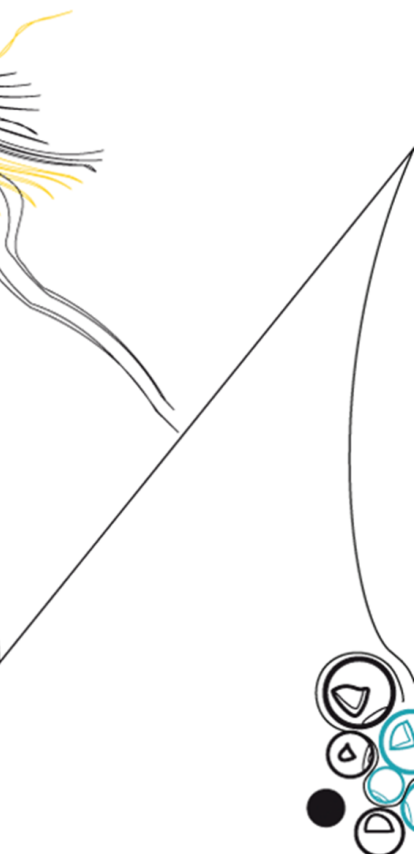




Computer Architecture for Embedded Systems Group
Faculty of Electrical Engineering,
Mathematics and Computer Science
University of Twente
P.O. Box 217
7500 AE Enschede
The Netherlands

ACKNOWLEDGEMENT

I would first like to express my special thanks to all of my supervisors, Dr. Andre Kokkeler, Dr. Arjan Meijerink, Siavash Safapourhajari and Zaher Mahfouz. Thanks for your sincere assistance and valuable guidance during my whole master project. Thanks for Siavash Safapourhajari. Your office was always open when I ran into trouble or had a problem in my project. Your careful review and modification on my thesis helps me a lot in improving my thesis. Thanks for Zaher Mahfouz who provides me support in experimental devices.

I would also like to thank my friends, Matheus Terrivel and Konstantino Fatseas. Thanks for your companionship and encouragement during my master project.

Finally, I must express my profound gratitude to my father Bin Wang and to my aunt and uncle Jian Zhang & Hao Lu for providing me with constant supports and continuous encouragements throughout my years of study in the Netherlands and through the process of my final project. This thesis would not have been possible without them. Thank you again.

Zhiyuan Wang

November 21, 2017

Enschede

The Netherlands

SUMMARY

The Internet of Things (IoT) extends the virtual cyber world into the real physical world by networking everyday smart physical objects, representing an upgraded version of Internet. The wireless sensor networks (WSN) are playing diverse sensing functions for IoT and feeding information from the physical world to the virtual cyber world. Nowadays, most of the WSNs are deployed to detect slow-changing physical quantities and data from every sensor node in these WSNs varies very slowly over a long time interval. Accordingly, a low data transmission rate is sufficient for the sensor nodes. Moreover, the low data rate makes it possible to apply very narrowband (VNB) radio communication with a bandwidth of a few kHz in wireless sensor nodes.

It is noteworthy that most of wireless sensor nodes are transmitting data in the unlicensed 2.4GHz frequency band where the dominant interference is characterized by relatively wide bandwidth. Therefore, if a very narrowband (VNB) radio is adopted in a wireless sensor node, only a small portion of co-channel wideband interference will overlap with the VNB signal transmitted by the wireless sensor node. Because only a small portion of the wideband interferer's power is superimposed onto the VNB signal, the VNB signal captured by a receiver has a relatively high signal-to-noise ratio (SNR). Therefore, it is possible to reduce the power of the transmitter or release the noise of the receiver.

Power consumption is a key factor that determines the lifetime of a sensor node because most of sensor nodes are powered by batteries. Once a battery is depleted, the lifetime of a sensor node will expire. The radio transceiver on a wireless sensor node consumes a lot of power when it is working. However, the VNB radio communication makes it possible for the radio transceiver to decrease its transmission power while guaranteeing the same bit error ratio (BER). Thus, the low power consumption is made possible by the VNB radio transceiver.

In this research project, the VNB radio transceiver is called "slow wireless radio". It aims at the wireless sensor nodes that transmit data at very-low instantaneous bit-rates (100~1000bps) in the 2.4GHz frequency band full of wideband interference while achieving low power consumption. The goal of this project is to build up a point-to-point VNB radio communication testbed based on software-defined radio (SDR). This testbed will serve as a design reference to investigate the feasibility of the VNB radio communication when it is used in wireless sensor nodes and facilitate the development of a slow-wireless sensor node prototype.

The VNB radio communication testbed is composed of a VNB transmitter and a VNB receiver where binary phase shift keying (BPSK) is used as modulation format and double differential encoding/decoding helps the testbed eliminate the effects of frequency and phase offsets. The bit transmission rate can be adjusted by repeater at the transmitter side while the accuracy of received bits is improved by DC-offset compensation and timing recovery at the receiver side.

The testbed can conduct the real-time measurements of signal-to-noise ratio (SNR) and bit error ratio (BER). The experimental results indicate that successful VNB wireless communication can be implemented on the testbed.

Contents

List of Acronyms	vi
1 INTRODUCTION	1
1.1 Research Background	1
1.2 Research Motivation	1
1.3 Works of Others	2
1.4 Slow Wireless Radio	3
1.5 Research Problem Description	4
1.5.1 Research Scope	4
1.5.2 Research Goals	5
1.6 Thesis Outline	5
2 EXPERIMENT PLATFORM	6
2.1 Software-defined Radio	6
2.2 SDR Development Kit	7
2.2.1 GNU Radio	7
2.2.2 Survey on other SDR development tools	8
2.2.3 Why is GNU radio chosen?	9
2.3 SDR Platform	10
2.3.1 What is SDR platform?	10
2.3.2 Survey on SDR platforms	11
2.3.3 Why is bladeRF chosen?	12
2.3.4 bladeRF Introduction	13
2.3.5 bladeRF Internal Structure	14
2.4 LMS6002D Transceiver	15
2.4.1 Introduction	15
2.4.2 Internal Structure	16
3 TRANSMITTER DESIGN	18
3.1 Transmitter Structure Overview	18
3.2 BPSK Modulator	19
3.3 Double Differential Encoder Design	20
3.4 Repeater	21
3.5 IQ Modulator	23
4 RECEIVER DESIGN	27
4.1 Receiver Structure Overview	27
4.2 IQ Demodulator	28
4.2.1 IQ Demodulation	28
4.2.2 IQ Imbalance	31
4.3 DC Offset Compensation	32
4.4 Double Differential Decoder Design	33

4.5	Matched Filter and Sampler Design	35
4.5.1	Repeater and Integrator	36
4.5.2	Matched Filter	37
4.5.3	Sampler	39
5	TESTBED DESIGN AND EXPERIMENT	41
5.1	Testbed Design	41
5.1.1	GNU Radio Data Types	42
5.1.2	Sample Rate vs. Symbol Rate	43
5.1.3	Transmitter Implementation	44
5.1.4	Receiver Implementation	45
5.1.5	Channel Model for Software Simulation	46
5.1.6	RF Interface for Hardware Experiment	47
5.2	SNR and BER	47
5.2.1	SNR and Eb/No	47
5.2.2	BER vs. Eb/No	48
5.2.3	SNR Estimator	48
5.2.4	BER Calculator	50
5.2.5	Cross Correlator	51
5.3	BER Performance	52
5.3.1	Software Simulation for BER versus Eb/No	53
5.3.2	Software Simulation for BER versus Phase Offset	54
5.3.3	Software Simulation for BER versus Frequency Offset	55
5.3.4	bladeRF Test	56
6	CONCLUSION	60
6.1	Conclusions	60
6.2	Recommendations	60
	Bibliography	62
	Appendices	
A	IQ MODULATION AND DEMODULATION	66
B	DOUBLE DIFFERENTIAL CODING	72

LIST OF ACRONYMS

AWGN	additive white gaussian noise
ADC	analog-to-digital converter
BER	bit error ratio
BPSK	binary phase shift keying
bps	bits per second
DC	direct current
DAC	digital-to-analog converter
DSP	digital signal processor
FPRF	field programmable radio frequency
FPGA	field programmable gate array
FIR	finite-impulse response
HDL	hardware description language
IF	intermediate frequency
IC	integrated circuit
ISM	Industrial, Scientific and Medical
IoT	Internet of Things
IADC	in-phase analog-to-digital converter
IDAC	in-phase digital-to-analog converter
IQ	in-phase and quadrature
LE	logic elements
LNA	low-noise amplifier
LO	local oscillator
LSB	least significant bit
LPF	low-pass filter
MAC	medium access control
M2M4	second and fourth order moment
NRZ	non-return-to-zero

PLL	phase-locked loop
PSK	phase shift keying
PA	power amplifier
QPSK	quadrature phase shift keying
QADC	quadrature analog-to-digital converter
QDAC	quadrature digital-to-analog converter
RF	radio frequency
RRC	root-raised cosine
RX	receive
SNR	signal-to-noise ratio
SDR	software-defined radio
SMA	subminiature version A
SoC	systems-on-chip
SPI	serial peripheral interface
SWIG	simplified wrapper and interface generator
TX	transmit
USRP	universal software radio peripheral
UWB	ultra-wideband
VNB	very narrowband
VGA	variable-gain amplifier
WSN	wireless sensor network
XOR	exclusive-OR

This chapter will start with a background introduction on the research. Then, it will state the motivation behind the research and reviews the works that have been done in this research field. Section 1.4 will propose the concept of slow wireless radio. The concrete research problems in this project will be formulated in section 1.5. This chapter will be concluded by the outline of the thesis.

1.1 Research Background

Wireless communication is defined as the transmission of information over a distance between two or more terminals without relying on any form of electrical conductors such as cables and wires. The wireless transfer of information is usually achieved by various types of electromagnetic waves (such as radio waves, microwaves and infrared light). Radio waves (with electromagnetic wave frequency between 3kHz to 300GHz) are the most common information carrier in most of wireless communication technologies. With radio waves, the wireless communication distance can span from a few meters to thousands of kilometers [1], [2]. In modern society, wireless communication technology is ubiquitous and applied in a wide range of portable electronic devices such as mobile phone, GPS receiver, laptop computer and so on.

The Internet of Things (IoT) refers to the inter-networking of a wide variety of smart physical objects. It represents an upgraded version of Internet which extends from virtual cyber world into real physical world by interconnecting everyday physical objects. With the help of IoT technology, the physical objects in the real world are no longer disconnected from the virtual world but can communicate with the virtual world on the Internet anytime and anywhere. Every smart object is equipped with microprocessor, sensor, actuator, communication and networking modules. It is able to collect, process and exchange data. The IoT makes ubiquitous computing possible and has great application potentials to revolutionize the future society. It is expected that more than 30 billion IoT-based smart objects will be connected to the Internet by 2020 [3], [4]. Because it is impractical to interconnect a tremendous number of smart objects via wires, the IoT is completely based on the wireless communication technology.

The wireless sensor network (WSN) refers to the network formed by a large number of spatially distributed wireless sensor nodes. Equipped with sensor, microprocessor, wireless transceiver module and power module, each node can detect, process and transfer various physical quantities such as temperature, humidity, pressure, etc. The WSN is one of the most pivotal constituent parts in the IoT, playing diverse sensing roles for IoT and feeding information from the physical world to the virtual cyber world [5]. Hence, WSN has become a hot research topic in academia and industry.

1.2 Research Motivation

Nowadays, most of WSNs are applied in monitoring environmental conditions such as temperature, humidity and pressure or structural conditions such as buildings and bridges. The data that is collected by each sensor node in a monitoring WSN typically varies very slowly over the time. The data packet that is transmitted by

each sensor node every time contains simply a very small amount of available information. That is to say, each sensor node only needs to transfer a small amount of data over a long time interval. Therefore, it is sufficient for wireless sensor nodes in monitoring applications to transfer data at very-low bit rate.

Meanwhile, the wireless transceiver modules which are embedded in most of wireless sensor nodes are working in 2.4 GHz Industrial, Scientific and Medical (ISM) band because the ISM band is an unlicensed radio frequency band. However, due to this advantage, the spectrum resources in this band are overused by other radio signals such as Wi-Fi and Bluetooth. As a result, strong radio interference is posing a major challenge for the wireless sensor nodes working in the ISM band [6].

Besides strong radio interference, a major challenge for wireless sensor nodes is power consumption. The amazing progress in integrated circuit (IC) manufacturing technology in last decades makes it possible to integrate sensor, microprocessor and transceiver within a tiny systems-on-chip (SoC). However, there has not been much progress in miniaturizing batteries with high-volume power density. The small structure of a wireless sensor node severely constrains the size of a battery that can be fitted on the sensor node. Ultra-thin button cell is the typical battery whose size fits for the wireless sensor node. In most of applications a large number of wireless sensor nodes are deployed in all kinds of harsh environments such that it is impossible to replace the battery for every sensor node. Hence, the lifetime of a sensor node is determined, to a large extent, by the lifetime of its battery. Once the battery is depleted, the lifetime of the sensor node will expire. In order to prolong the lifetime of a sensor node as much as possible (up to several years), the technologies achieving ultra-low power consumption are highly preferred in designing wireless sensor nodes. Otherwise, widespread deployment of wireless sensor nodes would still be infeasible in many applications.

In practice, the radio transceiver module consumes a large portion of energy in a sensor node [7]. Thus, it is absolutely necessary to design an energy-efficient radio transceiver with satisfying robustness to strong co-channel interference for the wireless sensor nodes working in monitoring applications.

1.3 Works of Others

Some solutions have been proposed to achieve low power consumption in radio transceiver modules used for wireless sensor nodes. These solutions are duty-cycling radio, wake-up radio and ultra-wideband (UWB) radio. Each solution has its own weaknesses and limitations.

The sensor node with duty-cycling radio [8]–[10] turns on its radio module in short bursts of time for data transmission and turns it off most of the time to conserve energy. Since the moments when data transmissions happen are uncertain, the duty-cycling radio needs very accurate timing but it is rather difficult to achieve accurate timings in a WSN formed by hundreds of sensor nodes.

The wake-up radio [11]–[13] enables a sensor node to consume very little power when listening the radio channel of interest and wake up its main radio module for data reception when the desired signal is detected. However, the wake-up radio is prone to co-channel interference. If the interference that exists in the channel of interest is strong and persistent, the wake-up radio would frequently activate its main radio module by mistake.

The UWB radio [14]–[16] transmits data by generating pulse trains in specific time intervals while occupying a wide bandwidth. Because of the ultra-wide bandwidth of the UWB radio, it demonstrates its good coexistence with narrowband radios in the same channel. However, due to radio emission regulations [17], it is restricted to short transmission range of a few meters. Moreover, its power consumption is relatively high compared with other low power radio schemes.

For the purpose of combining ultra-low power consumption and excellent robustness to interference in a wireless sensor node that is operating at a low data rate, slow wireless radio is proposed.

1.4 Slow Wireless Radio

The slow wireless radio is a wireless communication scheme which is capable of transferring data at instantaneous bit-rates as low as 100 bits per second (bps) in interference-intensive ISM band while achieving ultra-low power consumption.

Due to the fact that the majority of wireless sensor nodes are applied in detecting slow-varying physical quantities, the data traffic produced by a single sensor node is extremely low. Thus, it is doable for a sensor node to transmit data at a low bit rate. The low bit rate makes it possible to utilize very narrowband (VNB) radio communication in the order of a few kHz.

Furthermore, most of wireless sensor nodes are transmitting and receiving signal in the unlicensed 2.4 GHz ISM frequency band, which is overused by diverse radio signals such as Wi-Fi and Bluetooth. However, it is noteworthy that the dominant radio interference (like Wi-Fi) within the ISM band is spreading over wide bandwidth with relatively low power spectral density. This is because spread spectrum techniques are widely adopted in wireless communication technologies such as Wi-Fi and Bluetooth.

As a result, when a VNB signal with high power spectral density coexists with the wideband interferer with low power spectral density in the ISM band, only a small portion of the interferer overlaps with the VNB signal as illustrated in Fig. 1.1. In brief, the wideband interferer within the ISM band has limited impact on the VNB signal.

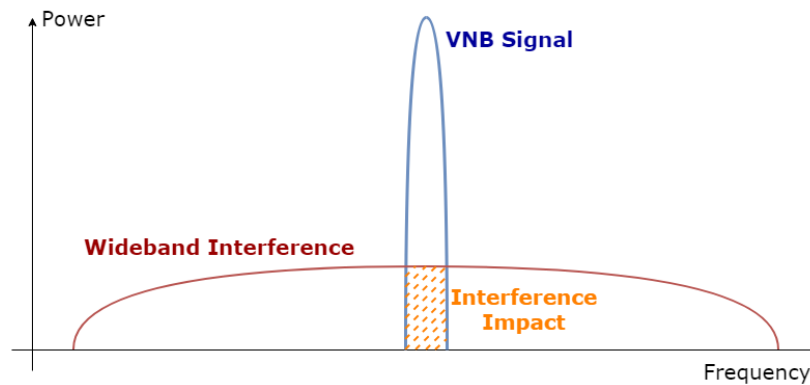


Figure 1.1: Wideband interference impact on very narrowband (VNB) signal

Therefore, the receiver can still capture the VNB signal with acceptable signal-to-noise ratio (SNR) in presence of interference. Moreover, due to higher SNR of the received VNB signal, it is also possible to reduce the transmission power at the transmitter side or relax the noise figure requirement at the receiver side while guaranteeing acceptable bit error ratio (BER) [18].

In a word, the VNB signal enables the slow wireless radio to have reduced transmission power while guaranteeing acceptable BER so that it is possible to achieve low power consumption on the slow wireless radio. To summarize, the slow wireless radio is aiming at the wireless sensor node that is transmitting data:

1. at very-low instantaneous bit-rate of 100~1000 bps
2. in interference-intensive ISM frequency band
3. with ultra-low power consumption
4. by virtue of VNB signal in order of a few kHz.

1.5 Research Problem Description

1.5.1 Research Scope

From the perspective of WSN, there are primarily four functional layers [19], [20] in its hierarchical model as shown in Fig. 1.2:

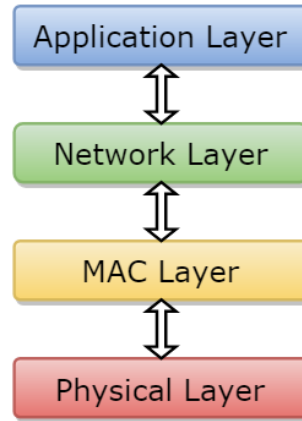


Figure 1.2: The hierarchical model of a wireless sensor network

- The application layer is responsible for formatting data and managing data flow in an optimal scheme for specific applications.
- The network layer is responsible for assigning addresses for nodes and routing data through the network.
- The medium access control (MAC) layer provides error-free transfer of data frames from one node to another over the physical layer. It is also responsible for data stream multiplexing, data frame detection, channel access control and error control.
- The physical layer transmits and receives radio signals over physical medium between any two nodes. This layer is also responsible for carrier frequency selection and generation, radio channel access, signal detection, conversion between bits and radio signals, modulation/demodulation, encoding/decoding, etc.

The hardware structure of a wireless sensor node that is illustrated in Fig. 1.3 generally includes four parts: a sensor, a microcontroller, a wireless transceiver and a power management module. The research scope is focused on the physical layer in a WSN or the wireless transceiver in a wireless sensor node.

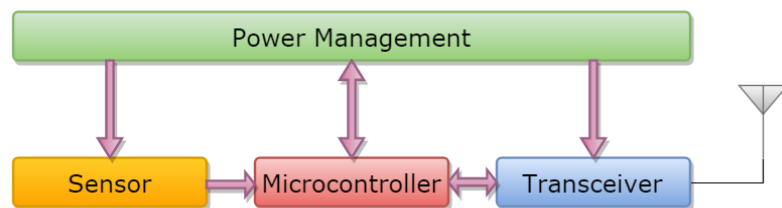


Figure 1.3: Hardware structure of a wireless sensor node [21]

1.5.2 Research Goals

In this research project, a VNB wireless communication testbed will be built up based on software-defined radio (SDR) for the purpose of simulating a real point-to-point VNB wireless communication link and measuring its relevant communication performance. This VNB communication testbed will assist researchers to investigate the feasibility of the VNB communication when it is applied in wireless sensor nodes. Furthermore, this testbed can also serve as a slow-wireless-radio reference design to facilitate the development of a wireless sensor node prototype that is based on the slow wireless radio.

The objectives in this research project are formulated as follows:

1. A VNB transmitter and a VNB receiver will be modeled in software.
2. Successful communication through a channel model between the VNB transmitter and the VNB receiver will be simulated in software.
3. A real-world point-to-point wireless communication testbed that integrates the VNB transmitter and the VNB receiver will be implemented on an SDR platform.
4. Real-time measurements on BER and SNR are conducted in software when the VNB communication link is running on the SDR platform.

In this research project, GNU radio is chosen as SDR development kit and bladeRF as SDR platform.

1.6 Thesis Outline

The remainder of this thesis is organized as follows: Chapter 2 starts with an introduction on software-defined radio and its development toolkit (GNU radio). Then, it introduces software-defined radio platform (bladeRF) in detail. Chapter 3 discusses the design of the VNB transmitter. All of important components in the transmitter including modulator, double differential encoder and repeater are described in this chapter. The design of the VNB receiver is explained in detail by chapter 4. All of main components in the receiver including demodulator, DC-offset compensator, double differential decoder, matched filter and sampler are described in this chapter. Chapter 5 introduces the design, implementation and test on the VNB communication testbed. Finally, we present conclusions and recommendations in chapter 6.

EXPERIMENT PLATFORM

This chapter will start with an introduction to software-defined radio. Then, it will discuss the details with regard to [SDR](#) development toolkit (GNU radio) and [SDR](#) platform (bladeRF) that are used in building up the [VNB](#) communication testbed in this research project. Finally, this chapter will conclude with the discussion of LMS6002D transceiver, which is a core component in bladeRF [SDR](#) platform.

2.1 Software-defined Radio

A radio communication system is a system which can communicate information via electromagnetic waves in radio frequency ([RF](#)) spectrum from 3kHz to 300GHz. A typical radio communication system is composed of a transmitter and a receiver. The transmitter modifies signal for efficient radio transmission and typically contains: modulator, encoder, digital-to-analog converter ([DAC](#)), filter, gain amplifier, mixer and power amplifier ([PA](#)). The receiver captures the [RF](#) signal and reconstructs the original signal. Typical components in the receiver are low-noise amplifier ([LNA](#)), mixer, gain amplifier, filter, analog-to-digital converter ([ADC](#)), demodulator and decoder.

Traditionally, all of the components in a radio communication system used to be implemented mostly by a variety of dedicated electronic circuits. As the processing capability of digital processors evolves rapidly, lots of signal processing tasks which used to be done by dedicated electronic circuits are now carried out by general-purpose high-performance digital processors instead. As a result, an emerging radio technology called software-defined radio (SDR) relies on signal processing software running on a personal computer or a general-purpose processor to implement as many components as possible in a radio communication system. The exact definition of the software-defined radio is as follows:

“Software-defined radio (SDR) is a radio communication system where some components that have been typically implemented in hardware (e.g. amplifiers, filters, modulators/demodulators, encoders/decoders, etc.) are instead implemented by means of software on a personal computer or embedded system [22].”

The block diagram for a typical SDR transceiver system is depicted in Fig. 2.1. It consists of:

- An SDR platform (such as bladeRF SDR).
- An SDR development kit and SDR software (such as GNU radio).
- A personal computer.

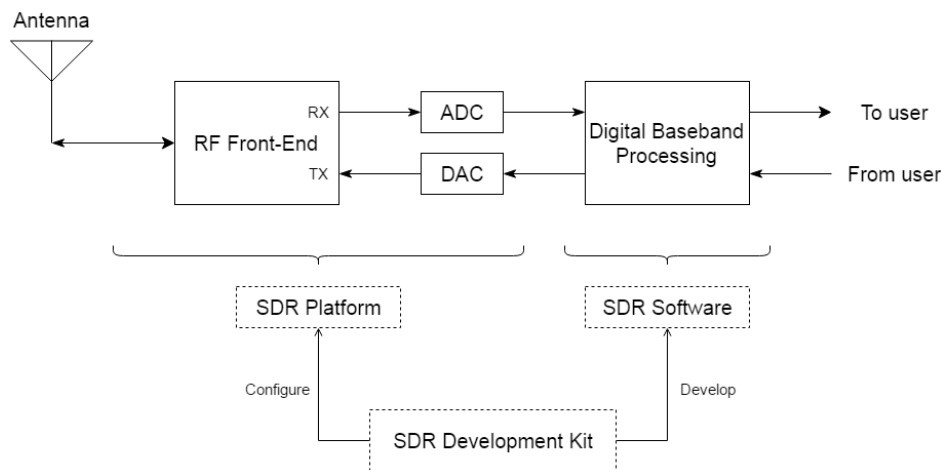


Figure 2.1: SDR transceiver system block diagram [23]

An **SDR** platform includes **RF** front-end and **ADC/DAC**. It serves as a bridge between analog RF signal and digital baseband signal, performing tasks such as mixing and analog-to-digital or digital-to-analog conversion which are impossible to be implemented by software.

Unlike the conventional radio system in which digital baseband processing is performed by dedicated circuit or specialized digital signal processor (**DSP**), an **SDR** system processes all of the digital baseband signals by means of SDR software which is running on a general-purpose computer. Relying on SDR development kit which is installed on the computer, user can easily configure SDR platform and develop SDR software for digital baseband signal processing.

The most significant advantages of the **SDR** system are reconfigurability, reusability and flexibility. The SDR platform can be easily reconfigured to work in varying frequencies, bandwidths, sample rate and so on. The SDR software can be easily developed and modified for a variety of digital baseband processing and the same software codes can be reused in different communication systems. Due to these features, the SDR system is considerably flexible in implementing diverse radio communication systems using the same hardware.

2.2 SDR Development Kit

The **SDR** development kit is a software development platform where all digital signal processing algorithms used in software-defined radio can be developed in flowgraph or coding. In this section, GNU radio is introduced in detail and some other popular SDR development tools are also investigated briefly.

2.2.1 GNU Radio

GNU radio is a free-of-charge and open-source software development toolkit for developing various signal processing algorithms used in radio systems. GNU radio features a rich variety of signal processing blocks, which enable user to easily develop and implement diverse **SDR** applications. Many signal processing blocks which can be found in conventional radio systems can also be found in the library of GNU radio such as filters, modulators/demodulators, coders/decoders and so on [24]. GNU radio development toolkit can be either used together with many kinds of SDR platforms to build up real-world radio communication systems or used alone without any SDR platform to simulate radio communication systems only by software [25].

A graphical user interface called GNU Radio Companion (GRC) is provided by GNU radio software toolkit to facilitate the development of signal processing algorithms by means of data-flow graph. All of the standard signal processing blocks can be found in the library of GNU radio. To achieve high-throughput and high-performance real-time signal processing, all of the signal processing blocks built in the GNU radio are entirely

written in C++ codes. However, the flow graphs in GNU Radio Companion are compiled and run in Python scripts. In case that some special signal processing blocks are missing in the library or unsatisfying users' requirements, users are able to quickly create their own signal processing blocks in Python. In GNU radio, C++ is basically used for low-level signal processing blocks whose computational performance is critical while Python is used for high-level flow graph which serves as a framework to control the data flows among different blocks.

In order to make sure that all of the C++ signal processing blocks are accessible from the Python flow graph, simplified wrapper and interface generator ([SWIG](#)) is used in GNU radio. The [SWIG](#) is a software interface compiler that connects programs written in C/C++ with a variety of high-level scripting languages such as Python. It works by parsing the interface of a C/C++ program and then generating the glue code for the target scripting language so that the scripting language can call or access the C/C++ program via the glue code [26].

Figure 2.2 reveals the GNU radio framework structure where the [SWIG](#) is used as an interface between the signal processing blocks written in C++ codes and the flow graph implemented by Python scripts. In order to communicate with SDR platform, GNU radio has some specialized SDR interface blocks which connect the GNU radio flow graph running on a computer with an SDR platform. The commonly-used SDR interface block is osmocom block, which supports most of available SDR platforms.

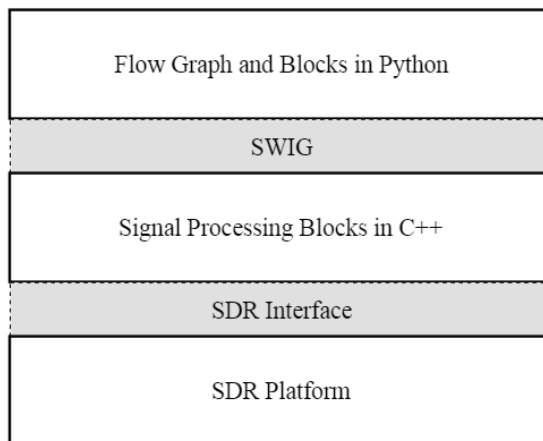


Figure 2.2: GNU radio framework structure [27]

2.2.2 Survey on other SDR development tools

In this section, we briefly investigate other two [SDR](#) development tools: MATLAB Simulink and NI LabVIEW.

