchannel index from the set $[0,1......N_{subcarries}-1]$,

$N_{subchannels}$ is the number of subchannels,

$p_s[j]$ is the series obtained by rotating{PermutationBase}cyclically to the left s times,

$ceil[\ ]$ is the function that rounds its argument up to the next integer,

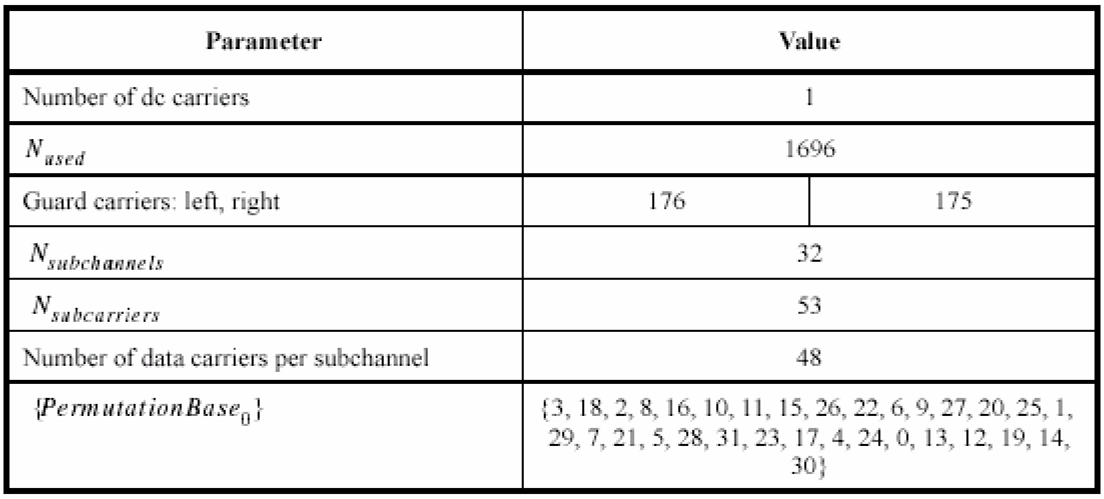
ID_{cell} is a positive integer assigned by MAC to identify this particular BS,

$X_{\mathrm{mod}(k)}$ is the remainder of quotient X/k (which is almost k-1),

2.3.2 Carrier Allocation for Uplink

In uplink OFDMA symbol, the FFT size remains 2048. The subcarriers are also divided into data carriers , pilot carriers and unused carriers, just as in downlink. The allocation of subcarriers is slightly different from downlink. There are 1696 used subcarriers, composed of 1536 data carriers and 160 pilot carriers. The rest 352 subcarriers are unused subcarriers as the guard band distributed on the edges of the symbol, and one DC carrier right in the middle of the OFDMA symbol. In IEEE 802.16a, those1696 used subcarriers are divided into 32 subchannels, each subchaneel consists of 48 data carriers and 1 fixed pilot carriers and 4 variable location pilot carriers. The carrier allocation for OFDMA in 802.16a is listed in Table 2.2.

The fix location pilot location is always at 26 in the subchannel. The variable location pilots change locations in each symbol, repeating every 13 symbols, according to $L_k =$ 0, 2, 4, 6, 8, 10, 12, 1, 3, 5, 7, 9, 11, where k=0 to 12. For k=0 the variable location pilots are placed at 0, 13, 27, 40. For other k values, the locations change by adding L_k to the index. The subchanel division scheme in uplink also obeys (2.14). And the detailed carrier illustration is given in Figure 2.6.



Parameter	Value	
Number of dc carriers	1	
N_{used}	1696	
Guard carriers: left, right	176	175
$N_{subchannels}$	32	
$N_{subcarriers}$	53	
Number of data carriers per subchannel	48	
$\{PermutationBase_0\}$	{3, 18, 2, 8, 16, 10, 11, 15, 26, 22, 6, 9, 27, 20, 25, 1, 29, 7, 21, 5, 28, 31, 23, 17, 4, 24, 0, 13, 12, 19, 14, 30}	

Table 2.2: OFDMA carrier allocation for Uplink

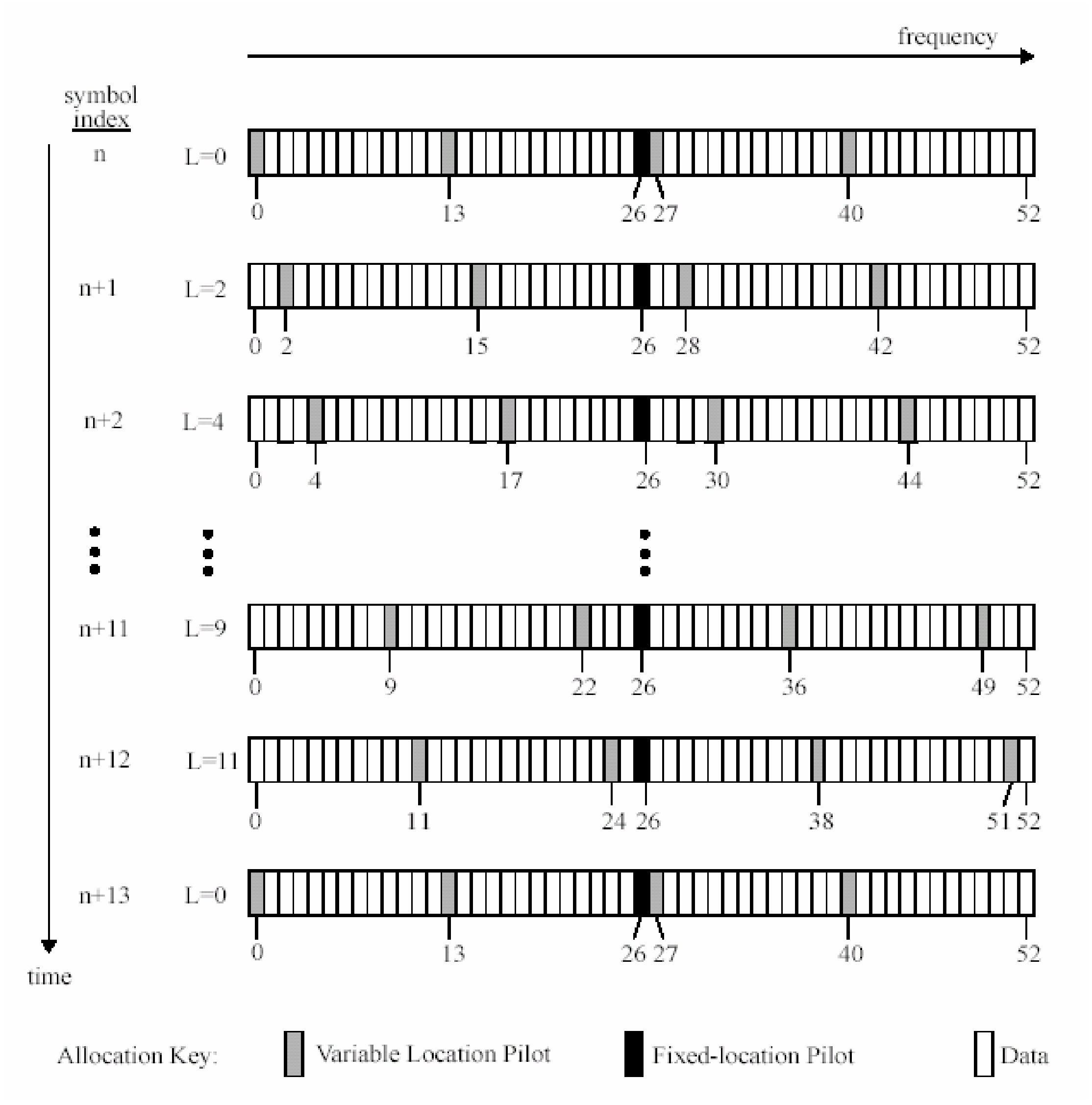


Fig 2.6: Carrier allocation in the OFDMA Uplink

2.3.3 OFDMA Frame Structure for TDD

IEEE 802.16a is designed to operate in the frequency band between 2 to 11GHz. If the system is operating in licensed bands, the duplexing method shall be either **FDD** *(Frequency Division Duplexing)* or **TDD** *(Time Division Duplexing)*. But when the system is operating in frequency-exempt band, the duplexing method must be **TDD**. We consider the **TDD** mode in this thesis. For the traffics of the downlink users and the uplink users are usually quite asymmetric. **TDD** permits efficient allocating the available resources, since uplink transmissions are usually much less then the downlink transmissions.

Figure 2.7 is the frame structure of **TDD OFDMA.** In order to fit the **FEC (**Forward Error Coding**)** block, each burst transmission consists of multiples of three **OFDMA** symbols. In each frame, one **DL** subframe and one **UL** subframe are contained. Each subframe contains numbers of bursts. In each frame, the **TTG (***Tx/Rx transition gap***)** and Rx/Tx transition gap (**RTG**) shall be inserted between the downlink and uplink at the end of each frame respectively to allow **BS (***Base Station***)** to turn around. **TTG** and **RTG** shall be at least 5us and integer multiple of the PS in duration, and start on a PS boundary. After the TTG, the BS receiver shall look for the first symbols of a UL burst. After **RTG**, the **SS** (*Subscriber Station*) receiver shall look for the first symbols of **QPSK** modulated data in the DL burst. The duration of a frame is limited from 2 ms to 20 ms and is specified by the frame duration code.

In the beginning of the frame, there are some important information for downlink and uplink. As we know, the control messages and the system parameters are very important for the **BS** and the **SS**. So, the first burst contains the parameters such as modulation type, forward error correction type, preamble length, guard time, etc. The first **FEC** block of each frame is the DL_Frame_Prefix. It contains the parameters of **FCH (***Frame Control Header***)** which includes **DL-MAPs** and **UL-MAPs** and other information. DL-Maps and UL-Maps messages define the access to the downlink and uplink information, including the burst profiles and allocation in the subchannel and time axes of the bursts.

After a short **TTG,** all pilot symbols called preamble will be inserted by the SS to help the channels estimation process in the receiver, since for uplink users, the subcarriers spread out in the whole channel, it will be very difficult to BS to make a precise channel estimation with only 5 pilots in a subchannel.

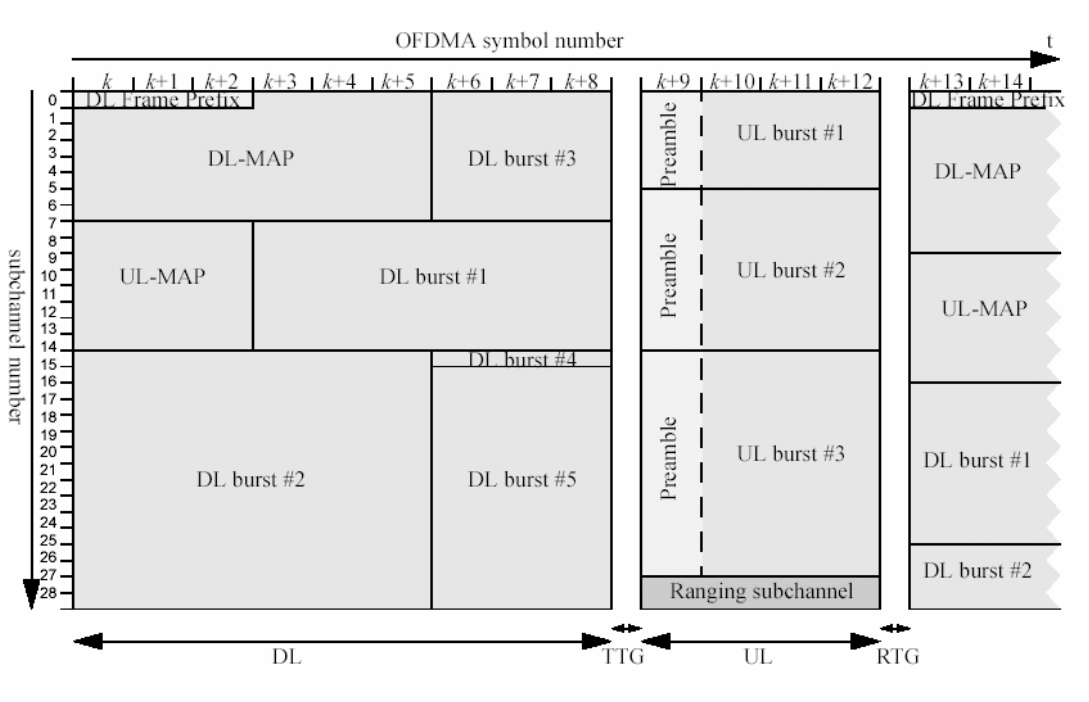
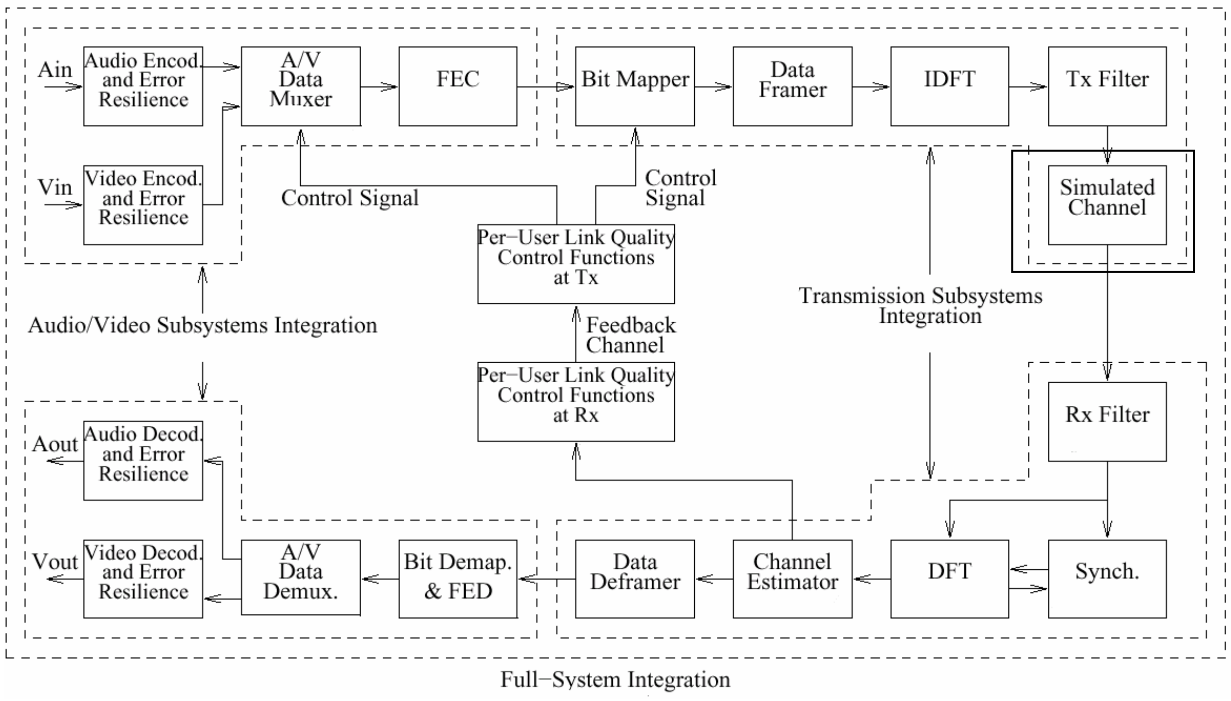


Fig 2.7: Time plan for one TDD time frame

Above, we briefly introduce the technique background ,operating environment , physical layer parameters and structures of IEEE 802.16a in **OFDMA** mode, other detailed information, such as **MAC** (*Medium Access Control*) layer, channel coding, interleaving scheme in **PHY,** synchronization problems, channels estimation difficulties, etc. are not discussed in this thesis, since the carrier allocation and frame structure concern channel simulator the most. Those skipped information could be found in IEEE 802.16a [4].

Chapter 3
System Overview

After the brief introduction to **IEEE 802.16a** in **OFDMA** mode, we would like to introduce our system which follows the specification.



Ain
Audio Encod. and Error Resilience
A/V Data Muxer
FEC
Bit Mapper
Data Framer
IDFT
Tx Filter
Vin
Video Encod. and Error Resilience
Control Signal
Control Signal
Simulated Channel
Per−User Link Quality Control Functions at Tx
Transmission Subsystems Integration
Audio/Video Subsystems Integration
Feedback Channel
Per−User Link Quality Control Functions at Rx
Rx Filter
Aout
Audio Decod. and Error Resilience
Vout
Video Decod. and Error Resilience
A/V Data Demux.
Bit Demap. & FED
Data Deframer
Channel Estimator
DFT
Synch.
Full−System Integration

Fig 3.1: The system block diagram.

Figure 3.1 shows a wireless transmission system in detail. The channel simulator that we are going to implement is shaded in the diagram for the simulation. The main task for a channel simulator can be expressed in Figure 3.2.

The channel simulator generates channel coefficients following some specific channel model, then convolutes the channel coefficients with the transmitted signals s(t) from Tx, and then adds the noise item. Finally, it sends the resulting signals y(t) to the Rx end. We can also express it by an equation:

$$y(t) = s(t) * h(t) + n(t) \tag{3.1}$$

where $h(t)$ denotes the channel coefficients which is our main task for the channel simulator, $n(t)$ is the noise item, and"$*$" represents the convolution operation in time domain.

