Design, Implementation and Assessment of BPSK and QPSK PAPR over OFDM signals using LimeSDR

A thesis submitted to the Department of Electrical Engineering,
UNIVERSITY OF CAPE TOWN, in partial fulfilment of the requirements for the degree of

Master of Engineering

at the

University of Cape Town

By

Tinashe Roxley Kagande

Supervised by:

DR. SIMON WINBERG

2022

Declaration

I know the meaning of plagiarism and declare that all the work in the document, save for that which is properly acknowledged, is my own. This thesis/dissertation has been submitted to the Turnitin module and I confirm that my supervisor has seen my report and any concerns revealed by such have been resolved with my supervisor.

Signature of Author: ..... Signed by candidate .....

University of Cape Town

Cape Town

ABSTRACT

Modern day communications required efficient and affordable wireless communication techniques. Traditional wireless communication systems involved a lot of hardware for each component within the system. Software Defined Radios (SDR) became popular and were a major area of research as they provided a highly adaptive software solution to the traditional analog hardware solutions that were commonly implemented for communication systems. The benefits of using SDRs outweighed those of traditional analog hardware by a significant margin, hence so much research and development was pursued in this area. SDRs offered upgradability and adaptation often without needing to change the hardware, giving them the edge over traditional hardware approaches that needed replacement for upgrades.

With Orthogonal Frequency Division Multiplexing (OFDM) being one of the most popular modulation techniques for Next Generation Networks (NGN), it was important to understand how best it could be delivered using low-cost SDRs. The biggest challenge of OFDM multiplexing was high PAPR, which could necessitate expensive circuitry for ADC/DAC components of an SDR solution. OFDM signals were popularly modulated either by Binary Phase Shift Keying (BPSK) or Quadrature Phase Shift Keying (QPSK), which then brought in the question *"What was their contribution to Peak-to-Average Power Ratio (PAPR) in an OFDM system?"*. Consequently, this project compared BPSK and QPSK in terms of PAPR in OFDM signals. A personal computer was used to host GnuRadio-based prototypes on which an OFDM system was developed using OFDM blocks inbuilt in the software. LimeSDR hardware was used to sample radio waves. A LimeSDR block was implemented in GnuRadio to interconnect with the LimeSDR module. An OFDM transceiver was designed in GnuRadio, and the code developed for this project was also open source. GnuRadio was selected specifically for its open-source flexibility, that allowed adaptability and the prospect to experiment with code, which was expected to be of benefit for future work.

For this project, pre-selected data stored on the host PC, was transmitted from the OFDM transmitter through the LimeSDR antennas and received by the Lime SDR antennas, then demodulated and saved in a different folder on the host PC. Once this was achieved, user interface facilities were added to facilitate use and testing. Results from the testing demonstrated the compatibility of LimeSDR and GnuRadio and showed significant differences between a BPSK modulated signal and QPSK modulated signal in terms of

PAPR. This project aimed to provide contributions to the radio and wireless communication field as well as being supportive towards other ongoing projects taking place in the UCT Electrical Engineering Department that connected to pertinent considerations for 5G and IoT wireless remote sensing solutions.

ACKNOWNLEDGEMENTS

First and foremost, I would like to give gratitude to my supervisor, Dr. Simon Winberg, for introducing me to Software Defined Radios, for which I found passion in. He further gave me all the support I needed to complete this thesis and guided me from start to finish. Gratitude also goes to my fellow lab and course mates for the unwavering support they gave during the time I did my project. Thanks also goes to the department of Electrical engineering in the Faculty of EBE at the University of Cape Town for all the technical support they gave. I would also like to thank the MasterCard Foundation for the financial support they gave during my studies and the funds that they availed for the purchase of LimeSDR which was a critical component in my research. Finally, I would like to thank my family for all the support they gave during my studies especially during the uncertain times of national lockdown during the COVID-19 pandemic.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF ABBREVIATIONS

ADC	Analogue to Digital Converter
AWGN	Additive White Gaussian Noise
BER	Bit Error Rate
BPSK	Binary Phase Shift Keying
CRC	Cyclic Redundancy Check
DAC	Digital to Analog Converter
DPSK	Differential Phase Shift Keying
DSP	Digital Signal Processing
FDD	Frequency Division Duplex
FDM	Frequency Division Multiplexing
FFT	Fast Fourier Transform
FPGA	Field Programmable Gate Array
GSM	Global System for Mobile communication
GUI	Graphical User Interface
HPA	High Power Amplifier
IDFT	Inverse Discrete Fourier Transform
IEEE	Institute of Electrical and Electronics Engineers
IFFT	Inverse Fast Fourier transform
LNA	Low Noise Amplifiers
LPF	Low Pass Filter
LTE	Long Term Evolution
MIMO	Multiple Input Multiple Output
NCO	Numerical Controller Oscillator
NGN	Next Generation Networks
NRZ	Non-Return-to-Zero
OFDM	Orthogonal Frequency Division Modulation
OSI	Open System of Interconnection
PAPR	Peak to Average Power
PC	Personal Computer
PPA	Personal Package Archive
PSK	Phase Shift Keying
PTS	Partial Transmit Sequence

QPSK	Quadrature Phase Shift Keying
RX	Receive
SDR	Software-Defined Radio
SNR	Signal to Noise Ratio
SOC	System On Chip
SQNR	Signal to Quantization Noise Ratio
SWIG	Simplified Wrapper and Interface Generator
TDD	Time Division Duplex
TRX	Transceiver
TSP	transceiver signal processor
TX	Transmission
USRP	Universal Software Radio Peripheral

NOMENCLATURE

Analogue to digital Converter (ADC): is an electronic integrated circuit that is used to convert data from its analogue form to a digital form of 1s and zeros.

Binary Phase Shift Keying (BPSK): It is a digital modulation technique that modulates data using two phases using the carrier wave as a reference point.

Digital to Analog Converter (DAC): is an electronic circuit that converts digital data that is in the form of 1s and 0s into an analogue form.

Field Programmable Gate Array (FPGA): It is an integrated circuit (IC) which is configurable by the end user to their own specifications.

GnuRadio: is a free & open-source software development toolkit that provides signal processing blocks to implement software radios.

Orthogonal Frequency Division Modulation (OFDM): Is a type of modulation scheme where a single steam of data is divided into multiple steams and then transmitted using multiple channels.

Peak to Average Power Ratio (PAPR): is the relation between the maximum power of a sample in a given OFDM transmit symbol divided by the average power of that OFDM symbol.

Quadrature Phase Shift Keying (QPSK): is a type of phase shift keying in which only 2 bits are modulated at once using one of the four (0, 90, 180, or 270 degrees) possible carrier phase shifts.

Software-Defined Radio (SDR): is a radio communication system that has most of the technology that has been traditionally implemented through hardware is replaced by software in this case we used LimeSDR.

CHAPTER 1: INTRODUCTION

The focus of this project is to perform a comparative analysis of an Orthogonal Frequency Division Multiplexing (OFDM) signal modulated with either Binary Phase Shift Keying (BPSK) or Quadrature Phase Shift Keying (QPSK). Robust modulation schemes such as (OFDM) are implemented in next-generation networks for example in Long Term Evolution (LTE) and fifth generation networks. OFDM allows a large chunk of data to be transmitted via radio waves from the transmitter to the receiver. The radio signal is divided into small sub-signals which are then transmitted at the same time but at different frequencies. In his research, Chang[1] put forward the idea of OFDM which was an advancement of Frequency Division Multiplexing (FDM). [2]–[5] gives detailed research in OFDM that will relate to this thesis. Current solutions for the use of OFDM-based technology require the use of expensive Analogue to Digital Converters (ADC) to cater for the high Peak to Average Power Ratio (PAPR). There are many methods to generate data on the time signal where data would have been modulated by the OFDM system which include BPSK, QPSK, DPSK, etc. BPSK and QPSK are digital phase modulation techniques that are commonly used in the generation of OFDM modulated signals. Software-Defined Radios (SDR) have opened a whole lot of opportunities in the communication industry. Fig 1.1 illustrates an OFDM signal transmitted using five different subcarriers showing the average peak amplitude of each subcarrier. Since SDRs are the future of Radio Access Networks (RANs), the comparative analysis between BPSK and QPSK is intended to be done on a platform that supports SDR.

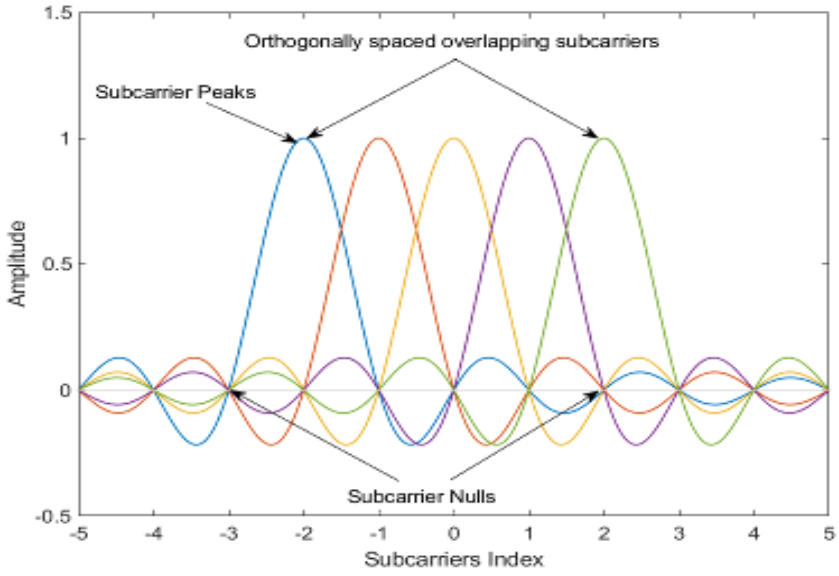


Fig 1. 1: OFDM signal transmission with different subcarriers.

1.1 BACKGROUND

OFDM is the future for Next Generation Networks (NGN).The authors of [6], [7] emphasise how OFDM is important for NGN. The authors of [7] gives an important background for this thesis as they explain in-depth the contribution of OFDM systems to NGN. To build an efficient infrastructure for NGN, research must be done on how to efficiently implement OFDM in the communication infrastructure. As stated in the introduction, one major issue of adopting OFDM in wireless communication systems is high PAPR. Major research such as in [8] has been done on how to reduce PAPR. Solutions provided in [8] such as Partial Transmit Sequence (PTS), clipping and filtering, and coding schemes have proven to be effective but are complex to implement. These techniques work for both BPSK and QPSK. Even if PAPR is reduced by methods described in [8] it can further be reduced by selecting a more effective modulation technique. The main concern in [8] is to reduce PAPR after the OFDM signal was already generated which brings the question if it is possible to generate the OFDM signal with low PAPR.

As stated in the introduction BPSK and QPSK are the most popular techniques used in OFDM signal generation. OFDM systems offer unparalleled bandwidth savings hence higher spectral efficiency. Since the future of communication is high-speed connectivity, which can be provided by OFDM, there is a gap in research on how to efficiently and effectively implement OFDM in digital communications. One such area that will be of major focus is to demonstrate and measure the feasibility of an OFDM system and measure the performance of the two modulation techniques considering PAPR.

The future of communication systems is based on SDR. SDRs offer an appealing approach for flexible prototyping of a communication system. The functionality of the system is SDR relies on the software implemented in the programmable device: field-programmable gate array (FPGA), system on chip(SOC), or general-purpose processor(GPP). These present opportunities such as reconfiguration of the system even during run-time. Such opportunities make it possible to implement SDR on wireless communication systems such as prototyping an SDR OFDM communication system using an open-source tool GnuRadio and the RF hardware interface LimeSDR.

Research done in [9] is one of the most recent and relevant projects in which OFDM is implemented on an SDR platform. The researchers designed a transmitter and a receiver using Universal Software Radio Peripheral (USRP) and signal processing was done on a GnuRadio. The authors in [9] proved that SDR can implement OFDM in communication

systems. Results showed that SDR allows the design of flexible and scalable systems which can adapt to new applications and sundry communication systems. Their project has layered a foundation for this research as they left a gap in determining the most efficient way to implement OFDM on an SDR platform. It is this gap that necessitates the need to expand this research and accommodate an efficient way signals are modulated using the SDR platform.

1.2 PROBLEM DESCRIPTION

This research intends to develop an OFDM receiver prototype on a GnuRadio and implement two case scenarios in which the first one, BPSK is used and the second one QPSK is used for comparison sake of PAPR and bit error rate (BER) analysis. One of the most significant problems in OFDM systems is handling very large PAPRs. When all the sub-carriers align themselves, then the peak signal will occur with the largest amplitude as illustrated in Fig 1.1. To process all the peaks in an OFDM signal, the radio frequency (RF) power amplifier (PA) must provide gain without compressing any peak power level. This means that the power amplifier must increase its average power requirement which will, in turn, reduce the efficiency of the PA.

Fig 1.2 represents the signal's amplitude over time. The horizontal axis represents time, while the vertical axis (ordinate) represents the amplitude of the signal. In the graph, each point represents the amplitude of the signal at a particular moment in time. The amplitude of the signal can vary significantly over time due to the use of multiple subcarriers in OFDM. As a result, the amplitude of the signal can occasionally reach very high levels, causing the PAPR to be high.

Highly linear upconverters are also required to cater to the high PAPR in the OFDM modulation scheme. This disadvantage of having highly linear upconverters is that the upconverter must have a high-level compression point which draws a lot of power. Therefore the cost of design will be high as the power supply becomes more expensive. Implementing power supplies that dissipate high output power will need more vigorous heat dissipation techniques such as special board designs and bigger heat sinks which will, in turn, increase the overall design cost.
In conclusion, OFDM systems face the challenge of high-power consumption by the receiver and hence require expensive components to counter the power issues which are due to high

PAPR. Also, high PAPR decreases the efficiency of the amplifier and reduces the signal to quantization noise ratio (SQNR) of the ADC. This research seeks to see to compare the effects of BPSK and QPSK on PAPR if there are any and make a recommendation on which modulation technique to use for the generation of OFDM signals with the idea of designing cheap and efficient receivers systems.

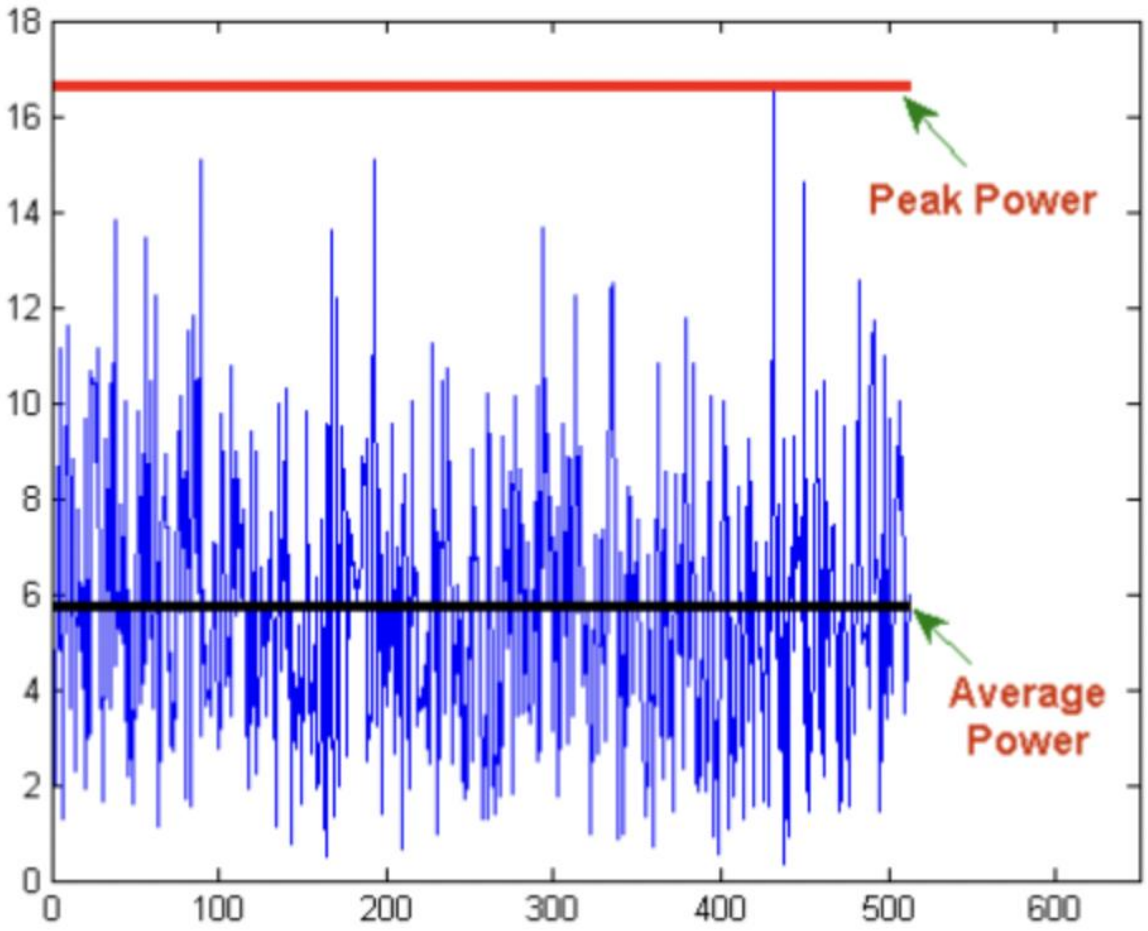


Fig 1. 2: Illustration of PAPR in a transmitted OFDM signal. (Image Source:[10])

1.3 FOCUS

The focus of this research is to compare BPSK and QPSK modulation techniques concerning PAPR and give a recommendation on which modulation technique is best for OFDM in wireless networks with a lower PAPR. GnuRadio will be used together with LimeSDR as software to generate, modulate and compare results in an SDR environment. The LimeSDR will be used to capture radio data in the unlicensed Wi-Fi frequency range of 2.4GHz to 5.8GHz. Digital signal processing of this data will then be done by the GnuRadio which will implement a radio block. All OFDM parameters will be defined in the GnuRadio and the

captured data will be processed twice, firstly using BPSK and then QPSK. A code will also be developed that will interface the GnuRadio with the LimeSDR.
After successfully implementing the OFDM modulation scheme on GnuRadio a comparative analysis will be carried out comparing BPSK and QPSK in terms of PAPR. Other analyses such as BER analysis will also be carried out to give an in-depth conclusion and recommendation.

1.1 OBJECTIVES

The main objective of this project is to prototype an OFDM system using GnuRadio and perform a comparative analysis of BPSK and QPSK in terms of PAPR. This objective will be met by achieving the following sub-objectives:
a). Development of the OFDM system:

- Use LimeSDR antennas to capture radio waves
- Develop a C++ code to interface LimeSDR with GnuRadio
- Design and implement a GnuRadio block for an OFDM system that will interface with LimeSDR
- Perform an efficiency test of the OFDM GnuRadio block in terms of connectivity with LimeSDR.

b). Evaluation of BPSK and QPSK:

- Implement BPSK and QPSK on the OFDM GnuRadio block for data coming from LimeSDR
- Capture and evaluate results for both BPSK and QPSK in terms of PAPR.
- Make a comparison of the results and give a recommendation.

1.2 SCOPE AND LIMITATIONS

The scope of this thesis is limited to designing an OFDM system on GnuRadio interfaced with LimeSDR. The main focus is on the receiver side of the communication system. Data will be captured by the LimeSDR antennas which then extract information signals from the radio waves. This information is then passed to the GnuRadio where it passes through a bandpass filter to separate noise from the desired signal. This signal is then amplified and finally, information is extracted from the OFDM modulated signal. Since the agenda of this project is to perform a comparison between BPSK and QPSK in an OFDM signal, both

BPSK and QPSK will be implemented to the same data and their results will be compared in terms of PAPR and BER.

1.3 METHODOLOGY OVERVIEW

This project follows the Rapid Application Development (RAD) methodology. RAD consists of four major building blocks which are defining the requirements, prototyping, receiving feedback, and finalizing on software [11].