- MATLAB Simulink

MATLAB Simulink developed by MathWorks is a graphical programming and development environment for modeling, simulating, analyzing, testing and verifying model-based dynamic systems. User can easily build up a dynamic system by drawing flowgraph. Algorithms written in MATLAB codes can be incorporated into modules in the flowgraph and simulation results from model-based system in flowgraph can be exported into MATLAB for further analysis.

The MATLAB communication system toolbox provides ready-made modules and algorithms for modeling, simulating, analyzing and verifying communication systems in the MATLAB Simulink. Hardware support packages provided by the communication system toolbox enable users to connect their communication system models with [SDR](#) platforms so that their designs can be tested and verified by real over-the-air radio transmission. Some popular SDR platforms are supported by the hardware support packages in the communication system toolbox, including Ettus SDR platforms and Xilinx FPGA-based SDR platforms. These

supported SDR platforms can directly exchange data with communication system models in the MATLAB Simulink. The parameters for these SDR platforms can also be configured in the MATLAB Simulink [28], [29].

- NI LabVIEW

The laboratory virtual instrument engineering workbench (LabVIEW) developed by National Instrument (NI) is a system design and development environment. Its intuitive graphical programming method enables user to design system model in visual flowgraph.

The LabVIEW communication system design suite provides all necessary functional blocks used for building communication system models. C/C++ and MATLAB codes can also be embedded into functional blocks in LabVIEW flowgraph. The LabVIEW communication system design suite supports some popular SDR platforms such as Ettus SDR platforms and NI SDR platforms [30].

Table 2.1 summarizes and compares the main characteristics of the SDR development kits that we have investigated.

Table 2.1: Comparison on SDR development kits

Type	GNU Radio	MATLAB Simulink	NI LabVIEW
Open source	✓	×	×
Free-of-charge	✓	×	×
Graphical programming	✓	✓	✓
C++ support	✓	✓	✓
Python support	✓	×	×
SDR platform support	✓✓✓	✓✓	✓
Online resource	✓✓✓	✓✓	✓

2.2.3 Why is GNU radio chosen?

It can be seen in Tab. 2.1 that GNU radio has more advantages than the other alternatives. GNU radio is chosen as SDR development kit in this project based on the following reasons:

1. It is supported by most of available SDR platforms on the market.
2. It is completely free-of-charge and open-source.
3. It is easy to use because of graphical programming support.
4. Both C++ and Python programming are supported, which gives great flexibility in programming.
5. Rich documentations are available online.

Although the SDR development kit like GNU radio is capable of carrying out complete software-simulation for any radio communication system on a computer, it is necessary to build real radio communication system in order to research and assess the performance of a radio communication system in a real scenario. Therefore, the SDR development kit is usually used with SDR platform to implement a real radio communication system. The following section will introduce SDR platform in detail and provide insights into the bladeRF SDR platform which is adopted in this project.

2.3 SDR Platform

2.3.1 What is SDR platform?

Figure 2.3 illustrates the block diagram of a typical SDR platform. The main function of SDR platform is to carry out conversion between analog RF signal and digital baseband signal.

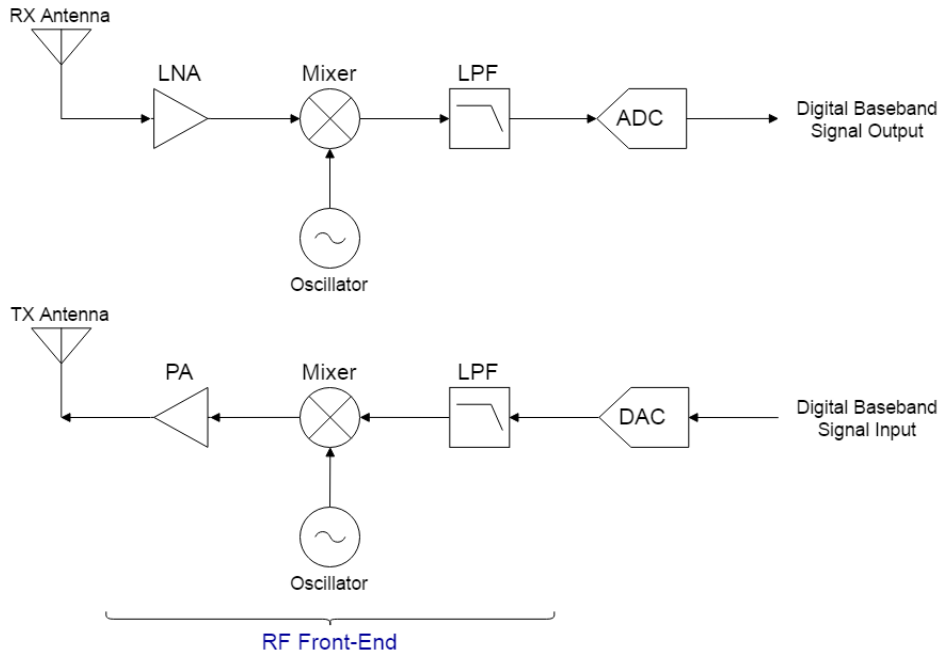


Figure 2.3: SDR platform block diagram [23]

The RF front-end, which is directly connected with transmit (TX)/receive (RX) antennas, is primarily composed of several modules: amplifier (LNA, PA), mixer, oscillator and filter. Most of SDR platforms have two independent signal paths: one for signal reception and the other for signal transmission.

- The signal reception (RX) path:

The RX antenna captures RF signal over the air. Because the captured RF signal usually has weak strength, it has to be amplified by the LNA in the first place. The LNA guarantees that only very low noise is added into the received RF signal after amplification. The mixer down-converts the received RF signal to the baseband signal using carrier which is generated by the oscillator. Then, the low-pass filter (LPF) removes high-frequency contents of the down-converted signal. In the end, the ADC converts the baseband signal from analog domain to digital domain for further digital baseband processing.

- The signal transmission (TX) path:

After digital baseband processing, the digital baseband signal is converted into analog by the DAC. Then, the bandwidth of the analog baseband signal is confined by the LPF. Next, the mixer up-converts the baseband signal to RF signal using carrier from the oscillator output. After up-conversion, the power of the RF signal is boosted up by the PA so that TX antenna can transmit the RF signal over the air for long distance.

An SDR platform has several important specifications which determine its performance and should be known before it is used.

1. **RF tuning range:** The RF tuning range refers to the range of radio frequency spectrum in which the SDR platform can transmit and receive RF signal. The oscillator in Fig. 2.3 generates carrier, whose frequency range is the RF tuning range.

2. **RF bandwidth**: The RF bandwidth refers to the channel bandwidth that the SDR platform allows when it is running. It is the RF bandwidth that determines the minimal and maximal channel bandwidth. The bandwidths of band-pass filter and low-pass filter in the SDR platform determine the RF bandwidth.
3. **ADC/DAC resolution**: The resolution decides the number of bits per digital sample that is going into DAC or out of ADC.
4. **ADC/DAC sample rate**: The sample rate determines the number of digital samples per second at the input of the DAC or at the output of the ADC and it is measured in **sps!** (**sps!**).
5. **Duplex mode**: The term ‘duplex’ signifies bidirectional communication between two devices. That is to say, both devices can transmit and receive signal. There are two duplex modes: full-duplex and half-duplex. The full-duplex SDR platform allows the signal to be transmitted and received simultaneously. In the half-duplex SDR platform, the transmission and reception of the signal must happen alternately. Most of SDR platforms have two independent signal paths that support the full-duplex communication but some low-end SDR platforms can only support the half-duplex communication.
6. **Interface**: The digital baseband signal between SDR platform and host computer is communicated via the interface, which limits the maximum bit rate between the SDR platform and the host computer. Most of SDR platforms support either USB interface or Ethernet interface. Since the Ethernet interface allows higher data throughput than the USB interface, it is mostly applied in high-end SDR platforms.
7. **Power supply**: Power supply via either USB-bus or DC power jack is supported by most of SDR platforms. If more transmission power is needed, the external power supply via DC power jack is recommended.
8. **Software support**: It means SDR software development kits such as GNU radio, MATLAB Simulink and NI LabVIEW that are supported by the SDR platform. Currently, GNU radio is the only SDR development kit that is compatible with most of SDR platforms.
9. **Price**: Because of limited project-budget, a reasonable price is also a key factor to be considered before an SDR platform is chosen.

Some popular SDR platforms will be introduced briefly in the following section.

2.3.2 Survey on SDR platforms

There are a number of commercial SDR platforms. An overview with regard to some commonly-used **SDR** platforms is presented as follows:

- Ettus USRP

Ettus Research is a subsidiary company of National Instruments (NI) that provides the universal software radio peripheral (**USRP**) platforms. The **USRP** series include a wide range of SDR platforms, enabling the RF tuning from direct current (**DC**) to 6 GHz [31].

- Nuand bladeRF

Nuand bladeRF is a low-cost SDR platform which covers the RF range from 300 MHz to 3.8 GHz. It is supported by many SDR development kits including GNU radio and MATLAB Simulink. Furthermore, it is completely open-source. This platform is equipped with an field programmable gate array (**FPGA**) and a LMS6002D field programmable radio frequency (**FPRF**) transceiver [32]. The subsection 2.3.4 will introduce it in detail.

- UmTRX

UmTRX is a dual-channel wideband SDR platform with the RF range spanning from 300 MHz to 3.8 GHz. An [FPGA](#) and two LMS6002D [FPRF](#) transceivers are integrated on the UmTRX board. Its firmware, driver and HDL codes are open-source for users [33].

- HackRF

HackRF designed by Great Scott Gadgets is an open-source SDR platform which can operate in the RF range from 1 MHz to 6 GHz. But the transceiver on HackRF only supports a half-duplex channel [34].

Table 2.2 lists all important specifications on the SDR platforms that we have investigated.

Table 2.2: Comparison on SDR platforms

Designer	Ettus	Ettus	Ettus	Nuand	Osmocom	Great Scott
Type	B2xx	N2xx	X3xx	bladeRF	UmTRX	HackRF
RF range (Hz)	70M ~ 6G	0 ~ 6G	0 ~ 6G	0.3 ~ 3.8G	0.3 ~ 3.8G	1M ~ 6G
RF bandwidth (Hz)	56M	50M	200M	28M	28M	20M
ADC resolution (bits)	12	14	14	12	12	8
DAC resolution (bits)	12	16	16	12	12	8
ADC sample rate (Msps)	61.44	100	200	40	40	20
DAC sample rate (Msps)	61.44	400	800	40	40	20
Channel	1	2	2	1	2	1
Duplex mode	full	full	full	full	full	half
USB interface	✓	✓	✓	✓	✓	✓
Ethernet interface	×	✓	✓	×	✓	×
GNU Radio support	✓	✓	✓	✓	✓	✓
Simulink support	✓	✓	✓	✓	×	×
LabVIEW support	✓	✓	✓	×	×	×
Price (\$)	675	1515	3900	420	950	299

2.3.3 Why is bladeRF chosen?

In this project a [VNB](#) wireless communication testbed is built based on bladeRF x40 SDR platform. There are several reasons to choose the bladeRF as SDR platform:

1. The testbed is designed to be operating in the 2.4G [ISM](#) band. The RF tuning range (0.3~3.8GHz) of the bladeRF SDR platform covers the band.
2. The target signal bandwidth is only 1~10kHz in the testbed. The minimum channel bandwidth of the bladeRF SDR platform is 1.5MHz. Thus, it is wide enough for the required signal bandwidth in the testbed.
3. The bladeRF SDR platform supports full-duplex communication mode in which the transmission and reception of signal happens simultaneously. Therefore, the transmitter and receiver of the testbed can be implemented on the same bladeRF SDR platform.
4. The bladeRF SDR platform is open-source, well-documented and supported by abundant online tutorials and examples.
5. The bladeRF SDR platform is completely supported by GNU radio.
6. The total cost is relatively low.

7. The telecommunication engineering group at the University of Twente possesses the bladeRF SDR platform at hand and it is possible to use the existing experiences from this group regarding the use of the bladeRF platform.

The bladeRF SDR platform that is used in this project is introduced in the next section.

2.3.4 bladeRF Introduction

The Nuand bladeRF shown in Fig. 2.4 is a software-defined radio platform designed to carry out various radio communication experiments. It has a wide tuning range from 300 MHz to 3.8 GHz, covering a wide range of radio frequency spectrum. Independent TX and RX signal paths allow the bladeRF to achieve full-duplex communication mode, which means that signal transmission and reception can be conducted simultaneously on the same platform. The USB 3.0 port enables high-speed data transfer between the bladeRF SDR and host computer. Meanwhile, it supports full USB-power supply over the same USB cable. In analog-to-digital or digital-to-analog data conversion, an analog continuous-time signal corresponds to a stream of digital discrete-time samples with each digital sample representing the amplitude of the analog signal at the sampling moment. The sample rate on the bladeRF can be set arbitrarily in the range from 160ksps! up to 40Msps! with every digital sample being 12-bit wide. The bladeRF is able to implement the channel bandwidth from 1.5MHz to 28MHz. In addition, the bladeRF is equipped with high-performance Altera Cyclone IV FPGA on board, which is fully programmable via JTAG port. There are two types of bladeRF, each of which has a different number of logic elements (LE)s in its FPGA. BladeRF x40 has an FPGA with 40k LEs while bladeRF x115 has an FPGA with 115k LEs [35]. There are two extensible gold-plated RF subminiature version A (SMA) connectors on the bladeRF platform, which can be used for external RF TX and RX antenna ports. The device driver for the bladeRF platform is stably compatible with the mainstream Linux, Windows and Mac operating systems. The last but not least, the bladeRF SDR platform is supported by many popular third-party SDR development kits such as GNU radio and MATLAB Simulink. The main technical specifications of bladeRF platform are listed in Table 2.3.

Table 2.3: bladeRF specifications

Type	bladeRF x40	bladeRF x115
RF range	0.3~3.8GHz	
Channel bandwidth	1.5~28MHz	
Duplex mode	full	
Resolution (ADC/DAC)	12-bit	
Sample rate (ADC/DAC)	0.16~40Msps	
Interface	USB 3.0 (5Gbps)	
Microcontroller	Cypress FX3	
FPGA	Altera Cyclone IV	
FPGA logic elements	40k	115k
Open source	HDL + firmware + schematics	
Software support	GNU radio / MATLAB Simulink	
Cost	\$420	\$650



Figure 2.4: Nuand bladeRF software-defined radio platform [36]

2.3.5 bladeRF Internal Structure

The architecture within the bladeRF is illustrated in Fig. 2.5. The following paragraphs will briefly discuss the functionalities of main blocks in the bladeRF architecture.

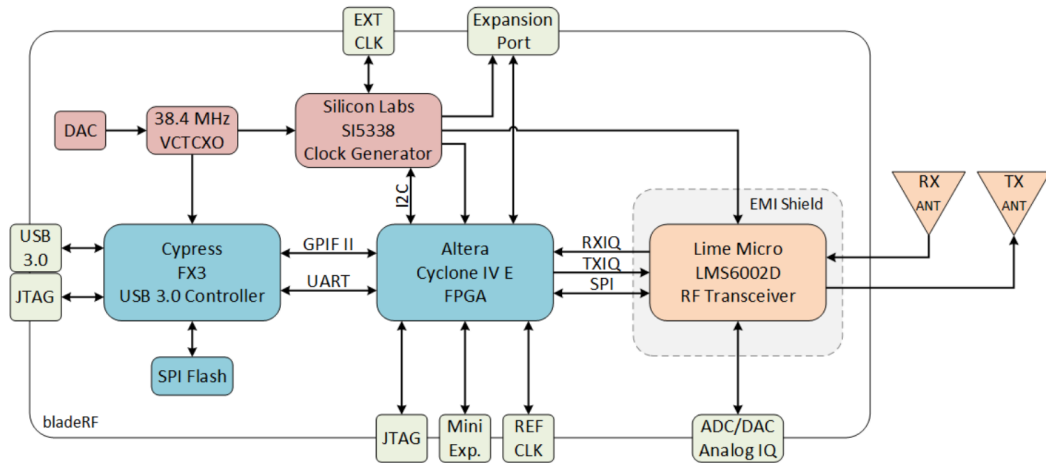


Figure 2.5: Nuand bladeRF architecture [35]

The USB 3.0 port is the interface between a host computer and the bladeRF platform with up to 5Gbps high-throughput. This USB port is also backwards compatible with USB 2.0 (480Mbps) but has some sample rate limitations. The resolution of the [ADC](#) or [DAC](#) on the bladeRF is 12-bit. The theoretical maximum sample rate on the bladeRF is 40 **Msp/s**!. As a result, the maximum bit rate via USB port should be $40M \times 12 = 480Mbps$. But, in practice, many host computers which only support USB 2.0 port can not fully achieve the maximum USB 2.0 throughput (480Mbps) so that the [ADC](#) or [DAC](#) on the bladeRF is unable to reach its maximum sample rate (40Msp/s). In addition, the USB port can also deliver power to the bladeRF while it is transferring data. For those who want to run the bladeRF standalone, the bladeRF also supports 5V DC power input jack.

The bladeRF is equipped with a Cypress FX3 USB 3.0 controller with an integrated 200MHz ARM926EJ-S core, which controls data to communicate between the USB 3.0 port and the parallel GPIF II interface (General Programmable Interface Generation II). The GPIF II interface is directly connected with Altera Cyclone IV [FPGA](#). The serial peripheral interface ([SPI](#)) flash connected with the Cypress FX3 controller is a 4 MB flash memory which stores firmware code for the Cypress controller and the Altera FPGA. When the bladeRF platform is booted up, the firmware will be loaded into the Cypress controller and the Altera FPGA. Moreover, the firmware stored in the SPI flash memory can be updated via USB or JTAG port.

The FPGA not only provides the interface between the Cypress FX3 USB 3.0 controller and the Lime Micro LMS6002D RF transceiver but also carries out additional signal buffering and processing as well as control on the LMS6002D RF transceiver. The data is communicated through independent RXIQ (for receive IQ signal) and TXIQ (for transmit IQ signal) paths between the FPGA and the RF transceiver, enabling the bladeRF platform to work in full-duplex mode. The Altera FPGA on the bladeRF possesses embedded hardware 18×18 multipliers and many general logic elements for custom digital signal processing and hardware acceleration. Since the bladeRF is designed to be fully programmable, all firmware codes for the Cypress controller and VHDL codes for the FPGA logic are available for free modification. The Cypress controller and the Altera FPGA on the bladeRF platform can be reprogrammed through JTAG port or USB port. With the free development tool suites provided by the hardware vendors (Cypress and Altera), controller firmware and FPGA logic can be easily customized to fulfill special signal processing tasks.

A voltage-controlled and temperature-compensated crystal oscillator (VCTCXO) in the bladeRF is running at 38.4MHz and supplying a stable frequency reference with the accuracy of ± 1 ppm for the Cypress controller and the SI5338 clock generator. The 16-bit DAC before the crystal oscillator (VCTCXO) provides precise voltage control on the VCTCXO to allow precise tuning of the frequency reference output from VCTCXO. The Silicon Labs SI5338 clock generator synthesizes highly-accurate and low-jitter clock signals for the Altera FPGA and the Lime Micro RF transceiver. The ADC and DAC within the RF transceiver are driven by clocks from the SI5338 clock generator so that the flexible setting of ADC or DAC sample rate is possible on the bladeRF.

The core component in the bladeRF SDR platform is the Lime Microsystems LMS6002D field programmable RF transceiver chip, which is placed in a silver electromagnetic interference (EMI) shield as shown in Fig. 2.4. The EMI shield is designed to protect the RF transceiver from the outside electromagnetic interference and meanwhile to minimize the amount of electromagnetic interference radiated by the RF transceiver. Since the LMS6002D RF transceiver chip contains all of the amplifiers, mixers, filters, ADC and DAC, it performs all of the RF front-end tasks such as gain control, mixing, filtering and A/D or D/A conversion. The RF antennas can be directly connected with the LMS6002D transceiver for RF signal transmission [36], [37].

2.4 LMS6002D Transceiver

2.4.1 Introduction

The most pivotal component in bladeRF SDR platform is the Lime Microsystems LMS6002D RF transceiver. It is a FPRF transceiver IC containing LNA, PA, TX/RX mixer, LPF, TX/RX variable-gain amplifier (VGA) and ADC/DAC. It converts digital baseband signal up to analog RF signal for radio transmission or converts received analog RF signal down to digital baseband signal for further signal processing.

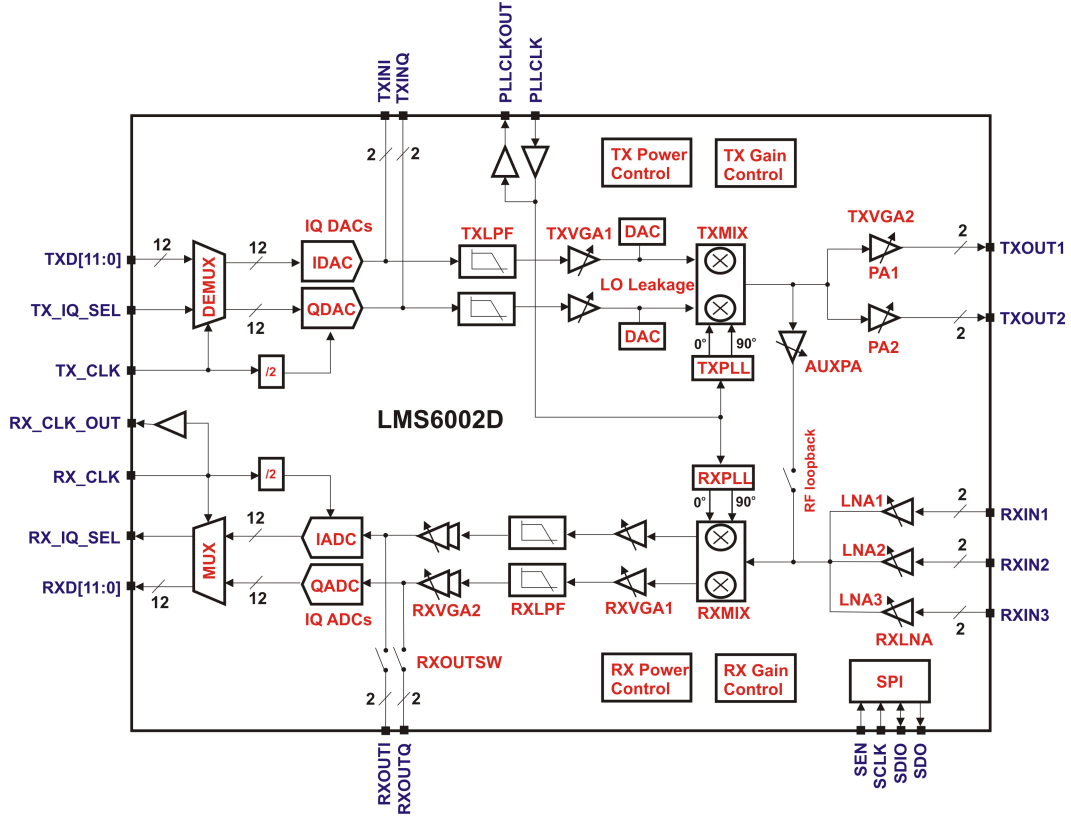
The transceiver IC can be finely tuned in a wide range of frequency spectrum from 300MHz to 3.8GHz (including both licensed and unlicensed bands), making it applicable to many wireless communications. The transceiver can run with 16 different programmable channel bandwidths from 1.5MHz up to 28MHz. Both the frequency and the channel bandwidth of the LMS6002D transceiver are fully controlled by a set of internal registers, which can be easily configured by software. The LMS6002D has a standard SPI port used for accessing and configuring its internal control registers. Moreover, it has two independent digital IQ signal interfaces (one for TX IQ baseband signal and the other for RX IQ baseband signal), both of which can directly interface with any digital signal processor [38]. The main technical specifications for LMS6002D transceiver [39] is summarized in Tab. 2.4.

Table 2.4: The specifications for LMS6002D transceiver

Type	LMS6002D
RF range (GHz)	0.3~3.8
Channel bandwidths (MHz)	1.5, 1.75, 2.5, 2.75, 3, 3.84, 5, 5.5, 6, 7, 8.75, 10, 12, 14, 20, 28
Duplex mode	full
ADC	dual 12-bit paths
DAC	dual 12-bit paths

2.4.2 Internal Structure

The LMS6002D RF transceiver whose internal architecture is depicted in Fig. 2.6 has two independent signal paths: transmit (TX) path and receive (RX) path. Each signal path has its own mixer that is driven by its own phase-locked loop (PLL)s (TXPLL or RXPLL). Although both the TXPLL and RXPLL share the same external reference clock source (PLLCLK), they can independently generate different carrier frequencies between 0.3 and 3.8GHz for their own mixer. Each PLL produces a pair of cosine carriers with 90 degrees of phase difference. The TX and RX signal paths are explained in the following.

**Figure 2.6:** LMS6002D architecture [39]

- Transmit Path:

Along the TX path, digital in-phase and quadrature (IQ) samples from a digital signal processor are provided to the LMS6002D on the 12-bit parallel TX data bus (TXD). A digital IQ sample can be viewed as a digital complex sample, where in-phase (I) sample and quadrature (Q) sample are real and imaginary parts respectively. The I and Q samples alternately appear on a stream of IQ samples. The TX IQ signal select-flag-bit (TX_IQ_SEL)

identifies I and Q samples from a stream of IQ samples on the TX data bus (TXD). When an I sample appears on the TX data bus (TXD), the IQ select-flag-bit (TX_IQ_SEL) is set to 1, controlling the de-multiplexer to push the I sample to the in-phase DAC (IDAC). When a Q sample is available on the TX data bus (TXD), the IQ select-flag-bit (TX_IQ_SEL) is set to 0, controlling the de-multiplexer to push the Q sample to the quadrature DAC (QDAC).

The IQ DAC is made up of a 12-bit IDAC and a 12-bit QDAC. The IQ DAC sample rate varies from 0.16~40MHz, meaning that the IQ DAC is capable of converting 0.16~40M I and Q samples per second. Since an IQ sample is the combination of an I sample and a Q sample, the input IQ sample rate is the half of the IQ DAC sample rate, ranging from 0.08~20MHz. After the IQ DAC, the digital I and Q samples become analog I and Q ones.

The TX LPF delimits the bandwidths of I and Q baseband signals and smoothens the waveforms of I and Q baseband signals. The LPF has 16 selectable bandwidths, spanning from 0.75~14MHz. Hence, after the TX LPF, the bandwidth of the I or Q baseband signal is limited in the range of 0.75~14MHz. Since the passband bandwidth of the channel is double the bandwidth of the I or Q baseband signal, the channel bandwidth is in the range of 1.5~28MHz.

On the TX branch there are two variable-gain amplifiers (TXVGA1 and TXVGA2), whose amplification gains can be configured by internal registers. TXVGA1, which is implemented on the I and Q branches, amplifies the I and Q baseband signals. Located between TXVGA1 and TXVGA2 is the TX mixer. After being amplified by the TXVGA1, the I and Q signals are mixed with the outputs of the transmit PLL (TXPLL) to up-convert both the I and Q baseband signals to the IQ-modulated RF signal. The RF signal is then split into two TX RF branches and amplified by TXVGA2, which serves as TX power amplifier (PA). There are two RF output ports (TXOUT1 and TXOUT2) for RF signal transmission but only one of them is active at any time.

- Receive Path:

Along the RX path, there are three different RF input ports (RXIN1, RXIN2 and RXIN3) for multi-band receptions. RXIN1 receives 0.3~2.8GHz RF signal input. RXIN2 receives the RF signal in the range of 1.5~3.8GHz. RXIN3 is the input port for 0.3~3.0GHz RF signal.

On the RX branch, there are three amplifiers: RXLNA, RXVGA1 and RXVGA2. The received RF signal is usually weak. So, it is first amplified by low-noise amplifier (RXLNA). Next, the RF signal is mixed with the outputs of the receive PLL (RXPLL) to down-convert the IQ-modulated RF signal to I and Q baseband signals. The variable-gain amplifier (RXVGA1) is intended for automatic gain control (AGC) to clamp the amplitudes of the I and Q baseband signals within a suitable level. Then, the I and Q baseband signals pass through RX low-pass filter (RXLPF). The RXLPF restricts the bandwidths of the I and Q baseband signals, allowing the I and Q signals within the selected bandwidth to pass through while rejecting the I and Q signals outside the chosen bandwidth.