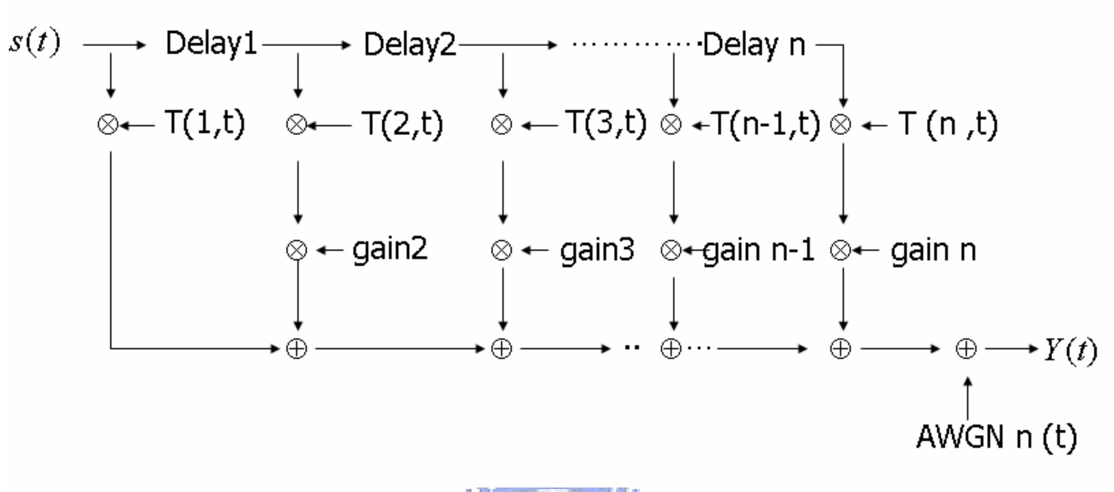


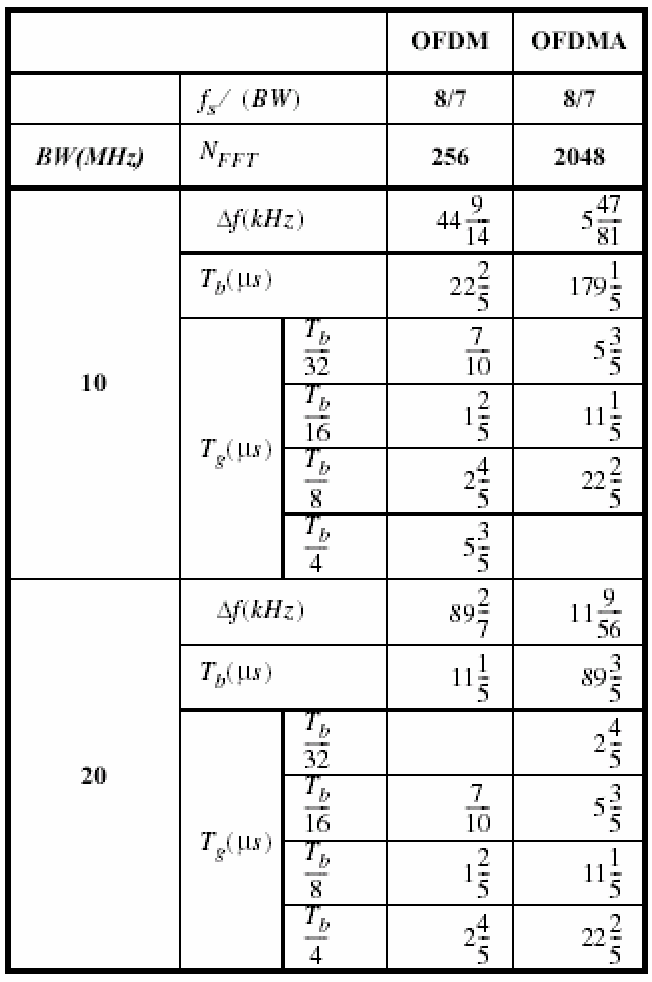
Fig 3.2: Main task of Channel simulator

In order to generate the proper channel coefficients to simulate the electro-magnetic wave propagation, we have to clearly understand the system channel parameters and choose a suitable channel model for the simulator.

3.1 Channel Parameters

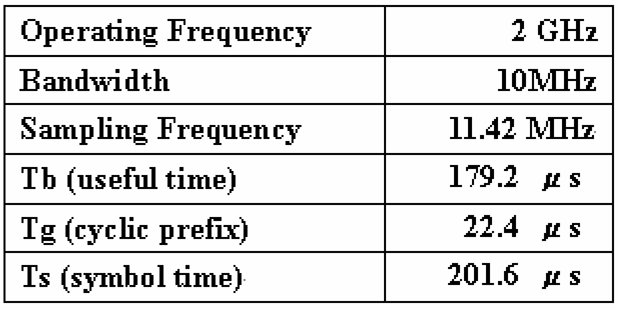
IEEE 802.16a supports the wireless access operating at frequencies 2-11 GHz. In this thesis, we assume that our system operates at 2 GHz band. From the Annex B in **IEEE 802.16a**, our system channel parameters are defined by the specification, which is shown in Table 3.1.

In Table 3.1, we can find that there are two bandwidths for us to choose, and we take 10 MHz in our study. The ratio of **CP** time to the useful time is supported as 1/32, 1/16, 1/8 and 1/4, and we pick 1/8. The number of **FFT (***Fast Fourier Transform***)** and the useful time **Tb** for **OFDMA** are defined as 2048 and 179.2 us, so the **CP** time, **Tg**, is 22.4 us. The sampling frequency is defined as (**BW** • 8/7), where **BW** is the bandwidth. The **OFDMA** symbol parameters and system channel parameters are now all determined and listed in Table 3.2. And Figure 3.3 shows a schematic picture of an **OFDMA** symbol in our system.



			OFDM	OFDMA
	$f_s/(BW)$		8/7	8/7
$BW(MHz)$	N_{FFT}		256	2048
10	$\Delta f(kHz)$		$44\frac{9}{14}$	$5\frac{47}{81}$
	$T_b(\mu s)$		$22\frac{2}{5}$	$179\frac{1}{5}$
	$T_g(\mu s)$	$\frac{T_b}{32}$	$\frac{7}{10}$	$5\frac{3}{5}$
		$\frac{T_b}{16}$	$1\frac{2}{5}$	$11\frac{1}{5}$
		$\frac{T_b}{8}$	$2\frac{4}{5}$	$22\frac{2}{5}$
		$\frac{T_b}{4}$	$5\frac{3}{5}$	
20	$\Delta f(kHz)$		$89\frac{2}{7}$	$11\frac{9}{56}$
	$T_b(\mu s)$		$11\frac{1}{5}$	$89\frac{3}{5}$
	$T_g(\mu s)$	$\frac{T_b}{32}$		$2\frac{4}{5}$
		$\frac{T_b}{16}$	$\frac{7}{10}$	$5\frac{3}{5}$
		$\frac{T_b}{8}$	$1\frac{2}{5}$	$11\frac{1}{5}$
		$\frac{T_b}{4}$	$2\frac{4}{5}$	$22\frac{2}{5}$

Table 3.l: OFDMA Channelization Parameters For License-exempt Bands [3].



Operating Frequency	2 GHz
Bandwidth	10MHz
Sampling Frequency	11.42 MHz
Tb (useful time)	179.2 μs
Tg (cyclic prefix)	22.4 μs
Ts (symbol time)	201.6 μs

Table 3.2: Channel Parameters and OFDMA Symbol Parameters

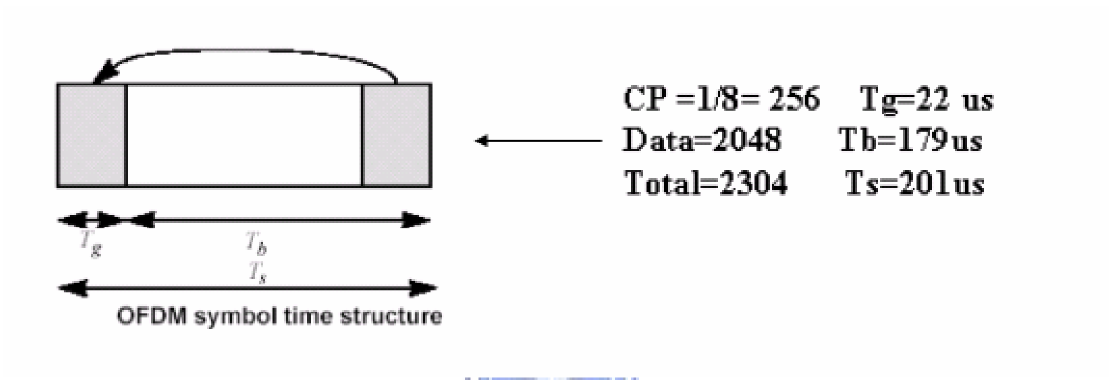


Fig 3.3: An **OFDMA** symbol in our system

3.2 Channel model[18]

After the channel parameters are determined, we now have to pick a suitable channel model for our system simulation.

Modeling the radio channel has been considered as one of the most difficult parts in wireless communication system design. The transmission paths between transmitter and receiver may vary with different terrains, buildings, the speed that mobile terminals move, the season factors, even the wind speed. [17]

The radio wave propagation effects are mainly divided into two parts. One is shadow fading, and the other is multipath fading. Shadow fading basically involves the electro-magnetic wave reflection, diffraction, scattering and attenuation, while small multipath fading involves multipath effects essentially. In most of the cases, multipath fading is more critical than large-scale fading. Multipath effects generally affect greatly the on the channel response estimation and synchronization of the system. Thus, we consider the multipath fading in the simulation only.

With multipath fading effects, the signal strength may changes within a short period of time, and the varying Doppler shifts on those different paths may undergo the random frequency modulation.

The cause of multipath fading is due to the interference between two or more versions of the transmitted signal which arrive at the receiver at the different time. Many factors influence the multipath fading, including multipath propagation, speed of the mobile, surrounding constructions, and transmission bandwidth as well. The constructive and destructive combinations cause fluctuations in the signal strength.

Moreover, the relative motion between the base station and the mobile terminal results in random frequency modulation due to different Doppler shifts on each if the multipath components. Doppler shifts may be positive or negative depending on whether the mobile receiver is moving toward or away from the base station. The Doppler shift f_d is given by

$$f_d = \frac{v}{\lambda}\cos\phi \quad \text{(Hz)} \tag{3. 2}$$

where v is the mobile speed (in meter/second), λ is the wavelength of the transmitted signal (in meter), and ϕ is the spatial angle between the direction of the mobile terminal and the direction of arrival of the wave.

There are also different types of multipath fading. Depending on the relation between the signal parameters (such as bandwidth, symbol period, etc.) and channel parameters (such as R.M.S delay spread and Doppler spread), different transmitted signals will undergo different types of fading. The time dispersion and frequency dispersion mechanisms lead multipath fading to four types. Figure 3.4 shows a tree of the four types of fading.

In flat fading, the multipath structure of the channel is such that the spectral characteristics of the transmitted signal are preserved at the receiver. However the strength of the received signals changes with time. On the other hand, in frequency selective fading, the channel impulse response has a multipath delay spread which is greater than the reciprocal bandwidth of the transmitted waveform. Thus, the channel induces **ISI**, and signal will distort.

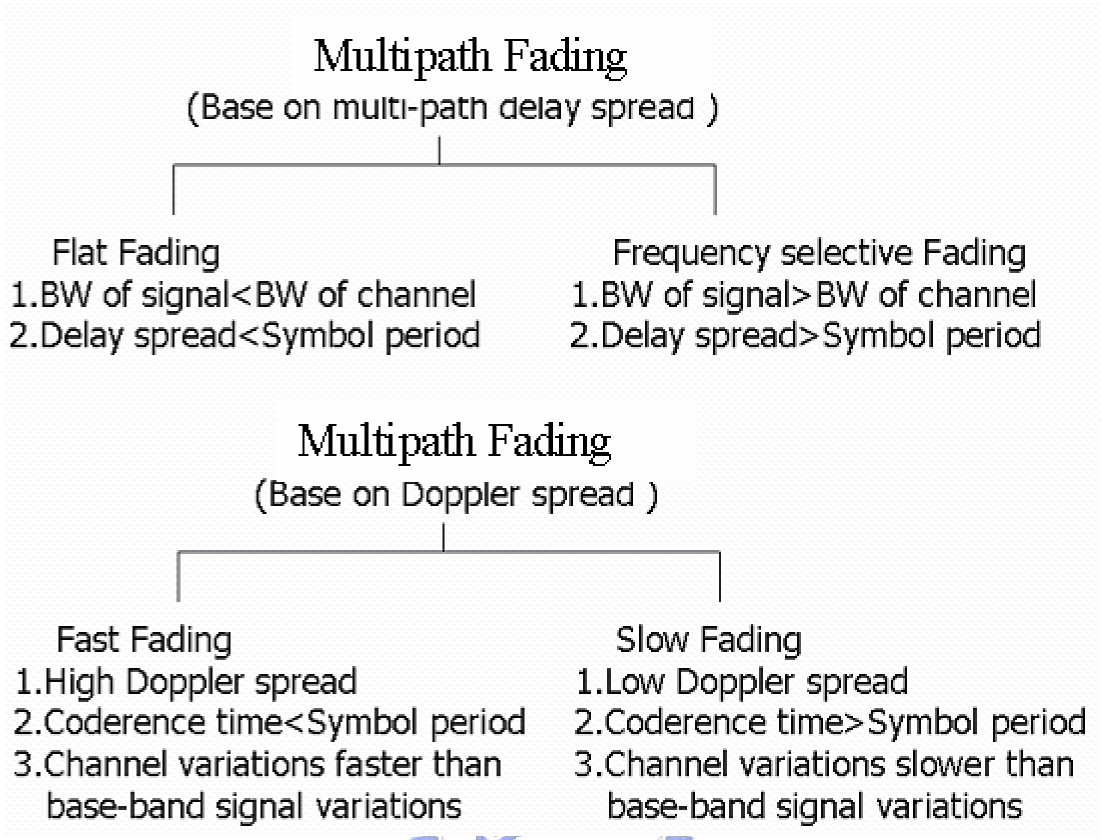

Multipath Fading
(Base on multi-path delay spread)
Flat Fading
1.BW of signal<BW of channel
2.Delay spread<Symbol period
Frequency selective Fading
1.BW of signal>BW of channel
2.Delay spread>Symbol period
Multipath Fading
(Base on Doppler spread)
Fast Fading
1.High Doppler spread
2.Coderence time<Symbol period
3.Channel variations faster than base-band signal variations
Slow Fading
1.Low Doppler spread
2.Coderence time>Symbol period
3.Channel variations slower than base-band signal variations

Fig 3.4: Types of multipath fading [37]

Depending on how rapidly the transmitted baseband signal changes as compared to the rate of change of the channel, a channel may be classified either as a fast or slow fading channel. In fast fading channel, the channel impulse response changes rapidly within the symbol duration, while in slow fading channel, the channel impulse response changes at a rate which is much slower than the transmitted signal.

Another important factor in the channel model is the noise item. There are untold random noise sources from the surrounding in an actual environment .Those noise sources may be from the atmosphere, the sun, cosmos, or manmade sources. We usually model the noise as *additive white Gaussian noise* (**AWGN**) with zero mean and constant variance.

A channel model consists of **AWGN**, channel multipath delay spread, and channel coefficients, and when all of them are determined, we can start to simulate our channel model.

3.2.1 Delay profile

A delay profile contains the information of channel multipath number, average power of each path ,etc. The delay profile changes over the terrestrial constraints and the moving speed. **ETSI** (*European Telecommunications Standard Institute*) provides us a suitable delay profile for our channel model in ESTI *TR 101112V.3.2.0* [19]. For each terrestrial test environment, a channel impulse response model based on a tapped-delay line model is given. The model is characterized by the number of taps, the time delay relative to the first tap, the average power relative to the strongest tap, and the Doppler spectrum of each tap. A majority of the time, r.m.s delay spreads are relatively small, but occasionally, there are "worst case" multipath characteristics that lead to much larger r.m.s delay spreads. Measurements in outdoor environments show that the r.m.s delay spred can vary over an oder magnitude, within the same environment. Although large delay spreads occur relatively infrequently, they can have a major impact on the system performance. To accurately evaluate the relative performance of candidate **RTTs** (*Run Trip Time*), it is desirable to model the variability of delay spread as well as the "worst case" locations where delay spread is relatively large.

As this delay spread variability cannot be captured using a single tapped delay line, up to two multipath channels are defined for three environments. Within each environment, channel A is the low delay spread case that occurs frequently, channel B is the median delay spread case that also happens frequently. Each of these two channels is expected to be encountered for some percentage of tine in a given test environment. Table 3.3 gives the percentage of time the particular channel may be encountered.

Test Environment	r.m.s. A (ns)	P(A) (%)	r.m.s. B (ns)	P(B) (%)
Indoor Office	35	50	100	45
Outdoor to Indoor and Pedestrian	45	40	750	55
Vehicular - High Antenna	370	40	4000	55

Table 3.3: Percentage of time the particular channel [19]

The following tables describe the tapped-delay-line parameters for each of the terrestrial environments. For each tap of the channels three parameters are given: the time delay relative to the first tap, the average power relative to the strongest tap, and Doppler spectrum of each tap. A small variation, $\pm 3\%$, in the relative time delay is allowed so that the channel sampling rate can be made to match some multiple of the link simulation sample rate[19].

Tap	Channel A		Channel B		Doppler Spectrum
	Rel. Delay (nsec)	Avg. Power (dB)	Rel. Delay (nsec)	Avg. Power (dB)	
1	0	0	0	0	FLAT
2	50	-3.0	100	-3.6	FLAT
3	110	-10.0	200	-7.2	FLAT
4	170	-18.0	300	-10.8	FLAT
5	290	-26.0	500	-18.0	FLAT
6	310	-32.0	700	-25.2	FLAT

Table 3.4: Indoor Office Test Environment Tapped-Delay-Line Parameters [19]

Tap	Channel A		Channel B		Doppler Spectrum
	Rel. Delay (nsec)	Avg. Power (dB)	Rel. Delay (nsec)	Avg. Power (dB)	
1	0	0	0	0	CLASSIC
2	110	-9.7	200	-0.9	CLASSIC
3	190	-19.2	800	-4.9	CLASSIC
4	410	-22.8	1200	-8.0	CLASSIC
5	-	-	2300	-7.8	CLASSIC
6	-	-	3700	-23.9	CLASSIC

Table 3.5: Outdoor to indoor and Pedestrian Test Environment Parameters [19]

Tap	Channel A		Channel B		Doppler Spectrum
	Rel. Delay (nsec)	Avg. Power (dB)	Rel. Delay (nsec)	Avg. Power (dB)	
1	0	0.0	0	-2.5	CLASSIC
2	310	-1.0	300	0	CLASSIC
3	710	-9.0	8900	-12.8	CLASSIC
4	1090	-10.0	12900	-10.0	CLASSIC
5	1730	-15.0	17100	-25.2	CLASSIC
6	2510	-20.0	20000	-16.0	CLASSIC

Table 3.6: Vehicular test Environment Tapped-Delay-Line Parameters [19]

3.2.2 Channel Coefficient

In 1968, Clarke developed a model where statistical characteristics of the electro-magnetic fields of the received signal at the mobile are deduced from scattering [20].

$$g(t) = g_c(t) + jg_s(t) \tag{3. 3}$$

where

$$g_c(t) = E_0 \sum_{n=1}^{N} C_n \cos\left(2\pi f_d t \cos\alpha_n + \phi_n\right) \tag{3. 4}$$

and

$$g_s(t) = E_0 \sum_{n=1}^{N} C_n \sin\left(2\pi f_d t \cos\alpha_n + \phi_n\right) \tag{3. 5}$$

f_d is the Doppler shift as expressed in (3.2), N is the path number, E_0 is the real amplitude of local average E-field, α_n, ϕ_n are the random phases distributed from 0 to 2π, C_n is a random variable representing the amplitude of individual wave, and

$$\sum_{n=1}^{N} C_n^2 = 1 \tag{3. 6}$$

Latter on, Gans developed a spectrum analysis for Clarke model in 1972 [21], the power spectral density can be expressed as

$$S(f) = \frac{P}{\sqrt{f_m^2 - (f - f_c)^2}} \left[p(\alpha) G(\alpha) + p(-\alpha) G(-\alpha) \right] \tag{3. 7}$$

where f_m is the maximum Doppler frequency, $p(\alpha)d\alpha$ is the fraction of the total incoming power within $d\alpha$ of the angle α, $G(\alpha)$ is the antenna gain, and P is the average received power with respect to an isotropic antenna. The Doppler power spectrum $S(f)$ has a U shape as shown in Fig. 3.5.