Requirements of the OFDM system are defined at the very beginning of the project based on a RAD methodology. A prototype will then be developed in the second stage based on the requirements of the first stage. This prototype will involve a LimeSDR interfaced to a personal computer (PC) with a GnuRadio installed on it. It will also involve developing a C++ code for the interface. GnuRadio will run on a Linux platform and it will simulate a receiver of an OFDM system. The prototype is then tested to see if it runs properly and provides feedback which is the third stage of RAD. If there are errors code is modified in the prototype stage until it runs properly. Finally, the final system will be developed based on the feedback. The code on the GnuRadio is modified to modulate the OFDM signal with both BPSK and QPSK so that results can be compared. Fig 1.3 shows the RAD cycle to be implemented in this project.

Chapter 3 will provide in detail following the RAD style on the methodology employed in this project. It will further look into user requirements, design restraints, design specifications and hardware design.

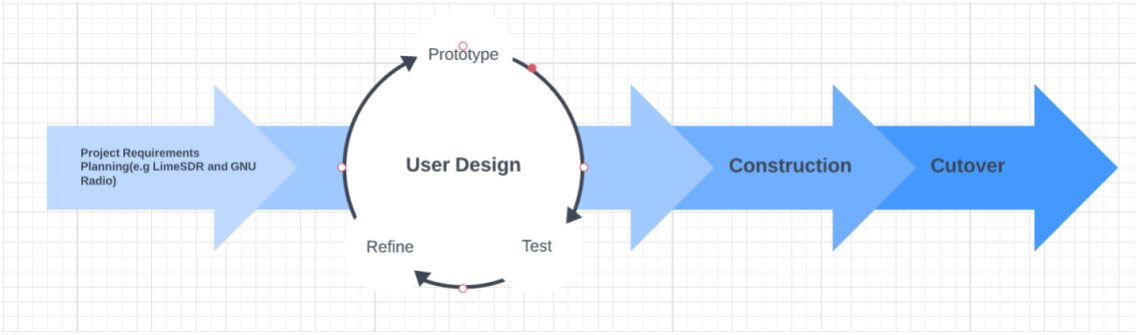


Fig 1. 3:RAD methodology cycle

1.4 THESIS OVERVIEW

This thesis consist of five chapters, a summary of each chapter is provided as follows:

Chapter 1 has already introduced the topic and has given an introduction to what this project is all about. It also gives the objectives of this project with the main objective being carrying out a comparative analysis of BPSK and QPSK modulation techniques and giving a recommendation on which one to use on NGN.

Chapter 2 gives the necessary literature review of OFDM modulation, QPSK, BPSK, LimeSDR, PAPR, and SDR. The chapter introduces SDR, defines it, and gives an overview of what SDR is, its advantages, and disadvantages. GnuRadio is then introduced and an overview of it is also given and why it was chosen as the simulation environment. GnuRadio will carry out the SDR functions like modulation and signal processing which will be explained in full in this chapter.
Furthermore, OFDM will also be introduced and explained in this chapter. Diagrams will be used to illustrate the OFDM signal that will highlight the issue of PAPR in its amplitude. Also, an extensive literature review on QPSK and BPSK will be carried out. LimeSDR is also introduced as the data acquisition platform which is an open-source SDR hardware board.

Chapter 3 gives the methodology used in this project. Guidelines used in the design of the OFDM system are provided in this chapter. User requirements, design specifications, hardware, and software design and evaluation are all done in this chapter based on a RAD approach. This chapter also provides system architecture and schematics of the LimeSDR and OFDM systems implemented on GnuRadio. Each block chosen on the GnuRadio will have an explanation of why it was chosen.

Chapter 4 explains the steps involved in the design process to come up with a functional system that will give desired results of the carried out experiments.

Chapter 5 Implements the designed system in Chapter 4 which is an OFDM system running on GnuRadio communicating wirelessly via a LimeSDR peripheral.

Chapter 6 is about capturing the results after performing the methodology in chapter 3. Results will include diagrams that will compare the amplitudes of an OFDM system with BPSK to that of QPSK. The results are then compared to see which modulation technique offers a lower PAPR and a recommendation is given.

Chapter 7 gives the conclusion and recommendations based on the results of the project.

CHAPTER 2: LITERATURE REVIEW

This chapter focuses mainly on the relevant literature that supports the build-up of this project. The first section, 2.1, will introduce the concept of SDR, its history, and its evolution through time. Many SDR platforms have been developed which include, USRP, UmTRX, Zeus ZS-1, and LimeSDR to mention but just a few. The SDR platform of choice for this project is LimeSDR and Section 2.2 will give a detailed literature review on it. The literature on OFDM, BPSK, QPSK, and PAPR will be discussed in sections 2.3, 2.4.1, 2.4.2, and 2.4.3 respectively. Based on the literature reviews comparison of BPSK and QPSK will be carried out. This comparison will give us expected results which will further be compared to actual results in Chapter 4. The last section, 2.5, will introduce GnuRadio.

2.1 OVERVIEW OF THE SDR APPROACH

The future of NGNs is based on SDR which will cater to the ever-increasing internet traffic. [12] gives a historical analysis of SDR and it is notable how previous radios were only designed to support specific waveforms with a specific function. This has however changed with the introduction of SDR as radios now support many different functions and waveforms. According to Wireless Innovation [13], Software Defined Radio is simply defined as a radio in which some or all the physical layer functions are software-defined. [13] further went on to define a radio as a device that wirelessly transmits and receives radio frequency (RF) part of the electromagnetic (EM) spectrum to facilitate information transfer. Radio technology is today included in equipment such as cellphones, computers, vehicles, and television sets. From the definition of SDR, the physical layer is a layer in the Open System of Interconnection (OSI) model and consists of the first four blocks[14]. Fig 2.1 shows an illustration of the OSI model. Also, the software-defined in the term SDR implies that different waveforms can be supported by just modifying the firmware and not the hardware.

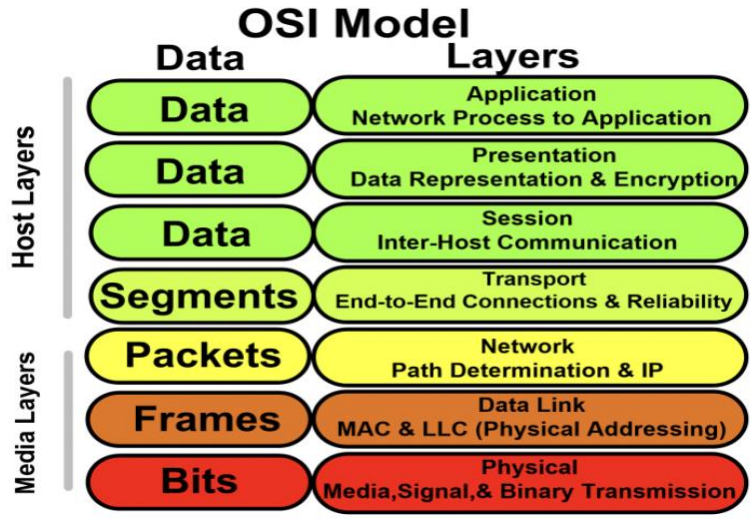


Fig 2. 1: OSI Model.

2.1.1 SDR ARCHITECTURE

An ideal SDR would be able to transmit and receive signals of varying frequency, bandwidth, power level, and modulation technique [15]. The architecture of an ideal SDR consists of a signal feed at the receiver that is then sampled, digitized by ADCs, and then digitally processed. Fig 2.2 shows the architecture of an ideal SDR. Achieving an ideal SDR is impossible and poses the following challenges:

- Maintaining a balance between a load required to process data and access to memory and avoid conflict in the processing pipelines

- Digitizing antennas for SDR communication systems would become very expensive as the antennas will have to cater for frequencies up to 60 GHz. This means that ADCs will have to sample at 120 GHz using Nyquist's sampling theorem [12].

Hence designing an ideal SDR architecture becomes impractical.

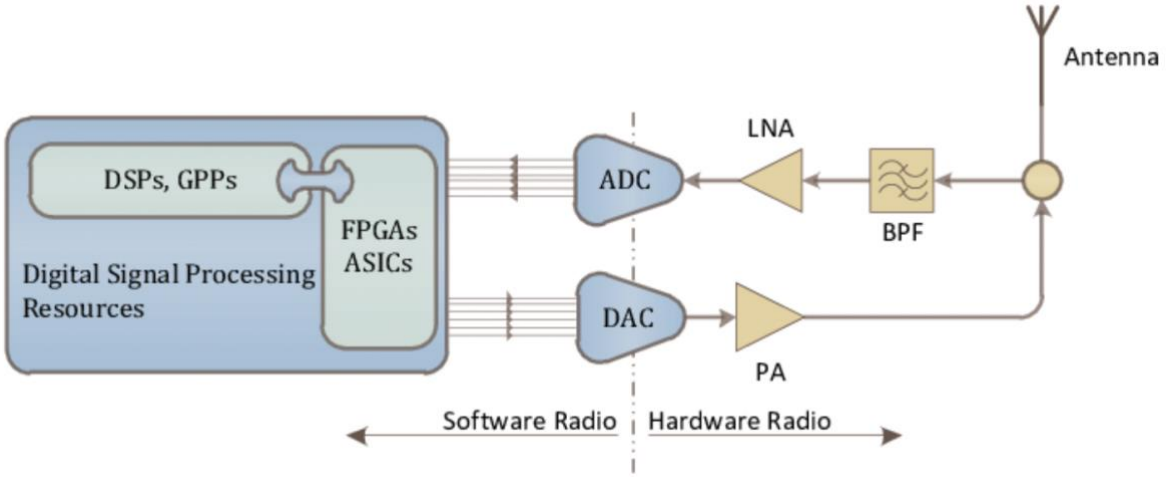


Fig 2. 2: Ideal SDR. (Image Source:[16])

Fig 2.3 shows an achievable real-world SDR architecture. This type of architecture consists of:

- ***Bandpass Filter –*** *Selects and accepts a certain frequency range and rejects all other frequencies.*
- ***Low Noise Amplifier (LNA)*** *– Amplifies a very low power signal to levels that can be processed more without degrading the SNR.*
- ***Mixer*** *– Creates new frequencies from the two frequencies fed into it.*
- ***Local Oscillator (LO)*** *– Used in heterodyning to create new frequencies together with the Mixer.*
- ***Low Pass Filter (LPF)*** *– It is a filter that only passes signals with frequencies lower than the cut-off frequency.*
- ***ADC*** *– Convert analogue signals to discrete-time digital signals.*
- ***Digital Signal Processor (DSP) –*** *Performs the necessary mathematical computations to the input signal.*

All these components are called the RF Front End (RFFE). The advantage of such an architecture is that it provides a dependable, stable, and robust approach. Such an approach makes digital baseband processing possible.

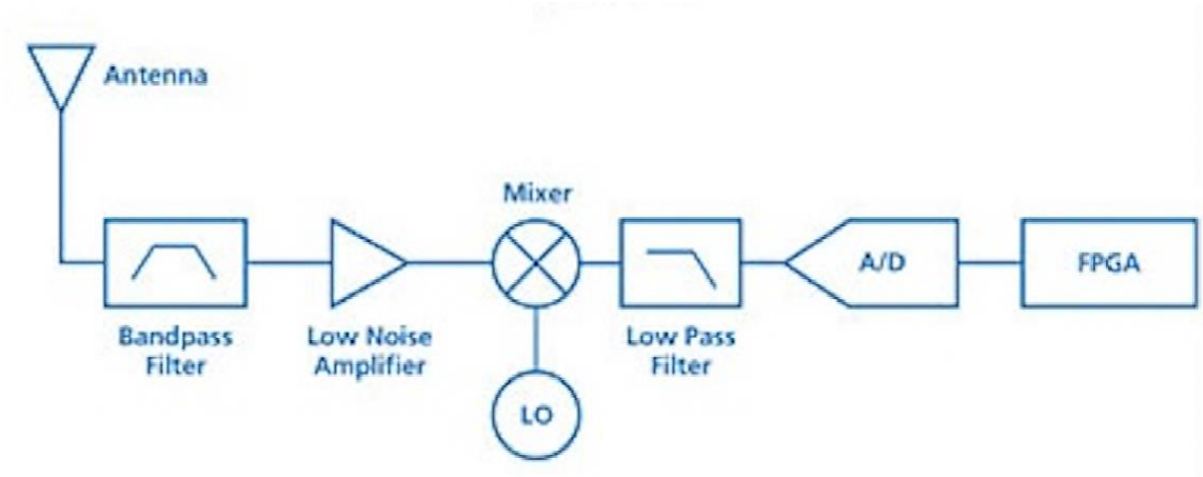


Fig 2. 3: Practical SDR Architecture.

SDRs have a hardware component responsible for sending and receiving electromagnetic signals. This is known as front-end hardware which also includes a reprogrammable general-purpose processor which does signal processing on the radio. It is this processor that differentiates SDR from hardware radios.

SDR modular design is mainly limited by the restriction of hardware components, that is, an SDR will not outperform its most limited component. SDR follows a hierarchical structure where each stage is dependent on other stages. Failure at one of the stages can cause unusual behaviour hence it is vital to make sure that there are no flaws in the SDR system. The design of SDR transmitters and receivers can be categorised into two main categories which are homodyne and superheterodyne. The superheterodyne scheme has for many years been mainly implemented in SDRs but recently homodyne transceivers are being implemented as they are more powerful. Homodyne transceivers have software radios that allow the transmission of signals via an intermediate frequency (IF) which is lower than the RF of the signal sent through electromagnetic waves. A drawback of using a superheterodyne transceiver is that it has to either step up or step down the RF of the signal outside its digital processor [17]. The differences between these two architectures are discussed below.

2.1.2 SDR RECEIVER ARCHITECTURE

Fig 2.4a shows a well-known superheterodyne receiver that operates by translating the received signal into a baseband signal using two down converters. The translated signal is then bandpass filtered and amplified. Before the signal can be processed it is converted to the digital domain. Such architecture is mainly implemented for higher-RF and millimeter-wave designs such as 5G radios and point-to-point wireless links[18]. Superheterodyne receivers

face a lot of problems when implemented for SDR solutions which include making a full-on chip integration difficult because of several fabrication technologies used. The biggest drawback of implementing superheterodyne receivers in SDR is its inability to expand for multiband reception.

Alternatively one can opt to implement a zero-IF receiver [19] which is illustrated in fig 2.4b which is a simplified architecture of superheterodyne. Zero-IF receiver accepts the whole received RF band through a bandpass filter which is then amplified by the LNA. The mixer then directly converts the signal to dc and uses an ADC to convert the dc signal to the digital domain. The main advantage this architecture has over a superheterodyne architecture is that it has fewer hardware components and its filter is less stringent in terms of specifications than the image reject filter used in a superheterodyne filter. This type of filter has its drawbacks such as dc offset which is a result of direct translation to dc [20].

A low-IF receiver is also another configuration similar to that of zero-IF architecture which functions by mixing the input RF signal down to a low or medium but non-zero IF signal. A Low-IF receiver makes use of an RF bandpass filter on the input RF signal which is then amplified. A high-performance ADC is then used to convert the signal to the digital domain. This architecture does not suffer from the drawbacks of zero-IF architecture since its signal is not converted to dc. Due to the requirement of a high conversion rate, ADC power requirement is increased and the image-frequency problem faced by superheterodyne systems is reintroduced.

Lastly, another alternative architecture is the bandpass sampling receiver which is shown in fig 2.4c. RF bandpass filter implemented for such a receiver can either be a tuneable filter or a bank of filters that act on the input RF signal. A wideband LNA then amplifies the signal, which is then sampled and converted to the digital domain by a sampling ADC. The signal is then digitised. Energy from the dc to the ADC input will return half of it to the first Nyquist zone [0, $f_s/2$] without the need for down conversion [17]. Bandpass sampling architecture utilises the properties of the sample and holds circuits. The resulting IF, f_{IF}, can be pinpointed by the following relationship [21]:

$$\text{If } fix\left(\frac{f_c}{f_s/2}\right) \text{ is } \begin{cases} even, & f_{IF} = rem(f_c, f_s) \\ odd, & f_{IF} = f_s - rem(f_c, f_s)' \end{cases} \quad \text{.....(1)}$$

f_c - Carrier frequency,

f_s - sampling frequency,

fix(x) - truncated portion of argument *x*,

rem(x,y)- remainder after division of *x* by *y*.

The vital role played by the bandpass filter is signal energy reduction which is not included in the Nyquist zone of the desired frequency band. If such signal is not filtered it is halved back to the first zone together with the desired signal hence a degraded SNR which is given by:

$$SNR = 10^{*}\log_{10}\left(\frac{S}{N_i+(n-1)^{*}N_0}\right), \qquad \text{.....(2)}$$

S – Desired Signal Power,

N_i – In-band noise,

N_o – Out-band noise,

n – Number of aliased Nyquist zones.

Bandpass sampling architecture reduces the number of components because the sampling frequency, f_s, and its processing rate are directly proportional to the carrier frequency, f_c.

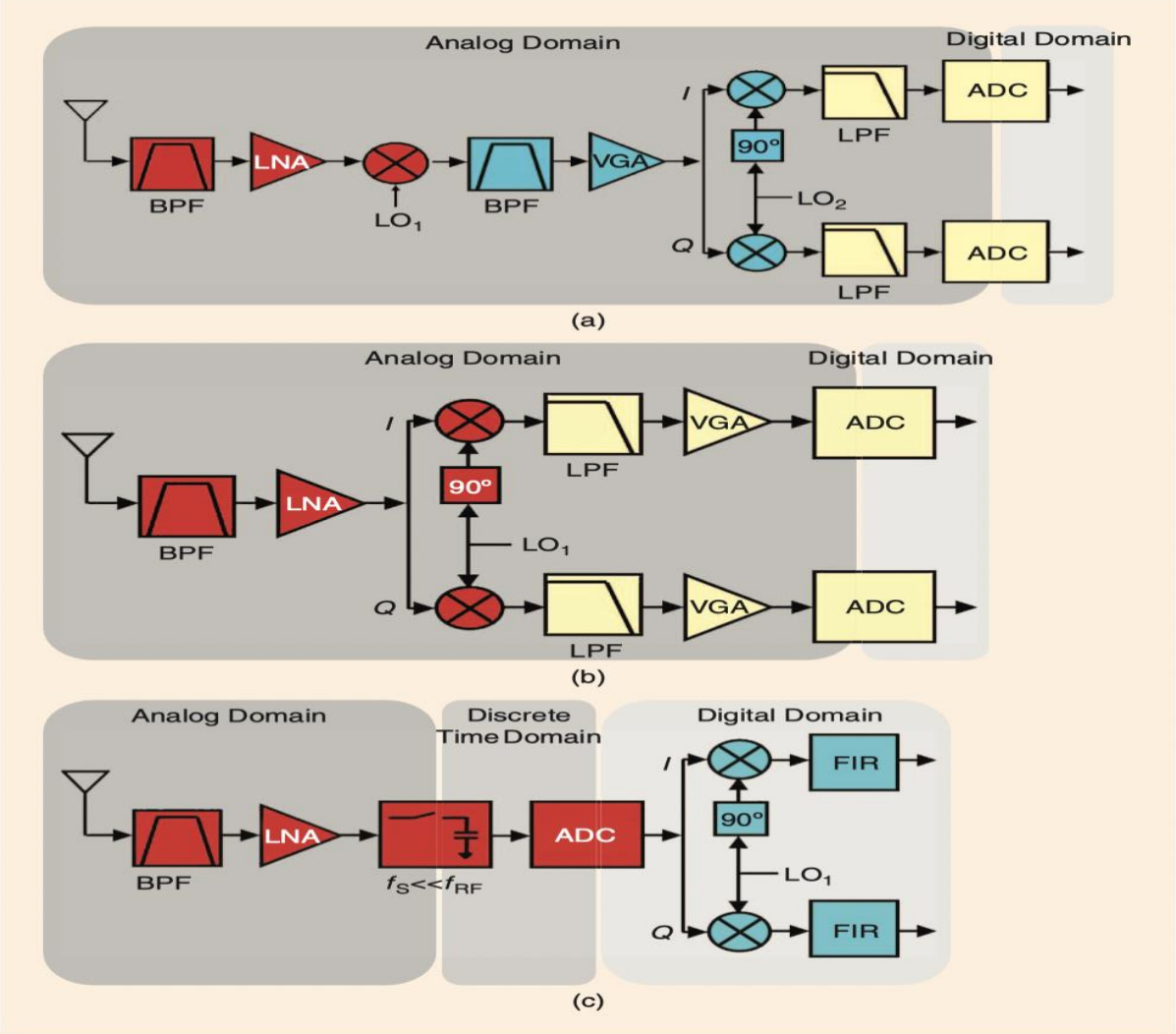


Fig 2. 4: (a) superheterodyne receiver (b) zero-IF receiver (c) bandpass sampling receiver. (Image Source:[17])

2.1.3 SDR TRANSMITTER ARCHITECTURE

The Front End

The transmitter is not only the power amplifier (PA) but a collection of several circuitry components which give it a collective name, the front end. The most important aspect of a transmitter design is the PA which has a direct link to coverage, product cost, and power consumption of a wireless communication system. Two types of transmitter architectures will be discussed, a superheterodyne transmitter and a direct conversion architecture. Thereafter, the PA will be discussed.