The IQ ADC has a certain dynamic range between its noise floor and its maximum level of the input amplitude. In a communication system, the signal strength may vary dramatically. If the signal strength is too small, the signal may get overwhelmed by the quantization noise of the ADC. Conversely, if the signal strength is so large that it goes beyond the maximum input level of the ADC, it may result in the saturation of the ADC output. Therefore, prior to the IQ ADC, the I and Q baseband signals have to go through another automatic gain control performed by RXVGA2 in order to further limit the I and Q signal amplitudes within the dynamic range of the ADC. Like the IQ DAC, the IQ ADC also consists of a 12-bit in-phase ADC (IADC) and a 12-bit quadrature ADC (QADC). An I sample and a Q sample constitutes an IQ sample. The IQ ADC allows the sample rate between 0.16~40MHz. In other word, the IQ ADC can produce 0.08~20M digital IQ samples per second or equivalently 0.16~40M digital I and Q samples per second.

After the IQ ADC, the digital I and Q samples are alternately output on the 12-bit parallel RX data bus (RXD) by the multiplexer. Moreover, the RX IQ signal select-flag-bit (RX_IQ_SEL) tells whether an I sample or a Q sample is available on the RX data bus (RXD). Finally, digital IQ samples on the RX data bus (RXD) are streaming into following digital signal processor for further processing.

TRANSMITTER DESIGN

In this chapter, the design of the transmitter will be explained in detail. This chapter will start with giving an overview about the transmitter structure. Then, follows the discussion of BPSK modulator where incoming binary bits are mapped into complex symbols. The rest of this chapter will explain the following issues. How are the complex symbols encoded by double differential encoder? Why do the encoded symbols pass through the repeater? How is IQ modulator working?

3.1 Transmitter Structure Overview

The transmitter part is made up of several blocks as shown in Fig. 3.1.

- Random signal source generates a stream of random binary bits to be transmitted.
- BPSK modulator maps a stream of binary bits (1's and 0's) into a stream of complex symbols ($-1+0j$ and $1+0j$).
- Double differential encoder encodes a stream of complex symbols ($1+0j$ and $-1+0j$) and outputs an encoded sequence of complex symbols.
- Repeater takes in a complex symbol with one complex sample per symbol. After repeating an incoming sample N times, it outputs a complex symbol with N complex samples per symbol.
- IQ modulator accepts a complex sample and splits it into real part (in-phase component) and imaginary part (quadrature component). Both the in-phase and quadrature components are converted by ADC from digital domain to analog domain. Then, both the in-phase and quadrature signals are up-converted to RF signal that is transmitted via antenna. In contrast with the preceding blocks which are implemented by software, the IQ modulator is implemented by SDR platform.

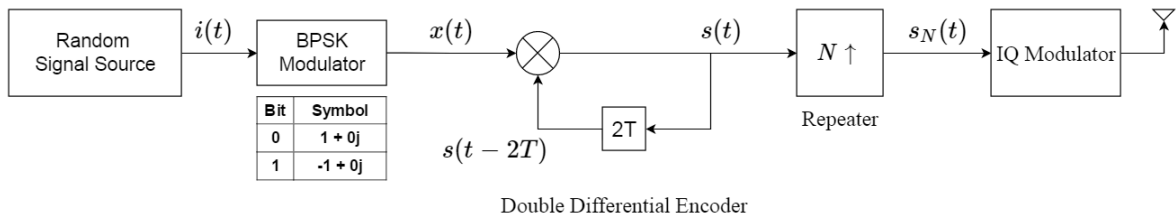


Figure 3.1: Transmitter block diagram

The following sections will present the detailed insights into BPSK modulator, double differential encoder and repeater. Working principle of IQ modulator/demodulator is explicated in appendix A.

3.2 BPSK Modulator

The phase shift keying (PSK) modulation is a digital modulation method which conveys information bits by the phase of a modulated signal. The PSK modulator maps information bits into a finite number of phases while the IQ modulator mixes the PSK-modulated signal with the carrier. After the IQ modulator, the carrier signal to be transmitted contains several phases.

Let $i(t)$ be the information bit from the signal source at any time instant t while $x(t)$ denotes the complex symbol output by PSK modulator at time t . The relation between $x(t)$ and $i(t)$ can be expressed in the following equations:

$$\phi(t) = \frac{2\pi \cdot i(t)}{M} \quad i(t) = 0, 1 \dots M - 1 \quad (3.1)$$

$$x(t) = A \cdot e^{j \cdot \phi(t)} \quad (3.2)$$

where A is the magnitude of the complex symbol, $\phi(t)$ denotes the phase that is mapped from $i(t)$ and M refers to the modulation order. The order M determines the number of different symbols that a modulation scheme can transmit. Let N denote the number of bits that a symbol conveys. N and M have the following relation:

$$N = \log_2 M \quad (3.3)$$

For binary phase shift keying (BPSK) modulation scheme, $M = 2$ and $N = 1 \text{ bit/symbol}$. So, the BPSK scheme can transmit two different complex symbols with each symbol standing for one bit. For quadrature phase shift keying (QPSK) modulation scheme, $M = 4$ and $N = 2 \text{ bits/symbol}$. So, the QPSK scheme can transmit four different complex symbols with each symbol representing two bits.

According to (3.1), BPSK scheme has two complex symbols whose phases are 0 and π . According to (3.2) and assuming unit magnitude ($A = 1$) for each complex symbol for simplification, the output $x(t)$ of the BPSK modulator in Fig. 3.1 is expressed as follows:

$$\begin{aligned} x(t) &= e^{j \cdot \phi(t)} \\ &= \cos \phi(t) + j \cdot \sin \phi(t) \\ &= \begin{cases} 1 + j0 & i(t) = 0 \\ -1 + j0 & i(t) = 1 \end{cases} \end{aligned} \quad (3.4)$$

The real and imaginary parts of a complex symbol $x(t)$ at time t are called in-phase component $I(t)$ and quadrature component $Q(t)$ respectively:

$$I(t) = \text{Re}\{x(t)\} = \cos \phi(t) = \begin{cases} 1 & i(t) = 0 \\ -1 & i(t) = 1 \end{cases} \quad (3.5)$$

$$Q(t) = \text{Im}\{x(t)\} = \sin \phi(t) = 0 \quad (3.6)$$

The constellation diagram of BPSK in Fig. 3.2 reveals the mapping relationship between information bits and complex symbols. The bits 0 and 1 are mapped into the complex symbols $1 + 0j$ and $-1 + 0j$. Both the symbols are lying entirely on the real axis without any projection on the imaginary axis.

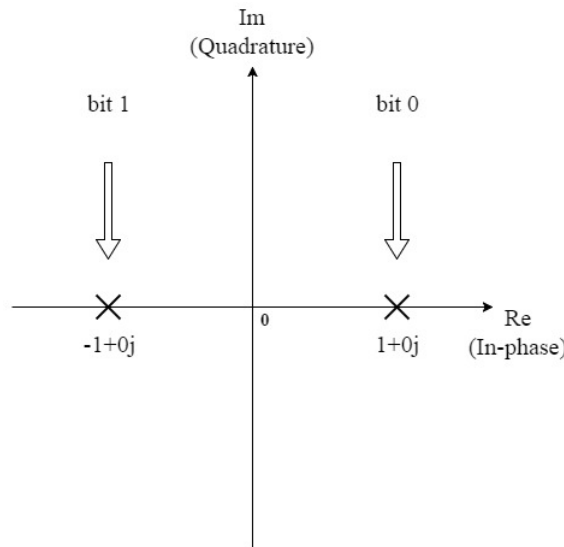


Figure 3.2: BPSK mapping

In all PSK schemes, BPSK is the simplest one since there are only two symbols used. Compared with other PSK schemes, the distance between different symbols is the largest in BPSK scheme. As a result, BPSK demodulator can make a correct detection on a symbol while tolerating the highest level of signal distortion and background noise. However, BPSK is only able to modulate 1 bit per symbol. Thus, it is not a suitable choice for high bit-rate transmission [40].

On one hand, this transmitter is applied for wireless transmission at very-low bit rate under strong interference and background noise. On the other hand, BPSK scheme can achieve a required low BER with less power while other high-order PSK schemes need more power to maintain the same BER. Therefore, the BPSK is selected as the modulation scheme. Besides, the BPSK scheme leads to simpler differential encoder and decoder.

3.3 Double Differential Encoder Design

Due to the presence of an arbitrary and unknown phase shift that is introduced by the communication channel, the received symbols may contain a phase offset. It means that the received constellation points are rotated by a phase angle. If the phase angle is greater than 90 degrees, the constellation points fall onto the opposite-half of complex plain. As a result, the BPSK demodulator is unable to distinguish the correct constellation points. In order to overcome the phase offset, a differential encoder is needed prior to IQ modulator.

Furthermore, due to mismatch between transmitter and receiver's local oscillators, there is usually a frequency offset in the down-converted baseband signal at receiver side. Another reason which results in frequency offset is communication channel, which introduces a random frequency shift into the carrier signal because of Doppler effect or channel fading. The introduced frequency offset will constantly rotate the received constellation points over time, resulting in incorrect symbol detection by BPSK demodulator. A single differential encoder is sensitive to frequency offset. Thus, another single differential encoder is needed to eliminate the frequency offset, resulting in a double differential encoder.

The double differential encoder is composed of two single differential encoders which are connected in series as illustrated in Fig. 3.3. Each single differential encoder outputs a complex symbol which depends not only on the current input symbol but also on the previous output symbol.

Let us define $x(t)$ as input signal, $m(t)$ as intermediate signal between the output of the first encoder and the input of the second encoder, $s(t)$ as the output signal which is going to be transmitted. As $x(t)$ is a

BPSK-modulated complex signal, $m(t)$ and $s(t)$ also take values +1 or -1.

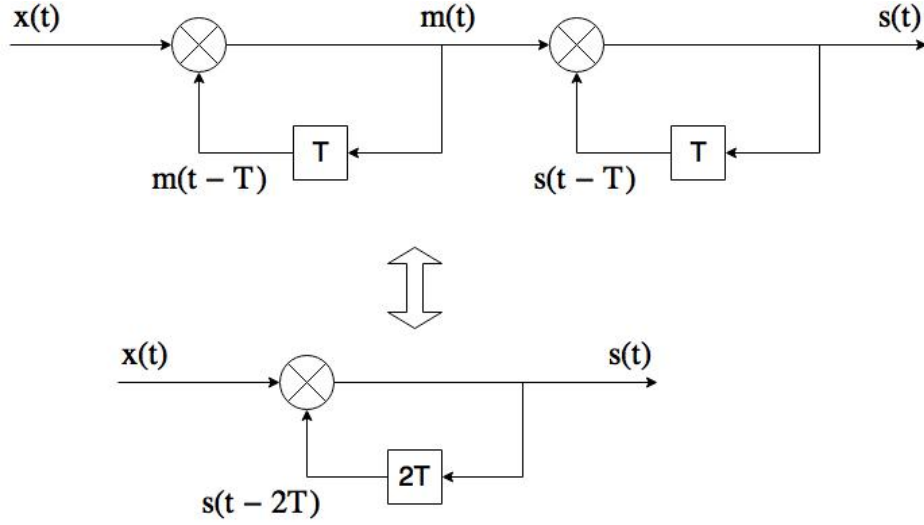


Figure 3.3: Double differential encoder structure

The output signal $s(t)$ is derived from the second encoder.

$$s(t) = s(t - T) \cdot m(t) \quad (3.7)$$

From (3.7), we can get:

$$s(t - T) = s(t - 2T) \cdot m(t - T) \quad (3.8)$$

The intermediate signal $m(t)$ is derived from the first encoder.

$$m(t) = m(t - T) \cdot x(t) \quad (3.9)$$

Substituting (3.8) and (3.9) into (3.7) while considering that $m(t) \cdot m(t) = 1$, we have:

$$\begin{aligned} s(t) &= s(t - 2T) \cdot m(t - T) \cdot m(t - T) \cdot x(t) \\ &= s(t - 2T) \cdot x(t) \end{aligned} \quad (3.10)$$

Hence, the two structures in Fig. 3.3 are equivalent according to the above derivations [41] [42]. For BPSK-modulated signals, a single differential encoder with the delay of two symbol periods $2T$ can supplant two cascaded single differential encoders with the delay of one symbol period T to implement double differential encoding and meanwhile simplify the structure. With regard to how the double differential encoder and decoder are working together to remove phase and frequency offsets, please refer to the appendix B.

3.4 Repeater

The repeater repeats each incoming sample several times. Before the repetition, every symbol is represented by a sample. After the repetition by an integer factor N , every symbol is represented by N samples. The purpose of the repeater is to adjust the symbol rate¹ on the testbed.

¹symbol rate = bit rate due to BPSK modulation

The whole communication testbed has a definite sample rate (or sample frequency) f_{sample} which is determined by ADC and DAC. As a result, the sample duration $T_{sample} = \frac{1}{f_{sample}}$ is fixed.

Let T_{symbol} be the symbol duration and R_s be the symbol rate (or symbol frequency). The symbol rate (sometimes called baud rate) is the number of symbols which are transmitted per second through the communication channel. Before the repetition, one symbol contains one sample.

$$T_{symbol} = T_{sample} \quad (3.11)$$

The sps is referred to as the number of samples per symbol. After the repetition by an integer factor sps , one symbol includes sps samples.

$$T_{symbol} = sps \cdot T_{sample} \quad (3.12)$$

The symbol rate (or baud rate) is determined by the sample rate and the number of samples per symbol (sps).

$$R_s = \frac{f_{sample}}{sps} \quad (3.13)$$

The bit rate R_b is dependent on the symbol rate R_s as well as the number of bits that a symbol can represent. The number of bits that a symbol can represent relies on the modulation scheme in use. For BPSK modulation, one symbol represents one bit.

$$T_{bit} = T_{symbol} = sps \cdot T_{sample} \quad (3.14)$$

$$R_b = R_s = \frac{f_{sample}}{sps} \quad (3.15)$$

Listed in Table 3.1 are several bit rates R_b and corresponding sps (samples per symbol) when the sample rate f_{sample} is fixed at 80 kHz. As we can see in Table 3.1, increasing sps leads to decreasing bit rate R_b .

Table 3.1: Bit rate and sps when $f_{sample} = 80$ kHz

R_b (bps)	sps
10000	8
1000	80
100	800

Let $s(t)$ be the input of the repeater and $s_N(t)$ be the output of the repeater. Since the output $s_N(t)$ is a BPSK-modulated complex signal, it has in-phase component $I(t)$ (real part) and quadrature component $Q(t)$ (imaginary part) as follows:

$$I(t) = \text{Re}\{s_N(t)\} = \pm 1 \quad (3.16)$$

$$Q(t) = \text{Im}\{s_N(t)\} = 0 \quad (3.17)$$

We ignored the quadrature component $Q(t)$ because it is always zero. The in-phase component $I(t)$ is a sequence of non-return-to-zero (NRZ) rectangular pulses. A rectangular pulse shown in Fig. 3.4 represents one bit or one symbol. Thus, the pulse duration is equal to the bit duration T_{bit} or symbol duration T_{symbol} .

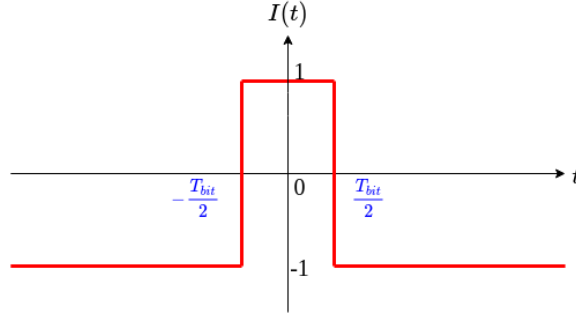


Figure 3.4: Non-return-to-zero pulse in time domain

The in-phase component $I(t)$ is a train of rectangular pulses in time domain. The Fourier transform of $I(t)$ is $I(f)$ which is a sinc function in frequency domain. Figure 3.5 displays the power spectrum for $I(f)$.

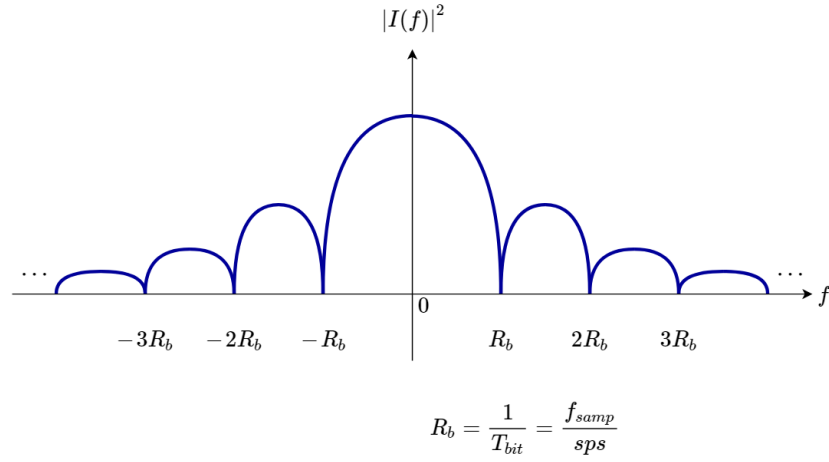


Figure 3.5: Power spectrum for non-return-to-zero pulses

The power spectrum $|I(f)|^2$ is a sinc function with zero-crossing points at $\pm k \cdot R_b$ where k is any integer and R_b is bit rate. Since the main lobe contains most of the pulse energy, the baseband bandwidth BW of the BPSK-modulated signal is equivalent to the bandwidth occupied by the main lobe.

$$BW = R_b = \frac{1}{T_{bit}} = \frac{f_{samp}}{sps} \quad (3.18)$$

When sps (samples per symbol) increases, one symbol contains more samples and an NRZ rectangular pulse has longer pulse duration T_{bit} . Accordingly, the pulses vary less frequently and the bit rate R_b decreases. The main lobe compresses and the baseband bandwidth BW of the BPSK-modulated signal becomes narrower.

As sps decreases, one symbol includes less samples and the pulse duration T_{bit} becomes shorter. Correspondingly, the pulses change more frequently and the bit rate R_b increases. The main lobe outspreads and the baseband bandwidth BW of the BPSK-modulated signal becomes wider. To summarize, by adjusting the parameter ' sps ' (samples per symbol) in the repeater, transmission at different bit rates can be easily achieved on the testbed.

3.5 IQ Modulator

The IQ modulator up-converts the baseband signal to the RF passband with carrier frequency f_c . After that, the RF carrier signal is transmitted over the air by antenna. In this project, the IQ modulator is implemented

by bladeRF [SDR](#) platform. In GNU Radio, there is a special block called ‘osmocom sink’ which is an interface between the digital baseband signal and the bladeRF. The structure of the IQ modulator is shown in Fig. 3.6.

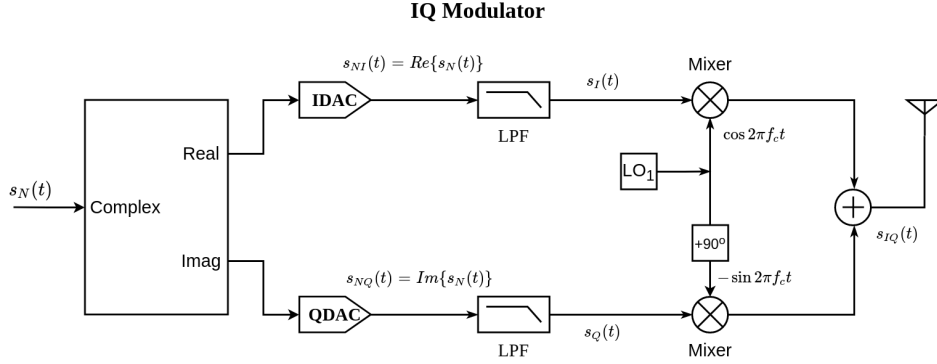


Figure 3.6: IQ modulator structure

The input signal $s_N(t)$ to the IQ modulator is a stream of complex samples. In this case, the input $s_N(t)$ is a BPSK-modulated complex signal with information bits stored in phase $\phi(t)$.

$$\begin{aligned}
 s_N(t) &= e^{j\cdot\phi(t)} \\
 &= \cos\phi(t) + j \cdot \sin\phi(t) \\
 &= \begin{cases} 1 + j0 & \phi(t) = 0 \\ -1 + j0 & \phi(t) = \pi \end{cases}
 \end{aligned} \tag{3.19}$$

At first, the complex signal $s_N(t)$ is decomposed into real part along the in-phase path and imaginary part along the quadrature path. The in-phase digital-to-analog converter ([IDAC](#)) and quadrature digital-to-analog converter ([QDAC](#)) perform digital-to-analog conversions on the real and imaginary parts of the signal $s_N(t)$ respectively. After the digital-to-analog conversion, both the real part $Re\{s_N(t)\}$ and imaginary part $Im\{s_N(t)\}$ become analog signals. Let us assume $s_{NI}(t) = Re\{s_N(t)\}$ and $s_{NQ}(t) = Im\{s_N(t)\}$:

$$s_{NI}(t) = Re\{s_N(t)\} = \cos\phi(t) = \pm 1 \tag{3.20}$$

$$s_{NQ}(t) = Im\{s_N(t)\} = \sin\phi(t) = 0 \tag{3.21}$$

In time domain, the real part $s_{NI}(t)$ is a sequence of [NRZ](#) rectangular pulses (1 and -1) while the imaginary part $s_{NQ}(t)$ remains at zero. Therefore, we could ignore the imaginary part $s_{NQ}(t)$ along the quadrature path. Let us assume that $S_{NI}(f)$ is the Fourier transform of $s_{NI}(t)$, that is $S_{NI}(f) = \mathcal{F}\{s_{NI}(t)\}$. In frequency domain, the $S_{NI}(f)$ is a sinc function whose side lobes outreach and occupy unlimited bandwidth as shown in Fig. 3.7.

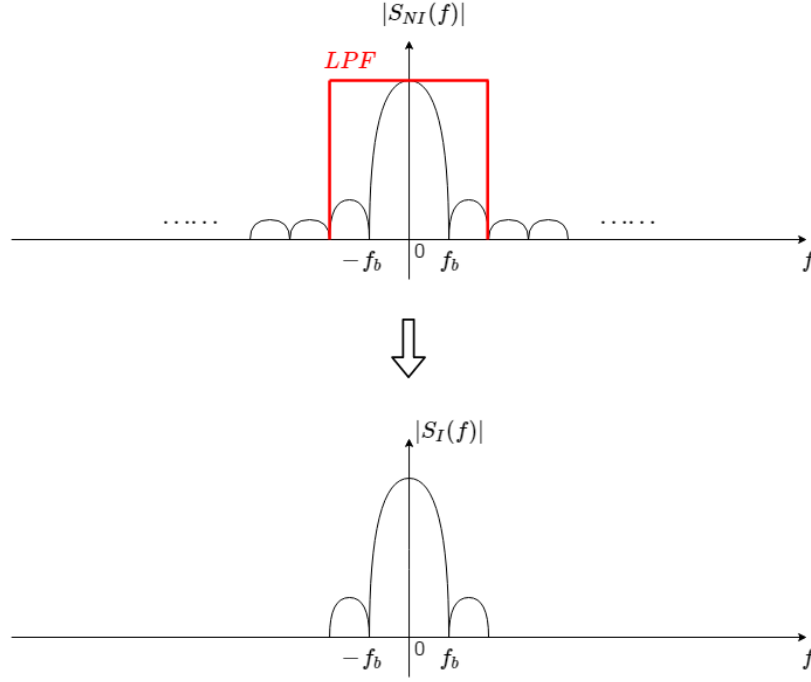


Figure 3.7: In-phase signal spectrum and its filtered spectrum

Then, the real part $S_{NI}(f)$ passes through the **LPF** where high-frequency side lobes are suppressed while low-frequency main lobe is allowed to pass through. Due to the suppression over the high-frequency side lobes, the **LPF** output along the in-phase path $s_I(t)$ is a train of slightly distorted pulses instead of strictly rectangular pulses in time domain. To be precise, the transition between two different successive pulses (such as -1 up to 1 or 1 down to -1) gets smoother instead of jumping between two discontinuous points. In frequency domain, the **LPF** output does not include high-frequency side lobes but keeps low-frequency main lobe which contains most of the energy and information as shown in Fig. 3.7. Hence, the **LPF** outputs $s_I(t)$ and $s_Q(t)$ hold the same information contents as the the **LPF** inputs $s_{NI}(t)$ and $s_{NQ}(t)$.

$$s_I(t) = s_{NI}(t) = \cos\phi(t) = \pm 1 \quad (3.22)$$

$$s_Q(t) = s_{NQ}(t) = \sin\phi(t) = 0 \quad (3.23)$$

After passing through the **LPF**, the in-phase baseband signal $s_I(t)$ and quadrature baseband signal $s_Q(t)$ are mixed separately with the in-phase and 90° -shifted quadrature carriers which are generated by the local oscillator (LO1). Mathematically, the mixer is treated as a multiplier, performing multiplication in time domain or convolution in frequency domain. The in-phase signal $s_I(t)$ is multiplied with the in-phase carrier $\cos 2\pi f_c t$ in time domain. In frequency domain, the multiplication turns into convolution operation and the mixed in-phase signal $s_I(t) \cdot \cos 2\pi f_c t$ becomes:

$$\begin{aligned} & \mathcal{F}\{s_I(t) \cdot \cos 2\pi f_c t\} \\ &= S_I(f) * \left[\frac{1}{2} \delta(f + f_c) + \frac{1}{2} \delta(f - f_c) \right] \\ &= \frac{1}{2} \cdot S_I(f + f_c) + \frac{1}{2} \cdot S_I(f - f_c) \end{aligned} \quad (3.24)$$

where the spectrum of the in-phase signal $s_I(t)$ is split into two parts: one centered at the carrier frequency f_c and the other centered at the negative carrier frequency $-f_c$. Figure 3.8 illustrates this up-conversion (mixing) process.

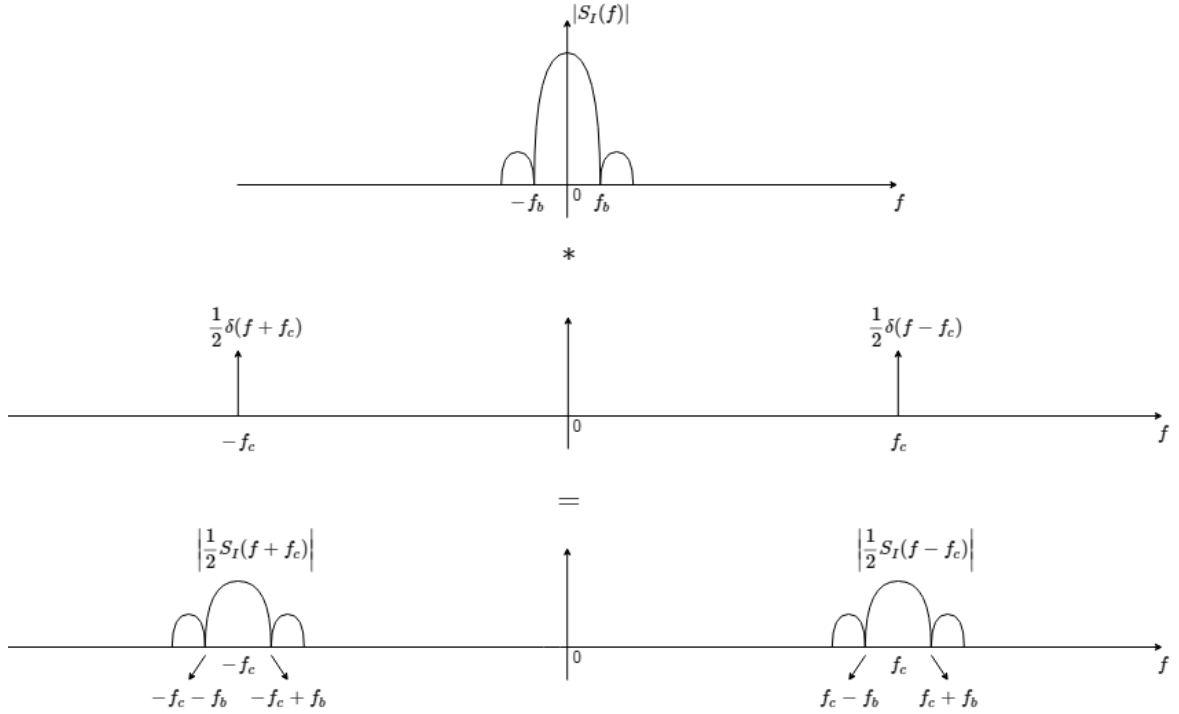


Figure 3.8: In-phase signal up-conversion

After mixing, the output RF signal $s_{IQ}(t)$ is the sum of the mixed in-phase and quadrature signals [43].

$$\begin{aligned}
 s_{IQ}(t) &= s_I(t) \cdot \cos 2\pi f_c t - s_Q(t) \cdot \sin 2\pi f_c t \\
 &= \cos \phi(t) \cdot \cos 2\pi f_c t - \sin \phi(t) \cdot \sin 2\pi f_c t \\
 &= \cos(2\pi f_c t + \phi(t))
 \end{aligned} \tag{3.25}$$

From (3.25), it can be seen that the output signal $s_{IQ}(t)$ is a cosine wave with carrier frequency f_c and binary phase $\phi(t)$. The information bit (0 or 1) determines the phase $\phi(t)$, which has two values 0 and π . With regard to the complete explanation and derivation for IQ modulation and demodulation, please refer to the appendix A.