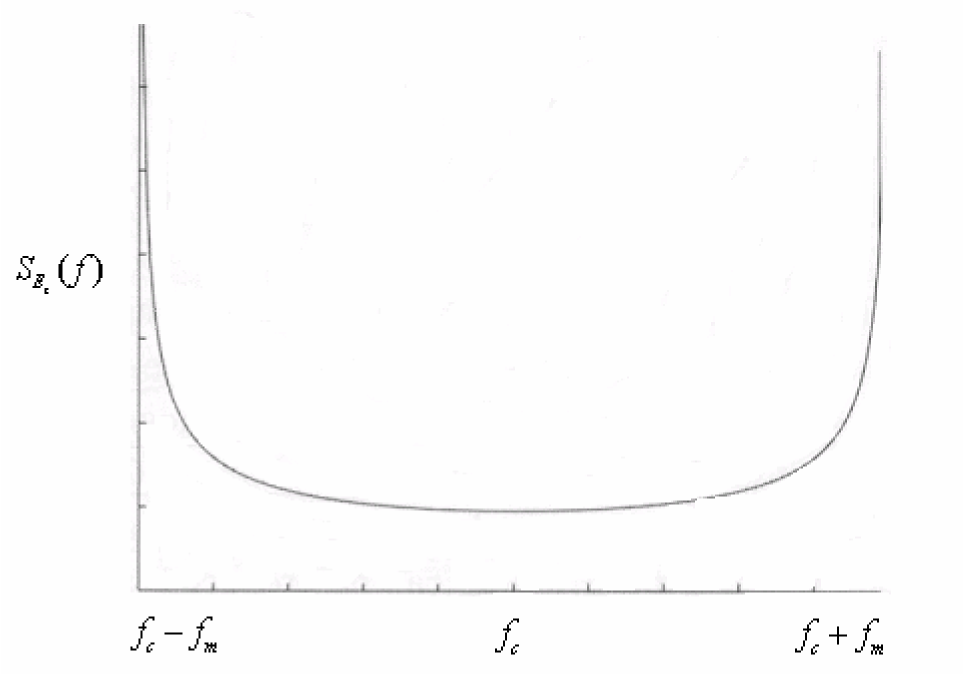


Fig 3.5: Doppler power spectrum [37]

With Doppler power spectrum, we can generate the fading channel coefficients. However, to generate the channel coefficients which follow the **Clarke** model takes three random number generators, it makes implementing random number generator itself a difficult task. In 1974, Jakes came up with a simplified model [22].

$$u(t) = u_c(t) + u_s(t) \tag{3. 8}$$

where

$$u_c(t) = \frac{2}{\sqrt{N}} \sum_{n=0}^{M} a_n \cos(w_n t) \tag{3. 9}$$

$$u_s(t) = \frac{2}{\sqrt{N}} \sum_{n=0}^{M} b_n \cos(w_n t) \tag{3.10}$$

and

$$N = 4M + 2$$

$$a_n = \begin{cases} \sqrt{2}\cos\beta_0, & n=0 \\ 2\cos\beta_n, & n=1,2\ldots M \end{cases} \qquad \beta_n = \begin{cases} \frac{\pi}{4}, & n=0 \\ \frac{\pi n}{M}, & n=1,2\ldots M \end{cases}$$

$$b_n = \begin{cases} \sqrt{2}\sin\beta_0, & n=0 \\ 2\sin\beta_n, & n=1,2\ldots M \end{cases} \qquad w_n = \begin{cases} w_d, & n=0 \\ w_d\cos\frac{2\pi n}{N}, & n=1,2\ldots M \end{cases}$$

This model has been used for decades for its simple structure. As we can see, there is no random variable needed, and this reduces the complexity of implementation.

However, Jakes model is a deterministic model, which means that if the parameters are given, the model will generate the same coefficients. While simulating multiple uncorrelated channels, Jakes model must be modified. Moreover, the statistical behaviors of Jakes model leave much space to improve.

From the Clarke model in (3. 3) to (3. 5), we find that the second order statistical properties are:

$$\begin{aligned} R_{g_c g_c}(\tau) &= E\left[g_c(t) g_c(t+\tau)\right] = J_0(w_d \tau) \\ R_{g_s g_s}(\tau) &= J_0(w_d \tau) \\ R_{g_c g_s}(\tau) &= 0 \\ R_{g_s g_c}(\tau) &= 0 \\ R_{gg}(\tau) &= E\left[g(t) g^*(t+\tau)\right] = 2J_0(w_d \tau) \end{aligned} \tag{3. 11}$$

where $R_{g_c g_s}(\tau)$ is the crosscorrelation of g_c and g_s, $R_{g_s g_c}(\tau)$ is the crosscorrelation of g_s and g_c, $R_{g_c g_c}$ and $R_{g_s g_s}$ are autocorrelations, and $J_0(\bullet)$ is the zero-th order Bessel function of the first kind [23]. While in Jakes model, the statistical properties are not quite the same.

$$\begin{aligned}
R_{u_c u_c}(\tau) &= \frac{4}{N}\left[\sum_{n=0}^{M}\frac{a_n^2}{2}\cdot\cos(w_n\tau)\right] \\
R_{u_s u_s}(\tau) &= \frac{4}{N}\left[\sum_{n=0}^{M}\frac{b_n^2}{2}\cdot\cos(w_n\tau)\right] \\
R_{u_c u_s}(\tau) &= \frac{4}{N}\left[\sum_{n=0}^{M}\frac{a_n b_n}{2}\cdot\cos(w_n\tau)\right] \\
R_{u_s u_c}(\tau) &= R_{u_c u_s}
\end{aligned} \tag{3.12}$$

Despite Jakes simulator and derivatives of Jakes model have gained widespread acceptance, literatures never stop improving the model [24]-[28]. In [29], the author proposed the idea of adding the random phase, so the model became **WSS** (*Wide Sense Stationary*). Further in [30], the authors modified the model into

$$X(t) = X_c(t) + jX_s(t) \tag{3.13}$$

$$X_c(t) = \frac{2}{\sqrt{M}}\sum_{n=1}^{M}\cos(\psi_n)\cdot\cos(w_d t\cos\alpha_n + \phi) \tag{3.14}$$

$$X_s(t) = \frac{2}{\sqrt{M}}\sum_{n=1}^{M}\sin(\psi_n)\cdot\cos(w_d t\cos\alpha_n + \phi) \tag{3.15}$$

with $$\alpha_n = \frac{2\pi n - \pi + \theta}{4M}\ , \quad n = 1,2\ldots..,\mathrm{M} \tag{3.16}$$

where θ, ϕ, φ_n are statistically independent and distributed over [- π , π] uniformly for all n (one thing we must notice that θ, ϕ are random variables chosen only once in the beginning of the model, while φ_n is a random variable changing from every n).

With this modified model (we call it as the **Xiao** model), when M reaches infinity, the envelope $|X|$ will be Rayleigh distributed, the PDF is given by

$$f_{|X|}(x) = x\cdot\exp\left(-\frac{x^2}{2}\right)\ , \quad x \geq 0 \tag{3.17}$$

and the second order statistical properties are

$$
\begin{aligned}
&R_{X_cX_c}(\tau) = J_0(w_d\tau)\\
&R_{X_sX_s}(\tau) = J_0(w_d\tau)\\
&R_{X_cX_s}(\tau) = 0\\
&R_{X_sX_c}(\tau) = 0\\
&R_{XX}(\tau) = E\left[g(t)g^*(t+\tau)\right] = 2J_0(w_d\tau)
\end{aligned}
\tag{3.18}
$$

Compared with (3.12), we find their second order statistical characteristics are the same. Moreover, the initial phase ϕ in the proposed model ensures the model stationary in the widesense, and the random variable θ randomizes the radian Doppler shift $w_d \cos\alpha_n$. This makes Xiao model be different from all the existing Jakes family simulators. The random φ_n ensures the quadrature components of the fading channel uncorrelated. Figures 3.6 to 3.9 show the simulation results of second-other statistical properties for this model.

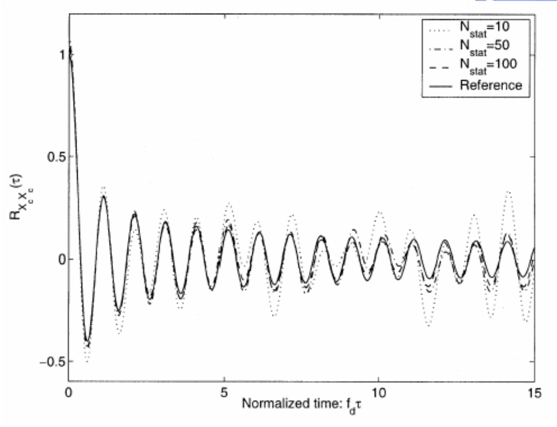


Fig3.6: Autocorrelation of $X_c(t)$ [30]

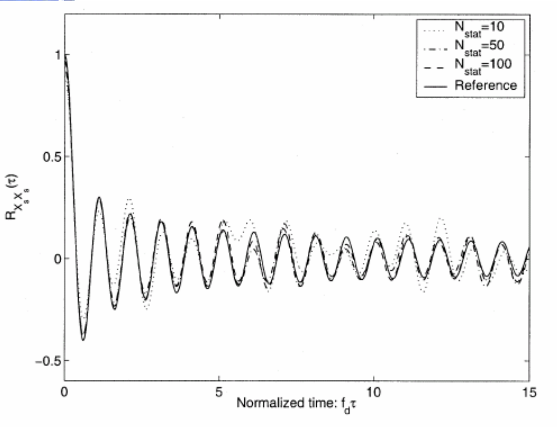


Fig3.7: Autocorrelation of $X_s(t)$ [30]

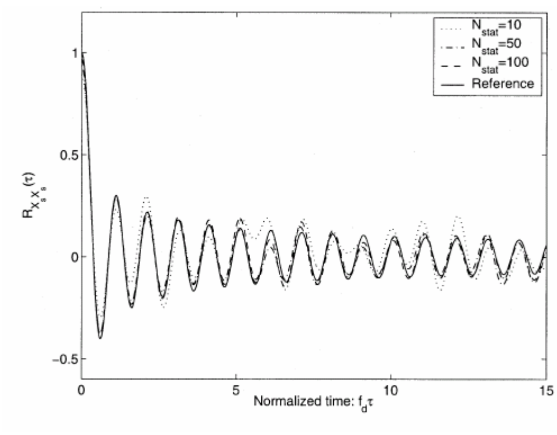


Fig 3.8: Crosscorrelation of $X_c(t)$ and $X_s(t)$

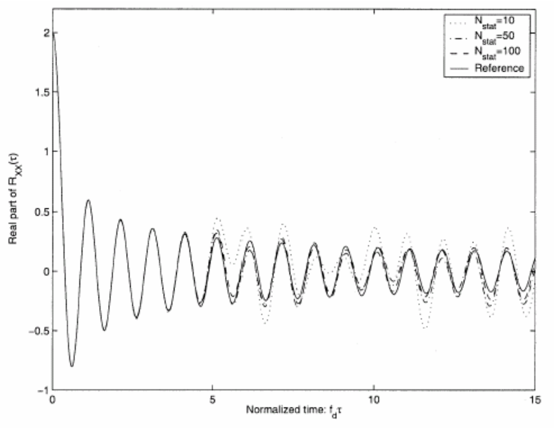


Fig 3.9: Autocorrelation of X(t) [30]

where reference $g(t)$ are Clarke model, and N_{stat} stands for the statistical trials.

In total, this model needs M+2 random variable, but it gains significant advantages in statistical.

3.3 Bottleneck for the system when operating at high speed

In Sections 3.1 and 3.2, we design a system following the 802.16a specification and pick a channel model for simulating the system. In this section, we would like to discuss the possible limit on the supporting mobility rate for 802.16a specification through theoretical calculation.

As described in Chapter 2, **OFDM** systems utilize **CP** to mitigate the **ISI** effects due to delay spread. Roughly speaking, **CP** time interval must be longer than the maximum delay tap. A better defined constraint imposed on **CP** length by **ISI** contribution to the decoder **SIR** **(** *Signal to Interference Ratio***)** can be expressed as [31]

$$\int_{0}^{T_{cp}} |h(t)|^2 \, dt > \left(1-\theta_{cp}\right)\int_{0}^{\infty} |h(t)|^2 \, dt \tag{3.19}$$

where $h(t)$ is the channel impulse response and T_{cp} is the **CP** time length. The θ_{cp} sets the requirement that the channel impulse response up to T_{cp} contains at least a fraction 1-θ_{cp} of the total impulse energy, and is chosen to system **SIR** requirements. Typical values for θ_{cp} range from 0.02 to 0.25, depending on **SIR**[32].

If we set **SIR** 10 dB in our system, θ_{cp} will be equal to 0.1. To capture $\left(1-\theta_{cp}\right)$=90% of impulse energy of worst-case delay spread in our chosen channel model[19], it require $T_{cp} \approx 10\mu s$, while in our **PHY** design, the CP time length is $\approx 20\mu s$, which is double the time length we need. Thus, in our system simulation, there should not exist **ISI** and the orthognality problems.

Due to the Doppler shift in the time varying channel response, the received **OFDM** symbol may be distorted by the time-domain smearing of the signal. We roughly define the constraint

$$T_b < \theta_d \cdot \tau_{chan} \tag{3.20}$$

where T_b is the data period , τ_{chan} is the channel coherence time, and θ_d is some predetermined fraction of the channel de-coherence time, which in turn depends inversely on the mobility rate. Equation (3.20) means that the data period T_b should be shorter than channel de-coherence time, so the signal will not be distorted. Previous simulation studies [32], and actual parameters used in the field OFDM systems indicate that θ_d is rarely larger than 0.1.

We apply this constraint to our system, operating in a bandwidth of 10 MHz, at a frequency of 2GHz, $T_b \approx 179\mu s$ as described in Section 3.1, and set θ_d to 5%. Through (3.20), τ_{chan} should be larger than 3580 μs, and could be expressed as

$$\tau_{chan} \approx \frac{1}{f_d} > 3580\mu s \tag{3.21}$$

By (3.21) the maximum Doppler frequency is 279.32 μs; through (3.2), the maximum supporting mobility rate shall be 41.9 (m/s), which means that 150 Km/hr is supported.

In this chapter, we introduce our system design, channel parameters, channel coefficients for the simulator, and the possible maximum mobility rate. We will go to the implementation stage in next chapter.

Chapter 4
Hardware Implementation

In the previous chapters, the backgrounds, including IEEE 802.16a, system parameters, and channel environment are given. In this chapter, we shall discuss the hardware implementation on the channel simulator.

4.1 Quixote Board[34]

The DSP-FPGA embedded card used in our system is Innovative Integration's Quixote-III**,** which is shown in Figure 4.1. It is a PCI bus compatible card housing one **TI**'s (*Texas Instruments company*) TMS320C6416 digital signal processor (**DSP**) and one Xilinx's Virtex-II XC2V6000 field programmable gate arrays (**FPGA**) in a symmetric multiprocessing relationship with high bandwidth inter-DSP-FPGA communication links. The block diagram of Quixote-III is shown in Figure 4.2.

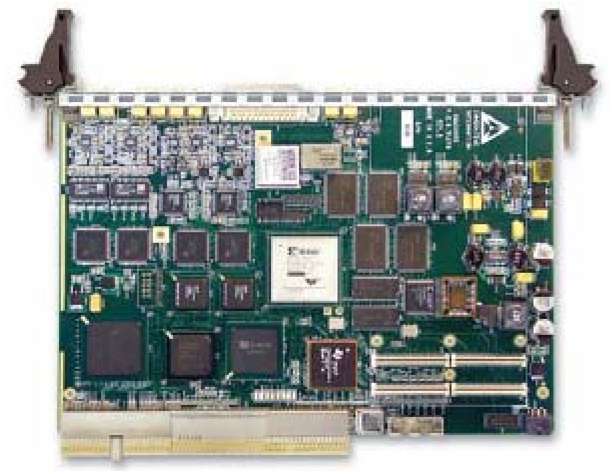


Fig 4.1: Quixote-III [34]

The main features are as follows:

1. TMS 320C6416 processor running at frequency up to 600 MHz.
2. On board 32MB SDRAM for DSP chip, enhanced cache controllers, 64 DMA channels, 3 MCBSP sync serial ports and two 32 bits timers.
3. A 32/64 bits PCI bus host interface with direct host memory access capability for busmatering data between the card and the memory.
4. Onboard 40MB/sec FIFO port for fast data transmission between II's DSP board.
5. 2 in, 2 out A/D, D/A conversion , 14 bit, DC-105MHz

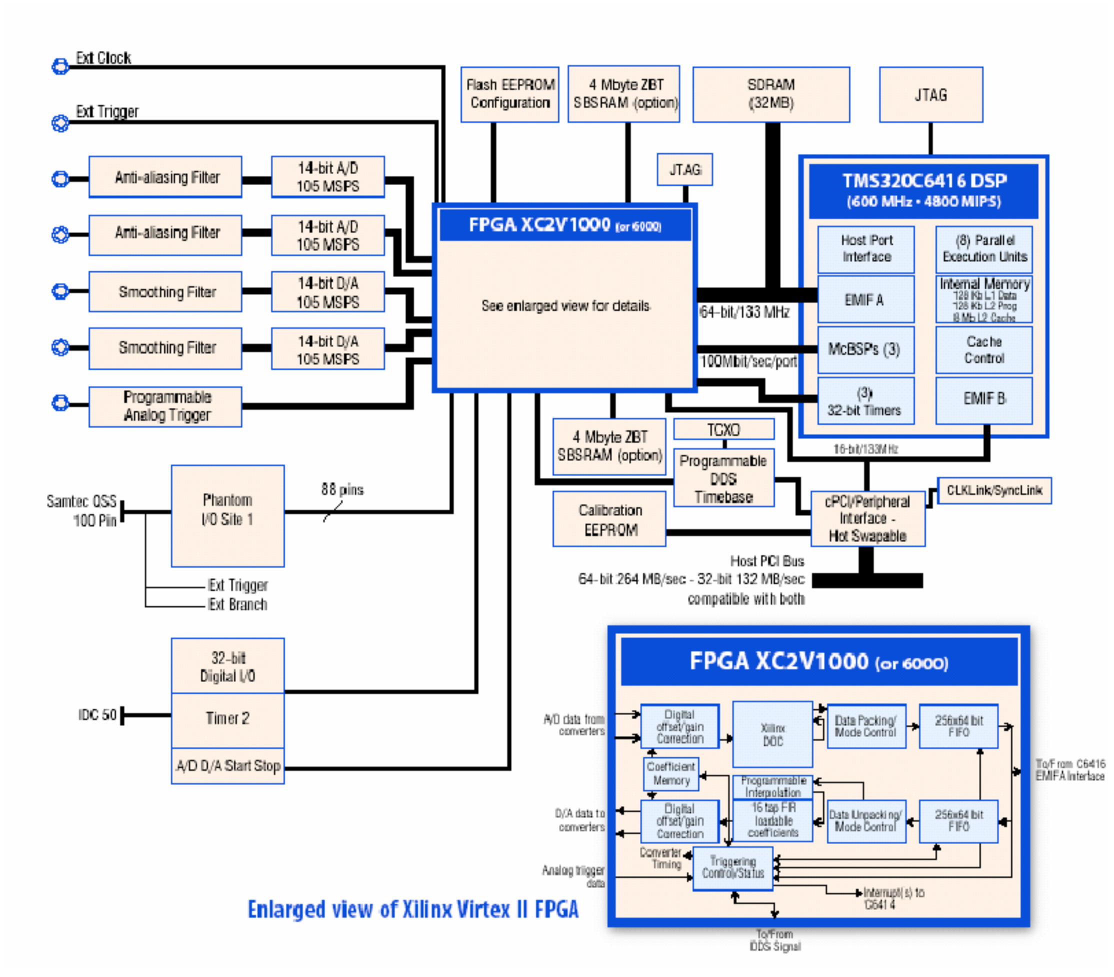


Fig 4.2: Block diagram of Quixote-III [34]

4.1.1 DSP Chip Introduction

The DSP chip used is TI’s TMS 320C6416. It employs the “VelociTI” architecture, a variant of the traditional **VLIW** architecture, which consists of multiple execution units running in parallel, performing multiple instructions during one cycle time. TMS 320C6416 is a fixed-point **DSP**, with 8 function units running at 600 MHz and 4800M instructions per second. It’s internal memory includes a two-level cache architecture with 16 KB of L1 data cache, 16 KB of L1 program cache, and 1 MB L2 cache for data/program allocation. On-chip peripherals include two multichannel buffered serial ports (**McBSPs**), two timers, a 16-bit host port interface (**HPI**), and 32-bit external memory interface (**EMIF**). Internal buses include a 32-bit program address bus, a 256-bit program data bus to accommodate eight 32-bit instructions, two 32-bit data address buses, two 64-bit data buses, and two 64-bit store data buses. The block diagram of TMS 320C6416 is shown in Figure 4.3.

