University of Wollongong
Research Online

University of Wollongong Thesis Collection

University of Wollongong Thesis Collections

2013

Implementation of OFDM systems using GNU Radio and USRP

Duc Toan Nguyen
University of Wollongong

Recommended Citation

Nguyen, Duc Toan, Implementation of OFDM systems using GNU Radio and USRP, Master of Engineering - Research thesis, School of Electrical, Computer and Telecommunications Engineering, University of Wollongong, 2013. http://ro.uow.edu.au/theses/3977

Research Online is the open access institutional repository for the University of Wollongong. For further information contact the UOW Library: research-pubs@uow.edu.au

UNIVERSITY OF WOLLONGONG

COPYRIGHT WARNING

You may print or download ONE copy of this document for the purpose of your own research or study. The University does not authorise you to copy, communicate or otherwise make available electronically to any other person any copyright material contained on this site. You are reminded of the following:

Copyright owners are entitled to take legal action against persons who infringe their copyright. A reproduction of material that is protected by copyright may be a copyright infringement. A court may impose penalties and award damages in relation to offences and infringements relating to copyright material. Higher penalties may apply, and higher damages may be awarded, for offences and infringements involving the conversion of material into digital or electronic form.

UNIVERSITY OF WOLLONGONG

School of Electrical, Computer and Telecommunication Engineering

"Implementation of OFDM systems using GNU Radio and USRP"

Duc Toan Nguyen

This thesis is presented as part of the requirements for the award of the Degree of Master by Research - Engineering from University of Wollongong

August 2013

ABSTRACT

As communications technology continues its rapid transition from analogue to digital, more functions of modern radio systems are implemented in software, rather than hardware, leading toward the software defined radio architecture. Software defined radio (SDR) is a promising technique for multi-type, high speed wireless communication system with low requirements on capability of hardware devices, low development cost and facilitation of the development process. SDR is an emerging and developing technique, thus it has not been matured yet.

In this project, a SDR testbed has been built to evaluate the practical error performance of Orthogonal Frequency Division Multiplexing (OFDM) based systems in different propagation conditions with different system configurations. In particular, the testbed has been built based on the GNU Radio Software platform and version-2 Universal Software Radio Peripheral (USRP2) devices.

Theoretical simulation results show that OFDM provides good error performance and high data rate transmission, compared to single carrier transmission techniques. However, the implementation of OFDM with SDR in actual hardware, as well as the verification of its performance in various realistic propagation conditions with different system configurations, has been almost unexplored. Evaluation of practical performance of OFDM systems is very important from practical points of view, including but not being limited to, the deep understanding of the effects of synchronisation and channel estimation techniques to the error performance and the awareness of what the realistic limitations of hardware and SDR are. Therefore, this project aims at evaluating the error performance of practical OFDM systems in various experimental environments, including typical laboratory rooms and corridors. The performance of OFDM with different digital modulation schemes is also evaluated in this project. The error performances of the implementation are then compared to the theoretical performances to confirm the effects of synchronisation and channel estimation processes and the limitations of hardware and SDR.

Outcomes of this project include the testbed which has been successfully built with SDR in the USRP2 hardware modules, a detailed technical documentation on how an OFDM system can be implemented with SDR and USRP2, a family of error performance curves of the implemented OFDM system with different modulation schemes in different propagation environments, in depth analyses of synchronisation and channel estimation processes and their effects to the experimental performances, and analyses of the realistic limitations of SDR and USRP2. These outcomes are significant, bearing in mind that SDR is an emerging-but-yet-mature technique.

The successful development of the OFDM testbed with SDR and USRP2 opens various opportunities to research further other advanced signal processing techniques for OFDM-based systems, such as the improved synchronisation and channel estimation techniques, the techniques to reduce the well known Peak-to-Average Power Ratio (PAPR) issue, and the emerging Multiple-Input Multiple-Output (MIMO) technique. These research directions will be our future works.

ACKNOWNLEDGEMENTS

First and foremost I would like to thank my principal supervisor, Dr. Le Chung Tran, for introducing me to the very interesting world of wireless networks, for encouraging me to pursue my research topic and for his guidance over the course of my time at the University. Special thanks also go to my co-supervisor Prof. Farzad Safaei for his support, guidance and knowledge. Without their illuminating instructions and continuous supports, this thesis could not have reached its present form.

I am also greatly indebted to all officers at the University of Wollongong, especially the school of Electrical, Computer and Telecommunications Engineering and SMART building, for their technical support.

I would additionally like to acknowledge my friend, especially, Linh Nguyen for his help to set up the system and collect experimental data.

Finally, my sincere thanks would go to my beloved family who lend me strong encouragement and full confidence when I felt frustrated during the research process, and to all my friends who gave me considerable assistance during the difficult moments of this thesis.

TABLE OF CONTENTS

ABSTRACT ..... **i**
ACKNOWNLEDGEMENTS ..... **iii**
TABLE OF CONTENTS ..... **iv**
LIST OF FIGURES ..... **vii**
LIST OF TABLES ..... **ix**
ABBREVIATIONS ..... **x**
1 Introduction **1**
1.1 Research introduction 1
1.2 Project significance 2
1.3 Research approach 3
1.4 Research outcomes 4
1.5 Publication 5
1.6 Thesis structure 5
2 Literature review **6**
2.1 Wireless communication channels 6
2.2 Orthogonal frequency division multiplexing 9
2.3 Synchronisation in OFDM systems 11
2.3.1 Time and frequency synchronisation effects 13
2.3.2 Synchronisation algorithms 14
2.4 Channel estimation in OFDM systems 18
2.4.1 Pilot-aided channel estimation 20
2.4.2 Training symbol-based channel estimation 22
2.5 Signal-to-noise ratio estimation 23
2.6 Software defined radio 25
2.7 GNU radio 28
2.7.1 GNU Architecture 29
2.7.2 GNU Radio Companion 32
2.8 Universal software radio peripheral 33
2.8.1 General architecture of USRP motherboards 33
2.8.2 Version-2 USRP 34
2.8.3 Daughter boards 36

2.9 Research questions 37
2.10 Contributions 38
2.11 Chapter summary 38
3 Channel and SNR estimation technique for OFDM systems 39
3.1 OFDM with perfect channel state information 39
3.2 Channel estimation in OFDM 42
3.3 SNR estimation 45
3.4 Simulation of OFDM systems with estimated channel state information . 48
3.5 Chapter summary 49
4 Implementation of OFDM systems with USRP2 and GNU Radio 50
4.1 Objectives, resources and limitations 50
4.2 GNU Radio's OFDM modules 51
4.3 Implementation of the OFDM modulator 54
4.4 Implementation of OFDM demodulator 56
4.5 Synchronisation 59
4.5.1 Time synchronisation and fine frequency offset with pseudorandom noise synchronisation 59
4.5.2 Coarse carrier frequency synchronisation 61
4.6 Implementation of OFDM systems with GNU Radio platform 62
4.6.1 Structure of a GNU Radio module 62
4.6.2 Block coding structure of a GNU Radio module 63
4.6.3 Rebuilding, reinstalling and debugging the modified block 65
4.6.4 Implementation of channel estimation 66
4.6.5 Implementation of SNR estimation 68
4.7 Chapter summary 71
5 Experiments 72
5.1 Test-bed settings 72
5.2 The OFDM performance in different configurations 74
5.2.1 OFDM performances in laboratory without obstruction 74
5.2.2 OFDM performances in laboratory environment with obstruction 77
5.2.3 OFDM performances in corridor environment without obstruction 80
5.2.4 OFDM performances in corridor environment with obstruction 82
5.2.5 Performance with obstruction versus performance without obstruction 85

5.3 Comparison of the simulated OFDM performances and the implemented OFDM performances 86
5.3.1 Simulated OFDM performances versus implemented OFDM performances in laboratory environment without obstruction 88
5.3.2 Simulated OFDM performances versus implemented OFDM performances in laboratory environment with obstruction 89
5.3.3 Simulated OFDM performances versus implemented OFDM performances in corridor environment without obstruction 91
5.3.4 Simulated OFDM performances versus implemented OFDM performances in corridor environment with obstruction 92
5.4 Chapter summary 93
6 Conclusions 94
6.1 Research contributions 94
6.2 Recommendations 95
6.3 Future works 96
6.4 Conclusions 96
REFERENCES 98

LIST OF FIGURES

Figure 2-1 Block diagram of the conventional OFDM system 10
Figure 2-2 Block diagram of a fundamental OFDM receiver 12
Figure 2-3 An example to create training symbols 16
Figure 2-4 Channel estimation in OFDM receiver 19
Figure 2-5 Typical training symbols and pilot subcarriers arrangement[40] 20
Figure 2-6 Software defined radio architecture 26
Figure 2-7 An example of flowgraph 30
Figure 2-8 GNU Radio Blocks 31
Figure 2-9 An example of a hierarchical block 32
Figure 2-10 FM receiver built by GRC 33
Figure 2-11 Decimating low pass filter function in FPGA [70] 34
Figure 2-12 Block diagram of USRP2 main components and signalling rates [83] .. 35
Figure 2-13 WBX daughterboard 36
Figure 3-1 OFDM system block diagram 40
Figure 3-2 Performance of OFDM systems in different propagation environments . 42
Figure 3-3 OFDM frame structure 43
Figure 3-4 Block diagram of OFDM systems with channel estimation 44
Figure 3-5 Comparison of LS and MMSE estimations in OFDM systems 45
Figure 3-6 Block diagram of OFDM systems with SNR estimation 46
Figure 3-7 Performance of blind SNR estimation in OFDM system 48
Figure 3-8 OFDM block diagram with estimated channel state information 49
Figure 4-1 OFDM system's module hierarchy 52
Figure 4-2 Block diagram of *ofdm_receiver* module 57
Figure 4-3 Block diagram of *frame_sink* module 59
Figure 4-4 PN synchronisation flow graph 60
Figure 5-1 Structure overview of the testbed 73
Figure 5-2 Laboratory environment without obstruction and the arrangement of transmitter and receiver in our experiments 74
Figure 5-3 Comparison OFDM performance with BPSK and QPSK constellations in laboratory environment without obstruction 76

Figure 5-4 Comparison performance of OFDM system with BPSK and QPSK in simulation 77
Figure 5-5 The laboratory environment with obstruction 77
Figure 5-6 Performance of OFDM systems with BPSK and QPSK constellations in the laboratory environment with obstruction between two antennas 78
Figure 5-7 Performance of OFDM system with BPSK versus QPSK in simulation . 79
Figure 5-8 Corridor environment used in the experiments 80
Figure 5-9 Comparison between performances of implemented OFDM with BPSK and QPSK in corridor environment without obstruction 81
Figure 5-10 Comparison performances of OFDM with BPSK and QPSK in simulation 82
Figure 5-11 Obstruction between two antennas in corridor environment 83
Figure 5-12 Performances of OFDM systems with BPSK and QPSK constellations in corridor environment with obstruction 84
Figure 5-13 Comparison performances of OFDM with BPSK and QPSK in simulation 84
Figure 5-14 Comparison between performances of the implemented OFDM system in our laboratory with obstruction and without obstruction 85
Figure 5-15 Performances of the implemented OFDM system in the corridor environment with and without obstruction 86
Figure 5-16 Simulated OFDM vs. implemented OFDM in laboratory environment without obstruction 88
Figure 5-17 Simulated OFDM vs. implemented OFDM in laboratory environment with obstruction 90
Figure 5-18 Simulated OFDM vs. implemented OFDM in corridor environment without obstruction 91
Figure 5-19 Simulated OFDM vs. implemented OFDM in corridor environment with obstruction 92

LIST OF TABLES

Table 2-1 USRP1 and USRP2 Comparison 35

ABBREVIATIONS

ADC	Analogue-to-Digital Converter
AGC	Automatic Gain Control
AMC	Adaptive Modulation and Coding
API	Application Programming Interface
AWGN	Addictive White Gaussian Noise
BER	Bit Error Rate
BPSK	Binary Phase Shift Keying
BRANs	Broadband Radio Access Networks
CFO	Carrier Frequency Offset
CIR	Channel Impulse Response
CPU	Central Processing Unit
CRC	Cyclic Redundancy Check
DAB	Digital Audio Broadcasting
DVB-T	Digital Video Broadcasting for terrestrial television
DAC	Digital-to-Analogue Converter
DECT	Digital Enhanced Cordless Telecommunications
FER	Frame Error Rate
FIFO	First In First Out
FIR	Finite Impulse Response
FPGA	Field-Programmable Gate Arrays
GI	Guard Interval
GNU	GNU's Not Unix
GPL	General Public License
GPP	General Purpose Processors
GRC	GNU Radio Companion

GSM	Global System for Mobile communications
HIPERLAN-II	High Performance Local Area Network type 2
IDFT	Inverse Discrete Fourier Transform
IF	Intermediate Frequency
IIR	Infinite Impulse Response
IO	Input and Output
ISI	Inter-Symbol Interference
LANs	Local Area Networks
LO	Local Oscillator
LOS	Line-of-Sight
LS	Least Square
MANET	Mobile Ad-Hoc Networks
MCM	Multi-Carrier Modulation
MIMO	Multiple-Input Multiple-Output
ML	Maximum-Likelihood
MMSE	Minimum Mean-Square Error
MSE	Mean Squared Error
MSPS	Mega Samples Per Second
NC-OFDM	Non-Contiguous OFDM
NNR	Noise-to-Noise ratio
OFDM	Orthogonal Frequency Division Multiplexing
PAPR	Peak-to-Average Power Ratio
PC	Personal Computer
PCI	Peripheral Component Interconnect
PN	Pseudo Noise
PSK	Phase Shift Keying
QA	Quality Assurance

QAM	Quadrature Amplitude Modulation
QPSK	Quadrature Phase Shift Keying
RF	Radio Frequency
SC-FDMA	Single Carrier Frequency Division Multiple Access
SD	Secure Digital
SDR	Software Defined Radio
SNR	Signal-to-Noise Ratio
UDP	User Datagram Protocol
UHD	USRP Hardware Driver
USRP	Universal Software Radio Peripheral
USRP2	Version-2 Universal Software Radio Peripheral
WLANs	Wireless Local Area Networks

1 INTRODUCTION

1.1 Research introduction

Wireless communication is one of the most exciting areas in the communication field today. It is used to transmit information over the air without the use of a guided medium. Due to the sharply increasing in demand for wireless connectivity, it has developed extremely fast during the last two decades. However, wireless communication is not as reliable as guided medium communication, due to fading and other propagation effects [1]. Accordingly, the techniques to improve its capacity and reliability become the essential objectives for current research. Numerous techniques have been proposed to enhance the performance and reduce the interference in wireless communication. A common technique is Orthogonal Frequency Division Multiplexing (OFDM). The aim of OFDM technique is to against frequency-selective fading or narrowband interference in wireless communication, consequently improving the system reliability. This technique allows data to be transmitted in parallel by modulating the data on a set of orthogonal sub-carriers. It has been widely applied in practice, such as in Digital Audio Broadcasting and Digital Video Broadcasting for terrestrial television (DAB/DVB-T), Wireless Local Area Networks (WLANs) such as IEEE802.11a /g/n, Broadband Radio Access Networks (BRANs), WiMax and High Performance Local Area Network type 2 (HIPERLAN-II) [2, 3].

To enhance wireless communication performance numerous techniques have been proposed and validated in simulations. However, validation of those techniques in actual hardware is much more challenging because implementation of a system in hardware takes a large amount of time, effort and with a high cost. Currently, the Software Defined Radio (SDR) approach could help researchers and engineers to reduce the time, effort and cost to implement a wireless system. Therefore, it has become more and more widespread in the formation of wireless networks. The main idea of SDR is to turn radio hardware problems into software problems. Most signal processing in wireless systems thus can be performed via software. This character of SDR system offers great flexibility for wireless communication researchers and developer to implement new protocols, methods or techniques. In SDR-related research field, opened GNU Radio software platform (GNU stands for GNU's Not

Unix [4]) and opened Universal Software Radio Peripheral (USRP) hardware are the most common software and hardware used to in SDR systems. Since they are open source, everyone can easily build, develop and modify their system.

The above advantage of SDR techniques motivates us to implement a wireless communication system based on SDR techniques using GNU Radio software platform and USRP hardware. Particularly, an OFDM-based system is implemented based on GNU Radio software platform and version-2 USRP (USRP2) devices. The performance of the implemented system is then analysed in different realistic environments with different signal constellations. Finally, the performances of the implemented OFDM system are compared to the performances of the simulated OFDM system.

1.2 Project significance

In this project, an OFDM testbed has been successfully developed using GNU Radio platform and USRP2 devices. The physical experimental platform differentiates this project from many preceding research projects which merely focus on simulated results. The successful testbed could be easily extended to verify performance of a different general wireless communication system, not just limited to an OFDM-based system, in different realistic propagation environments. Further, the project details the GNU Radio platform structure and the way to implement or modify a signal processing module. This contribution makes up for the current less informative Application Programming Interface (API) of GNU Radio software platform.

Although this project only implements an OFDM system with essential modules, it provides an approach to construct a SDR system using GNU Radio software platform and USRP devices. Based on this approach, new methods, new techniques, and new algorithms could be applied to enhance further the OFDM system, and the OFDM performance can be evaluated in different realistic environments. The successful implementation of OFDM systems in SDR using GNU Radio software platform and USRP devices opens the opportunity to implement other wireless communication system in SDR and verify easily the performance of the system in practical environments as well as to apply new techniques to enhance the performance of the system with less effort and lower cost.

In summary, the outcomes of this project have considerable contributions to the software defined radio research field, which is quickly developing, but not yet mature.

1.3 Research approach

This project focuses on the practical performance analysis of an OFDM-based system in different wireless transmission environments and with different system configurations. For the purpose, a testbed using GNU Radio Software and USRP2 has been built and tested. Comparisons of the practical performance with the theoretical performances are also carried out in this project. Detailed strategies to achieve the project goals are listed below.

- Provide a comprehensive literature review of OFDM systems, SDR techniques and the built-in example of an OFDM system;
- Simulate OFDM systems with channel estimation and Signal-to-Noise Ratio (SNR) estimation techniques in MATLAB for different propagation environments;
- Analyse the built-in example of an OFDM system given in GNU Radio;
- Find out how to modify, create and install a module in GNU Radio;
- Find out how to work with USRP devices;
- Implement an OFDM system with channel estimation and SNR estimation techniques based on GNU Radio software platform and USRP devices;
- Evaluate the performances of the implemented OFDM system in different environments with different configuration;
- Compare the performances of the simulated and implemented OFDM systems;
- Derive a few hands-on recommendations and suggestions for the implementation of OFDM-based systems using GNU Radio platform and USRP2.

1.4 Research outcomes

Outcomes of this project include

- A comprehensive literature review for two important techniques, namely OFDM and SDR with GNU Radio platform and USRP devices;
- An overview of the current channel estimation, SNR estimation, and synchronisation techniques which are directly related to the implementation of OFDM systems in hardware;
- Simulated performances of OFDM systems in MATLAB with channel estimation and SNR estimation techniques, and with different modulation schemes. These simulated performances will be used as benchmark performances to evaluate the implemented OFDM system in different propagation environments;
- An OFDM testbed (with channel estimation and SNR estimation techniques) developed based on GNU Radio software platform and USRP2 hardware. The SDR implementation, including the necessary modifications and new developments, has been documented in the thesis;
- A newly developed blind SNR estimation module. This module facilitates the evaluation of the BER performance of OFDM systems in different realistic environments;
- Intensive comparisons and analyses between the performances of the implemented OFDM system in different propagation environments with different digital modulation constellations and the simulated performances. These comparisons serve three purposes. First, they are the proof-of-concept for the developed testbed. Second, the comparisons allow us to understand the gaps between the simulated and implemented performances. Finally, they also point out the realistic limitations of SDR and hardware;
- Detailed documentation on how to develop an OFDM-based testbed using GNU Radio software platform and USRP devices;
- Helpful recommendations and suggestions on the implementation of an OFDM-based testbed using GNU Radio software platform and USRP devices.

1.5 Publication

The following paper is prepared for submission to IEEE International Conference On Acoustic, Speech, and Signal Processing (ICASSP):

D. T. Nguyen, L. C. Tran and F. Safaei, "A study of OFDM implementation in software defined radio with GNU Radio and USRP2", *to be submitted to the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP 2014)*, Florence, Italy, 4-9 May, 2014.

1.6 Thesis structure

The thesis is organised as follows:

- Chapter 1 introduces the background, significance, approach, and expected outcomes of the project.
- Chapter 2 provides a comprehensive literature review for this project. This chapter mainly focus on the review of SDR technique, OFDM technique, GNU Radio platform and USRP devices.
- Chapter 3 presents the simulated OFDM system on MATLAB
- Chapter 4 describes an example of the implemented OFDM-based system and the approach to implement OFDM systems using GNU Radio software platform and USRP devices.
- Chapter 5 analyses the performances of OFDM-based system in different realistic environments as well as in different experimental settings in simulation
- Chapter 6 summaries all research contributions of this project, provides some useful recommendations, suggestions and the directions of my future works.

2 LITERATURE REVIEW

This chapter presents the background of the following techniques, which involve in this project, and the research questions which will be addressed in this project.

- Wireless communication channels
- Orthogonal Frequency Division Multiplexing (OFDM)
- Synchronization in OFDM systems
- Channel estimations in OFDM systems
- Signal-to-Noise Ratio (SNR) estimation in OFDM systems
- Software Defined Radio (SDR)
- Universal Software Radio Peripheral (USRP)
- GNU radio

2.1 Wireless communication channels

The performance and quality of wireless communication depend significantly on the type of channels between the transmitter and receiver. Therefore, it is necessary to discuss about wireless communication channels.An introduction of the behaviours of the wireless communication channel, its effects, and methods for battling these effects can be found in [1, 3, 5-7]. In wireless communication, there could be more than one path from the transmitter to the receiver. The multipath is due to scattering, reflections and refraction from building and other objects. The multipath effect is classified into two groups namely small-scale and large-scale effects (shadow) referring to the time, space, position, shape scale over. A large-scale phenomenon includes effects of path loss and shadowing; a small-scale phenomenon includes multipath fading. Shadow fading is an attenuation of average signal power caused by terrain contours (high hills or dense urban) between the transmitter antenna and the receiver antenna; it is described in terms of path loss.

The multipath fading is referred to the fact that the transmitted signal follows many different paths before arriving at the receiver. This effect constitutes a multipath radio propagation channel. The quality of received signal depends on some characteristic parameter of the multipath fading channel, namely:

- root mean squared (rms) delay spread - the time between the first and the last arriving signal paths;

- Doppler spread - the range of observed frequencies due to the movement of transmitter and receiver;
- channel coherence time - time scale at which channel realization is roughly independent;
- coherence distance - distance between transmitter antenna and receiver antenna;
- channel coherence bandwidth - the range of contiguous frequencies; and
- number of multipaths - the number of path ways from the transmitter to the receiver.

Rayleigh and Rician fading channels are currently channel models wisely used to model narrow-band and wideband wireless channels in a number of realistic wireless communication environments. In realistic wireless communication environments, some propagation phenomena, including multipath scattering effects, the time dispersion and Doppler shifts, are typically present.

Rayleigh fading channel [3, 7-9] is the rational model when there are many object in the environment that scatter and reflect the transmitted signal and there are no direct path (Line-Of-Sight) between the transmitter and receiver. If there is no Line-of-Sight (LOS) then the constructive and destructive nature of multipath signal in flat fading can be approximated by a Rayleigh distribution.

In Rayleigh channel, the received signal can be expressed as

$$s(t) = \sum_{i=1}^{N} \alpha_i \cos(\omega_c t + \phi_i) \tag{2-1}$$

where N is the number of paths, α_i and ϕ_i are amplitude and phase of the i^{th} reflected signal at the receiver from an angle ψ_i relative to the direction of motion of the received antenna. The phase ϕ_i and angle ψ_i can be assumed to follow a uniform distribution over $[0, 2\pi]$. The Doppler shift is given by

$$\omega_{d_i} = \frac{\omega_c \upsilon}{c} \cos\psi_i \tag{2-2}$$

where υ is the velocity of the mobile terminal, c is the speed of light.

Taking the Doppler shift into consideration, the received signal can be written as

$$s(t) = \sum_{i=1}^{N} \alpha_i \cos(\omega_c t + \omega_{d_i} t + \phi_i) \tag{2-3}$$

Expressing the signal in-phase and quadrature phase form, (2-3) can be written as

$$s(t) = I(t)\cos(\omega_c t) - Q(t)\sin(\omega_c t) \tag{2-4}$$

where:

$$I(t) = \sum_{i=1}^{N} \alpha_i \cos(\omega_{d_i} t + \phi_i) \tag{2-5}$$

$$Q(t) = \sum_{i=1}^{N} \alpha_i \sin(\omega_{d_i} t + \phi_i) \tag{2-6}$$

The amplitude R of the complex envelop is defined as

$$R = \sqrt{I(t)^2 + Q(t)^2} \tag{2-7}$$

The probability density function of the in-phase and quadrature phase components is given by the Gaussian density

$$f_Q(q) = \frac{1}{\sqrt{2\pi}\sigma} e^{-q^2/2\sigma^2} \tag{2-8}$$

where σ^2 is the variance (power) of the in-phase or quadrature phase component. Therefore, the probability density function of the envelop R is given by

$$f_R(r) = \frac{r}{\sigma^2} e^{-r^2/2\sigma^2} \tag{2-9}$$

It is known as the Rayleigh probability density function.

Rician channel [7, 8, 10] occurs when there is a LOS path between the transmitter and the receiver. The received signal comprises both the directional and scattered (or reflected) paths. In this case, the scattered (or reflected) paths tend to be weaker than the direct path. The received signal is written as

$$s(t) = \sum_{i=1}^{N} \alpha_i \cos(\omega_i t + \omega_{d_i} t + \phi_i) + k_d \cos(\omega_c t + \omega_d t) \tag{2-10}$$

where constant k_d is the strength of the direct component, ω_d is the Doppler shift along the LOS path, and ω_{d_i} is the Doppler shift along the $i-th$ indirect path. The complex envelop in this case is given by

$$f_R(r) = \frac{r}{\sigma^2} e^{-(r^2+k_d^2)/2\sigma^2} I_0(\frac{ak_d}{\sigma^2}) \tag{2-11}$$

where $I_0(.)$ is the modified Bessel function of the first kind.

The Rician probability density function is characterized by the term of Rician factor K. It is defined as the ratio between deterministic signal power (from LOS path) and the diffuse signal power (from the scattered and reflected path)[7, 8, 10]. Rician factor K (dB) is given by

$$K(dB) = 10\log(\frac{k_d^2}{2\sigma^2}) \tag{2-12}$$

For $K = -\infty$, we have no direct path and the Rician distribution becomes the Rayleigh distribution. For $K = +\infty$, the Rician distribution turns into the Gaussian distribution.

2.2 Orthogonal frequency division multiplexing

Implementation of OFDM system using hardware and software-defined radio is the main focus of this thesis. Therefore, fundamental background of OFDM system is reviewed in this section.

OFDM is one of the special cases of the Multi-Carrier Modulation (MCM) scheme which modulates data symbols in parallel on orthogonal subcarriers [11]. The main idea of OFDM system is to separate the single high rate data stream into N parallel low data sub-streams that are modulated onto N orthogonal sub-carriers. This process is simply made in the discrete time domain through a N point Inverse Discrete Fourier Transform (IDFT) or Inverse Fast Fourier Transform (IFFT) element and the resulted signal is transmitted in sequence. The information at the receiver is retrieved by performing a DFT/FFT unit. The block diagram of a conventional OFDM system is shown in Figure 2-1.

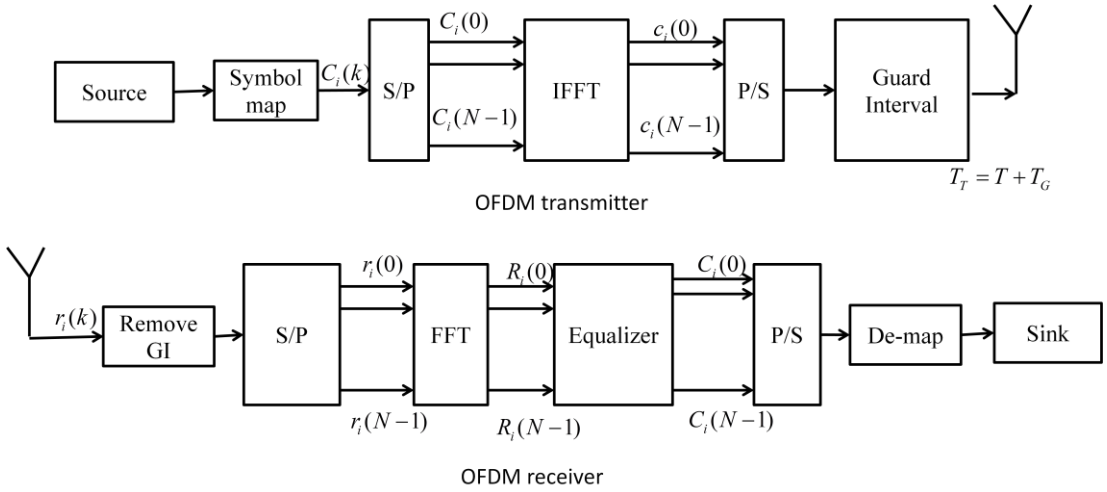

Figure 2-1 Block diagram of the conventional OFDM system

Briefly, the OFDM transmitter works as follows:

- Bit stream from the source is broken into groups of K bits ;
- Symbol mapping block maps each group of K bits to the corresponding complex symbol (the modulated symbol);
- Place these complex numbers into a vector;
- Compute Inverse Fast Fourier Transform (IFFT);
- Add a Guard Interval (GI) then feed the results to the up converter

OFDM receiver works as follows:

- Receive signals from the down converter, and remove GI
- Place the complex signal into vector
- Compute Fast Fourier Transform (FFT)
- The resulting signal will be fed to equalizers
- Then de-map to revert the original signal

The effect of Inter-Symbol Interference (ISI) in OFDM systems, caused by the channel delay spread is easily diminished. An OFDM transmitter is generally implemented by the Inverse Fast Fourier Transform (IFFT) with Cyclic Prefix (CP) insertion or Zero-Padded (ZP) suffix appending. At the receiver, a simple one-tap frequency domain channel equaliser, rather than a multi-tap time domain equaliser, is applied to each subcarrier after the Fast Fourier Transform (FFT) operation has been carried out. With these advantages, it has been widely applied in digital

communication systems nowadays, such as DAB/DVB-T systems, Wireless Local Area Networks (WLANs), Broadband Radio Access Networks (BRANs), and High Performance Local Area Network type 2 (HIPERLAN-II) [2, 3]. OFDM has been standardized as a part of IEEE802.11 and IEEE802.16 standard families for high bit rate data transmission over wireless Local Area Networks (LANs) [12-15] and Metropolitan Area Networks (MANs) respectively.