In this chapter, the design of the receiver will be discussed. First of all, the overall structure of the receiver will be presented. Then, five important blocks in the receiver will be discussed separately. They are IQ demodulator, DC-offset compensator, double differential decoder, matched filter and sampler.

4.1 Receiver Structure Overview

In comparison with the transmitter, the receiver structure is far more complicated. The block diagram of the receiver is shown in Fig. 4.1. The receiver processes complex samples at baseband, which are the output of the IQ demodulator on SDR platform. In this diagram, the receiver is composed of several blocks: IQ demodulator, DC offset compensator, double differential decoder, matched filter and sampler.

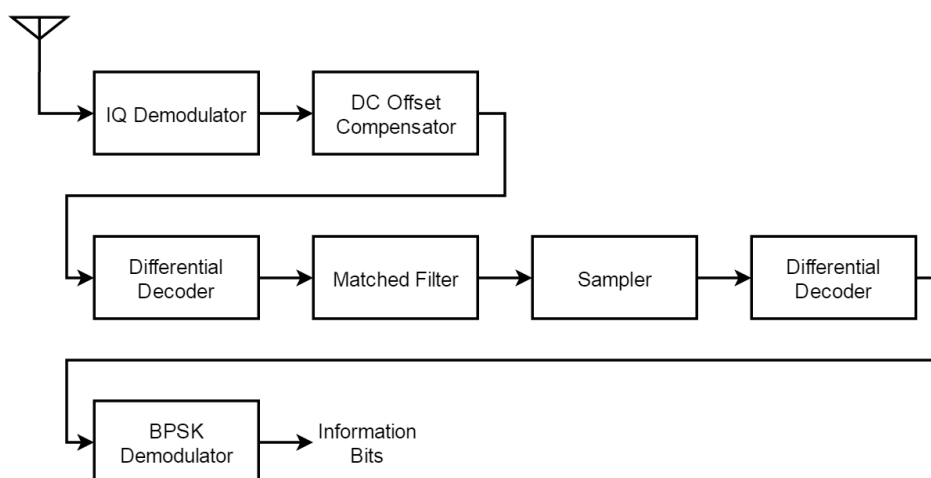


Figure 4.1: Receiver block diagram

The received complex samples pass through a series of functional blocks before being demodulated into information bits. The following part will briefly introduce the basic function of every block in the path:

- IQ demodulator: The IQ demodulator takes in analog RF signal and down-converts the signal from radio frequency to baseband. Then, the analog baseband signal is converted into digital baseband signal by the ADC. It outputs a stream of complex samples. Section 4.2 presents how IQ demodulator works in detail.
- DC offset compensator: The DC offset refers to the degree by which the mean value of a group of received samples deviates from zero. Let us take the BPSK signal as an example. If the absolute value of DC offset is so large that all of received complex samples fall on the negative-side or positive-side of constellation diagram, it will bring about incorrect symbol detection and wrong information bits. The

DC offset compensator estimates the mean value of a group of incoming samples and subtracts this mean value from every received sample to correct every sample so that the DC offset of the received samples stays close to zero. This block improves the accuracy of symbol detection. The details of DC offset compensation will be presented in section 4.3.

- First differential decoder: Since phase offset and frequency offset are introduced into received complex samples, the first differential decoder removes the already existing phase offset and meanwhile it converts frequency offset into constant phase offset. Section 4.4 provides the details on how it works.
- Matched filter: In the output of the first differential decoder, each symbol is represented by multiple samples. The matched filter, which is implemented by a moving average filter, integrates successive N incoming samples where N is the number of samples per symbol. It actually matches the pulse shape of the received symbol with the rectangular pulse shape of the transmitted symbol. Its outgoing sample sequence has triangle shapes where the peaky samples are extracted by the following sampler.
- Sampler: The sampler block looks for the peaky samples in the sequence output by the matched filter and samples them for the second differential decoder. The matched filter and the sampler together play a role of integration where all samples per symbol output by the first differential decoder are summed up and the sum result (one sample per symbol) is fed to the second differential decoder. The design details about the matched filter and the sampler are presented in section 4.5.
- Second differential decoder: The second differential decoder eliminates the remaining phase offset, which is a result of the frequency offset from the first differential decoder. Section 4.4 provides the detail explanations on how it works.

4.2 IQ Demodulator

4.2.1 IQ Demodulation

The IQ demodulator is implemented by the SDR platform. It down-converts the RF signal to baseband signal for further signal processing. Figure 4.2 exhibits the internal structure of the IQ demodulator. In this testbed, the IQ demodulation work is done in bladeRF SDR platform. In GNU radio, there exists a special block called ‘osmocom source’, which is an interface between the IQ demodulator in bladeRF and the signal processing blocks in GNU radio .

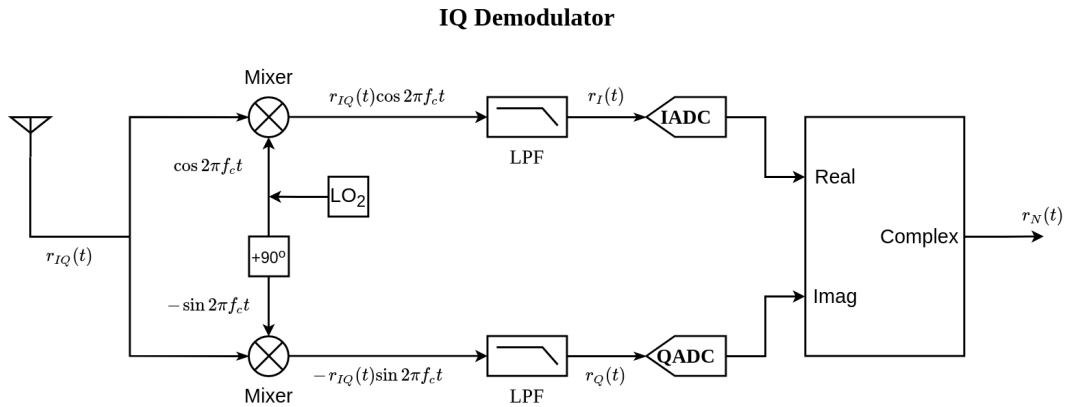


Figure 4.2: IQ demodulator structure

In order to simplify the explanation, it is assumed that the signal travels through an ideal channel without any noise so that the received RF signal $r_{IQ}(t)$ that is input to the IQ demodulator is undistorted and equal

to the RF signal $s_{IQ}(t)$ that is transmitted from the IQ modulator.

$$r_{IQ}(t) = s_{IQ}(t) = \cos(2\pi f_c t + \phi(t)) \quad (4.1)$$

The carrier frequency generated by the local oscillator (LO2) is tuned to be f_c , which is equal to the frequency of the received RF signal $r_{IQ}(t)$. Firstly, the received RF signal $r_{IQ}(t)$ is split into in-phase path and quadrature path respectively. The RF signal on the in-phase path is mixed with the in-phase carrier $\cos 2\pi f_c t$ while the RF signal on the quadrature path is mixed with the quadrature carrier $-\sin 2\pi f_c t$. After mixing, the mixed signals along the in-phase and quadrature paths are formulated as follows:

$$\begin{aligned} & r_{IQ}(t) \cdot \cos 2\pi f_c t \\ &= \cos(2\pi f_c t + \phi(t)) \cdot \cos 2\pi f_c t \\ &= \frac{1}{2} \cos \phi(t) + \frac{1}{2} \cos(2\pi \cdot 2f_c t + \phi(t)) \end{aligned} \quad (4.2)$$

$$\begin{aligned} & r_{IQ}(t) \cdot (-\sin 2\pi f_c t) \\ &= \cos(2\pi f_c t + \phi(t)) \cdot (-\sin 2\pi f_c t) \\ &= \frac{1}{2} \sin \phi(t) - \frac{1}{2} \sin(2\pi \cdot 2f_c t + \phi(t)) \end{aligned} \quad (4.3)$$

Both of the mixed signals contain high-frequency contents around $2f_c$. After the LPFs, the high-frequency components $\frac{1}{2} \cos(2\pi \cdot 2f_c t + \phi(t))$ and $\frac{1}{2} \sin(2\pi \cdot 2f_c t + \phi(t))$ are removed from the mixed signals. The LPF outputs along the in-phase and quadrature paths are $r_I(t)$ and $r_Q(t)$ respectively. Since the phase $\phi(t)$ takes the value of 0 or π , we get

$$r_I(t) = \frac{1}{2} \cos \phi(t) = \pm \frac{1}{2} \quad (4.4)$$

$$r_Q(t) = \frac{1}{2} \sin \phi(t) = 0 \quad (4.5)$$

The in-phase analog-to-digital converter (IADC) and quadrature analog-to-digital converter (QADC) convert the in-phase and quadrature signals from analog domain to digital domain for further signal processing. After the ADC, the in-phase signal $r_I(t)$ (real part) and quadrature signal $r_Q(t)$ (imaginary part) are combined into complex signal. The output of the IQ demodulator is

$$\begin{aligned} r_N(t) &= r_I(t) + jr_Q(t) \\ &= \frac{1}{2} (\cos \phi(t) + j \sin \phi(t)) \\ &= \frac{1}{2} e^{j\phi(t)} \end{aligned} \quad (4.6)$$

Now we can understand the IQ demodulation from the frequency-domain perspective. From section 3.5, it is known that the quadrature baseband signal $s_Q(t)$ from the IQ modulator is always zero while the in-phase baseband signal $s_I(t)$ from the IQ modulator is a sequence of NRZ pulses (+1 or -1). In frequency domain, the Fourier transform of $s_I(t)$, which is denoted by $S_I(f)$, is a sinc function. After passing through a zero-noise channel, the received RF signal $r_{IQ}(t)$ is equal to the transmitted RF signal $s_{IQ}(t)$.

$$\begin{aligned} r_{IQ}(t) &= s_{IQ}(t) \\ &= s_I(t) \cdot \cos 2\pi f_c t - s_Q(t) \cdot \sin 2\pi f_c t \\ &= s_I(t) \cdot \cos 2\pi f_c t \end{aligned} \quad (4.7)$$

The Fourier transform of the RF signal $r_{IQ}(t)$ is derived as follows:

$$\begin{aligned}
 R_{IQ}(f) &= \mathcal{F}\{r_{IQ}(t)\} \\
 &= \mathcal{F}\{s_I(t) \cdot \cos 2\pi f_c t\} \\
 &= \frac{1}{2} \cdot S_I(f + f_c) + \frac{1}{2} \cdot S_I(f - f_c)
 \end{aligned} \tag{4.8}$$

In the IQ demodulator, the received RF signal $r_{IQ}(t)$ is split into the in-phase and quadrature paths and then mixed with carriers $\cos 2\pi f_c t$ and $-\sin 2\pi f_c t$. The mixing process down-converts the RF signal $r_{IQ}(t)$ to the baseband and Figure 4.3 illustrates the down-conversion process performed by the in-phase mixer. It is noteworthy that the mixing process is a convolution (denoted by $*$) in frequency domain.

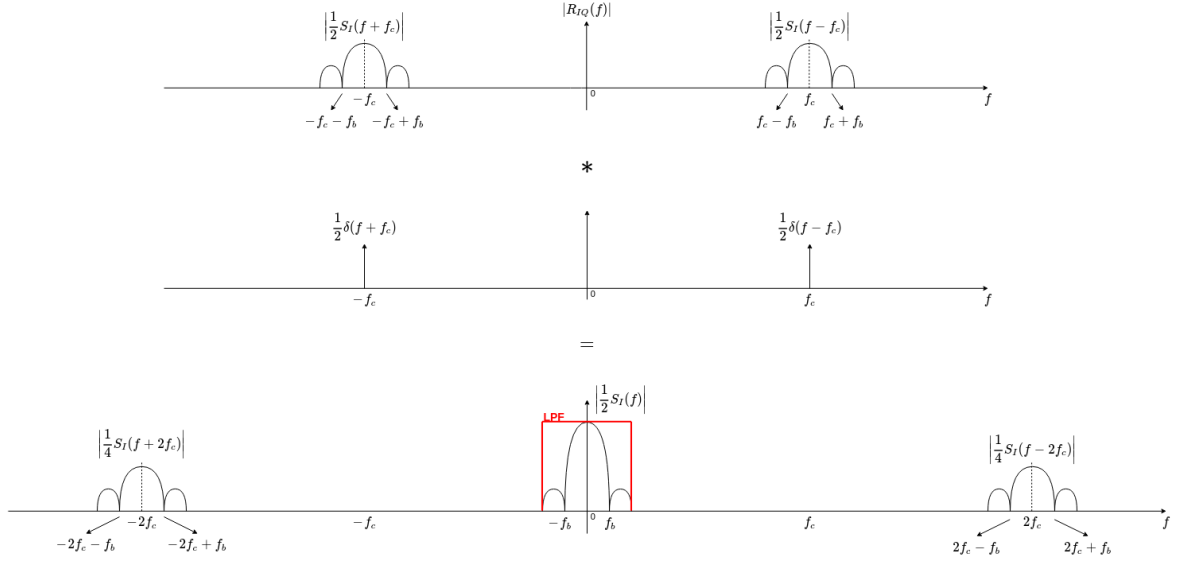


Figure 4.3: Received RF signal down-conversion

The output of the in-phase mixer $r_{IQ}(t) \cdot \cos 2\pi f_c t$ is expressed in frequency domain as follows:

$$\begin{aligned}
 &\mathcal{F}\{r_{IQ}(t) \cdot \cos 2\pi f_c t\} \\
 &= R_{IQ}(f) * \left[\frac{1}{2} \delta(f + f_c) + \frac{1}{2} \delta(f - f_c) \right] \\
 &= \frac{1}{2} \cdot R_{IQ}(f + f_c) + \frac{1}{2} \cdot R_{IQ}(f - f_c) \\
 &= \frac{1}{4} \cdot S_I(f + 2f_c) + \frac{1}{2} \cdot S_I(f) + \frac{1}{4} \cdot S_I(f - 2f_c)
 \end{aligned} \tag{4.9}$$

The output of the quadrature mixer $r_{IQ}(t) \cdot (-\sin 2\pi f_c t)$ is formulated in frequency domain as follows:

$$\begin{aligned}
 &\mathcal{F}\{r_{IQ}(t) \cdot (-\sin 2\pi f_c t)\} \\
 &= R_{IQ}(f) * \left[-\frac{j}{2} \delta(f + f_c) + \frac{j}{2} \delta(f - f_c) \right] \\
 &= -\frac{j}{2} \cdot R_{IQ}(f + f_c) + \frac{j}{2} \cdot R_{IQ}(f - f_c) \\
 &= -\frac{j}{4} \cdot S_I(f + 2f_c) - \frac{j}{4} \cdot S_I(f) + \frac{j}{4} \cdot S_I(f) + \frac{j}{4} \cdot S_I(f - 2f_c) \\
 &= -\frac{j}{4} \cdot S_I(f + 2f_c) + \frac{j}{4} \cdot S_I(f - 2f_c)
 \end{aligned} \tag{4.10}$$

After the mixers, the down-converted signals pass through the LPFs so that their high-frequency components $S_I(f + 2f_c)$ and $S_I(f - 2f_c)$ are removed [43] as shown in Fig. 4.4.

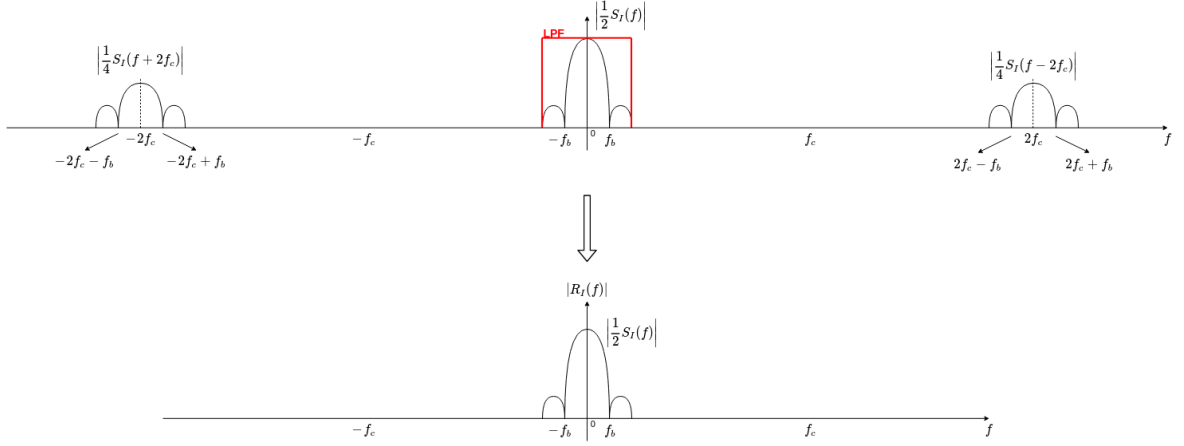


Figure 4.4: The down-converted in-phase signal passes through the LPF

The output of the in-phase LPF in frequency domain is

$$R_I(f) = \frac{1}{2} S_I(f) \quad (4.11)$$

The output of the quadrature LPF in frequency domain is

$$R_Q(f) = 0 \quad (4.12)$$

After the ADC, the in-phase and quadrature signals are combined into complex signal. The output of the IQ demodulator in frequency domain is

$$R_N(f) = R_I(f) + jR_Q(f) = \frac{1}{2} S_I(f) \quad (4.13)$$

where all information contents are contained in real part. With regard to the detailed explanation and derivation for IQ demodulation, please refer to the appendix A.

4.2.2 IQ Imbalance

There are two mainstream RF receivers: heterodyne receiver and homodyne receiver. The heterodyne receiver has at least two stages of mixers. The mixer at the first stage translates the received RF signal to intermediate frequency (IF) signal. Then, the down-converted IF signal is applied to IQ demodulator where the I and Q mixers at the second stage bring the down-converted IF signal to baseband signal or zero-IF signal. The superheterodyne receiver even has three stages of mixers. The first mixer brings the received RF signal to high IF signal and later the second mixer brings the high IF signal to low IF signal. In the end, the low IF signal is applied to IQ demodulator where the I and Q mixers at the third stage translate the low IF signal to baseband signal or zero-IF signal. The disadvantages of the heterodyne receiver are its complex structure and image frequency [44], [45].

The homodyne receiver, also known as zero-IF receiver or direct-conversion receiver, only needs one stage of mixers which are the I and Q mixers in IQ demodulator. The incoming RF signal is directly applied to the IQ demodulator which converts the RF signal to baseband signal or zero-IF signal. Differing from the heterodyne receiver that needs an IF mixer stage between the RF and baseband signals, the homodyne receiver does not have an IF mixer stage. Hence, the problem of the image frequency does not exist and an image rejection filter is not needed in the homodyne receiver [46], [47].

Nowadays, many SDR platforms including bladeRF SDR adopt the homodyne architecture because of its simplistic structure that only contains an IQ demodulator and its direct down-conversion to baseband. However, the homodyne receiver still suffers from a prominent drawback, IQ imbalance.

An ideal IQ demodulator should have perfectly symmetrical in-phase (I) and quadrature (Q) channels, where the amplification gain along the I branch is identical to the gain along the Q branch and the carrier input into I mixer is exactly 90 degrees out of phase compared to the carrier input into Q mixer. However, in reality, there is a mismatch between I and Q channels which leads to IQ imbalance. The IQ imbalance can be categorized into three main types: IQ DC offset, IQ amplitude imbalance and IQ phase imbalance [48]. It is necessary to compensate or correct IQ imbalance in order to decrease the symbol detection error at the receiver side.

The DC offset exists because of local oscillator (LO) leakage. It leads to the deviation of a received sample from its ideal position on the constellation diagram and furthermore results in degrading the symbol detection accuracy at the demodulator [49]. The compensation of DC offset will be discussed in section 4.3.

The IQ amplitude imbalance leads to phase offset because of amplitude variations in I and Q components of received samples. A carrier generated by the LO is input into I mixer while a copy of the same carrier delayed by 90 degrees in phase is input into Q mixer. In a real IQ demodulator, the actual delay is never precisely 90 degrees in phase. As a result, the IQ demodulator exhibits the phase imbalance between the I and Q branches, which also leads to phase offset. The elimination of the phase offset will be treated of in section 4.4.

4.3 DC Offset Compensation

In practice, DC offset exists in both I and Q branches. Nevertheless, the DC offset is undesirable because it causes the actual complex samples to deviate from their ideal positions as illustrated in Fig. 4.5. If the DC offset is large enough so that all of the received complex samples fall on either the left or right side of the constellation plane, the symbol detector will be unable to distinguish the received samples, resulting in a very high BER. In frequency domain, the DC offset is a spike in zero frequency as shown in Fig. 4.6.

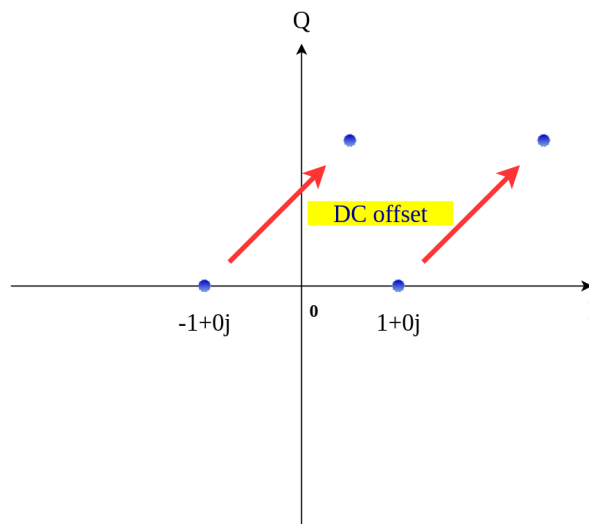


Figure 4.5: The effect of DC offset on constellation diagram

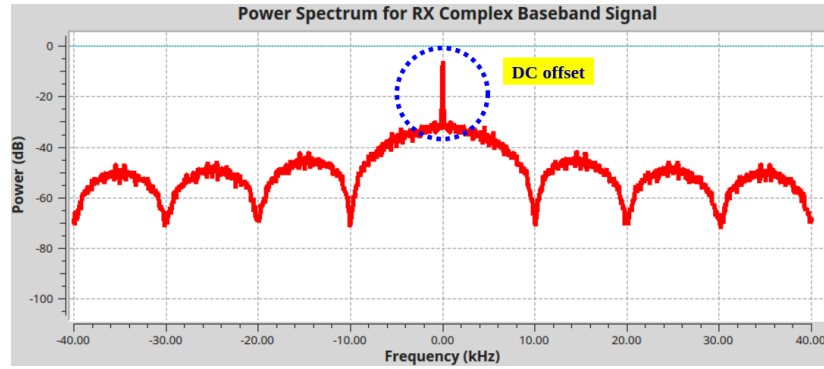


Figure 4.6: DC offset in baseband signal

The DC offset compensator in the receiver is to remove DC offset from samples by directly subtracting an estimated DC offset value from each sample. Mathematically speaking, the DC offset is the mean value of samples if the number of samples is really large. Figure 4.7 shows the internal structure of the DC offset compensator, where the moving average filter outputs the DC offset (the mean value of the latest 1000 samples) and the subtractor subtracts the latest DC offset from every incoming sample.

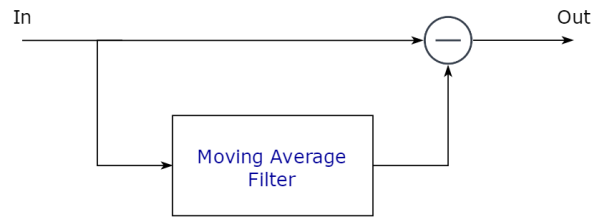
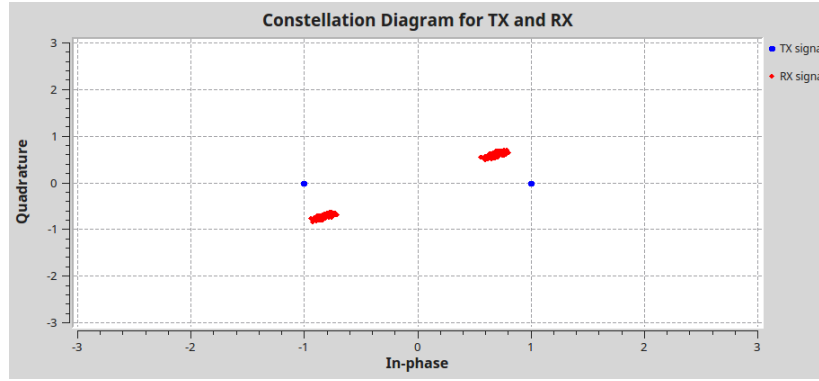


Figure 4.7: The structure of DC offset compensator

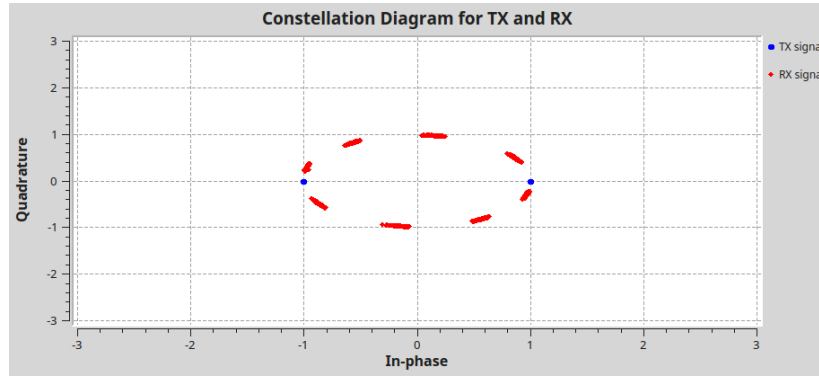
4.4 Double Differential Decoder Design

The double differential decoder is composed of two single differential decoders that are cascaded. Both decoders play a role in removing frequency offset. The first decoder removes the already existing phase offset while it converts the frequency offset to constant phase offset. The second decoder eliminates the remaining phase offset which is a result of the frequency offset from the first decoder.

The phase offset results from the unknown characteristics of communication channel and the IQ imbalance. Due to the phase offset, the positions of the received samples come closer to the imaginary axis (boundary) as shown in Fig. 4.8a. As a result, the error tolerance of the BPSK demodulator is degraded. The frequency offset arises from the mismatch between the frequency of the received RF signal and the frequency of the carrier produced by the LO. The received samples with frequency offset will rotate over time in the constellation diagram as shown in Fig. 4.8b, leading to incorrect symbol detection by the BPSK demodulator.



(a) Phase offset



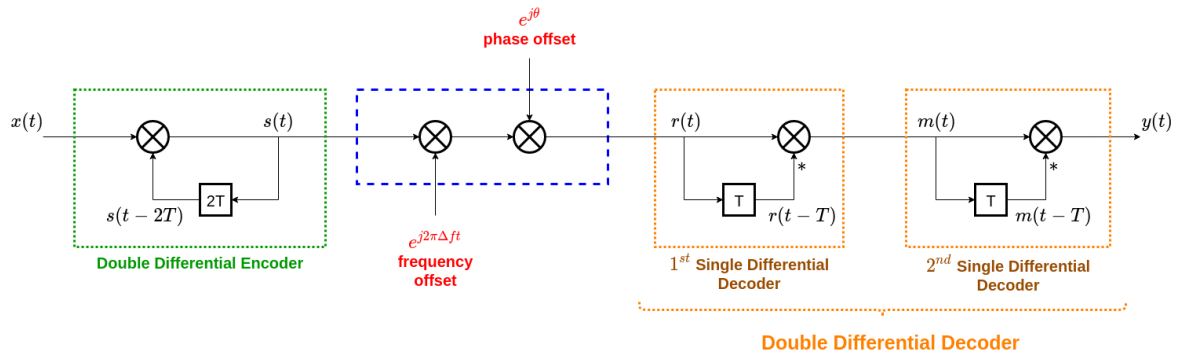
(b) Frequency Offset

Figure 4.8: The effects of phase offset and frequency offset

The structures of the double differential encoder and decoder are depicted in Fig. 4.9, where $x(t)$ refers to the complex samples that are input into the double differential encoder while $s(t)$ denotes the samples that are already double differential encoded. From section 3.3, it is known that:

$$s(t) = s(t - 2T) \cdot x(t) \quad (4.14)$$

where T is symbol period¹.