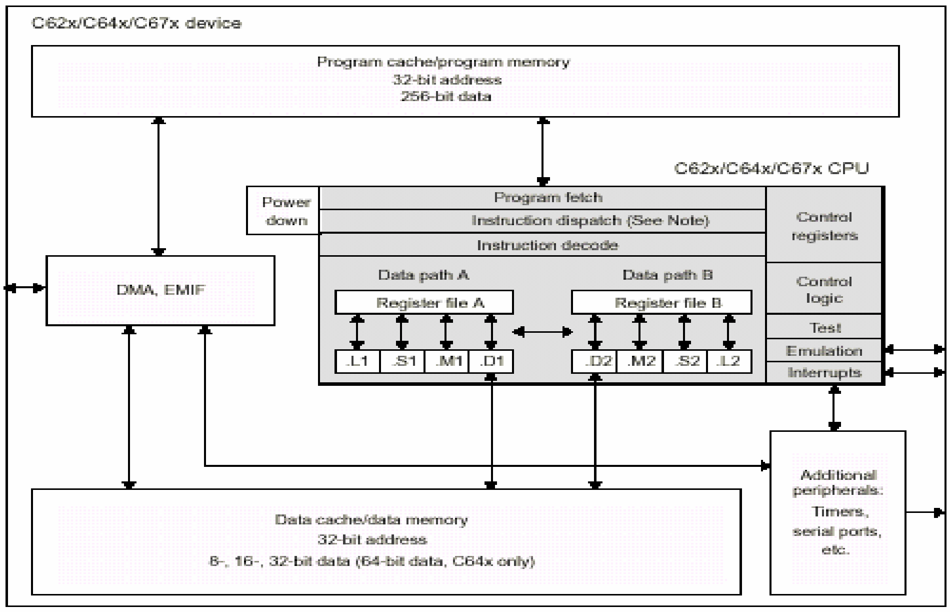


Fig. 4.3: Block diagram of TMS320C6416

From the schematic in Fig 4.3, we can schematically get the picture of CPU structure. Further in Fig.4.4, the detailed structure of C64X CPU is illustrated.

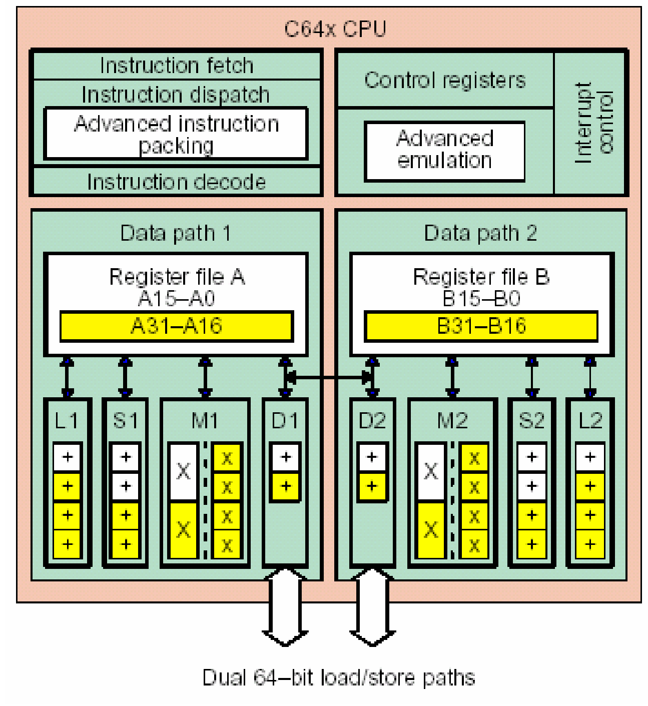
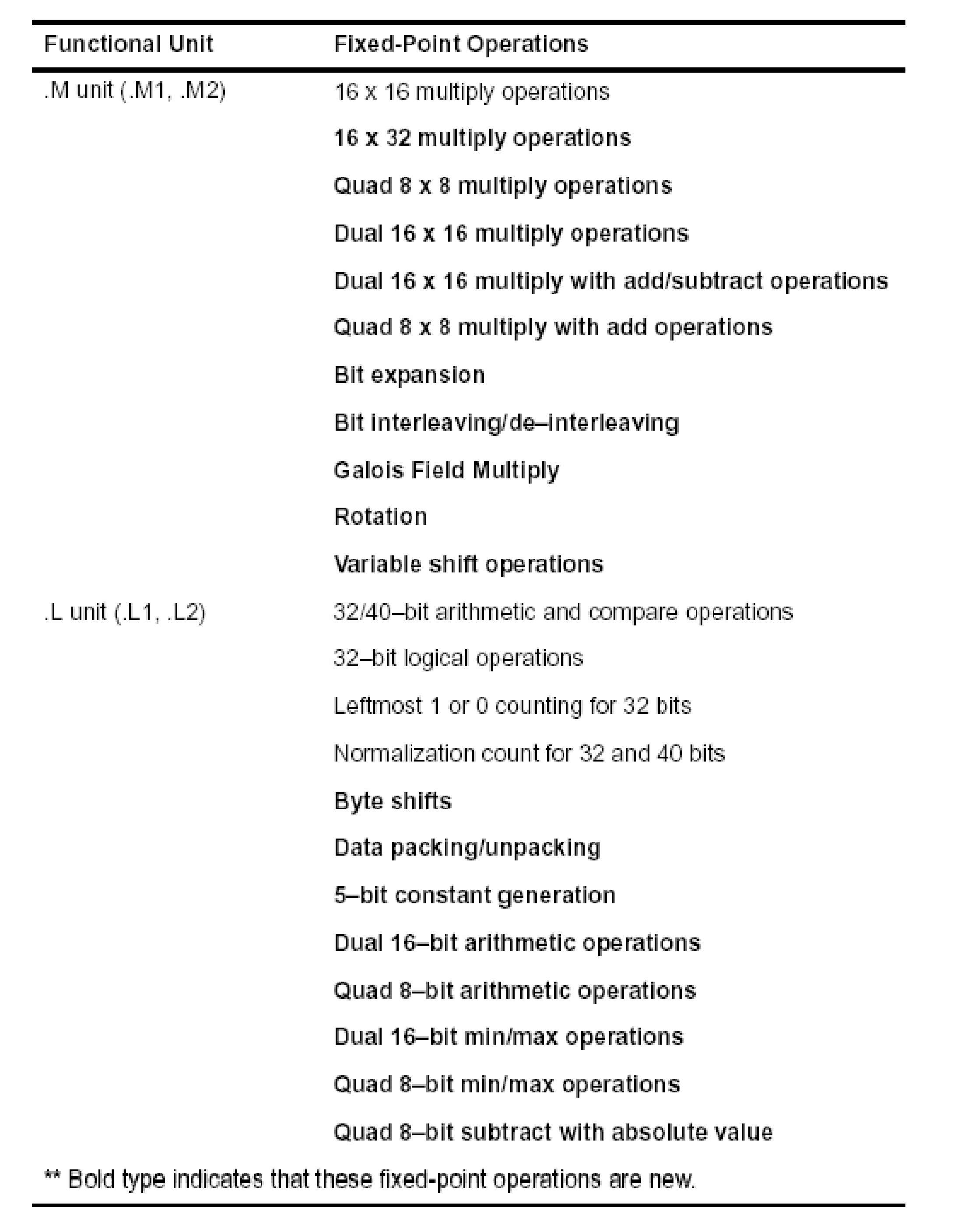


Fig. 4.4: C64X CPU structure [35]

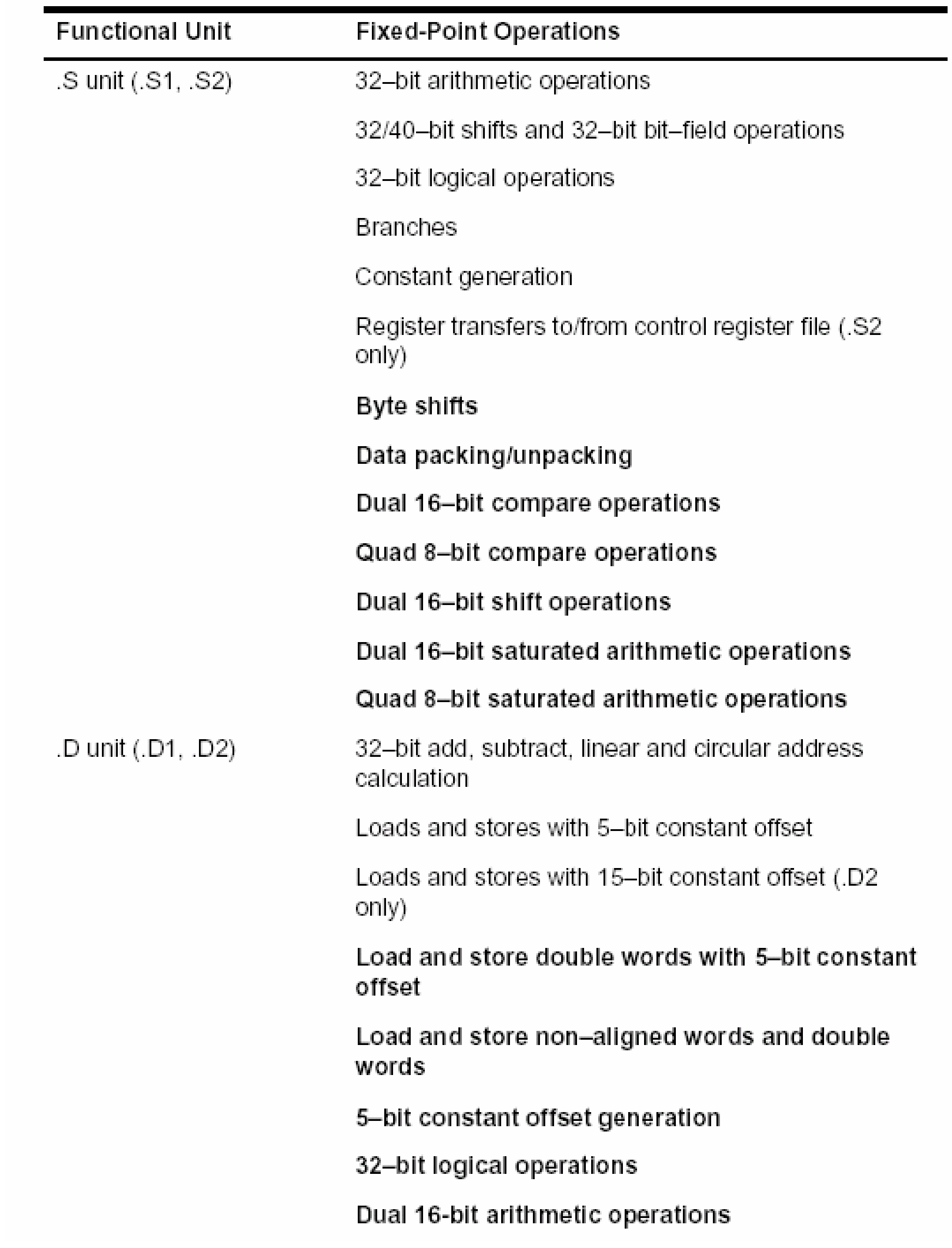
The C64X CPU consists of 2 general purpose register files (A and B), 8 functional units (L1,S1,M1,D1,L2,S2,M2,and D2), 2 load-from-memory paths(LD1 and LD2), 2 store-to-memory paths(SD1 and SD2), 2 data address paths (DA1 and DA2) and 2 register cross data paths(1X and 2X). The functional units can perform multiple modes of bits operation, such as $16\times16, 32\times16, 8\times8$ …etc, detailed information is given in Table 4.1 and Table 4.2.



Functional Unit	Fixed-Point Operations
.M unit (.M1, .M2)	16 x 16 multiply operations
	16 x 32 multiply operations
	Quad 8 x 8 multiply operations
	Dual 16 x 16 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
	Galois Field Multiply
	Rotation
	Variable shift operations
.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 arithmetic operations
	Quad 8–bit arithmetic operations
	Dual 16–bit min/max operations
	Quad 8–bit min/max operations
	Quad 8–bit subtract with absolute value

** Bold type indicates that these fixed-point operations are new.

Table 4.1: C64x CPU Function units and Operation performed [35]



Functional Unit	Fixed-Point 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 compare operations
	Quad 8–bit compare operations
	Dual 16–bit shift operations
	Dual 16–bit saturated arithmetic operations
	Quad 8–bit saturated arithmetic operations
.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)
	Load and store double words with 5–bit constant offset
	Load and store non–aligned words and double words
	5–bit constant offset generation
	32–bit logical operations
	Dual 16-bit arithmetic operations

Table 4.2: C64X CPU Function units and Operation performed [35]

4.1.2 Xilinx FPGA Chip [36]

Xilinx Virtex-II XC2V6000 is produced by 0.15 μ, 8-layer metal process with a likely running frequency up to 300MHz depending on the designed circuit. It has 33792 slices made up by 6M system gate counts, and that explains the name “XC2V6000”. “Slice” is the smallest area unit in **FPGA** design, which consists of 2 FF/LAT units (Flip-Flop/ Latch), 2 four-input LUTs (Look-up table) and other logic units. Figure 4.5 shows a general slice diagram. Through the routing combinations of slices, a **FPGA** chip can simulate different kind of complex circuits.

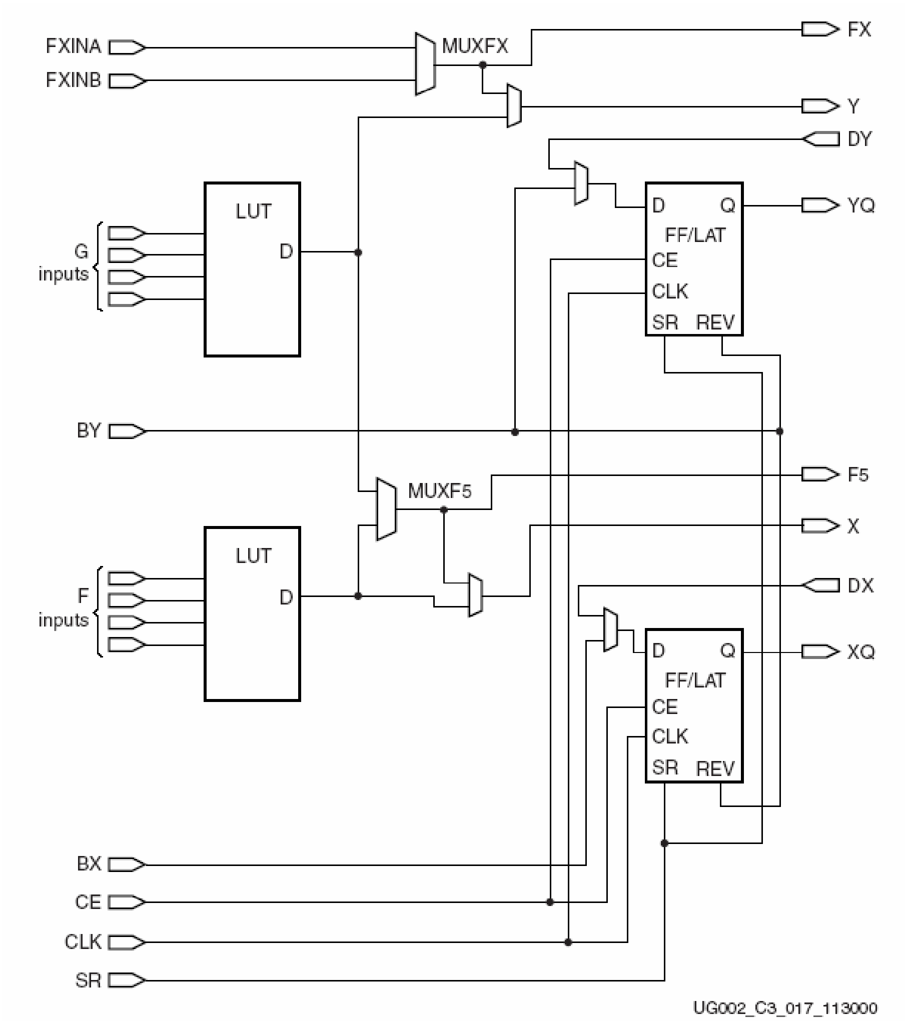


Fig. 4.5 : General Slice Diagram [36]

The main constraints of the **FPGA** chips are the area and the number of multipliers. Once the designed circuit exceed the chip area constraint, it can not be fitted into the

FPGA chip. Another barrier is the usage of multipliers. Through routing, Virtex-II XC2V6000 can offer 144 multipliers as a whole, once the multiplier usage of the circuit exceeds the constraint, the circuit will fail to fit in the chip, too. We list the key features and constraints below.

1. 6M system gates
2. 33792 slices
3. 144 multipliers
4. 16 GCLKs (Gate Clocks)
5. 67584 Slice Flip Flops and 4-input LUTs

4.1.3 Data Transmission Mechanism

In this section, we will introduce the data transmission mechanism between the Host **PC** and Quixote-III**.**

There are 2 modes of transmission interfaces between Host **PC** and Quixote. The primary busmaster interface is a streaming model where logically data is an infinite stream between the source and destination. This model is efficient because the signaling between the two parties in the transfer can be kept to a minimum and transfers can be buffered for maximum throughput. On the other hand the streaming model can have relatively high latency for a particular piece of data, this is because a data item may remain in internal buffering until subsequence data accumulates to allow an efficient transfer. Theoretically, the transmission rate can reach the limitation of PCI interface, that is around 1065Mbps.

Another transmission interface is the message interface. The **DSP** and host have a lower bandwidth communications link for sending commands or side information between host **PC** and and target **DSP,** with a lower transmission rate around 20K to 60K bps.

4.2 Fundamental Functions for Channel Coefficients

In section 3.2.2, we introduced three channel coefficient models for Rayleigh fading channel simulation, they are Clarke model (3.4) to (3.6), Jakes model (3.8) to (3.10), and Xiao model (improved Jakes model) (3.13) to (3.15).Among them, Clarke model is too complex for hardware implementation owing to the need of three random number generators. So, we will just implement Jakes model and Xiao model on the **FPGA** chip, and take Xiao model as our consideration to generate a set of 6-ray channels simultaneously with maximum Doppler shift of 2000 Hz support for our system.