However, besides the advantages, OFDM also have several disadvantages. First significant problem is the associated increased Peak-to-Average Power Ratio (PAPR) compared to single-carrier systems [16]. Second problem is that capacity and power loss due to the guard interval (CP or ZP) can be significant, which can be 20% like in IEEE802.11a. Third problem is that frequency offsets and phase noise are sensitive [3]. Thus, OFDM system requires more studies to ensure that the received signal is not distorted, the bandwidth is used more efficiently and the PAPR is reduced. Currently, bandwidth efficiency with OFDM without CP [17-19] and Non-Contiguous OFDM (NC-OFDM) [20, 21], PAPR reduction techniques [22, 23], synchronisation, channel estimation, and diversity techniques [24-31] to improve OFDM-based systems are receiving intensive attention from researchers.

2.3 Synchronisation in OFDM systems

Since timing and frequency errors in multi-carriers ruin orthogonal property among subcarriers which causes large performance degradation, time and frequency synchronisations are significant to construct a wireless communication system. Basically, the target of the function is to obtain significant parameters such as start of the frame, sampling clock and frequency offset from the received signal for reliable transmission. The synchronisation tasks in an OFDM system can be identified as following[32]:

- sampling clock synchronisation: in practical systems, there is a slightly difference between the sampling clock frequency at the receiver and the corresponding transmitted frequency. Consequently, the Inter-Carrier Interference (ICI) issue is introduced at the receiver and degrades the system performance. Thus, sampling clock synchronisation is aimed to reduce this destruction to an endurable level.

- timing synchronisation: the target of the timing synchronisation is to detect the start of each received OFDM symbol or each OFDM frame (which is group of OFDM symbols) to identify the exact position of the DFT window.
- frequency synchronisation: the goal of this operation is to correct frequency error between the transmitter and receiver which causes a degradation of orthogonal property among subcarriers. This results in the degradation of the system performance. Therefore, frequency synchronisation is essential to restore the orthogonal property of OFDM symbol by compensating for any frequency offset.

The receiver block diagram is presented in Figure 2-2. In the analogue frontend, the Local Oscillator (LO) generates two quadrature sinusoids to filter and down-convert the incoming waveform $r(t)$ to baseband. An Analogue-to-Digital Converter (ADC) then is applied to sample the baseband signal at the sampling frequency. The frequency f_{LO} of the DFT does not match to the received carrier frequency f_c due to oscillator instabilities and/or Doppler shifts. The difference $f_d = f_c - f_{LO}$ causing a phase shift of $2\pi k f_d$ is known as frequency offset or Carrier Frequency Offset (CFO). Thus, the received signal in baseband can be written as

$$r(k) = y(k)e^{j2\pi\varepsilon k/N} \tag{2-13}$$

where $\varepsilon = Nf_dT_s$ is the carrier frequency offset normalised to the subcarrier spacing $\Delta f = 1/(NT_s)$.

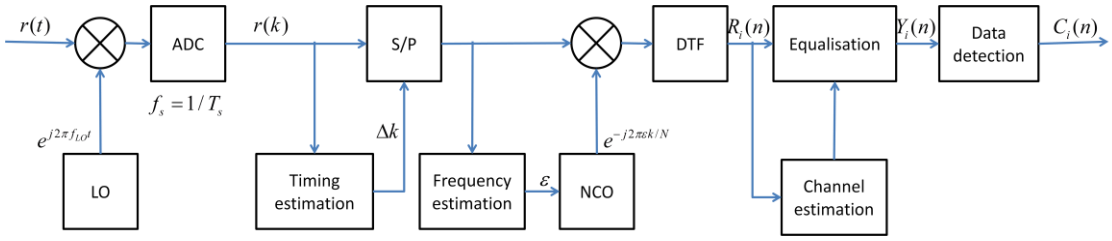

Figure 2-2 Block diagram of a fundamental OFDM receiver

Additionally, since the time scales at the transmitter is not absolutely aligned with the receiver time scales, at the receiver thus does not know the start position of the OFDM symbol. Therefore, the position of DFT window can be set in a wrong place. There is the fact that fractional timing errors do not lead to any degradation of the

system performance. Therefore, the start of each OFDM symbol at the receiver is sufficient to estimate within one sampling period. Denotes Δk to be the number of samples the time scale shifted compared to the perfect setting at the receiver. Thus, the samples from ADC are expressed by

$$r(k) = e^{j2\pi\varepsilon k/N} \sum_{i} \sum_{l=0}^{L-1} h(l) c_i (k - l - \Delta k - iN_T) + w(k) \tag{2-14}$$

where $h = [\mathrm{h}(0), \mathrm{h}(1), ..., \mathrm{h}(\mathrm{L}\text{-}1)]^T$ is the discrete time Channel Impulse Response (CIR), N_T is the number of symbols in transmitted signal, $w(k)$ is a complex-valued AWGN.

The timing and frequency synchronisation blocks in the OFDM system shown in Figure 2-2 utilise the received samples $r(k)$ to estimate ε and Δk. $r(k)$ is counter-rotated at an angular speed $2\pi\varepsilon k / N$, which is estimated frequency offset, by numerically controlled oscillator (NCO). Whereas the timing estimation block is aimed to estimate the beginning of the received signal. Specifically, the corrected samples $r(k)$ are passed to the DFT/FFT block and the result signal in the frequency domain is utilise to retrieve the data symbols transmitted by the $i-th$ OFDM symbol.

2.3.1 Time and frequency synchronisation effects

To estimate the effect of frequency offset on the performance of digital system, the loss in SNR value is used as a indicator [32]. It is defined as

$$\gamma(\varepsilon) = \frac{SNR_{ideal}}{SNR_{real}} \tag{2-15}$$

If the ideal timing synchronisation is corresponding to $\Delta k = 0$, the loss in SNR caused by the normalized frequency offset ε can be estimated by

$$\gamma(\varepsilon) = 1 + \frac{1}{3}\frac{E_s}{N_0}(\pi\varepsilon)^2 \tag{2-16}$$

where E_s is the average received subcarrier energy while $N_0 / 2$ is two-sided spectral density of the noise power.

Time synchronisation errors, unlike the frequency synchronisation errors, do not result in inter-subcarrier interference. Instead, if the receiver's FFT block spans samples from two consecutive OFDM symbols, ISI effect which causes a phase shift in the frequency domain for the transmitted symbol and BER degradation will occur.

If the frequency synchronisation is perfect, i.e... $\varepsilon = 0$ and consider in the case time offset is smaller than part of CP, the DFT output for the $i-th$ OFDM symbol is written as

$$R_i(n) = H_i(n)C_i(n)e^{-j2\pi\Delta km/N} + W_i(n) \tag{2-17}$$

where $W_i(n)$ is a addictive Gaussian distribution noise with variance σ_w^2 derived as

$$W_i(n) = \sum_{k=0}^{N-1} w(k)e^{-j2\pi kn/N} \qquad 0 \le n \le N-1 \tag{2-18}$$

$H_i(n)$ is the channel frequency response defined as a DFT/FFT of the channel impulse response given by

$$H_i(n) = \sum_{k=0}^{N-1} h(k)e^{-j2\pi kn/N} \quad 0 \le n \le N-1 \tag{2-19}$$

From (2-17), it can be easily seen that

- a fractional time offset after DFT/FFT only causes a linear phase rotation and can be eliminated by the channel equaliser;
- the timing offset does not ruin the orthogonal properties between subcarriers; and
- the effect of timing error is increasing a linear phase offset across the subcarriers.

2.3.2 Synchronisation algorithms

In an OFDM system, synchronisation algorithms can be classified into two categories [33, 34], namely

- non data-aided: make use of the correlation between the CP and the end of the symbol
- data-aided: make use of additional data such as preamble and pilot tones.

2.3.2.1 Non data-aided

In OFDM system, time and frequency offsets can be estimated by exploiting the redundancy created by the CP of OFDM symbol. This is most commonly done by averaging the correlation of CP and the end of OFDM symbol. In [35], Van de Beek's applied the joint maximum likelihood estimator to find time and frequency offset in OFDM system. The results show that the Beek's estimator could have a lower error variance when the number of CP sample is increased. In his paper, a correlation function is given by

$$\gamma(n) = \sum_{k=n}^{n+L-1} r(k) r^*(k+N) + E \tag{2-20}$$

$$E = \frac{1}{1+SNR^{-1}} \frac{1}{2} \sum_{k=n}^{n+L-1} |r(k)|^2 + |r(k+N)|^2 \tag{2-21}$$

where L is the CP length, N is the OFDM symbol length, and E is the normalised signal energy.

The timing offset firstly is estimated using the peak of $\gamma(n)$. Then the frequency offset can be calculated based on the phase shift between the CP and the samples at the end of the OFDM symbol. Intuitively, this algorithm works inefficiently in multipath channel because the CP is distorted by ISI. Besides the second-order cyclostationarity of the received signals is exploits, and then the information of symbol-timing offset and carrier frequency offset are obtained by the cyclic correlation [36].

Non data-aided time and frequency estimation algorithms archive better bandwidth efficiency, but worse performance than other time and frequency data-aided estimation algorithms because no training symbol is required. It is important to discuss data-aided estimation algorithms.

2.3.2.2 Data-aided

The accuracy of estimated time and frequency offset is significant to an OFDM system. Therefore, there is a trade-off between the system bandwidth efficiency and the system performance. In most cases, it is justified to lose some amount of bandwidth for synchronisation. The bandwidth can take the form of the preamble, some pilot tones in each OFDM symbol, or a combination of the two.

2.3.2.2.1 Preamble techniques

The most popular synchronisation technique using the preamble for OFDM systems was initiated by Schmidl and Cox [37], which has accuracy close to the Cramer-Rao bound. There have been numerous subsequent works based on this algorithm. Schmidl and Cox used a symbol preamble with special structure to estimate the timing and frequency offsets. They create the training symbol with two matching halves in time domain by in the frequency domain transmitting Pseudo-Noise (PN) sequence only on even subcarriers, and transmitting nothing on odd subcarriers, i.e., $X_{2k+1} = 0$ for $k = 0,..., N/2-1$. It can be seen in Figure 2-3.

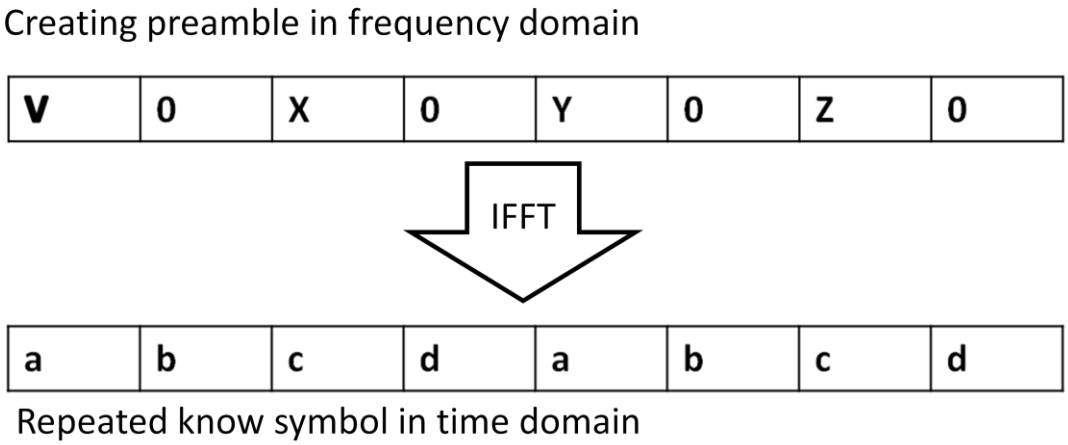

Figure 2-3 An example to create training symbols

The PN can be chosen so that it copes well with the high Peak to Average Power Ratio problem of OFDM systems. We have

$$\begin{aligned} c_i(n) &= \sum_{k=0}^{N-1} C_i(k) e^{-i2\pi kn/N} \\ &= \sum_{l=0}^{N/2-1} C_i(k) e^{-i2\pi 2ln/N} \\ &= c_i(n+N/2) \end{aligned} \tag{2-22}$$

The received samples in the time domain will be separated into two matched parts and will have approximately the same timing. Therefore if we multiply r_d, the $d-th$ received sample, with the conjugate of $r_{d+N/2}$ then the only factor affects the phase of the product is the carrier frequency offset. Since we are two matched half a symbol

in time domain, the phase difference $\phi = 2\pi\Delta f \frac{T}{2} = \pi\Delta f T$ with Δf is frequency offset, can be estimated by time metric as follows

$$P(d) = \sum_{k=0}^{N/2-1} r_i(k+d) r_i(k+d+N/2) \tag{2-23}$$

If d is the part of the CP that is unaffected by the ISI, or the start of OFDM symbol, $P(d)$ will be the sum of terms with the similar phase. By searching peak of $P(d)$, the optimal starting point of the OFDM symbol can be found. In [36], $P(d)$ is normalized by the energy of the second half of the received symbol

$$R(d) = \sum_{k=0}^{N/2-1} | r_i(d+k+N/2)^2 | \tag{2-24}$$

So the quantity to be maximized is $\frac{P(d)}{R(d)}$. Denote the optimal value of d to be d*

The phase difference is estimated as $\phi = \angle P(d^*)$. So the frequency offset is

$$\Delta f = \frac{\angle P(d^*)}{\pi T} \tag{2-25}$$

Since the phase difference $\angle P(d^*)$ is limited to $[-\pi, \pi]$ and the frequency offset is limited to less than the subcarrier spacing, Schmidl and Cox extended the result to the circumstance when the carrier frequency offset is larger than subcarrier spacing by introducing the term z

$$\Delta f = \frac{\angle P(d^*)}{\pi T} + \frac{2z}{\pi T} \tag{2-26}$$

z can be found by maximizing

$$B(g) = \frac{| \sum_{k=0}^{N/2-1} x^*_{1,2k+2g} v_{2k} x_{2,2k+2g} |^2}{2(\sum_{k=0}^{N/2-1} | x_{2,2k} |^2)^2} \tag{2-27}$$

the maximal value denoted as B^* of equation (2-27) is the maximum likelihood estimator of z [37].

2.3.2.2.2 Pilot techniques

The use of only pilot tones in each OFDM symbol to estimate time was initiated by Kim [38]. The behind idea of is that the phase offset between two subcarriers depends only on the frequency difference between those subcarriers.

$$\Delta\phi(q) = \phi_{q,k_{2n+1}} - \phi_{q,k_{2n}} = \frac{2\pi}{N}(k_{2n+1} - k_{2n})\tau \tag{2-28}$$

In (2-28), k_{2n} and k_{2n+1} are the position of the two tones of the $n-th$ pilot pair. If we transmit identical pilot tones then take the difference of the phases of the received tones, we can estimate the phase offset caused by timing error. The more pilot pairs are used, the better the timing synchronisation performance is. To take the advantage of all different pilot pairs to get timing offset estimated, the average phase offset, instead of the phase offset between two subcarriers, is used.

$$\Delta\phi(q) = \frac{1}{L}\sum_{n=1}^{L} \angle(Y_{q,k_{2n+1}} Y^*_{q,k_{2n+1}}) \tag{2-29}$$

where $Y_{q,2k+1}$ and $Y_{q,2k}$ are received pilot tones at the $k_{2n+1}-th$ and $k_{2n}-th$ subcarriers in the $q-th$ OFDM symbol.

2.4 Channel estimation in OFDM systems

Channel estimation is essential before the demodulation of OFDM received signals due to the distortion caused by frequency selective fading and time-varying fading. Channel estimation techniques can be classified in to two categories [39]:

- blind channel estimation techniques: these techniques estimate the channel information state without the knowledge of the transmitted signal; and
- data-aided channel estimation techniques: the known information added to the transmitted signals is used to estimate the channel response.

In the wireless communication systems using the blind channel estimation technique, no specialized reference (training) signal is needed, therefore, transmission efficiency is retained. On the other hand, without the knowledge of transmitted signals, in order to achieve a reliable estimation, a large amount of data must be collected. However, data-aided channel estimation techniques need to insert the known training signals to

the transmitted signals. The rapid and accurate channel estimation can be obtained by comparing the received signals and transmitted signals.

For the simplicity, the channel is assuming that receiver is perfectly synchronised $\varepsilon = 0 \text{ and } \Delta k = 0$. As a result, the received signal after DFT block during the $i-th$ symbol is expressed as

$$R(n) = H(n)C(n) + W(n), 0 \le n \le N-1 \tag{2-30}$$

where $C(n)$ is transmitted complex data symbol which is modulated by a PSK or QAM modulation, $H(n)$ is channel coefficient in frequency domain and $W(n)$ is defined in (2-18) .

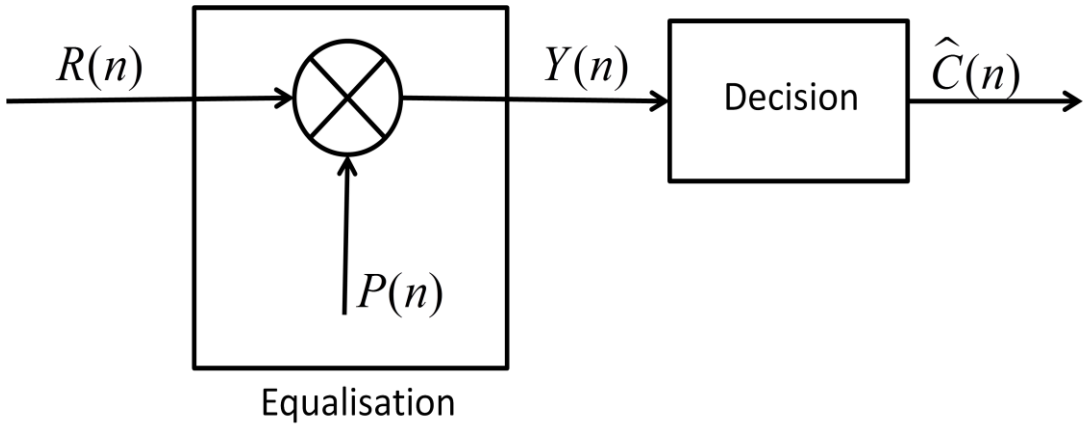

Figure 2-4 Channel estimation in OFDM receiver

An important advantage of OFDM system is that one-tap equalisation block can be operated independently over each subcarrier by a bank of one-tap equalisers. The $n-th$ DFT/FFT output $R(n)$ is multiplied by a complex-valued coefficient $P(n)$ to eliminate the phase rotation and channel-induced attenuation. The equalised sample $Y(n) = P(n)C(n)$ is subsequently fed to the decision block, which gives the final decisions $\hat{C}(n)$ to recover the $C(n)$ on the transmitted data. A simple channel estimation is that the equaliser coefficients is estimated in the pure channel inversion, which is known as Zero-Forcing (ZF) equalisation. The equaliser coefficients are then estimated by

$$P(n) = \frac{1}{H(n)} \tag{2-31}$$

while the DFT output has the following form

$$Y(n) = \frac{R(n)}{H(n)} = C(n) + \frac{W(n)}{H(n)} \tag{2-32}$$

The (2-32) equation shows that the Zero-Forcing equalisation is able completely to compensate any distortion caused by the wireless channel. On the other hand, at the equaliser output, the noise power, which is given by $\sigma_w{}^2 / |H(n)|^2$, may be extremely large due to strongly faded subcarriers and the low channel gains $H(n)$.

In this project, we focus mainly on data-aided channel estimation techniques because of the high accuracy of the technique. In data-aided channel estimation, known information is added to transmitted OFDM symbols so that the receiver can use the know information to estimate the current channel. There are two common data-aided channel estimation techniques, namely channel estimation with training symbols (sending known information over one or more whole OFDM symbols without any data), and pilots aided channel estimation (sending known information together with the data). The first type is referred to as a block type, while the latter is called comb type. These two techniques are shown in Figure 2-5.

Figure 2-5 Typical training symbols and pilot subcarriers arrangement[40]

2.4.1 Pilot-aided channel estimation

In the pilot aided channel estimation, the spacing of the pilot tones requires to be decided carefully. The pilot spacing in the frequency domain relies on the coherence

bandwidth of the wireless channel, which is related to the delay spread. According to the Nyquist sampling theorem, the number of subcarriers between two consecutive pilots in the frequency domain, D_p, has to be small enough so that the channel frequency response can be estimate correctly. That is,

$$D_p \leq \frac{1}{\tau_{\max} \Delta d_f} \tag{2-33}$$

where $\tau_{\max}$ is the maximum excess delay of channel and Δd_f is the subcarrier spacing. If the equation is not fulfilled, the channel then samples available at the pilot subcarriers do not reflect the actual channel accurately. Consequently, the estimation technique introduces an irreducible error floor since this causes the aliasing of the Channel Impulse Response (CIR) taps in the time domain [41].

If the pilot tones are inserted in time domain, the maximum spacing of plot tones across time [40] is given by

$$D_t \leq \frac{1}{2 f_{d_{\max}} T_f} \tag{2-34}$$

where $f_{d_{\max}}$ is the maximum Doppler spread and T_f is the OFDM symbol duration.

The pilots are transmitted continuously in each OFDM symbol, while the channel is varying in both time and frequency domains. Therefore the pilot tones have to be sampled at least at the Nyquist rate in order to reconstruct accuracy the channel. Consequently, the rate to insert pilots in the frequency domain and from one OFDM symbol to another should be set following (2-33) and (2-34). In general, the number of pilots within an OFDM symbol in the frequency domain should be larger than the maximum excess delay in the actual channel. Over the time, the Doppler spread is the main consideration for the pilot arrangements.

Numerous channel estimation studies based on the Mean Squared Error (MSE) of Least Square (LS) estimation were carried out to find the optimal pilot locations in both time and frequency domain and gave a minimum number of pilots in order to save energy in unnecessary pilot symbols and still keep the channel estimation accuracy. In addition, pilot arrangement not only minimizes MSE of the channel estimation, but also needs to reduce the complexity of the channel estimation

algorithm in order to reserve the system resources. For example, constant modulus pilots may be used to simplify the channel estimation algorithms as matrix operations reduce the complexity [42].

Another important consideration for pilot arrangements is the power distribution of the pilots. The power of pilot subcarriers and data subcarriers are commonly distributed equally. The performance of channel estimation can be enhanced by transmitting the pilot subcarriers with more power compared to data subcarriers [43]. However, this decreases the SNR over the data transmission due to increasing the total power. For pilot power allocation, studies show that pilot power in channel estimation based on the MSE of the LS estimation should be allocated equally [44].

In addition, channel estimation performs poorly at the edge due to the lack of the pilot subcarriers at the border of OFDM symbols [45]. It is also proved that due to the extrapolation the channel estimation performances at the edge subcarriers are significantly worse than those at the mid-bands. The simplest solution is increasing the number of pilot subcarriers at the edge of OFDM symbols [46]. However, this will reduce the spectral efficiency of the OFDM system. Using the periodic behaviour of the correlation of the Fourier Transform, the channel estimation can be improved at the edge subcarriers. Z. Jianhua and Z. Ping exploited this property to enhance the channel estimation performance at the edge subcarriers [47].

2.4.2 Training symbol-based channel estimation

Training symbols provide a good performance for channel estimation [48]. In training symbol-based channel estimation in OFDM system, all subcarriers of an OFDM symbol are dedicated for the training. In some communication wireless systems such as WLAN or WiMAX, two OFDM symbols are devoted for the training symbols. For slowly varying channels, if the training symbols are employed over two OFDM symbols, the channels for the same subcarriers is assumed to be the same over these two OFDM symbols [40]. In this case, the estimates on channel estimation can be averaged for further noise reduction. However, there is a trade-off between performance of channel estimation and the transmission efficiency due to the required overhead of training symbols.

It is commonly assumed that the channel is unchanged between OFDM training symbols [49]. Therefore, the channel response is used for next OFDM symbols until a new training symbol is received. This approach has been implemented for IEEE 802.11a//b/g and fixed WiMAX systems. On the other hand, this approach also introduces an error floor for non-constant channels including outdoor channels. The symbols located farthest from the training symbols has highest performance degradation[50]. Interpolation methods can be utilized in the time domain to overcome the above drawback. The simplest solution is a linear interpolation of the channel between the training symbols [51, 52]. The disadvantage of the interpolation approach is that the latency of the system is increased [53]. Therefore, if the system can tolerate the latency, then the performance of non-training OFDM symbol estimations can be improved with high order polynomials interpolation in time domain [54].

2.5 Signal-to-noise ratio estimation

Signal-to-Noise Radio is defined as the ratio of the desired signal power to the background noise power. It is a parameter commonly used to evaluate of the signal quality for communication systems. Moreover, SNR knowledge is commonly used to improve feedback channel estimation, and is a key decision parameter in adaptive processes such as dynamic reconfiguration of cognitive radios, Adaptive Modulation and Coding (AMC), turbo decoding and adaptive power allocation.

In the literature, most SNR estimation techniques proposed so far are related to single carrier transmission. Various SNR estimation algorithms were compared with the derivation of the Cramer-Rao bound in [55]. Some of single subcarrier SNR estimation schemes is directly used in OFDM systems with Additive White Gaussian Noise (AWGN) [56]. The SNR estimation techniques in OFDM system can be arranged roughly in two categories: data-aided and non-data-aided schemes (or blind schemes). Data-aided schemes require known signals such as pilot subcarriers can be transmitted in each OFDM symbol or a sequence of training OFDM symbol(s) that is called the preamble. SNR estimation schemes in OFDM system can also be categorized into the time-domain and the frequency-domain methods. As a result, SNR estimation schemes are divided in to one or a hybrid of the following four types: data-aided in time domain [57], data-aided in the frequency domain [56-60],

non-data-aided in time domain [61-63], and non-data-aided in frequency domain [64-66].

There are numerous SNR estimation algorithms for OFDM systems in frequency selective channels such as MMSE estimator [56], Boumard's estimator, Ren's estimator, and Periodic Sequence (PS) estimator [59, 65, 67]. The MMSE estimator derives the SNR estimation from an unknown information bearing portion of the received signal, which lead to degradation of performance. While MMSE estimator only estimates SNR in the time domain, Boumard's algorithm can performs the SNR estimation in both time domain and frequency domain. However, the disadvantage of the Boumard's estimate is highly sensitive to frequency selectivity and also high computational complexity. Ren's estimator can perform better than Boumard's SNR estimator in selective frequency channels, but it increases computational complexity. The PS estimator performs SNR estimation in white noise channel utilizing the preamble structure [68] proposed by Morelli and Mengali. It only requires the knowledge of the data and null subcarrier distribution for SNR estimation. The advantage of the PS estimator is lower complexity than Boumard's and Ren's estimator. The PS estimation, which performs SNR/sub carrier (the SNR is calculated in each subcarrier) instead of SNR/symbol or SNR/bits, is poor at low SNR value. Thus, at low SNR the estimate quires more complicated mechanisms. An enhanced estimation used the method of significant channel selection is proposed in [67] to improve performance of PS estimation at low SNR values.

Blind SNR estimation techniques in OFDM system can be applied to the payload field without knowledge of transmitted signal, such as preamble or pilot subcarriers. The high correlation between the time domain signal during the CP interval and that towards the end of the OFDM symbol can be used for the blind SNR estimation. The limitation of this approach is that since the most of the CP interval is interfered by the previous OFDM symbol, the number of useable samples in one OFDM symbol is small. Therefore, a large number of OFDM symbols must be collected to achieve the estimation accuracy [61, 62]. A high computational complexity method, Maximum-Likelihood (ML), was proposed in [63] with the assumption that the signal and noise covariance matrices are different and known. The disadvantage of this approach is also the difficulty to adapt to subcarrier or sub-band SNR estimation. The

expectation maximization algorithm with the assumption that the channel was known was used in [64] in order to solve the disadvantage. Thus, the performance of SNR estimation depends on channel estimation accuracy. A special preamble OFDM symbol which contains repeated signal segments in the time domain or equivalently with periodic virtual subcarriers in the frequency domain [59, 69] is used for SNR estimation. The limitation of these methods is that the SNR estimation in the preamble may not accurately represent the SNR of payload fields in frequency-selective and/or flat fading channel.

In this project, we implemented the blind SNR estimation with low complexity which is introduced in [65]. The detail algorithm will be presented in next chapter.

2.6 Software defined radio

Software Defined Radio (SDR) has become more prevalent in the formation of wireless networks. The idea behind SDR is to turn radio hardware problems into software problems [70]. In SDR systems, most signal processing is done (or at least managed) via software by using Field-Programmable Gate Arrays (FPGA), General Purpose Processors (GPP), or any other programmable device. The fundamental characteristic of SDR is that software defines the transmitted waveforms, and demodulates the received waveforms. This characteristic of SDR offers great flexibility for wireless communication research and development. A variety of newly-developed methods, algorithms and protocols can be implemented and verified easily in the similar manner as if they were done in a testbed constructed by real radio platforms. To achieve this, Radio Frequency (RF) components, such as mixers and filters, are also moved to the software domain, where digital signal processing is performed in baseband [71-75]. The emergence of low cost, high speed digital-to-analogue and analogue-to-digital converters has brought the software defined radio concept closer to reality. The SDR channel model is showed in Figure 2-6.

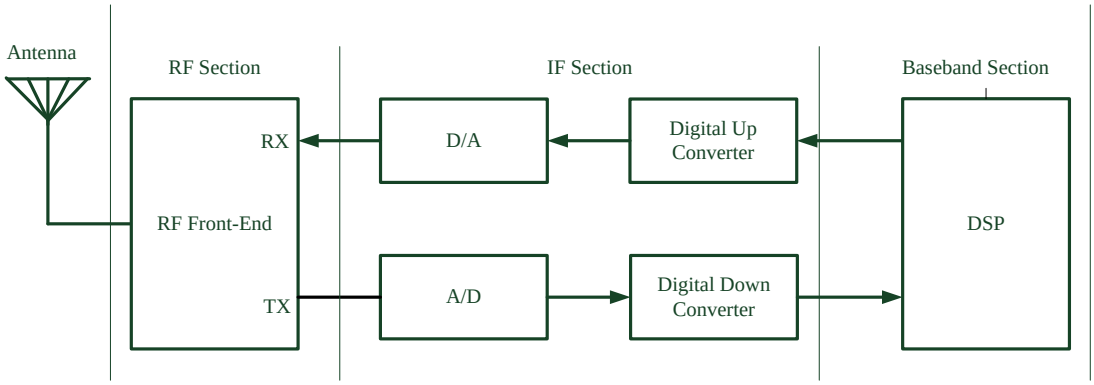

Figure 2-6 Software defined radio architecture

The SDR approach presented above in the ideal case is extremely simple and flexible. However, it is not practical due to limitations of actual hardware. The different hardware limitations are presented below

- Analogue-digital converters: The sampling rate according to the Nyquist sampling theorem must be at least twice as high as the bandwidth and the current ADCs are capable of sampling rates in the area of 100 Mega Samples Per Second (Msps). Therefore, the bandwidth limits to 50 MHz. While this bandwidth is enough for most current applications, the carrier frequency is usually higher than 50 MHz. Thus, a RF frontend is usually required to convert the received signal to an Intermediate Frequency (IF). In addition, the ADC resolution impacts the dynamic range of the receiver. The dynamic range can be generally estimated as

$$R = 6dB \times n \tag{2-35}$$

  where R is the dynamic range of ADCs and n is the number of bits in the ADC.
- Bus Speed: it is another issue to receive the data from the ADC to the computer. In practice, there is a maximum bus speed for the possible data rate, limiting the product of sample rate and resolution of the samples. The bus speed in PCs commonly ranges from a few Mbps to several Gbps.
- Performance of the Processing Unit: The Central Processing Unit (CPU) performance and the sample rate in real-time processing directly determine the number of computational operations that can be executed per sample and number of samples must be processed as fast as they receive. Therefore, fast CPUs, optimal programming and possibly parallelisation are required. If this

is not sufficient, there is a compromise to utilise a less optimal but faster signal processing algorithm in order to reduce the number of mathematical operations.

