

國立交通大學

電信工程學系碩士班
碩士論文

基於獨特碼架構之頻域等化單載波系統
之 FPGA 實現



FPGA Realization of a Unique Word Based
Single Carrier System
with Frequency Domain Equalization

研究生：陳欣瑤

Student: Hsin-Yao Chen

指導教授：李大嵩 博士

Advisor: Dr. Ta-Sung Lee

中華民國九十六年六月

基於獨特碼架構之頻域等化單載波系統之 FPGA 實現

FPGA Realization of a Unique Word Based
Single Carrier System
with Frequency Domain Equalization

研究生：陳欣瑤

Student: Hsin-Yao Chen

指導教授：李大嵩 博士

Advisor: Dr. Ta-Sung Lee

國立交通大學

電信工程學系碩士班

碩士論文



A Thesis

Submitted to Institute of Communication Engineering
College of Electrical Engineering and Computer Science
National Chiao Tung University

in Partial Fulfillment of the Requirements

for the Degree of
Master of Science

in

Communication Engineering

June 2007

Hsinchu, Taiwan, Republic of China

中華民國九十六年六月

基於獨特碼架構之頻域等化單載波系統 之 FPGA 實現

學生：陳欣瑤

指導教授：李大嵩 博士

國立交通大學電信工程學系碩士班

摘要

頻域等化之單載波系統(SC-FDE)在新一代之無線通訊系統中佔有相當關鍵性的地位，它不但能夠達到和正交分頻多工(OFDM)系統相當近似之效能和效率，亦同樣具有低訊號處理複雜度之優點。另一方面，SC-FDE 並不會面臨 OFDM 所面對之高峯值對均值功率比(PAPR)問題，因此在 IEEE 802.16 標準中，SC-FDE 成為了於 OFDM 以外另一實體層技術的選擇。在本論文中，吾人將使用自行研發之平台，實現一基於獨特碼(UW)架構之單載波系統，其中基頻演算法部分將實現於平台之 Xilinx Virtex-II FPGA 模組。在此系統中之演算法除包括通道估計器、頻率修正器、迴旋碼解碼器等之外，吾人也實現了基於獨特碼架構下之相位追蹤器，使得此一單載波系統之功能性更加完整。此外，吾人更進一步提出此一獨特碼架構於相位追蹤以外之應用，包括於時脈偏移下用以修正 FFT 窗之偏移，以及於移動之環境下用以更新通道之估計等。最後，本篇論文中將說明獨特碼的架構不但在理論上十分簡單，實際上也十分適合用於硬體平台的實現。


FPGA Realization of a Unique Word Based Single Carrier System with Frequency Domain Equalization

Student: Hsin-Yao Chen

Advisor: Dr. Ta-Sung Lee

Department of Communication Engineering
National Chiao Tung University

Abstract

The logo of National Chiao Tung University is a circular emblem. It features a gear-like outer border. Inside the circle, there is a stylized building or structure with the letters 'ES' and 'A' on it. Below the building, the year '1896' is inscribed. The entire logo is rendered in a blue color.

In recent years, Single Carrier System with Frequency Domain Equalization (SC-FDE) becomes a key technology in the development of new wireless communication systems. It has similar performance, efficiency as well as low signal processing complexity advantages as orthogonal frequency division multiplexing (OFDM), but does not suffer from the high peak to average power ratio (PAPR) problem as in OFDM system. Therefore, SC-FDE has been adopted by IEEE 802.16 standard as an alternative technique to OFDM in the physical layer. In this thesis, we propose a solution for building up a Unique-Word (UW) based SC-FDE system on a self-designed platform with Xilinx Virtex-II FPGA module mounted. In addition to channel estimator, frequency offset compensator and convolutional encoder, a UW-based phase offset tracker is realized to make the functionalities of the system more complete. Moreover, other applications of UW structure are presented, including FFT window synchronization and update of channel estimation in mobile environment, etc. Finally, we will show that the UW structure is not only theoretically simple, but also practically suitable for hardware implementation.

Acknowledgement

First, I am very grateful to my advisor, Dr. Ta-Sung Lee, for his enthusiastic guidance especially the training of oral presentation and being earnest in our works. I would also thanks to Jeff Tsai who gives me a lot of technical support on the circuit design of self-designed platform, and Yen-Yu Chen who is always patient with my consultation. Special thanks to Lih-Gong Wu, whose priceless comments and invaluable suggestions are indispensable to the completion of this thesis. Heartfelt thanks are also offered to all members in the Communication Signal Processing and System Design (CSPSD) Lab for their constant encouragement and help.

At last but not least, I would like to express my deepest gratitude to my family for their endless love and the support all the way from the very beginning of my postgraduate study, especially my mom for her tender encouragement, and my dad as a constant reminder of health.



Contents

Chinese Abstract	I
English Abstract	II
Acknowledgement	III
Contents	IV
List of Figures	VII
List of Tables	X
Acronym Glossary	XI
Abbreviation Glossary	XIV
Chapter 1 Introduction.....	1
Chapter 2 SC-FDE Baseband Transceiver Architecture.....	4
2.1 Overview of SC-FDE System.....	4
2.2 Transmitter Architecture	6
2.2.1 Convolutional Encoder	7
2.2.2 Mapper / De-mapper	8
2.2.3 Unique Word Structure	8
2.2.4 Preamble Channel and Frame Structure	10
2.2.5 Upsampler and Root Raised Cosine Filter.....	11
2.3 Receiver Architecture.....	14
2.3.1 Timing Synchronization.....	15
2.3.2 Frequency Synchronizer	21



2.3.3 Channel Estimator.....	22
2.3.4 Phase Estimator.....	23
2.3.5 Viterbi Decoder.....	25
2.4 Summary.....	27
Chapter 3 SC-FDE System Platforms	28
3.1 Self-designed Platform.....	28
3.1.1 RF Module.....	30
3.1.2 AD and DA Modules.....	32
3.1.3 MAC/BB Platform.....	33
3.1.4 USB Interface.....	35
3.2 Benefits of using VHDL.....	36
3.3 FPGA Design Flow.....	37
3.4 Debugging Tools.....	40
3.5 Summary.....	42
Chapter 4 SC-FDE System Realization.....	43
4.1 Design Flow.....	43
4.2 MATLAB Verification.....	44
4.3 FPGA Realization.....	50
4.3.1 Design Principles.....	51
4.3.2 Circuit Design.....	53
4.4 ModelSim Simulation.....	69
4.5 Experimental Results.....	71
4.6 Summary.....	73
Chapter 5 Other Applications on Unique Word Structure in	
SC-FDE System.....	74
5.1 Cyclic Prefix versus Unique Word.....	75

5.1.1 Comparison of CP and UW in Terms of Bandwidth Efficiency and BER Behaviour	76
5.2 Application of the Unique Word Structure.....	77
5.2.1 Synchronization	77
5.2.2 Channel Estimation.....	79
5.3 Simulation Results	83
5.4 Circuit Design of Proposed Methods.....	85
5.5 Summary.....	87
 Chapter 6 Conclusion	 88
 Bibliography.....	 90



List of Figures

Figure 2.1: Transmitter architecture of SC-FDE system.....	7
Figure 2.2: Convolutional encoder with code rate 1/3 and constraint length 5	7
Figure 2.3: QPSK, 16-QAM, and 64-QAM constellations.....	8
Figure 2.4: Transmitted block using the concept of UW	9
Figure 2.5: Training sequence and frame structure of IEEE 802.11a standard	11
Figure 2.6: Diagram of upsampler and pulse shaping filter.....	12
Figure 2.7: Evolution of the polyphase filter	13
Figure 2.8: Noble identity	13
Figure 2.9: Receiver architecture of SC-FDE system.....	14
Figure 2.10: Double sliding window packet detection	15
Figure 2.11: Block diagram of delay-locked loop.....	16
Figure 2.12: Detection set of delay-locked-loop.....	17
Figure 2.13: Oversampled waveform with the correct sample points	19
Figure 2.14: Original and new detection set in proposed DLL algorithm	19
Figure 2.15: Proposed DLL algorithm Flow.....	20
Figure 2.16: Constellation diagram of one equalized frame.....	23
Figure 2.17: Trellis diagram part 1.....	26
Figure 2.18: Trellis diagram part 2.....	27
Figure 3.1: Development environment of self-designed platform	29
Figure 3.2: Main board of self-designed platform	29
Figure 3.3: RF module on self-designed platform	31
Figure 3.4: Measured carrier spectrum form RF module.....	31
Figure 3.5: AD/DA module on self-designed platform.....	33
Figure 3.6: Measured data waveform from AD/DA module	33
Figure 3.7: MAC/BB platform.....	35
Figure 3.8: USB module on self-designed platform	36
Figure 3.9: FPGA design flow	38
Figure 3.10: Spectrum analyzer block diagram	41
Figure 3.11: Vector signal analyzer block diagram.....	41
Figure 4.1: FPGA design flow	44
Figure 4.2: Impulse and frequency response of RRC filter with $\beta=0.25$	45
Figure 4.3: (a) Original waveform (b) RRC shaped waveform on transmitter (c) RRC shaped waveform on receiver.....	46

Figure 4.4: Eye diagram of RRC shaped waveform	46
Figure 4.5: (a) Received oversampling waveform (b) DLL selected samples on receiver (c) Original samples on transmitter.....	47
Figure 4.6: Real and estimated channel frequency response	48
Figure 4.7: (a) One equalized block in the presence of residual CFO (b) One equalized block after the proposed phase tracking algorithm (c) One equalized block after the lower-complexity phase tracking algorithm...	49
Figure 4.8: System performance of SC-FDE and OFDM.....	50
Figure 4.9: Circuit design of transmitter	53
Figure 4.10: Circuit design of convolutional encoder.....	54
Figure 4.11: Circuit design of mapper	55
Figure 4.12: Circuit design of de-mapper	55
Figure 4.13: Circuit design of preamble generator	56
Figure 4.14: Circuit design of Unique Word generator.....	57
Figure 4.15: Circuit design of multiplexer.....	58
Figure 4.16: Circuit design of polyphase filter	59
Figure 4.17: Circuit design of receiver side.....	59
Figure 4.18: Circuit design of double sliding window packet detection method.....	60
Figure 4.19: Circuit design of the delay-locked loop.....	61
Figure 4.20: Circuit design of the frequency offset compensation block	62
Figure 4.21: Circuit design of fast Fourier transform	63
Figure 4.22: Conventional channel estimation and equalization strategy.....	64
Figure 4.23: Modified channel estimation and equalization strategy	64
Figure 4.24: Circuit design of channel equalizer	65
Figure 4.25: Circuit design of phase offset compensator.....	66
Figure 4.26: Circuit design of Viterbi decoder.....	67
Figure 4.27: Circuit design of branch metric generator	67
Figure 4.28: Circuit design of add, compare, and select block.....	68
Figure 4.29: SC-FDE transmitter ModelSim simulation result	69
Figure 4.30: SC-FDE receiver ModelSim simulation result.....	70
Figure 4.31: Transmitted waveform of SC-FDE system.....	70
Figure 4.32: Self-designed platform development environment.....	71
Figure 4.33: Self-designed platform experimental result: source data and detected data waveform on LA	72
Figure 5.1: Single Carrier with (a) Cyclic Prefix and (b) Unique Word.....	76
Figure 5.2: Synchronization and tracking of the FFT-window	78
Figure 5.3: Basic UW block structure.....	79

Figure 5.4: Comparison of the proposed preamble based channel estimation and
UW-update channel estimation84

Figure 5.5: Circuit design of the UW-based synchronizer85

Figure 5.6: Circuit design of the UW-based channel estimator86



List of Tables

Table 2-1: Main PHY Parameters of the Investigated SC-FDE System	6
Table 2-2: Proposed phase tracking algorithm	24
Table 2-3: State transition table	26
Table 4-1: Relative Resource consumption of the SC-FDE system at the transmitter	72
Table 4-2: Relative Resource consumption of the SC-FDE system at the receiver	73
Table 4-3: Time consumption of Synthesis and P&R in SC-FDE system	73
Table 5-1: Summary of simulated SC-FDE systems	84



Acronym Glossary

1G	first generation
ADC	analog to digital converter
AMPS	advanced mobile phone services
ASIC	application specific integrated circuit
AWGN	additive white Gaussian noise
BER	bit error rate
BMG	branch metric generator
BPSK	binary phase shift keying
BRAN	broadband radio access network
CDMA	code division multiple access
CIR	channel impulse response
CP	cyclic prefix
CPLD	complex programmable logic device
CPU	central processing unit
CSI	channel side information
DAC	digital to analog converter
D-AMPS	digital AMPS
DFP	delay flip-flop
DFT	discrete Fourier transform
DLL	delay-locked loop
DSP	digital signal processor
EDA	electronic design automation
ETSI	European Telecommunications Standards Institute
FDE	frequency domain equalization
FEC	forward error correction
FFT	fast Fourier transform
FIFO	first in, first out

FIR	finite impulse response
FPGA	field programmable gate array
FSM	finite state machine
GSM	global system for mobile communications
HDL	hardware description language
HiperMAN	high-performance MAN
I/O	input/output
IBI	inter-block interference
ICI	inter-carrier interference
IEEE	institute of electrical and electronics engineers
IF	intermediate frequency
ISI	inter-symbol interference
JTAG	joint test action group
LA	logic analyzer
LED	light emitting diode
LSB	least significant bit
LTE	long term evolution
LUT	look up table
MAC	media access control
MAN	metropolitan area network
MLSE	maximum likelihood sequence estimation
NLOS	non-line-of-sight
OBW	occupied bandwidth
OFDM	orthogonal frequency division multiplexing
OSC	oscillator
OTP	one time programmable
PAPR	peak-to-average power ratio
PCI	process control interface
PHY	physical layer
PLD	programmable logic device



PLL	phase-locked loop
QAM	quadrature amplitude modulation
QPSK	quaternary phase shift keying
RAM	random access memory
RF	radio frequency
ROM	read-only memory
RRC	root raised cosine
RTL	register transfer level
SC	single carrier
SNR	signal to noise ratio
SoC	system on a chip
SOHO	small office/home office
SRAM	static random access memory
USB	universal serial bus
UTMI	USB 2.0 transceiver macrocell interface
UTRA	universal telecommunication radio access
UW	unique word
VCO	voltage-controlled oscillator
VHDL	very high speed integrated circuit hardware description language
VSA	vector signal analyzer
WCDMA	wideband CDMA
WLAN	wireless local area network

Abbreviation Glossary

BB	baseband
BPF	bandpass filter
RBW	resolution bandwidth



Chapter 1

Introduction

Wireless communications is one of the most active areas of technology development of our time. Similar to the developments in wired line capacity in the 1990s, the demand for new wireless capacity is growing at a very rapid pace. From the first-generation (1G) radio systems developed in the 1970s and 1980s, transmitting voice over radio by analog communication techniques such as Advanced Mobile Phone Services (AMPS), to the 2G systems built in the 1980s and 1990s, featuring the adoption of digital technology such as Global System for Mobile Communications (GSM), Digital-AMPS (D-AMPS) and code division multiple access (CDMA), and further to today's 3G wideband CDMA (WCDMA) technologies, whose transmission data rate can be up to 2 Mbps in good conditions. Driven by the transformation of a medium supporting voice telephony into a medium that is demanded to support other services such as the transmission of video, images, text, and data, future wireless system must provide high data rate services to satisfy the increasing needs of the next-generation wireless networks. Recent air interface standards for such wideband wireless metropolitan area network (MAN) systems are being developed by the IEEE 802.16 working group and also by the European Telecommunications Standards Institute (ETSI) Broadband Radio Access Network (BRAN) High-Performance MAN (HiperMAN) group. Such systems are installed to operate over non-line-of-sight (NLOS) links, serving residential and small office/home office (SOHO) subscribers with high data rate transmission.

However, as the bit rate increases, the problem of inter-symbol interference (ISI) becomes more serious. The above wireless access systems in residential neighborhoods must cope with the dominant propagation impairment of multipath which causes multiple echoes of the transmitted signal to be received with delay spreads of up to tens of microseconds. For bit rates in the range of tens of megabits per second, this translates to inter-symbol interference that can span up to 100 or more data symbols. For example, at a 5 MHz symbol rate, a 20 μ s multipath delay profile spans 100 data symbols. This raises the question of what types of anti-multipath measures are necessary, and consistent with low-cost solutions.

Several variations of orthogonal frequency-division multiplexing (OFDM) have been proposed as effective anti-multipath techniques [1]-[4], primarily because of the favorable trade-off they offer between performance in severe multipath and signal processing complexity. However, it is shown that when combined with frequency domain equalization (FDE), the single-carrier (SC) approach delivers performance similar to OFDM, with essentially the same overall complexity [5]-[7]. In addition, SC modulation uses a single carrier, instead of the many typically used in OFDM, so the peak-to-average transmitted power ratio for SC-modulated signals is smaller [8]-[10]. This in turn means that the power amplifier of an SC transmitter requires a smaller linear range to support a given average power (in other words, requires less peak power backoff). As such, this enables the use of a cheaper power amplifier than a comparable OFDM system; and this is a benefit of some importance, since the power amplifier can be one of the more costly components in a consumer broadband wireless transceiver. Therefore, single carrier with frequency domain equalization (SC-FDE) has been adopted by IEEE 802.16 standard to be one of the three modes as an alternative technique of OFDM in physical layer, and it is also currently a working assumption for uplink multiple access scheme in 3GPP Long Term Evolution (LTE), or Evolved UTRA. These show its potential of being an important candidate for future mobile wireless systems.

Moreover, a novel approach considering phase tracking algorithms for SC-FDE systems, which make use of the concept of Unique Word (UW) blockwise extension instead of the classical concept of cyclic prefix (CP) like it is used in OFDM is

provided [11]-[13]. The UW is a very simple known sequence that are distributed along the frame, which in this paper is focused on its performance on phase tracking algorithms as a counterpart of the use of pilot tones for IEEE 802.11a like OFDM systems [14]. Besides, with the UW-based structure, the algorithm developer has a lot of freedom for the tasks of synchronization [15]-[17], channel estimation [18]-[19] or noise prediction [20], which we will discuss in Chapter 5.

The goal of this thesis is to realize a Single-Carrier system with Frequency-Domain Equalizer on field programmable gate array (FPGA)-based platforms, where we intend to verify the above-mentioned algorithms on a self-designed platform. The complete functional blocks in both the transmitter and receiver are provided, and the associated algorithms applied in each functional block are also presented. After giving an overview of system architecture, we propose a total solution to build up FPGA-based platforms for realizing the SC-FDE system, including MATLAB verification and FPGA realization. The developed system contains a baseband transmitter, a digital-analog converter, an analog-digital converter, and a baseband receiver.

The organization of this thesis is as follows. Chapter 2 describes the proposed SC-FDE transceiver architecture and its corresponding schemes. In Chapter 3, the development environments of the proposed self-designed platform are introduced. In Chapter 4, the overall system realization is presented, and the performance evaluation is also included. Later, a further discussion of the UW-based SC-FDE system will be provided in Chapter 5. Finally, we make our concluding remarks in Chapter 6.

Chapter 2

SC-FDE Baseband Transceiver Architecture

This chapter focuses on the SC-FDE baseband transceiver architecture. An overview of the SC-FDE system will first be given. Then we divide the developed architecture into transmitter and receiver, and provide functional descriptions and associated algorithms for each block. In addition, the modified delay-locked-loop algorithm for timing recovery and the Unique Word based phase tracking algorithm adopted on the system will be described.

2.1 Overview of SC-FDE System

Recently, SC-FDE system has received a lot of attention as an attractive alternative solution for the problem of ISI in the wideband wireless system. Compared to the time-domain equalization that requires one or more transversal filters with the tap number covering the maximum channel impulse response length, FDE outperforms the conventional time-domain equalization and requires less complexity by using fast Fourier transform (FFT) and is more suitable for long channels. With such a comparably lower complexity similar to OFDM, SC-FDE does not suffer high peak-to-average power ratio (PAPR) as well as sensitivity to frequency and phase offsets as they are in the OFDM systems. Previous work [5][7] has also shown that FDE has better performance than OFDM systems in uncoded and high coding rate systems.

In our system we investigate a SC-FDE system, where the parameters are adapted to the IEEE 802.11a OFDM based wireless LAN (WLAN) standard. The transmission format at the physical layer (PHY) is frames consisting of a preamble part and the data payload. The preamble comprises ten short preambles and two long preambles, which are used for different synchronization tasks at the receiver. The data payload consists of six SC-FDE frames, each appended with a known sequence, called Unique Word, and six frames together are attached after the preamble. Moreover, the UW is cyclically extended in the guard time, resulting in the cyclic-prefix-like structure and equalization criterion can be easily achieved in the frequency domain. The detailed structure of preamble will be discussed in Section 2.2.4.

Parameters to be synchronized are the temporal position of the transmission frame, the carrier frequency, the clock frequency, and the temporal position of the FFT-window. Additionally, an estimate of the channel transfer function is needed. The effect of carrier phase offsets and clock phase offsets on the system performance is compensated for by channel equalization. After the preamble based carrier frequency synchronization there will always be some remaining carrier frequency offset, which causes phase errors in the received and equalized data symbols. In IEEE 802.11a systems, pilot subcarriers are used to estimate and track these phase errors. In this project we show, that simple UW-based phase tracking algorithms provide almost optimum performance in SC-FDE systems, and the main parameters of the investigated SC-FDE system are shown in Table 2.1.

We summarize the advantages of SC-FDE as follows:

- SC modulation has reduced peak-to-average ratio requirements from OFDM, thereby allowing the use of less costly power amplifiers
- Performance of SC-FDE system is similar to that of OFDM system, even for very long channel delay spread
- Frequency domain receiver processing has a similar complexity reduction advantage to that of OFDM: complexity is proportional to log of multipath spread

- Coding, while desirable, is not necessary for combating frequency selectivity, as it is in non-adaptive OFDM
- SC modulation is a well-proven technology in many existing wireless and wired-line applications, and its radio-frequency (RF) system linearity requirements are well known

Table 2-1: Main PHY Parameters of the Investigated SC-FDE System

Number of frames/packet	6
Number of total symbols/frame	64
Number of data symbols/frame	48
Number of Unique Word symbols/frame	16
FFT Size	64
Long Preamble Size (symbols)	64+16(CP)
Short Preamble Size (symbols)	16
Modulation Schemes	QPSK
Coding Rates	1/3
Pulse Shaping	RRC($\alpha=0.25$)

2.2 Transmitter Architecture

The baseband SC-FDE transmitter architecture is shown in Figure 2.1. The source data is first fed into the channel encoder, e.g., using the convolution code for error correction at the receiver. After coding, the binary values are converted into Quadrature Phase Shift Keying (QPSK) values with a mapper, and a guard period is added between successive frames. The insertion of a guard period anticipates the blockwise processing needed in the receiver when using FFT operations. The guard period will then be filled up with a known, simple UW sequence with the UW generator, and every six UW-appended frames will be preceded by the preamble with the preamble generator. The UW and preamble are used for certain synchronization purposes and the detail functionalities will be discussed in Section 2.2.3. After pulse shaping (Root Raised Cosine Pulses) and digital-to-analog conversion the resulting I/Q signals are up-converted onto the desired frequency band.

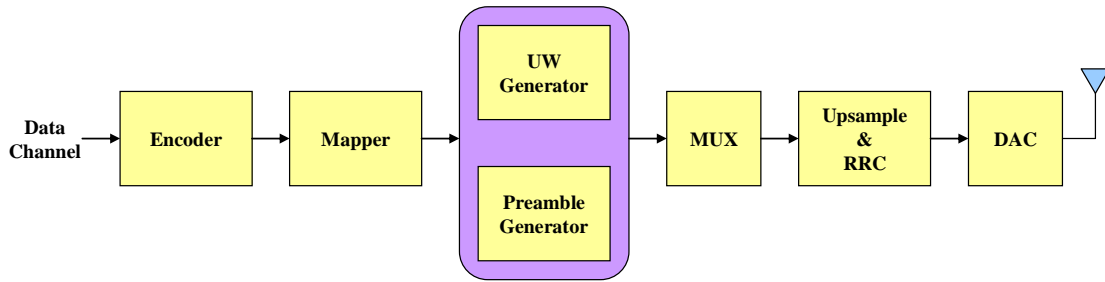


Figure 2.1: Transmitter architecture of SC-FDE system

2.2.1 Convolutional Encoder

A convolutional encoder typically will generate two or three output bits for each input bit. The output bits are dependent on the current input bit, as well as the state of the encoder. The state of the encoder is represented by several bits which precede the current bit. Figure 2.2 shows a convolutional encoder adopted in our system with code rate equal to $1/3$ and constraint length equal to 5. Convolutional coding adds redundant bits in such a way that the decoder can, within limits, detect errors and correct them.

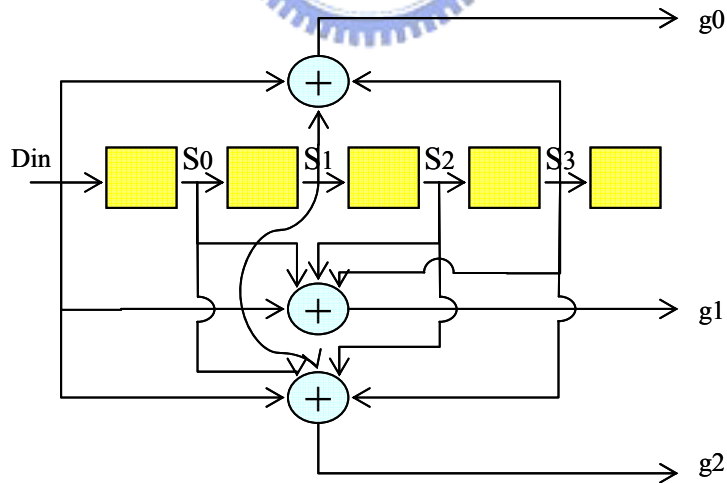


Figure 2.2: Convolutional encoder with code rate $1/3$ and constraint length 5

2.2.2 Mapper / De-mapper

Quadrature amplitude modulation (QAM) is the most popular type of modulation scheme since the rectangular constellations are easy to implement as they can be split into independent in-phase and quadrature parts. A mapper is used to map a small group of bits into a symbol according to the rectangular constellation adopted. Figure 2.3 shows the rectangular constellations of QPSK, 16-QAM, and 64-QAM. The higher modulation order the mapper adopts, the more information a symbol can carry, yet higher modulation order always suffers from interference more severely. In our system, we adopt QPSK as our modulation scheme.

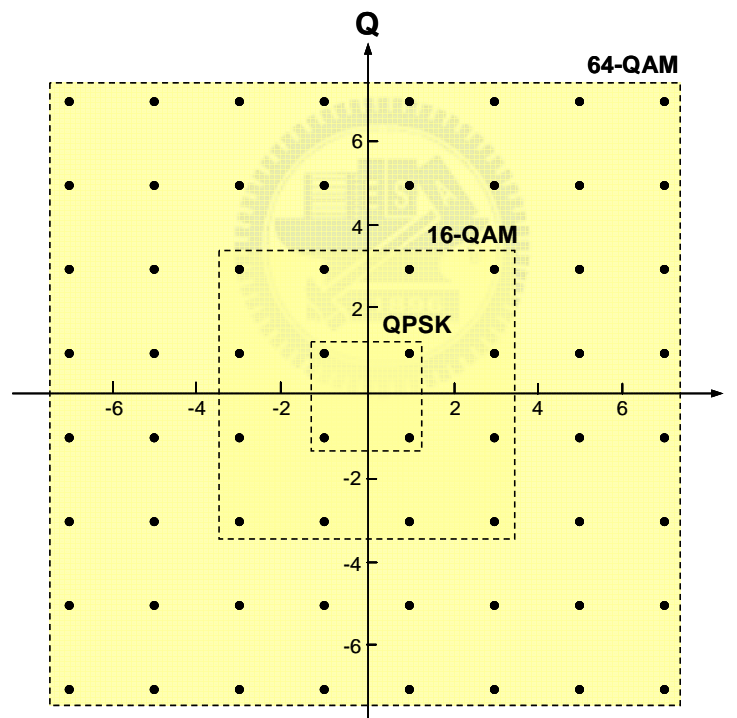


Figure 2.3: QPSK, 16-QAM, and 64-QAM constellations

2.2.3 Unique Word Structure

Frequency domain equalization for single carrier system is based on the equivalence between the convolution of two sequences in the time domain and the product of their Fourier transforms. Because of the use of FFT operations, the

received signals have to be processed blockwise. Therefore, performing a blockwise transmission and inserting a Cyclic Prefix (CP) between successive blocks is necessary, since the cyclic extension enables the circular convolution of the transmitted frame and the channel impulse response. The conventional CP structure, however, is less useful for other purpose like synchronization as long as the content of the CP is not known and varies with every single frame. The overhead induced by the CP could be used in a more efficient way if its content would be known before and could be chosen in a proper way. Therefore, instead of the cyclic prefix, a known sequence called Unique Word (UW) is part of every processed frame [11]-[13].