The first architecture to be discussed is the one depicted in fig 2.5a which is a superheterodyne transmitter that works hand in hand with the superheterodyne receiver in fig 2.4a. The digital domain is used to create the signal, which is then converted to the analogue domain using DACs. An IF is then applied during modulation which then leads the signal to be amplified and filtered to remove noise that was a result of modulation. The signal then passes through a local oscillator (LO_2) which upconverts the signal to RF. Unwanted image sidebands are then filtered, then an RF power amplifier is used to amplify the signal to the transmit antenna. This architecture makes use of an I/Q modulator instead of an RF-based modulator hence hardware components are easier to design if IF is to be implemented. Again, at IF, high-quality variable amplifiers can be designed hence the overall gain can be controlled. This architecture, however, is far from perfect. Its drawbacks include difficulties in implementing the multimode operation. Its operations are limited to mostly microwave point-to-point communication.

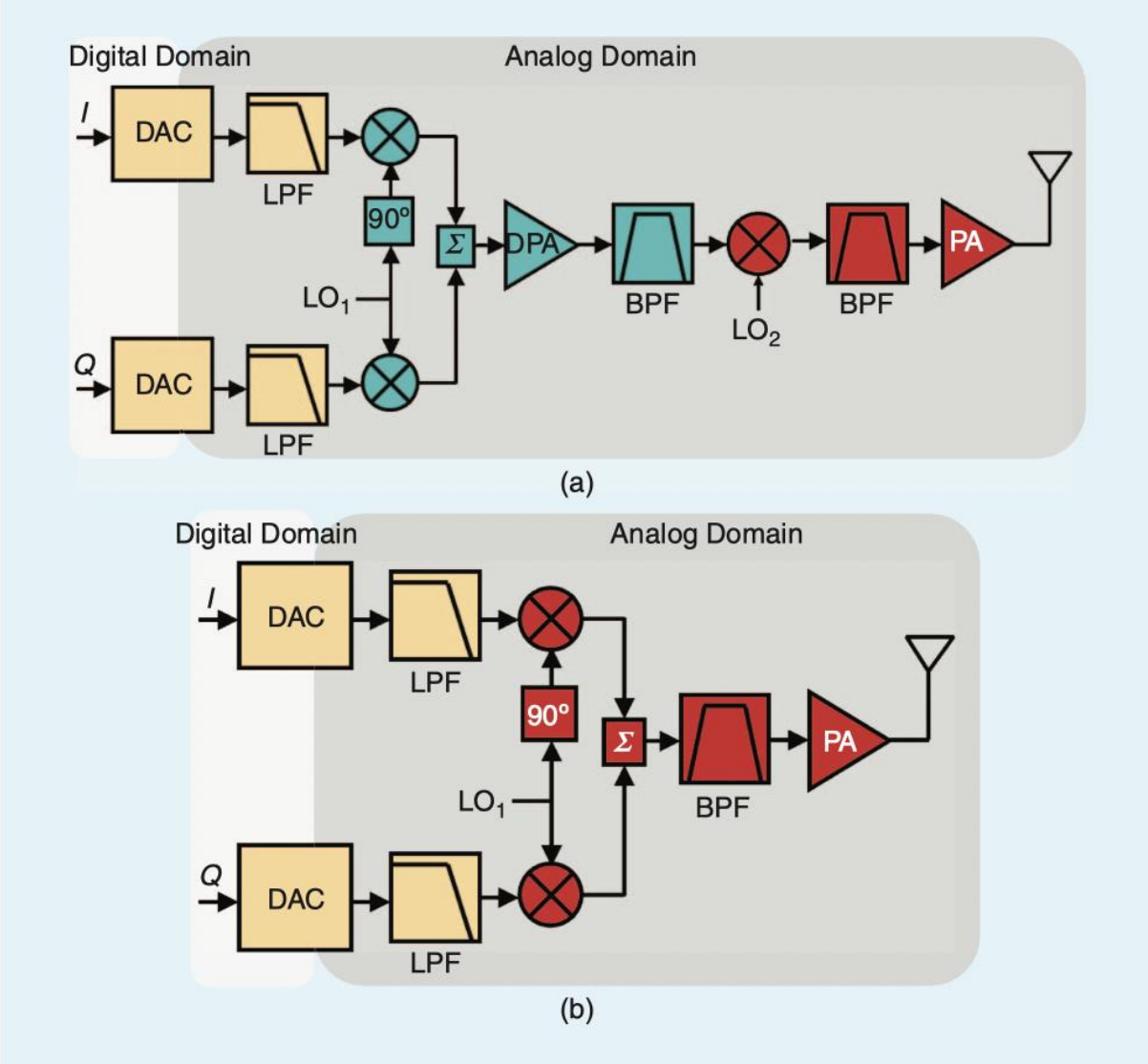


Fig 2. 5: (a) Superheterodyne Transmitter (b) A Direct Conversion Architecture. (Image Source: [17])

The second transmitter architecture to be discussed is the direct conversion transmitter which is depicted in fig 2.5b. This architecture makes use of two DACs to convert the signal to the digital I and Q signals to the analogue domain [22], [23]. An LPF is then used to discard Nyquist images and reduce the SNR. High-performance I/Q modulators are then used to directly modulate the signal at the RF. A bandpass filter tuned at the output frequency is then used to filter the signal which is then amplified by the power amplifier. This architecture reduces the amount of circuitry required and gives room for high-level integration. Its drawbacks include phase gain mismatch and possible carrier leakage. To get the best performance of a direct conversion transmitter, at the RF the gain must be controlled and the use of a high linear PA may be necessary.

2.1.4 PROS AND CONS – SDR

There are many advantages to adopting SDR in modern communication systems but at the same time, it comes with its setbacks such as the cost of design and implementation of SDR transceivers. In this section advantages and disadvantages of SDR will be discussed.

Pros of SDR

- **Interoperability –** The biggest advantage of an SDR system is that it can seamlessly communicate with incompatible devices. It is for this reason that the US military funds the SDR project for the past 30 years [12].
- **Resource Utilisation –** SDR systems can efficiently use their resources, even under varying conditions. In addition to this, a larger RF spectrum can also be utilized.
- **Frequency Reuse (Cognitive Radio) –** SDR systems can take advantage of the underutilized frequency spectrum, that is, when the owner of a certain spectrum is not using it, it releases the spectrum and SDR can borrow it until the owner needs it back.
- **Scalable –** SDR systems are highly scalable and can support the latest communication standards (Future-proofing).
- **Lower Cost –** SDR can be implemented in different markets with the same design hence reducing the cost to design different devices that are the same radio used in the mobile telecommunications sector can be used in the mining or automotive sector.
- **Simple Architecture –** SDR systems have fewer analogue parts as compared to traditional radios.

In conclusion, SDR's ability to change its hardware capabilities presents several opportunities. SDRs can be easily modified and implement different physical layer protocols which cannot be done in traditional radios [24]. This can be achieved by simple code editing without the engineer changing the hardware. Therefore, instead of developing modules of different radio protocols, one will only need a different software that supports the protocol. This reduces the physical complexity, size, and overall cost of the development and maintenance of a radio.

To some extent, SDR implementation is cheaper than having dedicated hardware in some aspects. For example, when vendor-based radio comes to its End of Life (EoL) it becomes a useless piece of hardware, unlike SDR which can be easily upgraded to support new protocols hence reducing the cost of developing new radios.

SDRs can easily test and implement communication standards as compared to traditional hardware-based radios. With traditional radios, when new radio protocols are developed, new electrical circuits must be developed for each test done and when the protocol is introduced it comes with new hardware. With SDR, the new protocol can be tested and implemented on the same platform which becomes simpler, quicker, and cheaper.
SDRs will be critical in the implementation of cognitive radios. Cognitive radios will be critical in spectrum utilisation as they can borrow unused spectrum and return it once it is needed. This will be an effective way to manage the spectrum which is a limited and scarce resource.

Cons of SDR

SDRs are not perfect radios as they come with their disadvantages. It is these disadvantages that will make traditional radios stick for a while. Some of the challenges that come with adopting SDR are defined below.

- **Cost and Power –** The cost that comes with the design and implementation of SDR systems is a major concern. Some applications will require less complex radios and an upgrade of such radio is less unlikely hence makes sense to use traditional radios. DSP for a wide range of RF bandwidth required in SDR systems consumes a lot of power. The power used by FPGA/GPPs is ten times that of Application-Specific Integrated Circuits (ASICs) which are used in traditional radios [12]. In addition, RFFE ADC/DAC implemented in SDR architecture consumes a lot of power hence increasing the cost of design.
- **Complexity –** It takes time and cost to implement an SDR radio because of the engineering effort needed to develop software and firmware needed to support a wide range of waveforms. In addition to this, an SDR has to support a set of baseline waveforms but also anticipate additional waveforms [12]. Hence some DSP resource margin must be able to support future waveforms. Therefore so much complexity is involved in the design and implementation of SDR.
- **Limited Scope –** SDR only addresses physical layer issues. Users cannot take advantage of link throughput improvements made by SDR which only addresses physical layer issues.

2.2 LIMESDR

LimeSDR, shown in fig 2.6, is a low-cost software-defined radio platform that is based on Lime Microsystems' second-generation FPRF transceiver LMS7002M, Altera Cyclone IV FPGA, and Cypress FX3 USB3 Superspeed microcontroller [25].

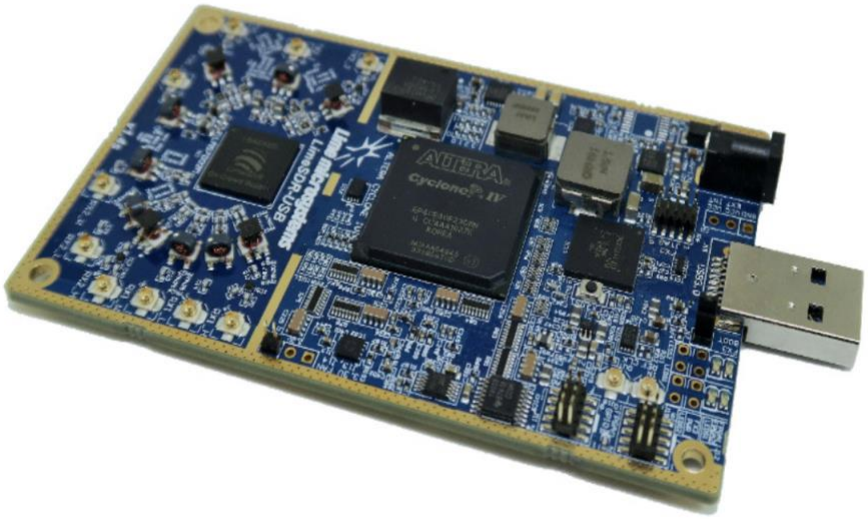


Fig 2. 6: LimeSDR

Fig 2.7 shows a block diagram of a LimeSDR shown in fig 2.6. Its architecture includes a direct conversion quadrature transceiver, ADCs and DACs, and a transceiver signal processor (TSP). LimeSDR makes it possible to make all the blocks fully configurable hence giving researchers the ability to optimise chip configuration as per application requirements. Use case scenarios for reconfigurability include routing of down-converted signals off-chip such that custom filters can be used and further allow routing back on-chip for analogue to digital conversion and digital processing[25]. Since off-chip routing is software-defined, operation modes can be interchanged.

The authors of [25] give another application example of LimeSDR which can be internally reconfigured for frequency division duplex (FDD) or time division duplex (TDD). The transmit (TX) and receive (RX) in FDD mode is set at different frequencies synthesised by two on-chip phased locked loops (PLLs) whereas, in TDD mode, TX and RX operate at the same frequency. The LimeSDR uses one PLL for both TX and RX when operating in TDD mode and the other PLL will be turned off.

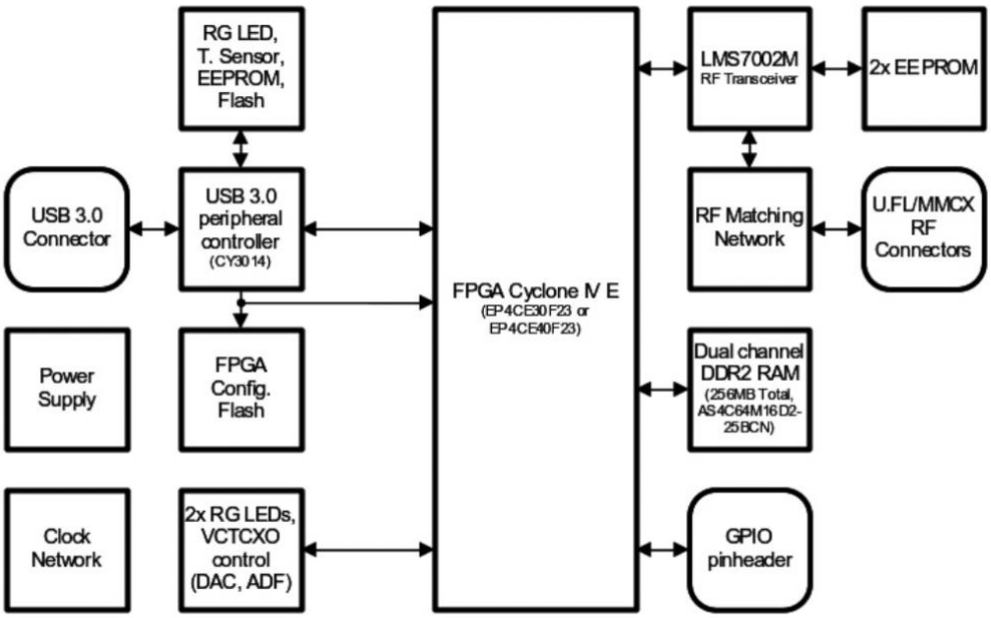
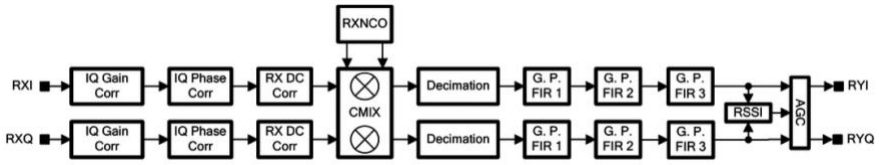
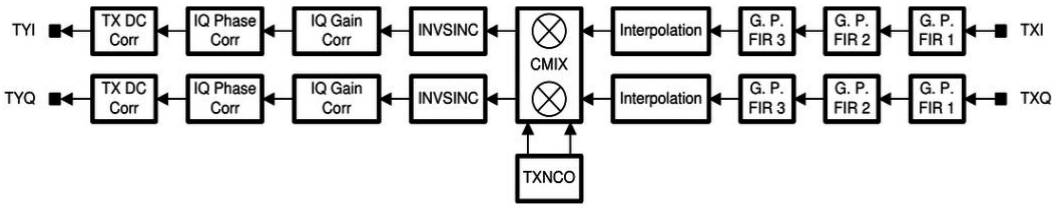


Fig 2. 7: Block diagram of LimeSDR. (Image Source: [26])

Fig 2.8 illustrates a block diagram of signal processing used in LimeSDR. Three general purpose filters are contained in the TX and RX path. These are put together in a cascade with a numerical controller oscillator (NCO), complex mixer, and I/Q correction blocks [25]. The signal is first upsampled by the interpolator contained in the TX before it is passed on to the complex mixer. The programmable decimator contained in the RX will be used to downsample the received signal before filtering. The main reason to perform upsampling and downsampling is to reduce computational load and the digital interface clock rate. All in all, this reduces the overall power consumption of the LimeSDR. The received signal indicator in the RX can be used to measure the received signal power hence no FPGA resources are required.



a.



b.

Fig 2. 8: a)RxTSP b)TxTSP. (Image Source: [27])

In this project, the LimeSDR will be used to perform RF measurements. As explained in this section, LimeSDR hardware is affordable as compared to other SDR platforms on the market and have the capability to transmit and receive on a single LimeSDR making it suitable for the experiments carried out in this project.

2.3 ORTHOGONAL FREQUENCY DIVISION MULTIPLEXING

The principal operation of an OFDM system is to utilise its wireless transmission technique in which a digital signal is encoded on multiple carrier frequencies. One high data rate frequency is used after combining multiple low data frequency channels. Phase Shift Keying (PSK) is an important aspect of OFDM as this is used to generate the modulated data to a time signal. A huge requirement for an OFDM system is that it must be linear. Non-linearity can cause inter-carrier interference which has a huge impact on transmission. Fig 2.9 shows a high-level design of an OFDM system implemented in this project.

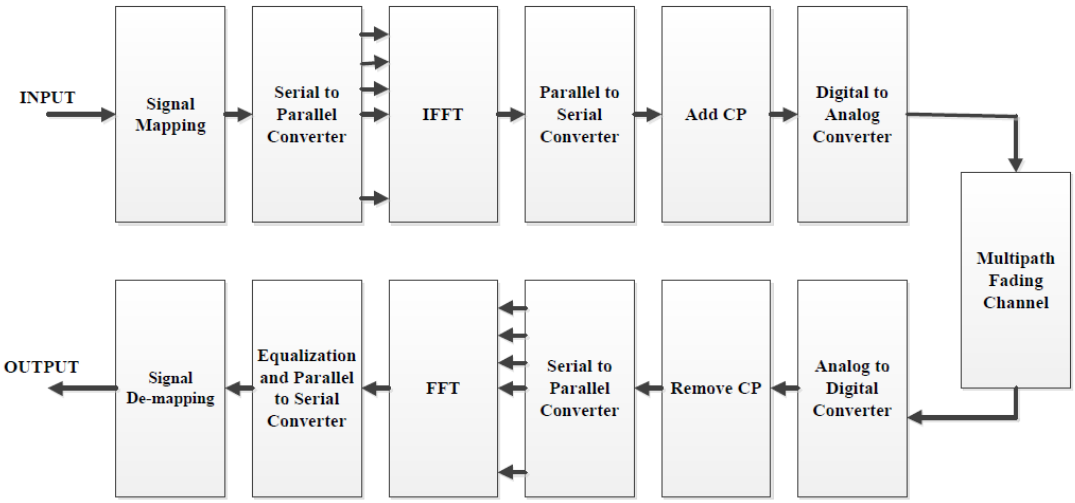


Fig 2. 9: Receiver and Transmitter of an OFDM system

With an ever-increasing need for high wireless data transmission rates, there is a need for technologies that utilise available electromagnetic resources as they are limited. Such technologies should have desirable features such as low power consumption, long-range, ease of implementation, high spectral efficiency, and robustness against multipath propagation. Technologies with all the above features are only ideal as those features are often conflicting hence a practical technology offers the best trade-off. With the emergency of the Fourth Industrial Revolution (4IR), which is being pushed by the internet of things (IoT), there is a need for a modulation technique that will support the high data rates, and currently, OFDM is the most suitable choice. The main advantage of an OFDM system is its ability to withstand frequency selective fading commonly referred to as narrowband interference. OFDM is a multicarrier system, hence a single fade will only cause a fraction of the subcarriers to fail whereas a fade in a single carrier system can cause the whole link to fail. Another advantage of a multicarrier system is its ability to correct erroneous subcarriers using the different types of error correction coding techniques. OFDM is a type of multicarrier modulation that splits the original signal into several signals which are then modulated to several different channels. The receiver will then combine the received signals from the multiple channels used [28]. Multi-channels implemented in an OFDM system are orthogonal to each other.