- Latency: Generally Personal Computers are not constructed for real-time processing applications. Therefore, in reality a rather high latency is introduced in SDR system. Many wireless standards require accurate timing. Thus, the latency should be considered carefully during implementing in an SDR system.

However, various solutions have been proposed to deal with these problems and help SDR become a promising low-cost research approach. There are numerous SDR platforms used to experiment. Most of them consist of free software that is constantly modified and open hardware which must be purchased to take care of RF part of the transmission. Currently, GNU Radio software and SDR4All Tool, which are free software for building SDR, and USRP, which is an open interface hardware platform for building SDRs developed by Matt Ettus [76], are used widely for numerous research purposes.

Wang et. al. [77, 78] proposed the use of USRP platforms with RFX2400 daughter boards and SDR4All tools with MATLAB for Cognitive Radio (CR) in order to discriminate three standard-like spectrums (e.g. GSM-like, UMTS-like and OFDM-like) under simple real transmission conditions. Tucker and Tagliarini [71] made use of GNU Radio and version-1 USRP (USRP1) to implement a Multiple-Input Multiple-Output (MIMO) Software Defined Radio platform in order to operate Mobile Ad-Hoc Networks (MANET) nodes. A. Kaszuba et. al. [79] implemented MIMO systems with the Alamouti STBC based on GNU Radio and USRP2. Experimental results of MIMO link performance in terms of Frame Error Rate (FER) versus SNR were presented there. Kun-chan and Mei-wen [80] presented a feasibility study on the use of FM radio for data transmission in vehicular networks, using GNU Radio and USRP boards to observe the packet loss rate and packet correct rate in different distances. Marwanto et. al. [81] explored the viability of using GNU Radio and USRP to transmit and receive the OFDM radio signal with QPSK and BPSK modulations. In [82], an energy detector is developed based on a USRP2 with XCVR2450 daughter boards, that work in the 2.4 and 5 GHz bands, in order to

verify detector performance in theory with its performance under realistic propagation environments. A Single Carrier Frequency Division Multiple Access (SC-FDMA) transceiver is implemented utilizing USRP2 and MATLAB in [83]. Experimental results in terms of the Bit Error Rate (BER) versus Signal-to-Noise Ratio were presented and compared to theoretical and simulated performances.

2.7 GNU radio

GNU Radio is an open software platform for building software defined radio systems. Anyone is allowed to utilise, exploit and adjust GNU Radio without limitation under the GNU General Public Licence (GPL). The target of GNU Radio according to the projects founder Eric Blossom is to "bring the code as close to the antenna as possible", and "turn hardware problems into software problems" [70].

GNU Radio offers various building blocks for signal processing, as well as a method to manipulate the data flow between the blocks. This is a partial list of the functions offered by GNU Radio software platform including:

- Mathematical operations such as add, subtract, multiply, divide, power, logarithm and logic operations;
- FFT/IFFT blocks
- Filters including High pass filter, low pass filter, band pass filter, band reject filter, FFT filter, Finite Impulse Response (FIR) filter, Infinite Impulse Response (IIR) filter and Hilbert;
- Modulations and demodulations such as FM, AM. PSK, QAM, Differential Encoder, Differential Decoder and OFDM ;
- Control blocks including Automatic Gain Control (AGC) blocks, Detect Peak block, Threshold block;
- Type conversions such as Float to Short block, Int to Fload block, and Complex to Real
- Interpolation and decimation;
- Line coding such as Scrambler, Descrambler and Adaptive Scrambler
- Channel coding such as Trellis coding and Viterbi decoding.

GNU Radio software platform also offer support for different signal sources and sinks as following:

- Constant source;
- Noise source;
- Pseudo random number source;
- Random vector source;
- USRP source and sink;
- Message source;
- Graphical sinks such as Oscilloscope sink, Eye Diagram, FFT sink, Waterfall sink, Constellation sink and Histogram sink;
- Audio source and sink;
- File source and sink;
- Transmission Control Protocol (TCP) and User Datagram Protocol (UDP) source and sink.

At the beginning, GNU Radio was established on a Linux platform. At the moment, it currently supports different platforms including UNIX, Windows and MAC platforms.

2.7.1 GNU Architecture

GNU Radio software architecture includes two components. The first one is the set of various C++ blocks which implement digital signal processing operations for example filtering, I/O operations, FFT/IFT, coding/decoding, and modulation/demodulation. The second one is the framework implemented as Python scripts to control the data flow among blocks. The use of Python scripts allows easy reconfiguration and manipulation of various functionalities and parameters of the system. Similar to connecting physical RF building blocks to construct a hardware radio, any user can build a SDR system by "wiring" together building blocks. USRP boards are general purpose RF hardware for GNU Radio architectures. The main task of USRP boards is performing computationally intensive operations such as filtering, up-conversion and down-conversion. The USRP, USRP2 and its current version USRP N-series are connected to a computer via a USB 2.0 and/or an Ethernet cable, respectively, and the GNU Radio software platform provides a robust Application Programming Interface (API) to control the USRP device.

A data flow shown in Figure 2-7 among different blocks is abstracted by a flowgraph, which is a directed acyclic graph where the vertices are the GNU Radio blocks and the edges correspond to data streams.

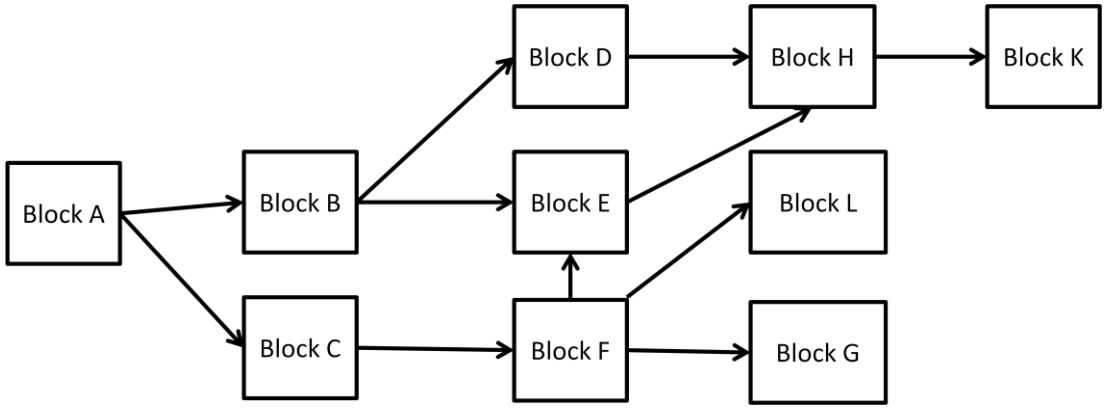

Figure 2-7 An example of flowgraph

Generally, GNU Radio blocks shown in Figure 2-8 perform on continuous data streams. Each block includes a set of input and/or output ports and it receives data from input port and produce data for its output port. Special blocks called sources and sinks only input port or output port. Examples of sources/sinks are blocks that read/write, respectively, from/to USRP receiver/transmitter ports, sockets and files. Each block defines the minimum and maximum number of input and output port the block can have, as well as the data type on the corresponding port. The supported data types are:

- c - complex interleaved floats (8 Bytes each),
- f - floats (4 Bytes),
- s - short integers (2 Bytes), and
- b - Byte integers (1 Bytes).

Each block defines a *general_work()* function that performs on its input to procedure output data streams. Blocks also provide a *forecast()* function that returns the system runtime the number of input data streams it requires to perform, and the number of output data streams it produces from the given input items.

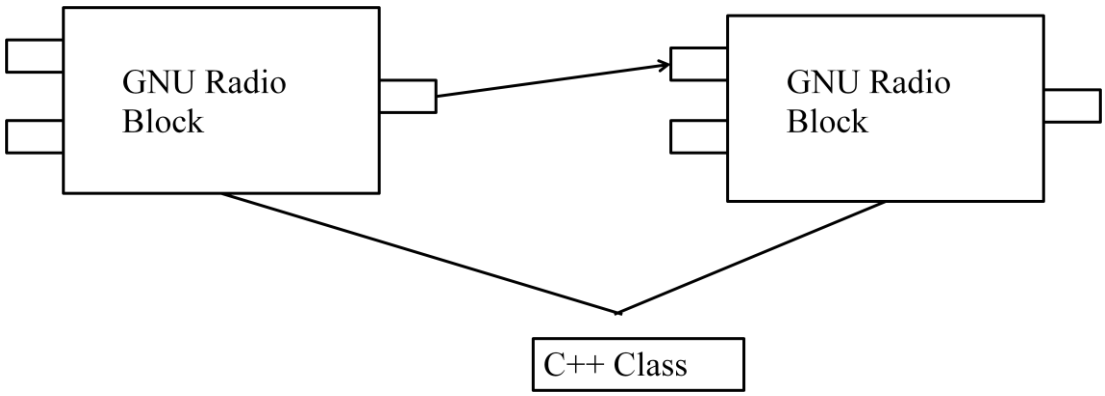

Figure 2-8 GNU Radio Blocks

The data rate on different input stream of a block can be different, but all output streams must have same data rate. Both input and output streams of a block have associated buffers. Each input/output data stream has a read/write buffer. The block read data from the read buffers for signal processing. After processing, the block write data streams to the appropriate write buffers of its output data streams. The data buffers are utilised to implement the edges in the flowgraph: the write buffers for one block are the read buffers of the upstream block in the flowgraph. GNU Radio buffers are single writer and multiple reader First In First Out (FIFO) buffers. It means that one output data stream can connect to one or more input data stream(s) and one input on receive data from only one output. Blocks in Python are connected by the virtual *connect* function which indicates how the output data stream(s) of a block connect to the input data stream of one or more upstream blocks. The flowgraph mechanism then automatically constructs the flowgraph and this is hidden from the user. The main function of the flowgraph mechanism is the allocation of data stream buffers to connect blocks. The mechanism takes into account the sizes of input and output data streams used by blocks and the associated data rate at which blocks consume and generate input and output data streams. After buffers are allocated, they are linked to the input and output data streams of the appropriate blocks.

A hierarchical block can be produced by combining several blocks as shown in Figure 2-9. A hierarchical block is constructed in Python and, together with other blocks; they can be combined into a new hierarchical block.

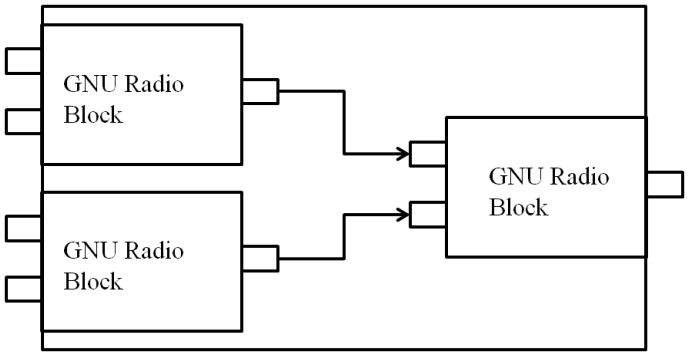

Figure 2-9 An example of a hierarchical block

During the GNU Radio scheduler execution, the scheduler check the input requirements of each block and the mentioned *forecast()* function is used to decide the received data rate of the block from its available inputs. If data is appropriate in the input buffers, block's *general_work()* function is executed. If a block does not have appropriate data in input buffer, the scheduler will skip the block and move on to the next block in the flowgraph. Skipped blocks will not be performed until more input data is sufficient. The GNU Radio scheduler is constructed to perform on runtime data streams.

2.7.2 GNU Radio Companion

GNU Radio Companion (GRC) is a graphical interface for GNU Radio user. GNU Radio Companion is an open source tool based on Python/C++ to build software radio systems. GRC allows connecting components and generating a signal flow diagram by using a graphical drag-and-drop interface. Component blocks are normally implemented in C++ and connected using the Python programming language. GRC currently is a part of GNU Radio and is under development by Josh Blum [84]. Though, the visual programming is not as powerful and flexible as the programming in Python, the visual programming is more intuitive than programming in Python for the user. With GRC, a simple FM broadcast receiver is easy to build in the SDR system as shown in Figure 2-10.

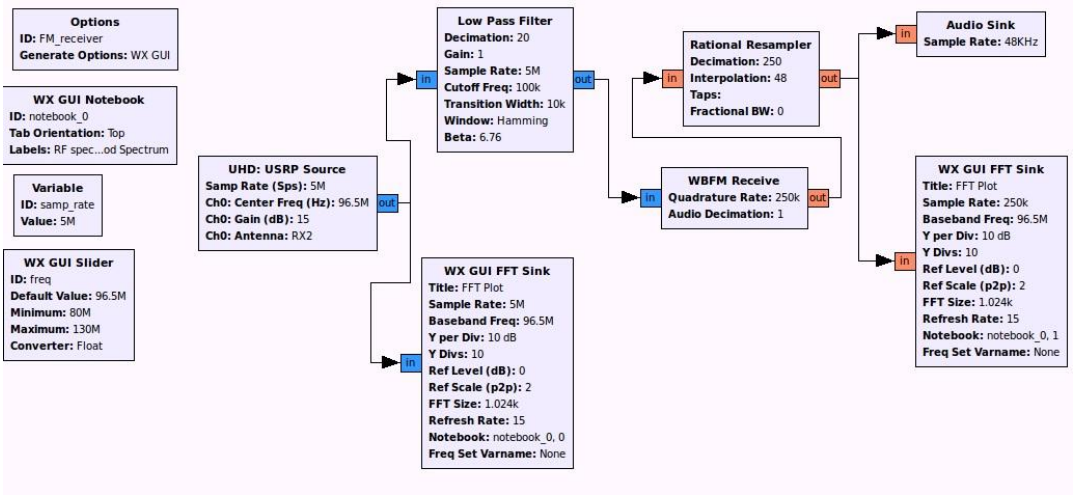

Figure 2-10 FM receiver built by GRC

2.8 Universal software radio peripheral

The Universal Software Radio Peripheral is the most common hardware used with GNU Radio to build a SDR system. USRP is a family of hardware devices found by Matt Ettus which facilitate making SDRs. It is composed of two main sub devices, a motherboard and various daughterboards which can convey and/or receive in different frequency ranges from 0 to 5.9GHz. The daughterboard can easily be exchanged. This allows USRP to work at various frequencies.

2.8.1 General architecture of USRP motherboards

Motherboard is composed of an onboard Field Programmable Gate Array (FPGA) and an onboard/removable memory. At the first time using the USRP device, USRP firmware and FPGA image must be downloaded into the memory. Then the device is able to correspond with any computer installed USRP Hardware Driver (UHD), via a Gigabit Ethernet (or USB port in USRP USB series). Apart from the digital down conversion, discussed later in this section, the FPGA does not perform any advanced digital signal processing, such as modulation, demodulation and coding. Such processing is always done in the computer.

The primary objective of USRP motherboards is to convert analogue IF signals to digital baseband signals and vice versa (RF-IF conversion is done on the daughterboard. This is further discussed in Section 2.8.3). The bandwidth of the IF signal processed by the motherboard depends on the daughterboard specifications

(40 MHz in the daughterboard used herein). To convert analogue signals to digital signals, the motherboard uses two high speed analogue-to-digital converters. The digitalised IF signal is then executed in the on-board FPGA. Therein, the digital decimating low pass filter preloaded in the FPGA processes the signal. Generally, the data rate of this digitalised signal is too high for general computer processing [70]. Therefore, the purpose of filtering is to reduce the sampling rate to suit the processing capabilities of the computer and the capacity of Ethernet cable. The functionality of FPGA in reducing the sampling rate is shown in Figure 2-11.

Along the transmit path, digital information is sent to the FPGA which is converted to an analogue signal with two high speed digital-to-analogue converters. The analogue signal is then fed to the daughterboard to transmit as a RF signal.

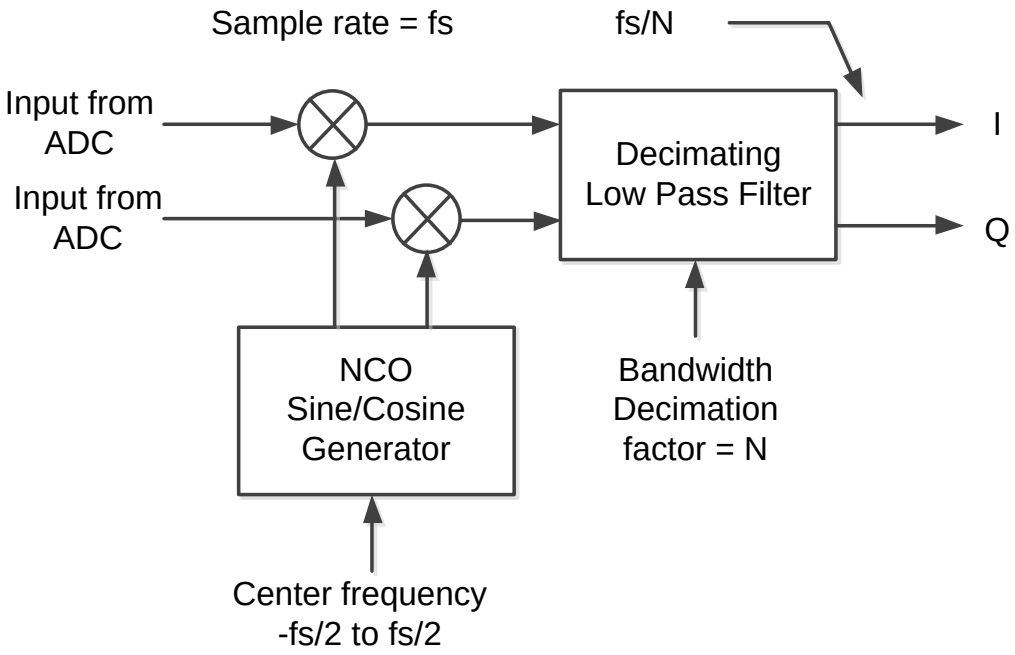

Figure 2-11 Decimating low pass filter function in FPGA [70]

2.8.2 Version-2 USRP

The version-2 USRP (USRP2) was built up on the success of the original USRP, offering better performance and increased flexibility at a lower price. It features a Gigabit Ethernet interface 25 Msps to communicate with PC, Xilinx Spartan 32000 Field Programmable Gate Array (FPGA), SRAM 1MB, 2 Analogue-to-Digital Converters (ADCs) at 100 MS/s, 14 bits, 2 Digital-to-Analogue Converters (DACs)

at 400 MS/s, 16 bits, and a Secure Digital (SD) card reader for the firmware. The USRP2 architecture and its features are shown in Figure 2-12.

The internal USRP2 clock is a 100 MHz Voltage Controlled Oscillator (VCO) with a 10pp nominal accuracy; USRP2 also provides a connection to an external reference clock as well as a Pulse-Per-Second (PPS) port [85]. USRP2 provides connections to daughter boards which serve as RF front-ends.

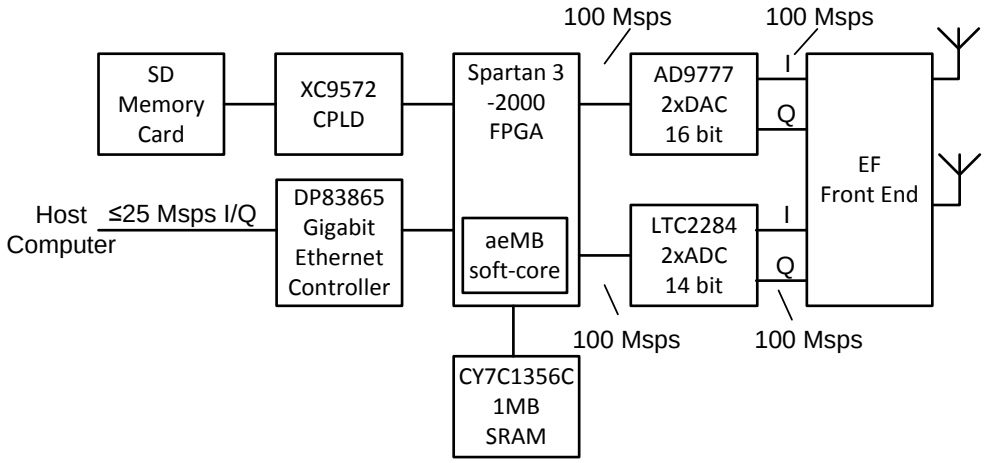

Figure 2-12 Block diagram of USRP2 main components and signalling rates [83]

This project uses USRP2 for experiments due to its better properties compared to USRP1 which are shown in Table 2-1.

Table 2-1 USRP1 and USRP2 Comparison

	USRP1	**USRP2**
Interface	USB 2.0	Gigabit Ethernet
FPGA	ltera EP1C12	Xilinx Spartan 3 2000
RF Bandwidth to/from host	8 MHz @ 16bits	25 MHz @ 16bits
ADC Samples	12-bit, 64 MS/s	14-bit, 100 MS/s
DAC Samples	14-bit, 128 MS/s	16-bit, 400 MS/s
Daughterboard capacity	2 TX, 2 RX	1 TX, 1 RX
SRAM	None	1 Megabyte
Power	6V, 3A	6V, 3A

2.8.3 Daughter boards

An SDR hardware platform should be able to capture and process weak signals in a wide dynamic range, which would demand for multi GHz speed ADCs. However, it is infeasible to realise ADCs at that ultra high speed (USRP device has 100 M Samples/s ADCs only). To solve this issue, Ettus Research had introduced the daughterboards that can be plugged into USRP motherboards via transmit/receive ports. The band pass filter on daughterboard narrows down the frequency range, and then the residual RF is converted down to IF such that the subsequent ADC can handle the speed of the signal.

USRP motherboards support various daughterboard, such as RFX 400/900/1200/1800/2400, LFTX, LFRX, TVRX2 and WBX. Most of these daughterboards have a narrow dynamic range, where the operating frequencies are in the MHz bands while WBX has the widest range from 50 MHz to 2.2 GHz. Therefore, it is an alternate for the RFX, LFTX, LFRX and TVRX2 range, except the 2400 MHz band. In particular, WBX could be used for broadcast television, white spaces, public safety, land-mobile communications, cell phones, wireless sensor networks, low-power unlicensed devices, and six amateur radio bands. Therefore, the daughterboard WBX (Figure 2-13) is chosen for this project. It transmits at the power of 30-100 mW and produces a full-duplex channel.

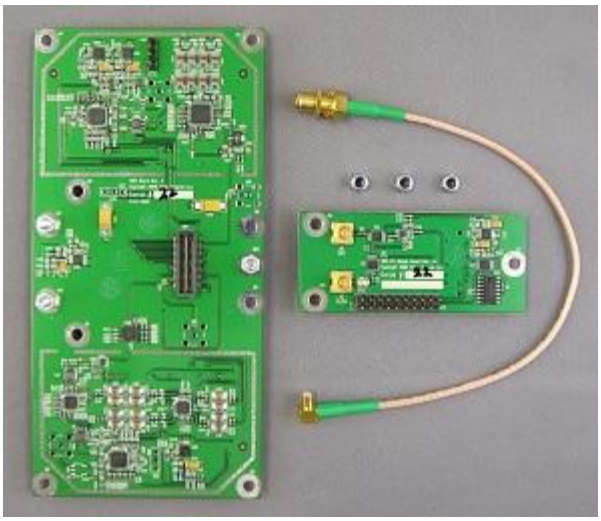

Figure 2-13 WBX daughterboard

The WBX daughterboard has two interfaces named TX/RX and RX2. It can provide up to two antennas, one on each interface. The TX/RX interface can be used either for transmission or reception at any given time, while RX2 interface is only used for reception. User can select the preferred interface for communication in the SDR script depending on the application. The interfaces available are:

- transmit only: User must select TX/RX for transmission
- receive only: Either TX/RX or RX2 can be selected. Both interfaces cannot be used at any given time to receive two frequencies.
- full-duplex mode: In this mode, regardless of the predefined choice, FPGA automatically selects TX/RX as the transmit path and RX2 as the receive path. Consequently, two antennas should be attached to the device in this full duplex mode, one per each interface.

2.9 Research questions

While OFDM with its advantages has been widely used in digital communication systems nowadays, SDR is an emerging and developing technique, thus it has not been matured yet. Therefore, the combination of SNR system and OFDM-based system becomes a interesting and attractive research field. Although, OFDM and SDR have been somewhat examined for general wireless networks in the literature such as [79, 81, 85-87], it has been almost unexplored due to the hardware limitations, the difficulties on system set-up and the large knowledge requirement on different topics such as channel estimation, time and frequency synchronisation, and SNR estimation. Consequently, there is a lack of comparisons between OFDM system performance in different practical environments, with different OFDM configurations, and with different hardware configurations. Particularly , the performance of OFDM system is just verified in AGWN channel and laboratory room in short distance

Thus, the research questions are summarised as follows:

- How does the performance of OFDM systems show in realistic propagation environments?
- How to verify the performance of OFDM systems by hardware?

- What are the differences between the performances of OFDM systems in MATLAB simulation and in actual hardware?
- What are the performances of OFDM systems in different propagation environments with different system configurations?

2.10 Contributions

In this thesis, we have

- Addressed all the above questions;
- Built successfully a testbed using USRP2 to verify the performance of OFDM systems;
- Collected and analysed collected data of OFDM performances in different propagation environments, including in-door, corridors, and out-door;
- Compared the experimental results with the simulated results;
- Explained and interpreted deeply the experimental results;
- Put forwards some recommendations and suggestions for further research in this topic

2.11 Chapter summary

This chapter provides readers with a state-of-the-art literature review of all the main techniques and basic concepts which are critical for understanding this thesis including, OFDM-based systems, SDR systems, USRP, GNU, synchronisation, channel estimation and SNR estimation. It also outlines the open research problems which are the main motivations for this project and the thesis contributions. The next chapter starts to present my approaches and methodologies used in this project.

3 CHANNEL AND SNR ESTIMATION TECHNIQUE FOR OFDM SYSTEMS

The fundamental structure of OFDM systems has been introduced in Section 2.2. In this chapter, the performance of OFDM systems with perfect channel state information, perfect SNR estimation and perfect channel estimation will be presented in Rayleigh and Rician fading channels. The chapter then discusses the channel estimation and SNR estimation techniques. The performance of these techniques is evaluated by MATLAB. Finally, this chapter provides the detailed block diagram of the complete OFDM system (basic OFDM system with channel and SNR estimations). This imitative system will be used later in this thesis to compare its performance with the practical error performance of the realistic OFDM system implemented in hardware.

3.1 OFDM with perfect channel state information

The error performance of an OFDM system with perfect channel estimation, synchronisation and SNR estimation is simulated in MATLAB with different propagation environment models, including the Rayleigh environment and the Rician environment with different K factor values.

The fundamental principle of OFDM is that a high-rate data stream is separeted into a number of lower rate streams that are modulated onto orthogonal carriers. This process is done in the discrete time domain through IFFT element and the resulted signal is transmitted parallel over a number of orthogonal subcarriers. The relative amount of dispersion in time caused by multipath delay spread is decreased because the symbol duration increases for these lower rate streams. The Inter-Symbol Interference (ISI) problem is effectively solved by a guard time in each OFDM symbol. By using the guard time, each OFDM symbol is cyclically extended to avoid ISI. An OFDM signal is a sum of subcarriers that are individually modulated by using Phase Shift Keying (PSK) or Quadrature Amplitude Modulation (QAM). The OFDM system used in our MATLAB simulations is depicted in Figure 3-1.

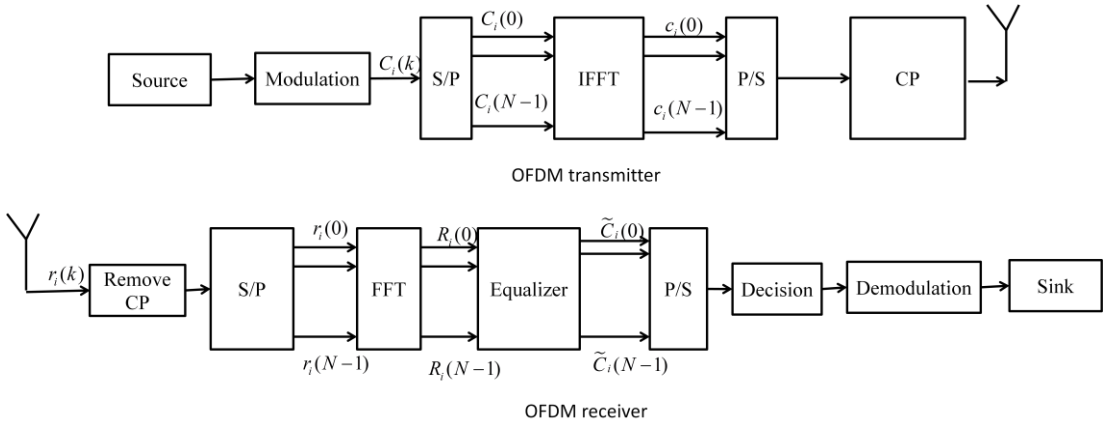

Figure 3-1 OFDM system block diagram

The bit stream from the source is separated into groups of M bits, and each group is then fed to the modulator. The modulator maps each group of M bits to the corresponding complex symbol (the modulated symbol). Let $C_i(n)$ denote the modulated symbol stream. The complex data symbols $C_i(n)$, for $n = 0,1,...,N-1$ within the *i*-th OFDM symbol are created by either a PSK or QAM constellation. The symbol stream is converted into parallel blocks of size N and the IFFT operation is applied to obtain time-domain OFDM symbols. The the *i*-th OFDM symbol after IFFT in time domain is given by

$$c_i(k) = \frac{1}{N}\sum_{n=0}^{N-1} C_i(n) e^{i2\pi kn/N} \quad 0 \le \mathrm{k} \le N-1 \tag{3.1}$$

To reduce the ISI between successive blocks of data, the transmitter copies a partial repetition of the tail part (N_g samples) of every block to the head of the block, which is known as a cyclic prefix, to construct an OFDM symbol consisting of $N + N_g$ samples.