First of all, Figure 2.4 depicts the structure of one transmitted frame, which consists of the original data sequence of N_S symbols and the guard interval with N_G symbols. The overall duration of one frame with $N = N_S + N_G$ symbols is

$$T_{FFT} = NT = (N_S + N_G)T \quad (2.1)$$

where T is the symbol duration and let $T_G = N_G T$ be the guard interval.

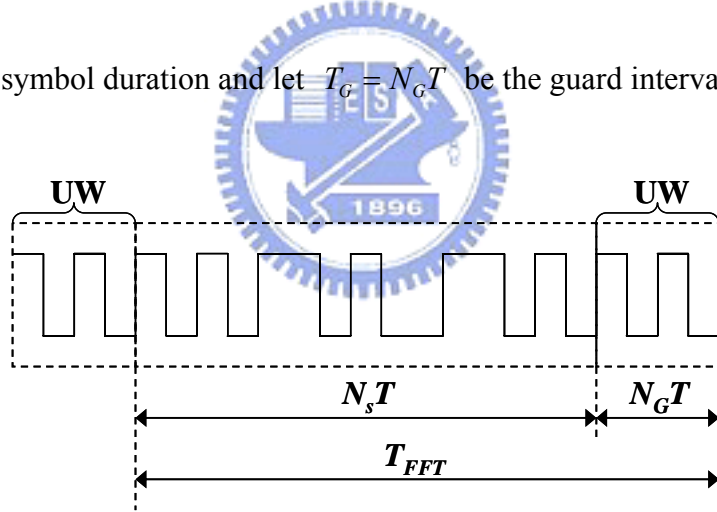


Figure 2.4: Transmitted block using the concept of UW

A mathematical description of the investigated SC-FDE system using the UW instead of the traditional CP is now given. Let us denote $s_{Data,i}(t)$ to be the continuous-time representation of the data symbols part of the i -th transmitted frame with $s_{Data,i}(t) = 0$ for $t \notin [0, T_{FFT} - T_G]$, and $uw(t)$ is the UW symbol sequence. Define $s_i(t)$ as the whole i -th frame. Therefore,

$$s_i(t) = \begin{cases} s_{Data,i}(t) & \text{for } t \in [0, T_{FFT} - T_G] \\ uw(t) & \text{for } t \in [T_{FFT} - T_G, T_{FFT}] \end{cases} \quad (2.2)$$

Including the UW from the previous frame, an extended frame $\hat{s}(t)$ can be defined as

$$\hat{s}(t) = \begin{cases} s_i(t) & \text{for } t \in [0, T_{FFT}] \\ uw(t + T_{FFT}) & \text{for } t \in [-T_G, 0] \\ 0 & \text{elsewhere} \end{cases} \quad (2.3)$$

With this cyclically extended frame the linear convolution ($*$) of the i -th frame with the channel impulse response becomes a circular convolution (\otimes) and the received block $\hat{r}_i(t)$ fulfils the condition

$$\hat{r}_i(t) = \hat{s}_i(t) * h(t) = s_i(t) \otimes h \quad (2.4)$$

within the interval $[-T_G + T_h, T_{FFT}]$, where T_h is the duration of the channel impulse response. When restricting the received frame to FFT window $[0, T_{FFT}]$ and applying the theorem of circular convolution to Eq.(2.4), we obtain the essential relation

$$\hat{R}_i(nf_0) = S_i(nf_0) \cdot H(nf_0) = R_i(nf_0) \quad (2.5)$$

For $f_0 = 1/T_{FFT}$ and $n \in Z$. The capitalization represents the Fourier Transform of the corresponding lowercase parameters in Eq.(2.4). Therefore, the frequency domain relation in Eq.(2.5) shows that the concept of UW is comparable to the concept of CP.

In this thesis, we focus on the design and performance of Unique Word based phase tracking algorithms [14], and we will show that the algorithm provide almost optimum performance in SC-FDE systems in Section 4.2 .

2.2.4 Preamble Channel and Frame Structure

Referring IEEE 802.11a standard, we attach the training sequence, also called preamble, in front of every packet. At the receiver, preambles can be utilized to do a number of tasks, such as timing synchronization, frequency synchronization, and channel estimation. The format of preamble channel and frame structure is shown in Figure 2.5. Preambles can further be separated into short preamble and long preamble, both of which are modulated by BPSK. Short preamble, as implied by the name, has a

shorter length compared with long preamble. Each short preamble contains 16 symbols with time-span $0.8 \mu\text{s}$, and ten short preambles have a total time-span of $8 \mu\text{s}$. The following parts are two long preamble symbols, and each one is protected by a guard interval filled with its cyclic extension, which have a total time-span of $8 \mu\text{s}$. After preamble channel, data symbols with cyclically UW extension follow.

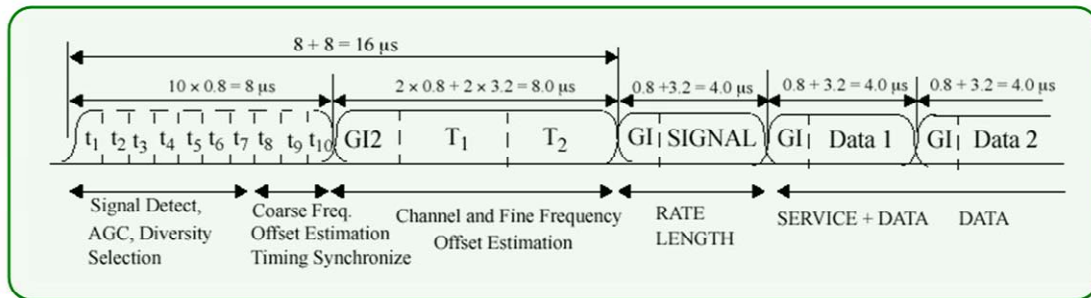


Figure 2.5: Training sequence and frame structure of IEEE 802.11a standard

2.2.5 Upsampler and Root Raised Cosine Filter

Upsampling is an operation that is often done before the pulse shaping filter to make the filter design simpler. If we do not increase the sample rate, we will need to design a very sharp filter which is not only very difficult and expensive to implement, but may sacrifice some of the spectrum in its roll off. Besides, a filter with a smooth roll off will have nicer phase characteristics as well.

Root raised cosines (RRC) filter is a commonly used pulse shaping filter in digital communication systems to limit ISI. The frequency response of an ideal root raised cosine filter consists of unity gain at low frequencies, the square root of raised cosine function in the middle, and total attenuation at high frequencies. The width of the middle frequencies is defined by the roll off factor constant β ($0 < \beta < 1$). Root raised cosine filter is generally used in series pairs, so that the total filtering effect is that of a raised cosine filter. The advantage is that if the transmit side filter is stimulated by an impulse, then the receive side filter is forced to filter an input pulse shape that is identical to its own impulse response, thereby setting up a matched filter and maximizing signal to noise ratio (SNR) while at the same time minimizing ISI. Mathematically, the frequency response $F_{rrc}(\omega)$ may be written as

$$F_{rrc}(\omega) = \begin{cases} 1 & \text{For } \omega \leq \omega_c(1 - \beta) \\ 0 & \text{For } \omega \geq \omega_c(1 + \beta) \\ \sqrt{\frac{1 + \cos\left(\frac{\pi(\omega - \omega_c(1 - \beta))}{2\beta\omega_c}\right)}{2}} & \text{For } \omega_c(1 - \beta) < \omega < \omega_c(1 + \beta) \end{cases} \quad (2.6)$$

where ω_c is half the data rate.

The operation of upsampling and RRC filtering is shown in Figure 2.6, where $H(z)$ is the frequency response of the pulse shaping filter and L is the upsampling rate.

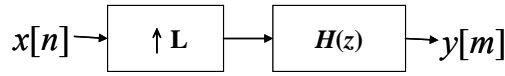


Figure 2.6: Diagram of upsampler and pulse shaping filter

Note that this procedure is computationally inefficient because the filter operates on a sequence that is mostly composed of zeros. To avoid operations on zero-valued samples, rearrangement of the preceding block diagram is required. First of all, transform $H(z)$ into its upsampled polyphase components, where $h[n]$ is the time domain counterpart of $H(z)$:

$$\begin{aligned} H(z) &= \sum_n (h[n]z^{-n}) \\ &= \sum_k \left(\sum_{p=0}^{L-1} (h[kL + p]z^{-(kL+p)}) \right) \quad \text{let } h_p[k] = h[kL + p] \\ &= \sum_{p=0}^{L-1} \left(\left(\sum_k (h_p[k]z^{-(kL)}) \right) z^{-p} \right) \\ &= \sum_{p=0}^{L-1} (H_p(z^L)z^{-p}) \end{aligned} \quad (2.7)$$

Therefore, Figure 2.6 can be redrawn as Figure 2.7(a). By the Noble identity for interpolation which is shown in Figure 2.8, Figure 2.7(a) can further evolve into Figure 2.7(b) and (c). Filters $h_p[n]$ in Figure 2.7(c) is called polyphase filters. Let the symbol rate before upsampling be $1/T$ and the length of $h[n]$ be N . Without polyphase simplification the computational cost is LN/T (computations/sec), while that with polyphase structure is N/T (computations/sec). Thus we save a factor of L .

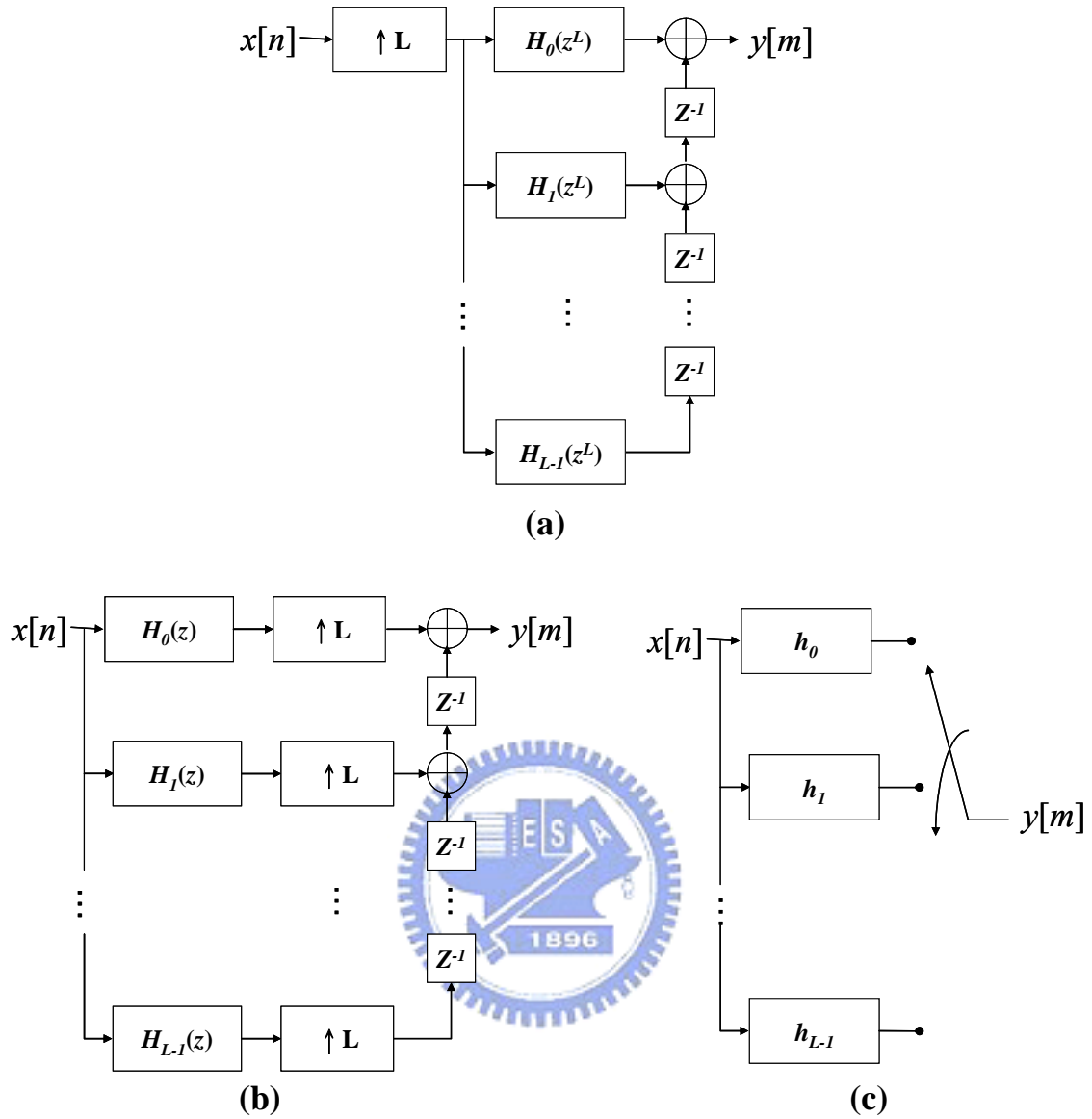


Figure 2.7: Evolution of the polyphase filter

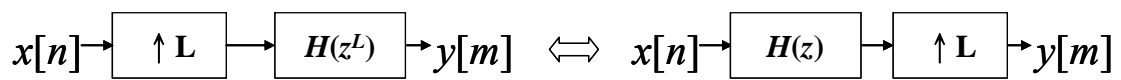


Figure 2.8: Noble identity

2.3 Receiver Architecture

The baseband function diagram of the proposed SC-FDE receiver is shown in Figure 2.9. The received signal is first down-converted to the baseband. After passing through the timing synchronization processing blocks, short preambles are separated as a means for frequency synchronization. The payload part is then processed blockwise. Each frame is transferred to frequency domain by the FFT block and equalized in the frequency domain. To acquire the channel information, long preamble is used to do frequency domain channel estimation. After being transformed back to the time domain, the phase shift of the equalized data is corrected by the known UW structure inserted in data channel to further improve the performance. The detected symbol streams are then recovered by a Viterbi decoder.

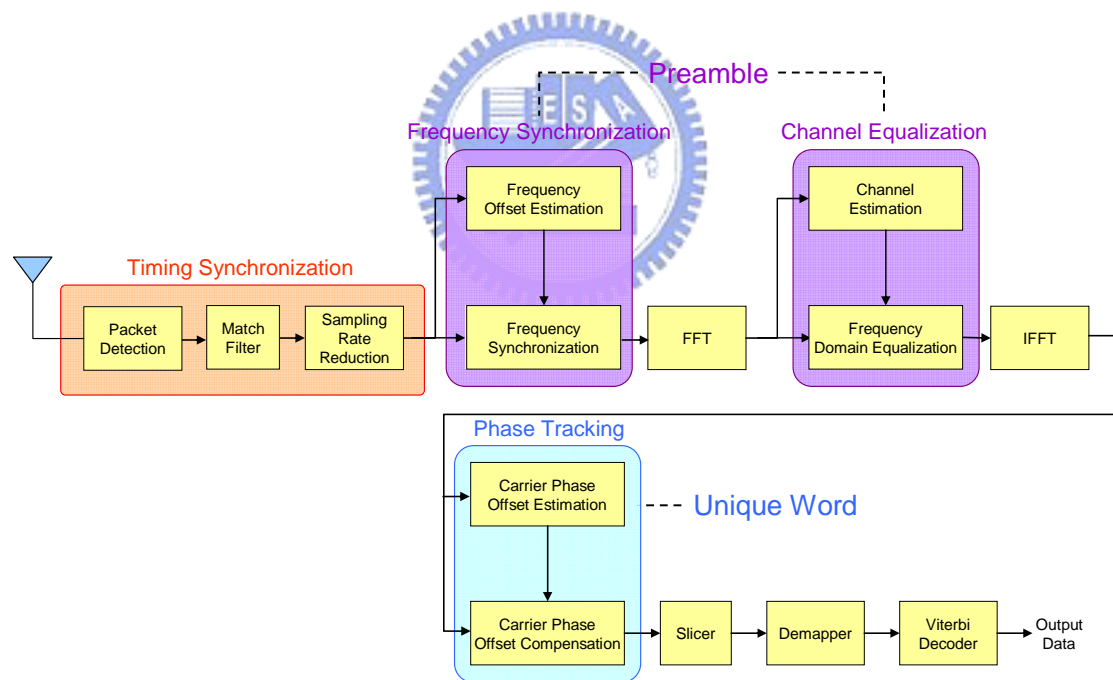


Figure 2.9: Receiver architecture of SC-FDE system

2.3.1 Timing Synchronization

2.3.1.1 Packet Detection

Packet detection is the task of finding an approximate estimate of the start of the preamble of an incoming packet. It is the first data-processing block of IEEE 802.11a baseband receiver. As such it is the first synchronization algorithm that is performed, so the rest of the synchronization process is dependent on good packet detection performance. On the other hand, power consumption can also be taken into consideration since the packet detection mechanism determines when the block behind should start to function. The double sliding window packet detection method is used in the thesis. It computes the signal energy over two sliding windows, **A** and **B**, as shown in Figure 2.10. When the packet starts to enter window **A**, the energy in **A** gets higher and higher. The basic principle is to form the decision variable m_n a ratio of total energy contained inside the two windows. The packet detection is declared when the value of m_n crosses over the threshold value.

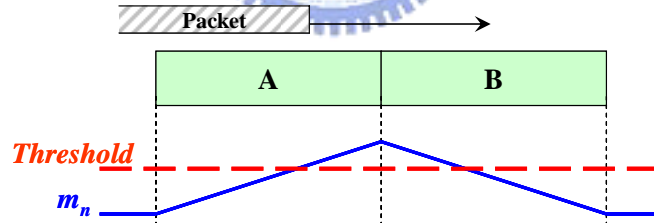


Figure 2.10: Double sliding window packet detection

The algorithm is described as

$$a_n = \sum_{m=0}^{M-1} r_{n-m} r_{n-m}^* = \sum_{m=0}^{M-1} |r_{n-m}|^2 \quad \text{where } M \text{ is length of window A}$$

$$b_n = \sum_{l=1}^L r_{n+l} r_{n+l}^* = \sum_{l=1}^L |r_{n+l}|^2 \quad \text{where } L \text{ is length of window B}$$

$$m_n = \frac{a_n}{b_n}$$

In addition to packet detection, finer timing synchronization in a SC-FDE system is required to decide where to place the start of the FFT window within one frame. Although an SC-FDE system exhibits a guard interval, making it somewhat robust against timing offsets, non-optimal symbol timing will cause more inter-symbol interference and inter-carrier interference (ICI) in delay spread environments. This will result in performance degradation. To eliminate timing offset induced by different path delays, fine timing synchronization will be performed after coarse timing synchronization.

2.3.1.2 Match Filter and Symbol Timing Recovery

After packet detection the received signal is fed through a matched filter and re-sampled at the symbol rate. The matched filter is simply an FIR filter with an impulse response matched to the transmitted pulse. It aids in timing recovery and helps suppress the effects of noise. The goal of symbol-timing recovery is to sample message signals at the receiver for best performance. Since upsampling is done at the transmitter, oversampling is performed at the analog-to-digital converter (ADC) as well to make the design of the filters simpler. Therefore, the received symbol rate should be reduced before the signal is sent to the digital processing blocks afterwards. Although the symbol rate is typically known to the receiver, the receiver does not know when to sample the signal for the best noise performance. One simple method for recovering symbol timing is performed using a delay-locked loop (DLL). Figure 2.11 is a block diagram of the necessary components. [22]-[24]

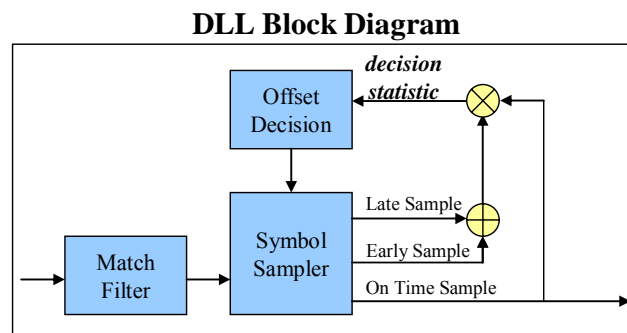


Figure 2.11: Block diagram of delay-locked loop

The goal of the DLL is to sample the waveform at the peaks in order to obtain the best performance in the presence of noise. If it is not sampling at the peaks, we say it is sampling too early or too late. The DLL finds peaks without assistance from the user. When it begins running, it arbitrarily selects a sample, called the on-time sample, from the matched filter output. The sample from the time-index one greater than that of the on-time sample is the late sample, and the sample from the time-index one less than that of the on-time sample is the early sample. The early, on-time and late samples together form a *detection set*. Figure 2.12 shows examples of detection sets where the on-time sample comes at a peak, before, and after the peak.

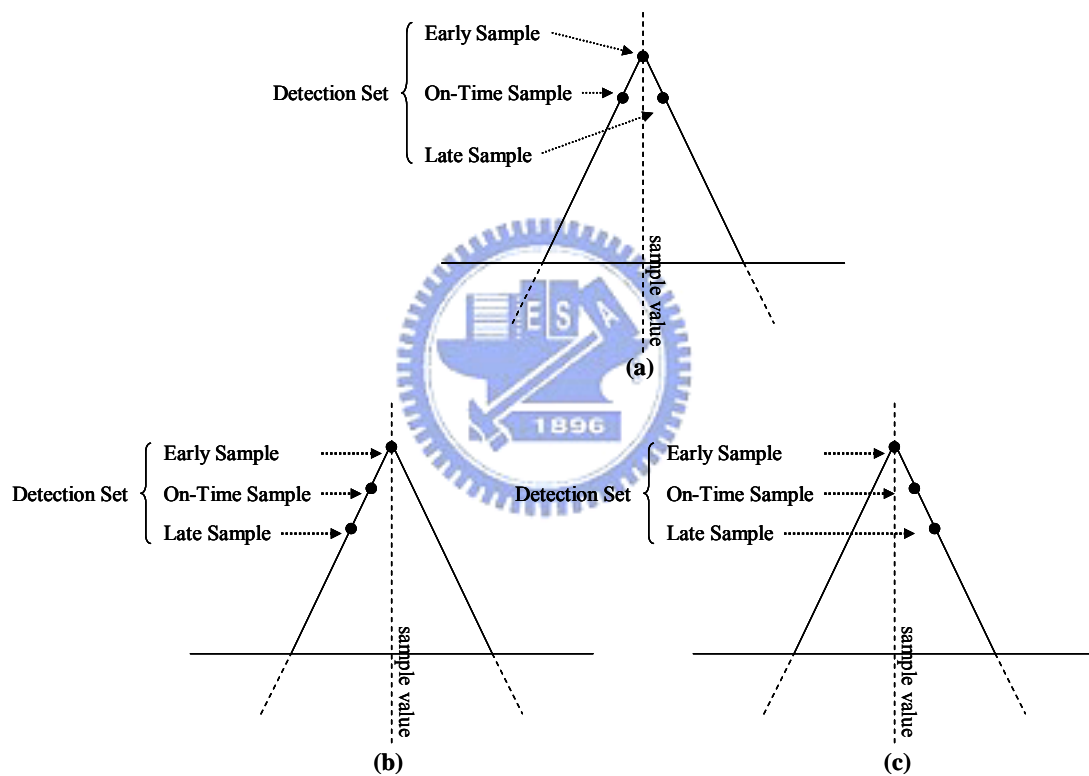


Figure 2.12: Detection set of delay-locked-loop

The on-time sample is the output of the DLL and will be used to decide the data bit sent. To achieve the best performance in the presence of noise, the DLL must adjust the timing of on-time samples to coincide with peaks in the waveform. It does this by changing the number of time-indices between on-time samples. Recall that the symbol duration is T and the upsample rate at the transmitter is 4. Accordingly, one transmitted symbol will occupies $4T$ in time after upsampling. Therefore, three cases are shown below:

1. In Figure 2.12(a), the on-time sample is already at the peak, and the receiver knows that peaks are spaced by $4T$. If it then takes the next on-time sample $4T$ samples after this on-time sample, it will be at another peak.
2. In Figure 2.12(b), the on-time sample is too early. Taking an on-time sample $4T$ samples later will be too early for the next peak. To move closer to the next peak, the next on-time sample is taken $4T + T = 5T$ samples after the current on-time sample.
3. In Figure 2.12(c), the on-time sample is too late. Taking an on-time sample $4T$ samples later will be too late for the next peak. To move closer to the next peak, the next on-time sample is taken $4T - T = 3T$ samples after the current on-time sample.

The input to the offset decision block in Figure 2.11 is called the *decision statistic*. When the decision statistic is positive, the on-time sample is too early; when it is zero, the on-time sample is at a peak, and when it is negative, the on-time sample is too late. The offset decision block could adjust the time at which the next on-time sample is taken based on the decision statistic. However, in the presence of noise, the decision statistic becomes a less reliable indicator. In this thesis, a modified DLL algorithm is proposed. Figure 2.13 shows a real case of the beginning of a received oversampled waveform, derived at the output of the match filter. From the packet structure discussed in Section 2.2.4, we know that this is the beginning part of short preambles. Since the amplitude of preamble symbol is known at the receiver, through simulation we know that the correct sample points of the oversampled waveform should be the ones circled as shown, with amplitude around 2 in our case.

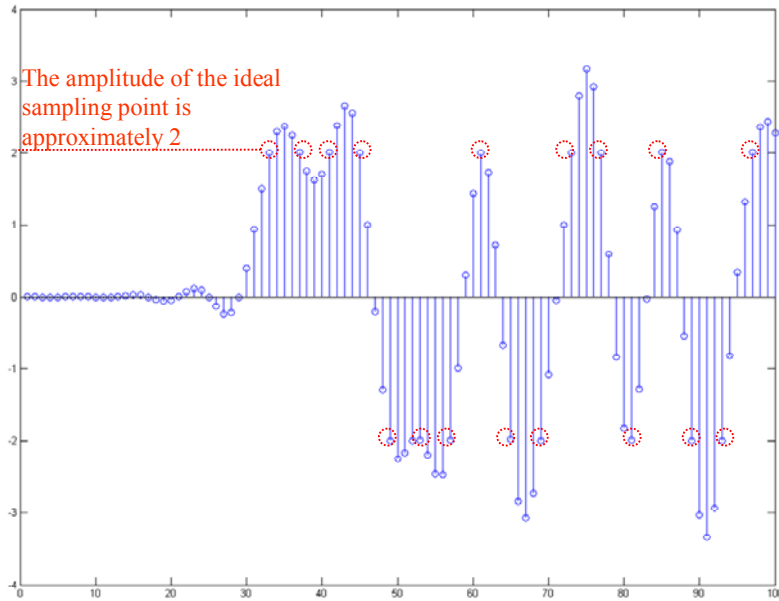


Figure 2.13: Oversampled waveform with the correct sample points

Therefore, if the detection set is chosen such that the on-time sample is “too far away” from the correct sample, the modified DLL algorithm will not let current on-time sample be the output, but shift the detection set by one symbol to make the new on-time sample be “closer to” the nearest correct sample and be the new output, as shown in Figure 2.14. The direction that the detection set is shifted is determined by the decision statistics as mention above. A full algorithm flow chart is shown in Figure 2.15.