2.3.1 BASICS OF OFDM

Say :

$$\{sn, k\}^{N-1}{}_{k=0}$$

$$with\ \ E|s_{n,k}|^2 = \sigma^2\ \ is\ the\ signal\ to\ be\ transmitted\ at\ the\ n^{th}\ OFDM$$

Then, the OFDM modulated signal is represented by the following formula:

$$s_n(t) = \sum_{k=0}^{N-1} S_{n,k} e^{j2\pi k\Delta f t},\ \ 0 \le t \le T_s \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots (1)$$

T_s - Symbol duration

Δf – Sub-channel space

N – Number of subchannels

To demodulate such a signal, T_s should be long enough such that $T_s\Delta f = 1$. Such a condition is called the orthogonal condition because it makes $e^{-j2\pi k\delta f t}$ orthogonal to each other for varying *k*. If this condition is met, then the transmitted signal $s_{n,k}$ can be detected at the receiver by:

$$s_{n,k} = \frac{1}{T_s} \int_0^{T_s} s_n(t) e^{-j2\pi k\Delta f t} dt \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots (2)$$

If and only if there is no channel distortion.

The baseband OFDM signal *s(t)* goes through sampling and its version can be represented as:

$$s_n\left(m\frac{T_s}{N}\right) = \sum_{k=0}^{N-1} s_{n,k} e^{j2\pi k\Delta f m\frac{T_s}{N}} = \sum_{k=0}^{N-1} s_{n,k} e^{j\frac{2\pi mk}{N}} \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots (3)$$

This is the inverse discrete Fourier transform (IDFT) of $\{sn, k\}^{N-1}{}_{k=0}$. This can be efficiently calculated using a fast Fourier transform (FFT).

It is also important to avoid inter-block interference (IBI) in OFDM systems which is a result of delay spread in wireless systems. Therefore it is vital to look at the cyclic prefix (CP) which is important in OFDM systems as far as IBI is concerned.

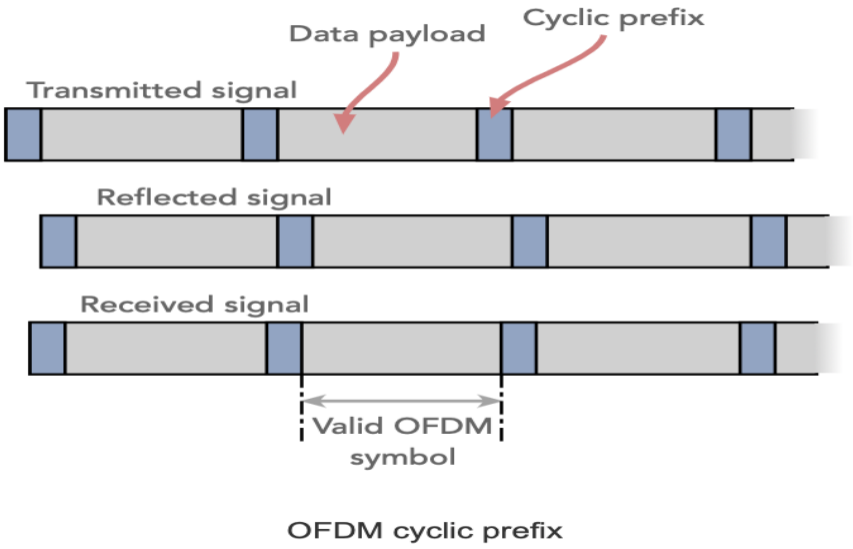


Fig 2. 10: OFDM CP.

The function of CP is well illustrated in fig 2.10. The symbol, T_s, represents the length of the OFDM signal without the CP. When the CP is added to the OFDM signal, the new length of the transmitted signal becomes $T = T_g + T_s$ and its expression is as follows:

$$\widetilde{s_n}(t) = \sum_{k=0}^{N-1} s_{n,k}\, e^{j2\pi k\Delta f t},\ -T_g \leq t \leq T_s \quad \text{.....(4)}$$

For the following condition, $-T_g \leq t \leq 0 \ \ then \ \ \widetilde{s_n}(t) = s_n(t + T_s)$. It is because of this reason it's called the cyclic prefix.

For a wireless channel, the impulse response is represented by [29]:

$$h(t) = \sum_t \gamma_i\, \delta(t - \tau_i), \quad \text{.....(5)}$$

where $\tau_i \ \ and \ \gamma_i$ represent the delay and complex amplitude of the *i*th path respectively. From this equation, the received signal is derived to:

$$x_n(t) = \sum_i \gamma_i \bar{s}_n\,(t - \tau_i) + n(t) \quad \text{.....(6)}$$

n(t) is the additive white gaussian noise (AWGN) at the receiver. For a *k*th channel, the impulse response is then represented by:

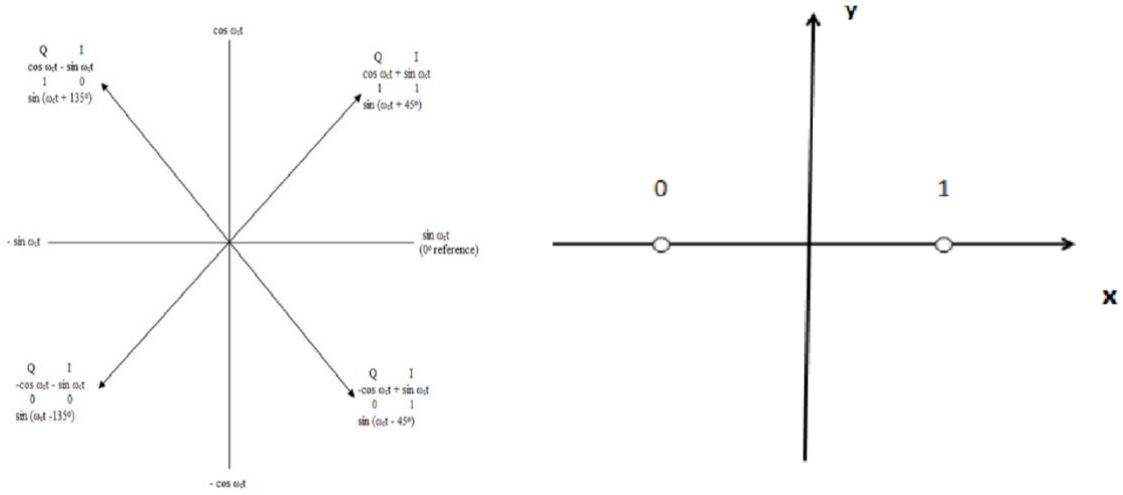
$$H_k = \sum_i \gamma_i\, e^{-j2\pi k\Delta f \tau_i} \quad \text{.....(7)}$$

With the above equation, it can be shown that *nk* are independent identically distributed complex gaussian with zero mean and variance $\sigma^2 n$ [30]. The significance of H_k is that an estimation of transmitted signals can be made. Signal detection in the OFDM system is simple hence one of the most popular techniques for wireless modulation.

2.4 MODULATION AND DEMODULATION

OFDM implements digital phase modulation which includes BPSK, QPSK, and DPSK. Digital phase modulation involves adding the phase of the carrier wave with one of the information bits to convey data.

- **BPSK –** 1 symbol and 1-bit composition. The level of the 1 and 0 information signal is changed to a 1 or -1 bipolar NRZ signal and the phase of the carrier wave is given a value of 0 or π.
- **QPSK –** 1 symbol 2-bit composition. The carrier wave phases $\frac{\pi}{4}, \frac{3\pi}{4}, \frac{5\pi}{4}, and\ \frac{7\pi}{4}$ are assigned with 2 bits and 4 statuses. Hence, PSK with 8 or 16 values is possible.



a. b.

Fig 2. 11: Constellation diagram of a) QPSK b) BPSK

2.4.1 BPSK MODULATION AND SYSTEM DESIGN

Firstly, in BPSK modulation the modulating signal (P(t)) is converted to a dipolar non-return to zero signal, which is then mixed with the carrier wave (C(t))in the mixer where the multiplication process occurs. The multiplication process shifts P(t) directly to C(t). The phase of C(t) is then changed so that it becomes proportional to the message of the signal. The carrier wave is a sine function that has the following equation:

$$C(t) = A_c . \cos(2\pi . F_c . t + \phi_c)$$

According to the above equation and behind the principle of BPSK the C(t)'s phase is either changed by 0 or 180 degrees. The resulting formula at the output will be as follows:

$$S_{psk}(t) = A_c.\cos\left[2\pi.F_c.t + \pi.m(t)\right]$$

And when $\emptyset c$ = 0, that is the phase of the carrier is equal to zero.

$$S_{psk}(t) = A_c.\cos(2\pi.F_c.t)\cos\left(\pi.m(t)\right)$$

From the above equation we can note that m(t) is our message hence from this we know that P(t) = cos($\pi m(t)$). Hence,

$$S_{psk}(t) = P(t).A_c.\cos(2\pi.F_c.t)$$

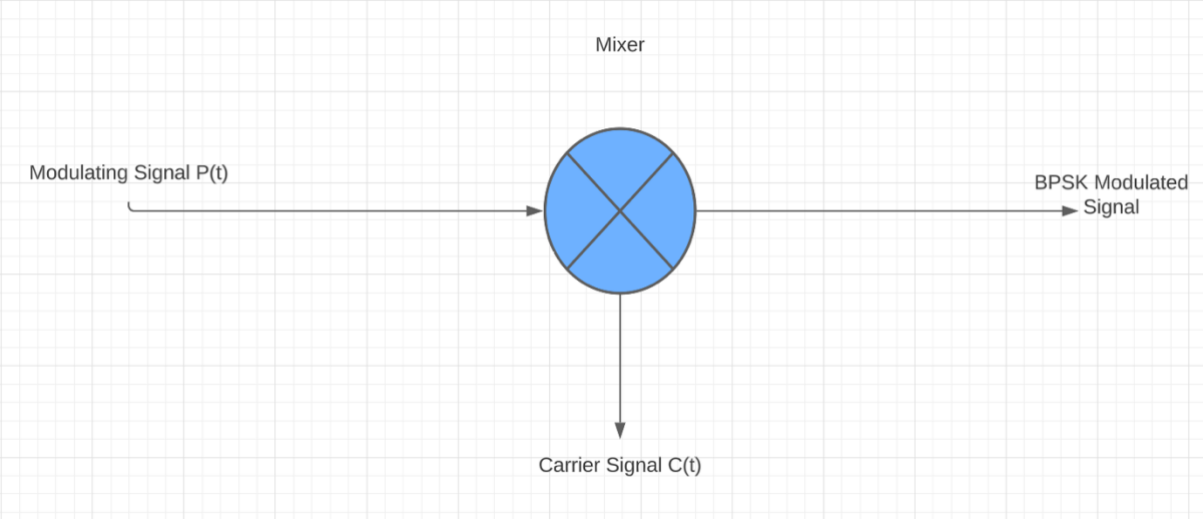


Fig 2. 12:BPSK modulator

2.4.2 BPSK DEMODULATION AND SYSTEM DESIGN

Making use of a synchronous detection system is a more efficient way of performing demodulation is BPSK. It makes use of a demodulator to multiply S(t) to regenerate C(t). At this stage, the regenerate C(t) must be synchronous with the C(t) at the transmitter. Lack of synchronization can cause varying amplitude levels hence risking errors. After regeneration our equation becomes :

$$S_{psk}(t).C(t) = P(t).A_c.cos^2(2\pi.F_c.t) = P(t).A_c\frac{1}{2}[1+\cos(4\pi.F_c.t)]$$

A low pass filter will then be used to filter out unwanted noise which is the second part of the equation. Therefore, at the receiver the output will be :

$$< S_{psk}(t).C(t) >_{LPF} = \frac{A_c}{2}P(t)$$

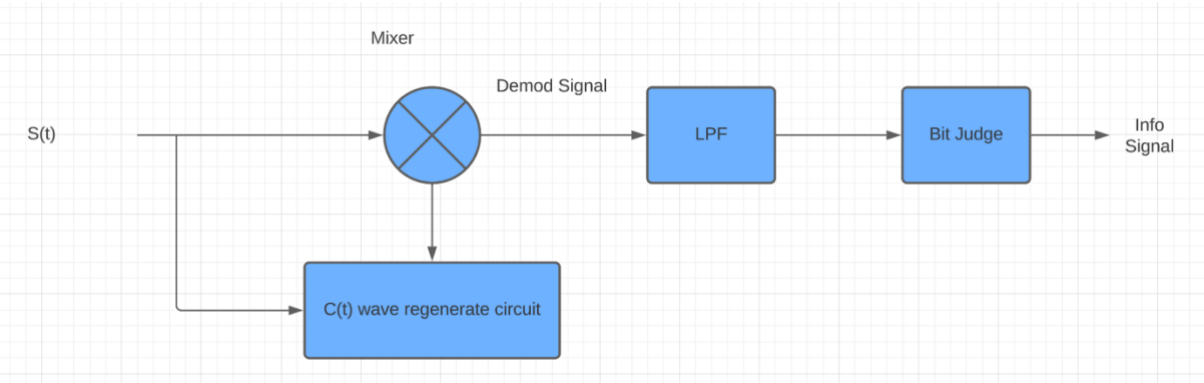


Fig 2. 13: BPSK demodulator

2.4.3 QPSK MODULATION

QPSK works differently from BPSK as 1 symbol can be used to represent 2 bits of information to be transmitted. QPSK assigns information in phases of π/4, 3 π/4, -π/4, and –3π/4 assigned directly to 00, 01, 10, and 11 respectively. QPSK is easily shown in an I and Q carrier wave as there is a change in phase between the message to be transmitted to the carrier wave. QPSK makes use of two carrier waves that follow the same properties of C(t). For this project, I defined the carrier waves as Ci(t) and Cq(t). Fig 4 shows a QPSK modulator that performs the multiplication of I-Q components of the dipolar modulating signals. The modulated information signal is mixed and multiplied with carrier waves of the I and Q components in the mixer. The P/S converter divides the information signal into the I and Q components. An adder will then be used to combine the two I and Q modulated signals to form our QPSK wave (S).

Important equations in QPSK include:

$$C_i(t) = A_c.\cos(2\pi.F_c.t)$$

$$C_q(t) = A_c.\sin(2\pi.F_c.t)$$

$$S = P_i(t).C_i(t) + P_q(t).C_q(t) = P_i(t).A_c.\cos(2\pi.F_c.t) + P_q(t).A_c.\sin(2\pi.F_c.t)$$

S represents the QPSK waveform which is a result of mixing two orthogonal basis functions which are $C_i(t)$ and $C_q(t)$. Before they are added together they are multiplied by the in-phase quadrature points $P_i(t)$ and $P_q(t)$ respectively, which are generated by the signal converter.

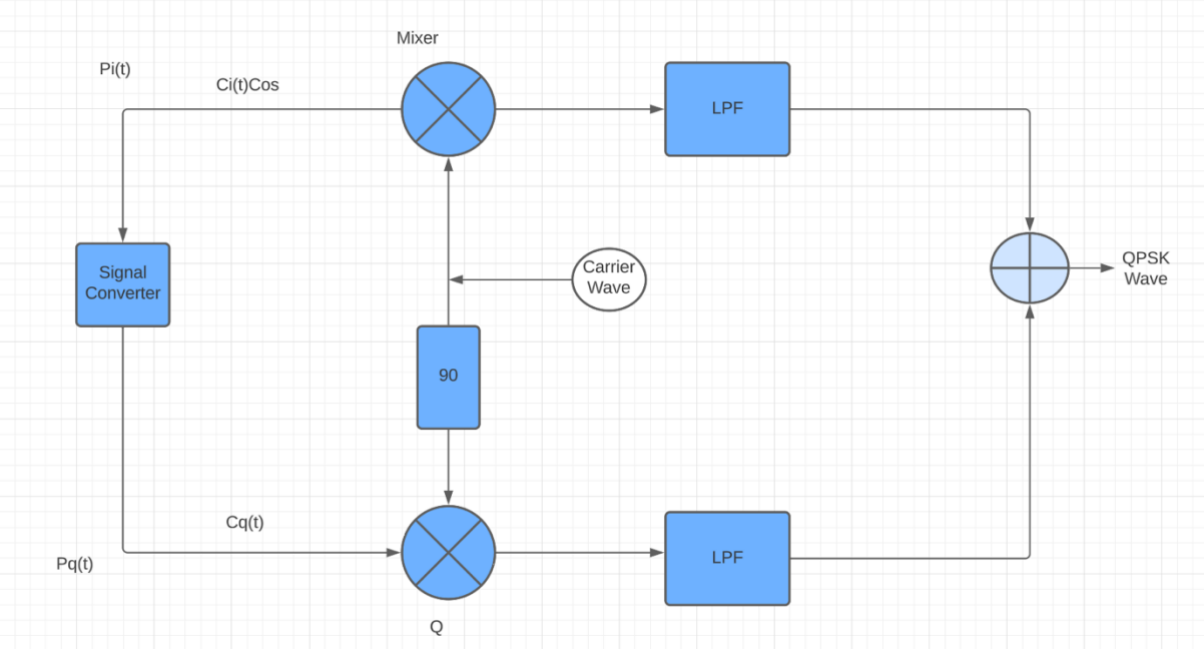


Fig 2. 14: QPSK Modulator

2.4.4 QPSK DEMODULATION

Demodulation is the reverse process of modulation. The QPSK demodulator makes use of reference frequency generators. The frequency generators make use of *cos(ωt)* and *sin(ωt)* functions to separate S. Two product detectors are used to demodulate the QPSK modulated wave. One detector makes use of the sine function whereas the other makes use of the cosine function. Low pass filters are then used to filter out unwanted noise. The architecture of QPSK demodulation is shown in fig 2.15.

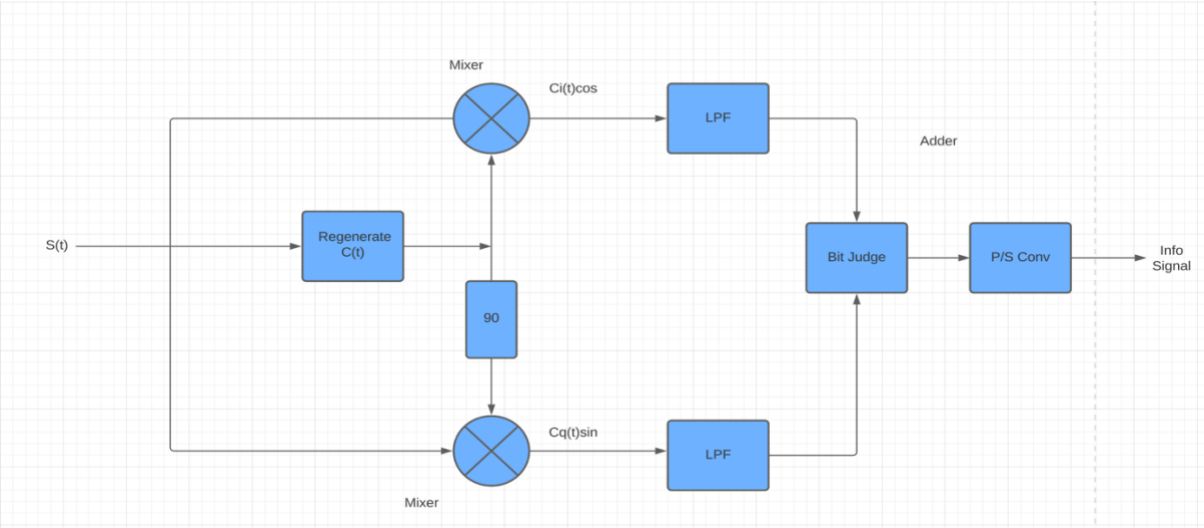


Fig 2. 15: QPSK Demodulation.