Transmitted signals over wireless channel are typically diffracted, reflected and scattered. Thus, arriving signals at the receiver are combination of different copies of the same signal with different amplitudes, time of arrival and phases. A common model to represent the wireless channel effects is use of the channel impulse response, which is given as

$$h(k) = \alpha(l) e^{j\theta(k)}, \text{ for } \mathrm{l} = 0,...,L-1$$

where L is the number of receiver signal paths in the transmission channel, $\alpha(k)$ and $\theta(k)$ are attenuation and phase shift of the $l-th$ path. In addition to multipath effects, noise is introduced to the transmitted signal. The most of the source noises is additive noise such as thermal background noise, electrical noise in the receiver amplifiers, and interference. The total effective noise at the receiver of an OFDM system can be modelled as Addictive White Gaussian Noise (AWGN) with a uniform spectral density and zero-mean Gaussian probability distribution. Therefore, the signal received through a channel with impulse response h and noise w can be expressed as

$$r = c * h + \mathrm{w} \tag{3.2}$$

where "*" denotes the linear convolution.

At the receiver, the cyclic prefix is removed from the received signal. The resulting signal is then converted into parallel blocks of size N. FFT operation is applied to obtain OFDM symbols in the frequency domain.

$$R_i(k) = \sum_{n=0}^{N-1} r_i(n) e^{-i2\pi kn/N} \quad 0 \le \mathrm{k} \le N-1 \tag{3.3}$$

The resulting signal is fed to the simple 1-tap equalizer block. Denote the channel frequency response to be H_f then H_f is given by the FFT of the channel impulse response h as follows

$$H_f(k) = \sum_{n=0}^{N-1} h(n) e^{-i2\pi kn/N} \quad 0 \le \mathrm{k} \le N-1 \tag{3.4}$$

If channel estimation is ideal, the signal after the equalizer block is given by

$$C_i(k) = R_i(k)./H_f(k), \text{ for } k = 0,1,...,N-1 \tag{3.5}$$

The maximum likelihood decision is then used to detect each symbol $C_i(k)$ from $C_i(k)$. The signal is finally demodulated and reversed to the original signal.

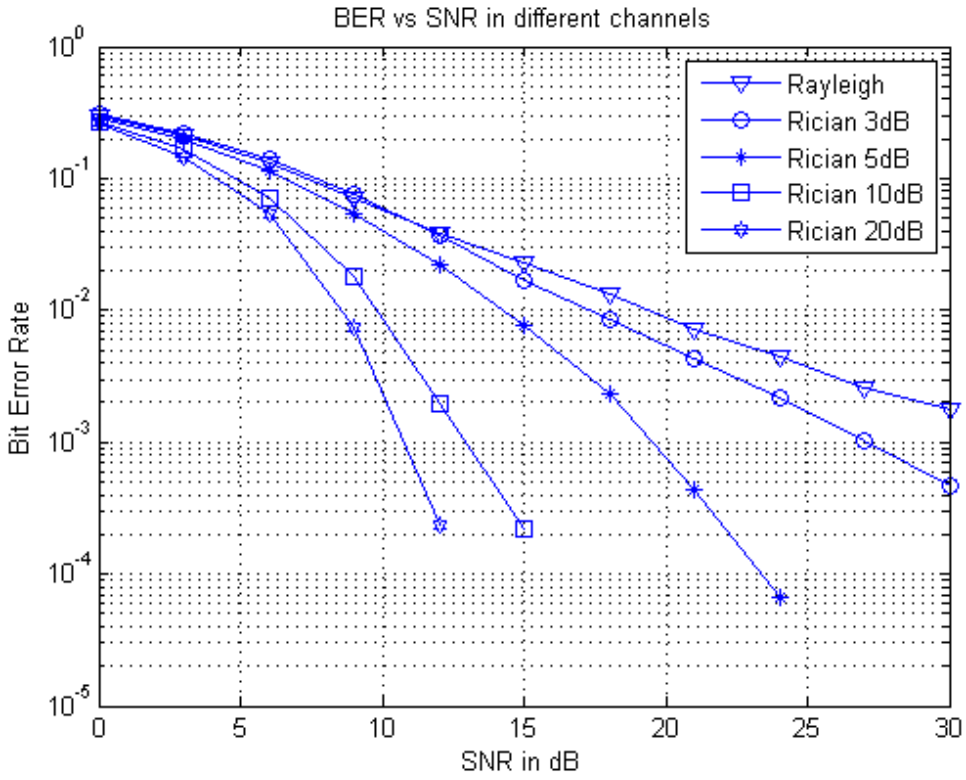

Figure 3-2 Performance of OFDM systems in different propagation environments

3.2 Channel estimation in OFDM

In this section, the preamble-based channel estimation algorithm for OFDM systems in the frequency domain proposed by Beek [88] is presented and simulated by MATLAB with the assumption that timing and frequency synchronisation is perfect and channel information state remains static over each frame duration, i.e. $H_i(n) = H(n)$ for $i = 1,...,I$ where I is the total number of OFDM symbols within one frame. Estimated channel response obtained from the preambles can be applied to coherently detect the whole frame.

The OFDM frame structure in our simulations is shown in Figure 3-3. An OFDM frame consists totally of $I = 8$ OFDM symbols (1 preamble and 7 OFDM data symbols). Each OFDM symbol consists of $N\_subcariers = 40$ data, $N\_zeros = 24$ zeros for synchronization, and $N\_CP = 16$ samples of the cyclic prefix for the guard interval. The preamble symbol has the same structure as other OFDM symbols. The only difference is that all of $N\_subcariers = 40$ subcarriers in the preamble are the known Pseudo Noise (PN) sequence used for channel estimation, while these

subcarriers in OFDM symbols are used for data transmission. The PN sequence only consists of ± 1 in even subcarriers and 0 in odd subcarriers.

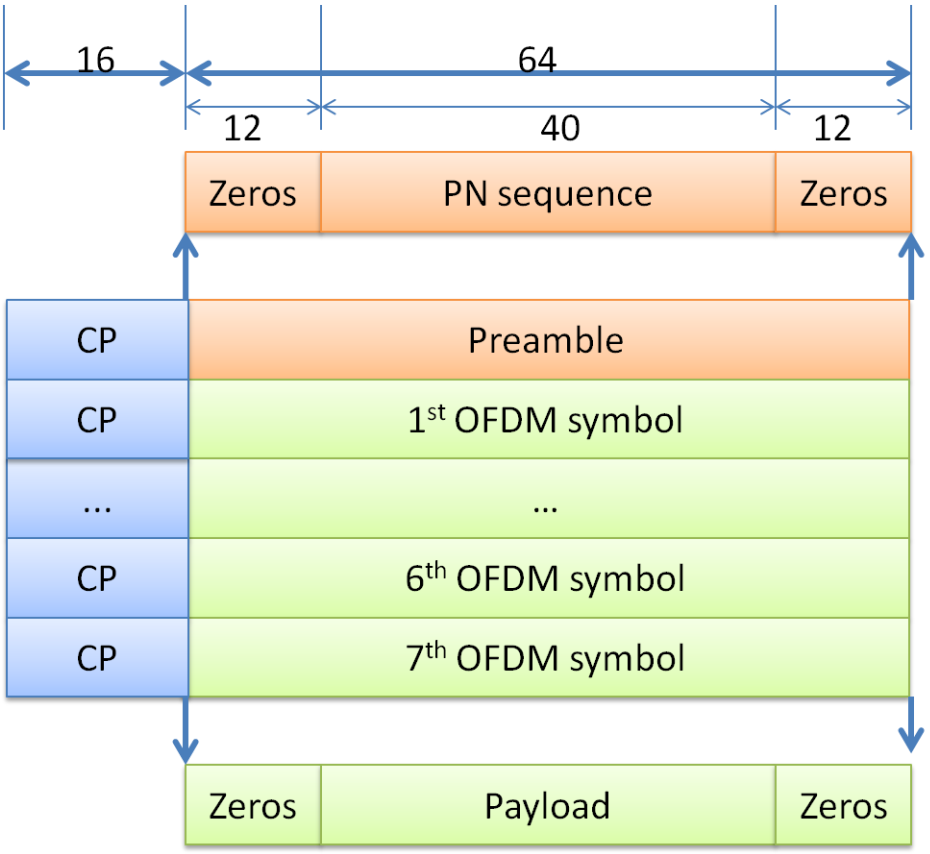

Figure 3-3 OFDM frame structure

The block diagram of an OFDM system used to simulate the performance of channel estimation algorithms is shown in Figure 3-4.

An OFDM frame has one preamble symbol. The preamble sample received at the output of FFT block can be written as

$$R(k) = C(k)H(k) + W(k) \text{ for } 0 \le k \le N-1 \tag{3.6}$$

where $C(k)$ are the complex data symbols within the preamble which are known to the receiver. The Least-Square (LS) estimate or Minimum Mean-Square Error (MMSE) estimate of the channel frequency can be obtained.

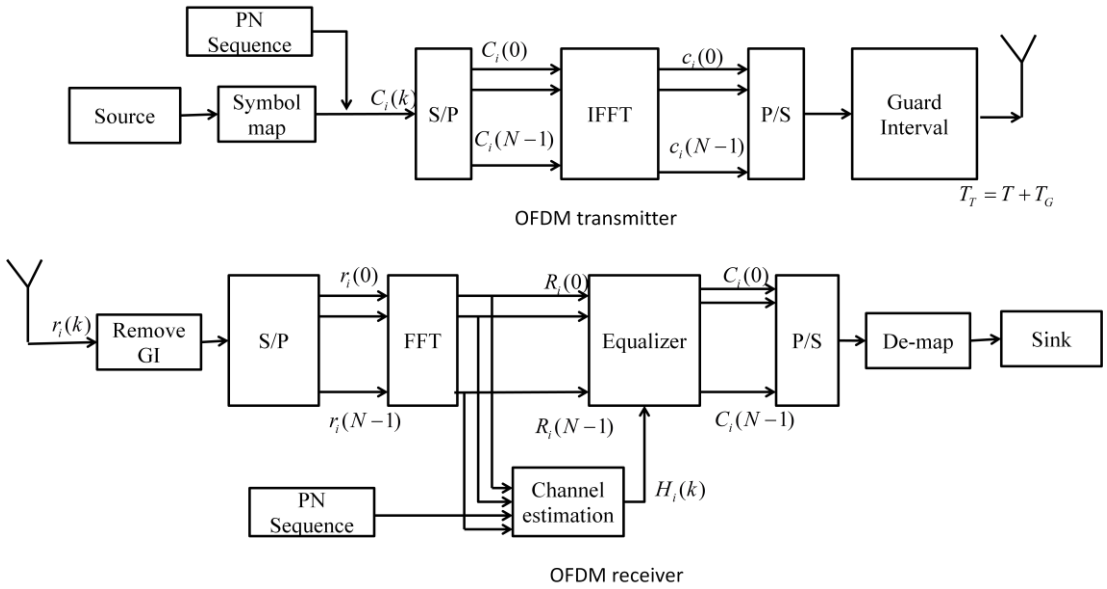

Figure 3-4 Block diagram of OFDM systems with channel estimation

In particular, the LS estimate is given below

$$H_{LS}(2k) = \frac{Y(2k)}{C(2k)} \text{ for } 0 \le \text{k} \le \text{N/2-1} \tag{3.7}$$

Because the PN sequence only consists of ± 1 in even subcarriers and 0 in odd subcarriers, the channel impulse responds for odd subcarriers must be interpolated. Popular interpolation methods include linear interpolation, second-order polynomial interpolation, and cubic spline interpolation. For simplicity, in this project, linear interpolation is used as follows

$$\begin{aligned} &H_{LS}(2k+1) = H_{LS}(2k) + H_{LS}(2k+2) = \text{ for } 0 \le \text{k} \le \text{N/2-1} \\ &\text{H}_{LS}(N-1) = H_{LS}(N-2) \end{aligned} \tag{3.8}$$

The MMSE estimate is given below

$$H_{MMSE} = R_{HH_{LS}}(R_{HH} + \frac{1}{SNR} * eye(FFT\_size))^{-1} H_{LS} \tag{3.9}$$

where R_{AB} denotes the cross-correlation matrix of two $N \times N$ matrices A and B.

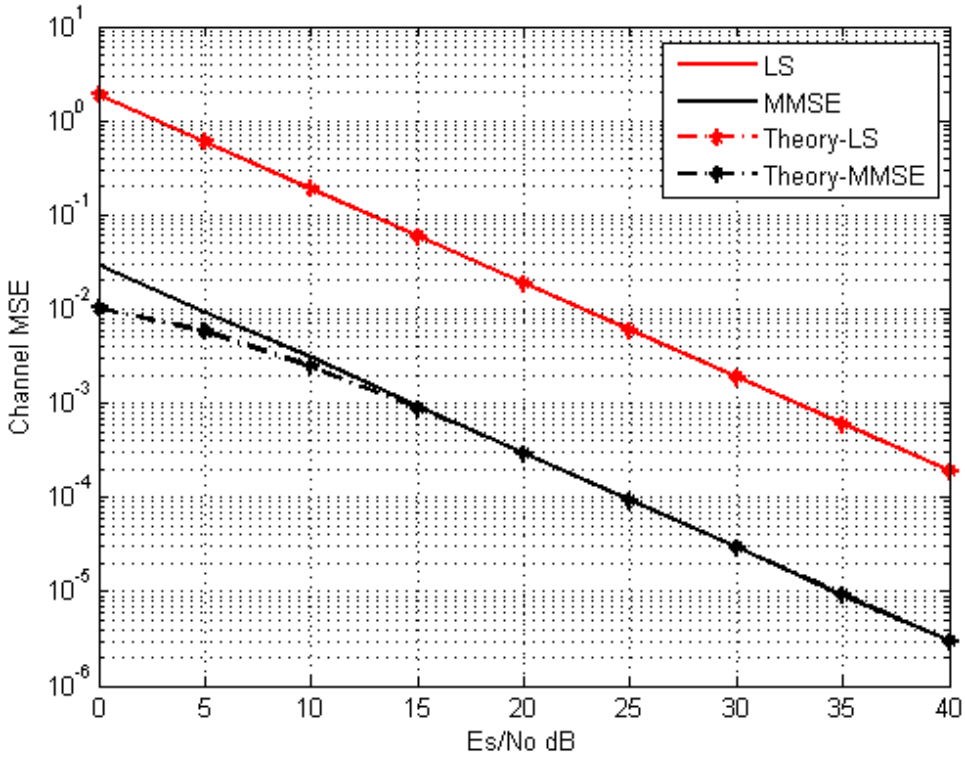

Figure 3-5 Comparison of LS and MMSE estimations in OFDM systems

Although channel estimation using MMSE estimates is better than LS estimates, we prefer to use LS estimation in simulations and experiments. It is because the main aim of this project is to compare the OFDM system performance in simulations and in experiments in different realistic propagation environments. Therefore, the algorithms used in this project require the ability to apply easily in both simulation and hardware implementation. MMSE estimates requires SNR and channel impulse response to be known at the receiver which are difficult to estimate in a real time system.

3.3 SNR estimation

In this section, SNR estimation in OFDM systems is simulated with the assumption that time and frequency synchronisations are ideal. The structure of OFDM frames and OFDM symbols has been shown earlier in Figure 3-3. The OFDM system model is used for SNR estimation is depicted in Figure 3-6.

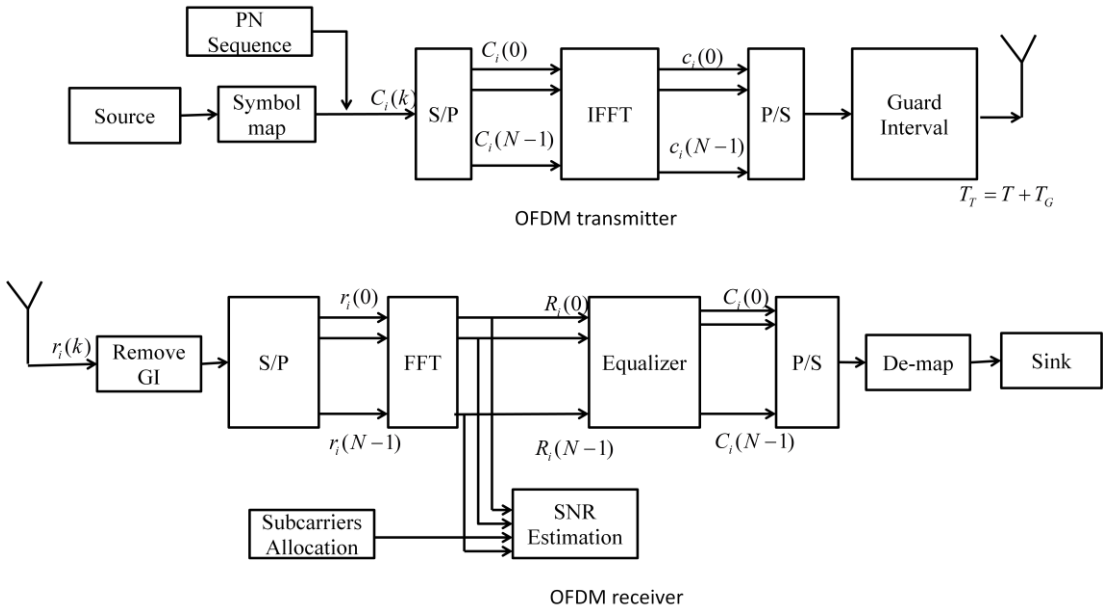

Figure 3-6 Block diagram of OFDM systems with SNR estimation

In the frequency domain, the received OFDM signal $R_j(k)$ at time j and subcarrier k is presented by

$$R_j(k) = C_j(k)H_j(k) + \mathrm{W}_j(k) \tag{3.10}$$

The instantaneous SNR ψ within interval J and the set of subcarriers K is given by

$$\psi = \frac{\sum_{j\in J}\sum_{k\in K} |C_j(k)H_j(k)|^2}{\sum_{j\in J}\sum_{k\in K} |W_j(k)|^2} = \frac{\sum_{j\in J}\sum_{k\in K} |C_j(k)H_j(k)|^2}{\sum_{j\in J}\sum_{k\in K} |R_j(k) - C_j(k)H_j(k)|^2} \tag{3.11}$$

If at the receiver, both the transmitted symbols $C_j(k)$ and the channel frequency responses $H_j(k)$ are known, the instantaneous SNR can be estimated by the equation (3.11). During the preamble interval $C_j(k)$ is known. Therefore, in the preamble fields estimated SNR can be compute as follows

$$\psi = \frac{\sum_{j\in J}\sum_{k\in K} |C_j(k)H_j(k)|^2}{\sum_{j\in J}\sum_{k\in K} |R_j(k) - C_j(k)H_j(k)|^2} \tag{3.12}$$

where $H_j(k)$ is the estimated channel frequency response. Therefore, the accuracy of the SNR estimation depends on the accuracy of the channel estimation. The simple LS channel estimate can be given by

$$H_j(k) = \frac{R_j(k)}{C_j(k)} \tag{3.13}$$

From (3.12) and (3.13), it is clear that the simple LS channel estimate cannot be applied for the blind SNR estimation since it leads the estimated SNR to infinity $\psi = \infty$. Therefore, we decide to use a blind SNR estimation in this project. The algorithm to estimate SNR in OFDM systems is presented in [65] and is reviewed below.

If Q is defined to be the set of subcarriers in an OFDM symbol where no signal is transmitted, the frequency domain samples for the subcarriers in Q at the receiver are primarily the noise samples $W_j(k)$

$$R_j(k) = W_j(k) \quad \mathrm{k} \in \mathrm{Q} \tag{3.14}$$

The Noise-to-Noise ratio (NNR) between noise power on the no-signal-transmitted subcarriers (the set *Q*) and noise power on the signal-transmitted subcarriers (the set *K*) is defined as follows

$$\eta = \frac{\sum_{j \in J} \sum_{k \in Q} W_j(k)}{\sum_{j \in J} \sum_{k \in K} W_j(k)} \tag{3.15}$$

If the noise is white noise such as AWGN, the expected value of NNR approaches a constant value

$$\eta = \frac{\| Q \|}{\| K \|} \tag{3.16}$$

where ||(.)|| is the number of subcarriers in the corresponding set. From (3.12) and (3.15), the estimated SNR ψ can be derived as below [2]

$$\psi = \frac{\sum_{j \in J} \sum_{k \in K} | R_j(k) |^2}{\sum_{j \in J} \sum_{q \in Q} | R_j(q) |^2} \eta - 1 \tag{3.17}$$

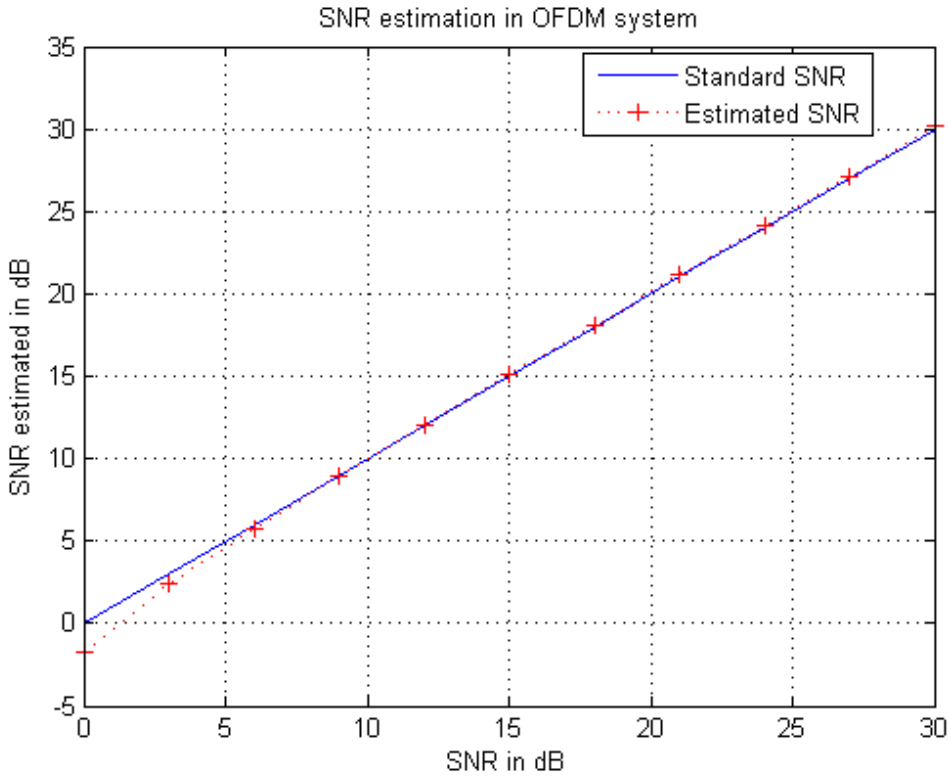

Figure 3-7 Performance of blind SNR estimation in OFDM system

3.4 Simulation of OFDM systems with estimated channel state information

Figure 3-8 presents the detailed block diagram of an OFDM system with both channel estimation (cf. Section 3.2) and SNR estimation (cf. Section 3.3), which is be used to simulate an OFDM system similar to the one implemented in hardware. The frame and symbol structures used in this system have been mentioned in Section 3.2. The performance of this imitative system model will be used for comparison with the performance of the realistic OFDM system. To ensure a fair comparison between the simulated performance and the realistic performance, we use the channel coefficient in the frequency domain collected from the experiments to estimate the number of channel multipaths and the channel coefficients, and then apply these channel state information (i.e. number of multipaths and channel coefficients) to simulate the imitative system.

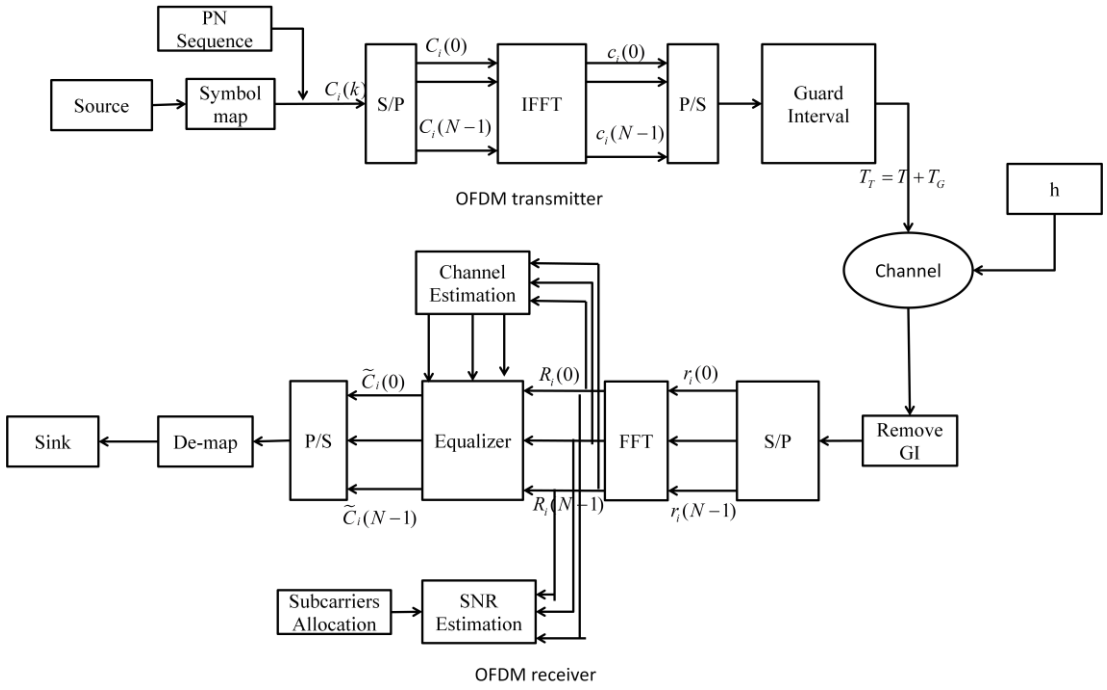

Figure 3-8 OFDM block diagram with estimated channel state information

In our experiments, channel coefficients will be estimated in the frequency domain and saved to a file. These channel coefficients are then converted to the corresponding channel impulse responses in the time domain using the IFFT operation. A threshold 10 dB is applied to the channel impulse response in the time domain to remove noise effects. Channel impulse responses which have the power lower than 10 dB compared to the maximum of all channel impulses are set to zeros. Therefore, the average number of non-zero channel coefficients in the time domain will be the average number of multipaths.

3.5 Chapter summary

In this chapter, we have discussed the channel and SNR estimation techniques for OFDM systems, which are similar to those techniques applied to the OFDM system in our hardware-based experiments performances of the channel and SNR estimation techniques have been evaluated by simulations in MATLAB. This chapter also provides the OFDM block diagram with our estimated channel state information approach. This imitative OFDM system will be simulated in MATLAB and its error performance will be used as a benchmark to compare with the error performance of the realistic OFDM system implemented in hardware.

4 IMPLEMENTATION OF OFDM SYSTEMS WITH USRP2 AND GNU RADIO

This chapter describes the implemented communications system in detail. First of all in section 4.1 the objectives of the implementation are explained. Then the requirements of these objectives are studied and compared to the resources available. This set some limitations in the systems that can be built with our resources. After that, the rest of the sections explain the software implementation of the different parts of the communication system in section 4.2 the OFDM modules will be described in general. Section 4.3 and section 4.4 will present the OFDM modulator and demodulator with more detail. Section 4.5 explains the used modules that implement the synchronization protocol. Section 4.6 explains our implementation of OFDM system. Finally, in section 4.7 the chapter summary is presented.

4.1 Objectives, resources and limitations

This project is aimed to implement the OFDM technique in an SDR communication system using GNU Radio platform and USRP2 hardware, evaluate the performance of this realistic system in different propagation environments, and compare the realistic performance of the systems with the theoretical performance. The reason for these motivations is that OFDM has been widely used in digital communication systems nowadays, such as DAB/DVB-T, WLANs, BRANs and HIPERLAN-II [2, 3], while SDR is a promising technique for multi-type, high speed wireless applications with low requirements on capability of the hardware device, low development cost and facilitation of development process. Therefore, one could evaluate the performance of OFDM system using SNR without any requirement of expensive experimental equipments. Once can also change the experimental setting without any difficulty.

The computational resources available in this project include two USRP2 devices, one Core i7 desktop computer and one Core i5 laptop computer, each computer is installed VMware Player and Ubuntu 10.04 which allows computer to run the GNU Radio software. One pair of the computers and USRP2 devices is used as a transmitter, while other pair is used as a receiver. The computational power of the two PCs has been to be an important factor that partially decides the experimental settings. The computational resources allow different configurations for OFDM

implementations, but they also have some boundaries. The computational power available limits the size of FFT/IFFT that can be computed in a limited time or some other operations that require a lot of resources in a short time and the bandwidth that can be increase total number of computation.

The USRP2 devices are equipped with a WBX daughterboard working in the frequency range from 50MHz to 2.2GHz. The connection between the desktop computer and the USRP2 is made by a Gigabit Ethernet cable. The connection between the laptop computer and USRP2 is made by a Gigabit Switch because the Ethernet card of the laptop computer does not support Gigabit Ethernet connection. As a matter of fact, the Gigabit Ethernet is also not fully used.

4.2 GNU Radio's OFDM modules

In this section the different OFDM modules in GNU Radio will be detailed. Afterward, the setup of the transceiver and the process of modulation and demodulation implemented in GNU Radio will be presented.

GNU Radio platform provides a quantity of software modules dedicated to build the OFDM modulation and demodulation. An OFDM application example has been offered in the GNU Radio's example directory which includes the *benchmark_tx.py*, *benchmark_rx.py* and optional *benchmark_add_channel.py* files. Most existing GNU Radio-based implementations of OFDM systems have been developed based on this application example. This example is a simple simulated OFDM system with both transmitter and receiver being completely created in software. The signals will be transmitted through realistic environments via USRP devices. The OFDM system also can be run completely in software. In this case, the whole transmission chain including the channel is simulated in optional file *benchmark_add_channel.py*. This example has been used as the basic source code for the implemented OFDM communication system in this project.

In the *benchmark_tx.py* and *benchmark_rx.py* files, the source code is separated in many files that are hierarchically sorted in different abstraction levels. The topmost levels set the most parameters such as the total amount of data to be sent, the size of packets, digital modulation, transmitter gain, FFT size, number of occupied tones and number of cyclic prefix in one OFDM symbol. However, only amount of data to be

sent and the size of the packets are used at the upmost abstraction level, while as in the hierarchy more specific parameters are sent to lower abstraction levels and processed later. Figure 4-1 shows the most important blocks and the connections in the hierarchy that will be explained in this section. The coloured boxes represent source code files. The ones painted light blue are written in Python and the ones painted dark blue are written in C++. The white boxes are instances of the modules that are used in their corresponding file.