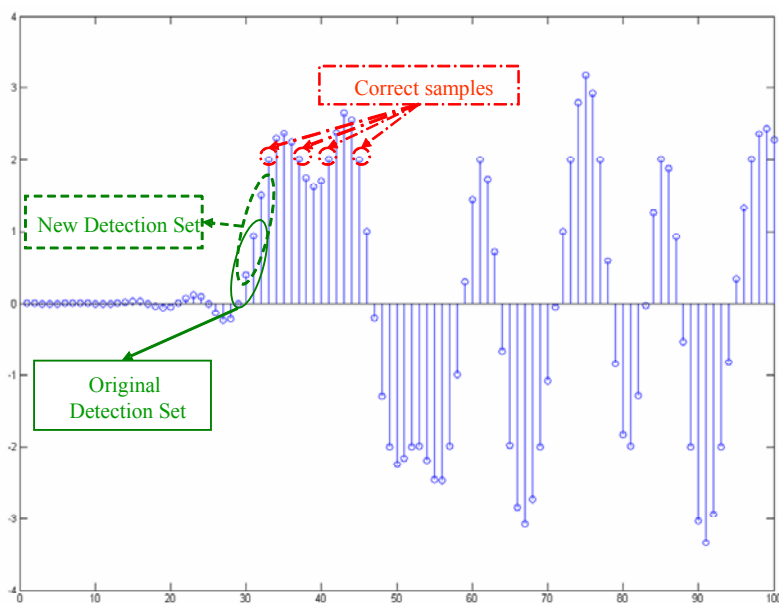


Figure 2.14: Original and new detection set in proposed DLL algorithm

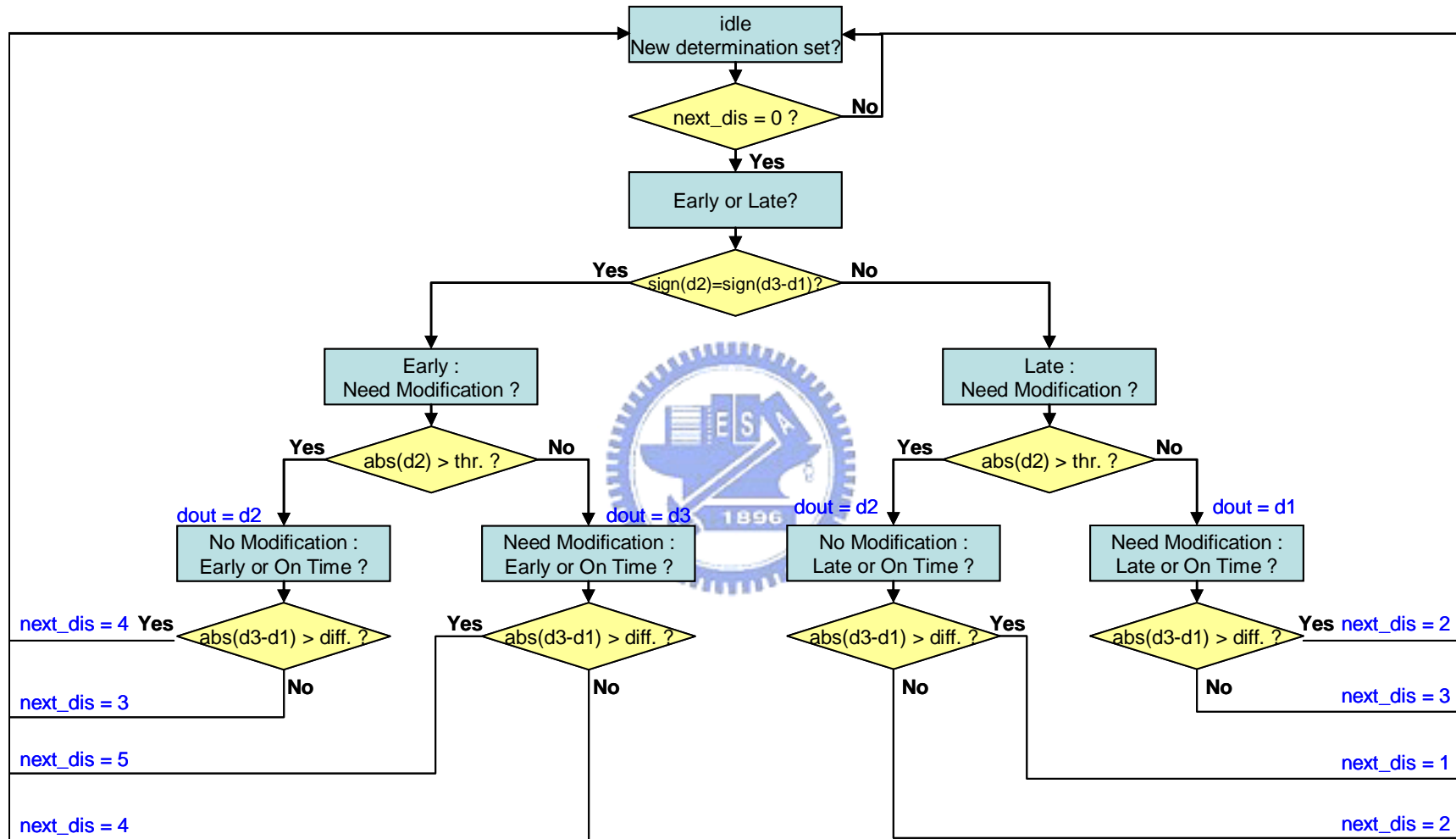


Figure 2.15: Proposed DLL algorithm Flow

2.3.2 Frequency Synchronizer

The purpose of frequency synchronization is to correct the frequency offset, which is caused by the difference of oscillator frequencies at the transmitter and the receiver and may degrade the system performance [25]. Therefore, we try to estimate the frequency offset and compensate the received signals.

Assuming that the absolute value of the frequency offset does not exceed $\frac{1}{2DT_d}$, where D is the delay between the identical samples of the two symbols; T_d denotes the sampling period. To derive the estimated frequency offset, an intermediate variable z is defined as:

$$z = \sum_{n=0}^{L-1} r_n r_{n+D}^* \quad (2.8)$$

where r_n is one of the two identical symbols, and L is the symbol length. After a series of derivation, the estimated frequency offset $\Delta\hat{f}$ can be shown by

$$\Delta\hat{f} = -\frac{\angle z}{2\pi DT_d} \quad (2.9)$$

where $\angle z$ can be computed by an arc tangent of the summation of conjugate multiplications between these two identical symbols. To do the above task, the preamble channel becomes the most proper candidate.

The 802.11a standard specifies a maximum oscillator error of 20 ppm; therefore the total maximum error is 40 ppm. Supposing that the carrier frequency is 5.3 GHz, the maximum possible frequency error is about 212 kHz. Owing to the inherent structures of short preamble and long preamble, the maximum unambiguous estimated frequency offset is 625 kHz for short preamble and 156.25 kHz for long preamble. Therefore, both short preamble and long preamble are required to estimate frequency offset so as to cover the probable frequency offset specified by the standard. In our thesis we use short preamble as a means to estimate the frequency offset.

2.3.3 Channel Estimator

Unlike conventional time domain equalization that uses one or more transversal filters taps with the number of data symbols spanned by the multipath, Frequency domain equalization has been shown to be an attractive solution for frequency selective channels in a single carrier system. Compared to the time-domain equalization, FDE outperforms the conventional time-domain equalization when the channel is highly dispersive, and it requires less complexity than maximum likelihood sequence estimation (MLSE) by using fast Fourier transform. The main task of the frequency domain equalizer is to eliminate inter-symbol interference within the individual frames. As long as the guard interval with duration T_G is longer than the channel impulse response (with duration T_h), there is no interference between the information symbols of successive frames.

The channel can be estimated using the known training symbols within the preamble. In our system, owing to the same symbol structure as data symbols, long preamble becomes the best candidate for performing this job. Let $R_{i,k}$ and X_k denote the frequency response of the i -th received long preamble and the original long preamble, the estimated channel frequency response is then derived by:

$$\begin{aligned}
 \widehat{H}_k &= \frac{1}{2|X_k|^2} (R_{1,k} + R_{2,k}) X_k^* \\
 &= \frac{1}{2|X_k|^2} (H_k X_k + N_{1,k} + H_k X_k + N_{2,k}) X_k^* \\
 &= H_k + \frac{1}{2|X_k|^2} (N_{1,k} + N_{2,k}) X_k^*
 \end{aligned} \tag{2.10}$$

Let D_k be the frequency response of received data symbol, the ZF channel equalization can be shown as:

$$\widehat{S}_k = \frac{D_k}{H_k} \tag{2.11}$$

where \widehat{S}_k denotes the frequency response of the recovered data symbol.

2.3.4 Phase Estimator

The processing of the preamble takes care of the initial synchronization of the SC-FDE receiver. However, there will always be some remaining phase offset after the initial preamble based carrier frequency synchronization, varying during the reception of the packet, making solely initial frequency synchronization insufficient. Furthermore, the system will experience phase noise produced by the combination of the RF oscillator and the phase-locked loop (PLL). In OFDM systems this residual offset causes inter-carrier interferences and a rotation of the constellation. In SC-FDE systems this residual offset also causes a rotation of the whole constellation from one received frame to the another, but instead of inter-carrier interferences, which cause a random spreading of the constellation points, the constellation points of the individual symbols experience a 1-D spreading along a circle (in the case of perfect timing synchronization). Figure 2.16 shows the constellation diagram of one equalized frame (48 data symbols, 16 UW symbols) in the QPSK mode in the presence of residual phase offset. It is, therefore, necessary to estimate and correct the rotation of the received constellation points.

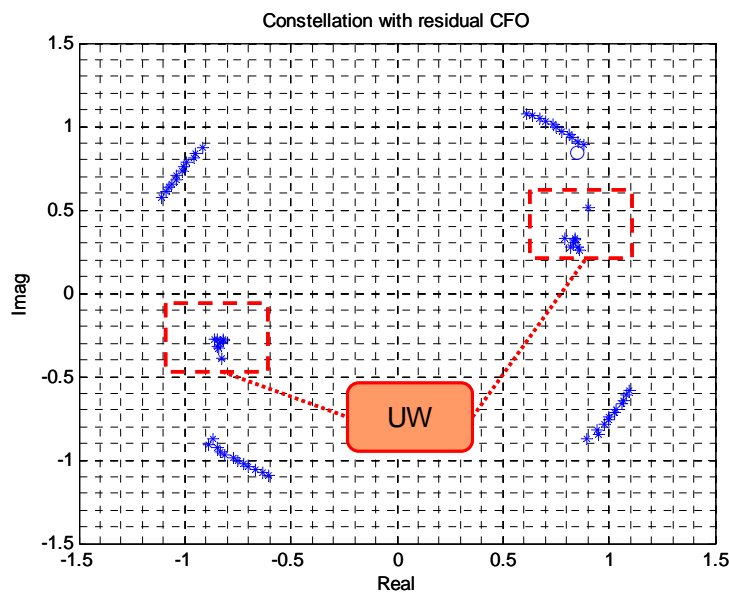


Figure 2.16: Constellation diagram of one equalized frame

The phase rotation from one frame to the next one caused by a residual CFO Δf is $2\pi\Delta fT_{FFT}$, and the phase rotation from one individual symbol to the other is $2\pi\Delta fT$. In the classical cyclic prefix approach (as used in OFDM, and in earlier SC-FDE proposals [10]), the guard interval is formed by data symbols, and the cyclic prefix is not processed any longer at the receiver. One major advantage of the UW approach is the fact, that the guard interval is part of the FFT-window and is therefore equalized together with the information symbols. Therefore the equalized UW-symbols can effectively be used for different synchronization tasks.

The mean rotation phase of the constellation diagram caused by a residual CFO Δf accumulates linearly from one frame to another. If the phase of the first frame is Θ_1 then the phase of the n -th block is given by $\Theta_n = \Theta_1 + (n-1)\Delta\Theta_n$ with $\Delta\Theta_n = 2\pi\Delta fT_{FFT}$. The straightforward approach to track the phase is to use an average of 16 Unique Word phase errors, and to de-rotate the constellation by the estimated phase. However, due to the fact that the UW-symbols are positioned at the end of a frame, they experience a larger phase rotation than the data symbols. Fortunately, according to the linearly-accumulated phase error properties mentioned above, it is possible to linearly de-spread the constellation points using an individual de-rotation once we derive the known average Unique Word phase errors [14].

The phase tracking algorithm is summarized in Table 2.2, and the phase compensation result will be given in the MATLAB floating point verification part in Chapter 4.

Table 2-2: Proposed phase tracking algorithm

<p><i>Set</i> $\Theta_0 = 0$ <i>for</i> $n = 1$ <i>to</i> N</p> <ul style="list-style-type: none"> - Equalize the n-th frame by $e^{-j\Theta_{n-1}}$ - Determine the estimate $\Delta\Theta_n$ of current frame by averaging over the 16 UW phase errors - De-rotate each received symbol $r(k)$ by $e^{-j\Delta\Theta_n 2kT/(2T_{FFT}-T_G)}$ - Determine the accumulated phase offset $\Theta_n = \Theta_{n-1} + \Delta\Theta_n$ <p><i>end</i></p>

2.3.5 Viterbi Decoder

Decoding of convolutional codes is most often performed by the Viterbi decoder, which is an efficient way to obtain the optimal maximum likelihood estimate of the encoded sequence [26][27]. Viterbi decoder can be further divided into hard-decision and soft-decision decoding, where hard-decision is adopted in our system. According to the design of the convolutional encoder in transmitter, we can derive the state transition table in Table 2.3 and then further illustrate the trellis diagram as shown in Figure 2.17 and Figure 2.18.

The Viterbi algorithm is a recursive sequential minimization algorithm that can be used to find the least expensive way to route symbols from one edge of a state diagram to another. To do this, the algorithm uses a cost analysis mechanism to calculate the distance between the received symbol and the symbol associated to that edge. The distance between the received symbol s and the symbol associated to that edge in the state diagram is often referred to as the branch metric. If $BM [i, j](s)$, is the metric of the branch from state i to state j , the problem is finding the path for which the metric, i.e. the sum of the branch metrics of the path edges, is at a minimum. The Viterbi algorithm solves this problem by applying the following recursive equation for each state transition

$$PM [j](t) = \min (PM [i](t-1) + BM [i, j](s)) \quad (2.12)$$

where $PM [j](t)$ is the path metric associated to the (minimum cost path leading to) state j at time t . At the end of the decoding, it is possible to reconstruct the maximum likelihood sequence through a trace back starting from the possible decoder states. Normally for decoders using non-punctured codes, the trace back depth equals five-times constraint length, which is sufficient to decode the correct output in the presence of noise. In our system, constraint length equals 5; therefore an appropriate trace back depth is 25.

Table 2-3: State transition table

din	s0	s1	s2	s3	state	g2	g1	g0	s0'	s1'	s2'	s3'	state'	din	s0	s1	s2	s3	state	g2	g1	g0	s0'	s1'	s2'	s3'	state'		
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	1	1	1	1	0	0	0	0	8
0	0	0	0	1	1	1	1	1	0	0	0	0	0	0	1	0	0	0	1	1	0	0	0	1	1	0	0	0	8
0	0	0	1	0	2	1	1	0	0	0	0	1	1	1	0	0	1	0	2	0	0	1	1	0	0	1	0	0	9
0	0	0	1	1	3	0	0	1	0	0	0	1	1	1	0	0	1	1	3	1	1	0	1	0	0	1	0	0	9
0	0	1	0	0	4	1	0	1	0	0	1	0	2	1	0	1	0	0	4	0	1	0	1	0	1	0	0	10	
0	0	1	0	1	5	0	1	0	0	0	1	0	2	1	0	1	0	1	5	1	0	1	1	0	1	0	1	10	
0	0	1	1	0	6	0	1	1	0	0	1	1	3	1	0	1	1	0	6	1	0	0	1	0	1	1	1	11	
0	0	1	1	1	7	1	0	0	0	0	1	1	3	1	0	1	1	1	7	0	1	1	1	0	1	1	1	11	
0	1	0	0	0	8	1	1	0	0	1	0	0	4	1	1	0	0	0	8	0	0	1	1	1	0	0	0	12	
0	1	0	0	1	9	0	0	1	0	1	0	0	4	1	1	0	0	1	9	1	1	0	1	1	0	0	0	12	
0	1	0	1	0	10	0	0	0	0	1	0	1	5	1	1	0	1	0	10	1	1	1	1	1	0	0	1	13	
0	1	0	1	1	11	1	1	1	0	1	0	1	5	1	1	0	1	1	11	0	0	0	1	1	0	1	0	13	
0	1	1	0	0	12	0	1	1	0	1	1	0	6	1	1	1	0	0	12	1	0	0	1	1	1	0	0	14	
0	1	1	0	1	13	1	0	0	0	1	1	1	6	1	1	1	0	1	13	0	1	1	1	1	1	0	0	14	
0	1	1	1	0	14	1	0	1	0	1	1	1	7	1	1	1	1	0	14	0	1	0	1	1	1	1	0	15	
0	1	1	1	1	15	0	1	0	0	1	1	1	7	1	1	1	1	1	15	1	0	1	1	1	1	1	0	15	

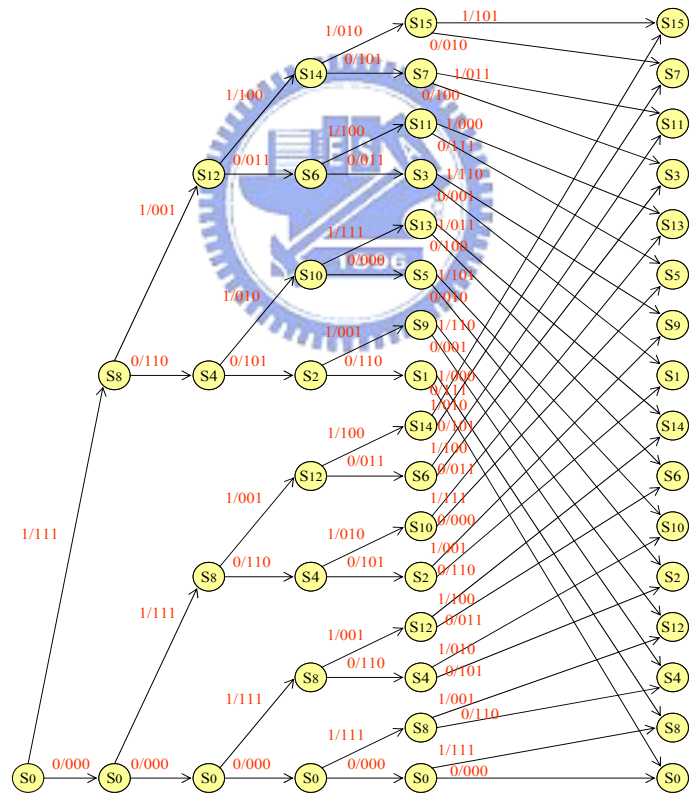


Figure 2.17: Trellis diagram part 1

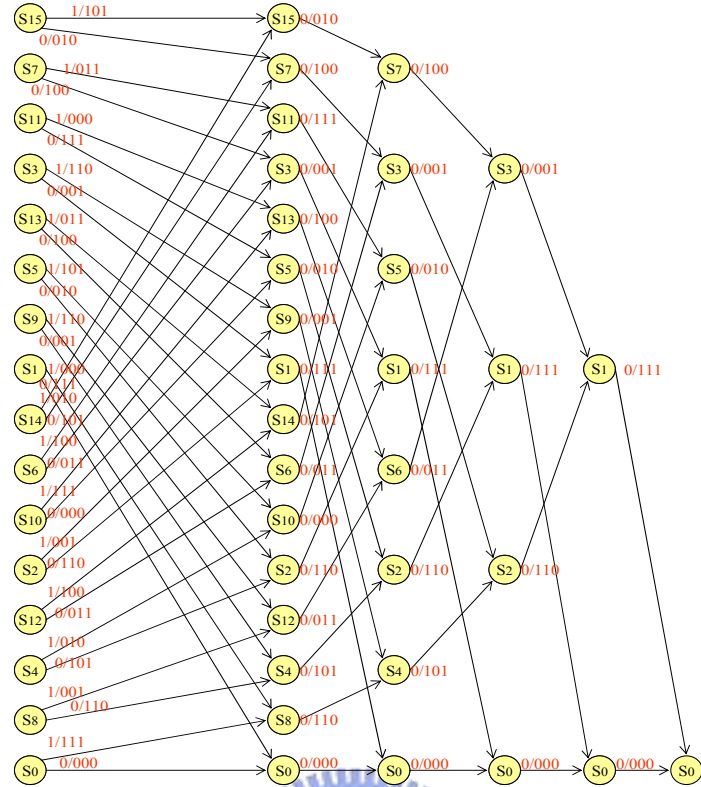


Figure 2.18: Trellis diagram part 2

2.4 Summary

In this chapter, we introduce the SC-FDE system, and propose our system architecture including the transmitter and receiver. At the transmitter, convolutional encoder, mapper, UW generator, preamble channel generator, the polyphase filter design of upsampler and root-raised cosine filter are gone through. At the receiver, timing synchronization is first mentioned, which consists of packet detection, match filtering and symbol timing recovery. Frequency synchronization and channel estimation then follow, and phase estimation is proposed. Finally, de-mapper, and Viterbi decoder are described in the rest part of the receiver. In this chapter we highlight the two algorithms proposed: the modified delay-locked-loop algorithm for timing recovery and the UW-based phase tracking algorithm implemented on the system as an independent section to give a detailed introduction. More experimental results and performance analysis will be given in Chapter 4.

Chapter 3

SC-FDE System Platforms

In Chapter 3, we will introduce our self-designed platform as the development environment. The platform is used to perform the verification of whole SC-FDE system including baseband and RF parts, where transmitter and receiver are implemented on two separated boards with their own RF modules each. The self-designed platform is closer to a real wireless communication system and therefore can take all phenomena and effects of the wireless system into account. In the following sections, hardware modules, hardware description language, software design flows, and the corresponding debugging tools of our platforms are detailed explained.

3.1 Self-designed Platform

In order to approach a real wireless communication system, the multi-synchronous and high-speed bus FPGA design, combined with our module-based RF, AD/DA, and MAC/BB hardware system, becomes the best solution. Our laboratory has finished and successfully tested RF, AD/DA and MAC/BB boards. The development environment is shown in Figure 3.1, and the close-up shot of main board is shown in Figure 3.2, where four Xilinx Virtex II 6000 FPGAs are mounted in MAC/BB board, and each MAC/BB board is able to connect with at most two AD/DA and two RF modules.

In order to avoid the interference between high speed digital bus, those layouts and interconnections of different modules shall be handled very carefully. Our measurements show that directly connected modules did achieve feasible solution which reduces the risk of facing interconnection problems. Further analysis and evaluation during development are given in the following sections.

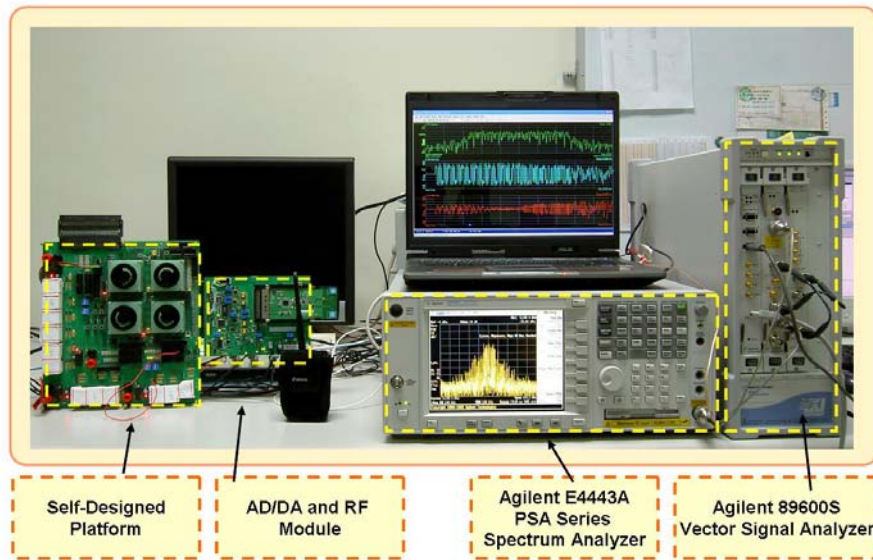


Figure 3.1: Development environment of self-designed platform

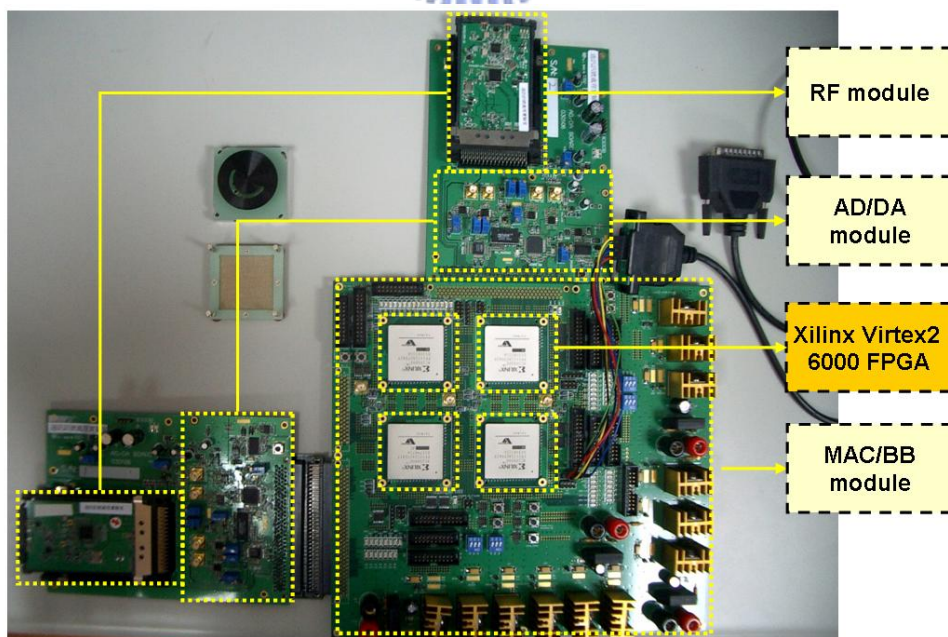


Figure 3.2: Main board of self-designed platform

3.1.1 RF Module

The RF module, as shown in Figure 3.3 consists of MAX2828, which is specifically designed for single-band IEEE 802.11a applications covering world-band frequencies of 4.9 GHz to 5.875 GHz. MAX2828 includes all circuitry required to implement the RF transceiver function, providing a fully integrated receive path, transmit path, voltage-controlled oscillator (VCO), frequency synthesizer, and baseband control interface. Only the RF switches, RF bandpass filters (BPF), RF baluns, and a small number of passive components are required to form the complete RF front-end solution. Because the balance of I/Q signals will impact on the waveform of RF output, the RLC components had been fine tuned. Besides, we also tested the frequency accuracy and power level of transmitted carriers in our interested band from 5.15 GHz to 5.875 GHz. One of those measurements is shown in Figure 3.4; the power level shall be further improved with fine tuning of matching circuits. We used 3-wires (Clock, Data and Latch) to control the RF module from PC currently, and then the control mechanism will be integrated into MAC/BB after verification. In sum, MAX2828 completely eliminates the need for external SAW filters by implementing on-chip monolithic filters for both the receiver and transmitter. The baseband filtering and the Rx/Tx signal paths are optimized to meet the IEEE 802.11a and 802.11g standards. It is also suitable for the full range of the corresponding 802.11a/g OFDM data rates (6Mbps, 9Mbps, 12Mbps, 18Mbps, 24Mbps, 36Mbps, 48Mbps, and 54Mbps) and 802.11g QPSK data rates (1Mbps, 2Mbps, 5.5Mbps, and 11Mbps), at the required sensitivity levels.