2.2.5 PAPR

The Peak-to-Average Power Ratio (PAPR) of an OFDM signal is a measure of the ratio of the maximum power in the signal to the average power in the signal. The general equation for calculating the PAPR of an OFDM signal can be expressed as follows:

$$\text{PAPR} = \frac{P_{max}^2}{P_{avg}}$$

Where Pmax is the maximum instantaneous power of the OFDM signal, and Pavg is the average power of the OFDM signal.

Another way to express the PAPR is to use the complementary cumulative distribution function (CCDF) of the signal envelope. The CCDF is a measure of the probability that the envelope of the OFDM signal exceeds a certain level. The PAPR can be defined in terms of the CCDF as follows:

$$\text{PAPR} = \frac{A^2}{E[|x(n)|^2]}$$

Where A is the amplitude at which the CCDF is evaluated, and $E[|x(n)|^2]$ is the expected value of the squared magnitude of the OFDM signal.

These equations provide a general definition of the PAPR of an OFDM signal, which can be used to evaluate the performance of the signal and optimize the system design.

2.5 GNURADIO

GnuRadio is the choice SDR platform for this project because of its rich resources and many communication blocks that are needed in a wireless communication system. This platform has been chosen over many other platforms such as powerSDR, SDR Sharp, and OSSIE to mention but just a few. All these platforms have a common advantage in that they are open-source. GnuRadio is a software-defined radio platform that is used to design, simulate and deploy real-world radio communication systems [31]. The platform is highly modular and comes with a comprehensive library of processing blocks. Its applications include mobile communications, radar systems, audio processing, GSM networks, and tracking satellites to mention but just a few.

As mentioned in the SDR section, traditional methods of developing radios, required circuitry development that would capture only a specific frequency range and this circuitry was only able to encode and decode this transmission class which made hardware costly. GnuRadio's framework makes it possible to write signal processing applications for commodity computers [31]. It offers reusable blocks which are easy to use, SDR solutions that are

scalable, a comprehensive library of standard algorithms, and its platform is open-source hence many people can contribute to the code hence making it better.

2.5.1 SIGNAL PROCESSING IN GNURADIO

General computers are used by GnuRadio, to generate communication functionality on digital signals. An example of signal generation is when an individual speaks over the microphone of a mobile phone, they create a sound signal which are waves of varying air pressure generated by the human vocal codes [31]. At the microphone, the waves are then converted into an electrical signal which is of variable voltage. At this stage the signal is analogue and computers cannot understand it hence it has to be converted into a digital format for computational purposes. This conversion will mean that:

- The signal will be of limited values only.
- The signals can only be defined for a non-infinite period.

The conversion from analog to digital is done by ADC and a conversion back to analogue is done by a DAC. The digital signal obtained from the ADC is a sequence of samples, which are continuous measurements of the analog signal. These samples are then quantized into binary bits to generate a digital representation of the analog signal. The sampling rate is the fixed period between samples. It is this digital format that most computers can understand and can perform many computational operations on the signal. Such computations include compression, filtering, signal transmission, and speech recognition to mention but just a few. The above example can be applied to radio waves where an antenna converts signals in form of electromagnetic waves to a time-varying voltage. The signal is then mixed with a carrier frequency which will make it possible to transmit the signal over a distance to the receiver.

GnuRadio offers a modular, flow-based approach to digital signal processing. It makes use of processing blocks that are connected by a flow indicating arrow as shown in fig 2.16. Each processing stage such as filtering and analysis is represented by a block and the block performs the computation required of it using the theory explained in this chapter. GnuRadio also allows individuals to write their blocks if it is not included in the library.

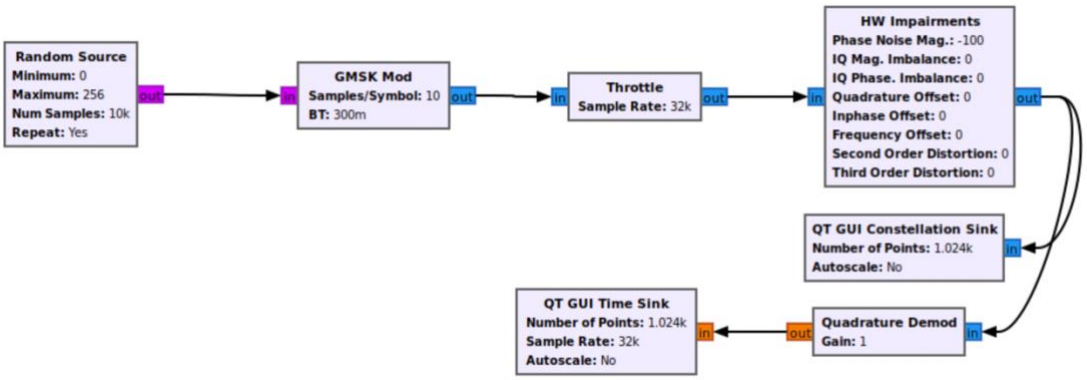


Fig 2. 16: GnuRadio Flow-based approach e.g. GMSK Mod

CHAPTER 3: METHODOLOGY

This chapter presents the methodology used in the design process of the LimeSDR and how it will be linked to the GnuRadio. The first part will look at the methodology adopted for this project, then items such as user requirements, and design restraints will be discussed. In addition to all this, Chapter 3 will be built upon the literature review given in Chapter 2.

3.1 METHODOLOGY

As stated in the introduction of this chapter, its main aim is to provide the guidelines used in the design process of the OFDM system. The focus will be on:

- User Requirements and design restraints
- System design
- Hardware Design
- Software design process
- Environment setup and system evaluation

To kick off the chapter, the user requirements, and the design restraints of LimeSDR are discussed in detail. To have a robust system, its requirements must be well defined and stipulated hence the design process of the LimeSDR interface to the GnuRadio will be detailed in this section. The hardware used is also an important aspect of this project hence will also be defined in this chapter. The software design process will also be defined, followed up with the environment setup and system evaluation.

3.2 USER REQUIREMENTS

When designing a system, there are mainly two requirements involved which are: functional and non-functional requirements. Functional requirements will include:

- Set the LimeSDR to only function in the unregulated frequency range (2400 to 2483.5) MHz
- GnuRadio to receive data as a baseband signal
- There must be compatibility between GnuRadio and LimeSDR
- The designed OFDM system must be able to interchange between BPSK and QPSK
- The system must be able to capture and analyse results.
- PC host running Linux
- LimeSDR

- GnuRadio software running on Linux

It is required to interface the LimeSDR with the host PC which will have Linux as its base operating system. The PC will have GnuRadio installed which will have an OFDM system setup. The LimeSDR communicates with the GnuRadio via the USB port on the PC.
Non-Functional requirements will include:

- The system should provide a high throughput of at least 2Mbps
- The system must have low latency
- The final system should be user friendly and must be reliable

Non-functional requirements are also vital in system design as they ensure that the system will run smoothly with minimum data loss and latency delays. Another important non-functional requirement is configurability which should be possible on the host PC (This aspect will be discussed in environment setup). Host PC should recognize the LimeSDR platform, and the GUI of the software-defined radio should be displayed on the PC so that a user can change parameters and make required adjustments for the project. To increase the efficiency of the system it is required to have minimum packet loss. The system should also be user-friendly and reliable as such factors increase usability and will require less technical expertise to operate.

3.2.1 DESIGN RESTRAINTS

The biggest challenge for this system is that at the time it was built it could not run on many platforms such as MacBook with M1 chips. The new silicon MacBooks cannot run virtual machines such as VirtualBox and hence cannot accommodate the LimeSDR suite. However, with older MacBooks, one should be able to install and run LimeSDR Suite.
With Linux, it can only run-on Ubuntu versions 16.04.x and 18.04.x which will be discussed in detail in the environment setup section. Hence the LimeSDR suite cannot be implemented on the latest Ubuntu versions and cannot utilize the upgrades which come with them. With Windows it should work perfectly well on windows 7 to 10.
Other restraints will be network-related and packet loss due to the use of USB ports and wireless access points through the Wi-Fi router and the antennas on the LimeSDR.

3.3 SYSTEM DESIGN SPECIFICATIONS

The design specification for the SDR is explained in detail in this section. These specifications will co-relate with the user specifications.

Fig 3.1 illustrates a low-level design of the system to be implemented. Radio signals are received by the RF antenna and are converted to an intermediate frequency by the RF front end. The analogue signal is then converted to a digital form by the analogue to digital converter (ADC). The ADC/DAC will be located in the LMS7002M Transceiver. The advantage of using such a transceiver is that it is highly programmable. Inside the LMS7002M are Low Noise Amplifiers (LNA), transmitter power amplifier drivers (TXPAD), RX/TX mixers, ADC/DAC converters, RX/TX filters, synthesizers, RX gain control, and TX power control. Traditionally all these components would be single separate hardware but in the case of SDRs, they are all combined into one small chip. The LMS7002M achieves multiple input and multiple output (MIMO) by equipping itself with two transmit and two receive chains. This chip has two phases of locked loops (PLL) which are shared between the transmitters and receivers.

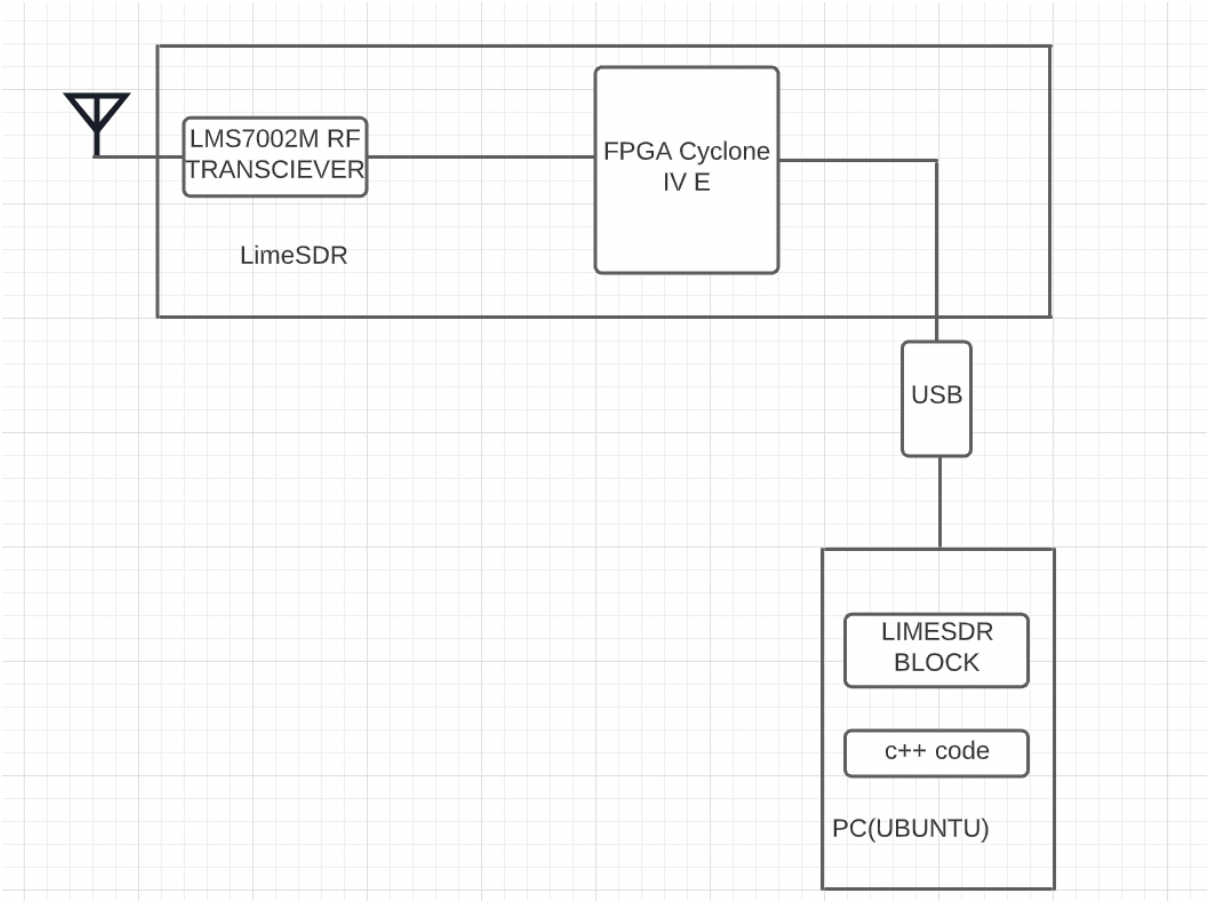


Fig 3. 1:*System Architecture*

The core of the LimeSDR is the Altera Cyclone IV FPGA which is responsible for transferring the digitized data between the USB 3.0 and the PC. This PC will be running Ubuntu version 18 and will have GNU and the lime suite radio installed. The LimeSDR

Block accepts the baseband signals from the LimeSDR for signal processing using the OFDM model and a comparison of PAPR will be carried out for both BPSK and QPSK.

3.4 HARDWARE DESIGN

No new hardware design will be implemented in this project but it is important to note down the hardware specifications. Hardware included in this project includes a LimeSDR, Wi-Fi Router, and PC.

3.4.1 LIMESDR SPECIFICATIONS

The LimeSDR will be based on an LMS7002M chip and has the following specifications:

- The RF Transceiver will be made of Lime Microsystems LMS7002M MIMO FPRF
- As mentioned in section 3.3, the FPGA will be based on the Altera Cyclone IV EP4CE40F23
- The device has an SDRAM of 256 MB
- The USB port is of version 3
- Has A Rakon RPT7050A Oscillator operating @30.72MHz.
- Operates at a continuous frequency range from 100 kHz to 3.8 GHz
- The LimeSDR will have a Bandwidth of 61.44 MHz
- will have 10 U.FL connectors (6 RX, 4 TX) for the RF connection
- It will have a power output of up to 10 dBm
- 2×2 MIMO architecture used for multiplexing
- The LimeSDR can be powered through the USB 3.0 but has an option for an external power supply
- The LimeSDR is also equipped with programmable LEDs

Other specifications of the LimeSDR were discussed in detail in section 2.2.

3.4.2 PC SPECIFICATIONS

As noted earlier the GnuRadio will run on a PC which should have the following specifications:

- Ubuntu 18.04.6 Operating System
- Has a processor of Intel® Core™ i7-8550U @1.80GHz - 1.99 GHz
- Installed RAM of 8.00 GB
- Total Disk space of 500GB

3.4.3 PROJECT SETUP

The project setup is quite simple as LimeSDR housed all its components inside a casing. The LimeSDR is connected to the PC via USB 3.0 as shown in fig 3.2. The USB 3.0 is responsible for the communication between the GnuRadio and the PC. An advantage of this over using a cable to connect the two devices is that cables are affected by attenuation hence by substituting them with a direct connection the shortfall is eliminated.

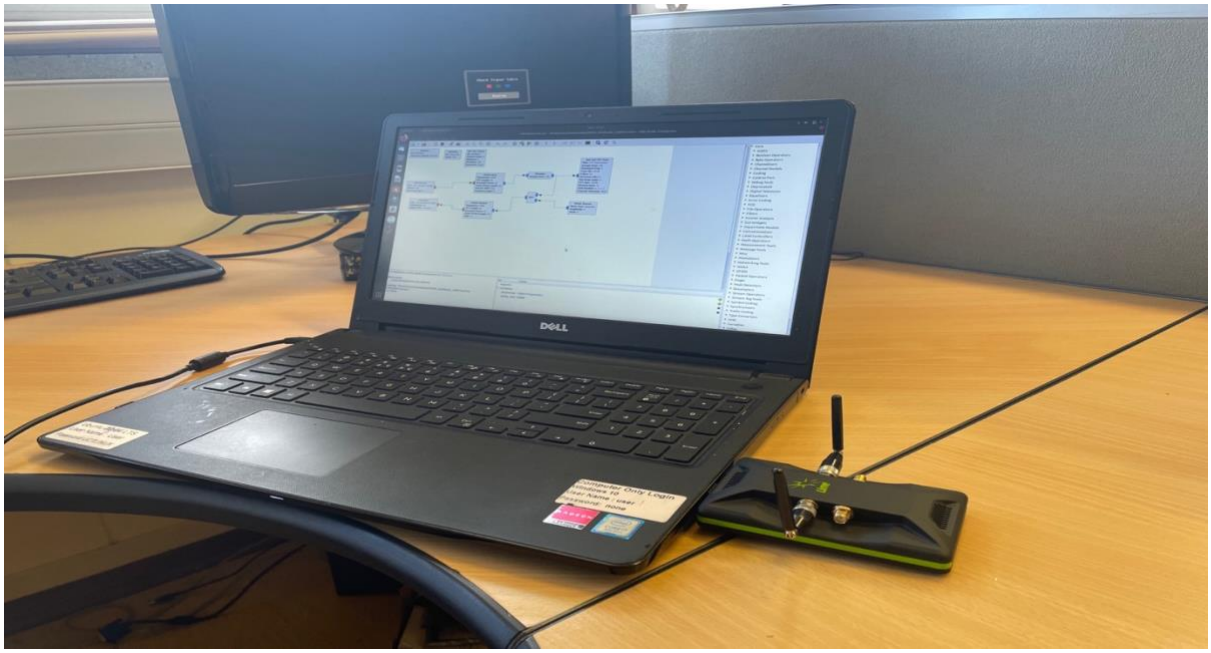


Fig 3. 2:PC connected to a LimeSDR with GnuRadio running.

CHAPTER 4: DESIGN

This chapter looks at system design used in all experiments done in this project. Before delving into the design, the Linux environment need to be setup to be conducive for the software used in this project. Therefore, the first section of the chapter will explain steps taken in setting up the environment then followed by a comprehensive look at the software design process.

4.1 ENVIRONMENT SETUP

It is crucial to understand the environment setup for this project to be successful. The PC specifications were already given in section 3.4.2. In this project, it is important to note that only Ubuntu 16.04.x and 18.04.x versions work, and I chose 18.04.6. It is also important to note that running old lime suite versions is not advised because no support is given for those hence I went with the latest one.

4.1.1 LIME SUITE

There are many ways to install the lime Suite on Ubuntu which are installing via Personal Package Archive (PPA) and building from the source. There are many steps involved when building from the source hence I chose to go with installing via PPA. The following steps are taken:

In the Ubuntu terminal section, the following commands are added:

```
sudo add-apt-repository -y ppa:myriadrf/drivers
sudo apt update
```

This command adds the PPA and then refreshes the available packages. The following command is also added to the terminal again to install the lime suite and all its dependencies

```
sudo apt install limesuite liblimesuite-dev limesuite-udev limesuite-images soapysdr soapysdr-module-lms7
```

4.1.2 GNURADIO

The GnuRadio is an important SDR platform for this project. The OFDM system is going to be designed on this platform. The following steps were taken to install it on Ubuntu 18.04.6: The first thing to do is to add the GnuRadio/GnuRadio-releases ppa:

```
$ sudo add-apt-repository ppa:gnuradio/gnuradio-releases
```

One will then need to access the following releases 3.9, 3.8, and 3.7. To do so the following

command is repeated for each release only modifying the release number to the specific one needed.

```
$ sudo add-apt-repository ppa:gnuradio/gnuradio-releases-3.9
```

All apt sources now need to be updated and then GnuRadio is installed.

```
$ sudo apt-get update
```

```
$ sudo apt install gnuradio
```

Then GnuRadio is installed on the Ubuntu PC.

4.1.3 GR- LIMESDR PLUGIN FOR GNURADIO

The gr- LimeSDR block is an important aspect of this project as it is the linking block between the LimeSDR and the GnuRadio. The prerequisites must be installed first which are the lime suite and the GnuRadio. There are also two ways in which the plugin can be installed which are via PPA and building from the source code. Again for this project, installing via PPA was chosen as it's the easiest route. The following commands are needed to download and install the plugin:

```
sudo add-apt-repository ppa:myriadrf/gnuradio
sudo apt-get update
sudo apt-get install gr-limesdr
```

After all these steps have been carried out, the environment is ready to run the experiment.