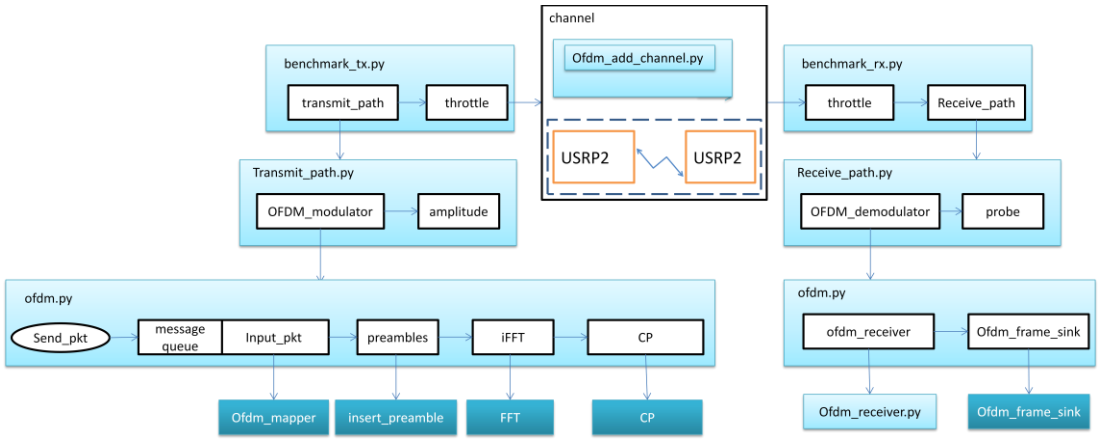

Figure 4-1 OFDM system's module hierarchy

Topmost level: The topmost level of the hierarchy shown in Figure 4-1corresponds to the Python files *benchmark_tx.py* and *benchmark_rx.py*. These are the executable files to start the transmitter and receiver. As shown in Figure 4-1, *benchmark_tx.py* includes the example of a transmit path and *benchmark_rx.py* includes the example of a receive path. The whole OFDM system can run entirely in software if in *benchmark_tx.py* the frequency parameter is not set and the optional output file parameter *--to-file=TO_FILE*, where *TO_FILE* is the file direction, is set. This corresponds to the original source code of the OFDM system that does not transmit through a realistic environment, i.e. it does not deploy the USRPs. A simulated channel to add the state channel information into modulated signals can be created by the executable file *benchmark_add_channel.py*. The '*throttle*' module controls the speed of the transmission by configuring the maximum available sample rate.

The whole communication system can be seen as an abstraction defined in *benchmark_tx.py*, *benchmark_rx.py* and *benchmark_add_channel.py*. The acquisition of the information to be sent happens directly within the python code of

benchmark_tx.py. Test data can be created by command lines or from file. The data fill the packets that will be transmitted through the system. This operation does not require a lot of computational power, thus a dedicated C++ module is not necessary. The creation of the test data is made in python and inserted in the module chain through a send function *send_pkt()*.

The channel in our experiments used is the air, since the signal is going to be sent from and received by the USRP antennas. USRPs are placed in a laboratory room, thus the channel would have multipath with a very short time delay. Additionally, the USRPs are stationary, thus the Doppler effect does not exist. The knowledge of the channel and some notions of the values of its parameters are very useful for setting the parameters of the realistic system and simulated system. In the original version of *benchmark_add_channel.py* there is a module called *'channel'* that simulates a channel with some parameters such as SNR, frequency deviation and phase deviation. However, it did not simulate multipath channel. The multipath effect can be simulated simply by FIR-filter where the complex filter coefficients can be estimated from practical environment or just taken from a channel model such as Rician and Rayleigh channel model. The signal from an input port is convey to the corresponding GNU Radio block *gr.fir_fillter_ccc*.

These parameters, which can be configured in the topmost level of the file hierarchy are related to different modules existing in python files *benchmark_tx.py and benchmark_rx.py*. For example, packet size parameter is related formation of the data packets module; maximum sample rate is related to the throttle and *uhd_transmitter* modules. A part from configuration at topmost level, total number of Megabytes transmitted parameter is set. This parameter is a global execution of the programme parameter and not directly dedicated to any single module. The rest of the configurable parameters of the system are defined in deeper levels of the hierarchy. At the moment of the execution of the programme (by running the *benchmark_tx.py* file ,then the *benchmark_add_channel.py* and finally *benchmark_rx.py* file if the system runs completely in software, or by running *benchmark_tx.py* and *benchmark_rx.py* at the same time if the system runs in realistic environment through the USRP hardware), the other parameters that belong to other modules will also be set at the topmost level, and they will be sent through the hierarchy until they reach

their target module. This makes the execution much simpler, as all the configurable parameters can be set at the same time and they will find their target destination automatically.

Below the topmost level: The next level in the hierarchy is also consists entirely of python files. In this level, the two relevant files are *transmit_path.py* and *receive_path.py*. The *transmit_path.py* file simply defines an OFDM modulator and provides two new parameters. One parameter is the number of messages that the modulator should buffer. The other is called 'padding for USRP', which makes sure that the data that is sent to the USRP is multiple of 128 samples in order to the USRP process the data correctly. The *receive_path.py* file is very similar to *transmit_path.py*. It defines the OFDM demodulator and the module called '*probe*' that is used for detecting if there is a on- going transmission.

Lowest level: The lowest level in the hierarchy is entirely in the python file *ofdm.py*. In this level, an abstraction of OFDM modulation and OFDM demodulation are built based on the instances of GNU Radio's modules which are written in C++. The modulator and demodulator in this level will be explained in the next two subsections.

4.3 Implementation of the OFDM modulator

The OFDM modulator is built in the python file *ofdm.py* together with the OFDM demodulator. It interconnects blocks, which are written in C++, via virtual connections as shown in Figure 4-1. In definition of the modulator in the python file, OFDM modulator has no input port and one output port. This is because the modulator includes a send function *send_pkt*. The function will take a payload parameter that is information needs to be sent and send it to the first module *pkt_input* of its module chain. The *send_pkt* function uses a common method to insert data in the module chain. In this case the module chain will be the one in the modulator itself. The *send_pkt* function receives the payload data that is created at topmost level. In the *sent_pkt* function, this payload data is converted into a data packet by calling a function named *make_packet* from the Python module *ofdm_packet_utils*. The *make_packet* function adds the Cyclic Redundancy Check (CRC) and the header into the payload data that will be used at the receiver to verify the validity of the received frame. The *send_pkt* function then calls the *insert_tail*

function, which belongs to the message queue of *pkt_input* module. The *insert_tail* function puts the transmitted data, including CRC and the header, at the end of this queue of messages, and the module itself will access to this data automatically. The access method suggests that in the system there are some buffers, which play a critical role in certain parts of the development of the prototype. The first module in the OFDM modulator is *pkt_input* module. It is the first C++ module that appears in the system and it is defined in the C++ file *gr_ofdm_mapper_bcv.cc*. This file is dedicated to OFDM modulations, and it takes a stream of bytes and maps it to a vector of complex constellation points suitable for the IFFT block. The *pkt_input* module has two different outputs. One output is the actual data, which output in a vector of the IFFT size, and the other output is a vector of characters, which contains one character for each output OFDM symbol. This character would be '1' if the OFDM symbol is the first symbol of the frame, and '0' otherwise. The second output indicates the preambles module the start of preamble, thereby the receiver knows where the beginning of each frames is.

The next module in *ofdm.py* is the *preamble* written in the C++ file *gr_ofdm_insert_preambles.cc*. This module adds one preamble to each OFDM frame to be sent. The preamble is a sequence of known symbols which are randomly created ±1s and are stored in the file *ofdm.py*. Both of the inputs of this module are fed by the outputs of the *pkt_input* module. The first input port contains OFDM symbols from the first output of the *pkt_input* module. The second input receives the characters from the second output of the *pkt_input* module that marks the beginning of frames. In this case, the OFDM data symbols are hold in a buffer; it puts out a preamble and then the buffered data symbols. This *preamble* module has two outputs. The first output is a vector of complex symbols having the length of the FFT size and including the preamble symbols and the payload symbols. The second output is stream of characters. This stream indicates whether the corresponding symbol on the first output is the first symbol of the packet (the first symbol of preamble). The character is '1' if the corresponding symbol is the first symbol and is '0' if otherwise.

Following the preamble module is the inverse Fourier transformation carried out by the IFFT written in the C++ file *gr_fft_vcc.cc*. This module takes a vector of complex

values from the output of the preamble module and computes the IFFT. This module can also be used for the computation of FFT, simply by editing its parameters. It also allows us to set other parameters such as the FFT/IFFT window or if the FFT should be shifted.

After the Fourier transformation, cyclic prefix must be added to the OFDM signal in order to against multipath effects. This is done by the next module *cp_adder* in the modulator. This module is defined in C++ in the file *gr_ofdm_cyclic_prefixer.cc*. It takes the OFDM symbol from its input port, copies a number of last symbols (the number of symbols to copy is specified by the *cp_size* parameter) and then inserts these symbols at the beginning of the OFDM symbol. Therefore, the symbol size at its output is the sum of the size of the input symbol and the size of the cyclic prefix.

Finally, the OFDM symbols are amplified by a multiplication with a constant value. This is done by the GNU block *gr_multiply_const_cc* and it is defined in the python file *gnu_core_genneral.py*. The OFDM signal is now ready to be sent to the wireless channel in the topmost layer of the application, which will be the air in this case, through the USRP2.

4.4 Implementation of OFDM demodulator

The OFDM demodulator is made of many C++ and Python files. Each files plays a specific role in the complete demodulation process. This section explains how all the files are joined and what their jobs are.

Similar to the OFDM modulator, the OFDM demodulator is also defined in the file *ofdm.py*. There are both input and output in the demodulator. Its input, in most cases, is fed by the channel output. Its output is the demodulated signal. In addition, the commonly used output mechanism is the parallel mechanism to the *send_pkt* function that was explained in Section 4.3. The output data is sent to a handler via the *callback* function. There is a *watcher* function, which is defined in *queue_watcher_thread* class. It runs in a separate thread to monitor the queue of demodulated data packets. When a new packet enters this queue, the function takes the packet and checks the correctness of the information in the packet by checking the CRC code. Afterwards the previously mentioned *callback* function is called with

the payload data as a parameter in the *ofdm.py* file and is executed in the topmost abstraction level of the hierarchy.

As opposed to the OFDM modulator, the demodulator has one extra layer of Python files. The demodulator structure includes two main parts as shown in Figure 4-1, corresponding to two blocks *ofdm_receiver* and *ofdm_frame_sink* which are defined in the python file *ofdm.py*. The first block *ofdm_receiver* performs the synchronization and equalization of the CRC signal, while the second block *ofdm_frame_sink*, is a state machine that demaps the symbols into bits, verifies the validity of the synchronized frames and sends the verified frames to a topmost layer by adding them to the queue of received data packets.

The *ofdm_receiver* module is defined in the Python file *ofdm_receiver.py* that includes a number of modules, such as low pass channel filter, time and frequency synchronisation, FFT and frame acquisition. Figure 4-2 shows the block diagram of the *ofdm_receiver* module. The dashed arrows in the figure correspond to the signal follows, while the solid arrows correspond to other types of data that are not the actual OFDM signal, such as frequency offset and synchronisation signals.

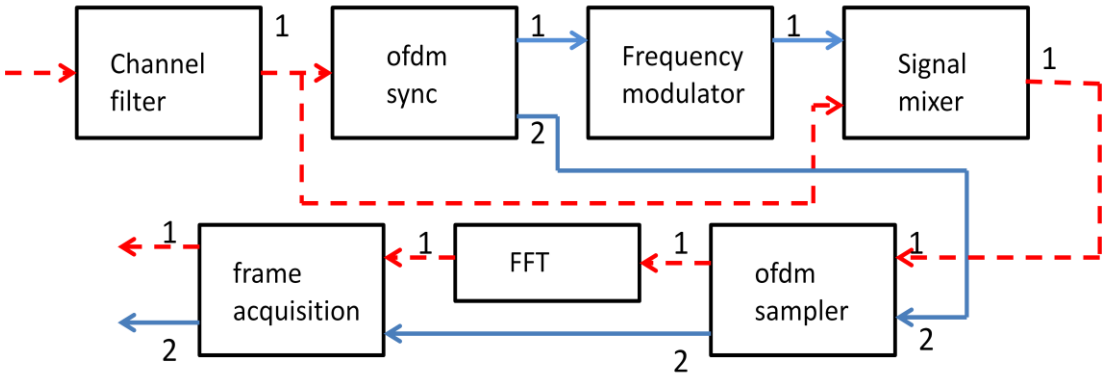

Figure 4-2 Block diagram of *ofdm_receiver* module

The first block of the receiver is a simple Fourier filter, which filters the signal received from the receive antenna. It takes the bandwidth at its output corresponding to the bandwidth of occupied tones containing actual data. The number of occupied tones is usually less than the FFT size. In our implementation, 40 subcarriers are used to carry the actual data, while the FFT size is 64. Once the signal has been filtered, it is fed into the synchronization block implemented in python file

ofdm_sync_pn.py. This synchronization block finds the right frequency offset and the start of the received frames. The frequency offset is fed to the frequency modulator to generate a signal proportional to the frequency error of the synchronisation block. The signal from the channel filter block is mixed with the signal coming from the frequency modulator to rotate the signal from the filter back. The synchronized signal in the form of vectors of the FFT size is fed to the FFT module that transfers the received signal into the frequency domain. The output signal includes data subcarriers. Each subcarrier contains information in its phase according to the digital modulation applied to that subcarrier. In our experiments, all data subcarriers are modulated with either BPSK or QPSK modulation. The last module of the *ofdm_receiver* is the frame acquisition module which takes care of finding the beginning of frames and equalizing each subcarrier. This module is defined in the C++ file *digital_ofdm_frame_acquisition.cc*.

The second module of the OFDM demodulator is the *frame_sink* module which is a state machine defined in C++. This state machine has three states. The first state is *"sync search"*, where the algorithm looks for the flag in the second input of *frame sink* module indicating the start of the frame. It corresponds to receiving the preamble of the frame. When the "*sync search*" state is found, the state machine will move to the "*have sync*" state. The *frame_sink* module will let the preamble go through. Afterwards the algorithm will demap the symbols and check the bytes corresponding to the header of the data. The header is built in a way that the first half is exactly the same as the last half, thus facilitating the validation of the header. If the header is correct, the state machine will move to the next state, namely "*have header*" state, where the algorithm will demap the rest of the frame and insert the resulting data bits in the output queue. As explained previously, that queue is monitored by a thread which will take the data, validate it and send the results to the upper layers of the system. Throughout the duration of the "*have sync*" and "*have header*" states, the algorithm will constantly look for the beginning of frame flags in its input port. In these state, the frame flag should be zeros, if it finds frame flag, it is an error. Therefore, it will reset the state machine to the "*sync search*" state. Figure 4-3 shows the behaviour of the state machine in the *frame_sink* module.

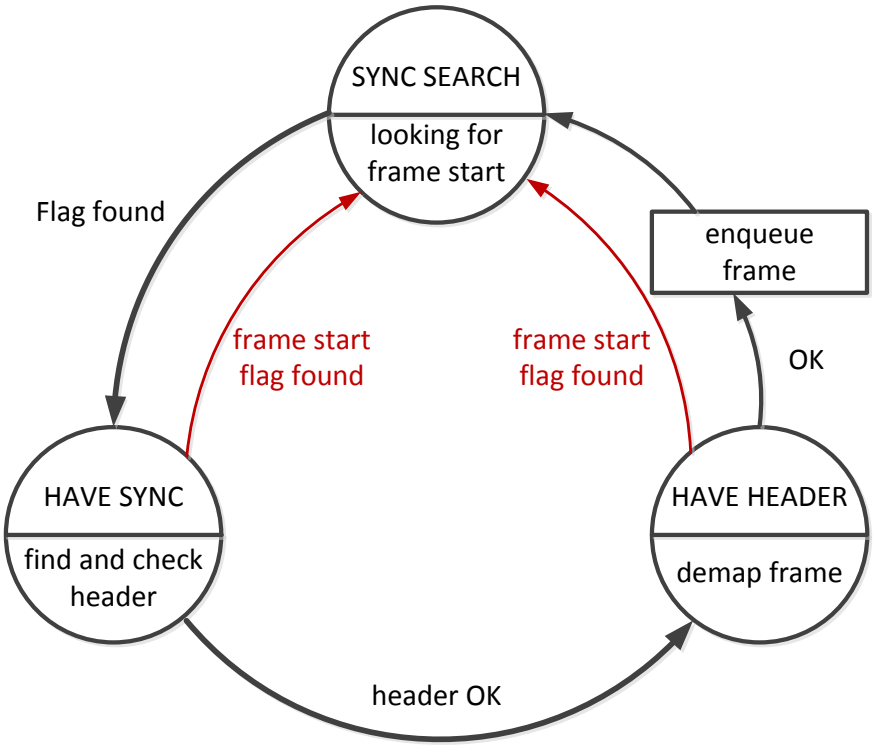

Figure 4-3 Block diagram of *frame_sink* module

4.5 Synchronisation

GNU Radio implements different synchronization methods for OFDM systems. The first method is the Maximum Likelihood (ML) synchronization. The second method is based on the correlation of PN sequences. The last method is called the fixed synchronization. It is only used for simulations where all channel state information is known at the receiver. Therefore, only ML and PN synchronisation methods are the good choices to use in realistic propagation environments. In this project, for simplicity, the PN synchronisation is chosen for our experiments.

4.5.1 Time synchronisation and fine frequency offset with pseudorandom noise synchronisation

In this thesis, the PN synchronization method has been chosen for the prototype. The synchronization module takes place in the C++ file *ofdm_sync_pn.cc*. The method was first proposed in [37] by Schmidl and Cox. It is easily seen in Figure 4-4 that the method includes two main parts. The first part is time synchronisation to find the start of each frame. It can be simply and precisely performed by observing the signal

energy, because the OFDM frames are differentiate by null symbols where the signal is zero.

To find null symbols, a FIR-filter with its taps to be set to one is applied to calculate a moving sum of the length T_{null} =32. The output of this FIR filter could be expressed as follows

$$y(n) = \sum_{k=0}^{T_{null}-1} x(n-k) \tag{4.1}$$

The signal is then multiply with -1 and the result signal is fed to a peak detector from GNU Radio *gr.peak_detector_fb*. Whenever a preset threshold in the peak detector is exceeded, it begins to search for a maximum until the signal power is lower than the threshold again.

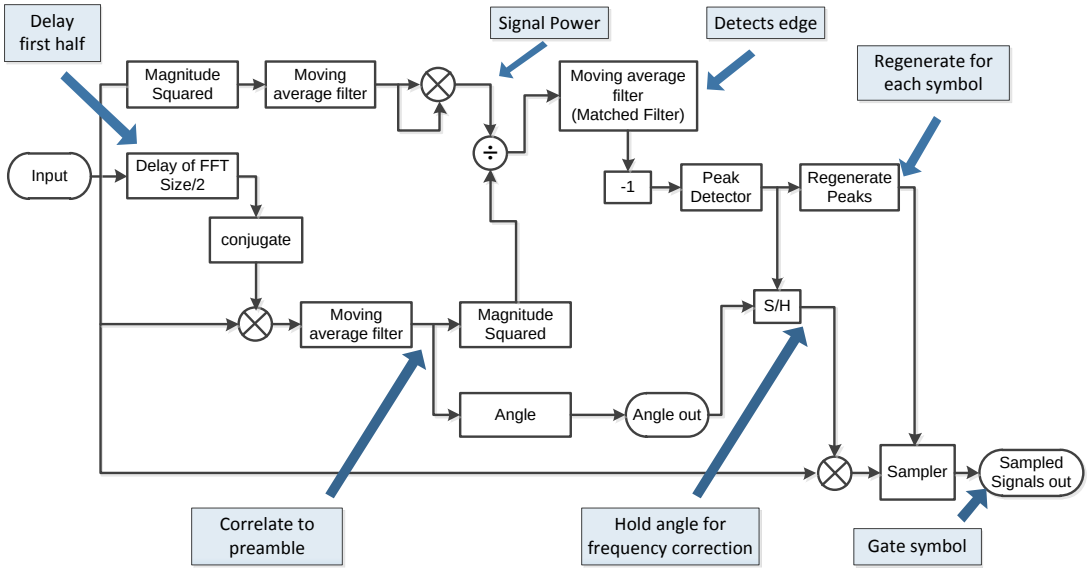

Figure 4-4 PN synchronisation flow graph

The second part is to find the fine frequency offset of the frame compared to the carrier frequency. The fine frequency offset is the remaining frequency error which is smaller than the subcarrier spacing. The fine frequency offset actually has two separate effects. One effect is that the amplitude of each OFDM symbol in time domain is reduced and the phase is shifted. Another effect is that it introduces Inter Carrier Inference (ICI) and ruin the orthogonal property among subcarriers. Thus, the accuracy of fine frequency is significant, especially, when the signal transmitted with

small subcarrier spacing and at low SNR. Schmidl and Cox [3] shown that the fine frequency offset can be estimated as

$$\Delta f = \frac{\angle P(d^*)}{\pi T} \tag{4.2}$$

where $P(d) = \sum_{k=0}^{N/2-1} r(k+d)r(k+d+N/2)$, T is duration of each OFDM symbol, N is FFT size and d^* is the optimal value to maximise the ratio $\frac{P(d)}{R(d)}$ with $R(d) = \sum_{k=0}^{N/2-1} |r(d+k+N/2)^2|$.

After the frequency offset is estimated, the signal can be corrected by multiplying the incoming signal with $\exp(-j\pi t_{\triangle} f / T)$.

4.5.2 Coarse carrier frequency synchronisation

The carrier frequency offset is frequency error in multiples of the subcarrier spacing. At this point of the coarse carrier frequency synchronisation, the fine frequency synchronisation has already been performed. Therefore, after the coarse carrier frequency synchronisation module the total frequency offset should be close to zero.

The coarse carrier frequency synchronisation must be completed with high accuracy. Otherwise, the coarse carrier frequency offset could move the subcarriers to the wrong FFT bins. Thus, the received data subcarriers could be interpreted in the wrong received OFDM symbols. Even if the wrong coarse carrier frequency offset makes the subcarriers shifts one subcarrier spacing, it could lead to a complete loss of information. The coarse carrier frequency synchronisation is performed easily after the FFT by observing at the energy of the FFT signals. The coarse carrier frequency offset estimation is therefore

$$f_c = \arg\max_n \sum_{0 \le i \le K, i \ne K/2} |X[i+n]|^2 \; n \subseteq [0, N-K] \tag{4.3}$$

where $X[i]$ is the *i*-th element in the FFT vector, K is the number of occupied tones (subcarriers used to transmit real data) and N is the FFT length.

The coarse carrier frequency synchronisation module is implemented in the C++ file *digital_ofdm_frame_acquisition.cc*. In this file, the unused subcarriers are removed and its output is a vector of K (the number of occupied tones) symbols.

4.6 Implementation of OFDM systems with GNU Radio platform

In the previous sections, the OFDM modules in GNU Radio, including OFDM modulation and OFDM demodulation blocks, have been presented in detail. This chapter show how to modify GNU Radio and what our modifications have been done in the GNU Radio platform to experiment the OFDM transmission in different realistic propagation environments.

4.6.1 Structure of a GNU Radio module

A GNU Radio module consists of several subdirectories, including *apps*, *cmake*, *docs*, *lib*, *include*, *swig*, *python* and *grc*. Each folder consists of a *Makefile.am* file because auto tools (*autoconf*, *automake* and *libtool*) are built to automatically compile and link C++ code into python code. For these tools, the following files are included in the root folder: *AUTHORS*, *bootstrap*, *ChangeLog*, *config.guess*, *config.sub*, *configure.ac*, *Makefile.am*, *Makefile.common*, *Makefile.swig*, *Makefile.swig.gen.t*, *NEWS*, *README*, *README.hacking*, *version.sh*.

- The *apps/* sub-direction includes complete applications (both for GRC and standalone executables) which are installed in the system alongside with the blocks.
- The *cmake/* folder consists of instruction files for auto tools and *CMake* to find the GNU Radio libraries.
- For documentation, the *docs/* folder includes some instructions to extract documentation from the C++ files and Python files and also make sure that the documentation is available as *docstrings* in Python. In addition, a custom documentation can be added here as well.
- *lib/* sub-direction includes source files (.cc and .h) for the developed blocks. Any source code written in C++ (or C, or any language that is not Python) is placed into *lib/*. C++ files usually have header files which are put into *include/* if they are to be exported, or in *lib/* if they are only relevant

during the compiling time, but are not installed after that time, such as *_impl.h* files.

- The *swig/* folder includes the files for the SWIG tool to build python interfaces for C++ classes.
- Python files will be put into *python/*. This folder consists of unit tests and parts of the Python module which are used when installing the module.
- A certain block requires the XML descriptions of the block must be added in the *grc/* subdirectory to be available in the GRC.

4.6.2 Block coding structure of a GNU Radio module

To modify a module in GNU Radio, it is important to understand the block coding structure of GNU Radio modules. For simplicity, a *gr-digital* module with the *digital_ofdm_frame_acquisition* block, which is a block given in the example OFDM system provided by GNU and will be modified in this project, is described below. The *digital_ofdm_frame_acquisition* block has three coding files as following:

- Firstly, the header file of the block is defined in */gr-digital/include/ digital_ofdm_frame_acquisition.h*, where the public and private variables are declared. The private variables are declared with prefix 'd_'. The accessors (set/get) functions are defined in this header if any private variable is needed to change and/or use outside the block.
- The second file is the source file, which is defined in */gr-digital/include/ digital_ofdm_frame_acquisition.h* and implements the actual code for *digital_ofdm_frame_acquisition* class. This file defines the *make* function for the public class as below.

```
digital_ofdm_frame_acquisition_sptr
digital_make_ofdm_frame_acquisition (
unsigned int occupied_carriers,
unsigned int fft_length,
unsigned int cplen,
const std::vector<gr_complex> &known_symbol,
unsigned int max_fft_shift_len)
{
  return gnuradio::get_initial_sptr(new
digital_ofdm_frame_acquisition (occupied_carriers,
fft_length, cplen,

known_symbol, max_fft_shift_len));
 }
```

The *digital_ofdm_frame_acquistion* class is inherited from the *gr_block* class. *gr_block* allows great flexibility to consume the input data streams and produce of the output data streams. Adroit use of *forecast()* and *consume()* functions allows variable rate blocks to be construct. The *forecast()* function tells the scheduler number of inputs of the *digital_ofdm_frame_acquisition* block are required to produce outputs of the block. It is able to design blocks that consume data at different rates on different input data streams, and generate output data streams at the same rate that is a function of the contents of the input data.

```
void
digital_ofdm_frame_acquisition::forecast (int
noutput_items, gr_vector_int &ninput_items_required)
{
  unsigned ninputs = ninput_items_required.size ();
  for (unsigned i = 0; i < ninputs; i++)
    ninput_items_required[i] = 1;
}
```

The *digital ofdm_frame_acquisition* block also defines a *general_work()* function. This function is the method doing the actual signal processing. It operates on its input to generate output streams. When the data is appropriate in the input buffers, the scheduler executes the block's *general_work()* function.

- The third file is *swig* interface file which is used for SWIG tool to compile from C++ code to Python code. Because the header file is used to describe

classes, functions and variables, we can simply include the headers in the main swig interface file.

From the block coding structure described above, block can be modified by altering the declaration in its header file and functions in the source file in C++.

4.6.3 Rebuilding, reinstalling and debugging the modified block

To apply the change in a block, we need to rebuild and reinstall the block. This can easily be done by the auto tool *CMake*.

```
# We're currently in the module's top directory

$ cd build/

$ cmake ../      # Tell CMake that all its config files are one
dir up

$ make           # And start building (should work after the
previous section)

$ make test      # Build test again if the change effects tests

$ make install   # Start to reinstall the modified module
```

In reality, when we create or change a block in GNU Radio system, we will certainly suffer some errors sooner or later. Therefore, a debugging tool is needed. The tool may help us to observe the data flowing out from the block during the run-time in order to find out the reason of the error. This is usually the case if a flow graph runs without crashing, but the final result is incorrect. The following options are easily implemented and are useful for GNU Radio users of any skill level.

- Using Quality Assurance (QA) codes

  This is the most obvious and simple tool anyone should use. The QA code should be added for every written or modified block. One can try to test as many options as possible in order to guess which might cause the trouble. One can also test each individual block to ensure that it works well. If the block still fails, there are some suggestions to find out the bugs. Firstly, we can use of the *ctest -V* command, instead of the *make test* command, to make the testing output more verbose. The second suggestion is that if it is

necessary, additional print statements in the code (*print* command in C++ code) are necessary to show the intermediate states until the tests pass.

- Using GRC and the graphical sinks

  This is a very simple solution. GNU Radio Companion provides graphical sinks which we can simply attach to our block. Among the *WX GUI Widgets* and the *QT GUI Widgets*, we easily find FFT plots, oscilloscopes, and number sinks which simply display the values of data. Depending on what kind of data we could choose an appropriate sink to monitor the ouput of our block. The graphical sink could be easily disabled or removed from our system.

- Dumping data into files between blocks

  For a more comprehensive analysis of the system data, beside GNU Radio, other tools, such as Octave, SciPy (with Matplotlib) and MATLAB, can be used to perform an off-line analysis. The simplest method is to connect a *file_sink* to the block that expectedly cause trouble. When the flow graph runs, the output data is saved into file. The data then is will be loaded to analyse with suitable tools.

4.6.4 Implementation of channel estimation

Channel estimation and equalisation are important before OFDM symbols can be demapped. These processes remove the distortion caused by frequency selective fading and time-varying fading. As mentioned in Section 3.2, the channel estimation method chosen in this project is the LS estimates for simplicity.