Figure 3.3: RF module on self-designed platform

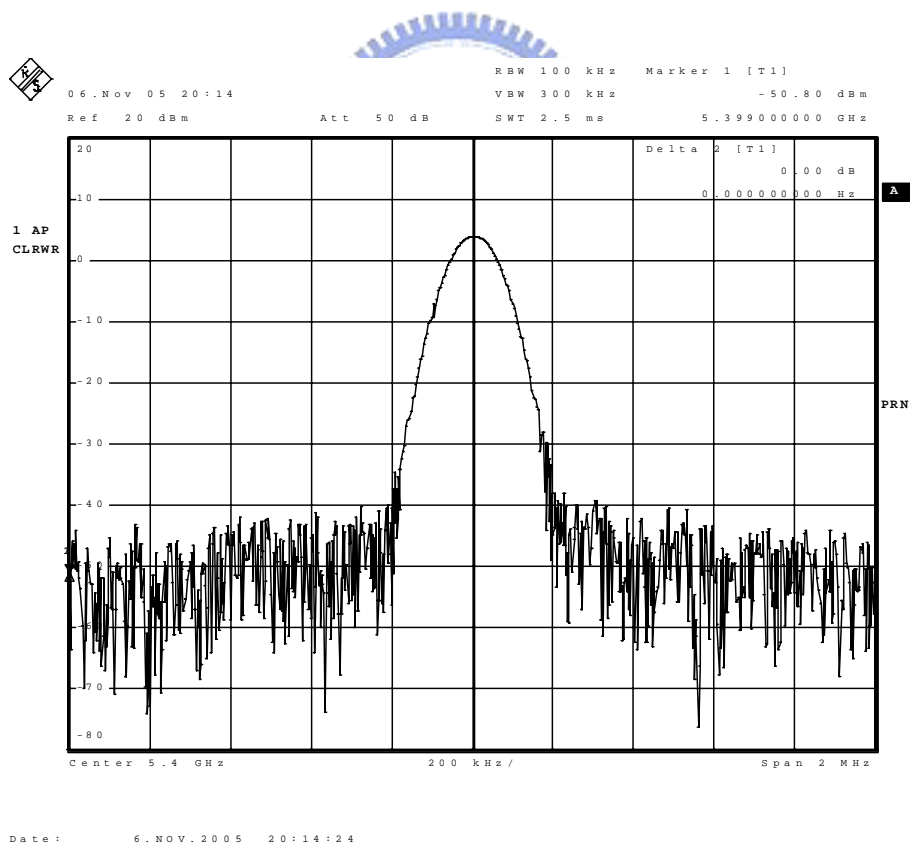


Figure 3.4: Measured carrier spectrum form RF module

3.1.2 AD and DA Modules

The AD/DA module, as shown in Figure 3.5, consists of ADS2807 and DAC2900. The ADS2807 is a dual, high-speed, high dynamic range, 12-bit pipelined Analog-to-Digital Converter (ADC). This converter includes a high-bandwidth track-and-hold that gives excellent spurious performance up to and beyond the Nyquist rate. The measured timing diagram is shown in Figure 3.6, which indicates the valid data during the high clock period. In addition, it is recommended that data hold time is 3.5 ns for saving data from bus to Static random access memory (SRAM), which had been verified on our AD/DA boards too. The differential nature of this track-and-hold and ADC circuitry minimizes even-order harmonics and gives excellent common-mode noise immunity. The track-and-hold can also be operated single-ended. Besides, it provides for setting the full-scale range of the converter without any external reference circuitry. The internal reference can be disabled allowing low-drive, external references to be used for improved tracking in multichannel systems.

The DAC2900 is a monolithic, 10-bit, dual-channel, high-speed Digital-to-Analog Converter (DAC), and is optimized to provide high dynamic performance while dissipating only 310mW on a +5V single supply. Operating with high update rates of up to 125MSPS, the DAC2900 offers exceptional dynamic performance, and enables the generation of very-high output frequencies suitable for "Direct IF" applications. The DAC2900 has been optimized for communications applications in which separate I and Q data are processed while maintaining tight gain and offset matching. Each DAC has a high impedance differential current output, suitable for single-ended or differential analog output configurations. In addition, the DAC2900 combines high dynamic performance with a high throughput rate to create a cost effective solution for a wide variety of waveform synthesis applications.



Figure 3.5: AD/DA module on self-designed platform

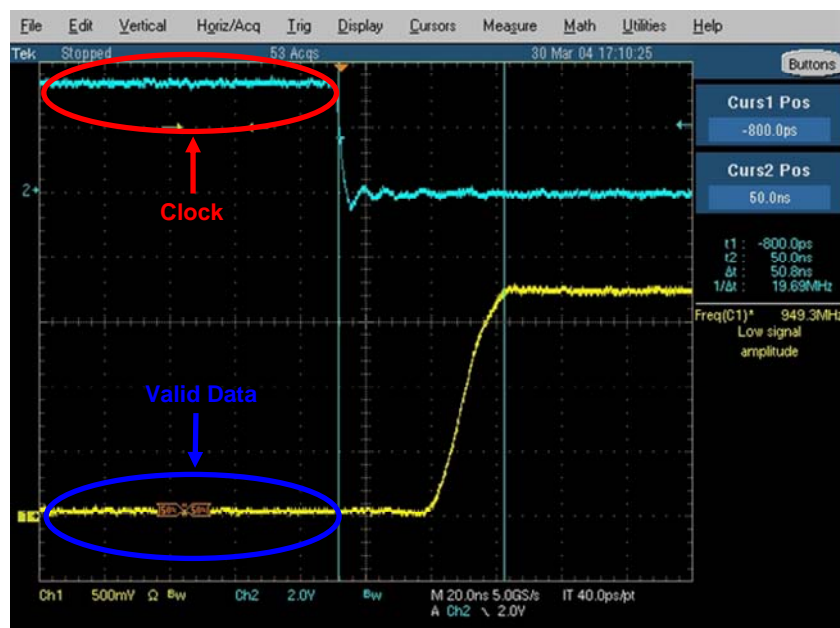


Figure 3.6: Measured data waveform from AD/DA module

3.1.3 MAC/BB Platform

The MAC/BB is an FPGA-based module which is composed of four Xilinx Virtex-II 6000 modules, as shown in Figure 3.7. Recently, the demand for more complex programmable hardware is constantly growing to meet the formidable

industry requirement. The major categories of programmable hardware are programmable logic device (PLD) and FPGA. A PLD consists of micro-cells and a central inter-connection logic. Typical PLD applications are “glue logic” for connecting other ASICs. On the other hand, FPGAs consist of even more complex logic block on one chip. Typical applications are central control units (CPU) and DSPs up to very complex SoC design. Therefore, we adopt some FPGA modules to realize our communication system. Generally, FPGA can be categorized into three types by its structure:

1. **Look-up-table (LUT):** Xilinx, Altera, AT&T
2. **Multiplexer:** Actel, Quicklogic
3. **Transistor array:** Cross point

If we focus on its programming architecture, there are two major types:

1. **SRAM:** Xilinx, Altera, AT&T, Atmel
2. **Anti-fuse:** Actel, Cypress, Quicklogic

SRAM type has a merit of being able to program repeatedly while Anti-fuse type has the feature of one time programmable (OTP). Anti-fuse type can offer security for design but cannot be modified further.

Compared to ASIC, FPGA has lower performance apparently, especially on power consumption and maximum supportable speed. However, as the technique of semiconductor industry grows, FPGA becomes more and more competitive to ASIC. Actually, FPGA has more integration ability and flexibility than ASIC, and undoubtedly, is the best candidate component for a fast-prototyping system. On the other hand, more and more DSP systems are implemented using FPGA rather than DSP processors, since when sample rates grow above a few Mhz, a DSP has to work very hard to transfer the data without any loss. An FPGA on the other hand dedicates logic for receiving the data, so can maintain high rates of I/O, Therefore, a high speed environment especially combined with rigid, repetitive tasks suits the FPGA. It outperforms conventional DSP processors on a board-for-board comparison, resulting in significant improvements in processing speed, size, weight, power, and costs.

On our MAC/BB platform, an FPGA-based module which is composed of four Xilinx Virtex-II 6000 modules, where each of them combines a wide variety of flexible features and a large range of densities up to 6 million system gates, enhancing programmable logic design capabilities and is a powerful alternative to mask-programmed gates arrays. With its advantages of very fast data rate, it can achieve full duplex and real time operating for wireless communication. The VHDL codes had been used to drive LEDs by differential clock rate from oscillator to verify its functionality.

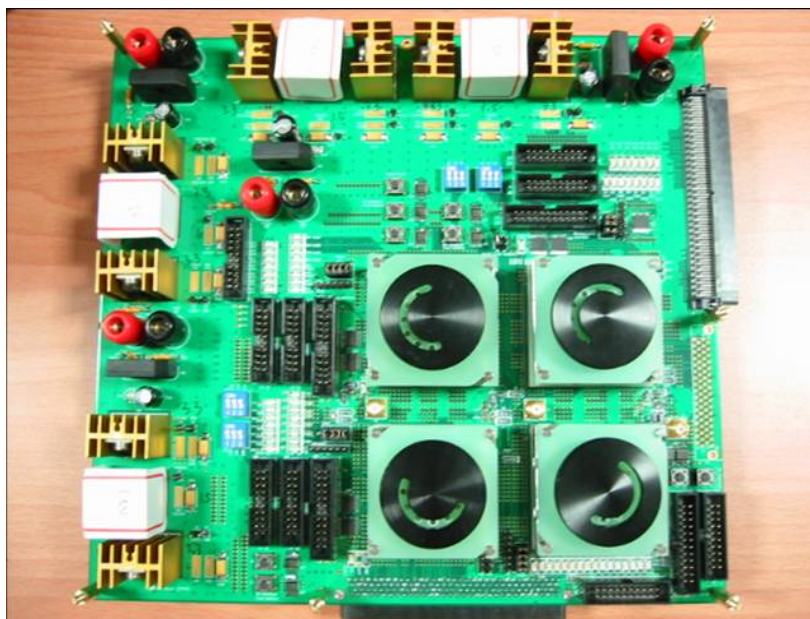


Figure 3.7: MAC/BB platform

3.1.4 USB Interface

In order to have a convenient input for the audio/video signal in the future, USB interface was designed into the platform, which is shown in Figure 3.8. It will comply with the USB specification revision 1.1, and be upgraded to USB 2.0 if necessary. The compatibility test is conducted with compliance software run at PC equipped with PCI to UTMI compliant interface card during test stage. This will make sure we can easily connect our platform with any signal source with USB port. The built-in USB interface codes for FPGA was defined and implemented.

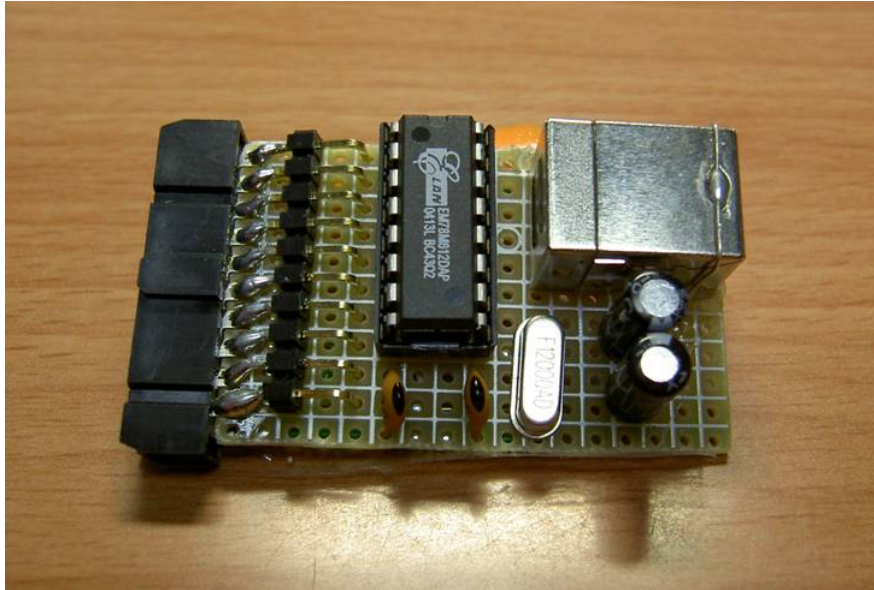


Figure 3.8: USB module on self-designed platform

3.2 Benefits of using VHDL

In our thesis we use VHDL (Very High Speed Integrated Circuit Hardware Description Language) as our hardware description language, since it has the following advantages:

(1) Executable specification

It is often reported that a large number of DSP designs meet their specifications first time, but fail to work when plugged into a system. VHDL allows this issue to be addressed in two ways: A VHDL specification can be executed in order to achieve a high level of confidence in its correctness before commencing design, and may simulate one to two orders of magnitude faster than a gate level description.

A VHDL specification for a part can form the basis for a simulation model to verify the operation of the part in the wider system context (e.g. printed circuit board simulation). This depends on how accurately the specification handles aspects such as timing and initialization. Behavioral simulation can reduce design time by allowing design problems to be detected early on, avoiding the need to rework designs at gate level. Besides,

it also permits design optimization by exploring alternative architectures, resulting in better designs.

(2) **Portability between tools**

VHDL descriptions of hardware design and test benches are portable between design tools, and portable between design centers and project partners. One can safely invest in VHDL modeling effort and training, knowing that he will not be tied in to a single tool vendor, but will be free to preserve the investment across tools and platforms. Also, the design automation tool vendors are themselves making a large investment in VHDL, ensuring a continuing supply of state-of-the-art VHDL tools.

(3) **Technology independent design**

VHDL permits technology independent design through support for top down design and logic synthesis. To move a design to a new technology one need not start from scratch or reverse-engineer a specification - instead one can go back up the design tree to a behavioral VHDL description, then implement that in the new technology knowing that the correct functionality will be preserved.

3.3 FPGA Design Flow

In our design, we choose Xilinx ISE 6.3 and Synplify Pro 8.2 as the development tool for the first half of the design flow. Figure 3.9 is the main system design flow with FPGA and later we will give more information about the flow.

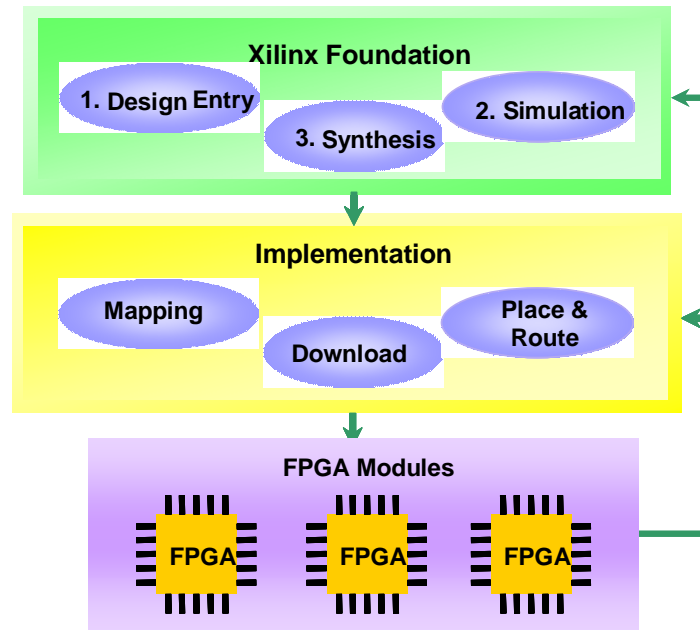


Figure 3.9: FPGA design flow

(1) Design Entry

In general, EDA tools are required to develop register transfer level (RTL) codes by appropriate methodologies. In Xilinx ISE 6.3, it supports three methods: HDL (hardware description language) Editor, Schematic Flow, and FSM (finite state machine) Editor. HDL Editor allows us to edit source files directly like VHDL [28]-[30] and Verilog [31]-[32], which are the most common HDLs in use today. Schematic Flow is another choice to create our source files by drawing the scheme with underlying HDL macros. FSM Editor allows us to edit by timing state diagram, which is suitable for realization controller, such as memory access controller.

(2) Synthesis

After completing editing RTL source files, we need to translate them into gate level called netlist files, which only contains information of logic gates and inter-connections. We choose to use Synplify Pro 8.2 for synthesis.

(3) Simulation

Design verification is an important aspect of each project design. Before implementing our circuit in the target device, it is a good idea to simulate and

verify the circuit. The most common verifications are functional simulation and timing simulation.

A. Functional Simulation

Functional simulation can be done after the schematic has been entered or a HDL file has been created and synthesized. Functional simulation gives information about the logic operation of the circuit, but it does not provide any information about timing delays.

B. Timing Simulation

The timing simulation will give us detailed information about the time it takes for a signal to pass from one gate to the other (gate delay) and gives information on the circuit's worst-case conditions. The total delay of a complete circuit will depend on the number of gates the signal sees and on the way the gates have been placed in the FPGA. One of the most popular simulation tools is ModelSim, which is completely integrated into Xilinx ISE 6.3, and can perform functional simulation and timing simulation very well. Thus, we choose ModelSim SE 6.1e as the simulation tool in our design flow.

(4) Implementation

The implementation is typically done after the design has been verified by functional simulation. The implementation tools will translate the netlist (schematic, HDL), place and route the design in the target device and generate a bitstream that can be downloaded into the device.

(5) Download to FPGA

After the process of implementation, we can download our design into hardware platform. To verify that signals are really working properly in circuit, we can use the logic analyzer (LA) to debug. Once the result does not match what we expect, we need to come back to modify our design and go through the whole design flow again. That is to say, iterative tests are required until we obtain the results we want.

3.4 Debugging Tools

As an old saying goes, “What is a workman without his tools.” In our self-designed platform, we do have some useful tools for debugging as follows.

1. Logic Analyzer:

We use Agilent 16702B LA to perform the major task of debugging. There are two modules installed on it. One is 16522A Pattern Generator Module, and the other is 16711A Measurement Module. The former is mainly used for generating desired signals, such as the reset signal or some selection signals for model selection; the latter is used for probing signals in FPGA on the self-designed platform.

2. Oscilloscope:

It is usually used when transmitted signals are prepared by FPGA and sent to the DA module by specific cables. Therefore, we can verify the waveform shown in the oscilloscope. For our system we may expect to see the waveform containing preambles in the form of square wave in the head part and data symbols appended with UW follow behind those preambles.

3. Spectrum Analyzer:

Agilent PSA Series Spectrum Analyzer E4443A is chosen. It offers high-performance spectrum analysis up to 6.7 GHz and beyond with swept-tuned measurements with digital Resolution-BandWidths (RBW) filters. In our debugging flow, E4443A capture the transmitted 5.2GHz signals, down convert them to 70MHz intermediate frequency (IF), and then fed out to vector signal analyzer to perform advanced analysis. Its block diagram is shown in Figure 3.10.

4. Vector Signal Analyzer:

Instead of swept-tuned measurements, vector signal analyzer 89600S performs FFT measurements with digital FFT filters, which can measure all

signal characteristics (i.e. phase) and avoid very long sweeps times required for narrow RBW. Figure 3.11 shows the block diagram of vector signal analyzer, notice that it is PC-based and therefore machines only capture the RF signal accurately and feeds to PC, where final analysis are performed on PC.

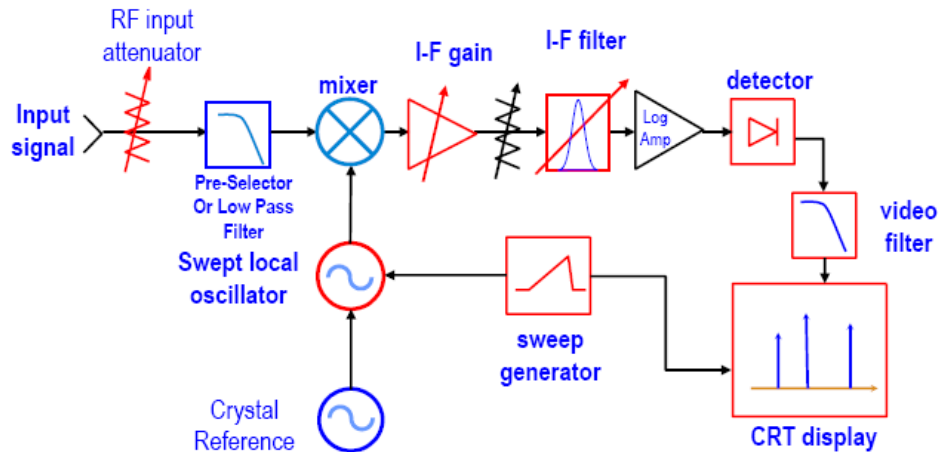


Figure 3.10: Spectrum analyzer block diagram

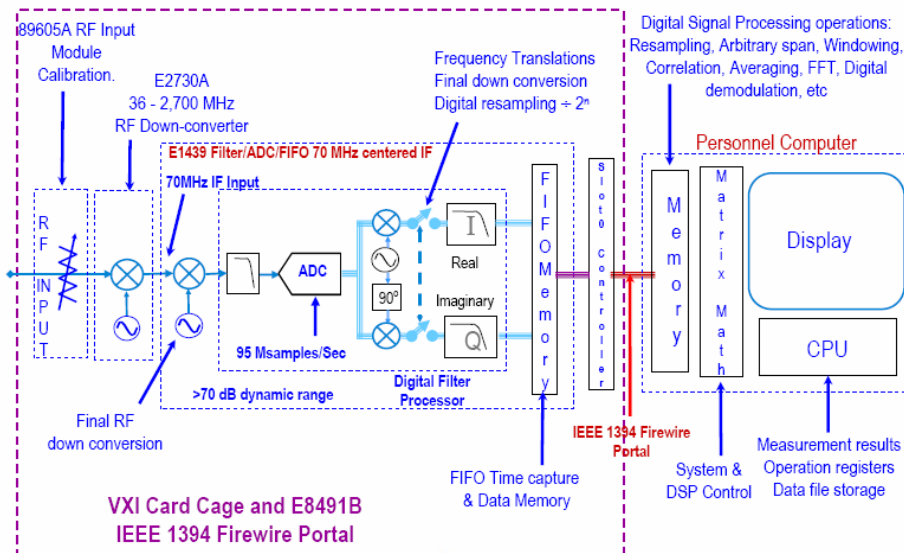


Figure 3.11: Vector signal analyzer block diagram

3.5 Summary

In this chapter, we introduce our self-designed platform used to perform the final verification of whole SC-FDE system. The platform equipped with FPGA, USB, and AD/DA modules as well as the RF modules by which realistic wireless channel characteristics can be generated. In addition, hardware description language and software design flows as well as corresponding debugging tools are mentioned; in particular the logic analyzer and oscilloscope are used to measure baseband signals, and spectrum analyzer and vector signal analyzer are used to capture and analyze RF signals.



Chapter 4

SC-FDE System Realization

The SC-FDE system is implemented on the FPGA-based hardware introduced in Chapter 3. This chapter is the major part of this thesis, which is organized as follows. In the first subsection, a complete design flow is proposed. Then the MATLAB verification is given, and algorithms proposed in Chapter 2 are demonstrated and the system performance of SC-FDE system is shown and compared with the OFDM system. In addition, the circuit design of the system on FPGA is detailed, and finally, the ModelSim simulation and experimental results will be presented, where the principles and concepts of circuit design on FPGA will specially be emphasized.

4.1 Design Flow

Digital Signal Processing (DSP) design has traditionally been divided into two types of activities — systems/algorithm development and hardware/software implementation. The majority of DSP system designers and algorithm developers use the MATLAB language for prototyping their DSP algorithm. Hardware designers take the specifications created by the DSP engineers and create a physical implementation of the DSP design by creating a register transfer level (RTL) model in a hardware description language (HDL) such as VHDL and Verilog. Our SC-FDE system can be regarded as a DSP system, and Figure 4.1 shows the design flow we adopt.

First, we have to program a floating-point MATLAB code in order to verify the algorithms mentioned in Chapter 2 as well as evaluate the system performance. Then, the floating-point MATLAB code is required to be manually converted into the fixed-point MATLAB code. Subsequently, RTL model is established, where we choose VHDL as our hardware description language and Xilinx ISE 6.3 as our development tool. Next, this RTL implementation is simulated by ModelSim SE 6.1e and synthesized onto a netlist of gates using Synplify Pro 8.2. Finally, the netlist of gates is placed and routed onto Xilinx FPGAs using Xilinx ISE 6.3. The detailed design flow will be discussed in the following sections

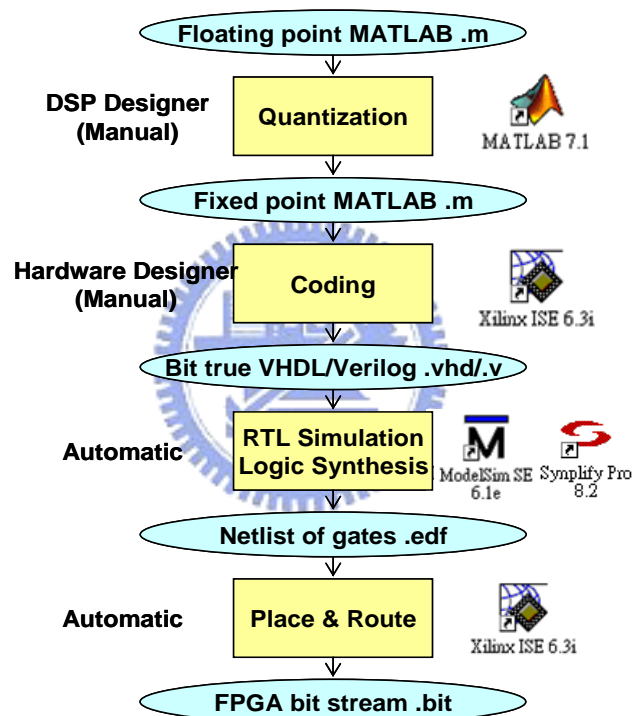


Figure 4.1: FPGA design flow

4.2 MATLAB Verification

As developing a communication system, MATLAB is one of the best candidates for us to model and simulate the system by means of its powerful matrix computation ability and well-defined communication functions. MATLAB matrix functions are shown to be versatile in doing analysis of data obtained in filter design or communication theory experiments. In addition, the interactive programming and

graphics of MATLAB also make designers easily illustrate the system performance with the effects of simulated channel and quantization error and so on. Therefore, in this section, the function blocks and adopted algorithms mentioned in Chapter 2 will be verified first, and then the whole system will be constructed and the system performance will be expressed.

1. RRC:

In our system, a 33-tap root raised cosine filter with roll off factor $\beta=0.25$ is designed, and its impulse response and frequency response is shown in Figure 4.2. In our system, the sampling rate is 40 MHz, which means that the data spectrum passes through the RRC pulse shaping filter cannot span wider than 20 MHz according to Nyquist sampling theorem. It can be clearly observed in the frequency response that signals with frequency higher than approximately 14 MHz are filtered, so that the waveform in time domain will become much smoother, and therefore can effectively combat the aliasing in AD/DA conversion and the ISI problem. Figure 4.3 shows the waveforms before and after RRC pulse shaping. Waveform in part (a) is the BPSK modulated short preamble. After the pulse shaping in transmitter side RRC, the smoother waveform will look like part (b). Next the waveform passing through RRC in the receiver side is shown in part (c). Finally, the eye diagram after RRC shaping is illustrated in Figure 4.4.