4.2 SOFTWARE DEVELOPMENT

As mentioned earlier, GnuRadio was used for developing the OFDM system. In this section, we look at the design and implementation of an OFDM system on GnuRadio. OFDM modules will be mainly used to construct our system and a LimeSDR block will also be added as it will communicate with the LimeSDR hardware. This project's main aim is to implement a software-defined radio on an OFDM system and do a comparison between QPSK and BPSK to see which performs better and measure the effectiveness of LimeSDR as they are new in the market dominated by USRPs.

4.2.1 OFDM MODULES IN GNURADIO

This section discusses the various OFDM modules used for this project in detail. This is then followed by a transceiver design that will show all blocks combined and how they interact with each other.

GNU offers a library with an OFDM example. This library has a directory of *benchmark_tx.py, benchmark_rx.py, and benchmark_add_channel.py* files [32]. This example is implemented in many OFDM solutions and this use case, it was also implemented. The transmitter and receiver will completely be designed in software on GnuRadio. The transmission channel is provided by the python script, *benchmark_add_channel.py*[32].

The python source code for *benchmark_tx.py* and *benchmark_rx.py* is hierarchical, which means that one part of the code can have many relationships with other parts of the python code but only communicates to the one below it. So, the top block communicates with the block below it. Therefore, the top blocks set the most parameters needed by lower blocks such as packet size, FFT size, modulation techniques, and the number of cyclic prefixes in a single OFDM symbol. In this topmost level, the data to be transmitted and the size of its packets are the only needed parameters while the rest are sent down the hierarchy to lower abstraction levels where they will be implemented at a later stage. Fig 4.1 illustrates all the blocks and how they are interconnected.

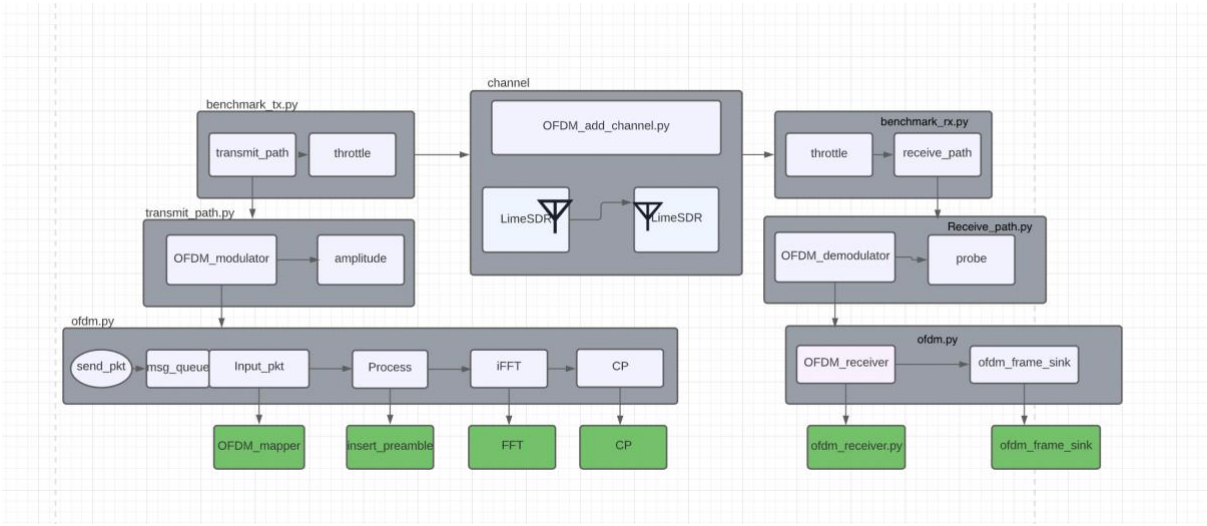


Fig 4. 1: Module Hierarchy in an OFDM system

From fig 4.1, the green boxes represent source code files and dark grey represents source code written in python. Light grey represents some of the building blocks provided by GnuRadio.

benchmark_tx.py and *benchmark_rx.py* represent the top-level blocks and have the task to start the transceiver. In these blocks, the transmit and receive paths are defined respectively. The frequency at which the LimeSDR will function is also set in these blocks while the throttle module is used to set transmission speed and sample rate. In the case, that one does

not have LimeSDR one can use the add channel to simulate a full OFDM communication system. As mentioned in early paragraphs, the OFDM system is hierarchical. The benchmark transmission python code is used to acquire information to be transmitted.
In this project, our transmission channel is via the electromagnetic spectrum, and signals are sent and received by LimeSDR antennas. The antennas are connected to the PC USB ports with one LimeSDR antenna set with the sole duty of transmitting while the other one receives as depicted in fig 3.2. Parameters such as SNR, frequency, and phase deviation are simulated in the add channel python source code[32].

The next set of blocks in the hierarchy consists of the transmit and receive paths python source code. The OFDM modulator is defined by the transmit_path.py and sets the parameters to message buffering rate and the other is padding for the LimeSDR. Padding simply multiplies data sent to the LimeSDR by 128 so that it is processed correctly. The receive_path.py is a python code that defines the OFDM demodulator and has an important parameter 'probe' which detects any ongoing transmission.
The last set of blocks is written in python and is constituted in the ofdm.py. GnuRadio instances are written in C++, hence in this level, OFDM modulation and demodulation are built based on the C++ files.

4.2.2 SOFTWARE DESIGN OF AN OFDM MODULATOR

Ofdmpy builds the OFDM modulator and demodulator. Virtual connections shown in fig 4.1 are used to interconnect blocks written in C++. The modulator is made in such a way that it has no ingress port but only one egress port. The *send_pkt* function is used to transmit the required information, which is the payload, to the *pkt_input* module. The payload data is transferred from the top of the hierarchy to the *send_pkt* which is then converted to packet data by the *make_pkt* found in the python module. A header and the Cyclic Redundancy Check (CRC) are added to payload data so that the authenticity of data can be verified at the receiver. The next stage is to allow this packet to be in a message queue which is made possible by the *insert_tail* function. There are two different outputs for the *pkt_input* which are the output for the actual data and the other output which has a vector of characters of the output OFDM symbol. The second output determines the start of a preamble so that the receiver can identify the start of each frame.

The ofdmpy tool has an important module, preamble, which adds a single preamble to the OFDM frame which is to be sent. Preamble constitutes known symbols randomly created and stored in the ofdmpy file. It has two ingress ports, the first one is linked to the first port of the *pkt_input* module and the second to the second characters output of the same module. Just like the *pkt_input* module, the preamble has two output ports, with the first one constituting FFT size, preamble symbols, and payload symbols and the second one is a character stream. The output of the preamble is then computed with the IFFT. If the parameters are edited, this module can also carry out FFT. The module is located in the C++ file.
The next module adds the cyclic prefix which protects the OFDM signal from the multipath effects. The C++ module is located within the modulator. The modulator takes the OFDM symbol from the input port and matches the last symbols to the start of the OFDM signal. The size of the output OFDM symbol, therefore, becomes the summation of the length of the input OFDM symbol and that of the cyclic prefix.
At long last, the OFDM signal is amplified using the GNU block gr multiply. The block is defined within the python file gnu core. Once amplified, the OFDM signal can now be transmitted over the air through the LimeSDR antennas.

4.2.3 SOFTWARE DESIGN OF AN OFDM DEMODULATOR

The OFDM demodulator is built the same way as the modulator but with additional blocks. It constitutes C++ blocks and python blocks. The demodulator in the python file has one input port and one output port. In this use case scenario, the input port is connected to the channel output of the modulator and its output is the demodulated signal. The demodulator uses queues to demodulate signals, when a new packet enters the queue, it waits for its turn. A function is then used to check the authenticity of the packet using the CRC method. OFDM receiver and OFDM frame sink are the two essential blocks defined in the demodulator. The OFDM receiver performs on the CRC signal and makes sure that the signal is synchronized and equalized. It is important to remap the symbols into bits and this is done by the second block, the OFDM frame sink. This is important as it also authenticates synchronized frames. Modules such as LPF, time synchronization, frequency synchronization, FFT, and frame acquisition are written in python and are included in the OFDM receiver module. Fig 4.2 illustrates the block diagram of the OFDM receiver module.

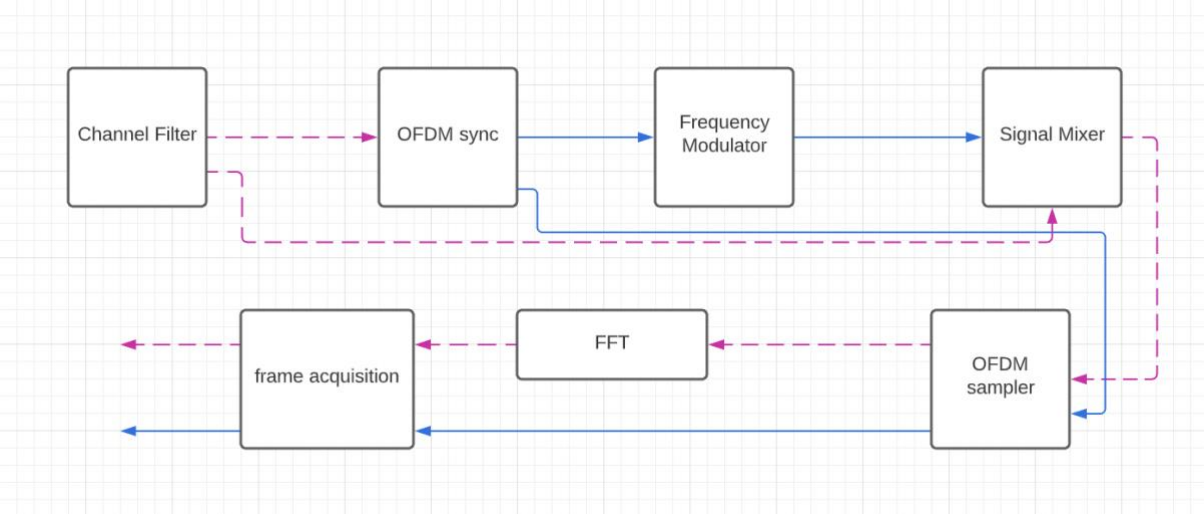


Fig 4. 2:*Block Diagram of OFDM receiver module*

As seen from the diagram, there are two possible paths followed by the output of the top block. The first path represented by the dashed pink arrow is the OFDM signal while the second path, represented by the blue solid arrow is any other signal like sync and frequency offset signals. The channel filter simply filters the wanted OFDM signal from the total signal received using Fourier transforms. For this project, the following parameters were chosen, FFT size is 64 and the number of subcarriers to be 40. These parameters are ideal for this project because it may be desirable to use fewer subcarriers to reduce the impact of interference from adjacent channels or other sources. By using only 40 subcarriers in an FFT 64 system, I achieved better performance in noisy or congested environments since I was using an unregulated frequency band. The practical application of subcarriers is to carry the actual data. An OFDM sync block is then used to locate the correct frequency offset so that it can start to receive frames. Once the Frequency offset is located it is transferred to the modulator where it is used to generate a proportional signal when compared to the frequency error of the OFDM sync block. The proportional signal is then mixed with the original signal from the channel filter which is then fed to the OFDM sampler. Also, in the OFDM sampler is a synchronized signal which is then transferred to the FFT which passes the received signal to the frequency domain. The subcarrier at the output contains the original information or data transmitted. In this project a comparison between BPSK and QPSK must be carried out, hence under the same conditions, our subcarriers are modulated, first using BPSK and then QPSK. The frame acquisition block essentially looks for the start of each frame and equates it to each subcarrier.

The demodulator has a frame sink with the sole purpose of defining the state of the machine. There are three states defined by the module:

- State 1 is simply used to search for a flag and is used to indicate the start of each frame. It is known as “sync search”
- When the flag is found it goes to state 2, known as “have sync”, which is responsible for finding the header and authenticating it.
- The last state is known as “have the header” which demaps the header.

CHAPTER 5: IMPLEMENTATION

In this chapter, the implementation steps involved in coming up with a full functional OFDM system for wireless communications are discussed. These steps will ensure that the initial objectives of the project are met. Features of OFDM system discussed in earlier chapters will be implemented in this chapter.

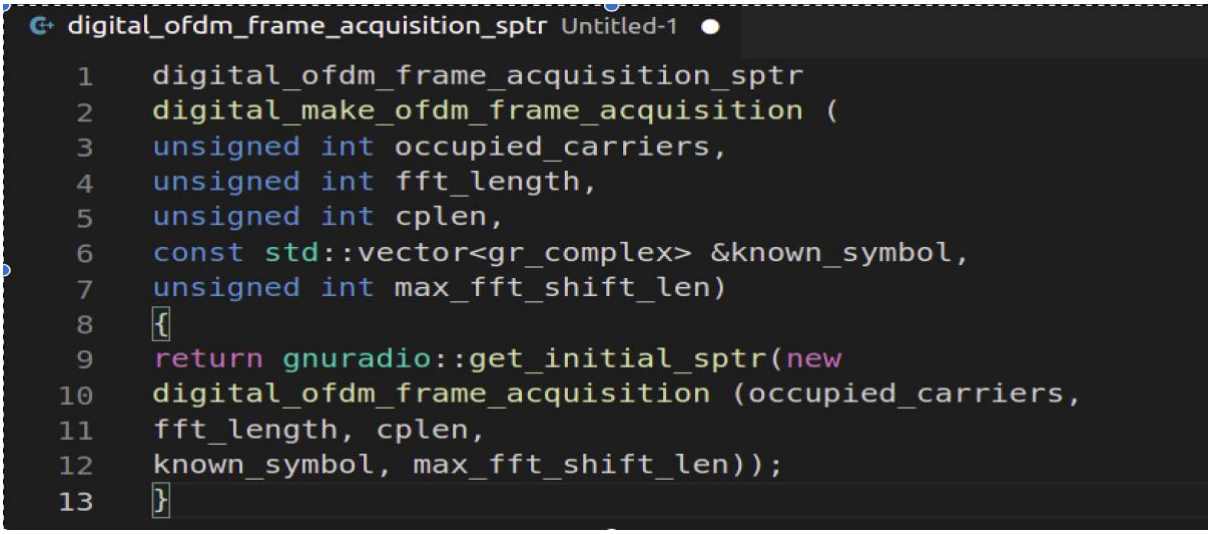
5.1 IMPLEMENTATION OF THE OFDM SYSTEM ON THE GNURADIO

In this section, we show how the various OFDM blocks were modified to suit this project. Within the GnuRadio module are subdirectories such as CMake, docs, and lib to mention a few. Each of these directories has a makefile.am file which has auto tools tasked to build, compile, and link C++ code to Python code automatically. Certain files are needed in these subdirectories for GnuRadio to run which are changelog, and *config.guess,* bootstrap, *makefile.common*, *makefile.swig*, authors news, *readme.hacking*, readme, and *version.sh*. Standalone executables and GCR applications are contained in the apps directory and instruction files are in the CMake folder and are used to locate libraries. Documentation is one of the most important features of GnuRadio and the set of instructions used to extract Python and C++ files is in the docs folder. In addition to this, any additional documentation can be manually added to the docs folder. The .cc and .h source files are in the lib subdirectory and essentially are the already developed blocks. Lib is used to store any source code written in another programming language other than python. For C++ code to communicate with python code there is a need for an interface which is accomplished by the SWIG tool which is contained in the swig folder. Python folder will store all python source code and related .py files.

5.1.1 GNURADIO BLOCK STRUCTURES

Block coding structures are an important aspect of GnuRadios. To be able to modify GnuRadio modules one must understand block coding structures. For this project, we modify the OFDM acquisition block which is in the *gr-digital* module. The OFDM acquisition block constitutes of three coding files:

- **Header File:** Variables that are either public or private are declared in this file. The prefix d_ is used to declare private variables. Set and get functions are used to change or use these variables outside the block.
- **Source File:** It is used to implement the actual code for the acquisition class. The source file defines the make function of the public class as shown by the code below. It is this code that can be modified to suit the project one is running. As stated earlier the acquisition block is inherited from the gr block module which is a highly flexible block. Its features allow the block to take in data at varying rates with different input streams of data. The output generated will be at the same rate as the input. The source file has a block as well that does the actual signal processing using the function *general_work()*.
- **Swig Interface:** Compiles the C++ code onto the python code. Headers can be included in the main swig file.



Once the code is modified to suit one's project, the block needs to be rebuilt and reinstalled and this can easily be done using the CMake auto tool.

5.1.2 CHANNEL ESTIMATION

Before OFDM symbols can be demapped, it is important to estimate the channel via which the signal will propagate. Channel estimation will remove unwanted distortion which is a result of fading.

The gr digital module implements channel estimation, together with the *digital_ofdm_frame_acquisition.cc* file. Fig 5.1 shows the code used to implement channel estimation.

```
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];
}
```

Fig 5. 1: Channel Estimation

This code includes the course carrier frequency because it is included in the file hence it is important in the calculations. The coefficients *d_hestimate* are used to save the channel impulse responses into the file. An additional block is needed to provide for the channel coefficient output which is not catered for in the OFDM acquisition block. Therefore, the original block must be modified as shown in fig 5.2.

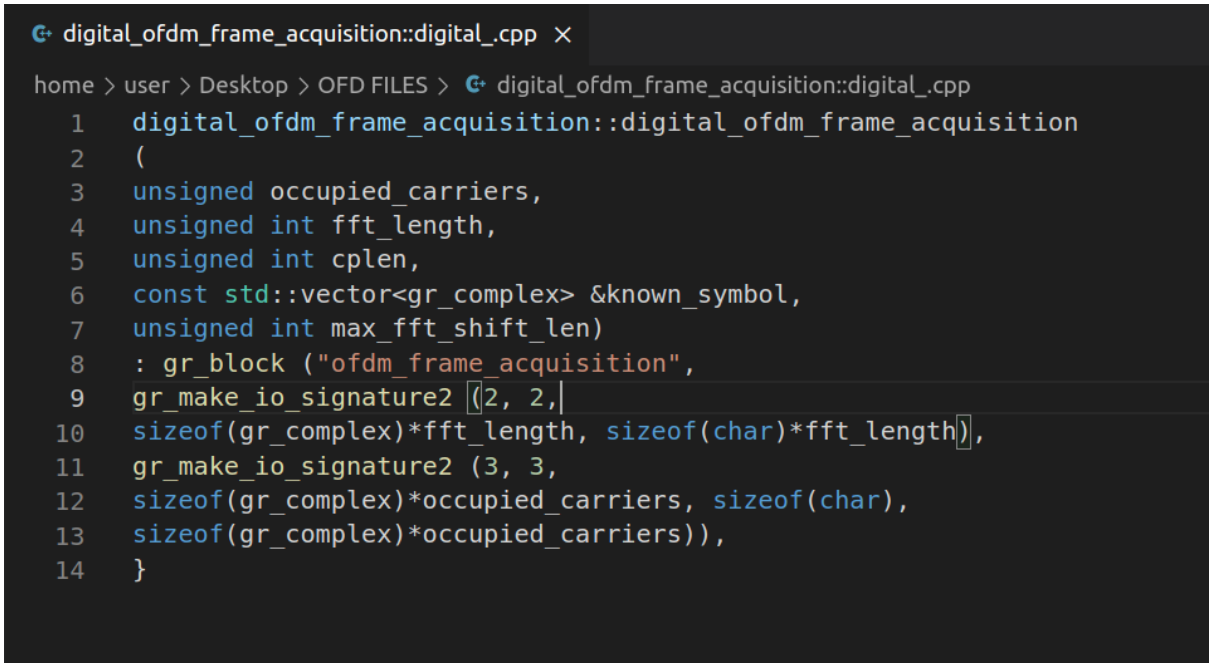


Fig 5. 2: Modification of Acquisition Block

In this File, there is also a function called *general_work()* which carries a new pointer that needs to be declared and its coefficient output needs to be stated as well as shown in fig 5.3.