**Figure 4.9:** The structure of double differential decoder [41], [42]

In Fig. 4.9, $r(t)$ indicates the received complex samples which are expressed as follows:

¹symbol period = sample period in this derivation

$$r(t) = s(t) \cdot e^{j2\pi\Delta f t} \cdot e^{j\theta} \quad (4.15)$$

where Δf is frequency offset and θ is phase offset. The frequency offset term $e^{j2\pi\Delta f t}$ makes $r(t)$ to form a phase angle $2\pi\Delta f t$ that is varying over time so that the received samples always rotates over time in the constellation diagram. The phase offset term $e^{j\theta}$ causes $r(t)$ to form a constant phase angle θ in the constellation diagram.

The first decoder multiplies signal $r(t)$ during each symbol with the complex conjugate $r^*(t-T)$ of previous symbol. It is actually correlating each symbol with previous one. The intermediate signal $m(t)$ is:

$$\begin{aligned} m(t) &= r(t) \cdot r^*(t-T) \\ &= s(t) \cdot e^{j2\pi\Delta f t} \cdot e^{j\theta} \cdot s^*(t-T) \cdot e^{-j2\pi\Delta f (t-T)} \cdot e^{-j\theta} \\ &= s(t) \cdot s^*(t-T) \cdot e^{j2\pi\Delta f T} \end{aligned} \quad (4.16)$$

As we can see, the intermediate signal $m(t)$ does not contain the phase offset term $e^{j\theta}$ any more. That is to say, the first decoder eliminates the phase offset θ . It is also noteworthy that the first decoder converts the frequency offset term $e^{j2\pi\Delta f t}$ into a constant phase offset term $e^{j2\pi\Delta f T}$. The remaining phase offset $2\pi\Delta f T$ in the signal $m(t)$ is removed by the second decoder.

The output of the second decoder is denoted by $y(t)$:

$$\begin{aligned} y(t) &= m(t) \cdot m^*(t-T) \\ &= s(t) \cdot s^*(t-T) \cdot e^{j2\pi\Delta f T} \cdot s^*(t-T) \cdot s(t-2T) \cdot e^{-j2\pi\Delta f T} \\ &= s(t) \cdot s^*(t-T) \cdot s^*(t-T) \cdot s(t-2T) \\ &= s(t) \cdot s(t-2T) \\ &= s(t-2T) \cdot x(t) \cdot s(t-2T) \\ &= x(t) \end{aligned} \quad (4.17)$$

where $s^*(t-T) \cdot s^*(t-T) = 1$ and $s(t-2T) \cdot s(t-2T) = 1$ because $s(t)$ only takes 1 or -1 at any moment while $s(t) = s(t-2T) \cdot x(t)$ is from (4.14). The output $y(t)$ indicates that the frequency offset term $e^{j2\pi\Delta f t}$ is completely removed by the second decoder.

4.5 Matched Filter and Sampler Design

As shown in Fig. 4.10, both the matched filter and the sampler are located between two single differential decoders. They together behave as a simple integrator which outputs the sum of all samples per symbol. There are N samples per symbol after the first differential decoder. The sample sequence from the first differential decoder moves into the matched filter where successive N incoming samples are integrated and the sum is output. When a sample moves into the matched filter, a sample that is the sum of the last N samples is output. The sample sequence output by the matched filter has triangular shapes. The sampler block samples peaky samples from the triangle-shaped sequence to feed them for the second differential decoder. In practical RF transmission, samples may get lost, delayed or distorted. A simple integrator does not know the position of the first sample per symbol so that it can not sum up all samples per symbol correctly. Hence, the simple integrator is replaced by the matched filter and the sampler in order to output the sum of all samples per symbol correctly.

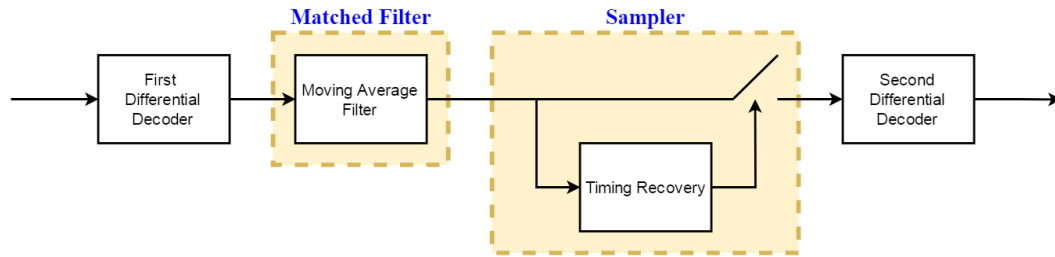


Figure 4.10: The position and structure of the matched filter and sampler

4.5.1 Repeater and Integrator

Figure 4.11 illustrates how the repeater and integrator process samples where there are four samples per symbol after the repetition.

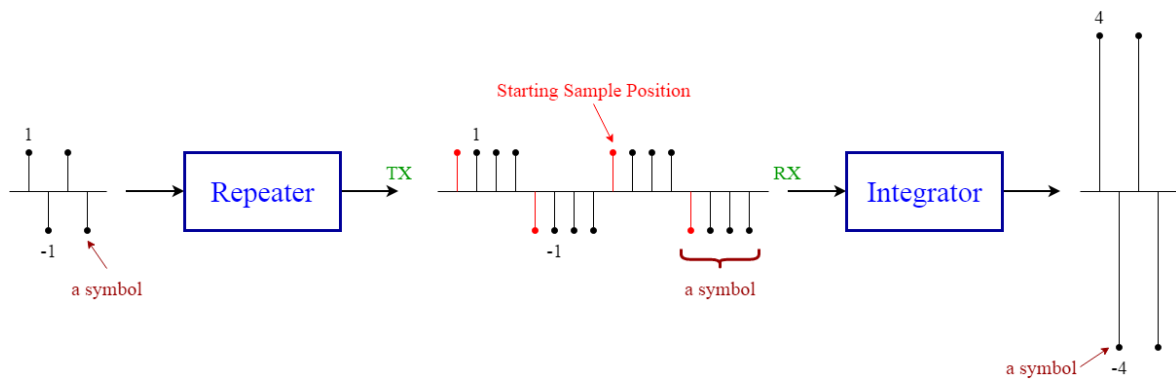


Figure 4.11: The repeater and integrator

At the transmitter side, a symbol is represented by one sample after being mapped from an information bit. Because of one sample per symbol, the symbol transitions are frequent and rapid, resulting in the baseband bandwidth of the symbols being large. For the purpose of VNB transmission, the bandwidth for a stream of symbols has to become narrow. In the testbed, the sample rate is determined. In order to achieve the narrow bandwidth, the symbols have to be repeated by the repeater so that each symbol is represented by multiple samples. After the repetition, a stream of symbols looks like a train of rectangular pulses. The duration of each rectangular pulse (each symbol) determines the symbol bandwidth. If more samples are repeated during each symbol, the duration of each rectangular pulse extends and the symbol bandwidth gets narrower.

At the receiver side, a received symbol contains multiple samples which are usually superimposed by additive white gaussian noise (AWGN). Because the white Gaussian noise is zero-mean, the more noisy samples during a symbol are integrated, the more noise in the symbol is averaged out. In other word, when all samples per symbol are added up, the useful signal is strengthened while the noise is attenuated. Thus, the integrator behaves like a filter which outputs symbols with high SNR. The symbols with high SNR can greatly improve the accuracy of symbol detection done by the BPSK demodulator. The output of the integrator is a stream of symbols with one sample per symbol as shown in Fig. 4.11.

In order to integrate all samples per symbol correctly, the integrator first has to decide the position of the beginning sample per symbol and then sums up all samples per symbol. In practice, these tasks are accomplished by the matched filter and the sampler.

4.5.2 Matched Filter

The symbols output by the repeater at the transmitter have multiple samples per symbol. Here, we assume that there are N samples per symbol. After the repetition, each symbol has a rectangular pulse shape. In order to correlate the received symbols with the transmitted symbols, the receiver employs a matched filter whose time-response is matching with the rectangular pulse shape of the transmitted symbols. The moving average filter plays a role of matched filtering. It is actually a simple low-pass finite-impulse response (FIR) filter that can remove unwanted high-frequency noise from intended samples. Because there are N samples per symbol before the matched filtering, the length of the moving average filter is equal to N . That is to say, there are N taps and $N - 1$ registers in the filter where successive N samples are input and the sum of them is output. The rectangular pulse shape of a transmitted symbol with N samples per symbol is defined as:

$$h[n] = \begin{cases} 1 & n = 0, 1 \cdots N - 1 \\ 0 & \text{otherwise} \end{cases} \quad (4.18)$$

The matched filter's response $h_m[n]$ is a time-reversed version of the rectangular pulse of the transmitted symbol:

$$h_m[n] = \begin{cases} h[N - 1 - n] & n = 0, 1 \cdots N - 1 \\ 0 & \text{otherwise} \end{cases} \quad (4.19)$$

The received samples are denoted by $r[n]$. If the received samples $r[n]$ pass through the matched filter $h_m[n]$, the filter's output $y[n]$ is the convolution of the received samples $r[n]$ and the matched filter's response $h_m[n]$ that is written as:

$$\begin{aligned} y[n] &= (r * h_m)[n] \\ &= \sum_{k=0}^{+\infty} r[k] \cdot h_m[n - k] \\ &= \sum_{k=0}^{+\infty} r[k] \cdot h[N - 1 - n + k] \\ &= \sum_{k=\max\{0, n-(N-1)\}}^n r[k] \cdot h[k - n + N - 1] \\ &= \sum_{k=\max\{0, n-(N-1)\}}^n r[k] \end{aligned} \quad (4.20)$$

The matched filter has $N - 1$ registers. When all $N - 1$ registers are filled up by the received samples $r[n]$, the filter's output is the sum of the latest N samples that move into the matched filter [50], [51].

$$y[n] = \sum_{k=n-(N-1)}^n r[k] \quad n \geq N - 1 \quad (4.21)$$

We take $N = 4$ as an example. The moving average filter with 4 taps has the structure as shown in Fig. 4.12 where

$$y[n] = \sum_{k=n-3}^n r[k] = r[n - 3] + r[n - 2] + r[n - 1] + r[n] \quad (4.22)$$

purpose.

4.5.3 Sampler

The sampler samples the output of the matched filter at the instants when peaky samples appear so that the SNR performance of the receiver can be maximized. Sampling the output of the matched filter at moments other than the ideal sampling instants will significantly degrade the effective SNR in symbols. The sampling offset Δ is defined as the difference between the real sampling instant and the ideal sampling instant. The higher the SNR in symbols is, the more accurate the symbol detection is. Therefore, it is important to keep the sampling offset Δ as close to zero as possible in order to acquire symbols with the highest SNR.

In order to estimate the optimum sampling points, a timing recovery method called Gardner algorithm [52] is employed. The Gardner algorithm is chosen because only two samples per symbol are required and its implementation is relatively simple. We assume $x[n]$ to be the input of the sampler. The Gardner timing error $e[n]$ at n -th sampling point during a symbol period N (the number of samples per symbol) is figured out using the following equation [53], [54]:

$$e[n] = (x[n + N] - x[n]) \cdot x[n + \frac{N}{2}] \quad (4.23)$$

The Gardner timing recovery algorithm uses three samples in two adjacent symbols. It is noteworthy that the Gardner timing error is relatively small or even close to zero when two neighboring symbols have the same polarity. Hence, it is better to use a stream of symbols with frequently alternate signs so that the timing recovery block in the sampler can quickly estimate correct sampling points. Figure 4.14 illustrates the different sampling points where the symbol period $N = 4$.

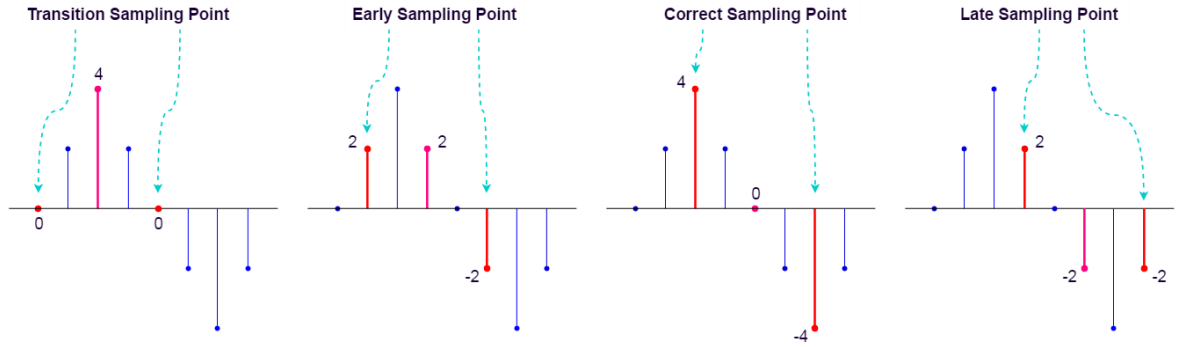


Figure 4.14: The sampling point at different positions

Since there are 4 samples per symbol, there are 4 sampling points in a symbol.

- Transition sampling point $n = 0$: $e[0] = (x[4] - x[0]) \cdot x[2] = (0 - 0) \cdot 4 = 0$.
- Early sampling point $n = 1$: $e[1] = (x[5] - x[1]) \cdot x[3] = ((-2) - 2) \cdot 2 = -8 < 0$.
- Correct sampling point $n = 2$: $e[2] = (x[6] - x[2]) \cdot x[4] = ((-4) - 4) \cdot 0 = 0$.
- Late sampling point $n = 3$: $e[3] = (x[7] - x[3]) \cdot x[5] = ((-2) - 2) \cdot (-2) = 8 > 0$.

When n continues to increase, the timing error $e[n]$ will repeat. Once the Gardner timing error is computed, the value and polarity of the error can be used to decide whether the sampling point is correct.

1. $e[n] < 0$: sampling point is early
2. $e[n] > 0$: sampling point is late

3. $e[n] = 0$:

- $e[n - 1] < 0$ (early) and $e[n + 1] > 0$ (late): sampling point is correct
- $e[n - 1] > 0$ (late) and $e[n + 1] < 0$ (early): sampling point is at the transitional position from current symbol to next symbol.

Due to signal distortion and noise, each sample in the output sequence of the matched filter deviates more or less from its ideal value, which results in non-zero Gardner timing errors at correct sampling points. The timing recovery block in the sampler compares the absolute value of $e[n]$ at sampling point n with the absolute values of $e[n - 1]$ at previous point and $e[n + 1]$ at next point respectively. If the absolute value of $e[n]$ is less than the absolute values of both $e[n - 1]$ and $e[n + 1]$, then the polarities of both adjacent errors $e[n - 1]$ and $e[n + 1]$ are probed. If the previous error $e[n - 1] < 0$ (early) and the next error $e[n + 1] > 0$ (late), the sampling point that the current timing error $e[n]$ corresponds to is the correct sampling point.

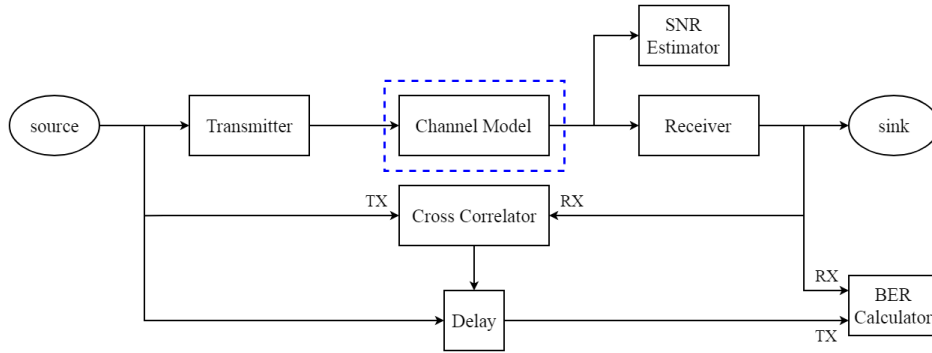
In brief, based on the Gardner algorithm, the sampler decides correct sampling points and extracts all samples at these points from the output sequence of the matched filter. The output of the sampler is a stream of symbols with the highest SNR and one sample per symbol.

TESTBED DESIGN AND EXPERIMENT

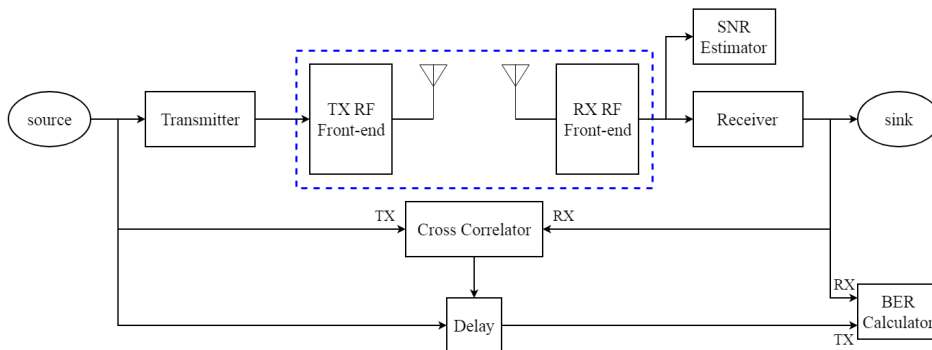
This chapter will start with presenting the overall structure of the communication testbed. Then, the GNU-radio implementations of main blocks such as transmitter and receiver, channel model and RF interfaces will be discussed. The remaining part of the chapter will contribute to the discussion about two measurement blocks, SNR estimator and BER calculator.

5.1 Testbed Design

The goal of this project is to design a VNB communication testbed where a successful communication link is built between the transmitter and the receiver. The communication testbed can be simulated by software on GNU Radio SDR development environment as shown in Fig. 5.1a, where all blocks including the channel model are implemented using software. Furthermore, the testbed can also run on an SDR platform as shown in Fig. 5.1b, where a real wireless communication link is built. In this case, only the channel model is replaced by RF front-ends in the testbed structure while other blocks remain unchanged.



(a) Testbed for software simulation



(b) Testbed for RF transmission

Figure 5.1: Testbed structure

Figure 5.1 reveals the structure inside the VNB communication testbed. The following part will provide a brief introduction to the function of each block in the testbed structure.

- **Transmitter:** The transmitter does mapping, encoding and repetition (or rectangular-pulse shaping). At the end, it outputs complex samples.
- **Channel model:** A channel model is simulated by software, where AWGN, phase offset and frequency offset are superimposed on complex samples that are passing through the channel model. The channel model is used in simulating the testbed via software.
- **RF front-end:** In a real RF communication experiment, the channel model in the testbed is replaced by RF front-ends while other blocks remain the same. The TX RF front-end up-converts the complex samples from transmitter into RF signal to transmit and the RX RF front-end down-converts the received RF signal to complex samples for receiver. A pair of RF interfaces connect the RF front-ends on SDR platform with the blocks implemented by GNU Radio software.
- **Receiver:** The receiver performs a series of tasks such as DC-offset compensation, integration, decoding and symbol detection.
- **SNR estimator:** The SNR estimator estimates the SNR over a stream of complex samples directly out of RX RF front-end in real-time.
- **Cross correlator:** The cross correlator performs cross-correlation estimation over a stream of received bits and a stream of transmitted bits, figuring out the delay between the RX bit stream and the TX bit stream in real-time. The delay time estimated by the cross correlator is fed to delay block.
- **Delay:** Because of communication channel and a series of processing blocks, the RX bit stream gets in the BER estimator block later than the TX bit stream. Therefore, the delay block is used to delay the TX bit stream so that both TX and RX bit streams arrive at the BER calculator at the same time.
- **BER calculator:** The BER calculator compares the TX bit stream with the RX bit stream to estimate the error ratio in the received bit stream in real time. In order for correct BER calculation, both the TX and RX bit streams have to arrive at the block at the same time.

All main blocks in the testbed will be discussed in subsequent sections. Before discussing the functional blocks in the testbed, we should make clear some important concepts in GNU radio such as data types, sample rate and symbol rate.

5.1.1 GNU Radio Data Types

In GNU radio, all data in flowgraph are expressed in discrete samples. There are three frequently used data types for samples: integer-8, float-32 and complex float-32. In this GNU-radio based testbed design, only these three types are used.

- **Integer-8:** It represents an integer sample with 8 bits in width. This 8-bit integer sample can represent either unpacked byte or packed byte. In an unpacked byte, only the least significant bit (LSB) carries information while the rest of the bits are zero. In a packed byte, all 8 bits carry information. In other word, an integer sample that represents an unpacked byte carries only 1 information bit on the LSB while an integer sample that represents a packed byte carries 8 available information bits.
- **Float-32:** It represents a floating-point real sample with the width of 32 bits.
- **Complex Float-32:** It represents a complex sample which is made up of real part and imaginary part. Both the real and imaginary parts are 32-bit floating-point real value so that a complex sample actually occupies 64 bits in total.

5.1.2 Sample Rate vs. Symbol Rate

- Sample rate

The sample rate (or sample frequency) f_{sample} refers to the number of samples which are allowed to pass through the GNU radio flowgraph per second. In GNU-radio software simulation, the maximum sample rate is determined by a throttle block in flowgraph. Without a throttle block in flowgraph, the software simulation for the flowgraph will run on processor as quick as possible and a deadlock may happen if the processor is overloaded by tremendous amount of data. When the testbed is running on the SDR platform such as bladeRF, the maximum sample rate in GNU-radio flowgraph is determined by ADC/DAC sample rate on the SDR platform.

We take the bladeRF SDR platform as an example. The bladeRF platform is equipped with the ADC/DAC with sample rate from 0.16~40 MHz. Both the ADC and DAC have two paths: I path and Q path. The ADC or DAC performs the data conversion on I and Q paths alternately. As a result, the sample rate on each path is the half of the A/D or D/A sample rate, ranging from 0.08~20 MHz. The sample rate in the whole GNU-radio flowgraph refers to the number of complex samples that pass through ADC/DAC on an SDR platform per second. In practice, very low data rate is required in the VNB communication testbed. Hence, it is enough to set the lowest value 0.16 MHz for both the ADC and DAC sample rates on bladeRF SDR platform. In this testbed, both the ADC and DAC sample rates are set at 160 kHz. As a result, the sample rate on the testbed flowgraph is $f_{sample} = 80kHz$.

- Symbol rate

The symbol rate f_{symbol} indicates the number of information symbols which are transferred through the communication channel per second. Each symbol includes a number of samples which are complex float-32 type.

In GNU-radio flowgraph, the symbol rate is determined by both the sample rate and sps (the number of samples per symbol). The minimum sample rate in bladeRF platform is constant and equal to 80kHz. So, the symbol rate can be set using different values of sps . The relationship among symbol rate f_{symbol} , sample rate f_{sample} and sps is:

$$f_{symbol} = \frac{f_{sample}}{sps} \quad (5.1)$$

The symbol rate f_{symbol} is dependent upon the bit rate R_b which the communication system intends to achieve. The relation between the symbol rate f_{symbol} and the bit rate R_b is formulated in the following equation:

$$f_{symbol} = \frac{R_b}{N} \quad (5.2)$$

where N refers to the number of bits conveyed by a symbol. As BPSK is used, $N = 1$ and $f_{symbol} = R_b$. For example, if the instantaneous bit rate R_b on the testbed is 10 kbps, the symbol rate $f_{symbol} = 10$ kHz. Since the sample rate f_{sample} is fixed at 80 kHz by ADC/DAC, $sps = \frac{f_{sample}}{f_{symbol}} = \frac{80}{10} = 8$. The sps is a parameter which is assigned to the repeater at the transmitter and the integrator at the receiver. It also determines the bandwidth of the symbol stream that is going into IQ modulator or coming out of IQ demodulator. If the sample rate f_{sample} is fixed and sps is increasing, a symbol will contain more samples, leading to lower symbol rate f_{symbol} and narrower bandwidth of symbol stream.

The received RF signal is down-converted to baseband signal. Then, the baseband signal is converted to digital samples by the ADC with sample rate f_{sample} . After the ADC, the spectrum of the digital baseband signal will appear at frequencies which are the multiple of the sample rate f_{sample} . In order to prevent aliasing in analog-to-digital conversion shown in Fig. 5.2, the sample rate f_{sample} should be at least two times larger

than the symbol rate f_{symbol} , that is, $f_{sample} \geq 2f_{symbol}$. As a consequence, $sps = \frac{f_{sample}}{f_{symbol}} \geq 2$. In GNU-radio flowgraph, the sps can not be smaller than 2 after the ADC. Otherwise, the received symbols will be severely distorted.

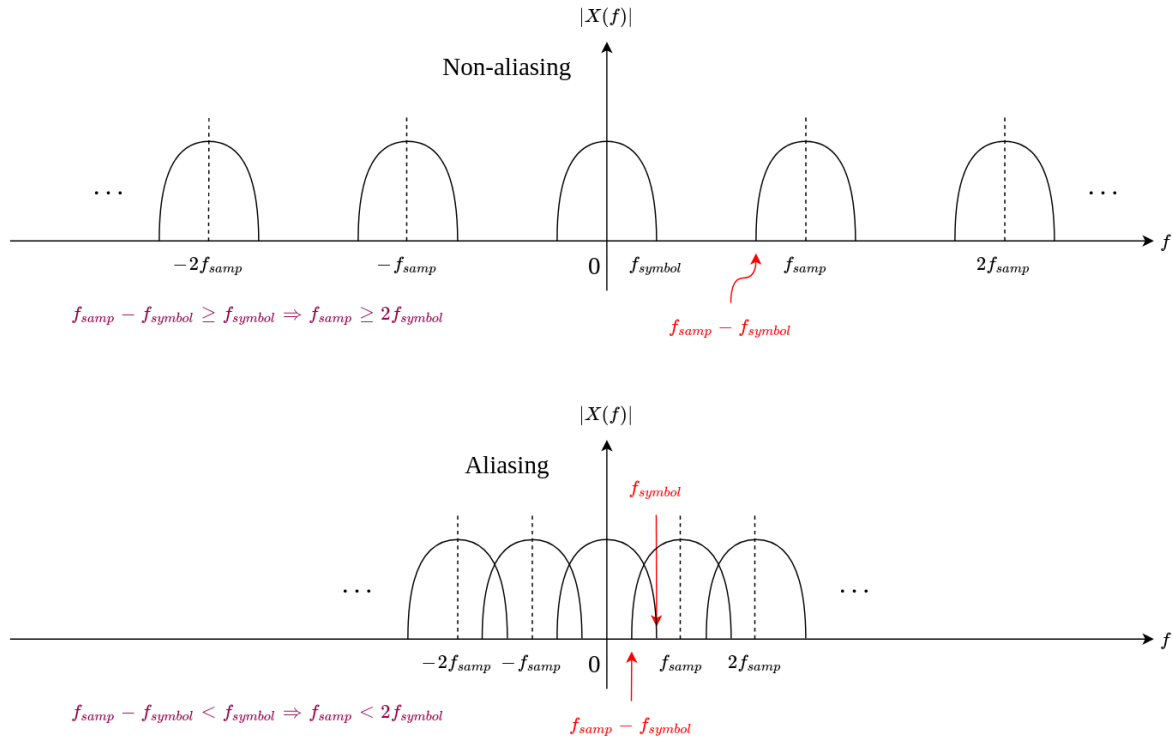


Figure 5.2: Non-aliasing and aliasing

5.1.3 Transmitter Implementation

Figure 5.3 illustrates the GNU-radio flowgraph for transmitter implementation. The description of the blocks in the transmitter flowgraph are as follows:

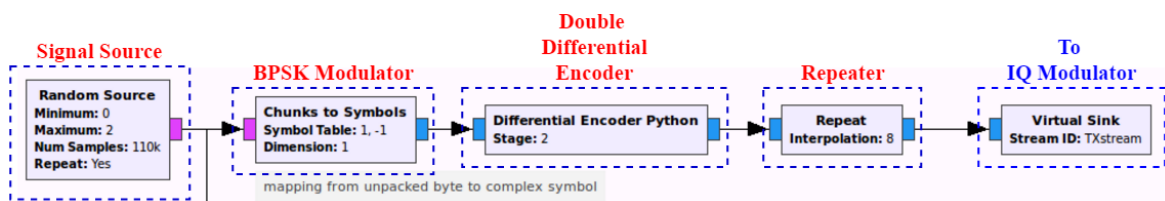


Figure 5.3: The GNU radio flowgraph for transmitter

- BPSK modulator: The chunks to symbols block behaves like a BPSK modulator, performing a mapping from binary bits to BPSK symbols (1 and -1). Its input samples are 8-bit unpacked bytes with each byte conveying only one bit. The samples going out of this block are complex samples with the data type of complex float-32, where each complex sample represents a symbol.
- Double differential encoder: The differential encoder python block implements the double differential encoder. This customized block is implemented by python code. Both the input and output of the block are complex samples.
- Repeater: The repeat block implements the repeater. The interpolation factor in the repeat block is

equal to sps (the number of samples per symbol). For example, if the interpolation factor is 8, the input-to-output ratio is 1 to 8. That is, when one complex sample is input, eight repeated complex samples are output.