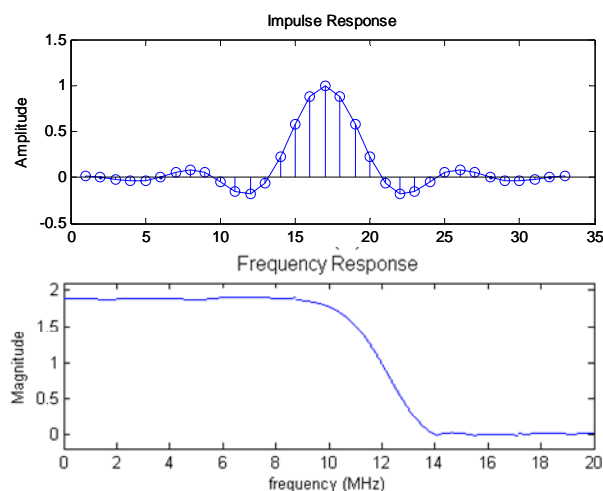


Figure 4.2: Impulse and frequency response of RRC filter with $\beta=0.25$

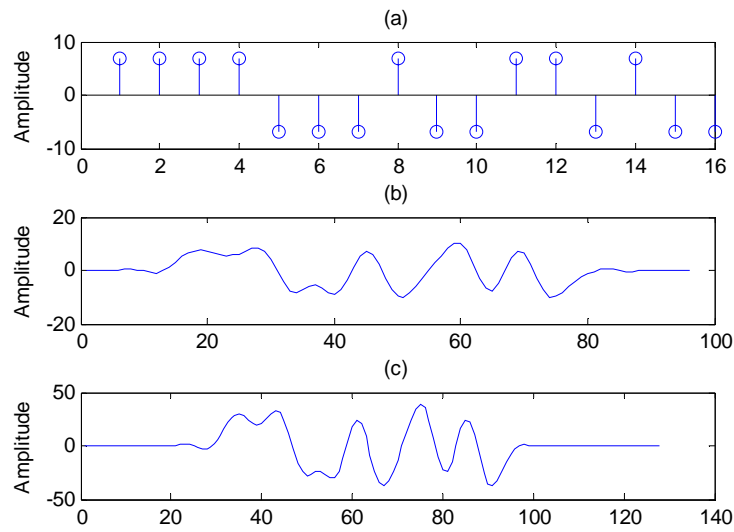


Figure 4.3: (a) Original waveform (b) RRC shaped waveform on transmitter (c) RRC shaped waveform on receiver

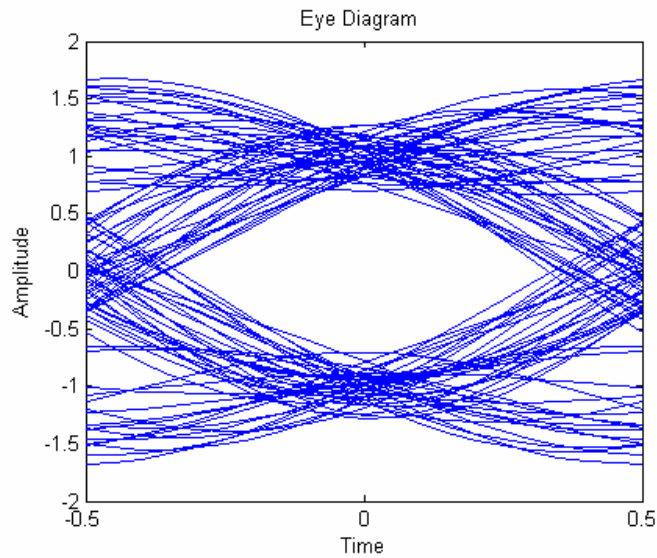


Figure 4.4: Eye diagram of RRC shaped waveform

2. Symbol timing recovery:

As we have mentioned in Section 2.3.1.2, a modified delay-lock-loop algorithm is adopted in our system. Here we pass our transmitted signal through a Rayleigh fading, multipath channel, and then process the post-RRC received signal by symbol timing recovery block. The waveform derived at the output of RRC is shown in Figure 4.5(a), while the waveform derived at the output of delay-lock-loop is shown in Figure 4.5(b). Compared with the

original short preamble pattern at the beginning of a packet, which is shown in Figure 4.5 (c), we can see that (b) and (c) are almost the same, except that the amplitude of (b) is not as constant as (c) due to the effect of multipath delay. Therefore, we have shown that the proposed symbol timing recovery algorithm not only performs the task of symbol rate reduction, but also take the correct sampling points precisely.

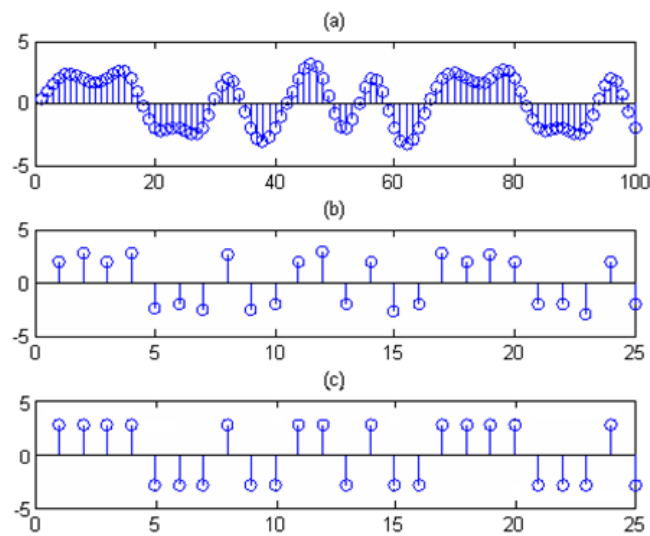


Figure 4.5: (a) Received oversampling waveform
 (b) DLL selected samples on receiver (c) Original samples on transmitter

3. Channel estimation:

The channel estimation result is shown in Figure 4.6, where the above one is the real channel frequency response, and another one is the estimated channel frequency response. We can see these two curves are almost the same though the long preamble channel estimation method.

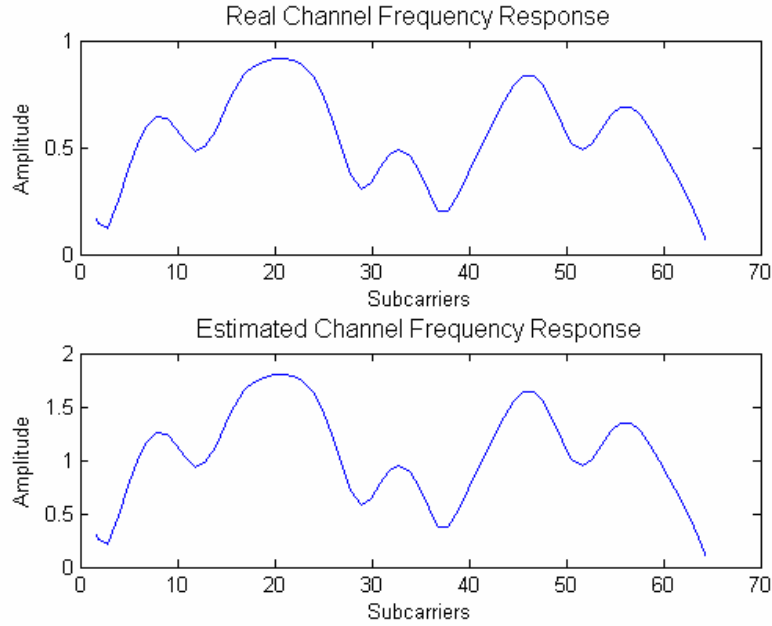


Figure 4.6: Real and estimated channel frequency response

4. Phase Tracking

The result of phase tracking algorithm proposed in Section 2.3.4 is shown as below. Figure 4.7(a) is the constellation with phase offset, and one can clearly identify the rotation caused by the residual carrier frequency offset. With the algorithm proposed, the constellation of the corrected frame is shown in Figure 4.7(b). We show that simple Unique Word based phase tracking algorithm provides almost optimum correction ability. On the other hand, the proposed algorithm involving symbol-by-symbol correction of linearly accumulated phase error is somewhat complicated for hardware implementation. For lower order modulation such as BPSK or QPSK that used in our system, to correct the constellation requires only that the constellation of each symbol falls in correct quadrant. Therefore, if we de-rotate the received symbols in one frame by the constant $e^{-j\Delta\Theta_n(T_{FFT}-T_G)/(2T_{FFT}-T_G)}$ instead of $e^{-j\Delta\Theta_n 2kT/(2T_{FFT}-T_G)}$, which is proposed in the phase tracking algorithm, the result is shown in Figure 4.7(c). This result, however, is enough for the slicer to determine the threshold for QPSK modulation, and the complexity to implement the algorithm is much lowered as well.

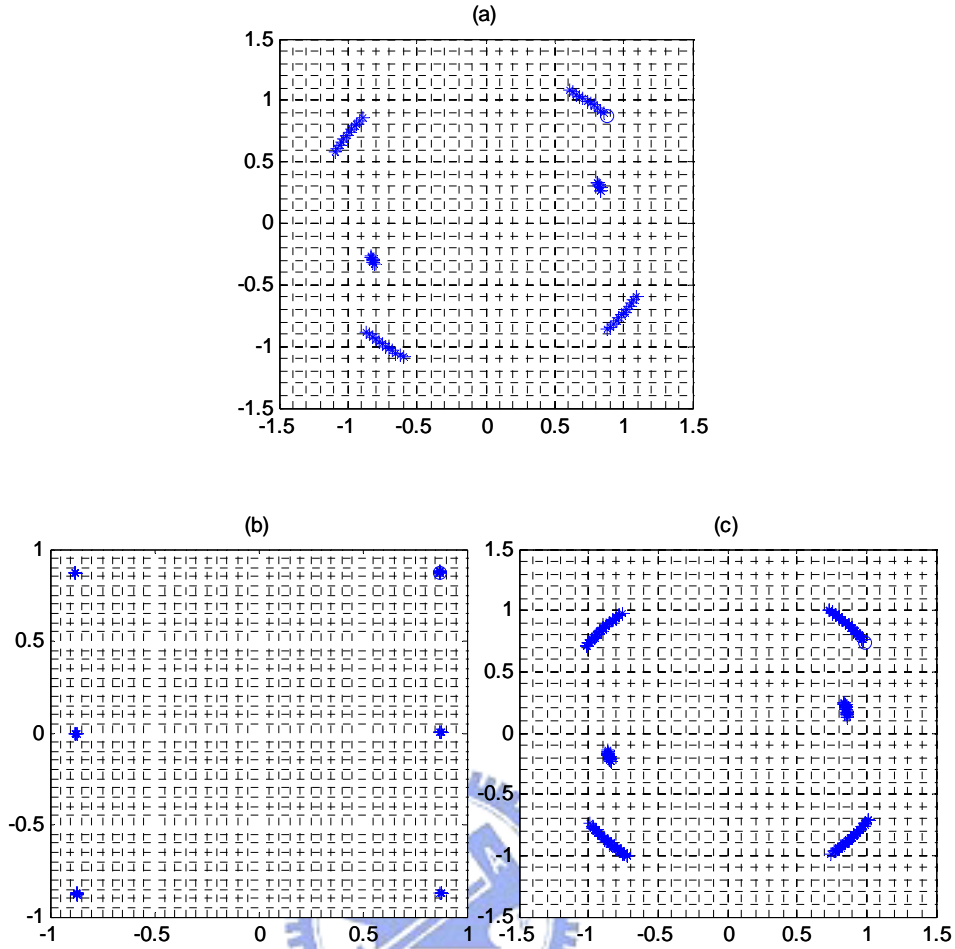


Figure 4.7: (a) One equalized block in the presence of residual CFO
 (b) One equalized block after the proposed phase tracking algorithm
 (c) One equalized block after the lower-complexity phase tracking algorithm

5. System Performance:

The BER to SNR system performance is shown in Figure 4.8, where a Rayleigh fading channel with AWGN noise is generated, and the total path number is three, including one main path and three multipaths. We can see that the performance of SC-FDE system and OFDM system is comparably the same. Besides, performance of OFDM in frequency selective fading is sensitive to the strength of its forward error correction (FEC) code, since the FEC code used by OFDM receivers must be powerful enough that its correction capability is not overwhelmed by the random occurrence of low SNR bits sampled from subchannels lying in frequency selective null regions. Therefore, performance of OFDM is a little bit inferior to that of the SC-FDE.

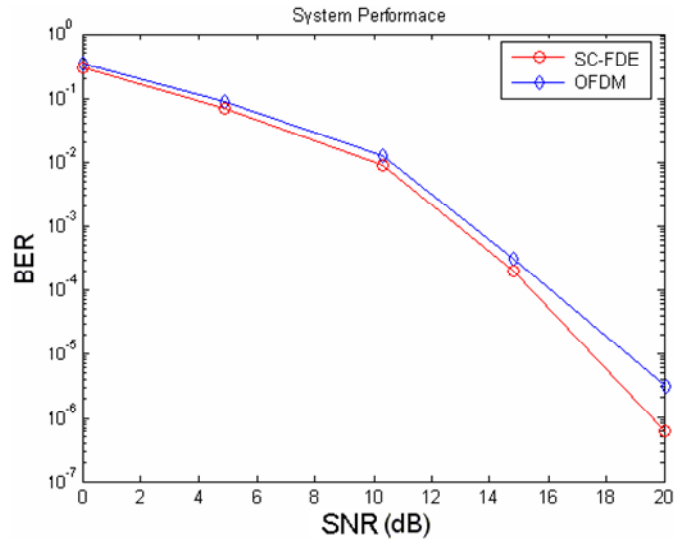


Figure 4.8: System performance of SC-FDE and OFDM

4.3 FPGA Realization

With the introduction of advanced FPGA architectures which provide built-in DSP support such as embedded multipliers and block RAMs on the Xilinx Virtex-II, a new hardware alternative is available for designers who can get even higher levels of performances than those achievable on general purpose DSP processors.

In our implementation, we adopt Xilinx Virtex-II series as our FPFA and VHDL as our hardware description language. The programming concepts that deserved to be mentioned in high level language like MATLAB and in hardware description language like VHDL are quite different. In general, high level language keeps its temporary data in a form of variables, and simply assigns the stored variable to another one which is used to be the input of next stage or functions if necessary, whereas hardware description language may need extra data buffer and related components to perform the same task. Since we have no choice but to add RAMs, FIFOs or register as data buffers, some index-related jobs or adding cyclic prefix or UW can be performed in the same time. The following sections will give readers more concepts and clear description about how we design in FPGA.

4.3.1 Design Principles

1. Parallel processing:

One of the biggest advantages of FPGA is its ability to support any level of parallelism. Complicated calculations can be divided into several simple operations and executed parallelly to enhance data throughput. The proposed phase tracking algorithm is a good example of parallel processing. Since each received symbol should be correct by two kinds of phase offset - the accumulated phase offset and the new phase offset generated in each symbol, the accumulated phase offset can then be corrected at the same time as the new phase offset is calculated by the Unique Word. The generally used function block, FIR filter, is another good example of parallel processing. An FIR filter consists of many multiply-and-accumulate operations so that and it is time-consuming if there is only a single, fixed multiply-and-accumulate used. To shorten the processing time, sufficient multiply-and-accumulate units are always utilized. There are other blocks utilizing parallel processing technique in this thesis which will be detailed later. By doing so, lot of buffer size can be saved and the area can be greatly reduced.

2. Resource Reuse:

The advantage of resource reusing is apparent but important. Obviously, if blocks in the design can be reused without affecting the result, the area of the design will be greatly saved. One of the resource reusing examples in our design is RAMs and ROMs. Since RAMs and ROMs are widely used and often occupy most of the area, reusing them will definitely save an appreciable resources. Another example in our design is to reuse certain large operation units such as multipliers and dividers, whose sizes grow with the width of the operands. Reusing those operation units will save a sizable area as well.

3. Independent Block Design:

A system is composed of lots of function blocks, and each block executes different functions. On designing we may sometimes modify certain blocks to achieve better performance or faster speed. It is therefore significant to design each block independently so that when particular blocks are updated, the functionalities of the whole system will not be affected. In our design, there is a pair of pins at every block call *rdy_in* and *rdy_out* which are used to “turn on” and “turn off” the block. With this design, every block is functioning independently and the designers can have a great freedom to modify the blocks or substitute the outdated blocks with the latest ones, without caring about adding delays and so forth.

4. Using FIFOs as Buffers Enables Pipelining:

Sometimes a complicated numerical computation is carried out in a block, and thus many summations and multiplications are serially executed in a path within a single clock period. Although high speed multiply-accumulators are embedded inside Xilinx FPGAs, these operations cannot be completely executed in time within a single clock period. Delays are often used as a solution to control the data flow between blocks. However, instructions in this delay-controlling structure can only be executed serially and critical path is long. To solve this problem, FIFO is used as an alternative solution. In our design, FIFO is inserted in the path and therefore the whole computation will be separate into few sections, each of which can be executed parallelly. Therefore using FIFOs enables pipeline processing, and the processing delay is greatly reduced.

5. Substitute real number computation for complex number computation:

Inevitably, large amount of complex number computations are included in our SC-FDE system. In MATLAB, these complex number computations can be easily computed, whereas become inconvenient in VHDL since complex number operations cannot be carried out directly in VHDL. Hence, in order to deal with complex number computations, the original complex

number arithmetic is separated into many real number segments. For example, one simple complex number computation, $(X+Y)/Z$, will become $(ae+ce+bf+df)/(e^2+f^2)+(-af-cf+be+de)i/(e^2+f^2)$ after the rearrangement, where $X=a+bi$, $Y=c+di$, $Z=e+fi$, and $a, b, c, d, e,$ and f are real numbers.

4.3.2 Circuit Design

In the following paragraphs, components are roughly divided into transmitter components and receiver components, and all circuits follow the principles introduced in the previous section. Additionally, every component is hierarchically designed.

4.3.2.1 Circuit Design of Transmitter

Figure 4.9 shows the overview of the circuit design of the SC-FDE transmitter. All the circuits are synchronized with the system clock, and are initialized by the system reset. In our design, one data symbol is generated every 16 system clocks. Therefore, no delay is used between any two blocks and all the blocks are independently designed by the *rdy_in* and *rdy_out* pins as mentioned above. This allows users to save a lot of space for delay and tremendously increases the performance. We can see that the transmitter design of the SC-FDE system is rather simple. Detailed circuit designs of function blocks are described as follows.

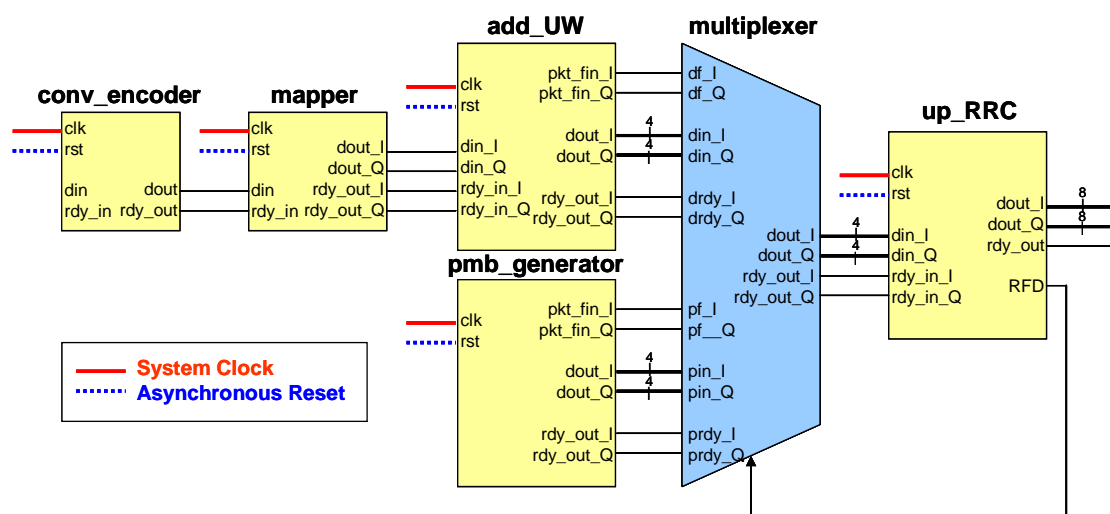


Figure 4.9: Circuit design of transmitter

(1) **Convolutional encoder:**

Figure 4.10 shows our circuit design of the convolutional encoder, named **conv_encoder**. Two entities are included in the **conv_encoder**, which are **conv** and **p2s**. The entity **conv** is the core of the convolutional encoder, which is responsible for generating coded bit, while **p2s** is used to serially output the three parallelly input coded bits. Both blocks are first initialized by the system reset to make sure that all the values at the output pins are set to specific initial values. The source data is then fed into the **conv** through pin *din*, and three coded bits, *da*, *db* and *dc* are generated simultaneously. Each source bit comes along with a strobe inputted to *rdy_in* pin, so that the **conv** will not function unless it is triggered by *rdy_in*. The same control mechanism is used at **p2s**. On the other hand, since one source bit comes every 16 clocks whereas the **p2s** requires only three clocks to output the three parallelly input data bits, no critical path problem will occur.

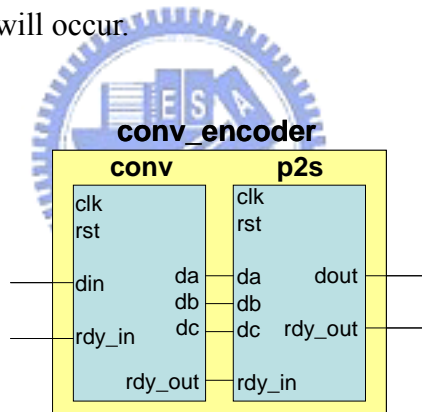


Figure 4.10: Circuit design of convolutional encoder

(2) **Mapper / de-mapper:**

The **mapper** and **de-mapper** blocks shown in Figure 4.11 and 4.12, respectively, are similar except that the data flow directions of the two blocks are opposite. In our system, QPSK modulation is adopted, so that coded bits are going to be modulated into in-phase and quadrature parts. Therefore, at every trigger of *rdy_in*, the **mapper** block functions as a switch that maps the input bit stream *din* to *d_I* and *d_Q* iteratively. Besides, it also outputs two control signals *rdy_out_I* and *rdy_out_Q* along with *d_I* and *d_Q*, respectively, as the trigger sources for the next stage. Note that since one source bit comes

every 16 clocks whereas the **mapper** requires only three clocks to output three coded bits, no buffer is required to store any intermediate value. Besides, though theoretically the output values of **mapper** are +1 and -1, we do not expand the output width but still use '1' to represent +1 and '0' to represent -1, since no arithmetic is required before the root-raised cosine filter. On the other hand, the **de-mapper** functions in a similar way except that the data flow is in an opposite direction. Two input data streams, din_I and din_Q , are sent into the **de-mapper** blocks with triggers rdy_in_I and rdy_in_Q , respectively. The **de-mapper** acts as a multiplexer that combines the two input streams into output $dout$.

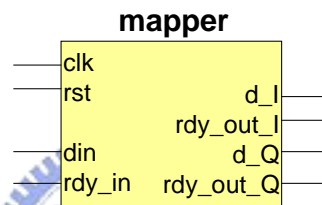


Figure 4.11: Circuit design of mapper

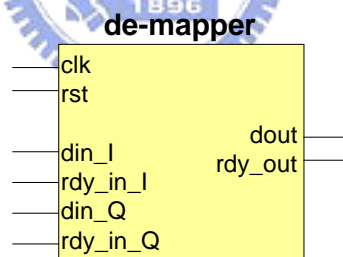


Figure 4.12: Circuit design of de-mapper

(3) Preamble generator:

As described in Chapter 2, preambles are BPSK modulated and consist of ten short preambles and two long preambles. Instead of using a large ROM to store the whole preamble symbol pattern with a ROM of depth $16 \times 10 + 16 + 64 \times 2 = 304$ symbols, two small ROMs called **st_rom** and **ln_rom** with depth 16 and 64 are used to store one short preamble pattern and one long preamble pattern, respectively. We repeatedly read out short preamble ten times and long preamble two times, then preamble channel is generated

with a lot of space saved. The preamble generator **pmb_gen** starts to function as soon as system is reset, and its output rate equals to clock rate, which is faster than normal data and UW symbol rate. This is because the 304 preamble symbols have to be generated as soon as possible since data and UW symbols are waiting.

Once preamble symbols are all sent, the *finish* pin of **pmb_gen** will be set to 1 as a flag for the multiplexer followed, and the *rdy_out* pin will also be set to 1 to start up the transmission of data and UW symbols. Meanwhile, a block called **new_pkt** is used to count the number of data symbols (excluding UW symbols) generated by the trigger of *rdy_in* from previous **mapper** block. This block is used to determine whether a packet of data symbols have been sent, and whether the preamble generator should start to function again. Therefore, when **new_pkt** is triggered for $48 \times 6 = 288$ times, it will output a *set* flag to reset the whole system. The preamble generator then starts to generate preamble symbols again after the reset.

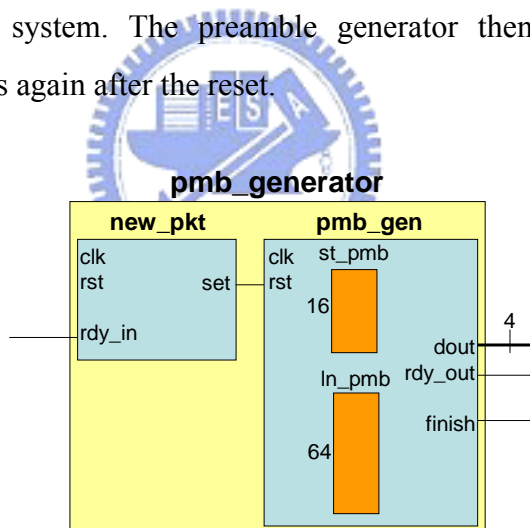


Figure 4.13: Circuit design of preamble generator

(4) Add the Unique Word to each frame:

The Unique Word is a known sequence that is added to the guard time between two frames as mention in Chapter 2. This operation is done in the **add_uw_i** and **add_uw_q** blocks, where a little ROM with depth 16 is used to store the UW values. To keep the symbol rate of output UW the same as that of input data, every UW symbol outputs when one data symbol inputs. Therefore, a synchronous FIFO (First-In-First-Out) of depth 16 is used to buffer the first

16 input data symbols when they are waiting for the transmission of UW, and these data symbols will first be generated in order as soon as the transmission of the 16 UW is finished. Besides, the word length of data symbols and UW symbols are extended at the **sign_ext** block. Since preamble is BPSK modulated and its value is set to be +7 and -7, the value of the QPSK modulated data symbols is set to be $\left[\pm 7/\sqrt{2} \right] = \pm 4$ such that the IQ gain of preamble and data symbols are approximately the same, which will benefit the design of amplifier. Therefore, we extend the word length of data symbol to 4 bits wide, which is the same as that of preamble symbols, such that both preamble and data symbols can share the input pin of RRC filter.

Another block called **pkt_fin** is a control flag that counts the number of bits that are outputted from this block. Once it counts to $(16 + 48) \times 6 = 384$, it will send a trigger signal through *finish* pin to indicate that all the data symbols as well as UW symbols are sent, which means that a packet of data symbols have been generated. This control signal together with the trigger signal from *finish* pin at preamble generator are used to control the multiplexer.

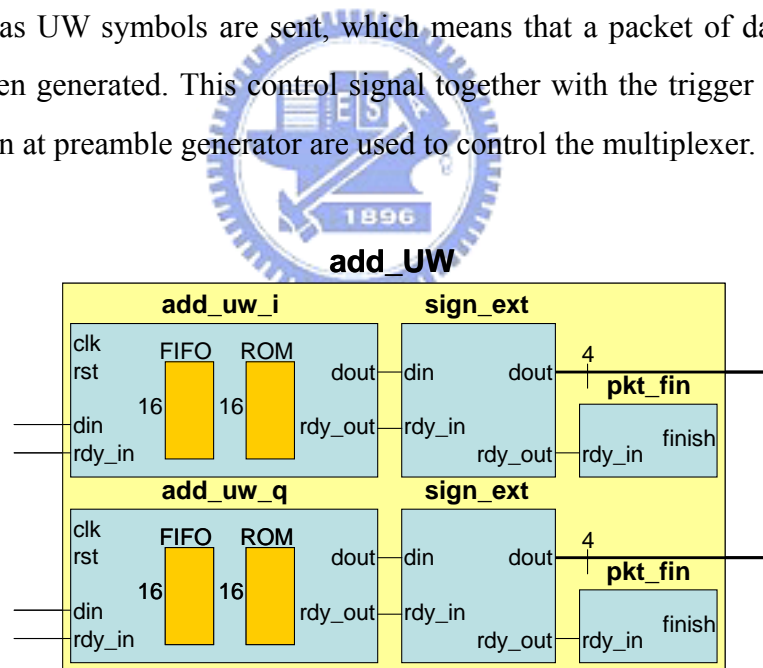


Figure 4.14: Circuit design of Unique Word generator

(5) Multiplexer

The multiplexer in our design acts a switch to pass the preamble symbols and UW-appended data symbols sequentially. The outputs of **selector**, *dout_I*, *dout_Q*, *rdy_out_I* and *rdy_out_Q* are initially connected to the corresponding preamble input - *pmb_I*, *pmb_Q*, *pmb_rdy_in_I* and *pmb_rdy_in_Q*,

respectively. Once the transmission of preamble symbols is finished and the *finish* flag of the preamble generators are set, the output of the **selector** will switch to the corresponding data symbol input - *data_I*, *data_Q*, *data_rdy_in_I* and *data_rdy_in_Q*, respectively.