As we can observe from those channel model equations, the most important part of the equations is the triangular function. We shall introduce three triangular function generators in section 4.2.1, and compare the advantages and disadvantages. In section 4.2.2, we also introduce a binary pseudo random number generator for the implementation of Xiao model. In this thesis, all the syntheses are done by ISE 6.0 developed by Xilinx.

4.2.1 Triangular Function Generator

Generally speaking, there are many methods generating the triangular function, we will introduce three of them, which are all popular triangular function generators.

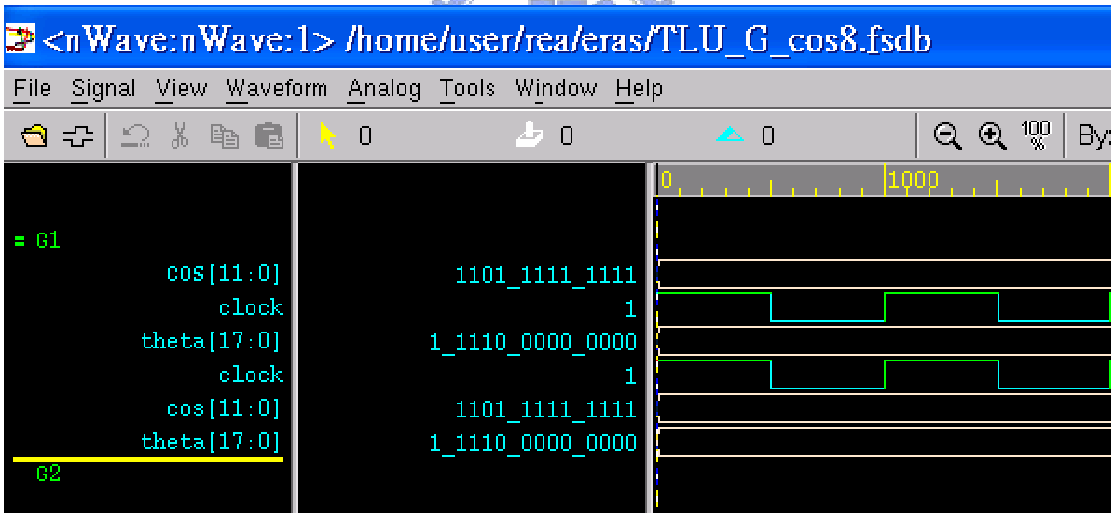
Look-Up Tables

The Look-Up Tables (LUTs) approximation is an extensively documented technique for triangular functions. An LUTs of length 2^x samples describes on period of a unit amplitude sinusoid: the LUTs is addressed by X most significant bits of the input phase ϕ, constituting the sinusoidal value to the nearest sample.

The advantages of LUTs triangular generation will be simple structured, and can generate sinusoidal values fast. The main disadvantage for LUTs will be the large table size.

Fortunately, in a channel simulator implementation, the channel coefficients are only statistical characterized, detail precision is not required. Thus, we can reduce the size of our LUTs.

In our works, we implement the LUTs with three approaches. First, we divide 0 to $\pi/2$ into 180 parts, and use the symmetric property of triangular function to find the rest values outside 0 to $\pi/2$. This method takes 2 clock cycles to complete the procedure. The advantage will be less **FPGA** chip area consuming. The rest two approaches will be directly divide 0 to 2π in to 360 parts and 720 parts respectively. They may be area consuming but fast. An example of direct LUTs-720 division is shown below



<nWave:nWave:1> /home/user/rea/eras/TLU_G_cos8.fsdb

```
G1
   cos[11:0]     1101_1111_1111
   clock                      1
   theta[17:0]  1_1110_0000_0000
   clock                      1
   cos[11:0]     1101_1111_1111
   theta[17:0]  1_1110_0000_0000
G2
```

Fig 4.6.: Example of direct LUTs-720 division

In the LUTs, we design sine and cosine for 12 bits, the first bit is sign bit, 0 for positive, and 1 for negative. The second bit is for integer and the rest 10 bits are for decimal.

The input data are 18 bits, the first bit is sign bit, the following 12 bits are for integer, and the rest 5 bits for the decimal. We give a desired input of theta=18'b0_000011110000_00000, which is 240 degree, and the output value of cosine is directly given as 12'b1_1_0111111111, which is -0.5, and the result is right.

Other implementing results are shown in Table 4.3. From Table 4.3, we find that the data yield rates are all around 100M per second, and their area occupied are all pretty small compared to the whole **FPGA** chip area. So, we decide to use the LUTs which is directly divided into 720 parts within 0 to 2π as our LUTs method.

Performance / Algorithm	Data yield rate (per second)	Area (Slice) Percentage of Area
LUTs-360 division	112.8 M	252 0.74%
LUTs-720 division	102.75 M	634 1.87 %
2-Stages LUTs	93.27 M	162 0.49%

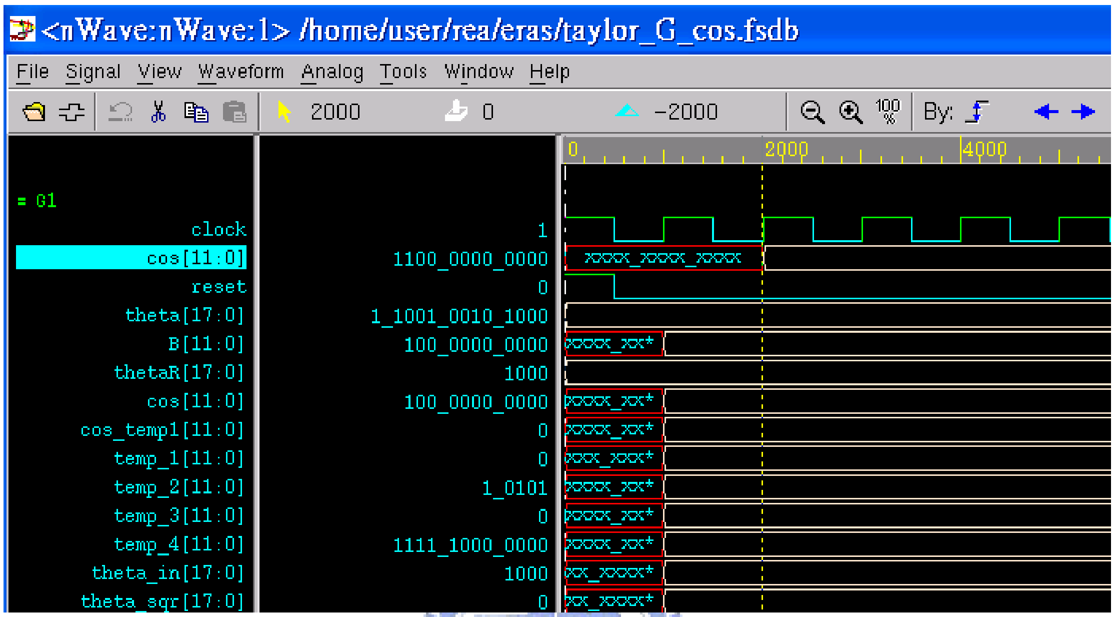
Table 4.3: Comparison of LUTs

Taylor series [38]

$$\cos(x) = 1 - \frac{1}{2!}x^2 + \frac{1}{4!}x^4 - \frac{1}{6!}x^6 + \ldots\ldots = \sum_{n=0}^{\infty} \frac{(-1)^n (x^{2n})}{(2n)!} \text{ where } -\frac{\pi}{2} \le x \le \frac{\pi}{2} \qquad (4.1)$$

$$= x^2\left(x^2\left(\frac{1}{4!} - \frac{1}{6!}x^2\right) - \frac{1}{2}\right) + 1 \qquad \text{For n= 3} \qquad (4.2)$$

Taylor series is one of the most popular methods calculating sinusoidal values. It is widely used in software programs. But in hardware design, especially in **FPGA** simulation, the number of multipliers is always limited. For (4.2), it takes 4 multiplies. If we calculate triangular functions with Taylor series, the need of multipliers will be a barrier. Moreover, Taylor series can only deal with the angle from $-\pi/2$ to $\pi/2$, we have to use the symmetric property to find the other values out of that range. So, it takes two clock cycles to find the right value. We also give a demonstration of Taylor's series cosine function.



<nWave:nWave:1> /home/user/rea/eras/taylor_G_cos.fsdb
File Signal View Waveform Analog Tools Window Help
2000
0
-2000
By:
0
2000
4000

```
= G1
        clock                   1
    cos[11:0]      1100_0000_0000
        reset                   0
  theta[17:0]    1_1001_0010_1000
      B[11:0]       100_0000_0000
 thetaR[17:0]                1000
    cos[11:0]       100_0000_0000
cos_temp1[11:0]                 0
  temp_1[11:0]                  0
  temp_2[11:0]             1_0101
  temp_3[11:0]                  0
  temp_4[11:0]     1111_1000_0000
theta_in[17:0]               1000
theta_sqr[17:0]                 0
```

Fig 4.7: Example of Taylor series cosine function

In Taylor series, we set the input angle 18 bits, the first bit is sign bit, the following 6 bits are for integer, and the rest 11 bits are for decimal. The sine and cosine for 12 bits, the first bit is sign bit, the second bit is for integer and the rest 10 bits are for decimal.

In Fig 4.7, we input a desired angle "theta" for 18'b0_000011_00100101000, which is almost 3.14 radians. The first step is to transform "theta" into "thetaR" which is between 0 to $\pi/2$, and the result is thataR=18'0000000_00000001000, which is almost 0. And the corresponding value of cosine is 12'b1_1_0000000000, which is -1, our output is confirmed.

CORDIC [39]

The third method we would like to introduce is **CORDIC** (*COordinate Rotation DIgital Computer*). **CORDIC** is an iterative algorithm for calculating trig functions including sine, cosine, magnitude and phase. By rotating the phase, multiplying it by a succession of constant values, **CORDIC** generates triangular function values. However, the "multiplies" can all be powers of 2, so in binary arithmetic they can be replaced with shifts and additions. Therefore, it is particularly suited for hardware implementations because it needs no multipliers.

Imagine a fan built from a sequence of right triangles. The longest leg is hinged to the hypotenuse of the preceding section (as shown in Figure 4.8). The angles that meet at the vertex of the fan are all acute, and decrease steadily toward zero. If the desired angle is larger than the accumulated angle, we will add the fan in the positive direction toward the desired angle, and vise versa, till the accumulated angle reaches the desired angle. The size of those angles are listed in Table 4.4. Note that each angle is larger than the half of the preceding one, so the accumulated angles can converge.

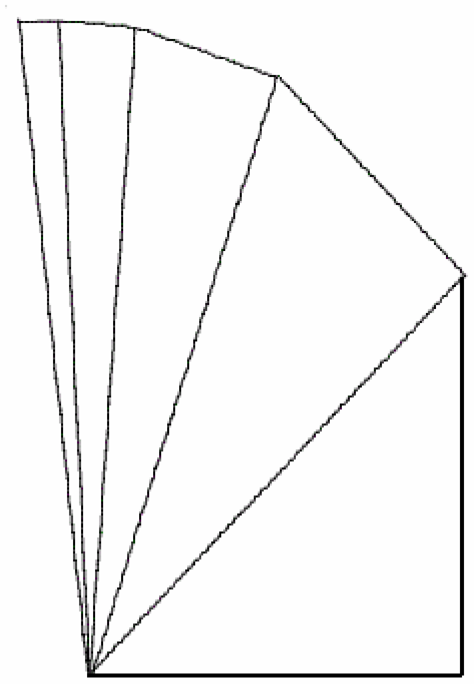


Fig 4.8: Trigonometric Fan

i	$\alpha_i = 2^{-i}$	$E_i = \tan^{-1} 2^{-i}$	i	$\alpha_i = 2^{-i}$	$E_i = \tan^{-1} 2^{-i}$
1	1.000 000	45.000000	8	0.003 906	0.223811
2	0.500 000	26.565051	9	0.001 953	0.111906
3	0.250 000	14.036243	10	0.000 976	0.055952
4	0.125 000	7.125016	11	0.000 488	0.027976
5	0.062 500	3.576334	12	0.000 244	0.013988
6	0.031 250	0.895174	13	0.000 122	0.006994
7	0.007 813	0.447614	14	0.000 061	0.003497

Table 4.4: The accumulating angles

From the coordinate view, as shown in Figure 4.9:

$$X_{i+1} = X_i - d_i Y_i \tan\alpha_i$$
$$Y_{i+1} = Y_i + d_i X_i \tan\alpha_i \qquad (4.3)$$
$$Z_{i+1} = Z_i - \alpha_i$$

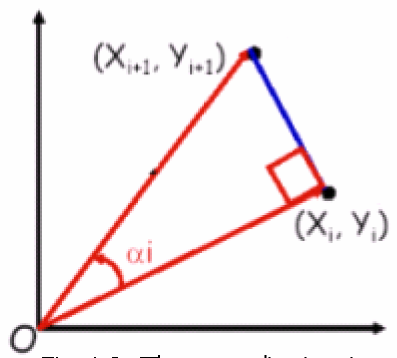


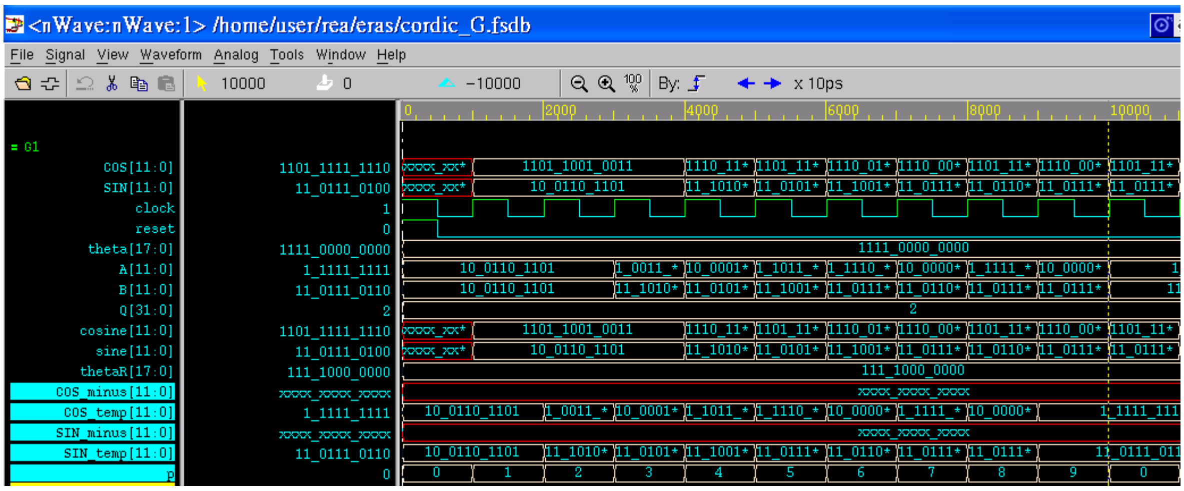
Fig 4.9: The coordinate view

where α_i listed in Table 4.4 is designed such that $\tan\alpha_i = d_i 2^{-i}$, and $d \in [-1,1]$, Z is the angle left to meet the desired angle. If $Z_i > 0, d_i = 1$, and vice versa. After n iterations, when Z_i is close to zero, we can rewrite (4.3) into

$$X_n = K(X\cos Z - Y\sin Z)$$
$$Y_n = K(Y\cos Z + X\sin Z) \qquad (4.4)$$
$$Z_n = 0$$

where $Z = \sum\alpha_i$, and K is the expansion factor of the hypotenuse which is equal to $\prod(1+\tan^2\alpha_i)^{1/2}$ and converges to 1.646760…..

If we set X=1/K=0.607252…and Y=0, we can compute $\cos Z$ and $\sin Z$ simultaneously. In addition, the range of **CORDIC** is also limited between -π/2 to π/2, we have to use the symmetric property to find the values out of this range.


```
<nWave:nWave:1> /home/user/rea/eras/cordic_G.fsdb
File  Signal  View  Waveform  Analog  Tools  Window  Help
10000    0    -10000    By:    x 10ps
0  2000  4000  6000  8000  10000

= G1
      COS[11:0]   1101_1111_1110   xxxx_xx* | 1101_1001_0011 | 1110_11* | 1101_11* | 1110_01* | 1110_00* | 1101_11* | 1110_00* | 1101_11*
      SIN[11:0]   11_0111_0100     xxxx_xx* | 10_0110_1101 | 11_1010* | 11_0101* | 11_1001* | 11_0111* | 11_0110* | 11_0111* | 11_0111*
          clock   1
          reset   0
    theta[17:0]   1111_0000_0000   1111_0000_0000
        A[11:0]   1_1111_1111      10_0110_1101 | 1_0011_* | 10_0001* | 1_1011_* | 1_1110_* | 10_0000* | 1_1111_* | 10_0000* | 1
        B[11:0]   11_0111_0110     10_0110_1101 | 11_1010* | 11_0101* | 11_1001* | 11_0111* | 11_0110* | 11_0111* | 11_0111* | 11
        Q[31:0]   2                2
   cosine[11:0]   1101_1111_1110   xxxx_xx* | 1101_1001_0011 | 1110_11* | 1101_11* | 1110_01* | 1110_00* | 1101_11* | 1110_00* | 1101_11*
     sine[11:0]   11_0111_0100     xxxx_xx* | 10_0110_1101 | 11_1010* | 11_0101* | 11_1001* | 11_0111* | 11_0110* | 11_0111* | 11_0111*
   thetaR[17:0]   111_1000_0000    111_1000_0000
 COS_minus[11:0]  xxxx_xxxx_xxxx   xxxx_xxxx_xxxx
  COS_temp[11:0]  1_1111_1111      10_0110_1101 | 1_0011_* | 10_0001* | 1_1011_* | 1_1110_* | 10_0000* | 1_1111_* | 10_0000* | 1_1111_111
 SIN_minus[11:0]  xxxx_xxxx_xxxx   xxxx_xxxx_xxxx
  SIN_temp[11:0]  11_0111_0110     10_0110_1101 | 11_1010* | 11_0101* | 11_1001* | 11_0111* | 11_0110* | 11_0111* | 11_0111* | 11_0111_011
              p   0                0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 0
```
Fig 4.10: Example of CORDIC

In **CORDIC**, we design sine and cosine for 12 bits, the first bit is sign bit, 0 for positive, and 1 for negative. The second bit is for integer and the rest 10 bits are for decimal. The input data are 18 bits, the first bit is sign bit, the following 12 bits are for integer, and the rest 5 bits for the decimal. And we let **CORDIC** rotate 8 times to reach the desired angle, it takes another 2 clocks to transform the angle to be less than 90 degree.

For example, we input a desired angle "theta" of 120 degree, which in binaty turn is 18'b0_000001111000_00000. The first action of the circuit will transform theta into "thetaR" which is between 0 to 90 degree, which is 18'b0_000000111100_00000, represents 60 degree. When counter "p" is equal to 9, the desired values of cosine and sine are 12'b1_1_1000000100 and 12'b0_0_1101110111. Respectively, they are -0.49999 for cosine and 0.86718 for sine, while the true value of cosine is -0.5 and sine is 0.86602. They are really close.