5.1.4 Receiver Implementation

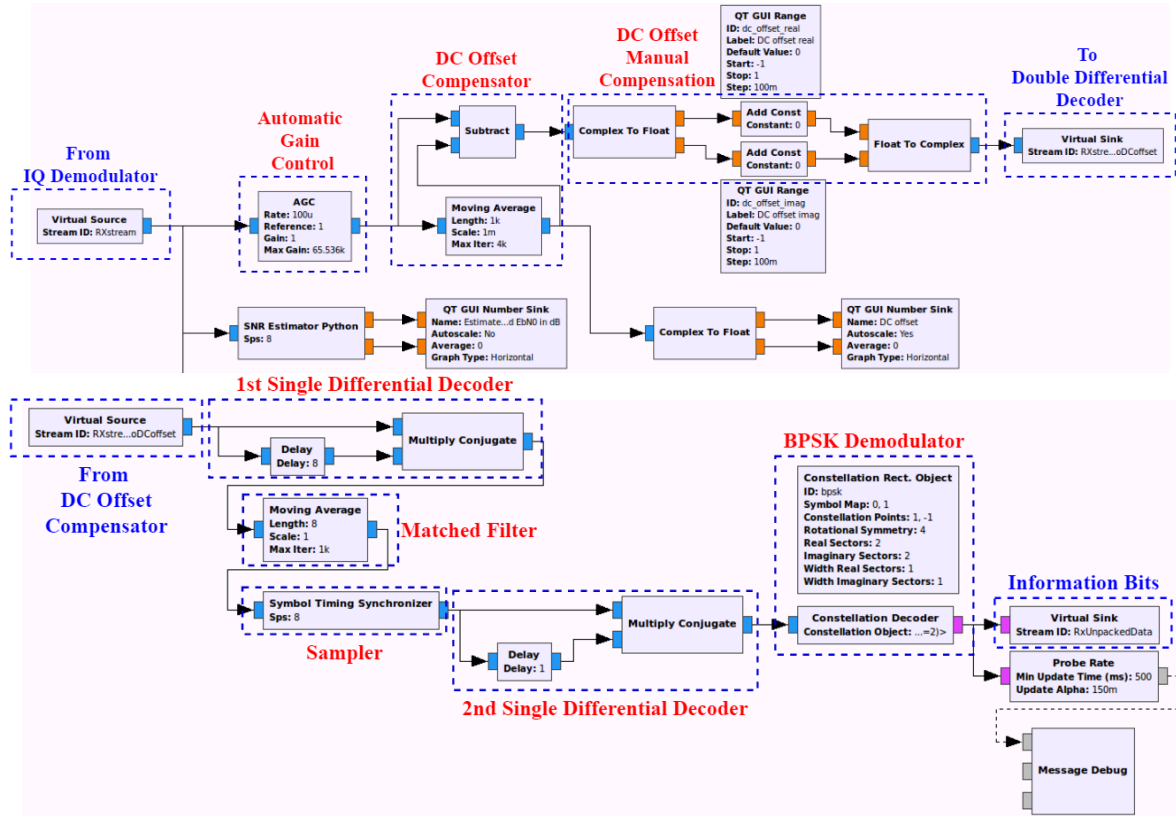


Figure 5.4: The GNU radio flowgraph for receiver

Figure 5.4 illustrates the GNU-radio flowgraph for receiver implementation. The description of the blocks in the receiver flowgraph are as follows:

- Automatic gain control: The AGC (automatic gain control) block scales up the weak magnitude of received complex samples to a suitable level.
- DC-offset compensator: The moving average block and subtract block constitute the DC-offset compensator to automatically cancel DC-offsets in complex samples. An additional DC-offset manual compensation is implemented after that.
- Single differential decoder: Both single differential decoders have the same structure. The difference is that the delay in the first decoder is equal to sps while the delay in the second decoder is equal to 1 because the first decoder decodes incoming symbols with sps samples per symbol while the second decoder decodes incoming symbols with one sample per symbol.
- Matched filter: The matched filter is implemented by a moving average filter where the received symbols are correlated with the pulse shape of the transmitted symbols. It outputs a triangle-shaped sequence where each peaky sample is the sum of all samples per symbol.

- **Sampler:** The sampler is implemented by the symbol timing synchronizer block where the peaky samples in the sequence output by the matched filter are sampled and fed to the second decoder. There are sps samples per symbol before the sampler and only one sample per symbol after the sampler. In addition, the symbol timing synchronizer block is a customized block implemented by python code.
- **BPSK demodulator:** The constellation decoder detects incoming complex symbols with one sample per symbol and outputs corresponding bits according to the demodulation scheme that is determined by the constellation object, where a mapping rule from complex symbols to bits are specified.
- **SNR estimator python:** The SNR estimator takes in complex samples that are directly coming out of the RX RF front-end and makes an SNR estimation over a sequence of incoming complex samples. The customized block is implemented by python code. The algorithm behind the SNR estimator is discussed in subsection 5.2.3.

5.1.5 Channel Model for Software Simulation

The channel model is placed between the transmitter and the receiver and used for simulating a communication channel on the testbed by software. Figure 5.5 shows the composition of the channel model.

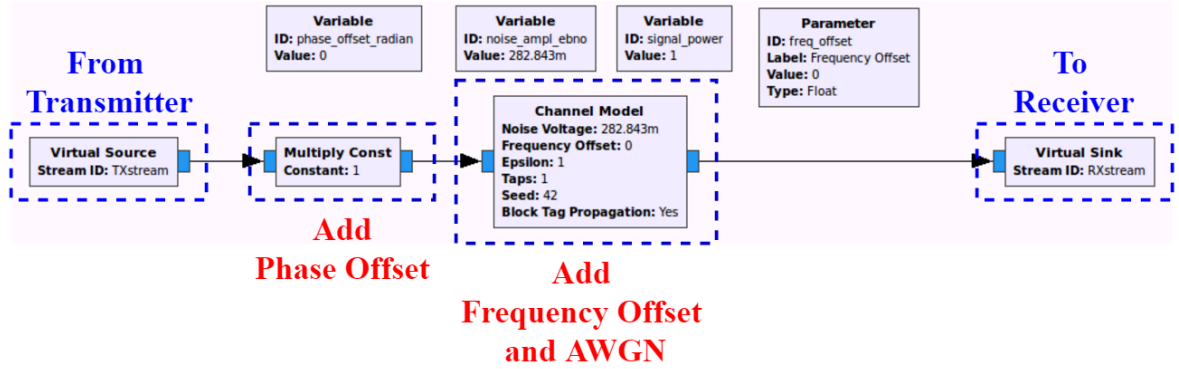


Figure 5.5: The channel model

When complex samples are passing through the channel model, phase offset, frequency offset and noise can be added into them.

- The phase offset can be added into a complex sample via multiplying the complex sample with a complex constant $e^{j\phi}$, where ϕ is the phase offset in radian.
- The frequency offset can be added into a complex sample by the channel model block, where the frequency offset is set based on a value normalized to the sample rate f_{sample} . A zero means no offset while a 0.25 indicates one quarter of the sample rate and 1 is equivalent to the sample rate. The normalized frequency offset that is assigned to the channel model is denoted by Δf_{norm} while the frequency offset is defined as Δf . Their relation is expressed as follows:

$$\Delta f_{norm} = \frac{\Delta f}{f_{sample}} \quad (5.3)$$

- The white Gaussian noise can be added into a complex sample by the channel model block, where the noise level is set by noise amplitude as follows:

$$A_{noise} = \sqrt{\frac{P_{signal} \cdot sps}{10^{\frac{E_b/N_o}{10}}}} \quad (5.4)$$

where E_b/N_o denotes SNR per bit in dB. The above formula can be derived from equations (5.5) and (5.6).

5.1.6 RF Interface for Hardware Experiment

The RF interfaces are used in the testbed flowgraph to implement real RF signal transmission between the transmitter and the receiver by means of SDR platform. Figure 5.6 shows the RF interfaces to the bladeRF SDR platforms that are employed in the testbed.

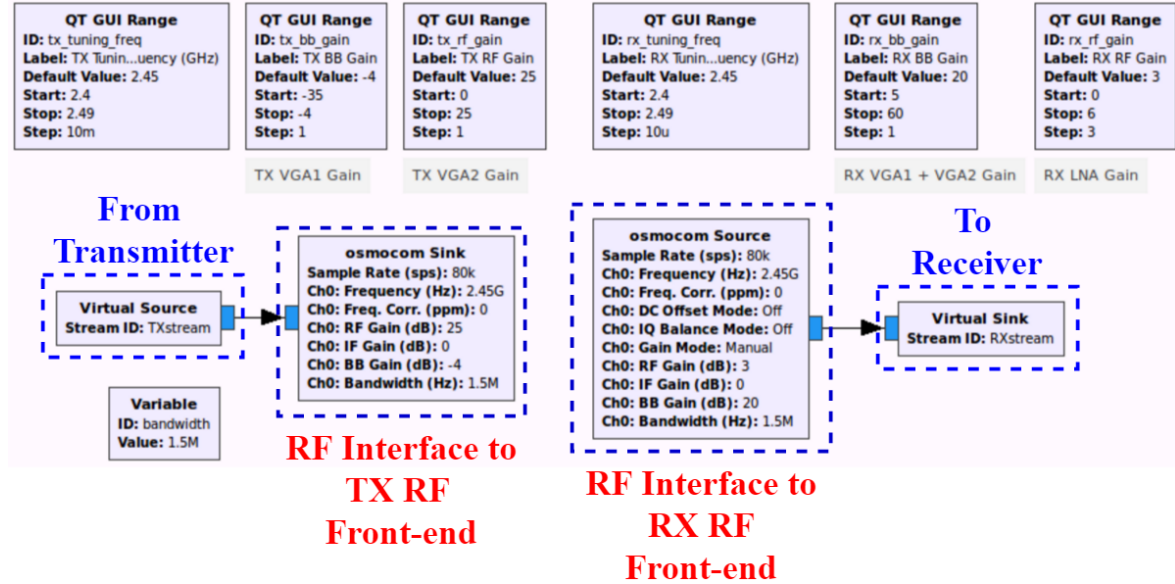


Figure 5.6: The RF interface to bladeRF SDR platform

There are two kinds of RF interfaces:

- osmocomb sink: The osmocomb sink is an RF interface to the TX RF front-end on the bladeRF SDR platform. Its incoming complex samples are directly from the transmitter block in GNU radio software environment. After entering into the osmocomb sink block, the complex samples are up-converted to RF signal by the TX RF front-end for wireless transmission.
- osmocomb source: The osmocomb source functions as an RF interface to the RX RF front-end on the bladeRF SDR platform. After received RF signal is down-converted to complex samples by the RX RF front-end, the osmocomb source outputs the complex samples, which are directly input into the receiver block in GNU radio software environment.

Additionally, the parameters for the bladeRF SDR platform such ADC/DAC sample rate, RF tuning frequency, bandwidth, amplification gain can be configured by modifying the attributes in both the osmocomb sink and osmocomb source blocks.

5.2 SNR and BER

The SNR and BER are two key factors that need to be measured on the testbed for evaluating the communication performance.

5.2.1 SNR and Eb/No

The SNR denoted by $\frac{S}{N}$ is defined as a scalar ratio of signal average power (denoted by S or P_s) over noise average power (denoted by N or P_n):

$$\frac{S}{N} = \frac{P_s}{P_n} = \frac{A_s^2}{A_n^2} \quad (5.5)$$

where A_s and A_n are signal amplitude and noise amplitude. The SNR per bit denoted by $\frac{E_b}{N_o}$ is another way to express the SNR and is more common in measuring SNR for a digital communication system. Herein, $\frac{E_b}{N_o}$ is defined as a scalar ratio of energy per bit E_b to noise power spectral density N_o ¹. The relation between $\frac{S}{N}$ and $\frac{E_b}{N_o}$ is as follows:

$$\frac{S}{N} = \frac{E_b}{N_o} \cdot \frac{R_b}{BW} = \frac{E_b}{N_o} \cdot \frac{f_{symbol}}{f_{sample}} = \frac{E_b}{N_o} \cdot \frac{1}{sps} \quad (5.6)$$

$$\frac{E_b}{N_o} = \frac{S}{N} \cdot sps \quad (5.7)$$

where R_b is bit rate and BW is the noise bandwidth of the receiver. $R_b = f_{symbol}$ because every symbol conveys one bit after BPSK modulation. On the other hand, every digital sample from the ADC is superimposed by noise and furthermore the ADC sample rate determines the noise bandwidth in digital domain. Therefore, $BW = f_{sample}$.

5.2.2 BER vs. Eb/No

The BER is defined as a ratio of the number of incorrectly received bits to the number of total transmitted bits and is usually taken as a measurement of the communication quality. The theoretical BER for a normal BPSK-modulated signal in an AWGN channel is:

$$BER = Q\left(\sqrt{\frac{2E_b}{N_o}}\right) = \frac{1}{2}erfc\left(\sqrt{\frac{E_b}{N_o}}\right) \quad (5.8)$$

where $Q(x)$ is the Q-function and $erfc(x)$ is the complementary error function. The complementary error function $erfc(x)$ is defined as

$$erfc(x) = \frac{2}{\sqrt{\pi}} \int_x^\infty e^{-t^2} dt \quad x \geq 0 \quad (5.9)$$

When $\frac{E_b}{N_o} = 0$, the theoretical BER for a simple BPSK modulation becomes:

$$BER = \frac{1}{2}erfc(0) = \frac{1}{\sqrt{\pi}} \int_0^\infty e^{-t^2} dt = \frac{1}{2} \quad (5.10)$$

The theoretical BER for a single differentially encoded BPSK signal is as follows:

$$BER = \frac{1}{2}e^{-\frac{E_b}{N_o}} \quad (5.11)$$

5.2.3 SNR Estimator

The SNR is directly measured at the output of the osmocomb source (the RF interface to the RX RF front-end on the bladeRF SDR platform) by the SNR estimator. The SNR estimator receives complex samples and stores them in a buffer with a fixed length. Once the buffer is filled up, the SNR estimator will evaluate SNR over a sequence of the received complex samples in the buffer. In the end, both SNR and Eb/No in dB are output by the SNR estimator and the buffer is cleared for next SNR estimation. This process repeats until the testbed stops running. The buffer in the SNR estimator can store N complex samples and the i -th complex sample in the buffer is denoted by x_i , whose magnitude is $|x_i|$.

¹The dimension of noise power spectral density N_o is watts/Hz

The SNR estimation method used in the SNR estimator is the combination of two algorithms, simple algorithm and second and fourth order moment (M2M4) algorithm. To estimate SNR using the simple algorithm [55], [56], the received signal average power P_s and the received noise average power P_n have to be figured out at first. The signal average power P_s is equal to the arithmetic mean μ of the squared magnitudes of the received samples x_i as formulated below:

$$P_s = \mu = \frac{1}{N} \sum_{i=0}^{N-1} (|x_i|)^2 \quad (5.12)$$

The noise average power P_n is equal to the variance σ_n^2 of the magnitudes of the received samples x_i as shown below:

$$P_n = \sigma_n^2 = \frac{1}{N} \sum_{i=0}^{N-1} (|x_i| - |\bar{x}|)^2 \quad (5.13)$$

where $|\bar{x}|$ is the mean of the received samples' magnitudes. Once the signal power P_s and the noise power P_n are figured out, SNR can be estimated by:

$$\frac{S}{N} = \frac{P_s}{P_n} = \frac{\mu}{\sigma_n^2} \quad (5.14)$$

The other algorithm used to estimate SNR is M2M4 [57]–[59]. First, the second moment $M2$ and the fourth moment $M4$ need to be calculated according to:

$$M2 = \frac{1}{N} \sum_{i=0}^{N-1} (|x_i|)^2 \quad (5.15)$$

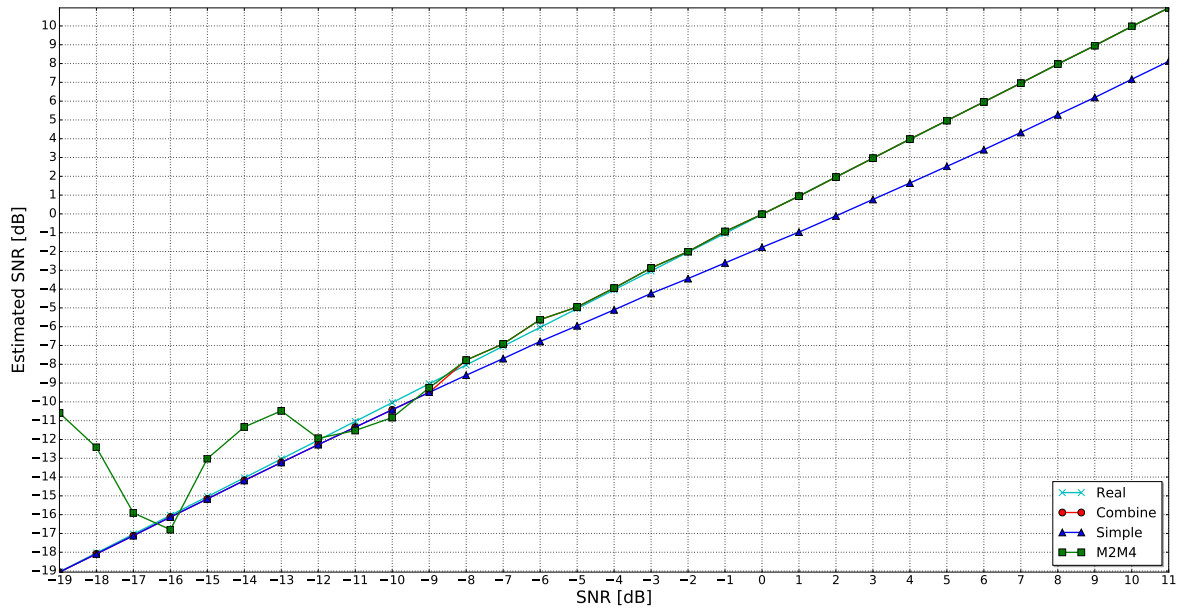
$$M4 = \frac{1}{N} \sum_{i=0}^{N-1} (|x_i|)^4 \quad (5.16)$$

Then, SNR is estimated by the formula:

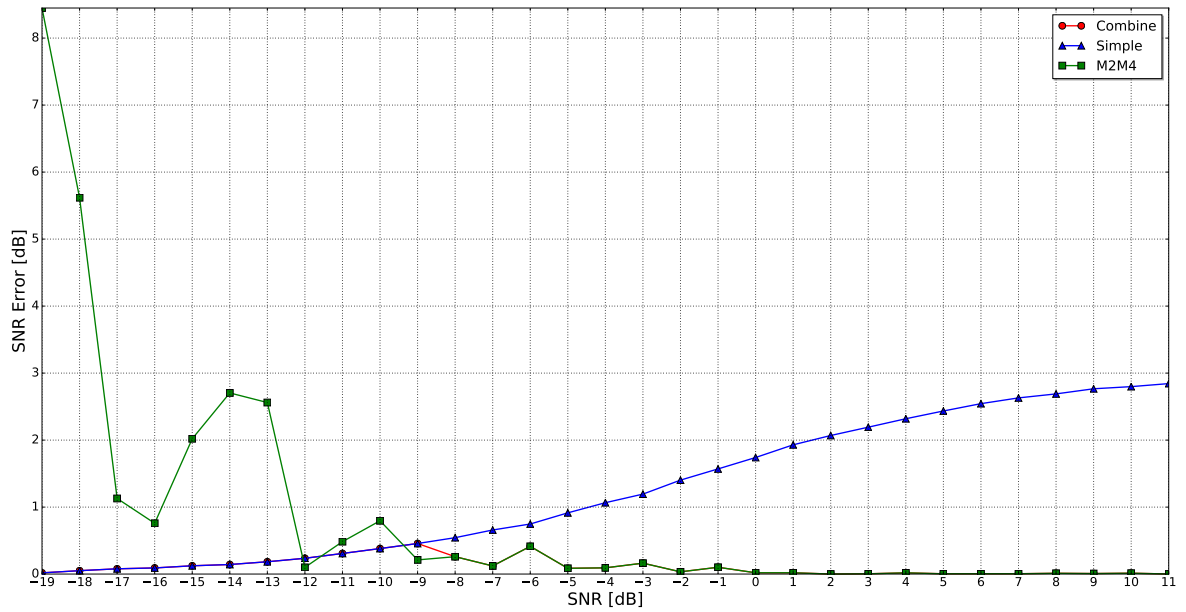
$$\frac{S}{N} = \frac{\sqrt{2 \cdot M2 \cdot M2 - M4}}{M2 - \sqrt{2 \cdot M2 \cdot M2 - M4}} \quad (5.17)$$

The SNR estimator employs the M2M4 method at first because SNR result evaluated by the M2M4 is more accurate than the simple method when actual SNR is large. If the SNR value estimated by the M2M4 method is smaller than -8 dB, the simple method will be adopted by the SNR estimator because the SNR calculated by the simple method is more precise than the M2M4.

The SNR estimator block with buffer size = 10000 is tested by software simulation where AWGN is added in the channel model and the noise level is controlled by a specified SNR. Figure 5.7a shows estimated SNR by simple algorithm, M2M4 algorithm and the combination of them as well as real SNR applied to the AWGN channel. Figure 5.7b displays the absolute errors between the estimated SNR values by different algorithms and the actual SNR on the AWGN channel model. It is clearly shown in Fig. 5.7b that SNR estimation error from the M2M4 method is smaller than the simple method when the actual SNR is larger than -8 dB. However, the SNR estimation error from the M2M4 method fluctuates drastically as the actual SNR decreases less than -8 dB. It is also noteworthy that the SNR estimation error from the simple method becomes more and more close to zero and almost has no fluctuation as the actual SNR decreases below -8 dB. That is the reason why observed value -8 dB is used as a threshold for the SNR estimator to switch between the simple method and the M2M4 method.



(a) Estimated SNR and real SNR



(b) Error between estimated SNR and real SNR

Figure 5.7: The comparison of SNR estimation algorithms

5.2.4 BER Calculator

The bit error ratio (BER) refers to a ratio of the number of received error bits divided by the total number of transmitted bits, which is dimensionless and varying from 0 to 1. Figure 5.8 displays the BER calculator in GNU-radio flowgraph. The BER calculator takes in **TX** bits from the signal source and **RX** bits from the receiver's output. Every incoming bit is actually a sample with the data type of unpacked byte. Every pair of incoming TX and RX bits passes through an exclusive-OR (**XOR**) operation block at first. If the TX and RX bits are identical, the XOR result is zero. Otherwise, the XOR result is one. A buffer inside the BER calculator stores the XOR results and the buffer size determines the BER accuracy. Every time when an XOR result is pushed into the buffer, the BER calculator block will sum up all XOR results in the buffer and divide the sum by the buffer size, outputting the updated BER.

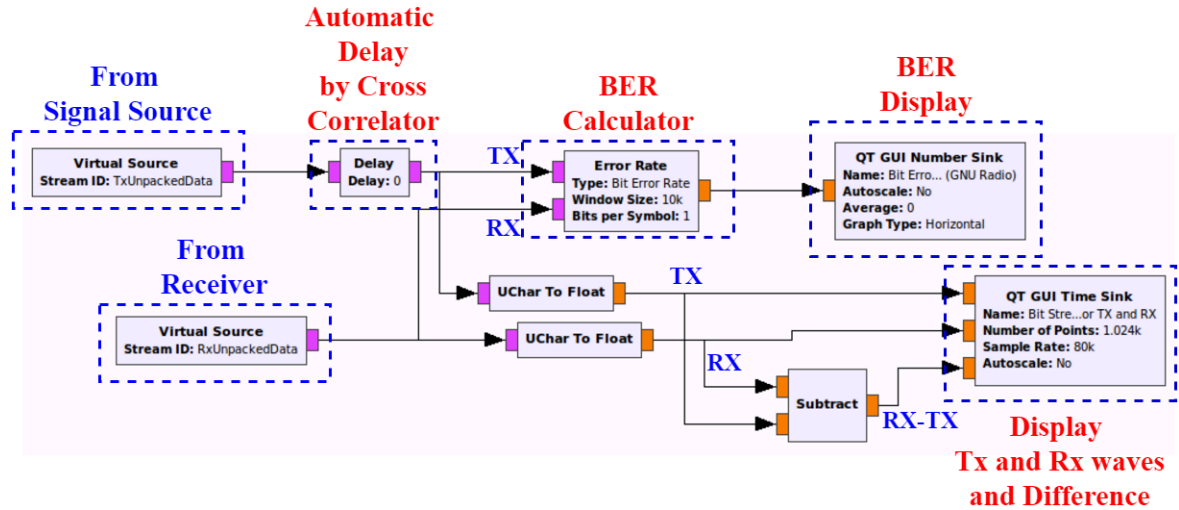


Figure 5.8: The GNU radio flowgraph for BER calculator

In software simulation, both TX and RX sequences arrive at the BER calculator at the same time because there is no time delay in the blocks that are used. However, in practical RF transmission, the received signal has to be passed through SDR platform and transmitted over the air. There will be definitely a delay between the RX sequence from the receiver and the TX sequence directly from the signal source. Thus, the TX and RX sequences can not be compared directly by the BER calculator.

In practice, the RX sequence arrives at the BER calculator later than the TX sequence. In order to output correct BER, the TX sequence has to be postponed so that both the TX and RX sequences can arrive at the BER calculator at the same time. Therefore, a delay block is appended along the TX path before the BER calculator as shown in Fig. 5.8 so that the TX and RX sequences are synchronized.

The delay is affected by many unknown factors that exist in SDR platform and RF channel, and it is always varying with time. Therefore, real-time delay estimation is needed to implement automatically-adjusted delay on the TX sequence before the BER calculator. The cross correlator discussed in the following subsection is responsible for the real-time delay estimation.

5.2.5 Cross Correlator

The cross correlator calculates the delay between the incoming TX and RX sequences. Figure 5.9 reveals the cross correlator in the GNU radio flowgraph.

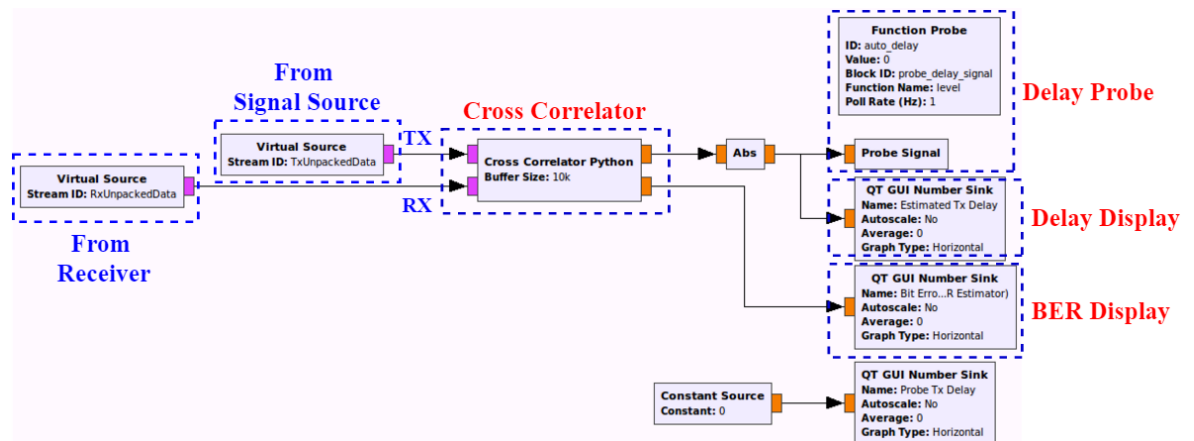


Figure 5.9: The GNU radio flowgraph for cross correlator

The cross correlator takes in TX bits directly from the signal source and RX bits from the receiver. Every incoming bit is actually a sample of unpacked byte. There are two buffers within the cross correlator: TX buffer stores the incoming TX bits while RX buffer holds the incoming RX bits. Both of the buffers have the same size. If the buffer size is small (< 1000), the estimated delay is inaccurate. If it is large (> 10000), the calculation time becomes too long. In this case, the buffer size is specified to be 10000. Once both the TX and RX buffers are filled up, the TX and RX sequences in the buffers will be extracted and the cross-correlation operation will be done over the two sequences. After the cross-correlation calculation, delay will be output. In the cross correlator, as an extra function, BER is also calculated over the TX and RX sequences after the TX sequence is aligned with the RX sequence by the delay. In practice, the output delay is negative because it means that the TX sequence is earlier than the RX sequence in time. However, in order to postpone the TX sequence by the delay block before the BER calculator, only positive delay value is needed. Therefore, an absolute block is appended after the cross-correlator output. The probe signal block in Fig. 5.9 detects the updated delay and feeds the delay back to the delay block in Fig. 5.8.