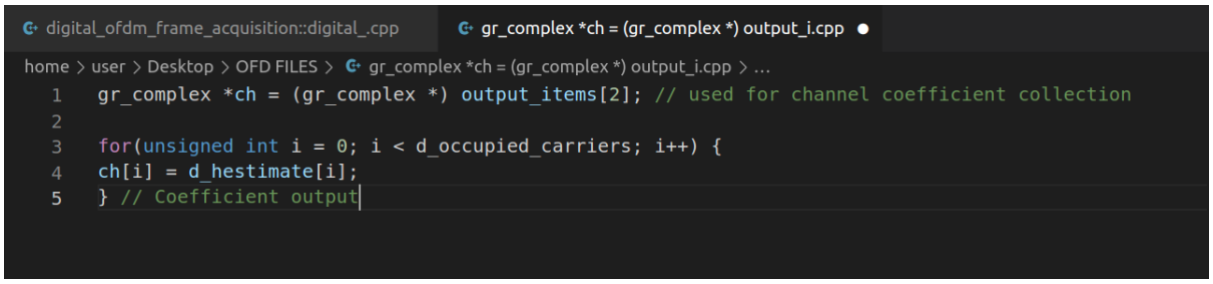


Fig 5. 3:Pointer Declaration

5.2 FINAL SYSTEM DESIGN

After all the steps discussed in this chapter, a final system can be put together that can carry out a comprehensive comparison between BPSK and QPSK in OFDM systems.

5.2.1 OFDM TRANSMITTER

Using the steps discussed in this chapter a transmitter was designed as shown in Fig 3.8. The transmission parameters are defined as variables and the source of information is a text file indicated in the block *File Source.* Stream to tagged stream is then used to tag data from the File Source this ensures that the data can be passed to lower blocks in our hierarchy. The information is then appended using CRC codes in the stream CRC32 block for error

checking. The information is then synchronized and can be either mapped to BPSK or QPSK symbols, depending on the modulation technique being tested. The payload bits are then mapped to either BPSK or QPSK. Preambles and pilots are then placed in the subcarrier positions of the IFFT input vector by the OFDM carrier allocator. Each OFDM symbol must be appended with a cyclic prefix (CP) in the time domain. Once this is done, The LimeSuite sink is used to transmit the signal. Important parameters in the LimeSuite need to be configured such as the serial of the LimeSDR which is 1D7514DB809138, a bandwidth of 1 MHz, the carrier frequency of 2GHz, and a sampling frequency of 1 MHz

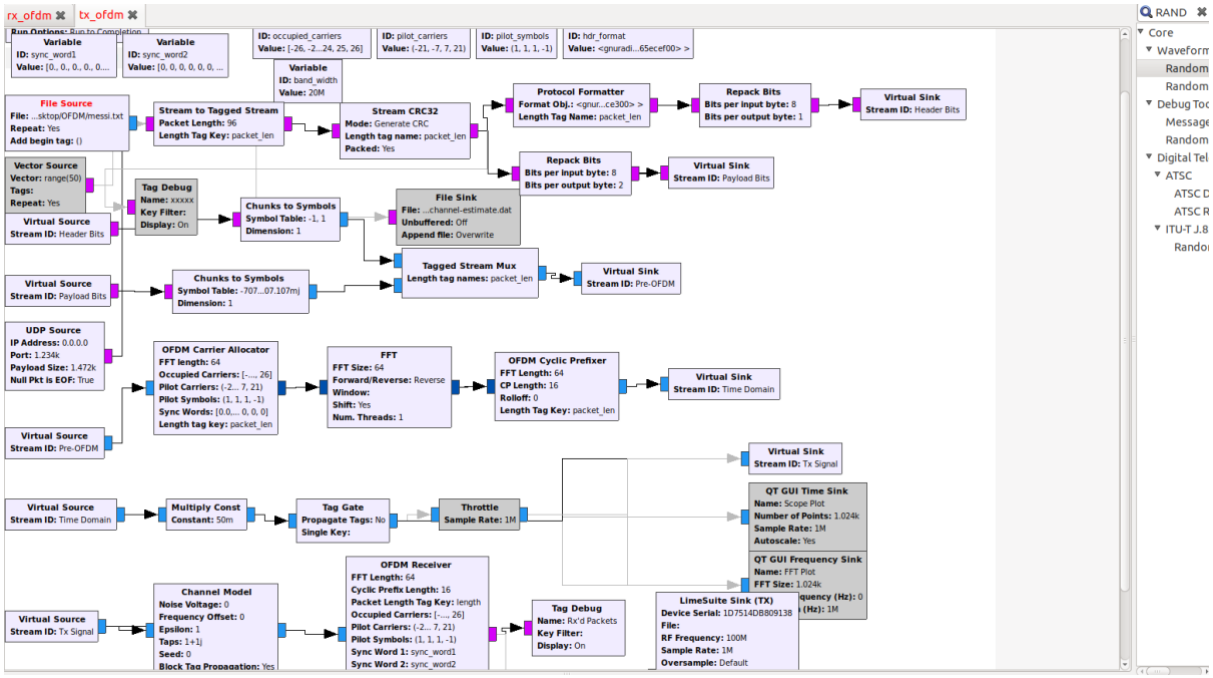


Fig 5. 4: OFDM Transmitter

5.2.2 OFDM RECEIVER

The steps discussed earlier in this chapter are used to design the receiver. The receiver undoes what has been done to the receiver. At the Receiver, the source is the LimeSuite and has the same parameters set as at the transmitter, but the serial of the Lime SDR changes as we will be using a different radio. In this case, the serial becomes 907060244381A. A synchronization algorithm is then implemented by the Schmidl and Cox OFDM block is then used to detect the start of OFDM symbols. After the start of the OFDM symbols has been detected, the Header/Payload Demux block extracts the payload and header streams. During transmission, channel distortion occurred hence there is a need to perform a channel

estimation to compensate for it. The output, which is the original sent information, is then saved in the destination folder. Fig 5.5 illustrates the design of the OFDM receiver.

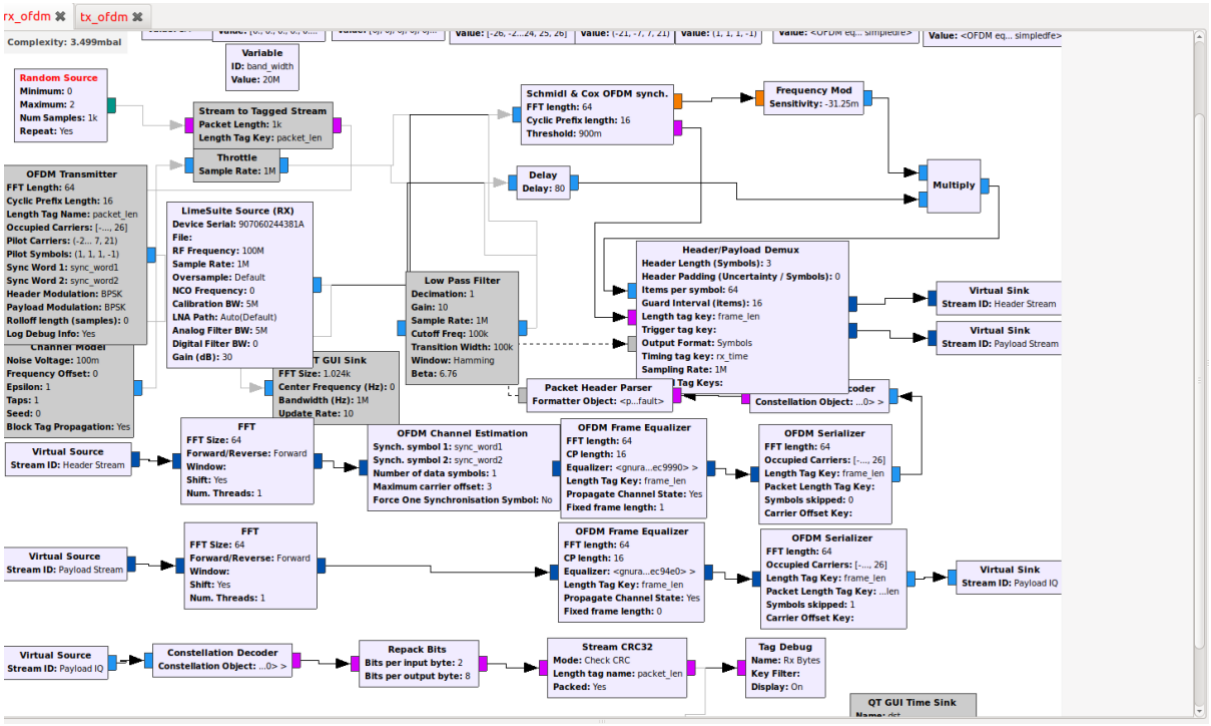


Fig 5. 5: OFDM Receiver

CHAPTER 6: RESULTS AND ANALYSIS

This chapter looks at the results of the experiments carried out in Chapter 3 and an evaluation of the techniques used during the project. An evaluation is necessary as it justifies the methodology and provides a proof of concept. In addition, results are necessary so that a comparison of BPSK and QPSK can be carried out in terms of PAPR.

6.1 METHODOLOGY EVALUATION

It is important to look at the methods used to come up with the system and prove that they work. Some of the things to test are to show that the LimeSDR works. Its ability to transmit and receive before we connect it to the GnuRadio.

6.1.1 LimeSDR EVALUATION

The first step of this evaluation is to look at the LimeSDR. To test the LimeSDR, a self-test file was uploaded on the LimeSDR GUI. This file contains several parameters that would be tedious to manually enter on the GUI, with the frequency set at 2100MHz being one of the important ones to note and all settings used in the test file have been uploaded to my GitHub, which is provided in the appendix. In this test, a waveform is sent from the host PC to the LimeSDR and OFDM signal is generated by the FPGA. The LimeSDR GUI has an RF switch that interlinks the TX and RX. The signal received is then displayed in the FFT viewer of the LimeSDR. Fig 6.1 shows the results of the transmitted signal.

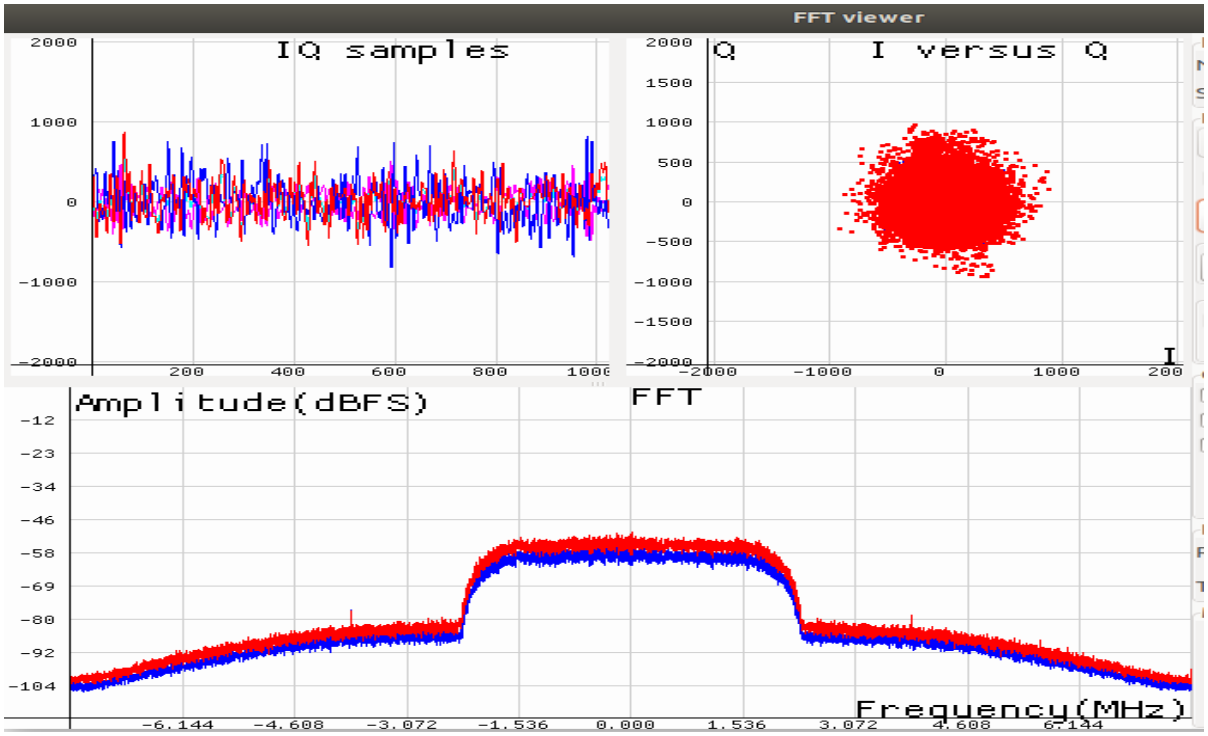


Fig 6. 1: Transmitted Signal

From the above diagram, we can see that the LimeSDR works perfectly fine for both channels A and B in terms of transmission. The next thing to test was receiving which the FFT plot for the received signal is shown in fig 6.2. As can be seen in the fig 6.2 the LimeSDR perfectly receives the signal in the 800MHz band.

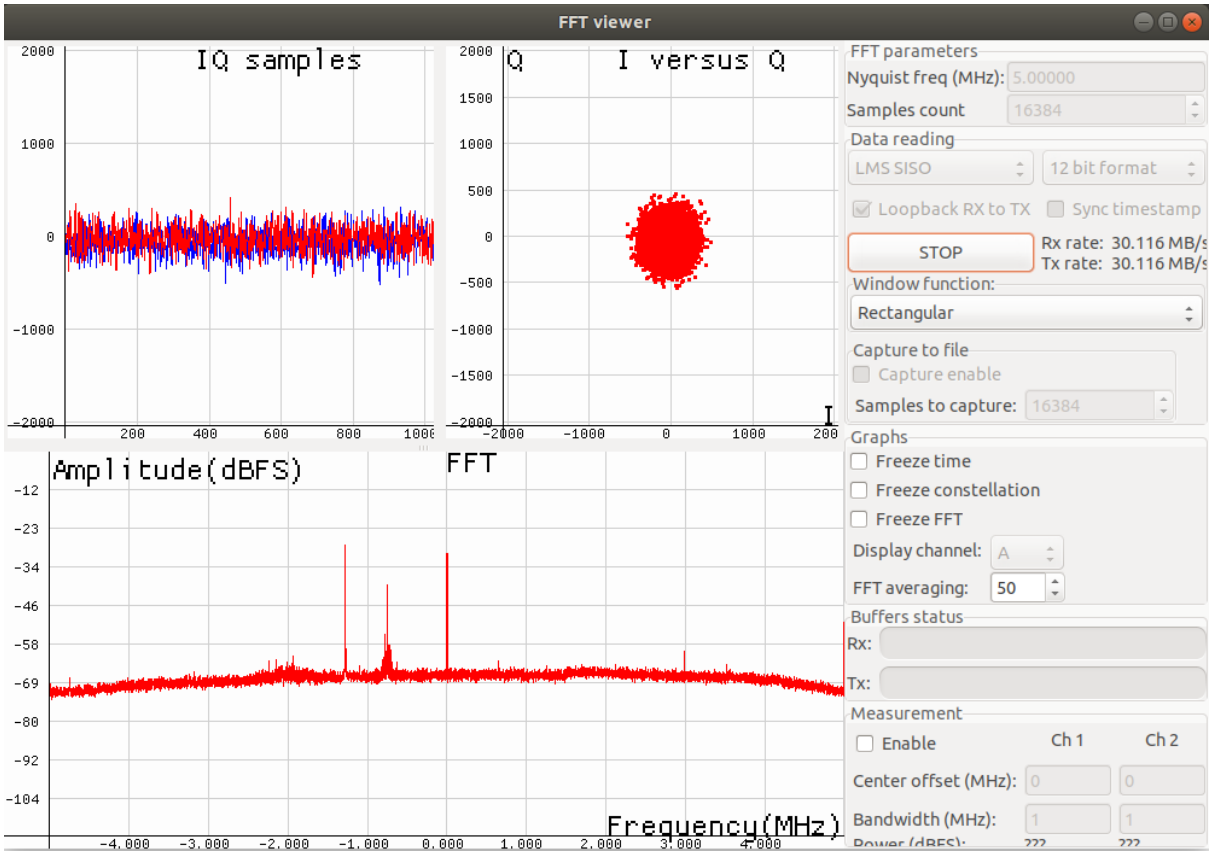


Fig 6. 2: LimeSDR Signal Reception

6.1.2 CONNECTIVITY BETWEEN LIMESDR AND GNURADIO

The next step was to test the connectivity between LimeSDR with GnuRadio. It is important for this project that the LimeSDR can send data frames to the GnuRadio. It is a test to see if the added LimeSDR blocks on GnuRadio work. If they do work signal should be plotted from the GUI Time and frequency sink. Fig 6.3 shows the LimeSDR source block connected to the GnuRadio so that its functionality can get tested. The output of the LimeSDR will be complex with a payload size of 2000kbits. The output of the GNU source block is connected to the input of the Complex to Float block which separates the complex output into real and imaginary parts. Also, the output of the LimeSDR is connected to the QT GUI Time Sink so that it can display the complex output. The real output of the complex to float block is then sent to the QT GUI Time Sink and QT GUI Frequency sink. It is important to set the input of the QT GUI blocks to type: *float* or else you will get an error as its default type is *complex.*

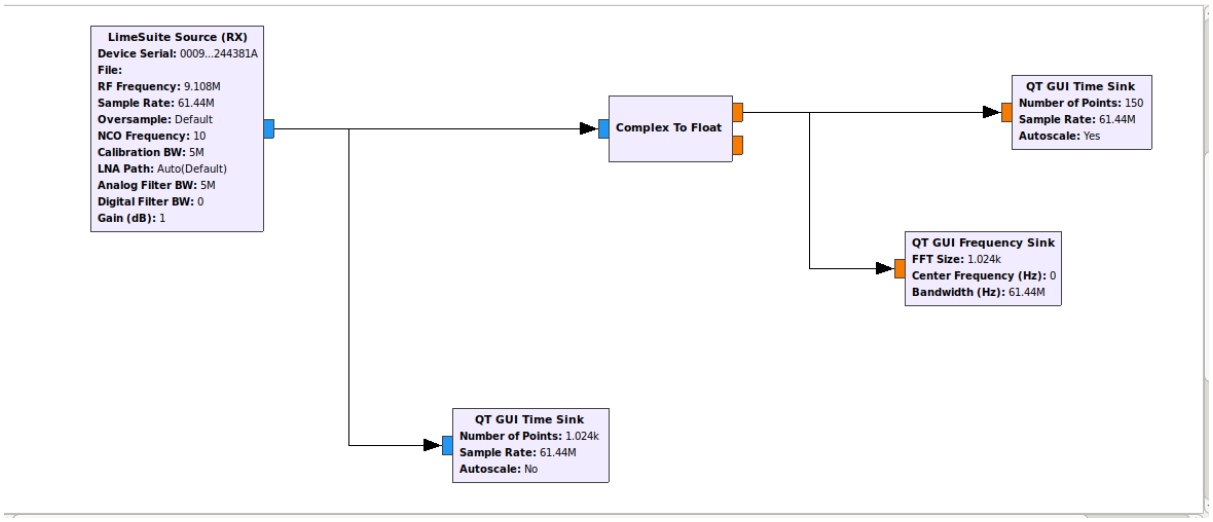


Fig 6. 3: LimeSDR Block diagram test with GnuRadio

Fig 6.4 shows the results of the above-mentioned experiment. The top chart shows the signal present in GnuRadio confirming that the LimeSDR block is compatible with GnuRadio. The top chart shows the complex output from the LimeSDR Source block where red represents the imaginary part and blue the real part. The middle chart shows the real part output of the Complex to Float block while the last chart shows the gain of the real output signal. Therefore the LimeSDR satisfies the needs of this project as it is compatible with GnuRadio.