Besides, since there is a root raised cosine filter following the multiplexer which requires several clocks to process one input symbol, a FIFO called **polyphse_buf** is used to queue the symbols that are to be processed. Data in **polyphse_buf** will be read out if “the buffer is not empty” and “the filter behind is ready for data”. This state is monitored by the **polyphse_buf_ctrl** block.

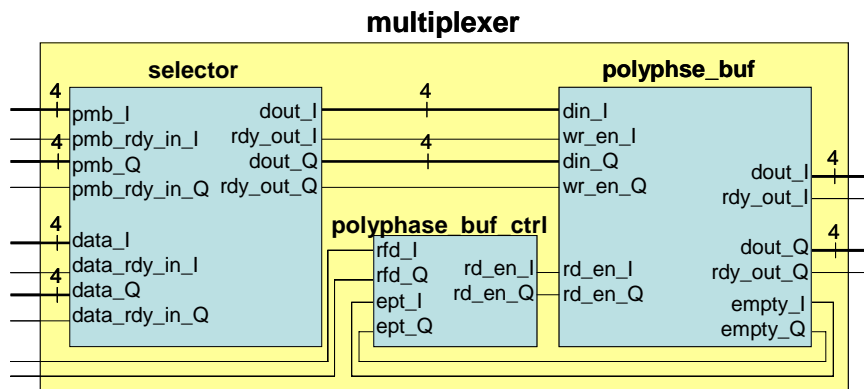


Figure 4.15: Circuit design of multiplexer

(6) Upsampler and Root-raised cosine filter :

In our thesis, the upsampler and root-raised cosine filter are co-designed as the polyphase structure as mentioned in Chapter 2. Since the number of taps of the RRC filter is 32 and the upsampling rate is 4, we will then have four polyphase interpolator filters, each with eight coefficients. Figure 4.16 shows the circuit design of the one polyphase interpolator filter. We can see that all eight multiply-and-accumulate operations are executed in one clock cycle; such parallel processing can maximize data throughput.

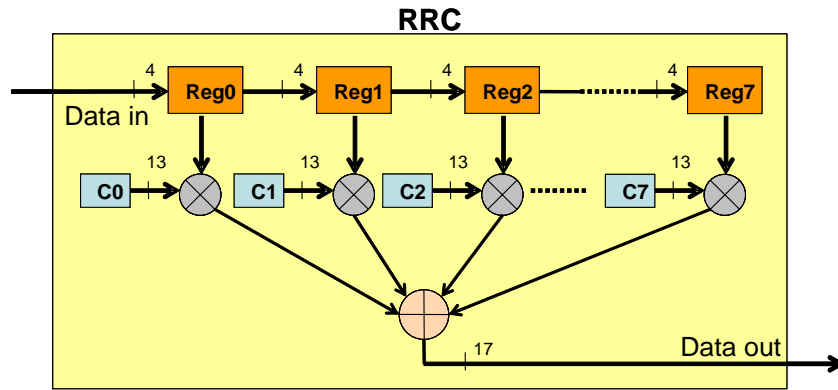


Figure 4.16: Circuit design of polyphase filter

4.3.2.2 Circuit Design of Receiver

Figure 4.17 shows the overview of the circuit design in the receiver. Certainly, a pipelined architecture is adopted as in transmitter. The circuit designs of function blocks are given as follows.

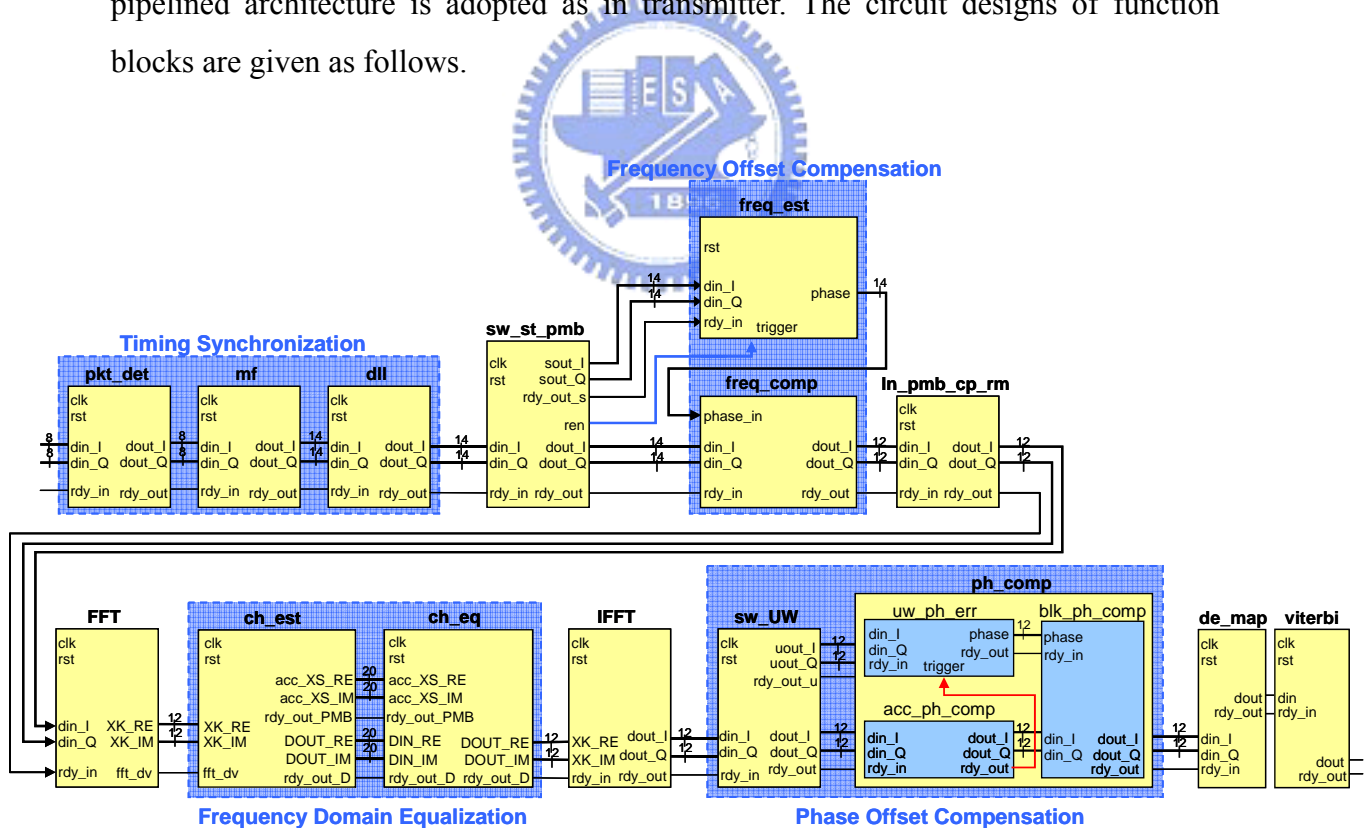


Figure 4.17: Circuit design of receiver side

(1) Packet Detection:

The packet detection block, **pkt_det**, is an implementation of double sliding window packet detection method. The design block diagram is shown in Figure 4.18. In floating point MATLAB verification, two shift registers with length five are used. In realization, only six Delay Flip-Flops (DFF) are used since we merely need the accumulation results in the two registers – five of them correspond to register A and the other one corresponds to register B in the algorithm mentioned in section 2.3.1.1. Received symbols are shifted in the six DFFs and at every shift, total power of symbols in register A and register B are calculated and the ratio of them is derived. Once the ratio exceeds the specific threshold, the **switch** is triggered and turned on and all the symbols can pass to the next stage. Therefore, the **pkt_det** block plays the role of a power controller since all the other blocks behind will not function before a packet is detected.

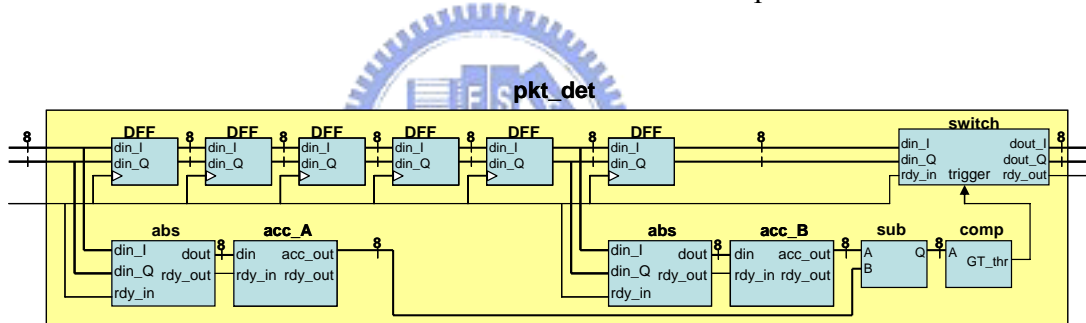


Figure 4.18: Circuit design of double sliding window packet detection method

(2) Symbol timing recovery (Delay-Locked Loop):

The **dll** block composes of three parts. The first and second parts, **dll_par** and **dll_algo** are truly the implementation of the algorithm. The third part, **dll_count**, is a control block that monitors the output of the whole **dll** block. In **dll_par** block, every three input symbols are collected as the detection set, and three relationships among them are determined: whether the amplitude of the on-time symbol is greater than the amplitude of the correct sample, whether the difference between the early and late samples are too large, and whether the sign of the on-time sample is equal to the sign of the difference between early and late samples. Once this information is collected, it is forwarded through pin

$d2_GT_thr$, $d3_d1_GT_diff$, $sign_d2_EQ_d3_d1$, to the second block, **dll_algo**, which performs the DLL algorithm, determines which sample in the detection set should be the output, and calculates the hop distance to next detection set. On the other hand, since tracking process of DLL algorithm is time consuming, the **dll_count** block is used to monitor the number of output symbols from **dll_algo** block to control the lifetime of the tracking. Once twenty consecutive samples outputted from **dll_algo** are on-time samples – the detection sets in twenty consecutive tracking processes are at an interval of four samples, we say the tracking loop is locked. Therefore, the **dll** algorithm will no longer be performed and all the rest of input symbols are downsampled by 4 directly. The **dll_count** block will monitor this phenomenon and determine when to turn off DLL tracking algorithm.

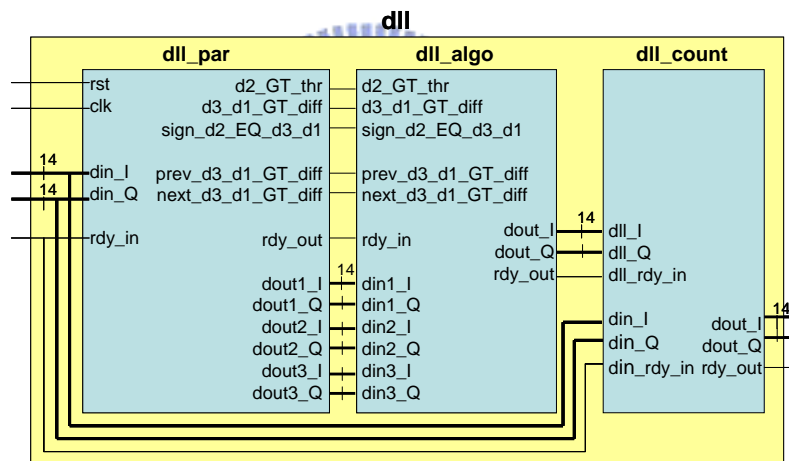


Figure 4.19: Circuit design of the delay-locked loop

(3) Frequency Offset Compensation :

According to Eq.(2.8) in Chapter 2, frequency offset is estimated by identical samples of repeated symbols. In our thesis, frequency offset is estimated by short preambles, yet only five short preambles – the third, fourth, fifth, sixth and seventh ones – are used in the task, so that four times of estimation are done. Since four is the second power of two, it is easily to average the four estimation results by discarding the two LSB of the summation result instead of using a divider.

Evolved from Eq.(2.8), let $z_i = \sum_{n=0}^{L-1} r_{i,n} r_{i,n+D}^*$ denote the intermediate result from the i -th and the $(i+1)$ -th short preamble, the frequency offset estimated by z_i is then $\Delta \hat{f}_i = -\frac{\angle z_i}{2\pi D}$ where D is the length of one short preamble. Since we take four times of estimation, the average frequency offset is derived by:

$$\Delta \hat{f} = \frac{\sum_{i=1}^4 \Delta \hat{f}_i}{4} = -\frac{\sum_{i=1}^4 \angle z_i}{4 \cdot 2\pi \cdot 16} \quad (4.1)$$

Let $y[n]$ be the data sequence that is to be frequency-offset compensated. The compensation is then:

$$\begin{aligned} & y[n] e^{j2\pi \cdot 4 \cdot \Delta \hat{f} n} \\ &= y[n] e^{-j \frac{\sum_{i=1}^4 \angle z_i}{16} n} \quad \text{let } \angle z_{sum} = \sum_{i=1}^4 \angle z_i \\ &= y[n] \left\{ \cos\left(\frac{\angle z_{sum}}{2^4} n\right) - j \sin\left(\frac{\angle z_{sum}}{2^4} n\right) \right\} \end{aligned} \quad (4.2)$$

Therefore, the five short preambles used are first separated from the data stream and fed into the **freq_est** block, and the value of $\frac{\angle z_{sum}}{2^4}$ is first calculated. Once long preamble symbols and data symbols are coming later, they will trigger the **freq_est** block to output the values of $\cos\left(\frac{\angle z_{sum}}{2^4} n\right)$ and $\sin\left(\frac{\angle z_{sum}}{2^4} n\right)$, where n accumulates according to the number of input symbols by counting the number of triggers from *count_in*. The estimated offset is then compensated in the **freq_comp** block. The full design diagram is shown in Figure 4.20.

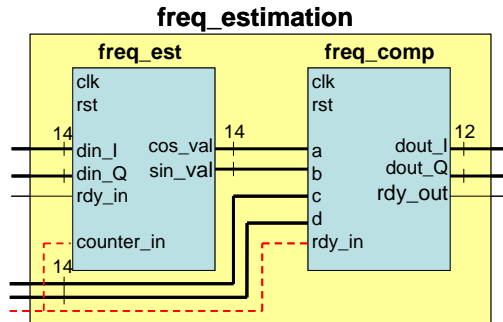


Figure 4.20: Circuit design of the frequency offset compensation block

(4) FFT/IFFT :

Fast Fourier transform (FFT) is a type of discrete Fourier transform (DFT), but only faster with fewer computations (summations and multiplications). A DFT takes N^2 computations to calculate a transform for N points, whereas the FFT takes around $N \log_2 N$ computations to complete the same thing. Here we adopt a 64-tap FFT which is provided by Xilinx and can operate 12-bit complex (12-bit real, 12-bit imaginary) samples, and a rough design concept is illustrated in Figure 4.21.

A pipelined implementation of a 64-point FFT requires a simple pipeline consisting of 6 butterfly computation modules. This method operates on two data points per clock cycle, yielding an effective data rate that is twice the clock rate, but requires customized butterfly computation modules for each stage of the FFT computation. Since a butterfly computation is carried out, the output signal will be in bit-reverse order.

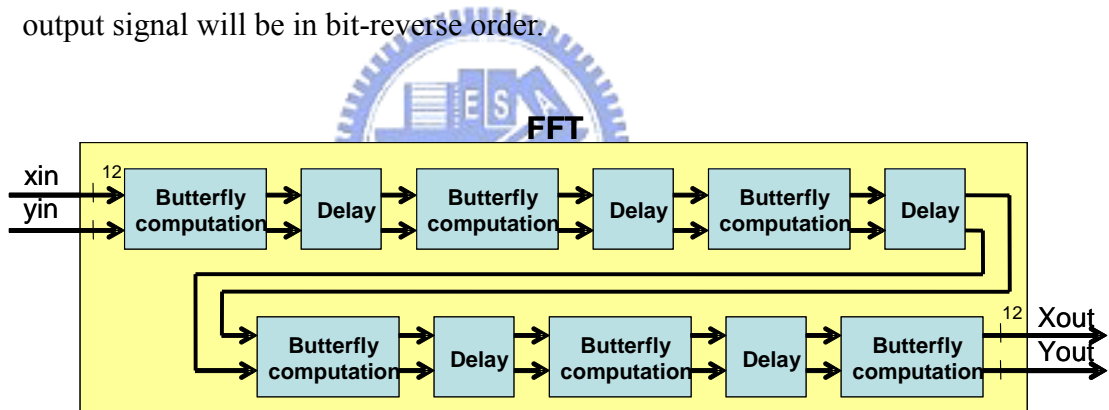


Figure 4.21: Circuit design of fast Fourier transform

(5) Channel estimation and Frequency Domain Equalization:

Long preambles are used to carry out the major task of channel estimation. Conventionally, channel frequency response is first estimated and stored, and channel equalization is another story followed. From Eq.(2.10) and Eq.(2.11) in Chapter 2, conventional channel estimation and equalization strategy is shown in Fig 4.22. It is clear that two division operations are required; besides, the equalization cannot be done –the input data symbols have to be buffered – until the estimated channel response is derived.

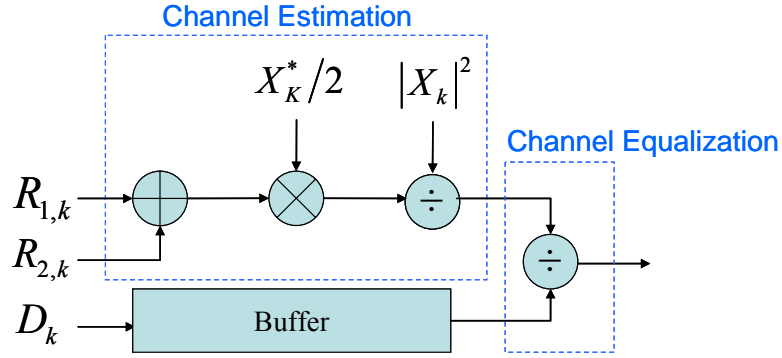


Figure 4.22: Conventional channel estimation and equalization strategy

In this thesis, we combine Eq. (2.10) and (2.11) into Eq.(4.3) as shown below, and the new strategy is shown in Fig. 4.23.

$$\hat{S}_k = \frac{2 \cdot D_k \cdot |X_k|^2}{R_{1,k} \cdot X_k^* + R_{2,k} \cdot X_k^*} \quad (4.3)$$

Obviously, only one division is required now. On the other hand, channel estimation and equalization are done parallelly, and data also share the task of channel equalization.

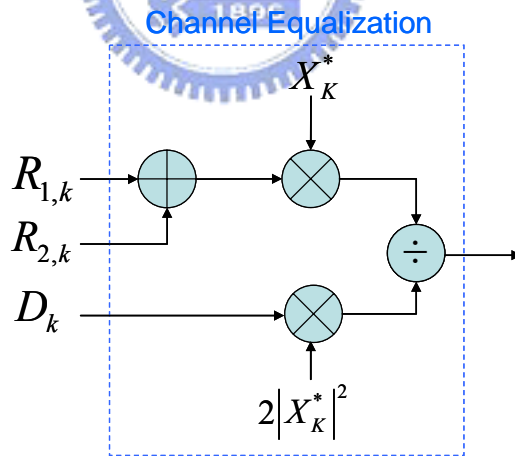


Figure 4.23: Modified channel estimation and equalization strategy

We observe that the numerator of Eq.(4.3) is the multiplication of received preamble and known transmitted preamble, $|X_k|^2$, while the denominator is that of the received data with known X_K^* . With this analogue, operations at numerator and denominator can be done by sharing one multiplier, which complies with the design principle: resources reusing.

Figure 4.24 shows the design block of channel equalizer, called **ch_EQ**. The known sequence, $|X_k|^2$ and X_K^* , are stored in two ROMs called **abs_xk** and **conj_xk** in **mul_sel** block, each with depth 64. The **mul_sel** block monitors the inputs and determines what will the multiplier and multiplicand to the multiplier **mult** behind be. If input is preamble, the stored X_K^* will be sent together with the preamble to the multiplier; on the contrary, the data $|X_k|^2$ will be sent if the input is data. Finally, the result of multiplication of preamble and X_K^* will be sent to the **half_ch** block since it stores the estimation of “half” of the channel value in a ROM called **half_ch_est** with depth 64. When the multiplication of data and $|X_k|^2$ is done, its result and the values in **half_ch_est** will be sent together to the **divider** block and complete the frequency domain equalization operation.

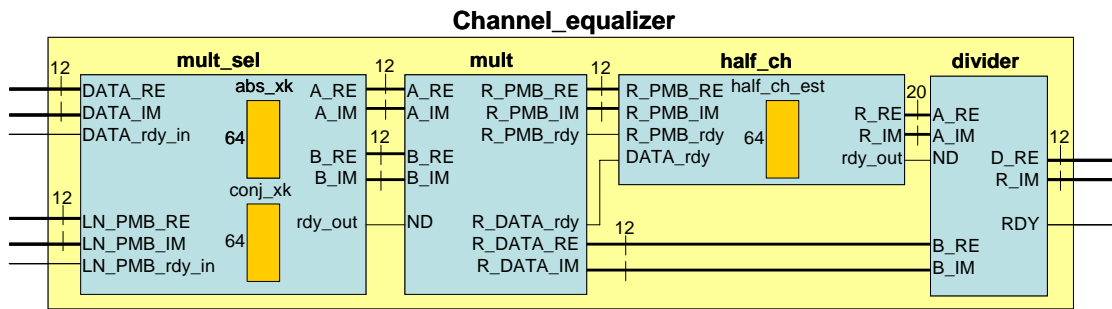


Figure 4.24 Circuit design of channel equalizer

(6) Phase offset compensator:

In Section 2.3.4, we have discussed the phase estimation algorithm. It is clear that the phase offset compensation is done in two steps: the compensation of the accumulated phase error and the compensation of the phase error inside one frame. On the other hand, the accumulated phase error is known at the very beginning of one frame, whereas the phase error inside one frame is not known until the phase offset estimation is derived by UW at the end of a frame. The algorithm then inherits a good nature of parallel processing: when the accumulated phase error is being compensated, the phase offset inside one frame can be estimated by UW. Figure 4.25 shows the design of phase estimator. The input symbols are first divided into data symbols part and UW symbols part by the **UW_sw**. The UW symbols are then used to estimate the

phase error inside one frame in the **UW_ph_err_est** block. Meanwhile, the input data symbols are firstly compensated by the accumulated phase offset in the **acc_ph_err_comp**. As the phase offset inside the n -th frame, $\Delta\Theta_n$, is derived, it will be written into a RAM called **acc_ph_err** in the **acc_ph_err_comp** block, and the data symbols then should be secondly rotated by $\Delta\Theta_n$. In our proposed algorithm, this rotate value is $(\Delta\Theta_n \cdot 2kT)/(2T_{FFT} - T_G)$, and a simplified algorithm proposed in Section 4.2, it is $\Delta\Theta_n (T_{FFT} - T_G)/(2T_{FFT} - T_G)$. The rotate value can be prepared and modified in the **rot_val** block by the designer. Finally, once the rotate value is ready, the data symbols can be further compensated.

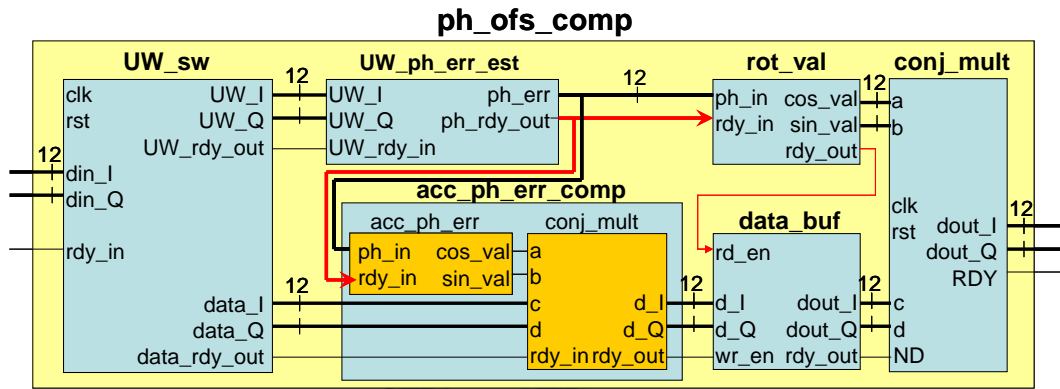


Figure 4.25 Circuit design of phase offset compensator

(7) Viterbi decoder:

The circuit design of the Viterbi decoder is shown in Figure 4.26. Three main blocks are included: branch metric generator (BMG); add, compare, and select (ACS) block; and the trace back unit (TBU). The BMG unit generates the branch metrics for each symbol of the input sequence by comparing the received code symbol with the expected code symbol for each connection of the trellis (state) and counts the number of different bits. For a 1/3 rate code adopted in our system, there are eight possible symbol combinations in the encoded sequence: 000, 001, 010, 011, 100, 101, 110, and 111; therefore eight BMG units are implemented in BMG block as shown in Figure 4.27.

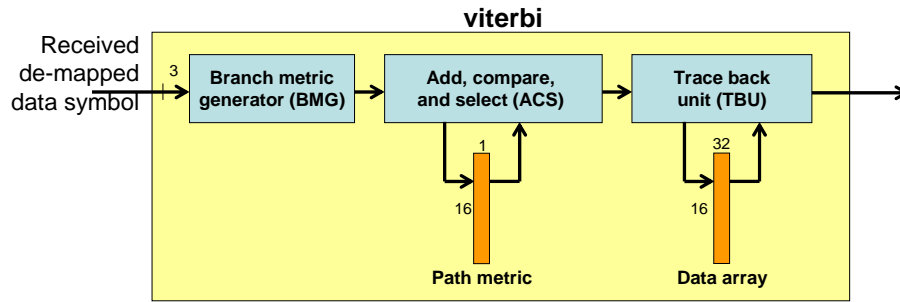


Figure 4.26: Circuit design of Viterbi decoder

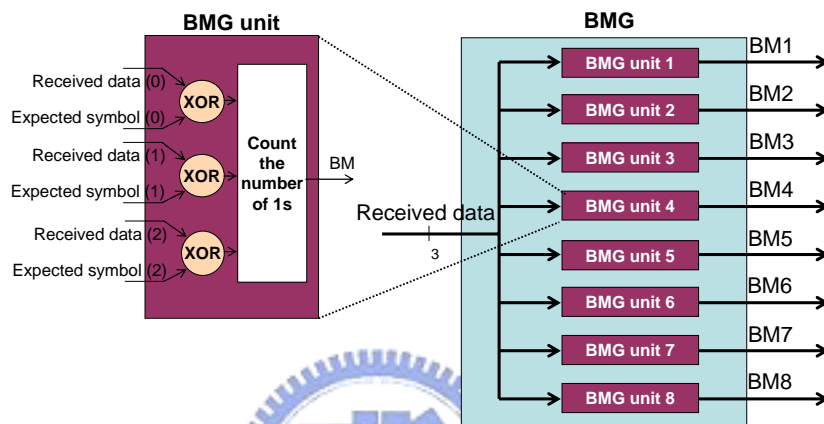


Figure 4.27: Circuit design of branch metric generator

The ACS unit is the heart of the Viterbi decoder. Each node in the trellis diagram corresponds to an ACS unit in the corresponding Viterbi decoder. Therefore, referring to the trellis diagram shown in Figure 2.17, there should be totally 16 ACS units in the ACS block as shown in Figure 4.28. The ACS unit has 4 inputs (two branch metrics and two path metrics) and two outputs (the new path metric and the survivor bit). The survivor bit is the most important information generated by the ACS unit. It indicates which sum between an input path metric and a branch metric generated the smallest result and was selected as the output path metric or local winner.