There is trick when implementing the **CORDIC** algorithm. That is, when the desired angle is close to 0, simple shift and addition will cause an error in **CORDIC**. Because if we simply shift the negative number, the value will become positive due to the addition of 0 in MSB. So, we have to detect the value, and fix it from the error as what have described. As shown in Fig.4.10, SIN_minus and COS_minus are designed to prevent the error from happening.

From the description above, we can conclude that using **CORDIC** may save the number of multiplier, on the other hand, it will be slow when generating the sinusoidal values, since it takes several iterations to reaches the desired angle. In our design, we decide to rotate the **CORDIC** 8 times for simplifying the circuit design. The implementation results of LUTs, Taylor series, and **CORDIC** are shown in Table 4.5.

From Table 4.5, we can clearly find that the Taylor series method leads to the fastest data rate and smallest **FPGA** chip area. But, the usage of multiplier reaches 4.2 % of the total multiplier which **FPGA** chip can offer. And LUts generates the sinusoidal values with 102.75 M per second, with largest **FPGA** chip area as expected. As for

CORDIC, the performance is a little bit disappointed, the data yield rate is only 14 M per second, and the occupied area is almost as large as LUTs method. We also list the comparison of theses three algorithms in Table 4.6.

Performance / Algorithm	Data yield rate (per second)	Area (percentage) Multiplier
CORDIC	14M	555 (1.6%) 0
Taylor series	187.5 M	72 (0.2%) 6 (4.2%)
LUTs-720 division	102.75M	634 (1.87 %) 0

Table 4.5: Comparison of Triangular Functions

Algorithm	Advantages	Disadvantages
LUTs	Fast No Multiplier	Not Precise Large Area
Taylor series	Very Fast Precise Small area	Many Multipliers
CORDIC	Precise No Multiplier	Slow Large Area

Table 4.6: Comparison of Triangular Algorithms

As what we have mentioned in the beginning of this section, for a channel simulator implementation, the channel coefficients are only statistical characterized, detail precision is not required. The **FPGA** chip area occupied and the usage of multipliers are the main constrains. From this point, we conclude that LUTs might be the suitable triangular function algorithm for **FPGA** implementation in a channel simulation.

4.2.2 Pseudo Random Binary Number Generator

After introducing the triangular function generator, we still need a random number generator to implement Xiao model.

Generating random numbers has been a hard problem for engineers for decades. Application of security protocols and encryption algorithms are basically based in the random number generator. For years, literatures came up with numerous kinds of random number generator architectures. But, most of them are very complex and difficult to be implemented on a **FPGA** chip, while we are going to implement a channel model on the same chip.

In this thesis, we consider a simple algorithm to implement the binary random number generator for our system. And we take pseudo random number generator as our consideration. Its primary advantages are that it is easily generated by feedback shift registers, and has a correlation function that is highly peaked for zero delay, which means that the randomness of the sequence is high. However, because it is predetermined, we call it pseudo random.

Figure 4.11 illustrates the generation of a pseudo random binary sequence of length $2^n - 1$, which is accomplished with the use of a set of shift register. After each shift of the contents of the shift register to the right, the contents of some predetermined position registers are used to produce an input to the first stage through an exclusive-or operation. As the procedure goes on, the content bits of all the registers will form a set of pseudo binary random numbers. Proper feedback connections for several values of n are given in Table 4.8. And we use $X^{23} + X^5 + 1$ as our use.

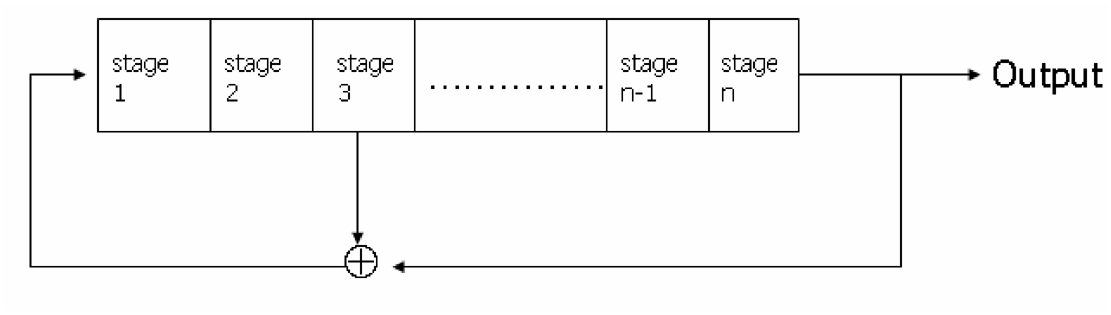


Fig 4.11: The generation of a pseudo random binary sequence

n	Sequence Length	Feedback digit	n	Sequence Length	Feedback digit
2	3	(1,0)	14	16383	(13,4,3,1,0)
3	7	(2,1,0)	15	32767	(14,5,3,1,0)
4	15	(3,1,0)	16	65535	(15,1,0)
5	31	(4,1,0)	17	131071	(16,5,3,2,0)
6	63	(5,2,0)	18	262143	(17,3,0)
7	127	(6,1,0)	19	524287	(18,5,2,1,0)
8	255	(7,1,0)	20	1048575	(19,5,2,1,0)
9	511	(8,4,3,2,0)	21	2097151	(20,3,0)
10	1023	(9,4,0)	22	4194303	(21,2,0)
11	2047	(10,3,0)	23	8388607	(22,1,0)
12	4095	(11,2,0)	24	1677215	(23,5,0)
13	8191	(12,6,4,1,0)	25	3354432	(24,4,3,1,0)

Table 4.8: Feedback connections for generation of pseudo random sequence

Considering its autocorrelation, the autocorrelation is shown in Figure 4.12. Generally speaking, for a sequence of length N, the minimal correlation is -1/N [33]. Because the autocorrelation function of a pseudo random binary sequence consists of a narrow triangle around zero and essentially zero otherwise, this explains the reason for the name " Pseudo".

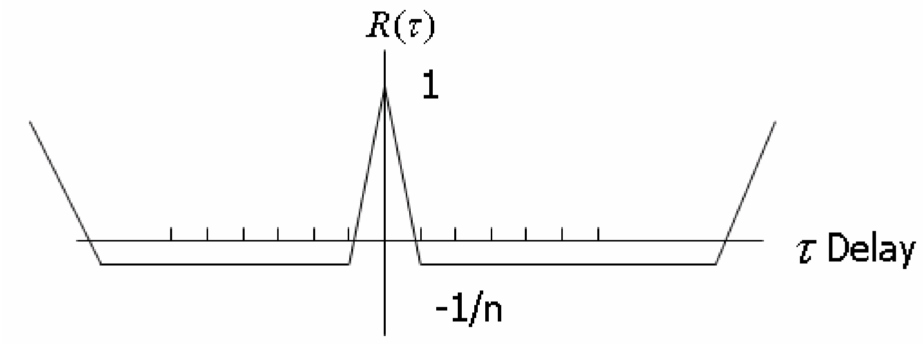


Fig 4.12: Correlation function of a Pseudo Random sequence [37]

In our works, we will use 8 Pseudo random binary generators simultaneously in the Jakes model implementation. The random seeds (initial values of each stage) are given in Table 4.9.

	Initial values of each stage (From MSB to LSB)
PN1	100010101110010100100110
PN2	010111001001110010101101
PN3	101101000111010110011100
PN4	101110011010011100100111
PN5	001010110110101001010100
PN6	010010111010100111100011
PN7	111010101101010101011010
PN8	101011110101010100010101

Table 4.9: The random seeds for each Pseudo random generator

Following are the FPGA implementing results.

Performance / **Algorithm**	**Data yield rate** (per second)	**Area (slice)** **Multiplier**
Pseudo Random	566 M	14 (0.04%) 0

Table 4.10: The FPGA implementation results of PN generator.

As we expected, the FPGA chip area occupied is small, and the speed is fast, since it uses shift registers and exclusive-or operation (that is, a binary add without carry).

4.3 Channel Simulator

After we have introduced the fundamental functions of channel coefficient, we can now start to implement the Rayleigh fading channel simulator. We will implement two single channels followed Jakes model and Xiao model as introduced in the preceding chapter. We will also implement a 6-ray Rayleigh fading channel to meet the channel model we have chosen for our system simulation.

4.3.1 Single-ray Channel Simulator

In the previous sections, we introduced two important components of the channel coefficient models, now we can start to implement a single channel of Jake model and Xiao model.

Jake model

Once again we review the equation of Jakes' model:

$$u(t) = u_c(t) + u_s(t)$$

$$u_c(t) = \frac{2}{\sqrt{N}} \sum_{n=0}^{M} a_n \cos(w_n t)$$

$$u_s(t) = \frac{2}{\sqrt{N}} \sum_{n=0}^{M} b_n \cos(w_n t)$$

where

$$N = 4M + 2$$

$$a_n = \begin{cases} \sqrt{2}\cos\beta_0, & n = 0 \\ 2\cos\beta_n, & n = 1,2 \ldots M \end{cases} \qquad \beta_n = \begin{cases} \frac{\pi}{4}, & n = 0 \\ \frac{\pi n}{M}, & n = 1,2 \ldots M \end{cases}$$

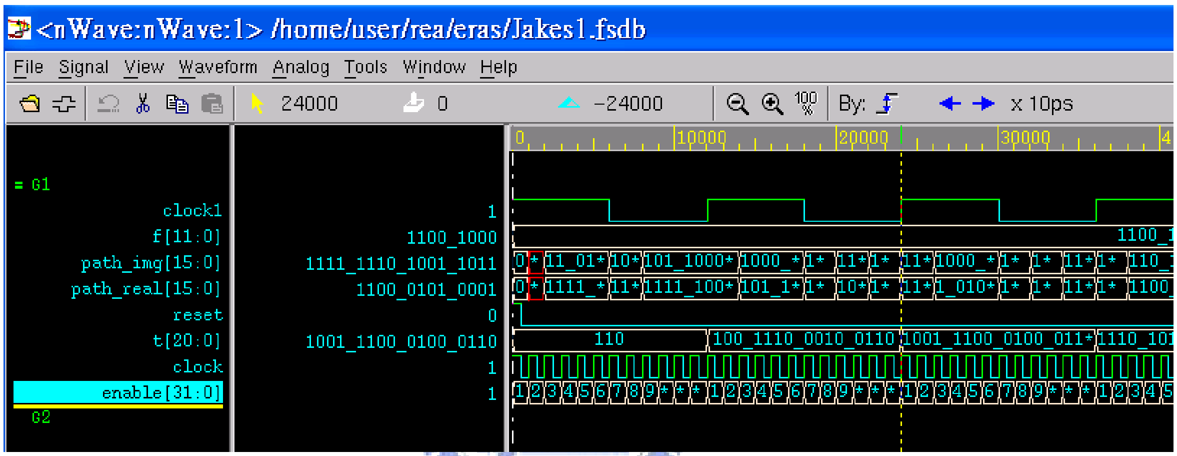
$$b_n = \begin{cases} \sqrt{2}\sin\beta_0, & n = 0 \\ 2\sin\beta_n, & n = 1,2 \ldots M \end{cases} \qquad w_n = \begin{cases} w_d, & n = 0 \\ w_d \cos\frac{2\pi n}{N}, & n = 1,2 \ldots M \end{cases}$$

In our implementation, we set N=34, which is sufficient large to this model. Even though we had made a conclusion that LUTs is the best suitable triangular function generating algorithm for **FPGA** channel simulator, we still implement Jakes model with LUTs, Taylor series, and **CORDIC** algorithms for comparison.

We demonstrate the Jakes model waveform below. In this example, we check for the input Doppler shift=200 Hz, which is represented by f =12'b000011001000 (all the 12 bits stand for integer). Another input t =21'b000000100111000100110, which represents t = 0.0095125 second, if we calculated the Jakes model with computer, the channel coefficient will be 1.51276 for the real part, and -0.18775 for the image part.

We first show the Jakes model implemented by **CORDIC.** We can see there are two clocks control the circuit. The one with smaller period controls the **CORDIC,** it takes 12 clocks for **CORDIC** to transform the input signal to the range it can calculate.

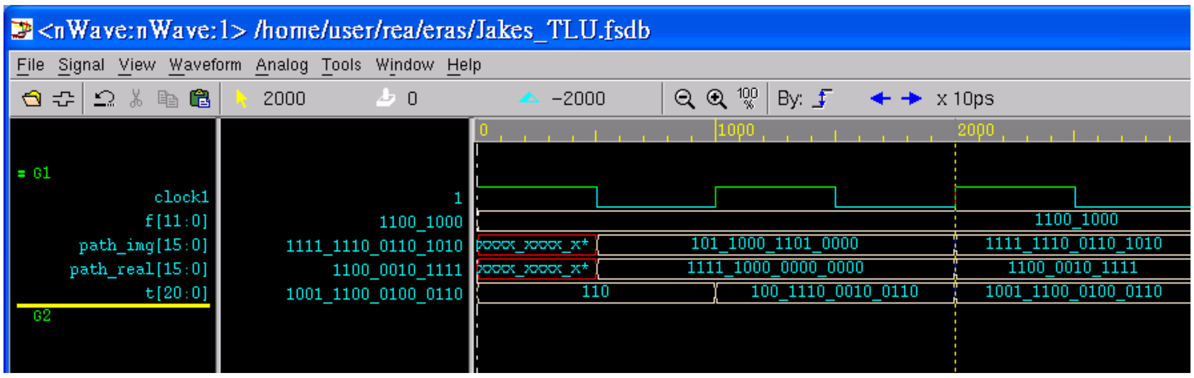
As we can see from Fig 4.13, the "path_real" stands for the real part, which is 16'b0_0001_10001010001, represent 1.51953. The first bit is sign bit, the following 4 bits represent integer, and the rest of 11 bits stand for the decimal. And "path_img" represent image part, which is 16'b1_1111_11010011011, represents -0.1742396. The results are correct. Next, we check the model implemented by LUTs.



<nWave:nWave:1> /home/user/rea/eras/Jakes1.fsdb
File Signal View Waveform Analog Tools Window Help
24000
0
-24000
By:
x 10ps
G1
clock1 1
f[11:0] 1100_1000
path_img[15:0] 1111_1110_1001_1011
path_real[15:0] 1100_0101_0001
reset 0
t[20:0] 1001_1100_0100_0110
clock 1
enable[31:0] 1
G2

Fig 4.13: The demonstration of Jakes model with CORDIC

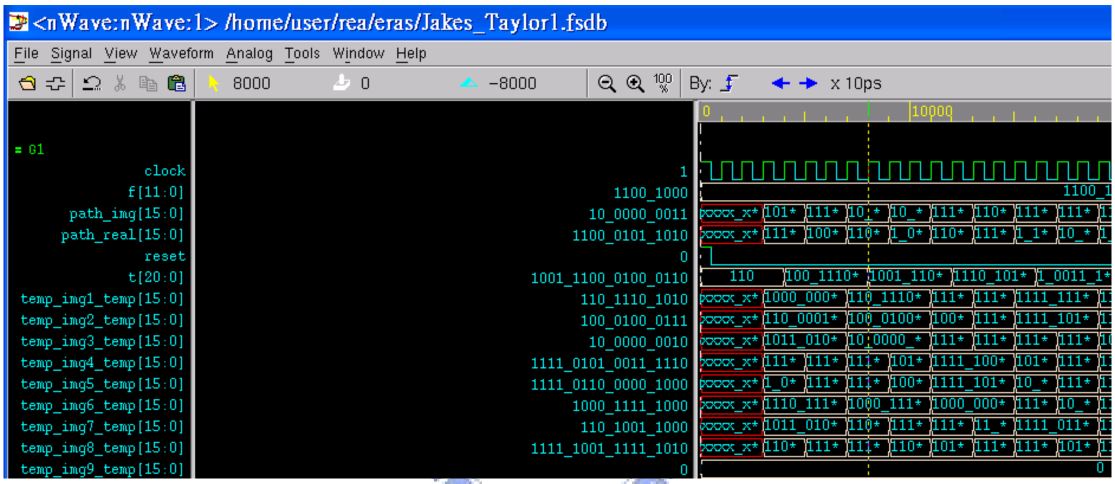
In this demonstration, we check Jakes with LUTs. The output value of real part “path_real” =16'b0_0001_10000101111, which represents 1.522949, and the output value of the image part “path_img” =16'b1_1111_11001101010, which represents -0.20019. While the true value is 1.51276 and -0.18775, the results are right.



<nWave:nWave:1> /home/user/rea/eras/Jakes_TLU.fsdb
File Signal View Waveform Analog Tools Window Help
2000
0
-2000
By:
x 10ps
G1
clock1 1
f[11:0] 1100_1000
path_img[15:0] 1111_1110_0110_1010
path_real[15:0] 1100_0010_1111
t[20:0] 1001_1100_0100_0110
G2

Fig 4.14: The demonstration of Jakes model with LUTs

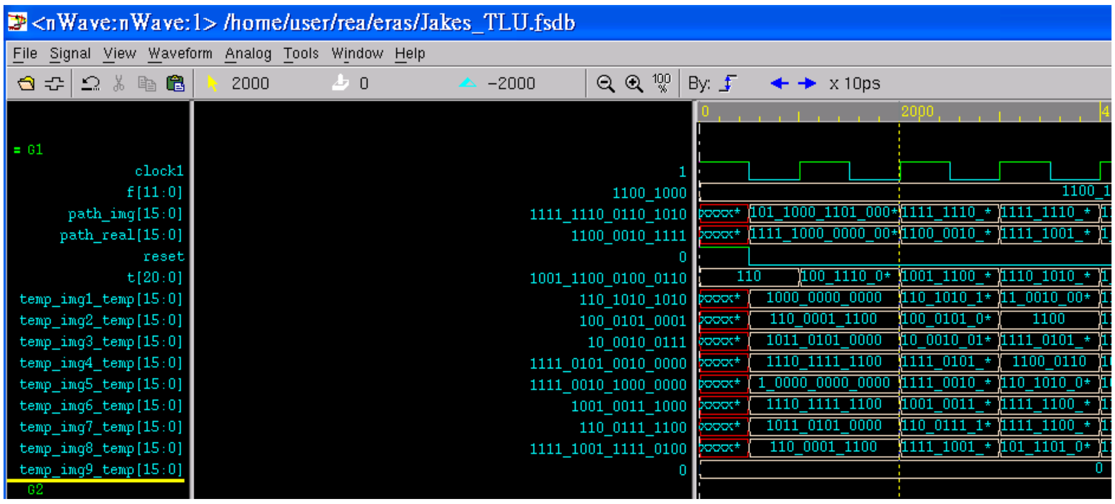
Finally, we check the model implemented by Taylor series. From Fig 4.15, we find that, “path_real”= 16’b0_0001_10001011010, which represents 1.54098, compared with the real value, is almost the same. The output value of the image part “path_img” =16’b0_0000_01000000011, which represent 0.25, is a little bit away form the real value. We try to check it in detail, compare the detailed value with Fig 4.16, and find that each small part of the image contributes a small difference, and make the result away from the real value, the circuit is still right.