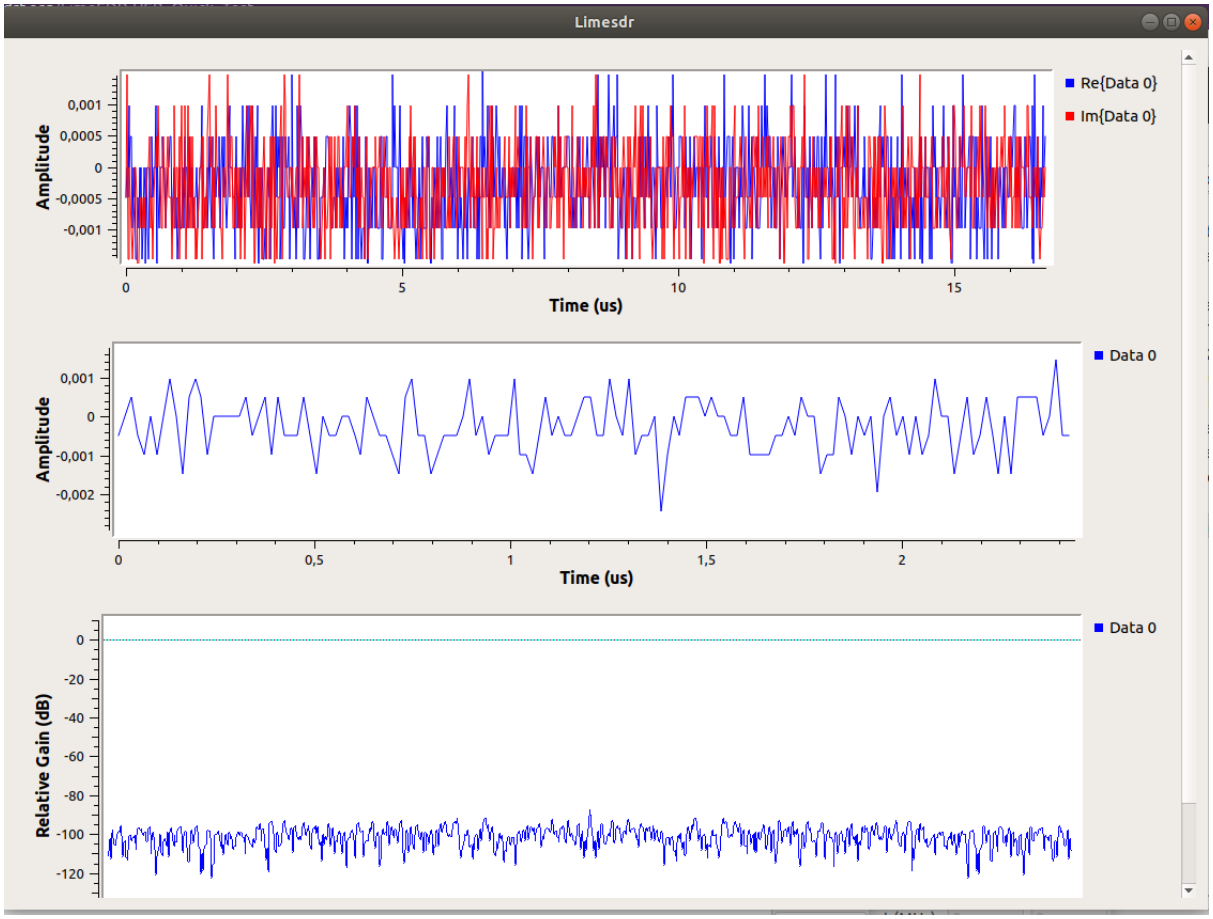


Fig 6. 4: QT GUI Output charts

6.2 BPSK MODULATION RESULTS

As mentioned in the earlier chapters the main research point for this thesis is to perform a comparative analysis between BPSK and QPSK in OFDM systems. When the signal was modulated in the first use case scenario BPSK was implemented. Fig 6.5 shows the amplitude of the received signal transmitted using BPSK.

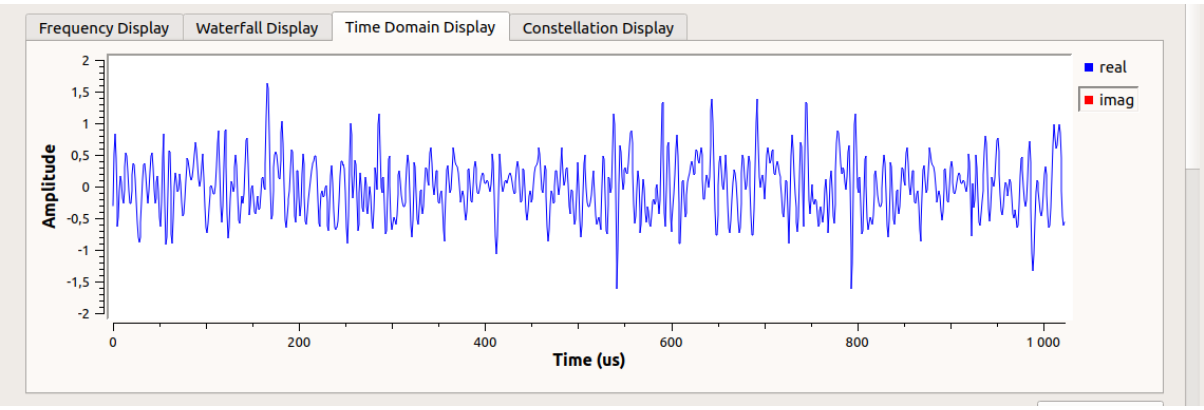


Fig 6. 5: BPSK Modulated signal

From the figure, we can note that there are multiple peaks in the signal ranging from 1.5 to -1.5. It can be fairly noted that the signal has high PAPR as such it would be expensive to transmit such a signal. From the diagram, we can note 7 peak formations.

6.3 QPSK MODULATION

With QPSK modulation amplitude peaks also range from 1.5 to -1.5 but they do not recur as much as in BPSK. There is only one peak formation. Fig 6.6 illustrates this.

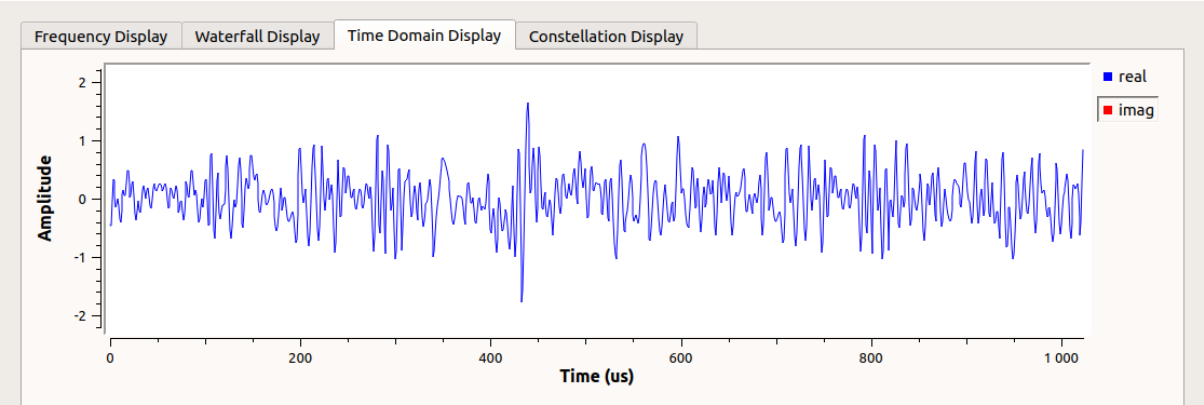


Fig 6. 6: BPSK Modulated signal

6.4 COMPARISON

From figs 6.5 and 6.6 we can note that BPSK has far more peak amplitudes reaching 1.5. compared to those of QPSK. QPSK has got only one signal which has an amplitude of 1.5 compared to the seven that of BPSK. For both BPSK and QPSK it can be noted that the average amplitude resonates between -0.5 and 0.5. Implementing an OFDM system based on BPSK would be expensive. OFDM's nature alone causes it to have high peaks. These high peaks are undesirable for amplification as they strain ADC/DAC circuitry. The main concern with BPSK modulated signal is that when it is transmitted through a High Power Amplifier (HPA), the high peaks noted in fig 6.5 would cause problems. Since a BPSK modulated signal has higher PAPR, it would require a linear component to transmit whereas HPA is non-linear. If the HPA are modified to become linear they will be now less efficient and expensive to construct. With these results, it would be logical to choose QPSK over BPSK as it has shown that it has fewer peak fluctuations hence a lower PAPR.

CHAPTER 7: CONCLUSION AND RECOMMENDATIONS

This chapter concludes this thesis based on the initial objectives presented in Chapter 1, the methodology employed, design, implementation and results and analysis. The conclusion will also review the initial objectives of the project and the actual results that were obtained in Chapter 6. In addition, recommendations for future work on SDRs and OFDM systems will be provided and how this project can be improved.

7.1 CONCLUSIONS

The following conclusions are formulated in response to the initial objectives that pursued for this project, as presented in Chapter 1, which were :

a). Development of the OFDM system:

- Use LimeSDR antennas to capture radio waves
- Develop a C++ code to interface LimeSDR with GnuRadio
- Design and implement a GnuRadio block for an OFDM system that will interface with LimeSDR
- Perform an efficiency test of the OFDM GnuRadio block in terms of connectivity with LimeSDR.

b). Evaluation of BPSK and QPSK:

- Implement BPSK and QPSK on the OFDM GnuRadio block for data coming from LimeSDR
- Capture and evaluate results for both BPSK and QPSK in terms of PAPR.
- Make a comparison of the results and give a recommendation.

The significance of these objectives is to use a low cost SDR (LimeSDR) for the transmission of an OFDM signal generated on GnuRadio. Chapter 2 gave an in depth discussion on the theoretical aspects of the thesis topic which help in the build-up of the methodology and the design of the OFDM system. After coming up with a viable methodology which employs RAD, I then designed and implemented the OFDM system on GnuRadio. After the system was completely implemented, several experiments were then carried out to determine LimeSDR compatibility with GnuRadio and then comparison of BPSK and QPSK to determine which modulation technique had lower PAPR.

Based on the experiments done and the results obtained conclusions can be drawn for the entire thesis. In this project we are implementing a fairly new SDR compared to the traditional USRP, KerberosSDR, Zeus ZS-1, Cyan and Myriad-RF to mention but just a few. LimeSDR are low cost, open source software defined radios implemented in this project to capture radio waves and transmit them via an OFDM system designed in a GnuRadio. From the results it can be noted that the antennas were able to capture radio waves based on Fig 6.1 and an integration of a LimeSDR block was possible in the GnuRadio. In one of the experiments, connectivity between LimeSDR and GnuRadio was tested and it can be noted that there was connectivity between the two hence LimeSDR can successfully be implemented in GnuRadio. Both BPSK and QPSK were implemented for separate experiments in the designed OFDM system and different results were also obtained for the two modulation techniques. For BPSK it was noted that there were too many peaks from the average peaks hence the signal would have a high PAPR whereas QPSK has fewer peaks from the average peak hence a lower PAPR. Therefor it can be concluded that it is best to implement QPSK in OFDM systems as it has lower PAPR hence the required HPA would be cheaper to construct.

7.2 RECOMMENDATIONS

Further work is needed for this project in order to improve it and get more accurate results. First recommendation is to implement OFDM system with various SDR hardware and compare results of PAPR with the results of LimeSDR. An example is to repeat this very same project but instead of implement a LimeSDR, one can use a RHINO board and with the results from a RHINO board comparison can be carried out with results from the LimeSDR. The same can be repeated using USRP, Myriad-RF and Cyan and all the results can be compared and from the results it can be determined which best SDR platform can be implemented for OFDM systems with lower PAPR.

Once the SDR platform is determined, and the best modulation technique on that platform be it BPSK or QPSK, I further recommend to look at other ways of reducing PAPR. There are several ways in which PAPR can be reduced and most of the methods are discussed in several journals. Some of the discussed methods in these journals include:

- Selective mapping
- Block coding technique
- Tone reservation

- Linear block code
- Partial transmit sequence
- Inter leaving technique
- Clipping and filtering
- Peak windowing.

All these methods can now be implemented with the SDR platform that produced the best results with QPSK as the modulation technique as it has lower PAPR. If the above recommendations are implemented then the best system can be designed which can ultimately lower the cost of hardware implementation in OFDM systems.

REFERENCES

[1] Z. Wang, E. Sun, and Y. Zhang, “An Overview of Peak-to-Average Power Ratio Reduction Techniques for OFDM Signals,” *International Journal of Mobile Network Communications & Telematics*, vol. 6, no. 3, Jun. 2016, doi: 10.5121/ijmnct.2016.6301.

[2] N. Sharma, “Peak-to-Average Power Ratio Reduction Techniques for OFDM Signals,” *Int J Comput Appl*, vol. 96, no. 22, Jun. 2014, doi: 10.5120/16929-7040.

[3] S. Rony, F. Mou, and M. Rahman, “Performance Analysis of OFDM Signal Using BPSK and QPSK Modulation Techniques,” *Performance Analysis of OFDM Signal Using BPSK and QPSK Modulation Techniques*, vol. 6, no. 1, pp. 108–117, 2017.

[4] M. Praveen, “Implementation of OFDM wireless communication model for achieving the improved BER using DWT-OFDM,” *International Journal of Engineering And Computer Science*, Jan. 2017, doi: 10.18535/ijecs/v6i1.17.

[5] S. K. P, S. M.G, and S. M, “Performance Analysis of Rayleigh Fading Channels in MIMO-OFDM Systems using BPSK and QPSK Modulation Schemes,” *The SIJ Transactions on Computer Networks & Communication Engineering*, vol. 04, no. 02, Apr. 2016, doi: 10.9756/SIJCNCE/V4I2/01010258.

[6] A. Abrol, “OFDM AS AN ACCESS TECHNIQUE FOR NEXT GENERATION NETWORK,” *International Journal of Advanced Technology in Engineering and Science*, vol. 6, no. 5, pp. 1–9, 2018.

[7] B. K. Mishra, S. K. Singh, and M. Sharma, “An OFDM based technique with QPSK and DQPSK modulation for next generation network,” *International Conference and Workshop on Emerging Trends in Technology 2011, ICWET 2011 - Conference Proceedings*, pp. 912–916, 2011, doi: 10.1145/1980022.1980217.

[8] A. Thakur and M. Rattan, “Implementation of convolution coded OFDM through different channel models on SDR platform,” *International Journal of Information Technology*, Dec. 2018, doi: 10.1007/s41870-018-0265-2.

[9] T. Tekin and B. Karakaya, “SDR implementation of OFDM transmitter and receiver in case of time and frequency offset,” in *2018 26th Signal Processing and Communications Applications Conference (SIU)*, May 2018. doi: 10.1109/SIU.2018.8404269.

[10] “What is PAPR (Peak to average power ratio), Why it matters to Power Amplifier ? - Techplayon.” https://www.techplayon.com/papr-peak-average-power-ratio-matters-power-amplifier/ (accessed Jul. 11, 2022).

[11] P. Beynon-Davies, C. Came, H. Mackay, and D. Tudhope, “Rapid application development (Rad): An empirical review,” *European Journal of Information Systems*, vol. 8, no. 3, pp. 211–232, 1999, doi: 10.1057/PALGRAVE.EJIS.3000325.

[12] E. Grayver, *Implementing Software Defined Radio*. Springer Science & Business Media, 2012.

[13] S. Hamill, “ What Is Software Defined Radio (SDR)? | Wireless Innovation Forum.” https://www.wirelessinnovation.org/what-is-sdr (accessed Apr. 30, 2021).

[14] J. D. Day and H. Zimmermann, “The OSI reference model,” *Proceedings of the IEEE*, vol. 71, no. 12, 1983, doi: 10.1109/PROC.1983.12775.

[15] J. Bard and V. J. Kovarik, *Software Defined Radio*. Chichester, UK: John Wiley & Sons, Ltd, 2007. doi: 10.1002/9780470865200.

[16] K. Sobaihi, A. Hammoudeh, and D. Scammell, “Automatic gain control on FPGA for Software-Defined Radios,” *Wireless Telecommunications Symposium*, 2012, doi: 10.1109/WTS.2012.6266106.

[17] P. Cruz, N. Carvalho, and K. Remley, “Designing and Testing Software-Defined Radios,” *IEEE Microw Mag*, vol. 11, no. 4, Jun. 2010, doi: 10.1109/MMM.2010.936493.
[18] S. Rommel *et al.*, “Real-time high-bandwidth mm-wave 5G NR signal transmission with analog radio-over-fiber fronthaul over multi-core fiber,” *EURASIP J Wirel Commun Netw*, vol. 2021, no. 1, Dec. 2021, doi: 10.1186/s13638-021-01914-6.
[19] D. Frizelle and F. Kearney, “Complex RF Mixers, Zero IF Architecture, and Advanced Algorithms: The Black Magic in Next-Generation SDR Transceivers,” *Microwave Journal*, Feb. 2017. https://www.microwavejournal.com/articles/29171-complex-rf-mixers-zero-if-architecture-and-advanced-algorithms-the-black-magic-in-next-generation-sdr-transceivers?page=1 (accessed May 13, 2021).
[20] R. Svitek and S. Raman, “DC offsets in direct-conversion receivers: characterization and implications,” *IEEE Microw Mag*, vol. 6, no. 3, Sep. 2005, doi: 10.1109/MMW.2005.1511916.
[21] J. Thabet, R. Barrak, A. Ghazel, and F. M. Ghannouchi, “Generalized Bandpass Sampling Algorithm for Multiband Wireless Receivers Suitable for SDR Applications,” *Circuits Syst Signal Process*, vol. 36, no. 3, Mar. 2017, doi: 10.1007/s00034-016-0341-4.
[22] A. Loke and F. Ali, “Direct conversion radio for digital mobile phones-design issues, status, and trends,” *IEEE Trans Microw Theory Tech*, vol. 50, no. 11, Nov. 2002, doi: 10.1109/TMTT.2002.804624.
[23] Jongsik Kim *et al.*, “A CMOS direct conversion transmitter with integrated in-band harmonic suppression for IEEE 802.22 cognitive radio applications,” in *2008 IEEE Custom Integrated Circuits Conference*, Sep. 2008. doi: 10.1109/CICC.2008.4672158.
[24] H. Haldren, “Studies in Software-Defined Radio System Implementation ,” Thesis, Liberty University, Virginia, 2014.
[25] D. N. Grujia, P. Jovanovic, and M. Savic, “Using software defined radio for RF measurements,” in *2017 Zooming Innovation in Consumer Electronics International Conference (ZINC)*, May 2017. doi: 10.1109/ZINC.2017.7968652.
[26] “LimeSDR-USB hardware description - Myriad-RF Wiki.” https://wiki.myriadrf.org/LimeSDR-USB_hardware_description (accessed Jul. 11, 2022).
[27] “LimeMicro:LMS7002M Datasheet - Myriad-RF Wiki.” https://wiki.myriadrf.org/LimeMicro:LMS7002M_Datasheet (accessed Jul. 11, 2022).
[28] V. Vahidi and E. Saberinia, “OFDM high speed train communication systems in 5G cellular networks,” in *2018 15th IEEE Annual Consumer Communications & Networking Conference (CCNC)*, Jan. 2018. doi: 10.1109/CCNC.2018.8319172.
[29] G. Ministeri and L. Vangelista, “Channel Impulse Response Estimation in IEEE 802.11p via Data Fusion and MMSE Estimator,” *International Journal of Vehicular Technology*, vol. 2015, Mar. 2015, doi: 10.1155/2015/670482.
[30] V. Pasi, P. Nigam, and V. Chaurasia, “Review on OFDM a Brief Survey,” *International Journal of Advanced Research in Computer Science*, vol. 3, no. 11, pp. 658–662, Sep. 2013.
[31] “https://wiki.gnuradio.org/index.php/Guided_Tutorial_Introduction.” https://wiki.gnuradio.org/index.php/Guided_Tutorial_Introduction (accessed Jun. 04, 2021).
[32] D. T. Nguyen, “Implementation of OFDM systems using GNU Radio and USRP,” Postgrad Thesis, University of Wollongong, 2013.

APPENDIX A

Link to GitHub for GnuRadio code and python code snippets:

https://github.com/Khuger01/Comparative-Analysis-of-OFDM-Signal-using-BPSK-and-QPSK-modulation-techniques