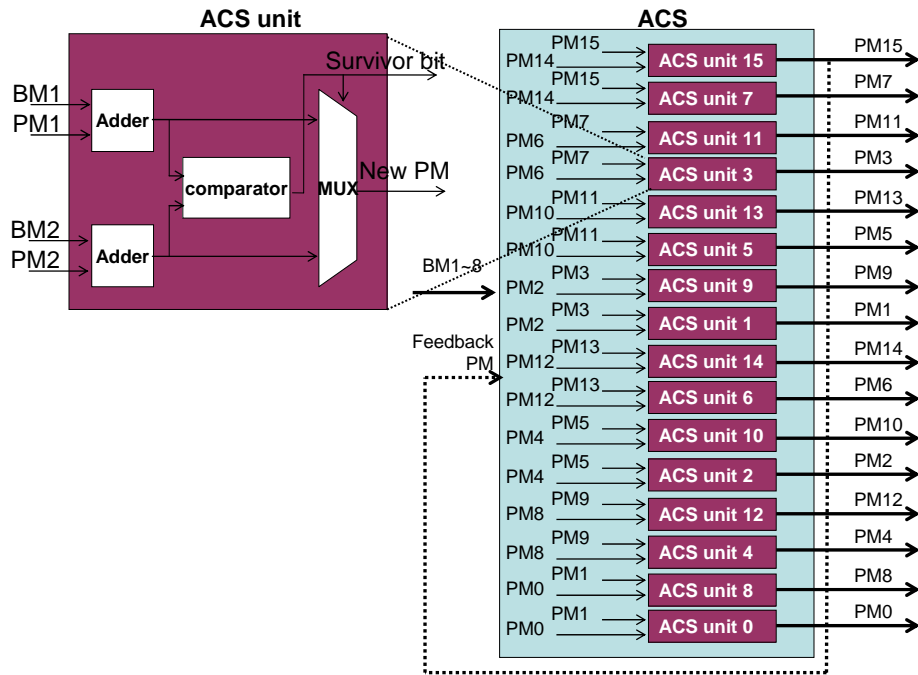


Figure 4.28: Circuit design of add, compare, and select block

The ACS block assigns the measurement functions to each state, but the actual Viterbi decisions on encoder states are based on the trace back operation to find the path of the states. Using the trace back operation, every state from a current time is followed backwards through its maximum likelihood path. The point at which the corrected bit streams starts is called the merger point (also called the trace back depth). The performance of Viterbi decoder largely depends upon the trace back depth. The increase in trace back depth increases the complexity and hardware exponentially so one has to trade off between the performance level and the complexity and hardware.

Normally for decoders using non-punctured codes, the trace back depth equals five-times constraint length, which is sufficient to decode the correct output in the presence of noise. In our system, the constraint length is 5, therefore twenty-five trace back depth is required. We adopt a 16x32 register array to store the path of the states.

4.4 ModelSim Simulation

When developing an FPGA system, ModelSim simulation can help designers develop in an efficient and accurate way. It can pull out all signals and simulate how they work simultaneously without a limitation on the number of debugging pins. Therefore, designers can save a lot of time downloading to FPGA and directly examine the changes and interactions between signals. Figure 4.29 and 4.30 show the data flows at the transmitter and receiver of the SC-FDE system, respectively. The six data symbols are conspicuously shown. Besides, from the figures certain design principles such as parallel processing is observed. In Figure 4.31 the output waveform of the transmitter is enlarged to show the real transmitted baseband signal. The IEEE 802.11a like output waveform with ten short preambles, two long preambles and six UW-appended SC-FDE data frames is clearly demonstrated, where the preamble part is BPSK modulated.

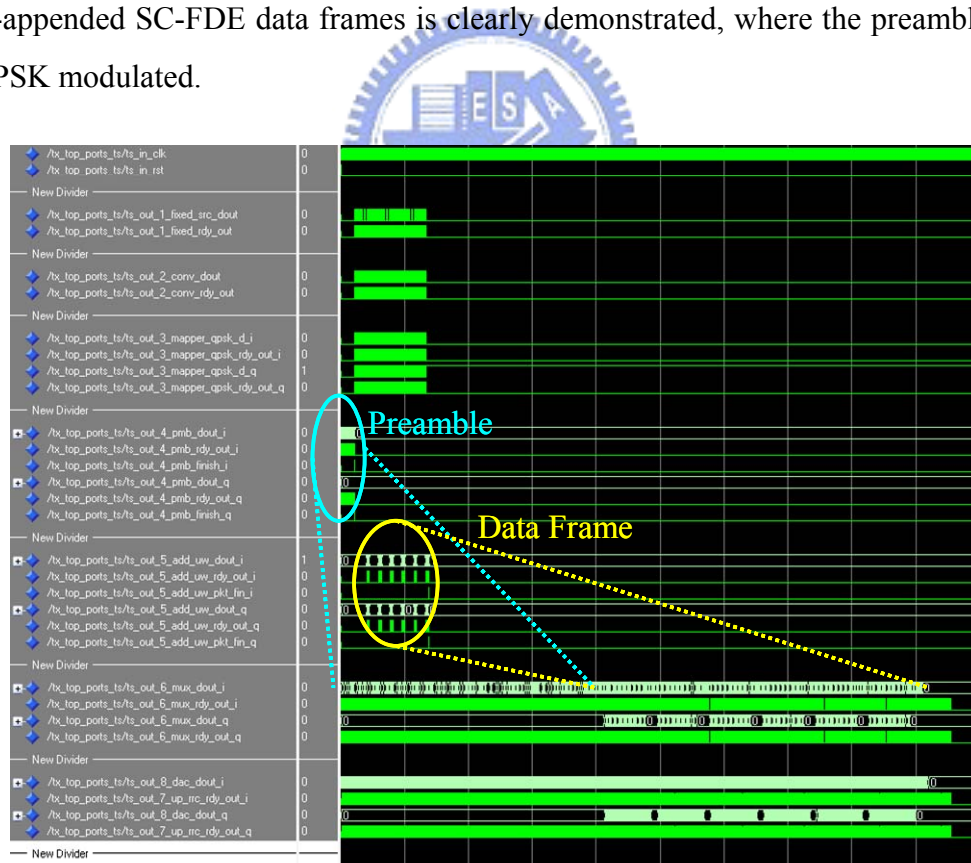


Figure 4.29: SC-FDE transmitter ModelSim simulation result

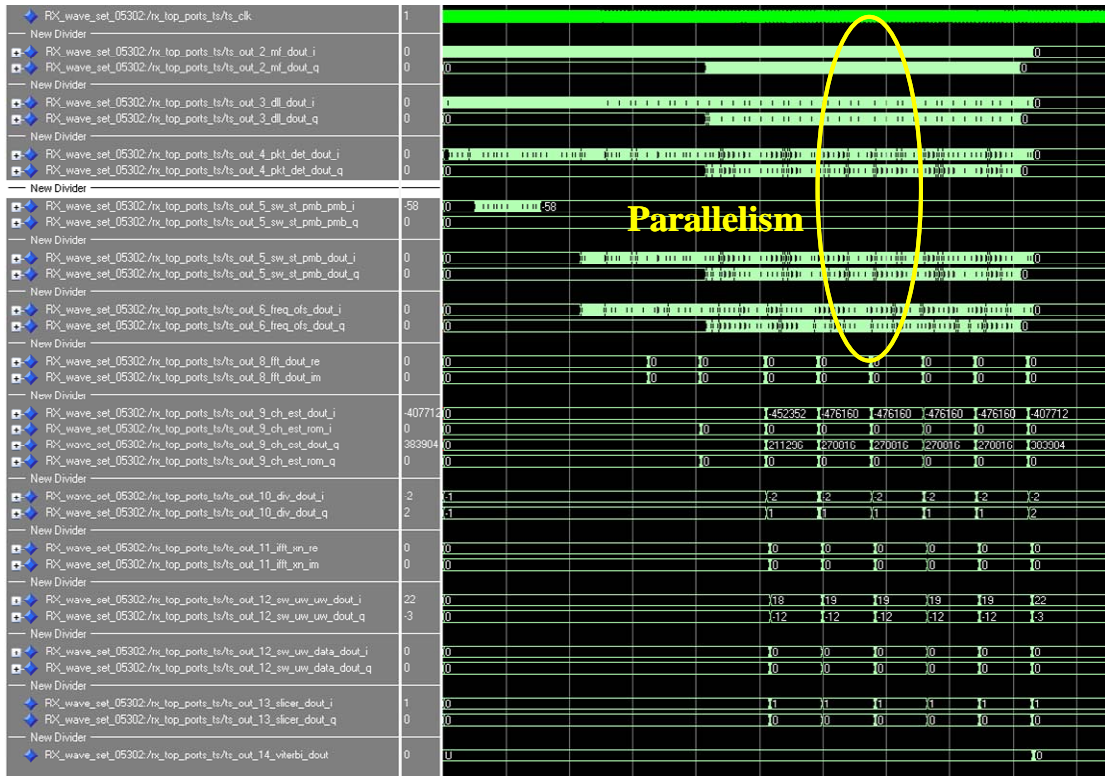


Figure 4.30: SC-FDE receiver ModelSim simulation result

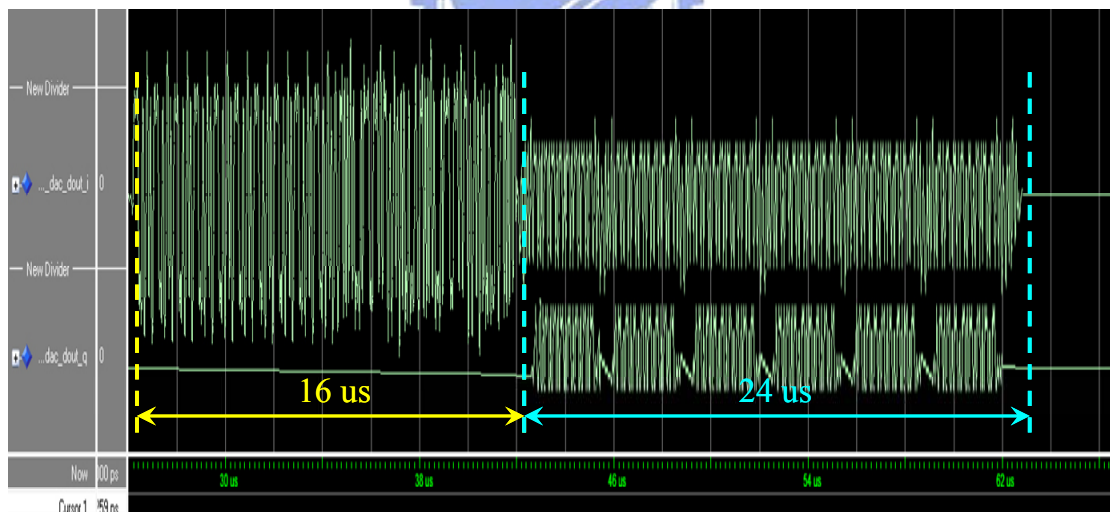


Figure 4.31: Transmitted waveform of SC-FDE system

4.5 Experimental Results

In the self-design platform, we attempt to establish a real wireless environment, under which the adopted algorithm can be tested. Figure 4.32 shows the experimental environment which has been shown in Chapter 3. First, source data are stored in a ROM in FPGA, and passed to DA after being processed by the transmitter algorithm on FPGA. Next, data are transmitted by the RF module, and a received antenna is allocated near the RF module. Subsequently data are received by the receive antenna and passed to spectrum analyzer E443A and vector signal analyzer 89600S. Finally, received data are analyzed and shown on PC.

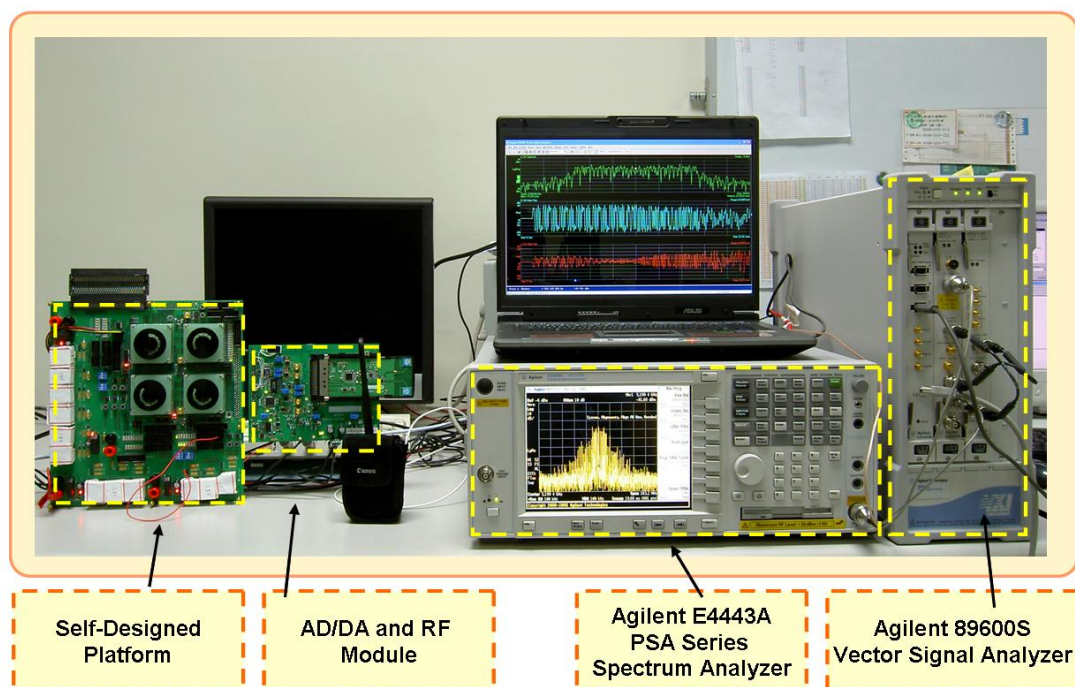


Figure 4.32: Self-designed platform development environment

Figure 4.34 shows the source data stream in the transmitter, transmitted data stream, and detected data stream in the receiver, where the source data stream and the detected data stream are specially expanded below. By comparing the source data stream with detected data stream we can find out that they are exactly the same, which confirms that our algorithm does work successfully.

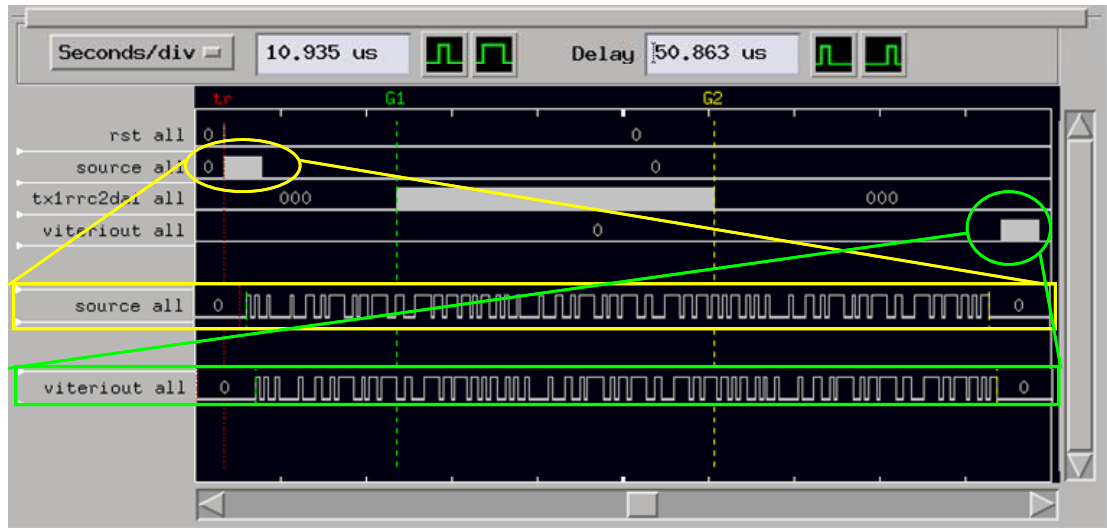


Figure 4.33: Self-designed platform experimental result: source data and detected data waveform on LA

Synthesis, map, and place and route are necessary steps in FPGA circuit design as well as the most time-consuming process. Besides, the insufficiency of FPGA gate count is another problem worth noting. The goal in our design is to achieve a compromise between the hardware resource requirement and the system time consumed. Table 4-1 and 4-2 show the relative resource consumption of transmitter and receiver. It can be seen that, at the transmitter side, the upsampler and RRC is the only part that has multiplication operation, while at the receiver side, FFT/IFFT are responsible for most of the complicated calculations. Finally, Table 4-3 shows time consumption in our development flow, and the whole design flow includes developing transmitter and receiver.

Table 4-1: Relative Resource consumption of the SC-FDE system at the transmitter

Selected Device : 2v6000ff1152-6						
	Slice	Slice FF	LUT	IOB	BRAM	MULT 18x18
add UW	45%	54%	53%	12%	0%	0%
Convolutional Encoder	4%	4%	5%	4%	0%	0%
QPSK Mapper	0%	0%	0%	5%	0%	0%
Mux	15%	17%	15%	30%	50%	0%
Preamble	25%	6%	15%	9%	0%	0%
Upsampler & RRC	11%	19%	12%	40%	50%	100%

Table 4-2: Relative Resource consumption of the SC-FDE system at the receiver

Selected Device : 2v6000ff1152-6						
	Slice	Slice FF	LUT	IOB	BRAM	MULT 18x18
Channel Estimation	10%	5%	9%	12%	6%	0%
Channel Equalization	28%	30%	21%	15%	0%	0%
Symbol Timing Recovery (DLL)	2%	2%	2%	5%	0%	0%
FFT	10%	12%	10%	6%	34%	47%
Frequency Estimation & Compensation	13%	17%	16%	6%	0%	0%
IFFT	10%	12%	10%	13%	41%	47%
Match Filter	1%	2%	1%	5%	19%	5%
Packet Detection	0%	0%	0%	5%	0%	0%
Viterbi	12%	5%	15%	2%	0%	0%
Unique Word Based Phase Tracking	14%	15%	16%	31%	0%	0%

Table 4-3: Time consumption of Synthesis and P&R in SC-FDE system

	TX	RX
Synthesis Time	2 min 37 sec	7 min 2 sec
Place and Route Time	3 min 58 sec	19 min 33 sec

4.6 Summary

In this chapter, a complete communication system design flow is presented, including MATLAB verification, FPGA realization, ModelSim simulation, and experimental results. Through this design flow, we developed a UW-based SC-FDE system on two FPGA-based platforms, where real wireless channel effects can be generated by means of RF module. We also introduce some RF debugging instruments which make our system become much closer to real communication systems. The designing principles we follow are described, and the designing concept of each function block is detailed. We especially show that the proposed UW-based phase tracking algorithm is not only theoretically suitable for the SC-FDE system but also practically applicable in hardware design. In Chapter 5, we will give some other applications based on UW which are also usable in SC-FDE system.

Chapter 5

Other Applications on Unique Word Structure in SC-FDE System

In this thesis we have considered and implemented the phase tracking algorithms for single carrier systems with frequency domain equalizers based on block-by-block of Unique Word (UW) insertion similar to that adopted in IEEE 802.11a OFDM systems. We focus on the design and performance of the algorithms, and show that it provides almost optimum performance in SC-FDE systems. Moreover, we also compare the result with pilot carrier based phase tracking algorithms in OFDM systems.

The Unique Word structure can be exploited not merely in the way described above. It can be observed that the overall baseband processing performance of SC-FDE systems largely depends on the design of channel estimation and synchronization algorithms. In fact, the deterministic properties of the UW give it a good nature to do various kinds of synchronization tasks as well as channel estimation especially in a mobile environment. Moreover, with the UW-based algorithms the SC systems can employ frequency-domain equalization at the receiver and benefit from low complexity which is suitable to implement on hardware. Therefore, in this chapter we investigate the use of UW and elaborate on the advantages it provides for equalization, channel estimation, and synchronization. A comparison between UW-based with CP-based SC-FDE system is also given, and their performance in terms of BER behavior and bandwidth efficiency are shown as well.

5.1 Cyclic Prefix versus Unique Word

The frequency-domain equalization for single carrier systems is based on the equivalence between the convolution of two sequences in the time domain and the product of their Fourier transforms. Besides, the use of FFT operations anticipates that signals have to be processed blockwise; not only applying blockwise processing at the received signal but also performing a blockwise transmission and inserting a cyclic prefix between successive transmitted blocks. The content of the CP is obviously not known and varies with every single block. With a slight modification to implement the cyclic prefix as a training sequence – the Unique Word in this thesis, however, it can play two important roles: avoid the inter-block-interference (IBI) and be used in synchronization and channel estimation. The topic of channel estimation is especially of utter importance in fast fading environment. In the following section we will show that the UW-based SC scheme (SC-UW) scheme offers the advantages at the expense of only a small fraction of a dB, while in other situations it has hardly any drawback compared to CP-based SC scheme (SC-CP) system.

Before introducing the algorithms taking advantage of the UW sequence that is provided, we can expect something *a priori*:

- From a performance point of view, the SC-UW scheme inherits from the properties of the SC-CP scheme: it offers a similar performance as for OFDM, with more robustness to nonlinear distortion and phase noise. Moreover, the UW sequence does not contain data. Hence, it can be optimized to get appropriate properties (e.g., autocorrelation) and its symbols could even be chosen from a separate alphabet. This avoids the accidental presence of the UW sequence in the useful data.
- From a synchronization point of view, the SC-UW acquisition is essentially the same as for the SC-CP: data-aided algorithms are known to perform better than their non-data-aided counterparts. They avoid decision directed algorithms and alleviate the problem of feeding the decisions back, which would mean a delay of one frame.

- For the channel estimation, the concept of UW is most useful when the channel is varying rather rapidly, like in mobile communications. The extension of SC-UW to the multiuser case (in a spatial division multiple access scheme) is easier than for SC-CP, as the users can be distinguished on the basis of their different UW.

5.1.1 Comparison of CP and UW in Terms of Bandwidth Efficiency and BER Behaviour

Figure 5.1 shows the structure of Cyclic Prefix and Unique Word. Two main differences are obvious when comparing the two concepts:

- The UW is not random as the CP
- Instead of having to throw away the cyclic prefix, we always process the UW, which is not removed at the receiver but is available after the equalization in the time domain. Hence, there is no gap anymore between two FFTs.

In practical situations, the FFT is usually taken in the middle of the UW to allow small timing synchronization errors. Moreover, as the UW is always present on both edges of the data block, the transformation from linear convolution to cyclic convolution is kept, and the performance of the original SC-CP is also kept.

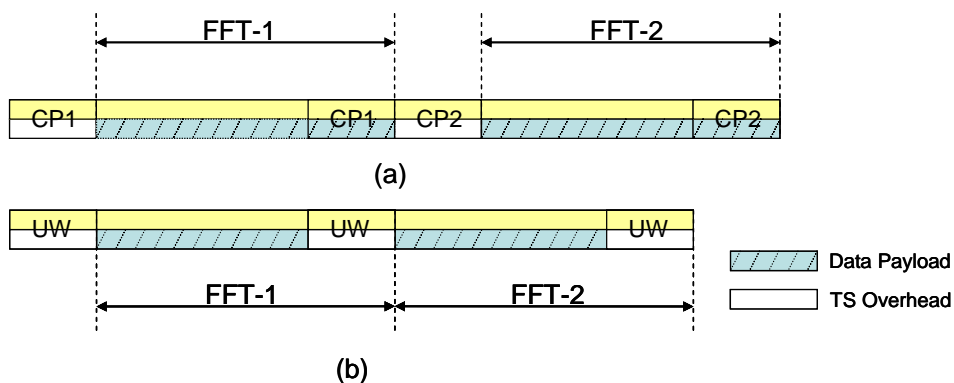


Figure 5.1: Single Carrier with (a) Cyclic Prefix and (b) Unique Word

The bandwidth efficiency is reduced for a SC-FDE by the guard period. Recall that T_{FFT} and T_G denote the FFT period and guard interval of a frame, respectively. The bandwidth efficiency of the described SC- CP and SC-UW systems without taking coding into account can be given as:

$$\begin{aligned}\eta_{CP} &= \frac{T_{FFT}}{T_{FFT} + T_G} = 0.8 \\ \eta_{UW} &= \frac{T_{FFT} - T_G}{T_{FFT}} = 0.75\end{aligned}\tag{5.1}$$

The result in Eq.(5.1) leads to an additional degradation of 5% in terms of bandwidth efficiency, assuming T_G to be 25% of T_{FFT} (In our thesis, frame length = 64 symbols and UW length = 16 symbols). Furthermore, a loss in terms of the BER behavior is expected, and a loss as a result of additional overhead compared to a single carrier system with time domain equalization is anticipated as well.

5.2 Application of the Unique Word Structure

Transmission over multipath channels makes channel estimation and synchronization not only necessary but also important. Due to the fact that the UW is known, it can be used for equalization, channel estimation, or synchronization purposes. In the following section some algorithms and results will be given for the mentioned application.

5.2.1 Synchronization

Synchronization is indispensable criteria for high data rate wireless transmission. In a time-invariant environment, initial channel estimation and block synchronization can be done by a preamble at the beginning of every burst . In time varying channels, however, clock-frequency-offsets or carrier-frequency-offsets make tracking necessary. Tracking is mainly based on the insertion of pilot symbols; implementing the structure of UW, pilot sequences are available automatically.

The variation of the sampling time between transmitter and receiver caused by the clock-frequency-offset will lead to rising displacement of the FFT-window. To solve the problem, an autocorrelation as shown in Eq.(5.2) of two consecutive, received UWs, denoted by u_k and u_{k+N} , which are separated by N symbols may result in distinctive correlation peaks if the symbols of the UW are chosen as to have good correlation properties (e.g. Pseudo noise sequences, Barker sequences)[15].

$$\phi(k) = \left| \sum_{k=1}^{N_G} \{ u_k \cdot u_{k+N}^* \} \right| \quad (5.2)$$

where u_k^* indicates the complex conjugate of u_k .

With the UW structure and a selected symbol sequence, this method shows conspicuous correlation peaks. Figure 5.2 shows the result of the autocorrelation, which indicates the beginning of every FFT window very precisely. The simulation is performed for $SNR = 25dB$ and multipath conditions; the UW is a PN sequence. Nevertheless it is to mention that, if due to the fact that UW is corrupted by the channel and noise on the one hand or the time duration of UW is too short on the other hand, the correlation of the two successive UWs may not show reliable enough correlation peaks – the autocorrelation properties of the investigated sequences are partly lost.

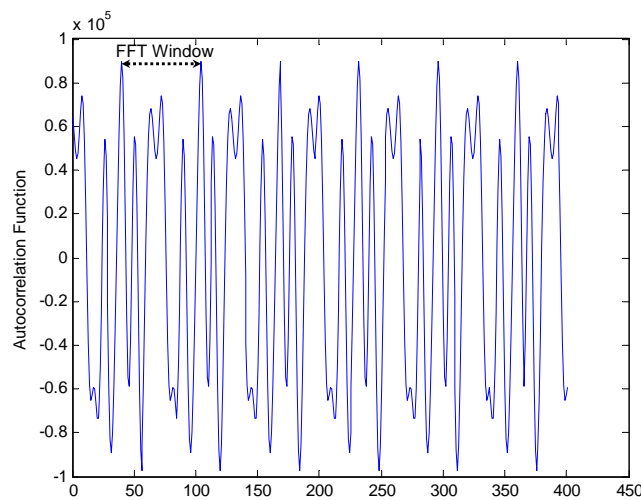


Figure 5.2: Synchronization and tracking of the FFT-window

5.2.2 Channel Estimation

Unavoidably, the equalization of the received message in systems employing single-carrier transmission is a fundamental problem in high data rate wireless communication, and performing the equalization requires knowledge of the channel. These parameters can be estimated relatively easily prior to data transmission in stationary environment; however, in non-stationary channels, they vary with time and must be tracked by some means.

The deterministic properties of the UW can be exploited in the channel estimation algorithms for SC-FDE systems to track a temporally fading channel. The algorithm introduced here relies on the ensemble averaging of the received signal to recover the channel state information (CSI). Although any time-domain or frequency domain equalization technique can be employed once the channel has been estimated, this algorithm lends itself to FDE systems since the use of the UW gives the system a cyclic nature.