The channel estimation is implemented in the *gr-digital* module and C++ file *digital_ofdm_frame_acquisition.cc*. Due to the fact that coarse carrier frequency offset is also in this file, the channel estimation need to count it on the calculators. Therefore, the channel estimates in the frequency domain are described as follows

$$H_{LS}(k) = \frac{R(k + \text{coarse}\_freq)}{C(k)} \times \text{coarse}\_\text{freq}\_\text{comp}\left(\text{coarse}\_\text{freq}, 1\right) \qquad (4.4)$$

where

$$\text{coarse}\_\text{freq}\_\text{comp}\left(\text{coarse}\_\text{freq}, d\_phase\_count\right) = e^{-2\pi \text{coarse}\_freq \times \frac{cp\_length}{fft\_length} \times symbol\_count}$$

coarse_freq is the carrier frequency shift, $symbol\_count$ is the position of the

current OFDM symbol in the supper frame, $R(k)$ is received symbol at k-th subcarrier and C(k) is preamble symbol at k-th subcarrier.

```
void
digital_ofdm_frame_acquisition::calculate_equalizer(const
gr_complex *symbol, int zeros_on_left)
{
  unsigned int i=0;
  // Set first tap of equalizer
  d_hestimate[0] =
(coarse_freq_comp(d_coarse_freq,1)*symbol[zeros_on_left+d_coar
se_freq]) / d_known_symbol[0];

  // set every even tap based on known symbol
  // linearly interpolate between set carriers to set zero-
filled carriers
  for(i = 2; i < d_occupied_carriers; i+=2) {
    d_hestimate[i] =
(coarse_freq_comp(d_coarse_freq,1)*(symbol[i+zeros_on_left+d_c
oarse_freq])) / d_known_symbol[i];
    d_hestimate[i-1] = (d_hestimate[i] + d_hestimate[i-2]) /
gr_complex(2.0, 0.0);
  }

  // with even number of carriers; last equalizer tap is wrong
  if(!(d_occupied_carriers & 1)) {
    d_hestimate[d_occupied_carriers-1]=
d_hestimate[d_occupied_carriers-2];
  }
```

To collect the channel impulse response, the channel coefficients *d_hestimate* is needed to save into file. However, *digital_ofdm_frame_acquisition* block does not provide the output for channel coefficient. Therefore, it is needed to add one more output in the *digital_ofdm_frame_acquisition* block to collect channel coefficient. Currently, the block has two inputs, and two outputs. Thus, the initial function of *digital_ofdm_frame_acquisition* class has to modify as following

```
digital_ofdm_frame_acquisition::digital_ofdm_frame_acquisition
(
unsigned occupied_carriers,
unsigned int fft_length,
unsigned int cplen,
const std::vector<gr_complex> &known_symbol,
unsigned int max_fft_shift_len)
  : gr_block ("ofdm_frame_acquisition",
             gr_make_io_signature2 (2, 2,
sizeof(gr_complex)*fft_length, sizeof(char)*fft_length),
             gr_make_io_signature2 (3, 3,
sizeof(gr_complex)*occupied_carriers, sizeof(char),
sizeof(gr_complex)*occupied_carriers)),

}
```

In addition, a pointer to handle the new output need to be declared in the *general_work()* function as follows.

```
gr_complex *ch = (gr_complex *) output_items[2]; // used for
channel coefficient collection
```

and channel coefficient output is

```
for(unsigned int i = 0; i < d_occupied_carriers; i++) {
      ch[i] = d_hestimate[i];
}
```

4.6.5 Implementation of SNR estimation

The blind SNR estimation algorithm presented in Section 3.3 is implemented in this project. This algorithm estimates SNR in the frequency domain, i.e. the SNR estimation is done after FFT. Therefore, it can also be implemented in the *digital_ofdm_frame_acquisition* block. However, the OFDM system is running in real time, thus only one OFDM symbol is allowed to pass through the *digital_ofdm_frame_acquisition* block at a time for processing. Therefore, only instantaneous SNR could be estimated in the block. The average SNR is estimated at the topmost abstract level.

Because the frequency is shifted due to frequency offset, the number of zero subcarriers is not stable. Therefore, the original blind SNR estimation algorithm cannot be applied on the whole super frame or the whole one symbol in the real

testbed. Instead, the blind SNR estimation algorithm must be modified to estimate SNR based only on the preamble with only occupied tones.

The structures of frames and preambles are shown in Figure 3.3. An OFDM preamble symbol includes 12 zeros which are located at each beginning of the OFDM symbol, the 40 symbol PN sequence, which only consists of known symbols ± 1 in the even subcarriers and 0 in the odd subcarriers, and finally 12 zeros which are located at the end of OFDM symbol. Therefore, the received preamble signal in the *i*-th super frame can be expressed as below

$$R_i(k+d) = \Pr(k) \times H_i(k) \times e^{2\pi d \times n\_cp/n\_fft} + N_i(k) \text{ k=1,2,...,K} \tag{4.5}$$

where d is the coarse carrier frequency, K is the number of occupied tones, n_cp is the length of cyclic prefix, n_fft is the FFT length, $\Pr(k)$ is the *k*-th (known) symbol in the preamble, $H_i(k)$ is the channel coefficient and $N_i(k)$ is the white noise.

SNR is generally defined as

$$SNR = \frac{P}{N} = \frac{\sum X_i(k) \times H_i(k)}{\sum N_i(k)} \tag{4.6}$$

Due to the white property of noise, SNR can be estimated as

$$SNR = \frac{\sum \Pr_i(k) \times H_i(k)}{\sum N_i(k)} \quad k = 1,3,...,K-1 \tag{4.7}$$

and

$$\sum_{k=1,3,..,K-1} N_i(k) = \sum_{k=2,4,..,K} N_i(k) = \sum_{k=2,4,...,K} P_i(k) \tag{4.8}$$

Frome (4.5), (4.7) and (4.8), it is easy to draw that

$$SNR = \frac{\sum P(Y_{preamble}(k_{odd} + d))}{\sum P(Y_{preamble}(k_{even} + d))} - 1 \tag{4.9}$$

where $P(.)$ is the signal power .

To retrieve the sum of signal powers in all odd subcarriers *d_power* and noise power in all even subcarriers *d_noise_power*, two private variables with accessor function are added into the *digital_ofdm_frame_acquisition* class as below.

```
/*!
   * \brief Return an estimate of the SNR of the channel
   */
  float snr() { return d_snr_est; }
  float power() { return d_power; }
  float noise_power() { return d_noise_power; }
```

The signal power in all odd subcarrier *d_power* and noise power in all even subcarrier *d_noise_power* are calculated as follows

```
float Tu_so=0;
  float Mau_so =0;
  for(i = 2; i < d_occupied_carriers; i+=2) {
        Tu_so  = abs(symbol[i+zeros_on_left +
d_coarse_freq])*abs(symbol[i+zeros_on_left+d_coarse_freq]) +
Tu_so;
        Mau_so = abs(symbol[i+zeros_on_left + d_coarse_freq
+1])*abs(symbol[i+zeros_on_left+d_coarse_freq+1]) + Mau_so;

  }
  d_power = Tu_so;
  d_noise_power = Mau_so;
```

The average SNR is estimated in the topmost abstraction level in the Python file *benchmark_rx.py*. Total transmitted power of the preamble signals in the odd subcarriers and total noise power (i.e. total power of preamble signals received in all even subcarriers) are declared as a global variable to make it possible to change its value after each frame is consumed. They are declared in *benchmark_rx.py* as below

```
global power,noise_power
    power =0
    noise_power=0
```

Once each OFDM frame is consumed, the signal power and noise power are updated as below

```
noise_power_current =
tb.rxpath.ofdm_rx.ofdm_recv.ofdm_frame_acq.noise_power()
power_current =
tb.rxpath.ofdm_rx.ofdm_recv.ofdm_frame_acq.power()

power = power + power_current
noise_power = noise_power + noise_power_current
snr_est = 10*math.log10(power*(noise_power**(-1))-1)
```

,the *snr_est* variable is the instantaneous average SNR estimated from the start of transmission until the considered time.

4.7 Chapter summary

This chapter first analyses clearly the OFDM system example given in GNU Radio, especially the OFDM modulation, OFDM demodulation, and OFDM synchronisation blocks. To verify an OFDM system performance in practical environment, the modifications of the channel estimation and SNR estimation blocks, has been presented. In the next chapter, the performances of our implemented OFDM system in different realistic propagation environments, including laboratory rooms and corridors, with different transmitter-receiver distances and/or with different modulation schemes will be presented. These performances will be compared to the simulated performance of the corresponding OFDM systems to evaluate the differences between the theoretical performances and the realistic ones.

5 EXPERIMENTS

In the previous chapters, the both implemented OFDM and simulated OFDM systems have been presented. This chapter investigates how well our testbed performs compared to the theoretical Bit Error Rate (BER) error performance. We will refer the performance of our implemented OFDM system in the testbed to as *implemented performance*, while the performance of OFDM systems simulated in MATLAB to as *simulated performance*. The chapter first details the testbed setup, including propagation environment properties, hardware devices and prototype used in the testbed. Implemented OFDM system performances in different realistic propagation channels and with different modulation schemes is then presented. Finally, comparisons between the simulated OFDM performances and the implemented OFDM performances are discussed.

5.1 Test-bed settings

The overall system design is illustrated in Figure 5-1. The transmitter includes one Personal Computer (PC) running on the Ubuntu 10.4 operating system, which has been installed via VMware Player; one Gigabit Ethernet Switch to synchronise the 100Mbit/s port on the PC and the Gigabit port on USRP2; a USRP2 module with one WBX daughter board, which has been described in Chapter 2; and one omni-directional (whip) antenna. Receiver comprises of a PC running on the Ubuntu 10.4 operating system, a USRP2 module with one WBX daughter board, and one whip antennal. The Gigabit Ethernet Switch is used in the transmitter because the PC in the transmitter system does not support Gigabit transmission. Therefore, the Gigabit Ethernet Switch is required to communicate with the USRP2 module, which supports Gigabit Ethernet transmission.

Similarly to any other hardware connected via an Ethernet port, each of the USRP2 devices has an IP address which is by default preconfigured to 192.168.10.2/24. The GNU Radio script running on the PCs will use this IP address to identify the connected USRP2 device. Preferably, the Gigabit Ethernet interface of the PC should have some IP address in the same subnet, i.e. 192.168.10.x/24. This IP address being in the same subnet as the USRP2 device avoids the necessity to have custom defined entries in the routing table of the PC. In this project, the IP addresses of the PC in both the transmitter and receiver are set to 192.168.10.1/24.

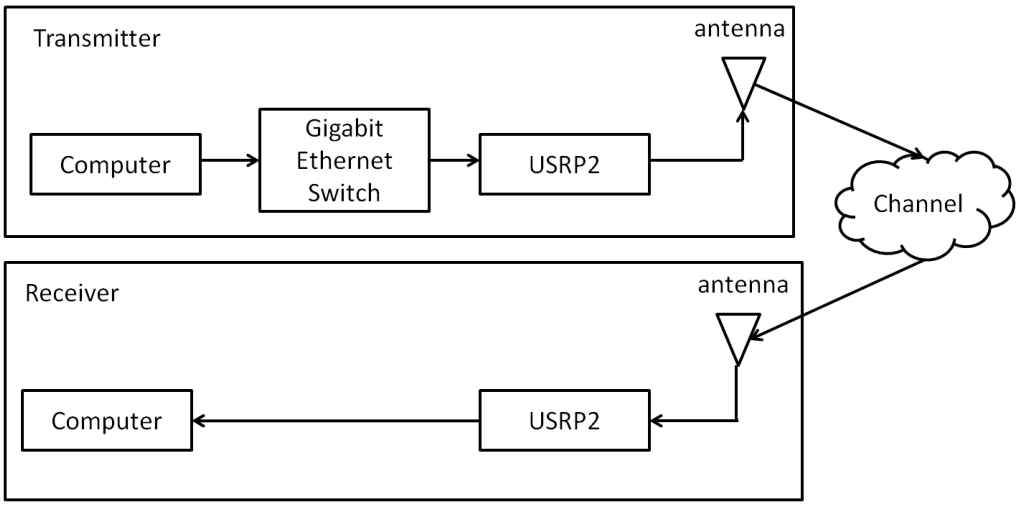

Figure 5-1 Structure overview of the testbed

As discussed in Section 2.8, the WBX daughter board has two RF interfaces, namely TX/RX and RX2. In the transmitter, only TX/RX interface can be used and it is connected to the transmit antenna for signal transmission. In the receiver, at least one of the TX/RX and RX2 interfaces should be connected to the receive antenna. In this project, RX2 interface is used to receive signals.

In this project, the SNR performance is presented as a function of the input level at the receiver as well as the receiver gain. The transmit power is adjusted by swapping the amplitude of the transmitted signal to achieve different input power levels at the receiver. The SNR can be estimated after all samples are gathered as described in Section 4.6.5. The overall receiver gain value is set through USRP Hardware Driver (UHD). It is actually set for a Low Noise Amplifier (LNA). The LNA gain can be set to a value from 0 to 30 dB. In addition, using a very high gain value can result in saturation at the receiver. Thus, we suggest the use of the receiver gain of 15 dB for the input power levels higher than -60 dBm and the gain of 30 dB for the lower input power level.

The system parameters are as follows. The carrier frequency is set to 300 MHz and the bandwidth is 500 kHz. The constellation is BPSK or QPSK. FFT size is 64 and the number of transmitted subcarriers (occupied tones) is 40.

5.2 The OFDM performance in different configurations

5.2.1 OFDM performances in laboratory without obstruction

In this section, performances of OFDM systems with different modulations in our laboratory are illustrated. The laboratory environment is portrayed in Figure 5-2. The distance between the transmitter antenna and the receiver antenna is 5 meters. There is no obstruction between the transmitter and receiver antennas.

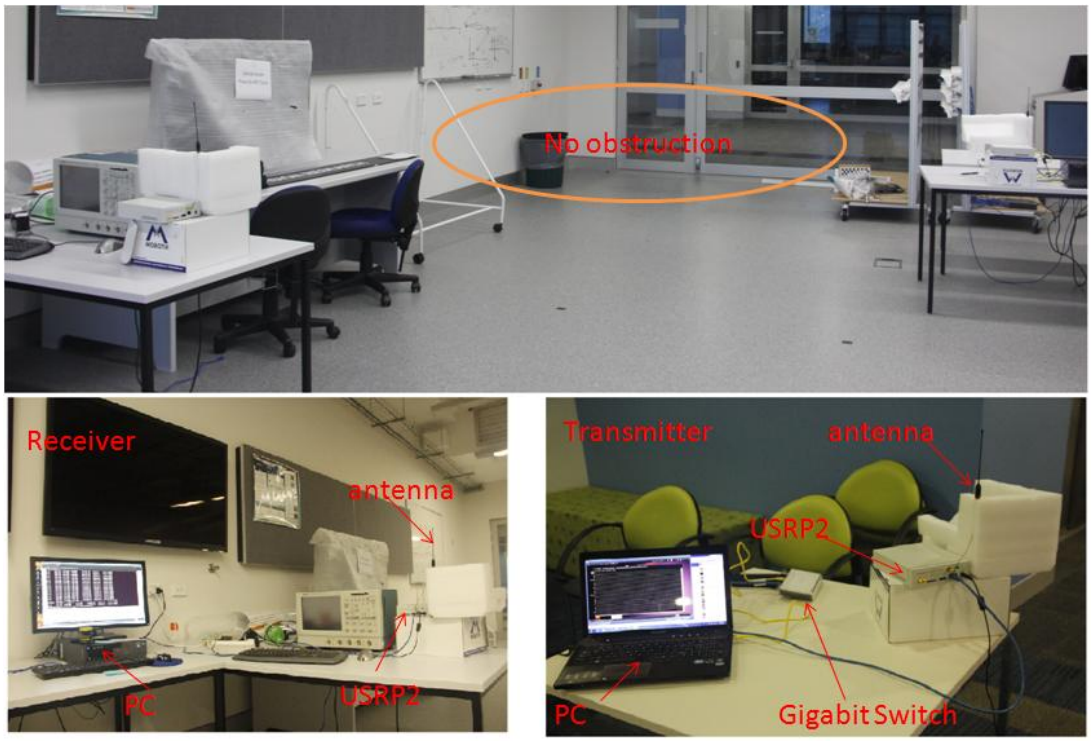

Figure 5-2 Laboratory environment without obstruction and the arrangement of transmitter and receiver in our experiments

The performances of OFDM system with BPSK and QPSK modulations in our laboratory environment without obstruction are shown in Figure 5-3. The OFDM system with BPSK performs well in the laboratory environment without obstruction. The BER performance achieves below 10^{-3} at the SNR of 19.5 dB. The BER performance of the OFDM system can be obtained via our hardware within the SNR range 8.5-21 dB, and the BER performance reaches the floor at SNR of 21 dB. The error floor issue arises due to the power saturation problem at the receiver.

The performance of the OFDM system with a QPSK constellation is worse than that with a BPSK constellation. The BER performance can only be obtained for SNR higher than 13 dB. The error floor starts to be seen at SNR of 22 dB.

Figure 5-3 also shows that the performance of OFDM systems with BPSK in the same environment is better than that with QPSK by approximate 3-4 dB. This is consistent with the simulation in

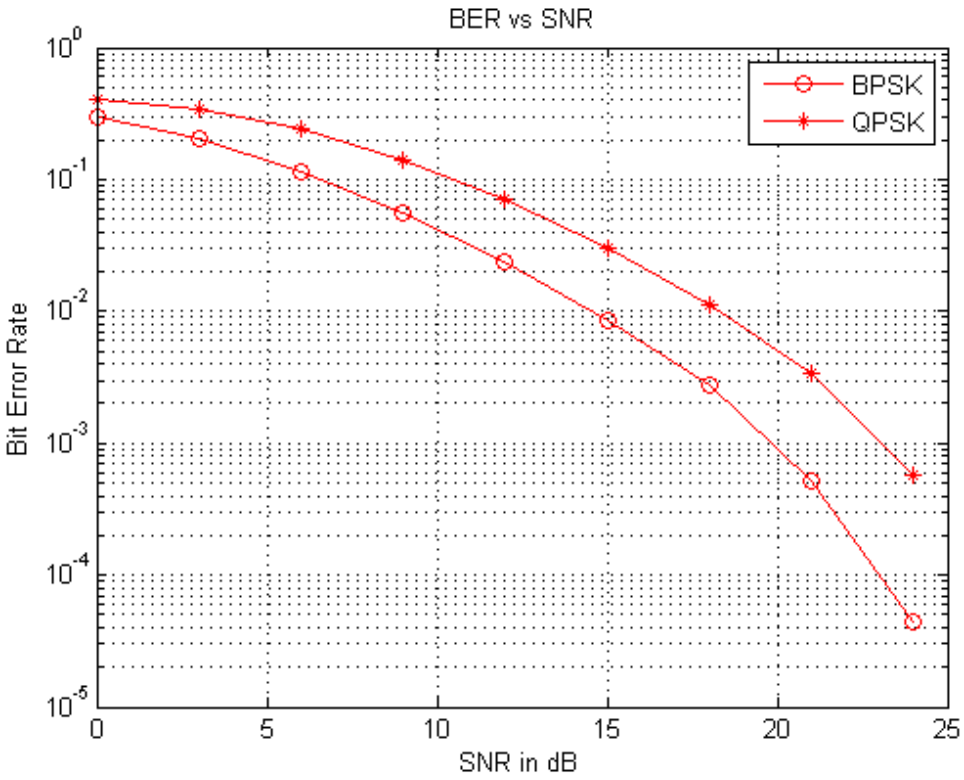

Figure 5-4 where the performance of OFDM systems with BPSK is better than that with QPSK by 3.5 dB. The OFDM systems with BPSK and QPSK constellations are simulated in MATLAB with the same channel impulse respond which collected from analysing the channel coefficients in the frequency domain.

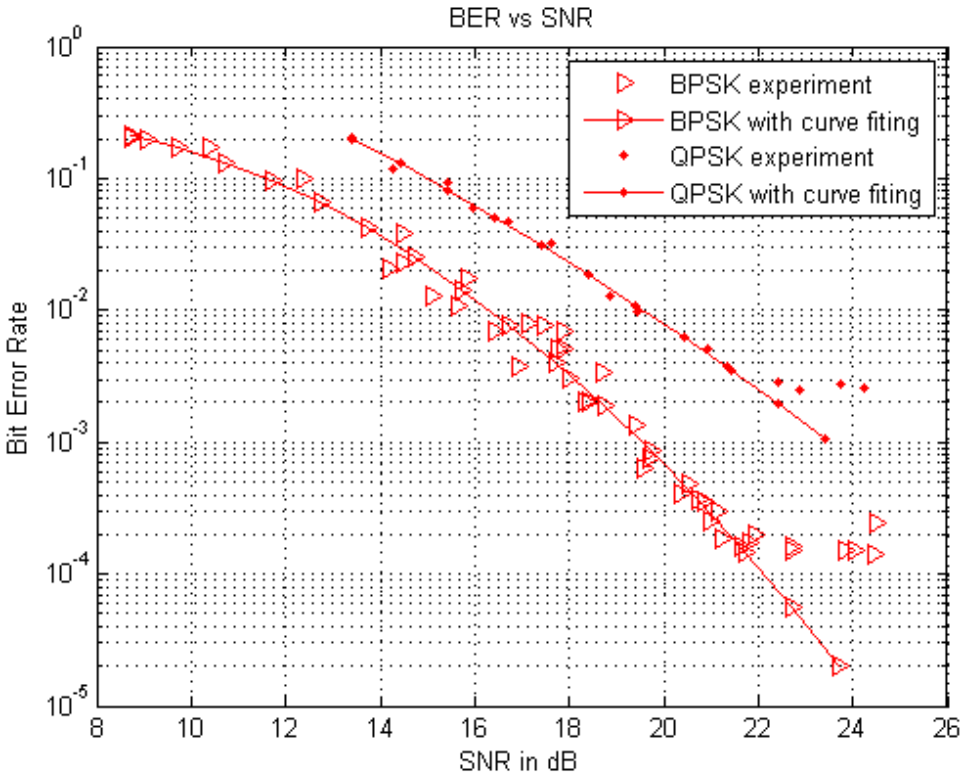

Figure 5-3 Comparison OFDM performance with BPSK and QPSK constellations in laboratory environment without obstruction

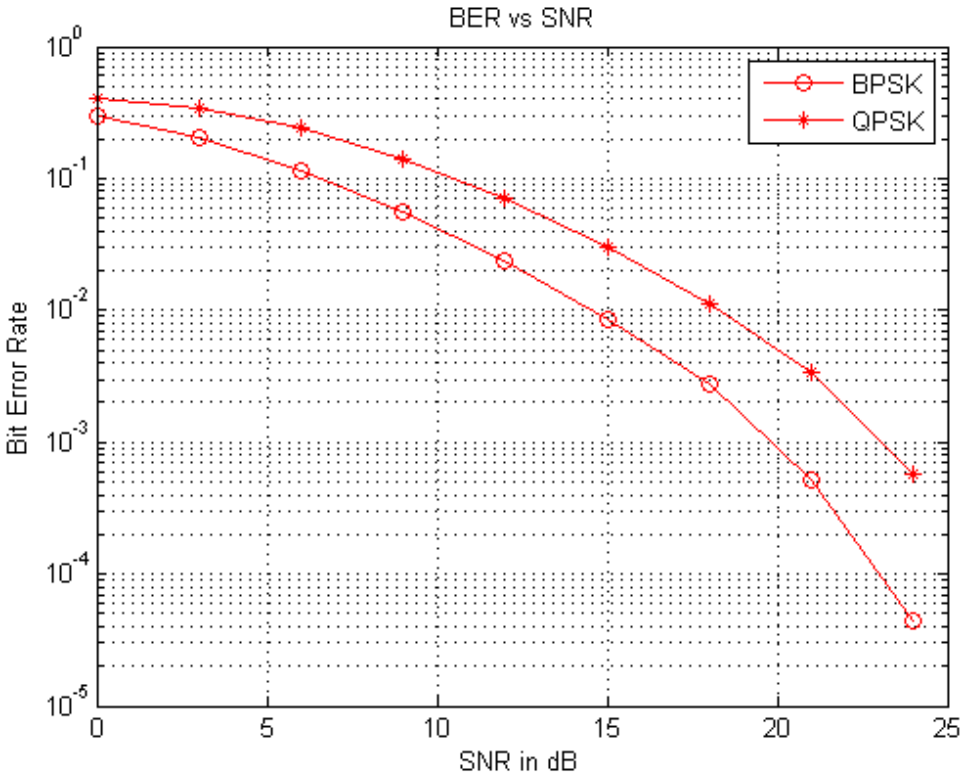

Figure 5-4 Comparison performance of OFDM system with BPSK and QPSK in simulation

5.2.2 OFDM performances in laboratory environment with obstruction

In this section, the implemented OFDM performances are illustrated with BPSK and QPSK constellations in our laboratory environment with obstruction between transmit and receive antennas. The laboratory environment is room 226 building 6 University of Wollongong and portrayed in Figure 5-5.

Figure 5-5 The laboratory environment with obstruction

The location and distance of two antennas are the same as those in the previous experiments mentioned in Subsection 5.2.1. The only difference is that there is one metal frame used as obstruction between two antennas in order not to allow wireless signal to pass directly from the transmit antenna to the receive antenna. The obstruction reduces the received signal power at the receiver. Therefore, to reach the same SNR values at the receiver, the transmitted power at the transmitter need to be increased. This can be done by increasing the amplitude of transmitted signals.

The performance of OFDM systems with BPSK and QPSK modulations in our laboratory environment with the obstruction between two antennas is shown in Figure 5-6 .

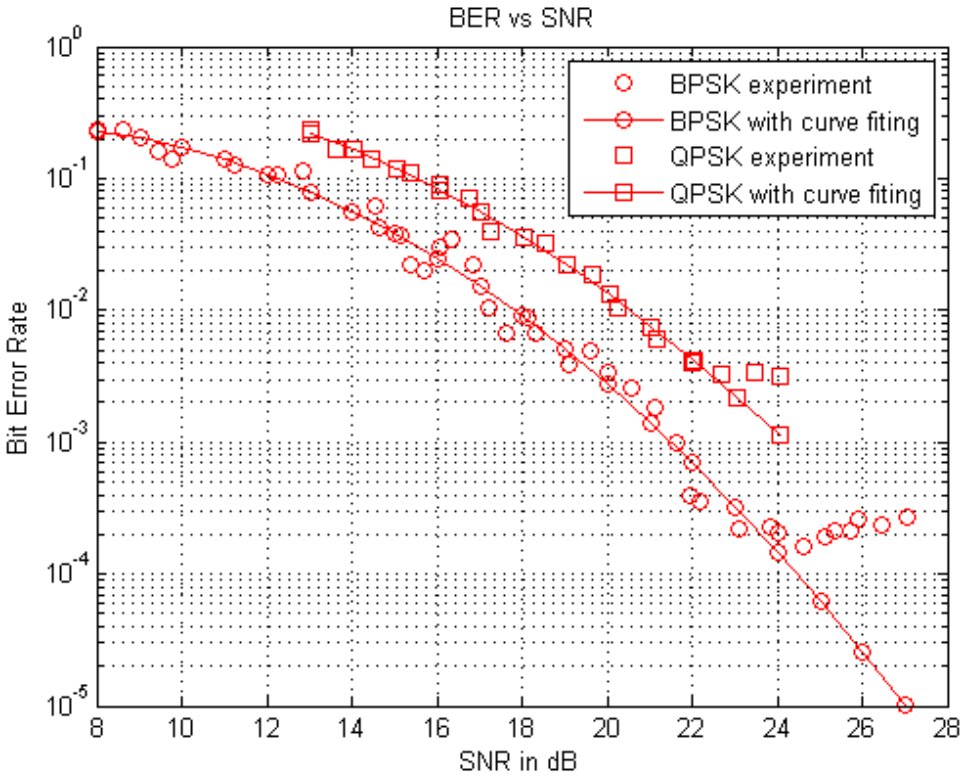

Figure 5-6 Performance of OFDM systems with BPSK and QPSK constellations in the laboratory environment with obstruction between two antennas

Similar to the case of no obstruction, the OFDM system with BPSK performs relatively well in the laboratory environment even with the obstruction between the two antennas. The BER performance reaches below 10^{-3} at SNR of 19 dB of SNR. The system BER performance can be obtained with our hardware for the SNR being

higher than 7 dB, and reaches the BER floor at SNR of 23 dB due to the saturation problem at the receiver.

The performance of OFDM systems with a QPSK constellation is worse than that with BPSK. In this scenario, the BER performance can only be obtained for SNR being higher than 13 dB. The error floor occurs at SNR of 23 dB.

In addition, the figure Figure 5-6 shows that the OFDM system with BPSK performs better than that with QPSK by about 3-4 dB in the same propagation environment. This is relatively consistent with the simulation where the performance gain is to be approximately 3.5 dB. The simulation of OFDM systems with BPSK and QPSK is given in Figure 5-7.

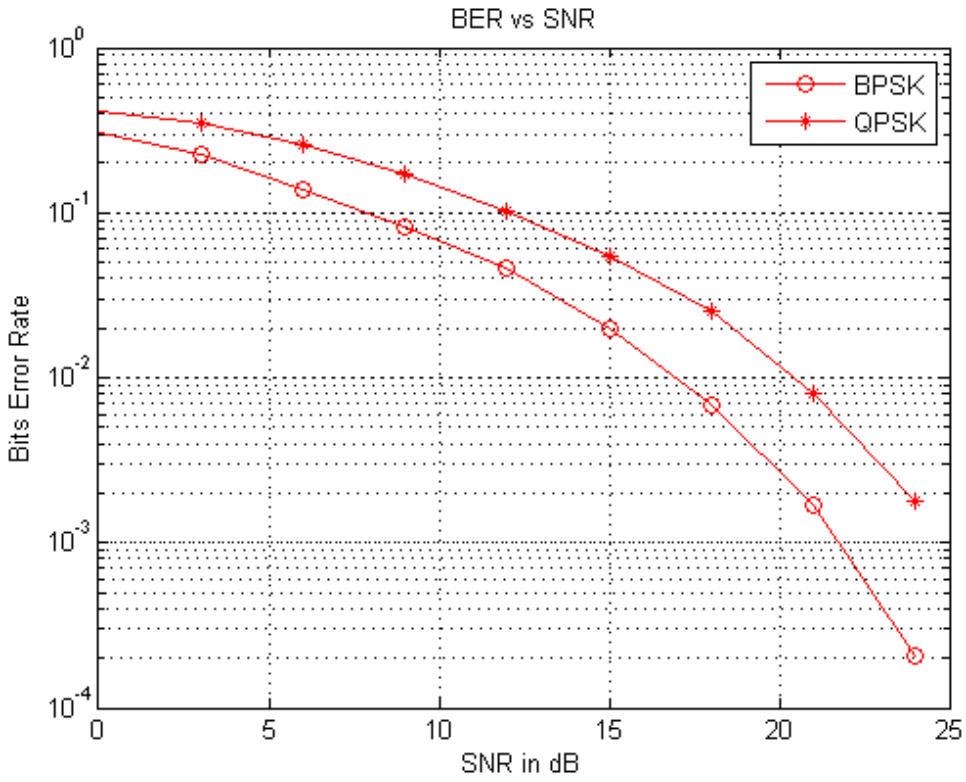

Figure 5-7 Performance of OFDM system with BPSK versus QPSK in simulation

Comparing Figure 5-3 with Figure 5-6, one can easily realise that the obstruction cause the BER performance degradation by approximate 0.5-2 dB in both BPSK and QPSK modulations. For instance, in the case of BPSK, SNR should be 16 dB in order to achieve BER = 10^{-2} when no obstruction is present, and should be 18 dB

when obstruction is present. This observation is discussed further later in Section 5.2.5.