The cross correlation is used to find where two sequences match and determine the time delay between them. Herein, we concisely discuss the cross-correlation operation used in the cross correlator. For discrete signals, the cross-correlation function $c[n]$ between TX sequence $s[n]$ and RX sequence $r[n]$ [60] is defined as:

$$\begin{aligned} c[n] &= s[n] \otimes r[n] \\ &= \sum_{k=-\infty}^{\infty} s^*[k-n] \cdot r[k] \end{aligned} \quad (5.18)$$

where $s^*[k]$ is the complex conjugate of $s[k]$ and n is the delay. The TX sequence $s[n]$ is the original sequence and the RX sequence $r[n]$ contains the TX sequence $s[n]$ with an unknown delay and added noise. The peak of $c[n]$ appears at the lag time n where $s[k-n]$ matches $r[k]$ at best. In the cross correlator, the TX sequence $s[n]$ and RX sequence $r[n]$ are stored in the TX and RX buffers respectively. The size of the buffers is fixed and denoted by N . In other word, both the TX and RX sequences have the same length N . The python function `numpy.correlate` with full mode is used to calculate cross-correlation results. Actually, returned by the `numpy.correlate` function is a sequence of cross-correlation results with the length of $2N - 1$. The cross-correlation function $c[n]$ in full mode is formulated as follows:

$$\begin{aligned} c[n] &= s[n] \otimes r[n] \\ &= \sum_{k=-(N-1)}^{N-1} s^*[k-n] \cdot r[k] \end{aligned} \quad (5.19)$$

where the lag n varies from the minimum lag $-(N-1)$ to the maximum lag $N-1$. The lag corresponding to the maximum of $c[n]$ is the delay between the TX and RX sequences. If the lag n is negative, the TX sequence $s[n]$ leads the RX sequence $r[n]$. If the lag n is positive, the TX sequence $s[n]$ lags behind the RX sequence $r[n]$. In practical case, the lag n is always negative.

5.3 BER Performance

The performance of the VNB communication testbed is evaluated by its BER performance. In this section, the BER performances in both software simulation and hardware experiment are presented. The accuracy of the BER values is determined by the buffer size in the BER calculator. Although larger buffer size in the BER calculator leads to more accurate BER result, it also causes longer calculation time. Hence, the buffer size of

10^6 is chosen for measuring BER performance after the trade-off between the accuracy and the calculation time is considered.

5.3.1 Software Simulation for BER versus E_b/N_o

The performance of a digital communication system is measured in BER versus SNR curve or BER versus E_b/N_o curve. In this project, the BER versus E_b/N_o curve is used in measuring the testbed performance when software simulation is conducted. The BER versus E_b/N_o curves for BPSK, SDBPSK and DDBPSK schemes are compared in Fig. 5.10 when the testbed is simulated at bit rate $R_b = 10\text{kbps}$ and sample rate $f_{\text{sample}} = 80\text{kHz}$ and meanwhile no phase and frequency offset is added in the channel model. The theoretical BPSK curve is directly derived from equation (5.8) while the theoretical SDBPSK curve directly stems from equation (5.11). The simulated BPSK curve comes from the testbed which only uses the normal BPSK modulation and demodulation. The simulated SDBPSK is the curve when the single differential encoder and decoder are used in the testbed while the simulated DDBPSK is the one when the double differential encoder and decoder are used.

As we can see among the three schemes (BPSK, SDBPSK and DDBPSK) in Fig. 5.10, the normal BPSK scheme without any coding has the best BER performance and the DDBPSK has the worst one while the SDBPSK is in between. Because the normal BPSK scheme does not multiply noisy samples, its BER performance is the best. In the SDBPSK scheme, the received noisy samples are multiplied once by the single differential decoder so that its BER performance is worse than the normal BPSK scheme. In the DDBPSK scheme, the received noisy samples are multiplied twice by the double differential decoder, resulting in the worst BER performance.

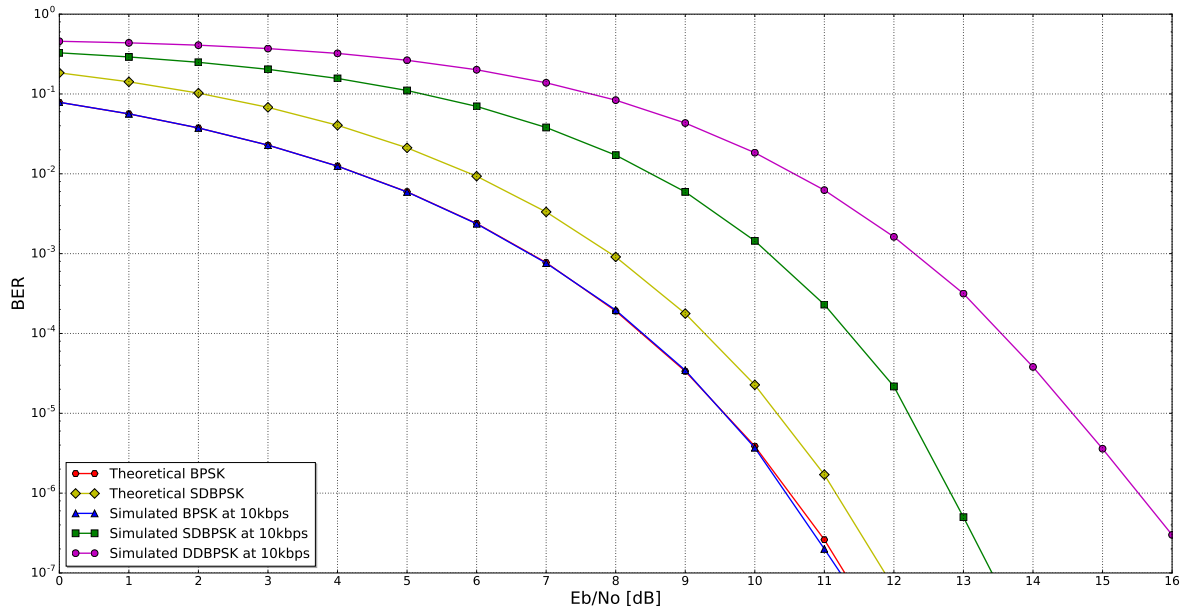


Figure 5.10: BER vs. E_b/N_o curves for BPSK, SDBPSK and DDBPSK when $R_b = 10\text{kbps}$ and $f_{\text{sample}} = 80\text{kHz}$

When the double differential BPSK scheme (DDBPSK) is used in the testbed, BER versus E_b/N_o curves are compared for the testbed that is simulated at different bit rates R_b in Fig. 5.11 where the sample rate f_{sample} is set to be 80kHz and there is no phase and frequency offset. As shown in Fig. 5.11, lower bit rate results in worse BER performance. Because $\frac{E_b}{N_o} = \frac{S}{N} \cdot \text{sps}$ and $\text{sps} = \frac{f_{\text{sample}}}{f_{\text{symbol}}} = \frac{f_{\text{sample}}}{R_b}$ from (5.6), it can be derived that when the sample rate ($f_{\text{sample}} = 80\text{kHz}$) is fixed, a lower bit rate R_b leads to a bigger sps (more samples per symbol) and meanwhile the bigger sps results in lower SNR and worse BER for the same $\frac{E_b}{N_o}$.

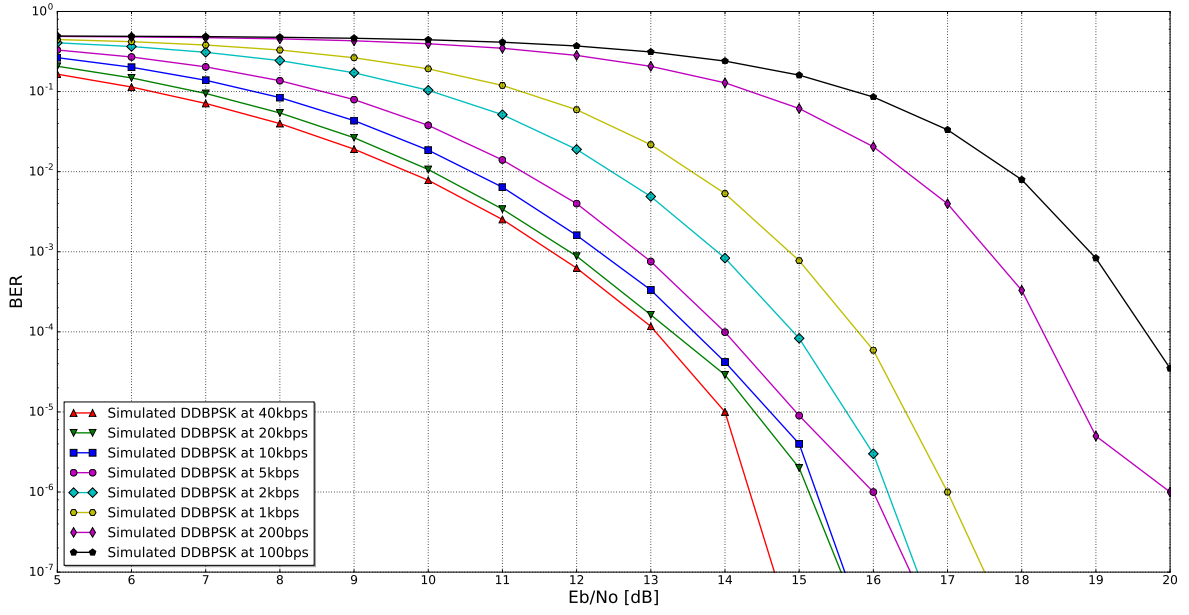


Figure 5.11: BER vs. E_b/N_0 curves for DDBPSK at different bit rates R_b when $f_{sample} = 80\text{kHz}$

Figure 5.12 compares BER versus SNR curves for the testbed that is simulated at different bit rates R_b while the sample rate f_{sample} is fixed to be 80kHz and there is no phase and frequency offset. The BER versus SNR curves change with the bit rate R_b . When the bit rate R_b becomes smaller, the BER performance gets better. In other word, lower bit rate results in the same BER with lower SNR.

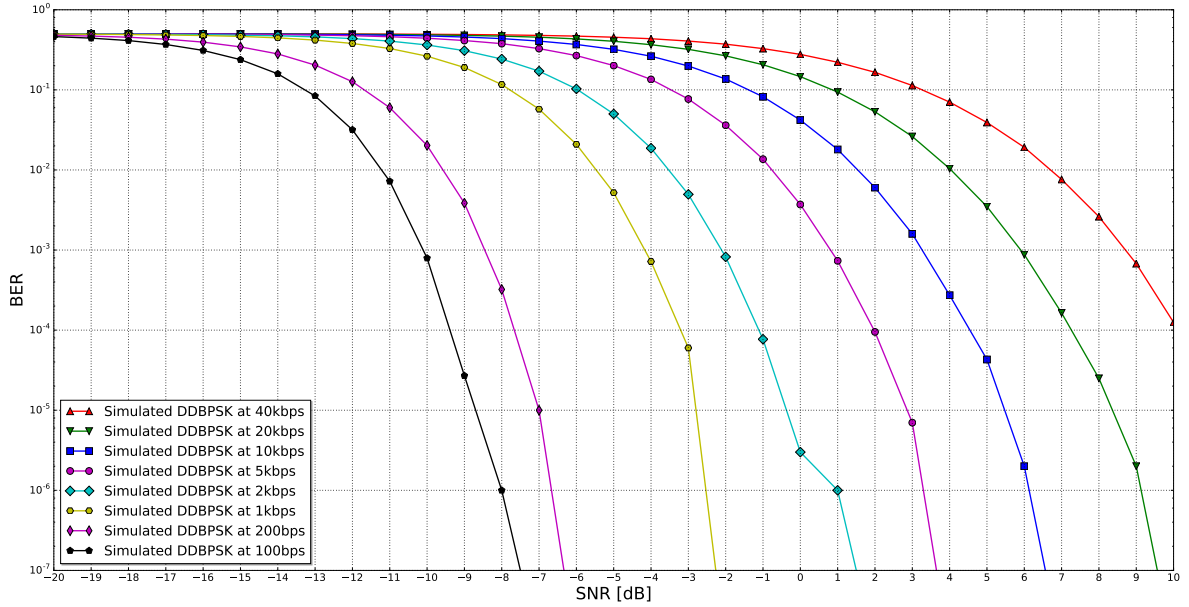


Figure 5.12: BER vs. SNR curves for DDBPSK at different bit rates R_b when $f_{sample} = 80\text{kHz}$

5.3.2 Software Simulation for BER versus Phase Offset

Figure 5.13 compares BER versus phase offset curves for three BPSK schemes (BPSK, SDBPSK and DDBPSK) where the phase offset varies from -180 to 180 degrees while other conditions are fixed such as $f_{sample} = 80\text{kHz}$, $R_b = 10\text{kbps}$, $E_b/N_0 = 10\text{dB}$ and frequency offset = 0.

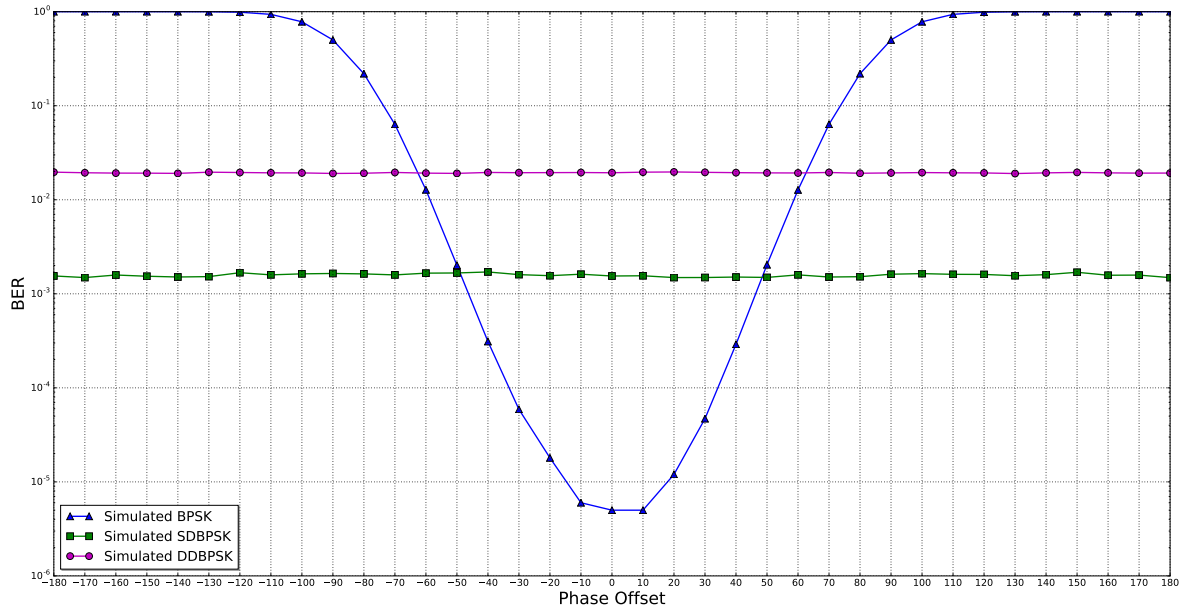


Figure 5.13: BER vs. phase offset for different BPSK schemes

As we can see, the normal BPSK scheme without differential coding (simulated BPSK) is not robust against the variation of the phase offset while other BPSK schemes with differential coding (simulated SDBPSK and simulated DDBPSK) have almost unchanged BER curves no matter how the phase offset is changed. It is also noteworthy that the BER in the simulated BPSK curve comes close to 1 (or 100%) when the phase offset goes beyond ± 90 degrees. Here, the 100% BER means that the detected bits are completely inverted. In brief, all BPSK schemes with differential coding have the ability to withstand the effect of the phase offset.

5.3.3 Software Simulation for BER versus Frequency Offset

Figure 5.14 compares BER versus frequency offset curves for three BPSK schemes (BPSK, SDBPSK and DDBPSK) where the frequency offset changes from -0.5 to 0.5 while other conditions are fixed such as $f_{sample} = 80\text{kHz}$, $R_b = 10\text{kbps}$, $E_b/N_0 = 15\text{dB}$ and phase offset = 0. The frequency offset varies from -0.5 to 0.5 because it refers to the normalized frequency offset Δf_{norm} where 0 means no frequency offset, 0.25 indicates one quarter of the sample rate and 0.5 indicates half of the sample rate.

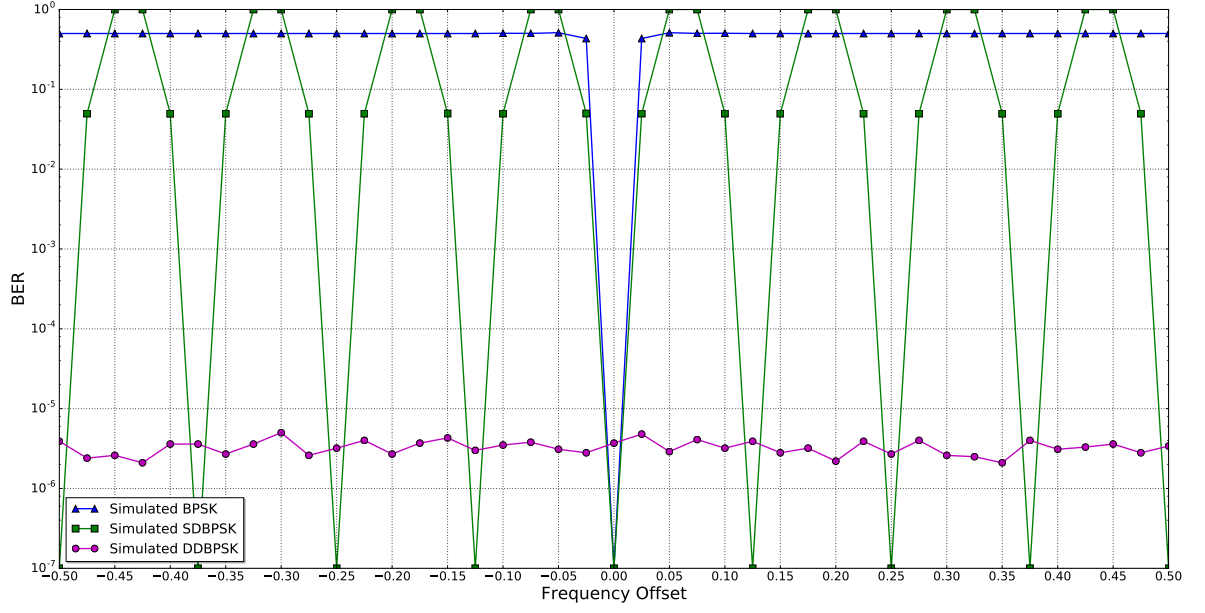


Figure 5.14: BER vs. frequency offset for different BPSK schemes

The BPSK scheme (simulated BPSK) and the single differential BPSK scheme (simulated SDBPSK) can not resist the variation of the frequency offset whereas the double differential BPSK scheme (simulated DDBPSK) has almost unchanged BER curve when the frequency offset varies. Therefore, only the double differential BPSK scheme (simulated DDBPSK) is capable of resisting the effect of the frequency offset.

As we have known in section 4.4, the first single differential decoder converts the frequency offset term $e^{j2\pi\Delta f t}$ into a constant phase offset term $e^{j2\pi\Delta f T}$ where the frequency offset $\Delta f = \Delta f_{norm} \cdot f_{sample}$ from equation (5.3) and the symbol period $T = sps \cdot T_{sample}$. Therefore, for the single differential BPSK scheme (simulated SDBPSK), the output of the single differential decoder contains the phase offset term $e^{j2\pi\Delta f T}$ where $\Delta f \cdot T = \Delta f_{norm} \cdot sps$ and $sps = 8$ due to that $f_{sample} = 80\text{kHz}$ and $R_b = 10\text{kbps}$. Table 5.1 lists the normalized frequency offset Δf_{norm} and the corresponding phase offset $2\pi\Delta f T$ from the single differential decoder. When $\Delta f_{norm} = 0, 0.125, 0.25, 0.375$ and 0.5 , the phase offset $2\pi\Delta f T$ that stems from the frequency offset is exactly the multiple of 2π . Hence, it does not affect the BER performance of the SDBPSK as shown in Fig. 5.14.

Table 5.1: The normalized frequency offset Δf_{norm} and phase offset $2\pi\Delta f T$

Δf_{norm}	$\Delta f T$	$2\pi\Delta f T$
0	0	0
0.125	1	2π
0.25	2	4π
0.375	3	6π
0.5	4	8π

5.3.4 bladeRF Test

The bladeRF SDR platform is used by the testbed for real RF transmission. Figure 5.15 shows the screenshot of the control panel where the frequency and gain of the bladeRF SDR platform can be tuned. When the TX frequency is the same as the RX frequency, the BER is smaller than 10^{-6} and the communication is successful as shown in Fig. 5.15 where $f_{sample} = 80\text{kHz}$, $R_b = 10\text{kbps}$, $\text{SNR} = 17.42\text{dB}$. When the TX and RX frequencies are not identical, the BER still remains very low ($<10^{-6}$) as shown in Fig. 5.16 and 5.17 where $f_{sample} = 80\text{kHz}$,

$R_b = 10\text{kbps}$, $\text{SNR} = 19\text{dB}$. In the real RF transmission, the bladeRF channel bandwidth is set to be 1.5MHz . The test results in Fig. 5.16 and 5.17 show that if the TX frequency is set to be 2450MHz , the BER remains very low ($<10^{-6}$) when the RX frequency is tuned in the range between 2449.25MHz and 2450.75MHz and $\text{SNR} = 19\text{dB}$. If the RX frequency is tuned to go beyond this range, the BER performance will get worse significantly because the RX frequency is out of the channel bandwidth. It indicates that the testbed is robust against the effect of the frequency offset within the channel bandwidth.

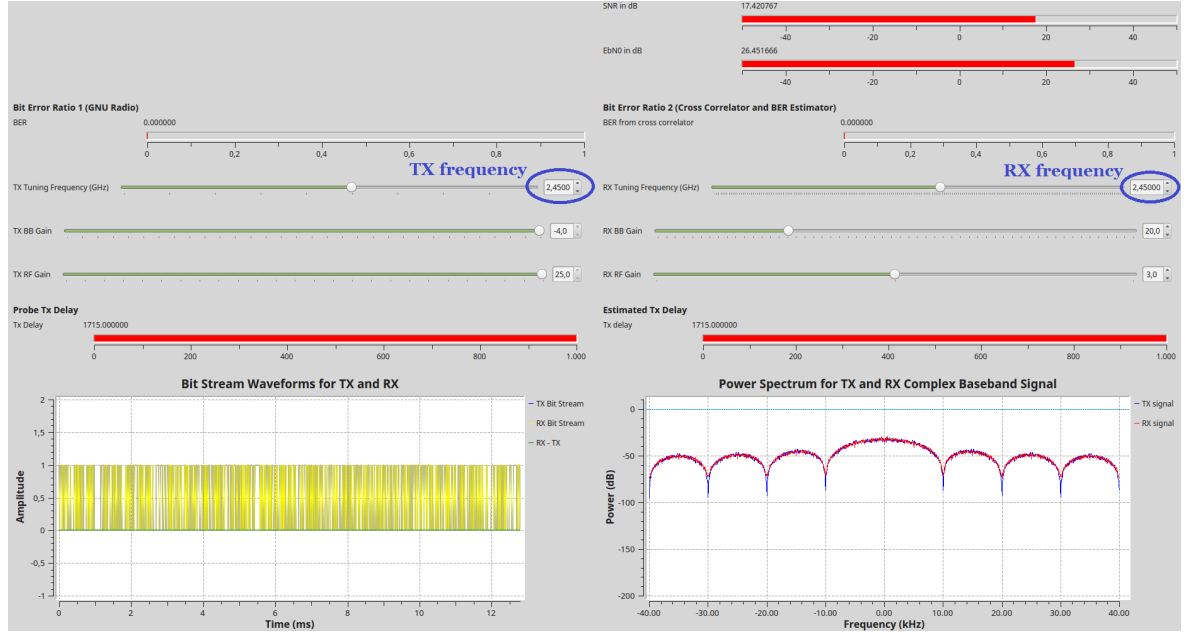


Figure 5.15: The panel screenshot for bladeRF test when RX and TX frequencies are equal

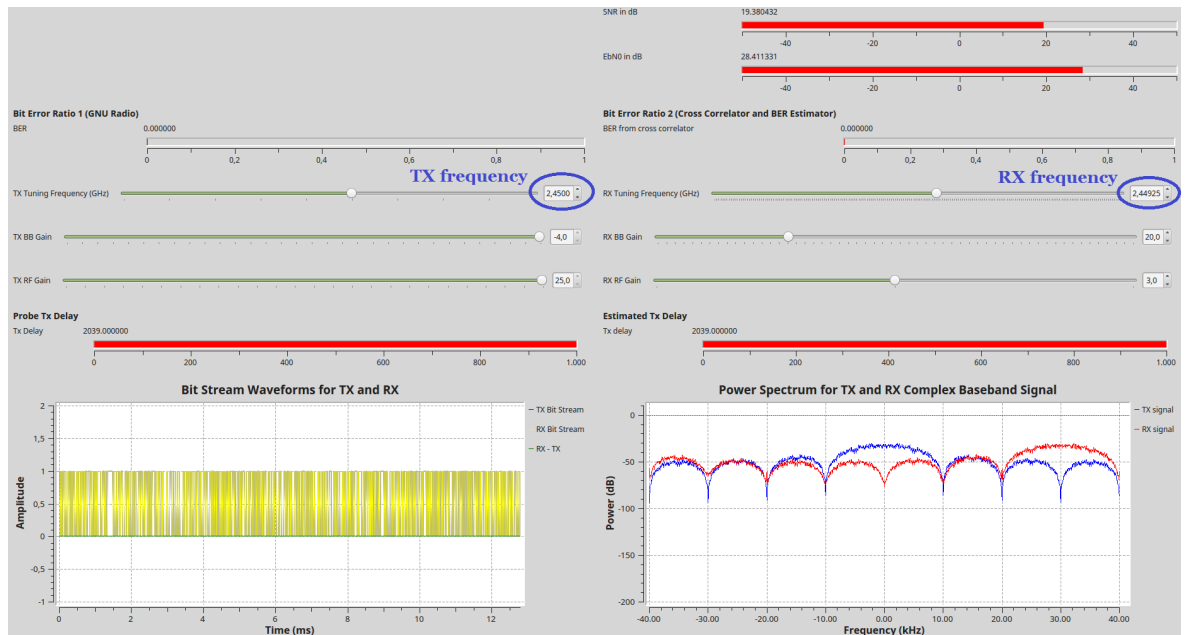


Figure 5.16: The effect of frequency offset when RX frequency is smaller than TX frequency

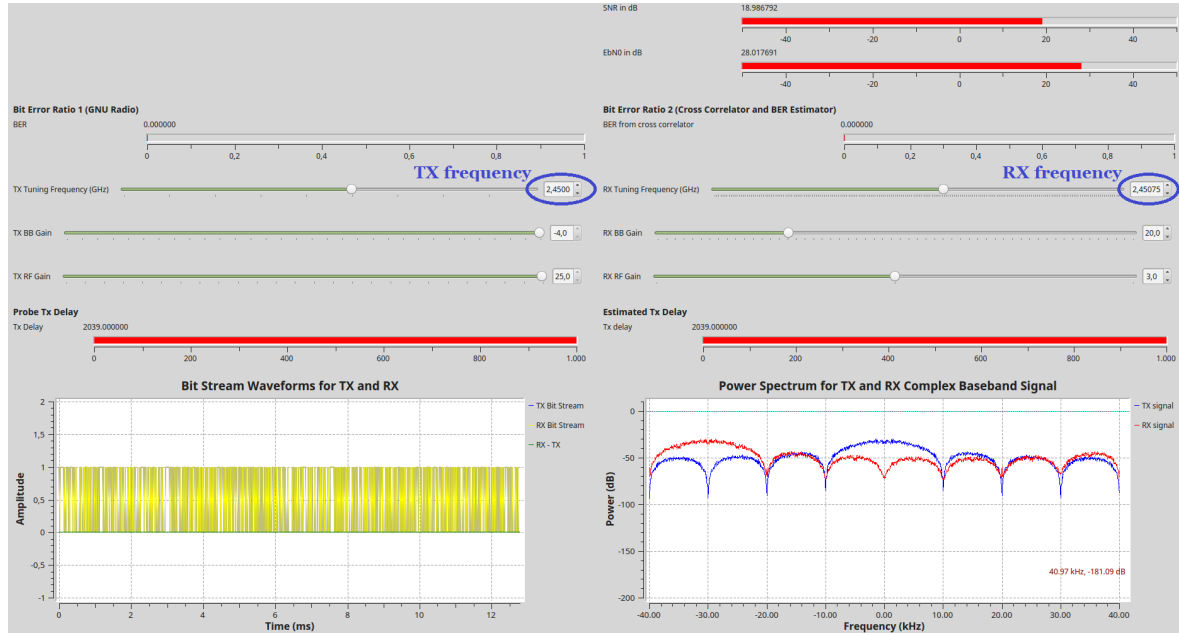


Figure 5.17: The effect of frequency offset when RX frequency is bigger than TX frequency

In the software simulation, the SNR can be controlled by noise level in the channel model. In the real RF transmission experiment, we connected the TX RF terminal and the RX RF terminal on the bladeRF via an attenuator with adjustable attenuation from 0 to 70dB as shown in Fig. 5.18. However, on the one hand, it is hard to precisely control real SNR in the RF communication channel by adjusting the attenuator and the amplification gain of the bladeRF platform. On the other hand, the SNR estimator is not able to output the correct SNR value when the real SNR is smaller than 0dB. Therefore, the accurate BER curve for the testbed running on the SDR platform can not be produced and presented here. Nevertheless, it can be a topic for future researchers who are interested in plotting the BER curve when the testbed is running on the SDR platform for RF transmission.