■ *System Model*

Consider a system employing SC block transmissions with a UW extension. The i -th length- K block $\mathbf{x}(i)$ of transmitted symbols is partitioned into a length- P vector $\mathbf{s}(i)$ of data symbols and a length- Q vector \mathbf{u} representing the UW. An illustration of this block structure is depicted in Figure 5.3.

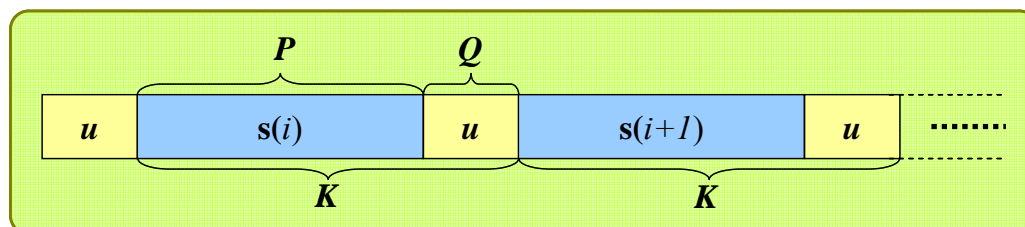


Figure 5.3: Basic UW block structure

In order to alleviate inter-block interference, we assume that $Q \geq L$ where L is the memory order of the channel impulse response (CIR). This condition also induces circularity in the system, which allows us to express the i -th length- K block of

received symbols $\mathbf{y}(i)$ by

$$\mathbf{y}(i) = \mathbf{h}(i)\mathbf{x}(i) + \mathbf{n}(i) \quad (5.3)$$

where $\mathbf{h}(i)$ is a $K \times K$ circulant matrix representing the channel at time i and $\mathbf{n}(i)$ is a length- K vector of uncorrelated, zero-mean, complex Gaussian noise samples, each with a variance of $\sigma_n^2/2$ per dimension. Specifically,

$$\mathbf{h}(i) = \begin{pmatrix} h_0(i) & 0 & \cdots & h_L(i) & \cdots & h_1(i) \\ \vdots & h_0(i) & \ddots & 0 & \ddots & \vdots \\ h_L(i) & \vdots & \ddots & \vdots & \ddots & h_L(i) \\ 0 & h_L(i) & & \ddots & & 0 \\ \vdots & & \ddots & & \ddots & \vdots \\ 0 & \cdots & 0 & h_L(i) & \cdots & h_0(i) \end{pmatrix} \quad (5.4)$$

where $h_m(i)$ is the m -th complex tap coefficient of the CIR at time i .

Denote \mathbf{F} as the normalized $K \times K$ DFT matrix where its (k,i) -th element is given by $F_{k,i} \triangleq \sqrt{K}^{-1} \exp(-j2\pi ki/K)$ for $k, i = 0, \dots, K-1$, and \mathbf{F}_m is the first m columns of \mathbf{F} while \mathbf{F}'_m is the last m columns of \mathbf{F} . Referring to Eq.(5.3), we consider the transformation of the received symbol vector $\mathbf{y}(i)$ into the frequency domain, which is given by

$$\mathbf{Y}(i) = \mathbf{H}(i)\mathbf{X}(i) + \mathbf{N}(i) \quad (5.5)$$

where $\mathbf{Y}(i) = \mathbf{F}\mathbf{y}(i)$, $\mathbf{N}(i) = \mathbf{F}\mathbf{n}(i)$, $\mathbf{X}(i) = \mathbf{F}\mathbf{x}(i)$, and $\mathbf{H}(i) = \mathbf{F}\mathbf{h}(i)\mathbf{F}^H$ is a diagonal matrix with the channel frequency response coefficients on the diagonal. In addition, the transmitted vector $\mathbf{X}(i)$ can be partition into a data part and a UW part as given by

$$\mathbf{X}(i) = \begin{bmatrix} \mathbf{F}_p & \mathbf{F}'_0 \end{bmatrix} \begin{bmatrix} \mathbf{s}(i) \\ \mathbf{u} \end{bmatrix} = \mathbf{S}(i) + \mathbf{U} \quad (5.6)$$

where $\mathbf{S}(i) = \mathbf{F}_p\mathbf{s}(i)$ and $\mathbf{U} = \mathbf{F}'_0\mathbf{u}$. Therefore,

$$\mathbf{Y}(i) = \mathbf{H}(i)\mathbf{S}(i) + \mathbf{H}(i)\mathbf{U} + \mathbf{N}(i) \quad (5.7)$$

■ *Channel Estimation Algorithm by UW*

If the channel is time-invariant (i.e. $\mathbf{H}(i) = \mathbf{H}$ for all i), the received frequency-domain vectors $\{\mathbf{Y}(i)\}_{i=1}^N$ can be treated as a sample set of a random process ψ where the mean of the process is given by

$$\bar{\psi} = \frac{1}{N} \sum_{i=1}^N \mathbf{Y}(i) \quad (5.8)$$

If a symmetric constellation such as QPSK is employed for data transmission and no *a priori* knowledge of the transmitted message is assumed, then $E\{\mathbf{S}(i)\} = 0$ and evaluating the expectation in Eq.(5.8) yields

$$\begin{aligned} \lim_{N \rightarrow \infty} \bar{\psi} &= E\{\mathbf{Y}\} \\ &= \mathbf{H}\mathbf{U} \end{aligned} \quad (5.9)$$

While channel varies with time, as is the case in mobile environments, the sample size N must be limited in some way to include only those blocks received within the last T_c seconds, where T_c is the coherence time of the channel. In this case, the recursive least square (RLS) algorithm can be employed with the cost function

$$\varphi(i) = \sum_{k=1}^i \rho^{i-k} |e(k, i)|^2 \quad (5.10)$$

where ρ is the standard RLS forgetting factor that is usually close to, but less than one. The error term $e(k, i)$ in Eq.(5.10) is defined as

$$e(k, i) = \mathbf{Y}(k) - \tilde{\mathbf{U}} \hat{\mathbf{H}}(i) \quad (5.11)$$

where $\tilde{\mathbf{U}} \triangleq D\{\mathbf{U}\}$ is the diagonal matrix with the elements of \mathbf{U} on the diagonal and $\hat{\mathbf{H}}(i)$ is a length- K vector of the i -th estimated channel frequency response coefficients.

Our goal is to find a channel estimation such that the error term in Eq.(5.10) is smallest. From the result derived from [20], taking the gradient of Eq.(5.10) with respect to $\hat{\mathbf{H}}(i)$, setting the result equal to zero, the channel estimate vector is given by

$$\widehat{\mathbf{H}}(i) = \frac{1}{\underbrace{\sum_{k=1}^i \rho^{i-k}}_{\mathbf{P}^{-1}(i)}} \widetilde{\mathbf{U}}^{-1} \underbrace{\sum_{k=1}^i \rho^{i-k} \mathbf{Y}(k)}_{\mathbf{r}(i)} \quad (5.12)$$

The channel can therefore be updated with the i -th received block by noting that

$$\mathbf{P}(i) = \rho \mathbf{P}(i-1) + \widetilde{\mathbf{U}} \quad (5.13)$$

$$\mathbf{r}(i) = \rho \mathbf{r}(i-1) + \widetilde{\mathbf{Y}}(i) \quad (5.14)$$

Note that Eq.(5.12) requires the inverse of $\mathbf{P}(i)$ to compute the updated channel estimate. Since $\mathbf{P}(i)$ is a diagonal matrix, however, it is easy to invert. Consequently, this method of mobile channel estimation benefits from very low complexity since only three complex multiplications are required to update the channel estimate on a given frequency tone. As with all applications of the RLS algorithm, this application requires the vector \mathbf{r} and the matrix \mathbf{P} to be initialized. If a reliable initial channel estimate $\widehat{\mathbf{H}}(0)$ is available (e.g. by the preamble based channel estimation), \mathbf{r} and \mathbf{P} can be initialized to

$$\mathbf{r}(0) = \beta \widetilde{\mathbf{U}} \widehat{\mathbf{H}}(0) \quad (5.15)$$

$$\mathbf{P}(0) = \beta \widetilde{\mathbf{U}} \quad (5.16)$$

where β is a positive real number. From Eq.(5.12) it is noted that

$$\sum_{k=1}^i \rho^{i-k} = \frac{1-\rho^i}{1-\rho} \quad (5.17)$$

Consequently, we may choose β to be

$$\beta = \lim_{i \rightarrow \infty} \frac{1-\rho^i}{1-\rho} = \frac{1}{1-\rho} \quad (5.18)$$

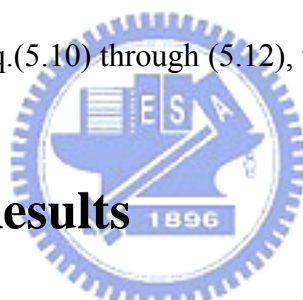
Defining β as above is equivalent to initializing the channel estimation by transmitting an infinite number of blocks containing only the UW over a static channel, which is denoted here by $\widehat{\mathbf{H}}(0)$, and computing $\mathbf{r}(\infty)$ and $\mathbf{P}(\infty)$. This definition of β produces good convergence results as shown in the simulation result later.

Also, a more complex version of the stochastic algorithm can be implemented. In this version, the received symbols $\mathbf{y}(i)$ are first equalized and the data symbols are detected. Using the length- P vector of the detected data symbols $\hat{\mathbf{s}}(i)$ and the previous channel estimate $\widehat{\mathbf{H}}(i-1)$, the contribution of the data to the received message is subtracted from the original received vector in the frequency domain to give

$$\mathbf{Y}_u(i) = \mathbf{Y}(i) - D \left\{ \mathbf{F}_p \hat{\mathbf{s}}(i) \right\} \widehat{\mathbf{H}}(i-1) \quad (5.19)$$

Replacing $\mathbf{Y}(i)$ with $\mathbf{Y}_u(i)$ in Eq.(5.10) through (5.12), the new channel estimate can be obtained

5.3 Simulation Results



The algorithm described in section 5.2 was implemented in computer simulations in order to observe its performance relative to other techniques. Two systems were simulated. In each of these systems, UWs were appended to every frame of QPSK data symbols to form blocks of $K = 64$ symbols. These blocks were transmitted over a 3-tap, exponentially decaying channel with a normalized Doppler spread of $f_D = 1.5 \times 10^{-6}$. The channel realizations were generated with a Rayleigh fading profile from burst to burst, and Jakes' model was used to simulate temporal fading within each burst. At the receiver, each system utilized its own knowledge of the channel to equalize the received message with a linear FDE. Each equalized symbol was then mapped to the nearest QPSK symbol.

The first system used an initial channel estimate, which was gleaned from a preamble, to construct a linear FDE, and only one channel estimate was obtained for each burst. The second system employed the stochastic channel estimation method

with feedback detailed in section 5.2.2. This system initialized the metrics \mathbf{r} and \mathbf{P} according to Eq. (5.15) and (5.16) where the initial channel estimate was obtained through a preamble. A forgetting factor of 0.96 was used. The two systems are summarized in Table 5-1.

Table 5-1: Summary of simulated SC-FDE systems

System	Channel Knowledge
1	Estimated by preamble only
2	Estimated by preamble and updated by UW

Figure 5.4 depicts the probability of bit error of each of the systems described above. It is observed that the system that employs stochastic channel tracking performs better than the preamble-only system. Indeed, the system employing a preamble-based only channel estimation suffers greatly even in this slow-fading environment.

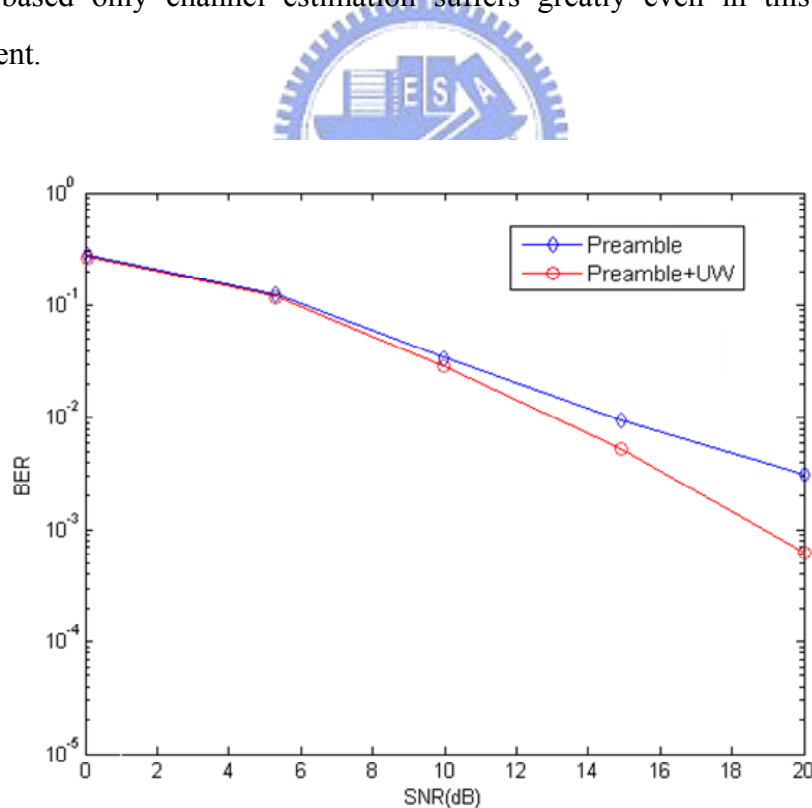


Figure 5.4: Comparison of the proposed preamble based channel estimation and UW-update channel estimation

5.4 Circuit Design of Proposed Methods

In this part, the circuit designs of the algorithms presented in previous section are proposed. Our purpose is to show that the UW-based synchronization and channel estimation algorithms are not only able to provide better performance but also have low complexity and are suitable for hardware designing. Besides, The design principles still follow the rules mentioned in section 4.3.1.

(1) UW-based Synchronizer

The design of the UW-based synchronizer is quite simple. As shown in Figure 5.5, the synchronizer block called **UW_sync** is roughly divided into three parts. The first part is a shift register, **80_DFF**, is used to store the latest 80 symbols input and its components can be implied by the name – 80 DFFs. The newest 16 symbols and the oldest 16 symbols, which are stored in DFF 65 through 80 and DFF 1 through 16 as indicated in Figure 5.5, perform the conjugate multiplications and summation. Once the result of the summation is large than certain threshold value, that is, two consecutive UWs are matched, it can be inferred that these two UWs are currently stored in DFF 1 through 16 and DFF 65 through 80. Thus, one frame of symbols - symbols in DFF 1 through DFF 64 then pass the **FFT_switch** and be sent to next stage.

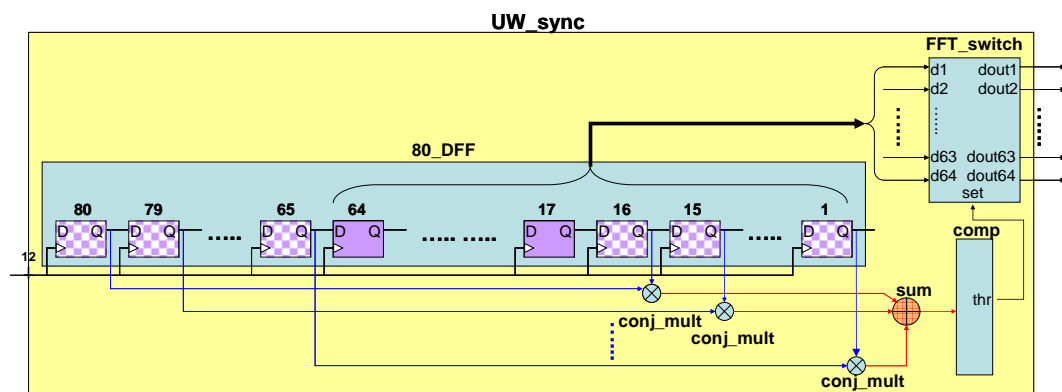


Figure 5.5: Circuit design of the UW-based synchronizer

(2) UW-based Channel Estimator

From Eq.(5.12) through (5.14) we can see that the proposed algorithm gives a conspicuous structure of parallelism, since Eq.(5.13) and (5.14) are in identical operation format. Besides, coupled with the preamble structure inside one packet, the initial channel estimate $\hat{\mathbf{H}}(0)$ is automatically available, and the initial values of $\mathbf{r}(0)$ and $\mathbf{P}(0)$ in Eq.(5.15) and (5.16) can be derived. Therefore, we can see our design in Figure 5.6 that two sets identical entities are implemented, except that the upper one is for calculating $\mathbf{P}(i)$ and the lower one is for $\mathbf{r}(i)$. Since $K=64$ in our system, two buffers **reg_p** and **reg_r** with depth 64 are used to store the value of $\mathbf{P}(i)$ and $\mathbf{r}(i)$, and every time the received $\mathbf{Y}(i)$ is inputted to this block, values of $\mathbf{P}(i)$ and $\mathbf{r}(i)$ will be updated according to Eq.(5.13) and (5.14), and then written back to corresponding buffers.

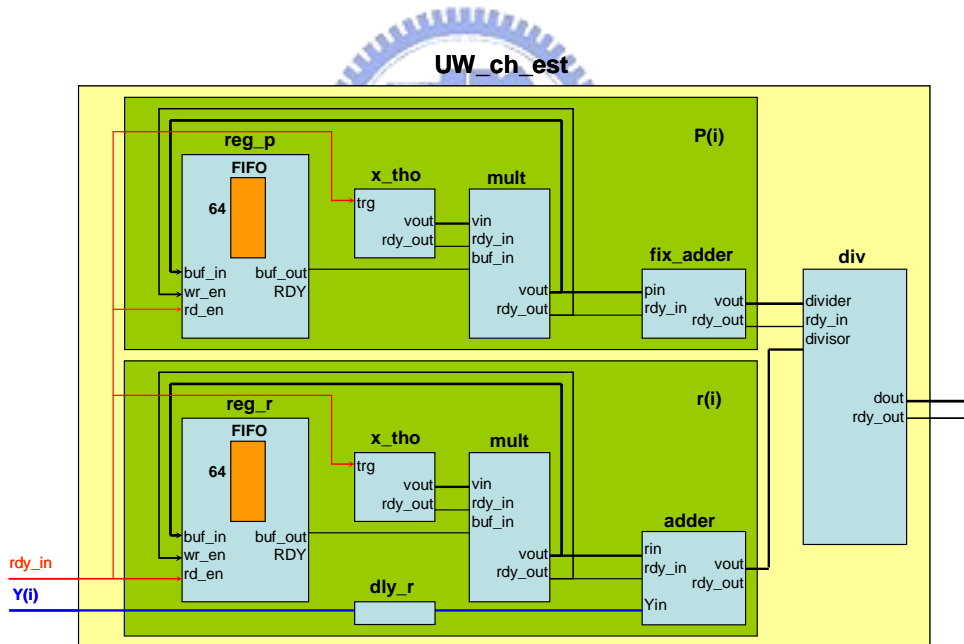


Figure 5.6: Circuit design of the UW-based channel estimator

5.5 Summary

In this section, we presented two applications of UW other than phase tracking mentioned in Chapter 2 and 4. We show that the structure of UW is not only useful in stationary system, but also useful in the tasks such as synchronization and channel estimation in a mobile environment. The synchronization algorithm employs UW as a selected training sequence while the channel estimation algorithm utilizes the constant nature of the UW extension in UW-based SC systems to obtain a moving average of the received signal over a finite period of time, which achieve a better performance than traditional preamble-only channel estimation. We also show that the two algorithms are not only theoretically uncomplicated, but also practically simple and therefore appropriate for hardware implementation.



Chapter 6

Conclusion

In future wireless communication systems, the demand of higher throughput and higher link quality is urgently called for, since various multimedia or home applications will be provided and thus reliable and affordable technologies are required to realize those contents. SC modulation, coupled with linear frequency domain equalization at the receiver, has less sensitivity to transmitter nonlinearities and phase noise than OFDM, and its complexity and performance are similar to those of OFDM in wireless communication. Single carrier with frequency domain equalization has been adopted by IEEE 802.16 standard to be one of the three modes as an alternative technique of OFDM in physical layer, and it is also currently a working assumption for uplink multiple access scheme in 3GPP Long Term Evolution, or Evolved UTRA. This reveals the potential of the SC-FDE technique and therefore encourages us to build up a hardware system based on SC-FDE system instead of the theoretical analysis only.

This thesis had described the signal processing concepts and algorithms of a SC-FDE system based on the UW structure in physical layer, and a self-designed platform equipped with four FPGA modules, USB interface, and RF modules is adopted to implement our system. A real wireless communication environment containing RF mismatch, multipath effects and so on are thus generated through real indoor experimental environment and RF modules on the self-designed platform. In our thesis we especially focus on the structure of the Unique-Word, and show that it is applicable to SC-FDE system in tasks such as phase tracking, synchronization and

channel estimation. What is more, corresponding circuit designs and analyses of the proposed UW-algorithms are presented; we show that those proposed UW-based algorithms are not only theoretically suitable for the SC-FDE system but also practically applicable in hardware implementation.

To summarize, hardware implementation is highly complicated. Therefore, the availability of MATLAB simulation, proper quantization algorithms, useful HDL simulation software, and powerful debugging tools becomes especially significant. Nevertheless, some future works still remain. For example, higher modulation order such as 16QAM, 64 QAM and so on can be realized; in addition, the UW-based synchronizer and channel estimator circuit structures provided in Chapter 5 can be implemented and coupled with our system to further raise the performance. Finally, although there is a lot of room for improvement, we believe that the SC-FDE system implemented on the FPGA-based platform we proposed is still highly advanced nowadays.



Bibliography

- [1] J. L. J. Cimini, "Analysis and simulation of a digital mobile channel using orthogonal frequency division multiplexing," *IEEE Trans. Commun.*, vol. 33, no. 7, pp. 665–675, Jul. 1985.
- [2] L. Deneire, P. Vandenameele, L. van der Perre, B. Gyselinckx, and M. Engels, "A low-complexity ML channel estimator for OFDM," *IEEE Trans. Commun.*, vol. 51, no. 2, pp. 135–140, Feb. 2003.
- [3] T. Pollet, M. V. Bladel, and M. Moeneclaey, "BER sensitivity of OFDM systems to carrier frequency offset and wiener phase noise," *IEEE Trans. Commun.*, vol. 43, pp. 191–193, Feb.–Apr. 1995.
- [4] O. Rousseaux, G. Leus, and M. Moonen, "A sub-optimal iterative method for maximum likelihood sequence estimation in a multipath context," *EURASIP JASP*, vol. 2002, no. 12, pp. 1437–1447, Dec. 2002.
- [5] H. Sari, G. Karam, and I. Jeanclaude, "Frequency-Domain Equalization of Mobile Radio and Terrestrial Broadcast Channels," *Proc. of the IEEE Global Telecommun. Conference*, Dec. 1994, pp. 1–5.
- [6] A. Czylik, "Comparison between adaptive OFDM and single carrier modulation with frequency domain equalization," in *Proc. IEEE 47th Vehicular Technology Conf.*, vol. 2, 1997, pp. 865–869.
- [7] D. Falconer, S. L. Ariyavisitakul, A. Benyamin-Seeyar, and B. Eidson, "Frequency domain equalization for single-carrier broadband wireless systems," *IEEE Commun. Mag.*, vol. 40, pp. 58–66, Apr. 2002.
- [8] M. V. Clark, "Adaptive Frequency-Domain Equalization and Diversity Combining for Broadband Wireless Communications," *IEEE JSAC* vol. 16, no. 8, Oct. 1998, pp. 1385–95.
- [9] D. Mansour and A. H. Gray, "Unconstrained Frequency-Domain Adaptive Filter," *IEEE Trans. on Acoustics, Speech, and Signal Processing*, vol.30, no.5, Oct. 1982.
- [10] N. Benvenuto and S. Tomasin, "On the comparison between OFDM and single carrier modulation with a DFE using a frequency-domain feedforward filter," *IEEE Trans. Commun.*, vol. 50, pp. 947–955, June 2002.

- [11] H. Witschnig, T. Mayer, A. Springer, A. Koppler, L. Maurer, M. Huemer, and R. Weigel, "A different look on cyclic prefix for SC/FDE," in *Proc. of the IEEE International Symposium on Personal, Indoor and Mobile Radio Commun.*, vol. 2, Sept. 2002, pp. 824–828.
- [12] J. H. Jang, H. C. Won, and G. H. Im, "Cyclic prefixed single carrier transmissions with SFBC over mobile wireless channels," *IEEE Signal Processing Lett.*, vol. 13, no. 5, pp. 261-264, May 2006.
- [13] A. Czylik, "Low overhead pilot-aided synchronization for single carrier modulation with frequency domain equalization," in *Proc. of the IEEE Global Telecommun. Conference*, vol. 4, 1998, pp. 2068–73.
- [14] M. Huemer, H. Witschnig, and J. Hausner, "Unique word based phase tracking algorithms for SC/FDE-systems," in *Proc. of the IEEE Global Telecommun. Conference*, vol. 1, Dec. 2003, pp. 70–74.
- [15] L. Deneire, B. Gyselinckx, and M. Engels, "Training sequence versus cyclic prefix—a new look on single carrier communication," *IEEE Commun. Lett.*, vol. 5, no. 7, pp. 292–294, July 2001.
- [16] R. Cendrillon and M. Moonen, "Efficient equalizers for single and multi-carrier environments with known symbol padding," in *Proc. of the Sixth International Symposium on Signal Processing and its Applications*, Aug. 2001.
- [17] J. A. Gansman, M. P. Fitz, and J. V. Krogmeier, "Optimum and suboptimum frame synchronization for pilot-symbol-assisted modulation," *IEEE Trans. Commun.*, vol. 45, pp. 1327–1337, Oct. 1997.
- [18] G. Leus and M. Moonen, "Semi-blind channel estimation for block transmissions with non-zero padding," in *Proc. of the Asilomar Conference on Signals, Systems and Computers*, Nov. 2001, pp. 762–766.
- [19] H. Sari, "Channel equalization and carrier synchronization issues in multicarrier transmission," in *IEEE Synchronization Workshop*, Belgium, pp. 29–36.
- [20] J. Coon, M. Beach, J. McGeehan and M. Sandell, "Channel and Noise Variance Estimation and Tracking Algorithms for Unique-Word Based Single-Carrier Systems," *IEEE Trans. Wireless Commun.*, vol. 5, no. 6, pp. 1488- 1496, Jun. 2006
- [21] H. Meyr, M. Moeneclaey, and S. A. Fletchell, *Digital Communication Receivers: Synchronization, Channel Estimation and Signal Processing*: Wiley, 1997.

- [22] Y. J. Jung et al., "A Dual-Loop Delay-Locked Loop Using Multiple Voltage-Controlled Delay Lines," *IEEE JSSC*, vol.36, no.5, pp. 784-791, May. 2001.
- [23] H. H. Chang et al., "A Wide-Range Delay-Locked Loop With a Fixed Latency of One Clock Cycle," *IEEE JSSC*, vol.37, no.8, pp. 1021-1027, Aug. 2002.
- [24] J. G. Maneatis, "Low-Jitter Process-Independent DLL and PLL Based on Self-Biased Techniques," *IEEE JSSC*, vol. 31, pp. 1723-1732, Nov. 1996.
- [25] J. V. de Beek, M. Sandell, and P. Börjesson, "ML estimation of time and frequency offset in OFDM systems," *IEEE Trans. Signal Processing*, vol. 45, pp. 1800–1805, July 1997.
- [26] S. Haykin, *Adaptive Filter Theory*, 3rd ed. Prentice Hall, 1996.
- [27] J. G. Proakis, *Digital Communications*, 4th ed. New York: McGraw-Hill, 2000.
- [28] J. Bhasker, *A VHDL Primer*, Englewood Cliffs, NJ: Prentice-Hall, 1998.
- [29] 林傳生，*使用VHDL 電路設計語言之數位電路設計*，儒林，2000.
- [30] 國家晶片系統設計中心，*VHDL*，July 2004.
- [31] 鄭信源，*Verilog 硬體描述語言數位電路設計實務*，儒林，2000.
- [32] 國家晶片系統設計中心，*Xilinx (PC)*，July 2004.