5.2.3 OFDM performances in corridor environment without obstruction

In this section, the implemented OFDM system is experimented in the corridor propagation environment outside of our laboratory without obstruction. The corridor environment is the corridor of floor 2 Building 6 in the University of Wollongong and portrayed in Figure 5-8.

Figure 5-8 Corridor environment used in the experiments

In this experiment, the distance between two antennas is 30 meters. The same settings for the OFDM system as mentioned in all previous experiments are remained

for this experiment. The performance of the implemented OFDM system in the corridor environment without obstruction is shown in Figure 5-9.

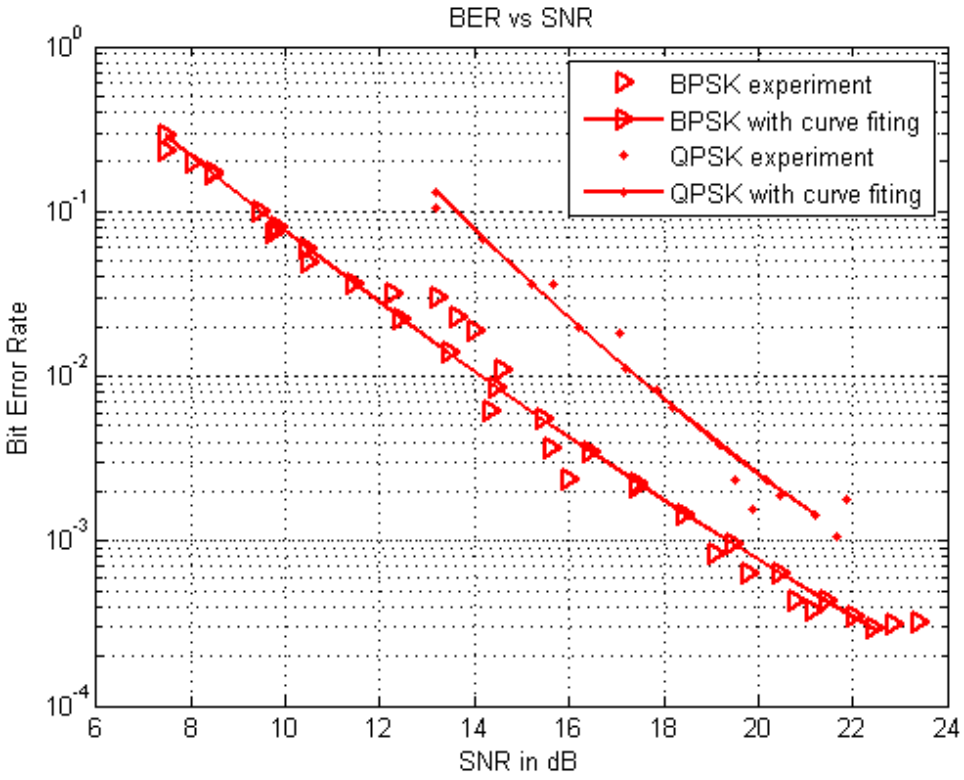

Figure 5-9 Comparison between performances of implemented OFDM with BPSK and QPSK in corridor environment without obstruction

The Figure 5-9 shows that the BER performance reaches below 10^{-3} at SNR = 19.5 dB in the case of BPSK. The system BER performance can only be obtained for SNR higher than 7 dB due to the limitation of our hardware. The BER floor starts to be seen at SNR of 23 dB, which may be because of the saturation issue at the receiver.

For the QPSK modulation, the BER performance can only obtained for SNR higher than 13 dB. The BER curve experiences the error floor at SNR = 22 dB. The OFDM performance with a QPSK constellation is worse than to that with the BPSK constellation in the same environment and with the same setup by about 3-3.5 dB, which is relatively close to the simulation gap of 3.5 dB.The simulation gap of OFDM systems with BPSK and QPSK constellations with estimated channel impulse response from analysing collected channel coefficient in the time domain is shown in Figure 5-10.

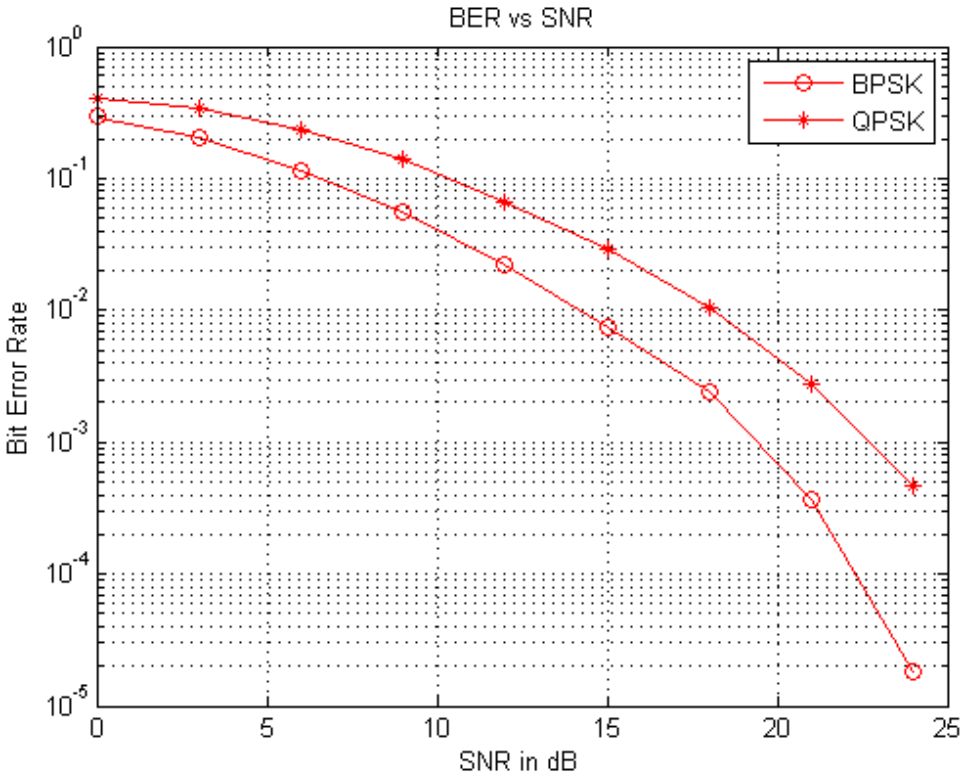

Figure 5-10 Comparison performances of OFDM with BPSK and QPSK in simulation

5.2.4 OFDM performances in corridor environment with obstruction

The corridor environment with obstruction is displayed in Figure 5-11. It is similar to the environment in Section 5.2.3. The only difference is that there is a metal frame between two antennas.

The settings of the OFDM system are kept the same as those in the previous experiment in the corridor environment without obstruction (cf. Section 5.2.3). The performance of the implemented OFDM system in the corridor environment with obstruction is illustrated in Figure 5-11. In this experiment, the BER performance can be obtained for SNR being higher than 8 dB for the BPSK case, and 12 dB for the QPSK case. The BER curves experience the error floors at SNR = 22 dB and SNR = 24 dB for the BPSK and QPSK modulations respectively due to the saturation issue of the receiver. The BER performance archive below 10^{-3} within 21 dB of SNR. The performance of OFDM system with QPSK constellation is worse than to the performance of the system with BPSK constellation in the same environment and

with the same setup. The BER performance is only obtained within 12 dB of SNR and get error floor within 24 dB of SNR.

Figure 5-11 Obstruction between two antennas in corridor environment

Similarly the OFDM performance with BPSK performs better than that with QPSK by approximate 2-3.5 dB, which are relatively close to their simulation gap of 3 dB. The simulation gap between performances of OFDM system with BPSK and QPSK constellation can be seen in Figure 5-13.

Comparing Figure 5-9 with Figure 5-12, we can realise that putting the obstruction between the transmitter and receiver antennas degrades the OFDM performance by about 0.5-2 dB for both BPSK and QPSK cases. This value turns out to be the same as the degradation due to the obstruction in the laboratory room as mentioned at the end of Section 5.2.2. Some further analysis on this observation will be provided later in Section 5.2.5 of this thesis.

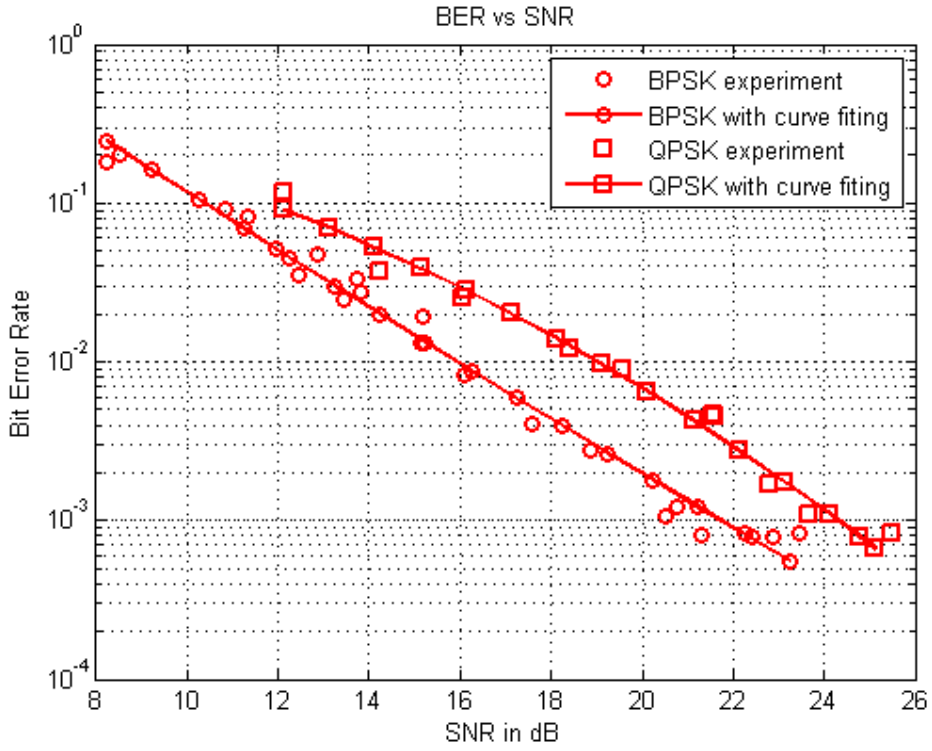

Figure 5-12 Performances of OFDM systems with BPSK and QPSK constellations in corridor environment with obstruction

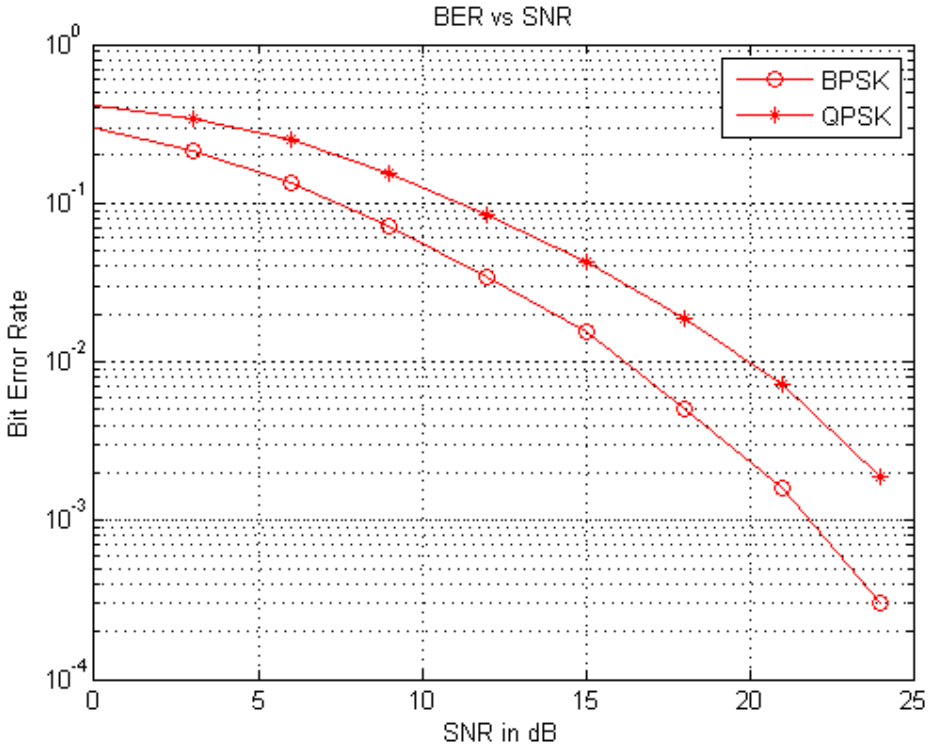

Figure 5-13 Comparison performances of OFDM with BPSK and QPSK in simulation

5.2.5 Performance with obstruction versus performance without obstruction

In this section, the performance of the implemented OFDM system is compared to that of the simulated system in the same propagation conditions. We will focus on two types of fading channels, namely Rician and Rayleigh fading channels. In Chapter 2, Rician and Rayleigh fading channels were briefly discussed. Rician environment is the environment where a LOS path exists between the transmitter and the receiver, while no LOS path exists in the Rayleigh environment. Therefore, it is reasonable to assume that the laboratory and corridor environments without obstructions are Rician fading channels, while the laboratory and corridor environments with obstructions are Rayleigh channels. Figure 5-14 presents performances of the implemented OFDM system (with BPSK modulation) in the laboratory room with and without obstruction.

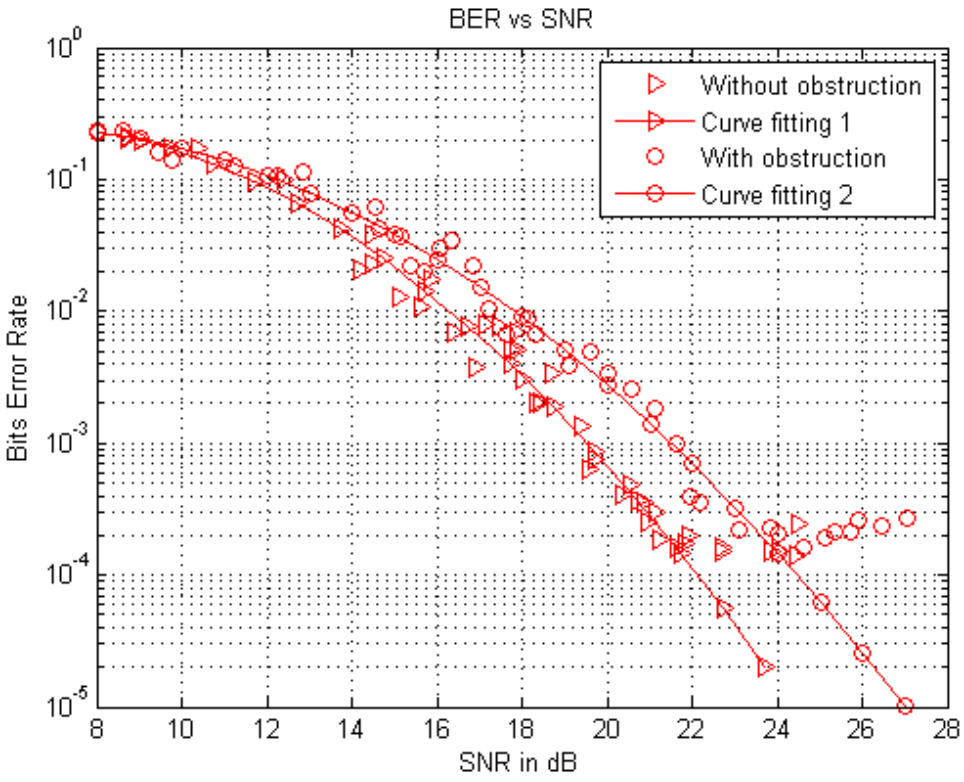

Figure 5-14 Comparison between performances of the implemented OFDM system in our laboratory with obstruction and without obstruction

The results shows that the performance of the implemented OFDM system in our laboratory without obstruction (Rician channel) is better than that with obstruction

(Rayleigh channel). This observation is reasonable because the Rayleigh fading channel does not have the LOS path, thus a higher transmitted power (i.e. higher SNR) is required to achieve the same BER as the Rician fading channel. As mentioned in Section 5.2.2 and 5.2.4, the typical gap between these two BER curves is from 0.5 to 2.5 dB.

Similarly, the performance of the implemented OFDM system in the corridor environment without obstruction is better than that with obstruction. This observation for the case of BPSK modulation is shown in Figure 5-15. As discussed earlier in Section 5.2.4, the performance of OFDM systems without obstruction (Rician fading channel) is better than that with obstruction (Rayleigh fading channel) by approximate 0.5-2.5 dB.

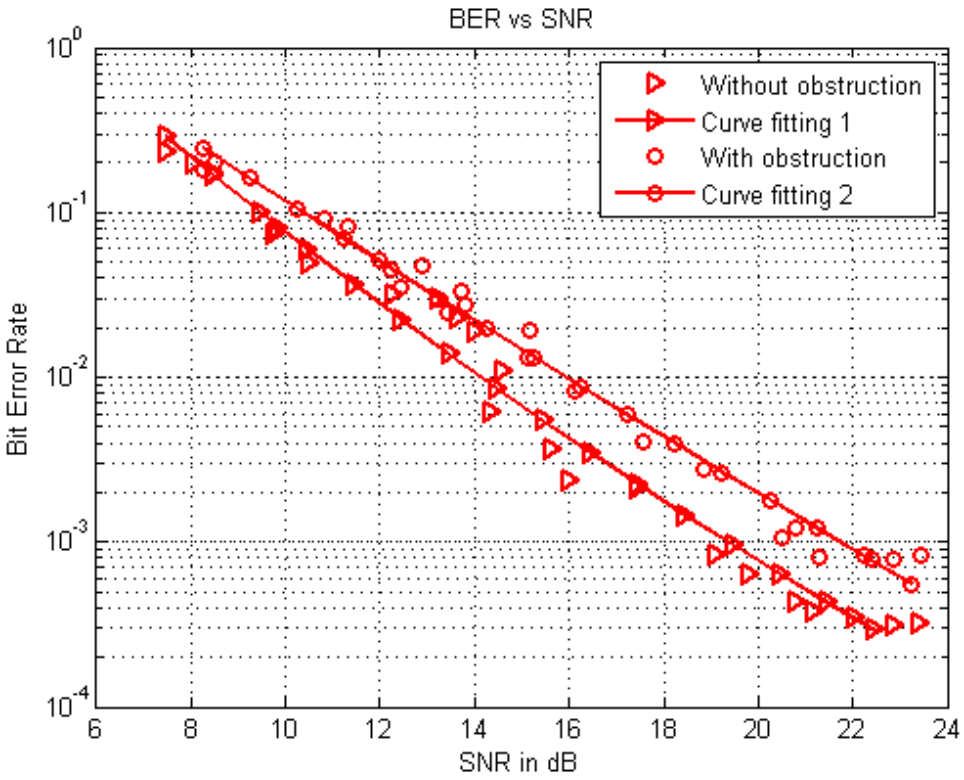

Figure 5-15 Performances of the implemented OFDM system in the corridor environment with and without obstruction

5.3 Comparison of the simulated OFDM performances and the implemented OFDM performances

The simulated performance of OFDM systems has been briefly presented in Chapter 3. This section explains how the simulated performance of OFDM systems is

obtained in detail. In particular, the channel state information collected in our experiments will be used to duplicate the channel impulse response, which will be utilised by MATLAB to simulate the OFDM system performance. In addition, in this section, the implemented OFDM performance is compared to the simulated OFDM performance for each propagation environment based on the channel state information collected in the experiment. Therefore, it is important to discuss about how to create fading channels for our simulations in order to compare the experimental performances of the OFDM system and the simulated performances.

As described in Chapter 4, channel coefficients in the frequency domain are estimated in the function *digital_ofdm_frame_acquisition*. The channel coefficients are then saved into a file. Based the channel coefficient file, an analysis is performed to estimate the number of channel paths. The number of paths is estimated as follows.

- Take one set of estimated channel coefficients of 40 occupied tones from the saved file
- Estimate the complete set of 64 channel coefficients corresponding to 64 subcarriers using linear interpolation algorithm. Denote this complete set of 64 channel coefficients to be $H = [\mathrm{H}_1\ \mathrm{H}_2\ ...\ \mathrm{H}_{64}]$;
- Reverse these channel coefficients from the frequency domain into the time domain using IFFT. Denote this channel impulse response to be $h = [\mathrm{h}_1\ \mathrm{h}_2\ ...\ \mathrm{h}_{64}]$;
- Calculate $P = 10\log_{10}(|h|^2) = 10\log_{10}([|\mathrm{h}_1|^2\ |\mathrm{h}_2|^2\ ...\ |\mathrm{h}_{64}|^2])$; and denote $P = [\mathrm{P}_1\ \mathrm{P}_2\ ...\ \mathrm{P}_{64}]$;
- Find the peak power value $P_{\max}$ of P_i $(1 \le i \le 64)$;
- Count L is the number path has power P_i within 10dB below the peak power value;
- Repeat all above steps for next set of channel coefficients;
- The average of L is the estimated average number paths in channel.

5.3.1 Simulated OFDM performances versus implemented OFDM performances in laboratory environment without obstruction

To fairly compare the performance of the simulated OFDM system and that of the experimental OFDM system, parameters of the simulated OFDM system are chosen to be the same as those in the implemented OFDM system (i.e. with BPSK constellation, FFT size of 64, 40 occupied tones, cyclic prefix length of 16).

After analysing the channel coefficient file, the estimated average number of paths in the realistic channel, i.e. in our laboratory room without obstruction, is found to be 2.78. In addition, since there is no obstruction between antennas, the wireless channel would be a Rician one. Therefore, based on our power analysis of the channel coefficients collected via our experiments, we could approximate the experimental wireless channel to be a Rician channel with three paths with the channel impulse response $h = [0.863 \;\; 0.432 \;\; 0.259]$.

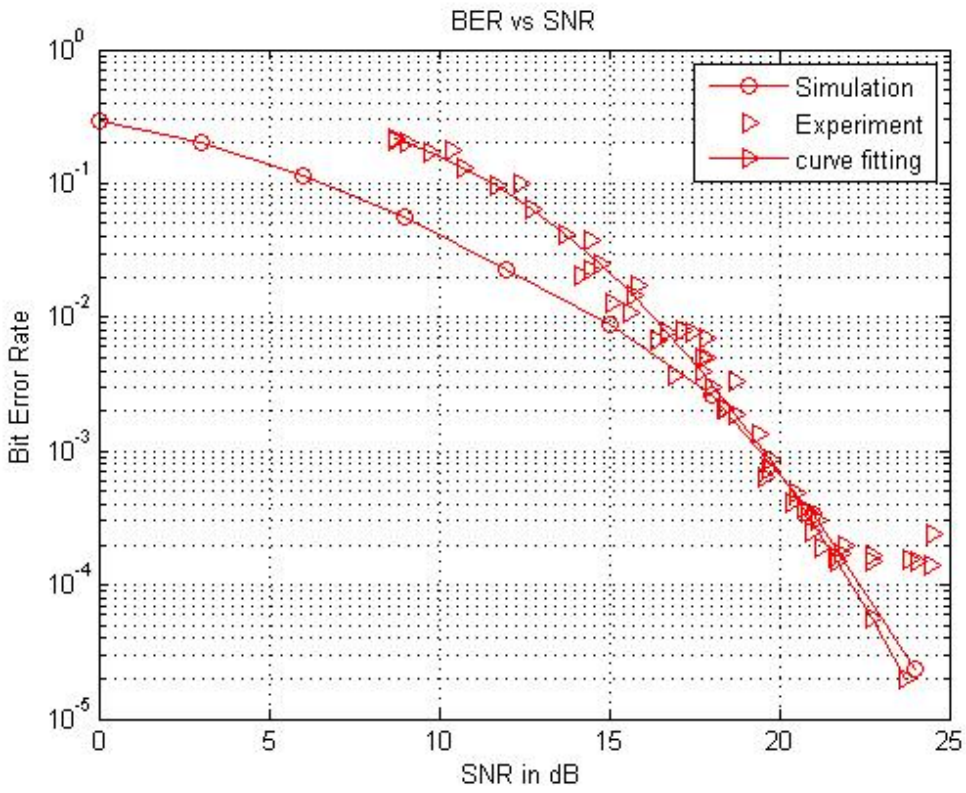

Figure 5-16 Simulated OFDM vs. implemented OFDM in laboratory environment without obstruction

The performances of the simulated OFDM system and the implemented OFDM system are illustrated in Figure 5-16. It is important to recall that, due to the limitation of our hardware, the BER performance of the experimental OFDM system can only be obtained for SNR higher than 8 dB, and that the BER performance of the experimental OFDM system starts to experience the error floor at SNR = 23 dB because the receiver is saturated. Meanwhile, for the simulated OFDM system, we can plot the BER curve for a wider SNR range. For these reasons, we only focus our analysis on the SNR range 8-23 dB.

The Figure 5-12 shows that the performance of the implemented OFDM system with SNR higher than 17dB is the same as the performance of the simulated OFDM system. It is worse than the simulated performance at SNR lower than 17 dB. This is explained by the fact that the synchronisation method and channel estimation method are affected high noise at low SNR. It is also due to the fact that no external frequency reference, such as GPS (Global Positioning System) signal, is used for the synchronisation process due to the limited project budget.

5.3.2 Simulated OFDM performances versus implemented OFDM performances in laboratory environment with obstruction

In this scenario, after the channel coefficient file is analysed, the estimated average number of paths in the realistic propagation channel (laboratory room with obstruction) is found to be approximate 2.34. Because there is no obstruction between the transmitter and receiver antennas, we suggest that the experiment channel in our experiment would be a Rayleigh channel with two to three paths. For the illustration purpose, we simulate the OFDM system with a three-path .Raleigh fading channel with channel impulse response $h = [0.8 \;\; 0.58 \;\; 0.089]$.

Figure 5-17 depicts the performances of the simulated OFDM system and the implemented OFDM system. Similarly to the performances of OFDM systems in the laboratory environment without obstruction, the performance of the implemented OFDM system relatively matches the simulated performance of simulated OFDM system for SNRs higher than 17dB and worse than the simulated performance at lower SNRs. This inferior behaviour is because of two reasons, namely the modest

accuracy of the implemented synchronisation and channel estimation methods, and the lack of very accurate external frequency reference chipset, such as a GPS chipset.

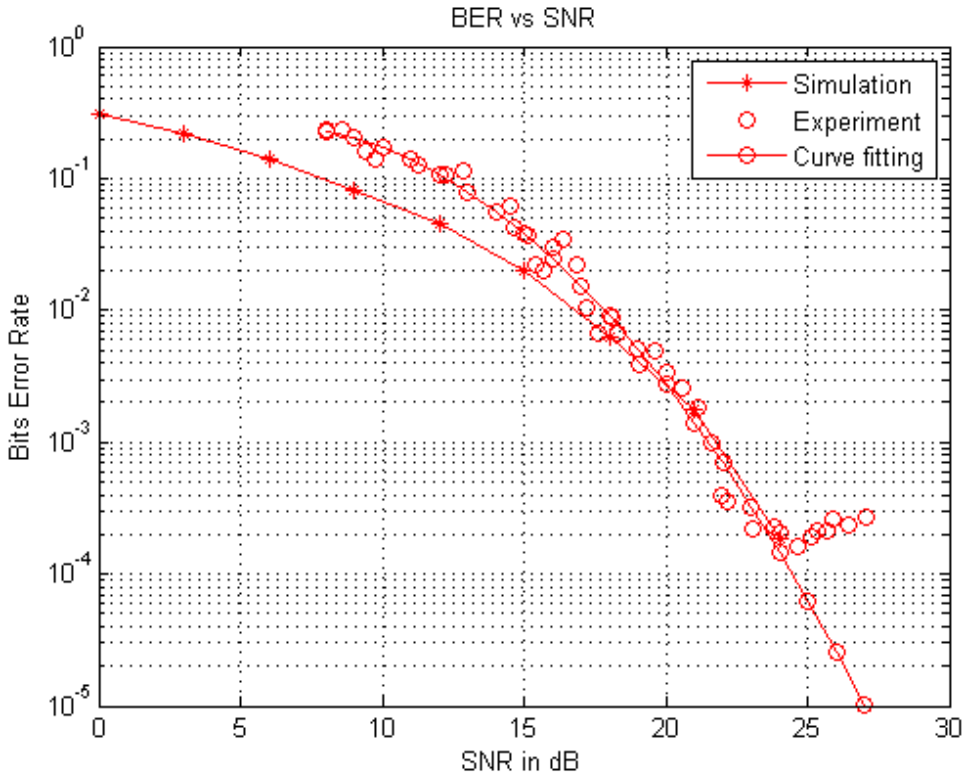

Figure 5-17 Simulated OFDM vs. implemented OFDM in laboratory environment with obstruction

5.3.3 Simulated OFDM performances versus implemented OFDM performances in corridor environment without obstruction

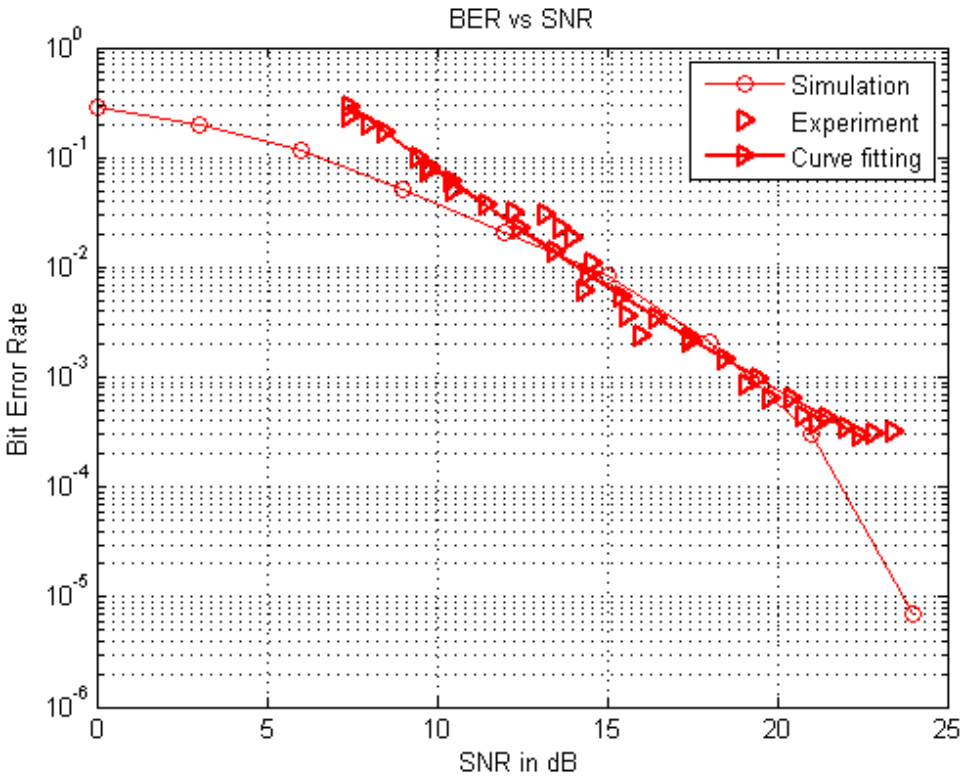

Figure 5-18 Simulated OFDM vs. implemented OFDM in corridor environment without obstruction

Our analysis based on the channel coefficient file indicates that the estimated average number of paths in this scenario (corridor, no obstruction) is around 2.78, similarly to the laboratory room without obstruction. Therefore, we suggest that the wireless channel in our experiment would be a Rician channel which has three paths with the following channel impulse response in time domain $h = [0.85 \;\; 0.41 \;\; 0.31]$.