<nWave:nWave:1> /home/user/rea/eras/Jakes_Taylor1.fsdb
File Signal View Waveform Analog Tools Window Help
8000 0 -8000 By: x 10ps
0 10000

```
= G1
                 clock                      1
               f[11:0]              1100_1000
        path_img[15:0]           10_0000_0011
       path_real[15:0]         1100_0101_1010
                 reset                      0
               t[20:0]    1001_1100_0100_0110
temp_img1_temp[15:0]          110_1110_1010
temp_img2_temp[15:0]          100_0100_0111
temp_img3_temp[15:0]           10_0000_0010
temp_img4_temp[15:0]    1111_0101_0011_1110
temp_img5_temp[15:0]    1111_0110_0000_1000
temp_img6_temp[15:0]         1000_1111_1000
temp_img7_temp[15:0]          110_1001_1000
temp_img8_temp[15:0]    1111_1001_1111_1010
temp_img9_temp[15:0]                      0
```

Fig 4.15: The demonstration of Jakes model with Taylor series



<nWave:nWave:1> /home/user/rea/eras/Jakes_TLU.fsdb
File Signal View Waveform Analog Tools Window Help
2000 0 -2000 By: x 10ps
0 2000

```
= G1
                clock1                      1
               f[11:0]              1100_1000
        path_img[15:0]    1111_1110_0110_1010
       path_real[15:0]         1100_0010_1111
                 reset                      0
               t[20:0]    1001_1100_0100_0110
temp_img1_temp[15:0]          110_1010_1010
temp_img2_temp[15:0]          100_0101_0001
temp_img3_temp[15:0]           10_0010_0111
temp_img4_temp[15:0]    1111_0101_0010_0000
temp_img5_temp[15:0]    1111_0010_1000_0000
temp_img6_temp[15:0]         1001_0011_1000
temp_img7_temp[15:0]          110_0111_1100
temp_img8_temp[15:0]    1111_1001_1111_0100
temp_img9_temp[15:0]                      0
  G2
```

Fig 4.16: The demonstration of Jakes model with LUTs (as comparison)

The implementation results are as shown below.

Performance / **Algorithm**	**Data yield rate** (per second)	**Area (slice)** **Multiplier**
Jakes with CORDIC	7.5 M	5874 (17%) 56 (38%)
Jakes with Taylor	135.4 M	1889 (5.56%) 110 (70%)
Jakes with LUTs	155M	5799 (16%) 56 (38%)

Table 4.11: Comparison of Jakes' model with different triangular Algos

Except for the Jakes model implanted by **CORDIC,** the data yield rates of the rest two models all exceed 100 M per second, and obviously the occupied **FPGA** chip area of Jakes with LUTs is much larger than Jakes with Taylor series. On the other hand, the usage of multipliers of Jakes model implemented by Taylor series is almost twice more than the other. One thing we should notice that, the usage of multipliers for Jakes model implemented with Taylor series almost reaches 70% of the **FPGA** chip limitation, and our **FPGA** chip, **Virtex-II XC2V6000** actually offers 144 multipliers which is already a large number for **FPGA** chips. Thus, we should be really careful with that model, or we just use Jakes model with LUTs if there is less constraint in the chip size.

So, it depends. If we only implement one channel coefficients generator on the chip, both method will do. If there are still other modules on the same chip, we can pick the one with larger margin to the hardware limitation, either in chip area or the number of multipliers.

Xiao model

We also review this improved channel model here.

$$X(t) = X_c(t) + jX_s(t)$$

$$X_c(t) = \frac{2}{\sqrt{M}} \sum_{n=1}^{M} \cos(\psi_n) \cdot \cos(w_d t \cos \alpha_n + \phi)$$

$$X_s(t) = \frac{2}{\sqrt{M}} \sum_{n=1}^{M} \sin(\psi_n) \cdot \cos(w_d t \cos\alpha_n + \phi) \tag{3.15}$$

with
$$\alpha_n = \frac{2\pi n - \pi + \theta}{4M} \qquad \text{n=1.2…..M} \tag{3.16}$$

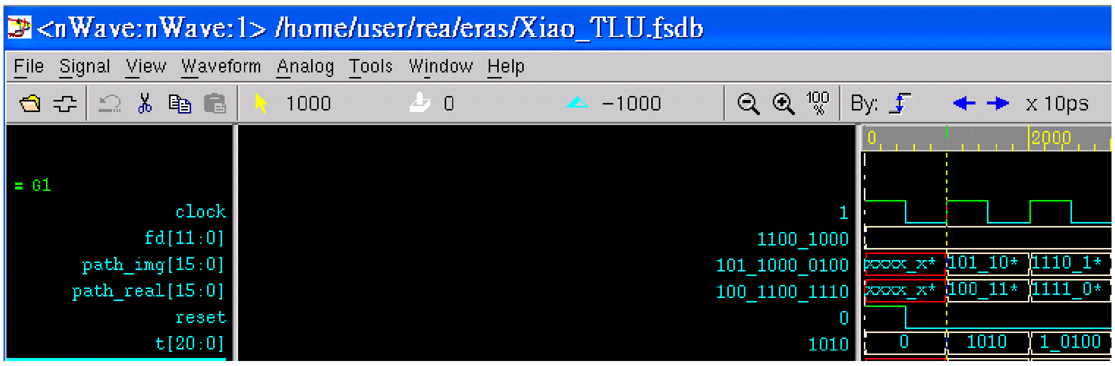
In this case, we chose M=8, which is the same with the parameter used in Jakes model. The random phases θ and ϕ are given -20 and 0 degree for simplicity. Also, for the first cycle, the random phases ψ_n are not calculated by our random generator, we give the value directly. They are listed in Table 4.12.

Initial phase	Binary expression	In degree
ψ_1	18'b0_000010010101_01110	149.4375
ψ_2	18'b0_000000010111_10101	26.625
ψ_3	18'b0_000001100101_11000	101.75
ψ_4	18'b0_000010000010_10111	130.71875
ψ_5	18'b0_000000010101_11011	21.84375
ψ_6	18'b0_000011101101_10110	237.6875
ψ_7	18'b0_000101001010_10111	330.71875
ψ_8	18'b0_000100010101_11101	277.90325

Table 4.12: Initial phase of ψ_n

We also implement such a channel model with three different kind of triangular functions, and try to compare the performance, but, Xiao model with Taylor series takes too many multipliers, and that exceeds the limitation of **Virtex-II XC2V6000**, so we just present other two models. The demonstrations of the works are given below. The input Doppler shift "fd" is 12'b000011001000, represents 200 Hz, when "t" is pretty small, t=21'000000000000000001010, which represents 4.678×10^{-6}. If we calculate the value by the equations (3.13) to (3.16), the real part should be 0.60437, and 0.69197 for the image part.

First, we present the result of the model implemented by LUT's. We can find that, the output value of the real part "path_real"=16'b0_0000_10011001110, which is 0.6005859, the image part "path_img"=16'b0_0000_10110000100, which is 0.70703. While the real values are 0.60437 and 0.69197.The results are verified, which is shown in Fig.4.17

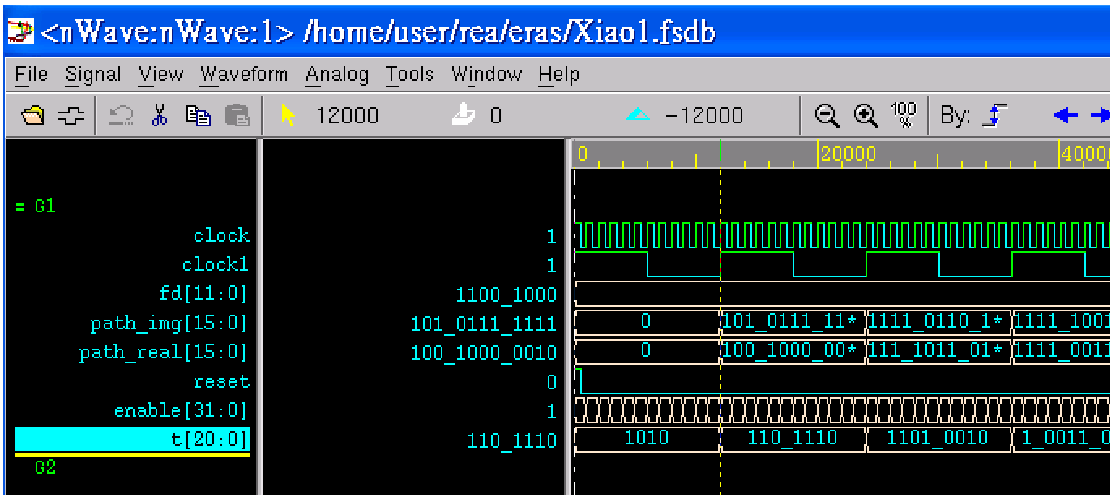


<nWave:nWave:1> /home/user/rea/eras/Xiao_TLU.fsdb
File Signal View Waveform Analog Tools Window Help
1000
0
-1000
By:
x 10ps
0
2000

```
= G1
           clock                    1
         fd[11:0]           1100_1000
    path_img[15:0]     101_1000_0100   xxxx_x*  101_10*  1110_1*
   path_real[15:0]     100_1100_1110   xxxx_x*  100_11*  1111_0*
           reset                    0
          t[20:0]                1010     0      1010    1_0100
```

Fig 4.17: The demonstration of Xiao model with LUTs

Second, we check the results of Xiao model implemented by **CORDIC.** We can find that the result of real part "path_real"= 16'b0_0000_10011001110, which represents 0.57031, and "path_img"=16'b0_0000_10101111111, which represents 0.687256. While the real values are 0.60437 and 0.69197. The results are also verified.



<nWave:nWave:1> /home/user/rea/eras/Xiao1.fsdb
File Signal View Waveform Analog Tools Window Help
12000
0
-12000
By:
0
20000
40000

```
= G1
           clock                    1
          clock1                    1
         fd[11:0]           1100_1000
    path_img[15:0]      101_0111_1111     0     101_0111_11*  1111_0110_1*  1111_1001
   path_real[15:0]      100_1000_0010     0     100_1000_00*  111_1011_01*  1111_0011
           reset                    0
     enable[31:0]                   1
          t[20:0]            110_1110   1010    110_1110      1101_0010     1_0011_0
G2
```

Fig 4.18: The demonstration of Xiao model with CORDIC

The simulation results are shown below.

Performance / Algorithm	Data yield rate (per second)	Area (slice) Multiplier
Xiao with CORDIC	9 M	9594 (28%) 80(55%)
Xiao with LUTs	100 M	9522 (28%) 80(55%)

Table 4.13: Comparison of Xiao model with different triangular Algos

From Table 4.13, we can find that the model with LUTs generates channel coefficients with almost 100M per second, and Xiao model with LUTs occupied larger chip area compared with Jakes model with LTUs. And the model implemented by **CORDIC** is still slow.

The statistical property analyses of our channel simulator output are given in Figure 4.19, Figure 4.20, Figure 4.21, and Figure 4.22. The picture in the left is the idea value, and the picture in the right is our simulations. We took the ensemble averages of 20 sets of coefficient, M=8, the normalized $f_d\tau$ =0.025. Because in our simulator, the triangular functions are not precise, the curve does not fit the idea curves so well. But generally speaking, the results are right. Another drawback is the random phase produced by the Pseudo random generator, which in the equation should be uniformly distributed within 0 to 360 degrees. However, we generate a sequence of 18 random binary numbers, which is uniformly distributed within 0 to 4096 degree, that is a little non-uniform to the range of 0 to 360 equivalently. That might be one of the reasons that the second order statistic properties of our works doesn't perform pretty well compared with the idea curves.

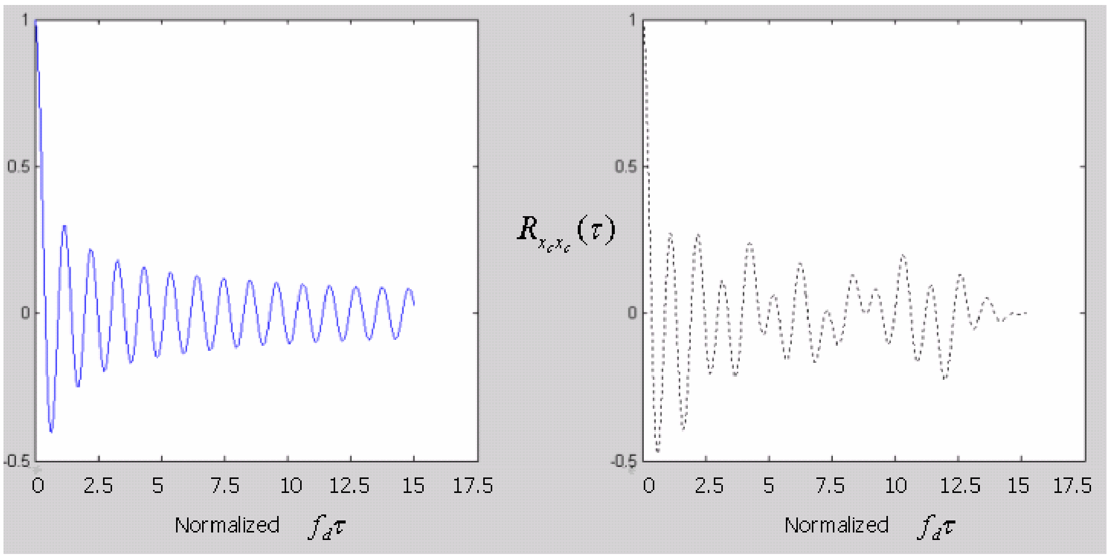


Fig 4.19: Autocorrelation of $X_c(t)$

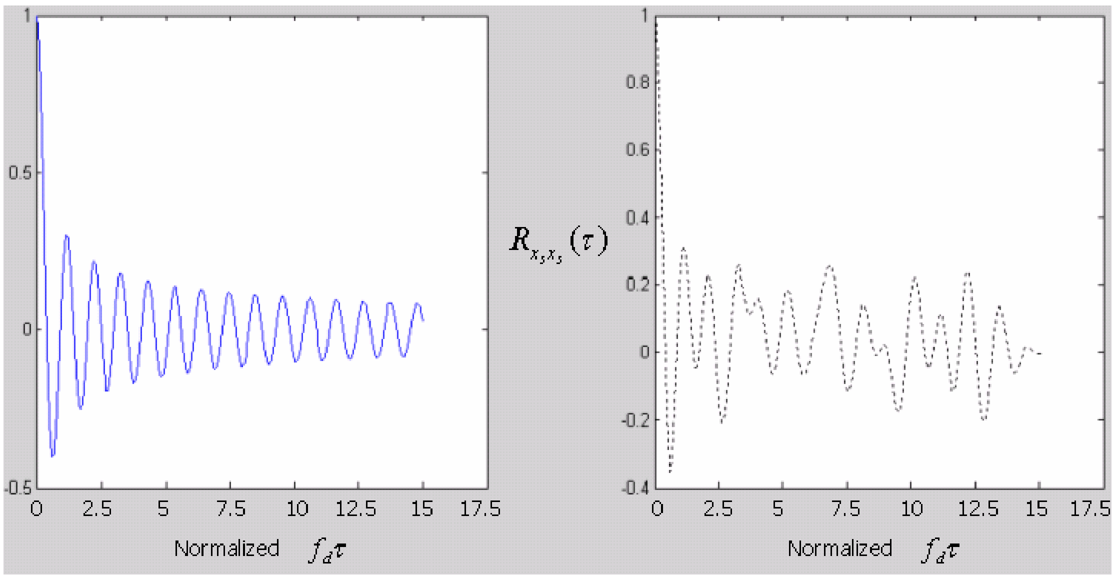


Fig 4.20: Autocorrelation of $X_s(t)$

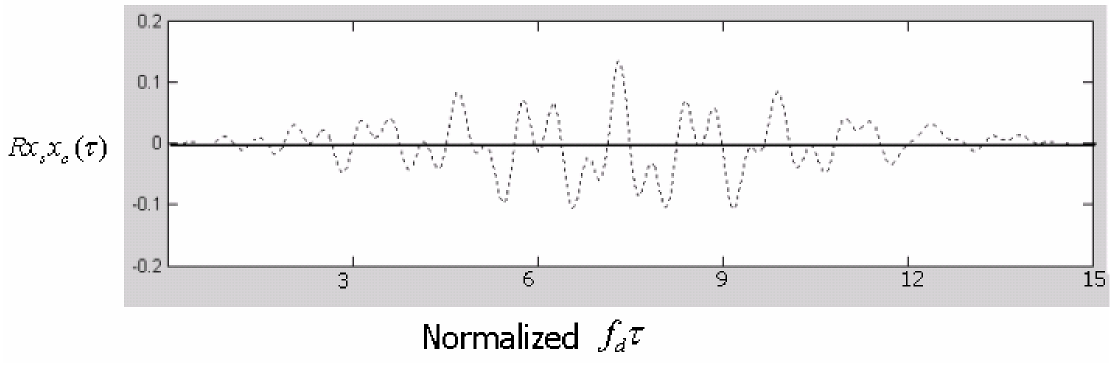


Fig 4.21: Crosscorrelation of $X_s(t)$ and $X_c(t)$

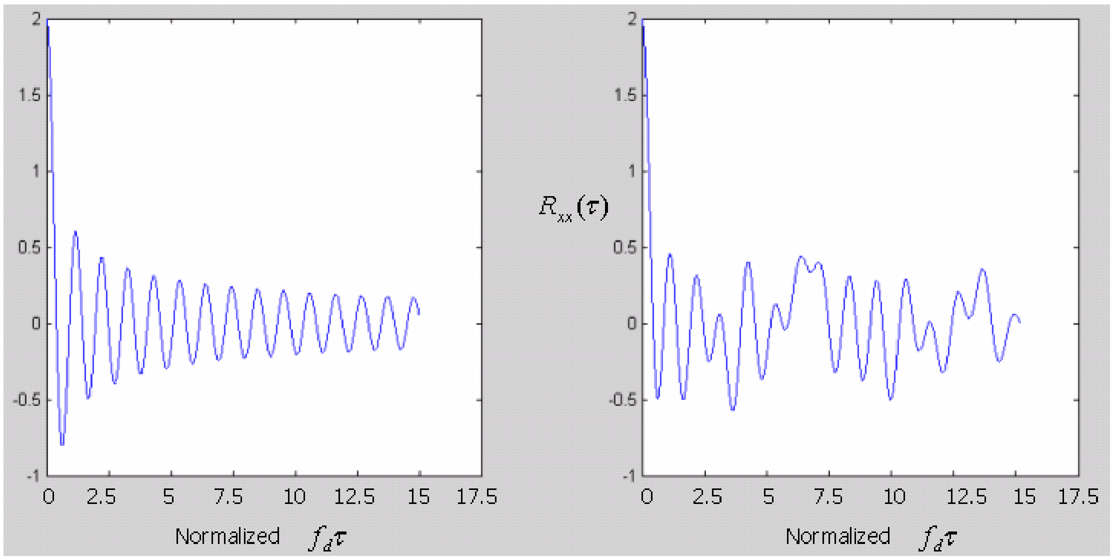


Fig 4.22: Real part autocorrelation of X(t)

For single channel simulation, Jakes model occupies smaller chip area and fewer multipliers, while Xiao model is almost 1.7 times larger. Both of them yield more than 100M data per second. Although Jakes model is less complex, Xiao model has good statistical properties. So, it depends on the limitation of hardware. In a smaller chip, Jakes model is more suitable, while a lager chip, Xiao model is the one we recommend.