The implemented and simulated performances of OFDM systems are shown in Figure 5-18. It exhibits that the performance of implemented OFDM system is basically the same as the simulated performance for SNRs higher than 12 dB.

5.3.4 Simulated OFDM performances versus implemented OFDM performances in corridor environment with obstruction

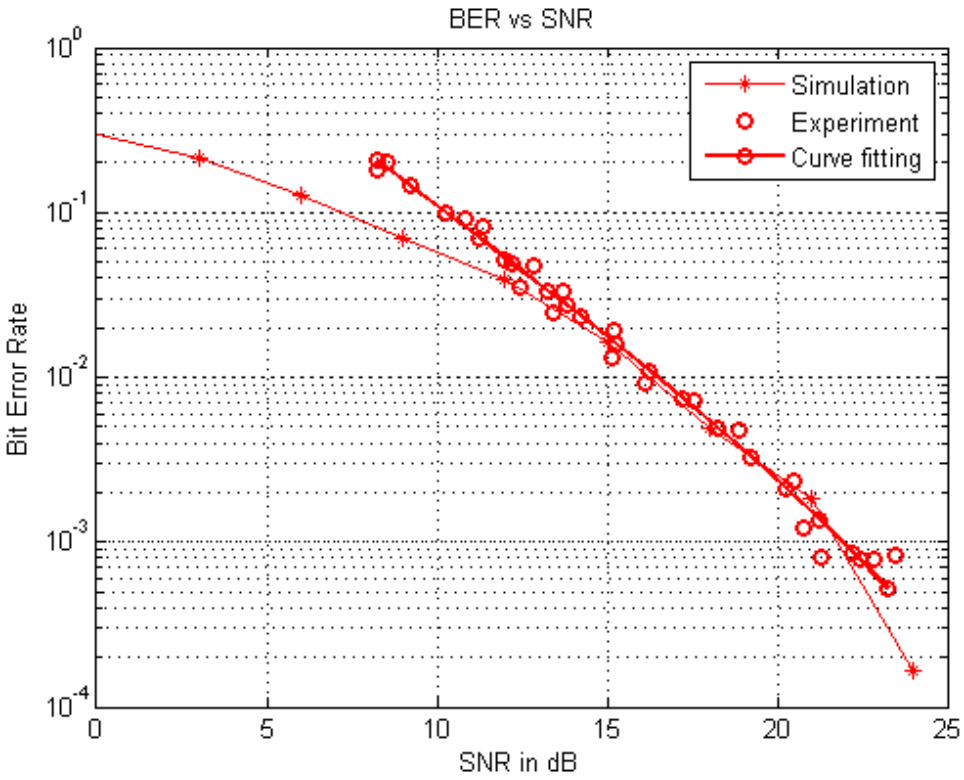

Figure 5-19 Simulated OFDM vs. implemented OFDM in corridor environment with obstruction

Finally, we discuss the corridor propagation environment with an obstruction between the transmitter and receiver antennas. Our analysis based on the channel coefficient file shows that the estimated average number of paths in this case is 2.41, which is slightly higher than that in the case of the laboratory room with obstruction (2.34). Based on our power analysis of the multipath collected via our experiments, we suggest that the experiment channel in our experiment is a Rayleigh channel having three paths with the channel impulse response in time domain $h = [0.83 \;\; 0.52 \;\; 0.18]$.

The simulated and implemented performances of OFDM system are illustrated in Figure 5-15. The results show that the implemented performance is the same as the simulated performance at SNRs higher than 12 dB, i.e. in most of the SNR range we measured by our hardware. Again, the mismatch at the SNR range 8-12 dB is due to

the modest accuracy of the implemented synchronisation and channel estimation methods, and the lack of very accurate external frequency reference chipset, such as a GPS chipset.

Similar to the analysis mentioned in Section 5.3.3, by comparing Figures 5-13 and 5-15, it shows that the implemented synchronisation and channel estimation methods starts to be accurate at SNRs higher than 12 dB for the corridor scenario and 17 dB for the laboratory room. Again, we can interpret the above observation that the indoor environment affects more significantly to the accuracy of the implemented synchronisation and channel estimation processes at low SNRs than the corridor environment does, despite of the fact that both scenarios have similar average numbers of paths and the transmitter-receiver distance in the corridor is six times longer than the distance in the case of laboratory environment. This might be due to the significant absorption of electromagnetic energy by the furniture and equipments in the laboratory room, while almost no furniture or equipment is present in the corridor.

5.4 Chapter summary

This chapter presents the performances of the implemented OFDM system in different propagation environments (laboratory room and corridor) with different constellation modulation schemes (BPSK and QPSK). The performances of the implemented OFDM system are compared to each other. Comparisons have been also made between the implemented performances and the simulated performances. The comparisons confirm that the implemented OFDM system in the testbed works relatively well, especially at high SNRs. At lower SNRs, the testbed suffers from the degradation due to the modest accuracy of the implemented synchronisation and channel estimation processes. These conclusions suggest that more accurate and complicated synchronisation and channel estimation methods as well as a very accurate external frequency reference, such as GPS signal, should be used if one requires an accurate experimental result.

6 CONCLUSIONS

The OFDM technique is widely used in reality, such as in WiFi, WiMax, Long Term Evolution (LTE) systems. In order to have a better understanding of the OFDM system performance, both simulations and actual hardware implementations are required. Implementation of wireless communications using Software Defined Radio (SDR) is one of the emerging research areas nowadays. SDR is a promising technique for multi-type, high speed wireless applications with low requirements on capability of hardware devices, low development cost, while facilitating development processes. In this project, we have implemented an OFDM system based on the GNU platform and USRP2 hardware to evaluate the OFDM performances in different realistic environments and different modulation constellations. The experimental performances are then deeply analysed in comparison with the simulated OFDM performances.

This chapter summarises the whole thesis and highlights our contributions. This chapter is organised as follows. Section 6.1 summarises the research contributions in this thesis. Section 6.2 discusses the future works regarding to the topic of this thesis. Section 6.3 provides overall conclusions for this thesis.

6.1 Research contributions

Our research activities and contributions in this thesis can be summarised as follows

- We have provided a comprehensive literature review for two important techniques, namely OFDM technique and SDR technique with GNU Radio platform and USRP devices. An overview of the current channel estimation, SNR estimation, and synchronisation techniques, which are directly related to the implementation of OFDM systems in hardware, has also been provided. The literature review shows that the implementation of OFDM systems in hardware with SDR has been almost unexplored and there is a lack of comparisons between the performances of implemented OFDM systems in different propagation environments and the simulated OFDM performances, which is the main motivation of this thesis;
- Simulated performances of OFDM systems in MATLAB with channel estimation and SNR estimation techniques and with different modulation schemes have been presented. These simulated performances is used as

benchmark performances to evaluate the implemented OFDM system in different propagation environments;

- A testbed where the OFDM system is implemented has been successfully developed based on GNU Radio software platform and USRP2 hardware. The SDR implementation, including the necessary modifications and new developments, has been documented in the thesis;
- A blind SNR estimation module has been developed to evaluate the BER performance of OFDM in different realistic environments;
- Intensive comparisons and analyses between the performances of the implemented OFDM system in different propagation environments with different digital modulation constellations and the simulated performances have been derived. These comparisons serve three purposes. First, they are the proof-of-concept for the developed testbed. Second, the comparisons allow us to understand the gaps between the simulated and implemented performances. Finally, they also point out the realistic limitations of SDR and hardware.

6.2 Recommendations

Through our development of the OFDM testbed in SDR systems using GNU Radio software platform and USRP devices, some hands-on recommendations could be drawn as follows

- GNU Radio software platform and USRP devices are suggested to implement SDR system because of their advantages such as great flexibility and low cost;
- When researching SDR techniques for the first time, the GNU Radio software and USRP Hardware Driver are recommended to be installed on the VMware software (or any similar software) for simple configurations and modifications;
- The more powerful computers with Gigabit Ethernet cards should be used to improve the performance of the implemented communication system;
- More precise external reference frequency such as GPS could be used to improve the synchronisation accuracy;

- Power attenuators could be used to solve the early power saturation issues at the receivers, thus the experimental SNR range could be expanded.

6.3 Future works

The successfully developed testbed opens various directions to research further the advanced signal processing techniques for OFDM-based systems, such as Multiple-Input Multiple-Output techniques, channel coding, and block spreading. It also facilitates the evaluation of OFDM performances in various other propagation environments, such as out-door environment and mobile wireless channels (where the receiver moves relatively to the transmitter). Therefore, some of my possible future works regarding to the topic of this project could be listed as follows

- Experiment the implemented OFDM system in other realistic environments, including but not limited to out-door propagation environments and mobile wireless channels;
- Add channel coding into the implemented OFDM system;
- Apply enhanced channel estimation and synchronisation techniques to the implemented OFDM system to improve its performance;
- Implement the OFDM system following several standards, such as WLAN, DAB, DVB and IEEE802.11;
- Implement some advanced OFDM systems, such as Block spreading OFDM or MIMO OFDM.

6.4 Conclusions

In this thesis, we have simulated OFDM systems in MATLAB and implemented them in Software Defined Radio, using GNU Radio software platform and USRP2 hardware devices. In addition, OFDM systems have been analysed in different propagation environments with different signal constellations. Performances of the implemented OFDM system in different realistic environments have been then compared to those of the simulated OFDM system in MATLAB.

To fulfil the aforementioned tasks, we first described and analysed the inbuilt example of an implemented OFDM system provided in GNU. Based on this example, we have modified some modules, including the channel estimation module and the synchronisation module, as well as created new modules in order to implement our OFDM system which also takes synchronisation and channel

estimation processes into consideration. The newly created blocks are the blind SNR estimation module and BER calculator. Necessary modifications and new developments have been documented in detail in this project. Performances of the implemented OFDM system have been evaluated in our laboratory room and in a corridor, with and without obstructions between the transmitter and receiver, and with different digital modulation schemes.

To compare the simulated OFDM performances in MATLAB with the implemented OFDM performances, we have managed to create as similar propagation conditions as well as system settings to the implemented OFDM system as possible. It means that most algorithms used in the implemented OFDM system have been applied to the simulated OFDM system in MATLAB. The simulated OFDM system is built based on the analysis of the implemented OFDM system.

Overall, performances of the simulated and implemented OFDM systems are relatively well matched in the SNR ranges which we could measure with our hardware. This performance matching confirms the validity of our testbed and our implemented signal processing techniques. This performance match can be seen with both BPSK and QPSK modulations, in different realistic propagation environments, and in both unobstructed and obstructed scenarios. The comparisons also show the limitation of the hardware that the receiver would be saturated at high SNRs, given that we do not use any attenuator in our experiments. The saturation levels are variable, depending on particular propagation conditions, as shown in our project.

In conclusion, SDR techniques with GNU Radio software platform and USRP2 devices can be used to implement OFDM-based systems. More research on this research field is required to enhance the performance of OFDM-based systems on hardware.

REFERENCES

[1] D. Tse and P. Viswanath, *Fundamentals of Wireless Communication*, Cambridge University Press, 2005.

[2] L. Hanzo, et al., *Coherent versus Non-coherent and Cooperative Turbo Transceivers*, Wiley-IEEE Press, 2011.

[3] H. D. Schulze and C. Lueders, *Theory and applications of OFDM and CDMA: wideband wireless communications*, John Wiley & Son, Chichester, 2005.

[4] *Overview of the GNU System*, Available: http://www.gnu.org/gnu/gnu-history.html, [Access date: 1 July 2013]

[5] M. C. Jeruchim, P. Balaban, and K. S. Shanmugan, *Simulation of Communicaton Systems*, 2nd ed., Kluwer Academic/Plenum Publishers, New York, 2000.

[6] A. W. Graham, N. C. Kirkman, and P. M. Paul, *Mobile Radio Network Design in the VHF and UHF Bands*, John Wiley & Sons, 2007.

[7] S. Haykin and M. Moher, *Modern Wireless Communication*, Prentice Hall, 2005.

[8] A. Doukas and G. Kalivas, "Rician K Factor Estimation for Wireless Communication Systems", *Proc. International Conference on Wireless and Mobile Communications*, pp. 69-69, Jul. 2006.

[9] U. Madhow, *Fundamentals of Digital Communication*, Cambridge University Press, 2008.

[10] P. M. Shankar, *Introduction to Wireless Systems*, John Wiley & Son, 2002.

[11] S. Weinstein and P. Ebert, "Data Transmission by Frequency-Division Multiplexing Using the Discrete Fourier Transform", *IEEE Transactions on Communication Technology*, vol. 19, no. 5, pp. 628-634, Oct. 1971.

[12] X. Jinpeng, et al., "FPGA implementation of frame synchronization and symbol timing synchronization based on OFDM system for IEEE 802.11a", *Proc. International Symposium on Intelligent Signal Processing and Communication Systems (ISPACS)*, pp. 1-4, Dec. 2010.

[13] R. Nakamura and Y. Sanada, "Experimental investigation of IEEE802.11n reception with fractional sampling", *Proc. 21st International Symposium on Personal Indoor and Mobile Radio Communications (PIMRC)*, pp. 1044-1048, Sept. 2010.

[14] W. Sun and Y. Zhang, "A Frame Synchronization and Symbol Timing Synchronization Algorithm in Burst OFDM Communication Based on IEEE802.11a", *Proc. International Forum on Information Technology and Applications (IFITA)*, pp. 190-193, May 2009.

[15] S. Toshiya, et al., "Experimental investigation of fractional sampling in IEEE802.11a WLAN system", *Proc. 11th IEEE Singapore International Conference on Communication Systems (ICCS)*, pp. 1368-1373, Nov. 2008.

[16] L. Hanzo, *OFDM and MC-CDMA for broadband multi-user communications, WLANs, and broadcasting*, John Wiley & Son, New York, 2003.

[17] H. Yafei and T. Hase, "New OFDM system without guard interval", *Proc. Digest of Technical Papers International Conference on Consumer Electronics*, pp. 1-2, Jan. 2009.

[18] A. Selim, B. Ozgul, and L. Doyle, "Efficient Cyclic Prefix Reconstruction for Shaped OFDM Systems without Cyclic Prefix", *Proc. Global Telecommunications Conference (GLOBECOM 2010)*, pp. 1-5, Dec. 2010.

[19] U. Yunus, L. Hai, and K. Yamashita, "A Novel Equalization Method for OFDM Systems without Guard Interval", *Proc. Global Telecommunications Conference (GLOBECOM 2010)*, pp. 1-5, Dec. 2010.

[20] T. A. Weiss and F. K. Jondral, "Spectrum pooling: an innovative strategy for the enhancement of spectrum efficiency", *IEEE Communications Magazine*, vol. 42, no. 3, pp. S8-14, 2004.

[21] R. Rajbanshi, A. M. Wyglinski, and G. J. Minden, "Peak-to-Average Power Ratio Analysis for NC-OFDM Transmissions", *Proc. Vehicular Technology Conference (VTC)*, pp. 1351-1355, Oct. 2007.

[22] C. Jung-Chieh, "Partial transmit sequences for PAPR reduction of OFDM signals with stochastic optimization techniques", *IEEE Transactions on Consumer Electronics*, vol. 56, no. 3, pp. 1229-1234, Aug. 2010.

[23] J. Shiann-Shiun and C. Jia-Ming, "Efficient PAPR Reduction in OFDM Systems Based on a Companding Technique With Trapezium Distribution", *IEEE Transactions on Broadcasting*, vol. 57, no. 2, pp. 291-298, June 2011.

[24] A. Bury, J. Egle, and J. Lindner, "Diversity comparison of spreading transforms for multicarrier spread spectrum transmission", *IEEE Transactions on Communications*, vol. 51, no. 5, pp. 774-781, May 2003.

[25] M. L. McCloud, "Optimal binary spreading for block OFDM on multipath fading channels", *Proc. Wireless Communications and Networking Conference (WCNC)*, pp. 965-970 Vol.2, March 2004.

[26] M. Jalloh and P. Das, "On the Performance of Transmit and Receive Diversity OFDM Systems with Phase Noise and Imperfect Channel Estimation", *Proc. Vehicular Technology Conference (VTC)*, pp. 1-6, April 2009.

[27] G. V. Rangaraj, Jalihal, D & Giridhar, K., "Exploiting Multipath Diversity in Multiple Antenna OFDM Systems With Spatially Correlated Channels", *IEEE Transactions on Vehicular Technology*, vol. 54, no. 4, pp. 1372-1378, July 2005.

[28] A. V. Sarad and S. Srikanth, "Improved interference diversity in multicellular OFDMA systems", *Proc. First International Communication Systems and Networks and Workshops (OMSNETS)*, pp. 1-8, Jan. 2009.

[29] L. H. Xing, et al., "Channel Estimation for Transmitter Diversity OFDM Systems", *Proc. IEEE Conference on Industrial Electronics and Applications*, pp. 1-4, May 2006.

[30] Y. Yiwei, H. Xiaojing, and E. Dutkiewicz, "Block spread OFDMA system with space-time coded MIMO over frequency selective fading channels", *Proc. Third International Conference on Communications and Networking in China (ChinaCom)*, pp. 464-468, Aug. 2008.

[31] M. L. McCloud, "Analysis and design of short block OFDM spreading matrices for use on multipath fading channels", *IEEE Transactions on Communications* vol. 53, no. 4, pp. 656-665, April 2005.

[32] M.-o. Pun, M. Morelli, and C. C. J. Kuo, *Multi-Carrier Techniques For Broadband Wireless Communications: A Signal Processing Perspectives*, Imperial College Press, London, 2007.

[33] J.-H. Wen, et al., "PN code-aided timing estimation, channel estimation and signal compensation in OFDM systems", *Digital Signal Processing*, vol. 20, no. 3, pp. 860-868, May 2010.

[34] W. Qi, M. Simko, and M. Rupp, "Modified Symbol Timing Offset Estimation for OFDM over Frequency Selective Channels", *Proc. Vehicular Technology Conference (VTC Fall)*, pp. 1-5, Sept. 2011.

[35] J. J. van de Beek, M. Sandell, and P. O. Borjesson, "ML estimation of time and frequency offset in OFDM systems", *IEEE Transactions on Signal Processing*, vol. 45, no. 7, pp. 1800-1805, Jul. 1997.

[36] P. Byungjoon, et al., "A blind OFDM synchronization algorithm based on cyclic correlation", *IEEE Signal Processing Letters*, vol. 11, no. 2, pp. 83-85, Nov. 2004.

[37] T. M. Schmidl and D. C. Cox, "Robust frequency and timing synchronization for OFDM", *IEEE Transactions on Communications*, vol. 45, no. 12, pp. 1613-1621, Dec. 1997.

[38] K. Dong-Kyu, et al., "A new joint algorithm of symbol timing recovery and sampling clock adjustment for OFDM systems", *IEEE Transactions on Consumer Electronics*, vol. 44, no. 3, pp. 1142-1149, Aug. 1998.

[39] T.-D. Chiueh, P.-Y. Tsai, and I.-W. Lai, *Baseband Receiver Design for Wireless MIMO-OFDM Communications*, 2nd ed., John Wiley & Sons, Singapore, 2012.

[40] M. K. Ozdemir and H. Arslan, "Channel estimation for wireless ofdm systems", *IEEE Communications Surveys & Tutorials*, vol. 9, no. 2, pp. 18-48, Jul. 2007.

[41] H. Arslan and T. Yucek, "Estimation of Frequency Selectivity for OFDM Based New Generation Wireless Communication Systems", *Proc. World Wireless Congress*, San Francisco, May 2003.

[42] L. Ye, "Simplified channel estimation for OFDM systems with multiple transmit antennas", *IEEE Transactions on Wireless Communications*, vol. 1, no. 1, pp. 67-75, Jan. 2002.

[43] A. Dowler, A. Doufexi, and A. Nix, "Performance evaluation of channel estimation techniques for a mobile fourth generation wide area OFDM system", *Proc. IEEE Vehicular Technology Conference*, Vancouver, Canada, pp. 2036-2040, Sept. 2002.

[44] I. Barhumi, G. Leus, and M. Moonen, "Optimal training design for MIMO OFDM systems in mobile wireless channels", *IEEE Transactions on Signal Processing*, vol. 51, no. 6, pp. 1615-1624, Jun. 2003.

[45] S. Boumard and A. Mammela, "Channel estimation versus equalization in an OFDM WLAN system", *Proc. IEEE Vehicular Technology Conference*, Rhodes, Greece, pp. 653-657, May 2001.

[46] M. Morelli and U. Mengali, "A comparison of pilot-aided channel estimation methods for OFDM systems", *IEEE Transactions on Signal Processing*, vol. 49, no. 12, pp. 3065-3073, Dec. 2001.

[47] J. Zhang and P. Zhang, "An improved 2-dimensional pilot-symbols assisted channel estimation in OFDM systems", *Proc. IEEE Vehicular Technology Conference*, pp. 1595-1599, April 2003.

[48] Y. S. Cho, et al., *MIMO-OFDM Wireless Communications with MATLAB*, John Wiley & Sons, 2010.

[49] T. Haiyun, K. Y. Lau, and R. W. Brodersen, "Interpolation-based maximum likelihood channel estimation using OFDM pilot symbols", *Proc. IEEE Global Telecommunications Conference*, pp. 1860-1864, Nov. 2002.

[50] S. Yan, W. Xiaowen, and K. J. R. Liu, "A joint channel estimation and unequal error protection scheme for video transmission in OFDM systems", *Proc. International Conference on Image Processing*, pp. I-549-I-552, Sept. 2002.

[51] H. Chunping, S. Shenghui, and C. Dazhong, "Channel estimation for adaptive OFDM system and effects of estimation error on system performance", *Proc. International Workshop on Mobile and Wireless Communications*, NewYork, pp. 200-204, Sept. 2002.

[52] Y. Heejung, K. Myung-Soon, and L. Sok-Kyu, "Channel estimation and equalization for high speed mobile OFDM systems", *Proc. Conference Record of the Thirty-Seventh Asilomar Conference on Signals, Systems and Computers*, Daejeon, South Korea, pp. 693-697, Nov. 2003.

[53] F. Said and H. Aghvami, "Linear two dimensional pilot assisted channel estimation for OFDM systems", *Proc. IEE Conference on Telecommunications*, London, UK, pp. 32-36, Apr. 1998.

[54] T. Zijian, et al., "Pilot-Assisted Time-Varying Channel Estimation for OFDM Systems", *IEEE Transactions on Signal Processing*, vol. 55, no. 5, pp. 2226-2238, May 2007.

[55] D. R. Pauluzzi and N. C. Beaulieu, "A comparison of SNR estimation techniques for the AWGN channel", *IEEE Transactions on Communications*, vol. 48, no. 10, pp. 1681-1691, Oct. 2000.

[56] D. Athanasios and G. Kalivas, "SNR Estimation for Low Bit Rate OFDM Systems in AWGN Channel", *Proc. International Conference on Networking, International Conference on Systems and International Conference on Mobile Communications and Learning Technologies*, pp. 198-198, Apr. 2006.

[57] Y. Wang, et al., "A New Noise Variance Estimation Algorithm for Multiuser OFDM Systems", *Proc. IEEE International Symposium on Personal, Indoor and Mobile Radio Communications*, pp. 1-4, Sept. 2007.

[58] A. Doukas and G. Kalivas, "A Novel SNR per Subcarrier Estimation Scheme for OFDM Systems in Frequency Selective Channels", *Proc. IEEE International Conference on Wireless and Mobile Computing Networking and Communications*, pp. 340-345, Oct. 2008.

[59] M. Zivkovic and R. Mathar, "Preamble-Based SNR Estimation in Frequency Selective Channels for Wireless OFDM Systems", *Proc. IEEE Vehicular Technology Conference*, pp. 1-5, Apr. 2009.

[60] R. S. Manzoor, et al., "Front-end estimation of Noise Power and SNR in OFDM systems", *Proc. International Conference on Intelligent and Advanced Systems*, pp. 435-439, Nov. 2007.

[61] C. Tao and C. Tellambura, "Power delay profile and noise variance estimation for OFDM", *IEEE Communications Letters*, vol. 10, no. 1, pp. 25-27, Jun. 2006.

[62] F. X. Socheleau, A. Aissa-El-Bey, and S. Houcke, "Non data-aided SNR estimation of OFDM signals", *IEEE Communications Letters*, vol. 12, no. 11, pp. 813-815, Dec. 2008.

[63] Lo, et al., "Maximum Likelihood SNR estimation for asynchronously oversampled OFDM signals", *Proc. IEEE Workshop on Signal Processing Advances in Wireless Communications*, pp. 26-30, Jul. 2008.

[64] C. H. Aldana, et al., "Accurate noise estimates in multicarrier systems", *Proc. IEEE Vehicular Technology Conference*, pp. 434-438 vol.1, Sept. 2000.

[65] L. Yunxin, "Blind SNR estimation of OFDM signals", *Proc. International Conference on Microwave and Millimeter Wave Technology*, pp. 1792-1796, May 2010.

[66] H. Jingyu, et al., "Novel Scheme for Joint Estimation of SNR, Doppler, and Carrier Frequency Offset in Double-Selective Wireless Channels", *IEEE Transactions on Vehicular Technology*, vol. 58, no. 3, pp. 1204-1217, Mar. 2009.

[67] Y. Kang, K. Kim, and H. Park, "Efficient DFT-based channel estimation for OFDM systems on multipath channels", *IET Communications*, vol. 1, no. 2, pp. 197-202, Apr. 2007.

[68] M. Morelli and U. Mengali, "An improved frequency offset estimator for OFDM applications", *Proc. Communication Theory Mini-Conference, 1999*, pp. 106-109, Jun. 1999.

[69] M. Zivkovic and R. Mathar, "Preamble-based SNR estimation algorithm for wireless MIMO OFDM systems", *Proc. International Symposium on Wireless Communication Systems*, pp. 96-100, 2009.

[70] E. Blossom, *Exploring GNU Radio*, Nov. 2004, Available: http://www.gnu.org/software/gnuradio/doc/exploring-gnuradio.html, [Access date: Jan. 5, 2012]

[71] D. C. Tucker and G. A. Tagliarini, "Prototyping with GNU radio and the USRP - where to begin", *Proc. IEEE Southeastcon*, Atlanta, GA, pp. 50-54, Mar. 2009.

[72] A. Mate, L. Kuo-Hao, and I. T. Lu, "Spectrum sensing based on time covariance matrix using GNU radio and USRP for cognitive radio", *Proc. IEEE Long Island Systems, Applications and Technology Conference (LISAT)*, pp. 1-6, May 2011.

[73] R. K. Ganti, et al., "Implementation and Experimental Results of Superposition Coding on Software Radio", *Proc. IEEE International Conference on Communications (ICC)*, pp. 1-5, May 2010.

[74] M. A. Sarijari, et al., "Energy detection sensing based on GNU radio and USRP: An analysis study", *Proc. IEEE Malaysia International Conference on Communications (MICC)*, pp. 338-342, Dec. 2009.

[75] M. Bruno, et al., "Widely tunable RF transceiver front end for software-defined radio", *Proc. IEEE Military Communications Conference*, pp. 1-6, Oct. 2009.

[76] M. Ettus, *Universal Software Radio Peripheral*, Available: http://www.ettus.com, [Access date: Feb. 25, 2012]

[77] H. Wang, et al., "Blind standard identification with bandwidth shape and GI recognition using USRP platforms and SDR4all tools", *Proc. International Conference on Cognitive Radio Oriented Wireless Networks & Communications (CROWNCOM)*, pp. 1-5, Jun. 2010.

[78] H. Wang, et al., "Blind Bandwidth Shape Recognition for Standard Identification Using USRP Platforms and SDR4all Tools", *Proc. Sixth Advanced International Conference on Telecommunications (AICT)*, pp. 147-152, May 2010.

[79] A. Kaszuba, R. Checinski, and J. Lopatka, "MIMO Implementation with Alamouti Coding Using USRP2", *Proc. Progress In Electromagnetics Research Symposium*, pp. 899-902, Mar. 2011.

[80] L. Kun-chan and L. Mei-wen, "Feasibility study of using FM radio for data transmission in a vehicular network", *Proc. 2010 International Computer Symposium (ICS)*, pp. 55-60, Dec. 2010.

[81] A. Marwanto, et al., "Experimental study of OFDM implementation utilizing GNU Radio and USRP - SDR", *Proc. 9th Malaysia International Conference on Communications (MICC)*, pp. 132-135, Dec. 2009.

[82] P. Alvarez, et al., "Energy detection and eigenvalue based detection: An experimental study using GNU radio", *Proc. 14th International Symposium on Wireless Personal Multimedia Communications (WPMC)*, pp. 1-5, Oct. 2011.

[83] P. B. Jorgensen, et al., "Implementation of LTE SC-FDMA on the USRP2 software defined radio platform", *Proc. Communication Technologies Workshop (Swe-CTW)*, pp. 34-39, Oct. 2011.

[84] *GNU Radio Companion*, Available: http://gnuradio.org/redmine/projects/gnuradio/wiki/GNURadioCompanion, [Access date: 12 July 2013]

[85] G. Berardinelli, et al., "An SDR architecture for OFDM transmission over USRP2 boards", *Proc. Conference Record of the Forty Fifth Asilomar Conference on Signals, Systems and Computers (ASILOMAR)*, pp. 965-969, Nov. 2011.

[86] A. Blad, E. Axell, and E. G. Larsson, "Spectrum sensing of OFDM signals in the presence of CFO: New algorithms and empirical evaluation using USRP", *Proc. IEEE International Workshop on Signal Processing Advances in Wireless Communications (SPAWC)*, pp. 159-163, Jun. 2012.

[87] M. Rahman, et al., "The Study of OFDM ICI Cancellation Schemes in 2.4 GHz Frequency Band Using Software Defined Radio", *Proc. International Conference on Wireless Communications, Networking and Mobile Computing (WiCOM)*, pp. 1-6, Sept. 2011.

[88] J. J. van de Beek, et al., "On channel estimation in OFDM systems", *Proc. IEEE Vehicular Technology Conference*, pp. 815-819 vol.2, 1995.