4.3.2 Multipath Channel Simulator

In section 3.2.1, we choose a 6-pathes channel model for our system, which means that we have to implement a multipath channel simulator. In section 3.2.2, we have mentioned that Jakes model is a deterministic model, it is not the right model to generate multiple channels simultaneously without being modified. By introducing the random phase, Xiao model can generate uncorrelated multipath channels simultaneously. Thus, we just implement the channel coefficients generator with Xiao model.

We first rewrite equation (3.13) to (3.16) into a multipath channel mode:

$$
\begin{aligned}
X_k(t) &= X_{c,k}(t) + jX_{s,k}(t) \\
X_{c,k}(t) &= \frac{2}{\sqrt{M}} \sum_{n=1}^{M} \cos(\psi_{n,k}) \cdot \cos(w_d t \cos\alpha_{n,k} + \phi_k) \\
X_{s,k}(t) &= \frac{2}{\sqrt{M}} \sum_{n=1}^{M} \sin(\psi_{n,k}) \cdot \cos(w_d t \cos\alpha_{n,k} + \phi_k) \\
\alpha_{n,k} &= \frac{2\pi n - \pi + \theta_k}{4M}
\end{aligned}
\tag{4. 5}
$$

The parameters are defined in Table 4.14

	Random θ_k	**Random** ϕ_k	**Initial** $\psi_{n,k}$
Channel 1	-20	31	149.4375
Channel 2	30	22.5	23.65625
Channel 3	65	19.40625	101.75
Channel 4	-40	8.03125	130.71875
Channel 5	145	32.0625	21.84375
Channel 6	-100	30.968	237.6875

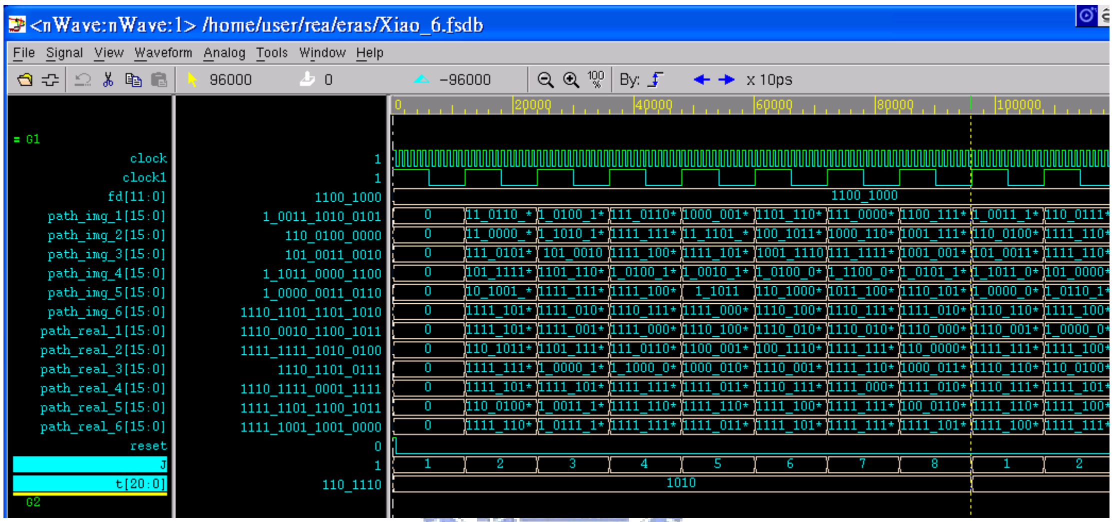
Table 4.14: Simulation parameters

From Table 4.13, we notice that the chip area occupied and the usage of multipliers is about 28% and 55% for Xiao model implemented by LUTs, 28% and 55% for Xiao model implemented by **CORDIC** as well. Thus, we can not just put six copied circuits on the chip, but sum the every $\cos(\varphi_n)\cdot\cos(w_d t\cos\alpha_n+\phi)$ for the real part and $\sin(\varphi_n)\cdot\cos(w_d t\cos\alpha_n+\phi)$ for the image part sequentially, giving each path a pseudo binary random number generator with different random seeds. After 8 clocks, we can get a set of six uncorrelated Rayleigh fading channel coefficients at a time. As for the Xiao model with Taylor series, the usage of multipliers exceeds the maximum number **Virtex-II XC2V6000** can offer, so we just skip it.

First, we show the results of the 6 uncorrelated Rayleigh fading channel with

CORDIC. The Pseudo Random seeds are the same with PN 1 to PN 6. And we set the Doppler shift to 200 Hz still, with input time $t = 4.768\times10^{-6}$. Note that, we still have two clocks control the circuit, the clock will smaller period is for **CORDIC**. And when the counter **J** counts to 8, the circuit will accumulate the right channel coefficients, but the values are shown in the next clock cycle.

Because we use the random generator implemented by ourselves, we can't calculate the idea values by computer, but calculate by ourselves. The results are listed in Table 4.15, which shows that, the circuit output values are very close to the idea values.



<nWave:nWave:1> /home/user/rea/eras/Xiao_6.fsdb
File Signal View Waveform Analog Tools Window Help
96000
0
-96000
By:
x 10ps

```
= G1
            clock                  1
           clock1                  1
          fd[11:0]         1100_1000
  path_img_1[15:0]    1_0011_1010_0101
  path_img_2[15:0]      110_0100_0000
  path_img_3[15:0]      101_0011_0010
  path_img_4[15:0]    1_1011_0000_1100
  path_img_5[15:0]    1_0000_0011_0110
  path_img_6[15:0]  1110_1101_1101_1010
 path_real_1[15:0]  1110_0010_1100_1011
 path_real_2[15:0]  1111_1111_1010_0100
 path_real_3[15:0]     1110_1101_0111
 path_real_4[15:0]  1110_1111_0001_1111
 path_real_5[15:0]  1111_1101_1100_1011
 path_real_6[15:0]  1111_1001_1001_0000
            reset                  0
                J                  1
          t[20:0]           110_1110
  G2
```

Fig. 4.23: The demonstration of Xiao model with CORDIC

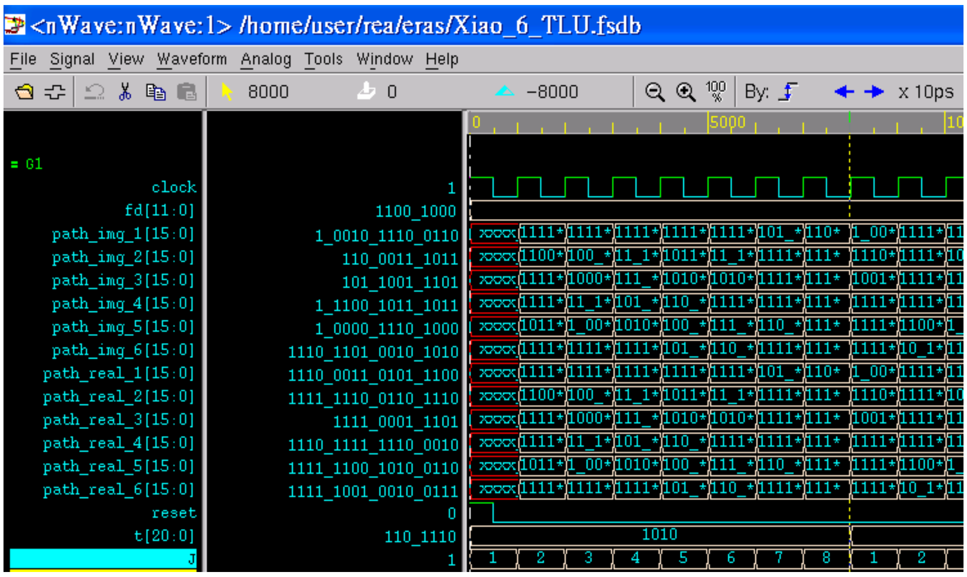
Results	Real Part	Out put	Difference	Image Part	Output	Difference
Channel 1	-3.6825	-3.6503	0.03210	2.45328	2.43752	0.01576
Channel 2	-0.0311	-0.04492	0.01382	0.78223	0.75312	0.02911
Channel 3	1.87265	1.85498	0.01767	0.62765	0.64843	0.02078
Channel 4	-2.2315	-2.19375	0.0378	3.42703	3.38085	0.04618
Channel 5	-0.2431	-0.27539	0.03229	2.09108	2.05273	0.03835
Channel 6	-0.7622	-0.80468	0.04248	-2.22121	-2.28125	0.06004

Table 4.15: Simulation results

Next, we demonstrate the result of the 6 uncorrelated Rayleigh fading channel with LUTs. The inputs and the parameters setting are all the same. And the results are confirmed.

Results	Real Part	Out put	Difference	Image Part	Output	Difference
Channel 1	-3.6825	-3.5795	0.1029	2.45328	2.36229	0.09099
Channel 2	-0.0311	-0.1958	0.1647	0.78223	0.77734	0.00486
Channel 3	1.87265	1.8891	0.0165	0.62765	0.70117	0.07352
Channel 4	-2.2315	-2.0127	0.2188	3.42703	3.50859	0.08156
Channel 5	-0.2431	-0.41797	0.1749	2.09108	2.10937	0.01829
Channel 6	-0.7622	-0.85546	0.0932	-2.22121	-2.35351	0.13230

Table 4.16: Simulation results



<nWave:nWave:1> /home/user/rea/eras/Xiao_6_TLU.fsdb
File Signal View Waveform Analog Tools Window Help
8000
0
-8000
By:
x 10ps

```
= G1
             clock                    1
          fd[11:0]            1100_1000
 path_img_1[15:0]     1_0010_1110_0110
 path_img_2[15:0]       110_0011_1011
 path_img_3[15:0]       101_1001_1101
 path_img_4[15:0]     1_1100_1011_1011
 path_img_5[15:0]     1_0000_1110_1000
 path_img_6[15:0]  1110_1101_0010_1010
path_real_1[15:0]  1110_0011_0101_1100
path_real_2[15:0]  1111_1110_0110_1110
path_real_3[15:0]       1111_0001_1101
path_real_4[15:0]  1110_1111_1110_0010
path_real_5[15:0]  1111_1100_1010_0110
path_real_6[15:0]  1111_1001_0010_0111
             reset                    0
           t[20:0]             110_1110
                 J                    1
```

Fig 4.24: The demonstration of Xiao model with LUTs

The simulation results are given below.

Performance / Algorithm	Data yield rate (each path per second)	Area (slice) Multiplier
Xiao_6 with CORDIC	1.25 M	8717(25%) 67 (46%)
Xiao_6 with LUTs	12.5M	8273(24%) 67 (46%)

Table 4.17: Comparison of Xiao multichannel models

From Table 4.17, we can clearly find that the Xiao multichannel model implemented by LUTs is 10 times faster than the one implemented by **CORDIC,** with even smaller chip area. Now we have a set of multichannel coefficients with the generating speed of 12.5M per second for each path.

4.4 Complexity and Performance

In this section, we try to analyze the complexity of this channel simulator and make a summery.

4.4.1 Analyses of Complexity

Generating Channel Coefficients

In section 3.3, we deduced the possible mobility supporting limitation for IEEE 802.16a. The limitation would be 150 Km/hr, the system operating frequency is 2 GHz, so the maximum Doppler shift will be

$$f_d = \frac{41.9 \times 2G}{c} = 279.32Hz$$

so, the coherence time will be

$$\tau_{chan} \approx \frac{1}{f_d} = 3580\mu s$$

and our OFDMA symbol time is $201.6\mu s$, which means every 17 symbol times, the channel response shall change. For each path,

$$(1/201.6\mu_s)/17 \approx 292$$

channel coefficients are needed to simulate the desired environment.

4.4.2 Realistic Performance

Generating Channel Coefficients

As the works shown in section 4.3.2, we implement a circuit which can yield channel coefficients at the rate of 12.5M per second each path for a 6 –ray fast fading channels, which is largely beyond the system's need. And this channel coefficients generator can be used for other fast fading channel simulation as well.

In this chapter, we introduced the system hardware implementation. Three triangular functions are compared and discussed. And we think LUTs is more suitable for channel simulator implementation, since it needs no multiplier and fast.

Two single-ray Rayleigh fading channel simulators are also implemented with different triangular function algorithms. Both channel simulators can yield more than 100 M channel coefficients per second. Jakes model is less complex, Xiao model has good statistical properties. So, it depends on the limitation of hardware. In a smaller chip, Jakes model is more suitable, while in a lager chip, Xiao model is the one we recommend.

Finally, a 6-ray fading channel simulator is made. The one implemented by LUTs is recommend, it can generate 12.5M channel coefficients per second each path.

Chapter 5
Conclusion

FPGA is a very powerful hardware for implementing the digital signal processing circuits. The programmable property for **FPGA** makes it flexible to different kind of circuits, we can implement the circuits for specific uses, design and optimize the circuit with its specialty. Just as we used a simplified LUTs with not much precision, and it is satisfactory for implementing our channel simulator, since the coefficients are statistical characterized.

We first introduced an **OFDM**-based system, IEEE 802.16a, including the background, **OFDM** basic ideas, and the specification. We also defined several parameters and picked a suitable channel model for our channel simulation. The possible mobility supporting rate limitation is discussed as well. The structure of the **DSP-FPGA** embedded card—**Quixote-III**— and the chip introduction were then given. Implementation considerations of our works and the performance of the simulator are explained. Three triangular function generator algorithms, two single-ray channel models and a 6-ray Rayleigh fading channel were discussed and compared.

An **FPGA** structured channel simulator was implemented finally. It can emulate a 6-ray Rayleigh fading channels environment and yield 12.5 M channel coefficients per second.

Reference

[1] I. Koffman and V. Roman, ”Broadband wireless access solutions based on OFDM access in IEEE 802.16,” *IEEE Commun. Mag.*, vol.40,no.4.pp.96-103,Apr.2002.

[2] http://*WirelessMan.org/pub/backgrounder.html*.

[3] IEEE 802.16

[4] IEEE Std 802.16a-2003,*IEEE Standard for Local and Metropolitan Area Networks*

[5] O. Edfors, et al. “An introduction to orthogonal frequency division multiplexing”, Sep, 1996

[6] R.W Chang, “Synthesis of band-limited orthogonal signals for multichannel data transmoission” *Bell System Tech. J.*,45:1775-1796,Dec 1966.

[7] B.R Salzberg, “Performance of an efficient parallel data transmission system”, *IEEE Trans. Commun*., COM-15(6):805-811, Dec 1967.

[8] S.B. Weinstein and P.M. Ebert. “Data transmission by frequency-division multiplexing using the discrete Fourier transform.” *IEEE Tans. Commun.*, COM-19(5): 628-634, Oct. 1971.

[9] A. Peled and A .Ruiz. “Frequency domain data transmission using reduced computational complexity algorithems.” *In Proc. IEEE Int. Conf. Acout., Speech ,Signal Processing pages*, 964-967, Denver, Co, 1980

[10] Radio broadcasting systems; Digital Audio Broadcasting (DAB). ETS 300 401, ESTI European Telecommunications Standards Institute, Valbonne, France, Feb. 1995

[11] http://www.hiperlan2.com/

[12] http://www.dvb.org/

[13] IEEE Std 802.11b-1999,*IEEE Standard for Local and Metropolitan Area Networks*

[14] IEEE Std 802.11g-1999,*IEEE Standard for Local and Metropolitan Area Networks*

[15] IEEE Std 802.11a-1999,*IEEE Standard for Local and Metropolitan Area Networks*

[16] A.Oppenheim and R.Schafer. “*Discrete-time signal processing*.” Prentice-Hall,

1989

[17] V. Ereg. "802.20 Presentation on Channel Modeling." *IEEE C802.20-03/09*

[18] T.S. Rappaport, Wireless Communications: *Principles and Practice*. New Jersey: Prentice Hall,1996.

[19] ETSI TR101 112 v.3.2.0(1998-04). "*Universal Mobile Telecommunications Systems(UMTS)*; *Selection procedures for the choice of radio transmission technologies of the UMTS.*

[20] R.H. Clarke "A statistical theory of mobile-radio reception" *Bell Syst Tech.J* pp. 957-1000, 1968

[21] M.J. Gans. " A power spectral theory of propagation in the mobile radio environment." *IEEE Transactions on Vehicular Technology*, Vol. VT-21,pp. 27-38, Feb 1972

[22] W.C. Jakes. "Microwave Mobile Communications " *John Wiley*,1974

[23] I.S. Gradshteyn and I.M. Ryzhik, "Table of Integers, Series, and Products, 6th ed," A. Jeffery, Ed. New York: Acadamic, 2000

[24] P. Dent, G.E. Bottomley, and T. Croft, " Jakes fading model revisited," Electron. Lett., vol. 11,pp. 1162-1163, June 1993

[25] M. Patzold, U. Killat, F. Laue, and Y. Li, " On the statistical properties of deterministic simulation models for mobile fading channels," *IEEE Trans. Veh. Technol.*, vol 47,pp.254-269, feb,1998

[26] Y.X. Li and X. Hunan," Statistical properties of Jakes' fading channel simulator," *in Proc. IEEE ICC'00*, 2000, pp. 41-45

[27] Y.B. Li and Y.L Guan," Modified Jakes model for simulating multiple uncorrelated fading waveforms," *in Proc. IEEE ICC'00*, 2000, pp.46-49.

[28] M. Patzold, F. Laue. " Statistical properties of Jakes' fading channel simulator," *in Proc. IEEE VTC'98*, 1998,pp. 712-718

[29] M.F. Pop and N.C, Beaulieu, "Limitations of sum-of-sinusoidals fading channel simulators," *IEEE Tans, Commun.*,vol. 49,pp. 699-708, Apr. 2001

[30] Y.R. Zheng and C. Xiao, " Simulation models with correct statistical properties for Rayleigh fading channels," *IEEE Tans, Commun.*,vol.51, No. 6, June 2003

[31] IEEE C 802.20-03/15r1

[32] W. Henkel, et. Al., “The cyclic prefix of OFDM/DMT- an analysis,” *IEEE 2002 Int’l Zurich Seminar on Broadband Comm*, Feb. 19-21, ETH Zurich,SW.

[33] Ziemer and Tranter, “Principles of Communications”, Ed 4th. John Wiley,1995

[34] “Quixote user’s manual “, Innovative Integration, Dec, 2003

[35] “TMS320CC64x Technical overview”, TI, SPRU395B, Jan, 2001

[36] “Virtex II platform FPGA user guide”, Xilinx, UG002 (v1.7) Feb,2004

[37] Theodore S. Rappaport, “Wireless communications principles and practice”, 2nd ed, Prentice Hall, 2002

[38] Angus E. Taylor, “Advanced Calculus”, John Wiley, 1983

[39] Jack E. Volder, “The CORDIC Trigonometric Computing Technique”, IRE Transactions on Electronic Computers, September 1959.

作者簡歷

余子瀚，男，一九七九年出生於台灣省屏東。二零零二年畢業於國立交通大學電子工程學系。同年九月，升入國立交通大學電子研究所碩士班，在通訊電子與訊號處理實驗室中從事無線通訊相關領域之研究。二零零四年六月取得碩士學位，論文題目爲“可程式化閘陣列之快速瑞立衰褪通道模擬器 ”。研究範圍和興趣包括 ：無線通道模擬、通道估測、行銷管理等。