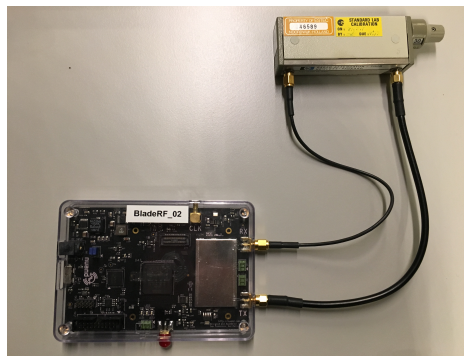


Figure 5.18: The bladeRF platform is connected with an adjustable attenuator

Although the accurate BER curve for the bladeRF test can not be plotted, we expected SNR value to manually plot BER versus E_b/N_0 curve as shown in Fig. 5.19 where the bladeRF platform is running at sample rate =

80kHz and bit rate = 10kbps. The attenuator has a step of 10dB. When the attenuator is set to be 30dB and the bladeRF platform has 25dB TX RF gain, -4dB TX BB gain, 3dB RX RF gain and 20dB RX BB gain, the SNR value estimated by the SNR estimator is approximately 17dB. This value is close to the real SNR value because it has a change of approximately 1dB when the bladeRF TX RF gain has a variation of 1dB. When the real SNR is smaller than 0dB and the bladeRF TX RF gain continues to decrease with 1dB step, the estimated SNR increases rapidly. Therefore, we concluded that the estimated SNR is incorrect when the real SNR is smaller than 0dB.

Therefore, we took the estimated SNR value of 17dB as a reference when the bladeRF testbed has 25dB TX RF gain, -4dB TX BB gain, 3dB RX RF gain and 20dB RX BB gain and the attenuator has 30dB attenuation. The expected SNR that is used to plot BER curve is based on the reference SNR. For example, when the attenuator is set to be 40dB and the TX RF gain is 15 dB while other gains remain the same, the expected SNR is 17dB (reference SNR) - 10dB (attenuation) - 10dB (TX RF gain decrease) = -3dB. By adjusting the attenuator and the bladeRF TX RF gain, we got multiple expected SNR values and corresponding BER values. Then, based on these values, we plotted the BER vs. SNR curve for the bladeRF test as shown in Fig. 5.19 where the BER performance degrades after SNR is smaller than -3dB.

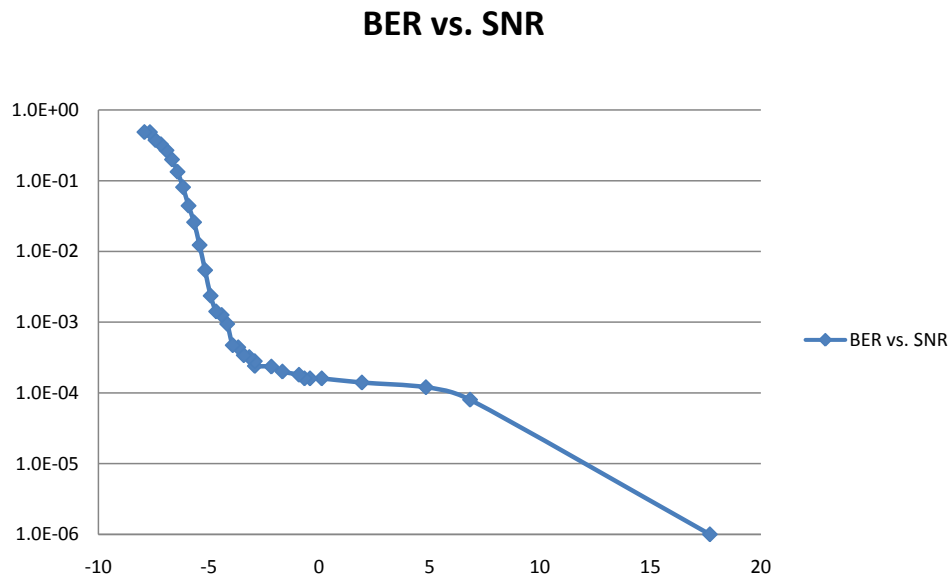


Figure 5.19: BER vs. SNR curve for bladeRF test

6

CHAPTER

CONCLUSION

This ending chapter consists of two sections. At first, it will draw some conclusions with regard to this research project. In second part, some recommendations about improvements and possible future works will be given.

6.1 Conclusions

In this project, both the [VNB](#) transmitter and VNB receiver have been modeled in GNU-radio flowgraph. Some blocks used in the transmitter and receiver such as double differential encoder and decoder, sampler are completely implemented by python codes. A point-to-point VNB communication testbed has been built up where software-simulated communication via a channel model and real RF transmission by means of bladeRF [SDR](#) platform can be conducted with success.

When the testbed is running, some important performance parameters such as [SNR](#) and [BER](#) can be measured in real-time. The communication testbed can overcome the effects of the phase and frequency offsets, and meanwhile its overall communication reliability is satisfying when it is running on the bladeRF SDR platform.

To conclude, the slow wireless radio is feasible in real-world and it can achieve reliable communication even if received RF signal has low [SNR](#).

6.2 Recommendations

The symbols at the transmitter are shaped as rectangle by the repeater. A matched filter at the receiver matches the pulse shapes of the received symbols with the transmitted symbol shape. In future work, root-raised cosine ([RRC](#)) filter can be considered in shaping symbol because the spectrum of symbols with [RRC](#) pulse shapes is closer to ideal rectangle. At the transmitter, the RRC pulse shaping work can be done by an RRC filter instead of a repeater. At the receiver, an RRC matched filter can replace the moving average filter.

In the sampler block, there are some spaces for improvement. When the number of samples per symbol (*sps*) is large, it takes longer time for the sampler to output optimum samples. Hence, improving the timing recovery algorithm used in the sampler is also worth considering in future work.

The double differential encoder and decoder as well as sampler which are implemented in python can also be implemented in C++ so that the execution efficiency for these blocks can be improved. All of the blocks within the transmitter and receiver are implemented in software. The bladeRF SDR platform is equipped with an [FPGA](#). The user is allowed to program the FPGA on the bladeRF platform so that some digital signal processing tasks can be performed on it. Therefore, it is possible to code the transmitter chain (BPSK modulator + double differential encoder + repeater) and the receiver chain ([DC](#)-offset compensator + first differential decoder + matched filter + sampler + second differential decoder + BPSK demodulator) in hardware description language ([HDL](#)) and implement them on FPGA hardware.

With regard to the SNR estimation at the receiver, the estimated SNR is incorrect and unstable when the real SNR is smaller than 0dB in RF transmission scenario. Nowadays, there are many algorithms for low SNR estimation. Anyone who is interested in this topic can explore it on the testbed in future research work. When the testbed is running on the bladeRF platform, it is difficult to control the real SNR and estimate it accurately. That is the reason why we can not plot BER vs. EbNo curve when the testbed is running on the bladeRF. One future work can be concentrated on how to precisely control real SNR and estimate it when the testbed is running on the SDR platform so that the BER vs. EbNo curve can be plotted. In addition, it is also a valuable topic to find out a feasible way to precisely measure the TX and RX powers on the bladeRF SDR platform or other SDR platforms.

The VNB wireless communication is an interesting research field. We hope this testbed could help future researchers when they are exploring the VNB wireless communication and its potential applications in WSNs.

Bibliography

- [1] Wikipedia, “Wireless Communication.” <https://en.wikipedia.org/wiki/Wireless>. [last accessed on 17 August 2017].
- [2] Wikipedia, “Electromagnetic Spectrum.” https://en.wikipedia.org/wiki/Electromagnetic_spectrum. [last accessed on 17 August 2017].
- [3] Wikipedia, “Internet of things.” https://en.wikipedia.org/wiki/Internet_of_things. [last accessed on 17 August 2017].
- [4] F. Mattern and C. Floerkemeier, “From the internet of computers to the internet of things,” *From active data management to event-based systems and more*, pp. 242–259, 2010.
- [5] C. Alcaraz, P. Najera, J. Lopez, and R. Roman, “Wireless sensor networks and the internet of things: Do we need a complete integration?,” in *1st International Workshop on the Security of the Internet of Things (SecIoT 10)*, 2010.
- [6] R. Musaloiu-E and A. Terzis, “Minimising the effect of WiFi interference in 802.15. 4 wireless sensor networks,” *International Journal of Sensor Networks*, vol. 3, no. 1, pp. 43–54, 2008.
- [7] F. Luo, C. Jiang, H. Zhang, X. Wang, L. Zhang, and Y. Ren, “Node Energy Consumption Analysis in Wireless Sensor Networks,” in *Vehicular Technology Conference (VTC Fall), 2014 IEEE 80th*, pp. 1–5, IEEE, 2014.
- [8] R. C. Carrano, D. Passos, L. C. Magalhaes, and C. V. Albuquerque, “Survey and taxonomy of duty cycling mechanisms in wireless sensor networks,” *IEEE Communications Surveys & Tutorials*, vol. 16, no. 1, pp. 181–194, 2014.
- [9] C. Enz, N. Scolari, and U. Yodprasit, “Ultra low-power radio design for wireless sensor networks,” in *Radio-Frequency Integration Technology: Integrated Circuits for Wideband Communication and Wireless Sensor Networks, 2005. Proceedings. 2005 IEEE International Workshop on*, pp. 1–17, IEEE, 2005.
- [10] R. K. Dokania, X. Y. Wang, S. G. Tallur, and A. B. Apsel, “A low power impulse radio design for body-area-networks,” *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 58, no. 7, pp. 1458–1469, 2011.
- [11] I. Demirkol, C. Ersoy, and E. Onur, “Wake-up receivers for wireless sensor networks: benefits and challenges,” *IEEE wireless Communications*, vol. 16, no. 4, 2009.
- [12] J. Oller, I. Demirkol, J. Casademont, and J. Paradells, “Design, development and performance evaluation of a low-cost, low-power wake-up radio system for wireless sensor networks,” *ACM Transactions on Sensor Networks (TOSN)*, vol. 10, no. 1, p. 11, 2013.
- [13] P. Le-Huy and S. Roy, “Low-power 2.4 GHz wake-up radio for wireless sensor networks,” in *Networking and Communications, 2008. WIMOB’08. IEEE International Conference on Wireless and Mobile Computing*, pp. 13–18, IEEE, 2008.
- [14] J. Zhang, P. V. Orlik, Z. Sahinoglu, A. F. Molisch, and P. Kinney, “UWB systems for wireless sensor networks,” *Proceedings of the IEEE*, vol. 97, no. 2, pp. 313–331, 2009.

- [15] M. Shen, T. Koivisto, T. Peltonen, E. Zheng, Tjukanoff, and H. Tenhunen, "UWB radio module design for wireless sensor networks," in *NORCHIP Conference, 2005. 23rd*, pp. 184–187, IEEE, 2005.
- [16] I. Oppermann, L. Stoica, A. Rabbachin, Z. Shelby, and J. Haapola, "UWB wireless sensor networks: UWEN-a practical example," *IEEE Communications Magazine*, vol. 42, no. 12, pp. S27–S32, 2004.
- [17] W. Hirt, "The european UWB radio regulatory and standards framework: overview and implications," in *Ultra-Wideband, 2007. ICUWB 2007. IEEE International Conference*, pp. 733–738, IEEE, 2007.
- [18] The Telecommunication Engineering Group at University of Twente, "Slow Wireless Project Summary," https://www.utwente.nl/en/eemcs/te/projects/SRR/slow_wireless/, 2017. [last accessed on 20 Oct 2017].
- [19] A. Alkhatib and G. S. Baicher, "Wireless sensor network architecture," in *2012 International Conference on Computer Networks and Communication Systems (CNCS 2012)*, 2012.
- [20] G. Reiter, "Wireless connectivity for the Internet of Things," *Texas Instruments White Paper*, June 2014.
- [21] International Electrotechnical Commission, "Internet of Things: Wireless Sensor Networks," *IEC White Paper*, 2014.
- [22] M. Dillinger, K. Madani, and N. Alonistioti, *Software-defined radio: Architectures, systems and functions*. John Wiley & Sons, 2005.
- [23] F. K. Jondral, "Software-defined radio: basics and evolution to cognitive radio," *EURASIP journal on wireless communications and networking*, vol. 2005, no. 3, pp. 275–283, 2005.
- [24] GNU Radio Community, "GNU Radio Wiki Page." https://wiki.gnuradio.org/index.php/Main_Page. [last accessed on 18 Oct 2017].
- [25] N. Manicka, *GNU radio testbed*. PhD thesis, University of Delaware, 2007.
- [26] SWIG Community Homepage, "What is SWIG?." <http://www.swig.org/index.php>. [last accessed on 18 Oct 2017].
- [27] D. Valerio, "Open source software-defined radio: A survey on GNU radio and its applications," *Telecommunications Research Center, Vienna, Technical Report FTW-TR-2008-002*, 2008.
- [28] Mathworks, "Design and simulate the physical layer of communications systems." <https://nl.mathworks.com/products/communications.html>. [last accessed on 6 Nov 2017].
- [29] Mathworks, "Model-based design for software-defined radio with MATLAB Simulink." <https://nl.mathworks.com/discovery/sdr.html>. [last accessed on 6 Nov 2017].
- [30] National Instruments, "NI Software Defined Radio." <http://www.ni.com/nl-nl/innovations/wireless/software-defined-radio.html>. [last accessed on 6 Nov 2017].
- [31] Ettus Research, "Ettus Research Official Website." <https://www.ettus.com/>. [last accessed on 6 Nov 2017].
- [32] Nuand, "Nuand Homepage." <https://nuand.com/>. [last accessed on 6 Nov 2017].
- [33] Myriad RF, "UmTRX Project." <https://myriadrfr.org/projects/umtrx/>. [last accessed on 6 Nov 2017].
- [34] Great Scott Gadgets, "HackRF One Homepage." <http://greatscottgadgets.com/hackrf/>. [last accessed on 6 Nov 2017].

- [35] Nuand, *bladeRF USB 3.0 Software Defined Radio Brief Datasheet*, 2017.
- [36] Nuand, “Nuand Product Homepage.” <https://www.nuand.com/blog/product/bladerf-x40/>, 2017. [last accessed on 6 Nov 2017].
- [37] J. S. Robert Ghilduta, Brian Padalino, *An SDR-based family radio service transceiver*. Nuand, June 2016.
- [38] Lime, “Lime Microsystems Official Website,” June 2017. [last accessed on 9 June 2017].
- [39] Limemicro, *LMS6002D Multi-band Multi-standard Transceiver Datasheet*. Lime Microsystems, 1.1.0 ed., March 2012.
- [40] Wikipedia, “Phase-Shift Keying.” https://en.wikipedia.org/wiki/Phase-shift_keying. [last accessed on 2 Oct 2017].
- [41] M. K. Simon, S. M. Hinedi, and W. C. Lindsey, *Digital Communication Techniques: Signal Design and Detection*. Prentice Hall PTR, 1995. Chapter 8.
- [42] M. K. Simon and D. Divsalar, “On the implementation and performance of single and double differential detection schemes,” *IEEE Transactions on Communications*, vol. 40, no. 2, pp. 278–291, 1992.
- [43] S. Haykin, *Communication Systems*. Wiley, 2001. Chapter 2 & 6.
- [44] Wikipedia, “Heterodyne Detection.” <https://en.wikipedia.org/wiki/Heterodyne>, 2017. [last accessed on 23 Oct 2017].
- [45] Wikipedia, “Superheterodyne Receiver.” https://en.wikipedia.org/wiki/Superheterodyne_receiver, 2017. [last accessed on 23 Oct 2017].
- [46] Wikipedia, “Direct-conversion Receiver.” https://en.wikipedia.org/wiki/Direct-conversion_receiver, 2017. [last accessed on 23 Oct 2017].
- [47] RF Wireless World, “Heterodyne receiver vs Homodyne receiver - difference between Heterodyne receiver and Homodyne receiver.” <http://www.rfwireless-world.com/Terminology/heterodyne-receiver-versus-homodyne-receiver.html>, 2017. [last accessed on 23 Oct 2017].
- [48] Wikipedia, “IQ imbalance.” https://en.wikipedia.org/wiki/IQ_imbalance, 2017. [last accessed on 23 Oct 2017].
- [49] S. Yamada, O. Boric-Lubecke, and V. M. Lubecke, “Cancellation techniques for LO leakage and DC offset in direct-conversion systems,” in *Microwave Symposium Digest, 2008 IEEE MTT-S International*, pp. 1191–1194, IEEE, 2008.
- [50] L. Litwin, “Matched filtering and timing recovery in digital receivers,” *RF design*, vol. 24, pp. 32–49, September 2001.
- [51] S. W. Smith *et al.*, *The scientist and engineer’s guide to digital signal processing*. California Technical Pub. San Diego, 1997. Chapter 15.
- [52] F. M. Gardner, “A BPSK/QPSK timing-error detector for sampled receivers,” *IEEE Transactions on Communications*, vol. 34, pp. 423–429, May 1986.
- [53] Nutaq, “Symbol timing recovery methods for digital I/Q demodulator.” <https://www.nutaq.com/blog/symbol-timing-recovery-methods-digital-iq-demodulator>. [last accessed on 29 Oct 2017].

- [54] K. Sheng, M. Anjum, and M. A. Dida, "Principle of bit-synchronization loop," in *Multi-Topic Conference (INMIC), 2014 IEEE 17th International*, pp. 115–118, IEEE, 2014.
- [55] H. Xu and H. Zheng, "The simple SNR estimation algorithms for MPSK signals," in *Signal Processing, 2004. Proceedings. ICSP'04. 2004 7th International Conference*, vol. 2, pp. 1781–1785, IEEE, 2004.
- [56] J. Vartiainen, H. Saarnisaari, J. J. Lehtomaki, and M. Juntti, "A blind signal localization and SNR estimation method," in *Military Communications Conference, 2006. MILCOM 2006. IEEE*, pp. 1–7, IEEE, 2006.
- [57] G. Ren, Y. Chang, and H. Zhang, "A new SNR estimator for QPSK modulations in an AWGN channel," *IEEE Transactions on Circuits and Systems II: Express Briefs*, vol. 52, no. 6, pp. 336–338, 2005.
- [58] M. Bakkali, A. Stephenne, and S. Affes, "Generalized moment-based method for SNR estimation," in *Wireless Communications and Networking Conference, 2007. WCNC 2007. IEEE*, pp. 2226–2230, IEEE, 2007.
- [59] D. R. Pauluzzi and N. C. Beaulieu, "A comparison of SNR estimation techniques for the AWGN channel," *IEEE Transactions on communications*, vol. 48, no. 10, pp. 1681–1691, 2000.
- [60] Z. Gajic, *Linear dynamic systems and signals*. Prentice Hall Pearson Education, 2003. Chapter 9.

IQ MODULATION AND DEMODULATION

This appendix will provide the detailed mathematical explanation with regard to up-conversion and down-conversion for complex signal. From here on, we use IQ modulation and IQ demodulation to refer to up-conversion and down-conversion respectively. Figure A.1 illustrates the internal structure of the IQ modulator and demodulator, where all important notations are marked in corresponding positions to help reader understand the mathematical derivations.

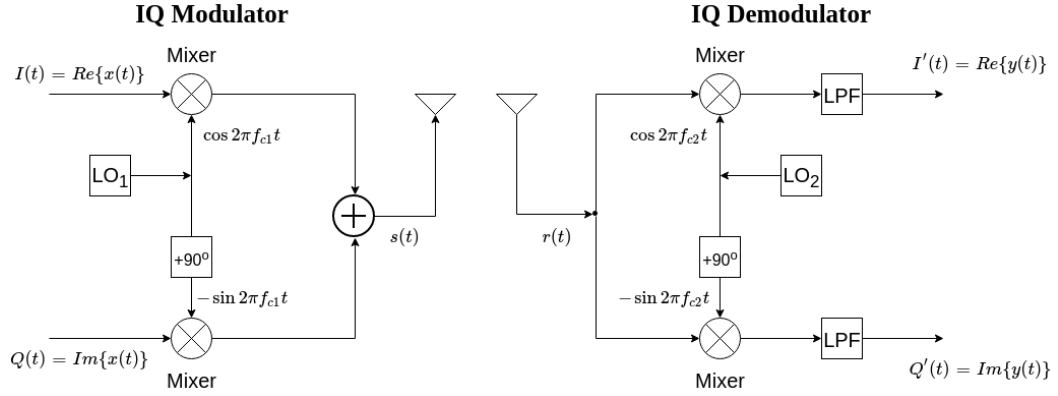


Figure A.1: IQ modulator and demodulator

In this project, we adopted the [BPSK](#) scheme to modulate digital bits into phase changes. The digital information bit is expressed with

$$i(t) = \begin{cases} 0 \\ 1 \end{cases} \quad (\text{A.1})$$

The binary bit 0 or 1 is modulated into binary phase 0 or π with the modulation order $M = 2$.

$$\phi(t) = \frac{2\pi \cdot i(t)}{M} = \pi \cdot i(t) = \begin{cases} 0 & i(t) = 0 \\ \pi & i(t) = 1 \end{cases} \quad (\text{A.2})$$

The [BPSK](#)-modulated signal $x(t)$ is expressed in complex form with magnitude of A . For simplification, we suppose that the magnitude A is unitary.

$$\begin{aligned}
x(t) &= A \cdot e^{j \cdot \phi(t)} \\
&= e^{j \cdot \phi(t)} \\
&= \cos \phi(t) + j \cdot \sin \phi(t) \\
&= \begin{cases} 1 & i(t) = 0 \\ -1 & i(t) = 1 \end{cases}
\end{aligned} \tag{A.3}$$

The BPSK-modulated complex signal $x(t)$ can be decomposed into in-phase component $I(t)$ and quadrature component $Q(t)$ respectively.

$$I(t) = \text{Re}\{x(t)\} = \cos \phi(t) \tag{A.4}$$

$$Q(t) = \text{Im}\{x(t)\} = \sin \phi(t) \tag{A.5}$$

Before proceeding, we listed the trigonometric formulas that are used frequently in the mathematical derivation.

$$\cos(x + \phi) = \cos x \cdot \cos \phi - \sin x \cdot \sin \phi \tag{A.6}$$

$$\cos(x - \phi) = \cos x \cdot \cos \phi + \sin x \cdot \sin \phi \tag{A.7}$$

Combining the above two formulas, we get:

$$\cos x \cdot \cos \phi = \frac{1}{2} \cos(x + \phi) + \frac{1}{2} \cos(x - \phi) \tag{A.8}$$

Both the IQ modulation and demodulation are based on the formula A.8. The carrier frequency of the local oscillator (LO₁) at the IQ modulator is defined as f_{c1} while the carrier frequency of the local oscillator (LO₂) at the IQ demodulator is defined as f_{c2} . The difference between the two carrier frequencies is $\Delta f = f_{c1} - f_{c2}$. Ideally, the frequency f_{c1} matches f_{c2} completely. As a result, the frequency difference Δf remains at zero. In practical case, the frequency f_{c1} does not match f_{c2} , and thus some frequency difference Δf exists.

In the IQ modulator, the in-phase component $I(t)$ is mixed with $\cos(2\pi f_{c1}t)$ while the quadrature component $Q(t)$ is multiplied with $\cos(2\pi f_{c1}t + \frac{\pi}{2}) = -\sin 2\pi f_{c1}t$. The signal $s(t)$ that is going to be transmitted is expressed as follows:

$$\begin{aligned}
s(t) &= I(t) \cdot \cos 2\pi f_{c1}t - Q(t) \cdot \sin 2\pi f_{c1}t \\
&= \cos \phi(t) \cdot \cos 2\pi f_{c1}t - \sin \phi(t) \cdot \sin 2\pi f_{c1}t \\
&= \cos(2\pi f_{c1}t + \phi(t))
\end{aligned} \tag{A.9}$$

To simplify the mathematical derivation, we assume that the signal travels through a channel without any noise. Hence, in the IQ demodulator, the received signal $r(t)$ is equal to the transmitted signal $s(t)$.

$$r(t) = s(t) = \cos(2\pi f_{c1}t + \phi(t)) \tag{A.10}$$

The received signal $r(t)$ is split into in-phase path and quadrature path in the IQ demodulator. Along the in-phase path, the received signal $r(t)$ is mixed with $\cos 2\pi f_{c2}t$.

$$\begin{aligned}
& r(t) \cdot \cos 2\pi f_{c2}t \\
&= \cos(2\pi f_{c1}t + \phi(t)) \cdot \cos 2\pi f_{c2}t \\
&= \frac{1}{2} \cos(2\pi f_{c1}t + \phi(t) + 2\pi f_{c2}t) + \frac{1}{2} \cos(2\pi f_{c1}t + \phi(t) - 2\pi f_{c2}t) \\
&= \frac{1}{2} \cos(2\pi(f_{c1} + f_{c2})t + \phi(t)) + \frac{1}{2} \cos(2\pi(f_{c1} - f_{c2})t + \phi(t)) \tag{A.11}
\end{aligned}$$

After being mixed, the received signal comprises two components, one with high frequency $f_{c1} + f_{c2}$ and the other with low frequency $f_{c1} - f_{c2}$. Then, the mixed signal $r(t) \cdot \cos 2\pi f_{c2}t$ passes through a low-pass filter (LPF) to remove its high frequency part. The in-phase component $I'(t)$ of the information signal is recovered near the baseband with half amplitude and slight frequency offset Δf .

$$\begin{aligned}
I'(t) &= \frac{1}{2} \cos(2\pi(f_{c1} - f_{c2})t + \phi(t)) \\
&= \frac{1}{2} \cos(2\pi\Delta f t + \phi(t)) \tag{A.12}
\end{aligned}$$

Similarly, the received signal $r(t)$ is mixed with $\cos(2\pi f_{c2}t + \frac{\pi}{2}) = -\sin 2\pi f_{c2}t$ along the quadrature path.

$$\begin{aligned}
& r(t) \cdot (-\sin 2\pi f_{c2}t) \\
&= \cos(2\pi f_{c1}t + \phi(t)) \cdot (-\sin 2\pi f_{c2}t) \\
&= \cos(2\pi f_{c1}t + \phi(t)) \cdot \cos\left(2\pi f_{c2}t + \frac{\pi}{2}\right) \\
&= \frac{1}{2} \cos\left(2\pi f_{c1}t + \phi(t) + 2\pi f_{c2}t + \frac{\pi}{2}\right) + \frac{1}{2} \cos\left(2\pi f_{c1}t + \phi(t) - (2\pi f_{c2}t + \frac{\pi}{2})\right) \\
&= \frac{1}{2} \cos\left(2\pi(f_{c1} + f_{c2})t + \phi(t) + \frac{\pi}{2}\right) + \frac{1}{2} \cos\left(2\pi(f_{c1} - f_{c2})t + \phi(t) - \frac{\pi}{2}\right) \\
&= -\frac{1}{2} \sin(2\pi(f_{c1} + f_{c2})t + \phi(t)) + \frac{1}{2} \sin(2\pi(f_{c1} - f_{c2})t + \phi(t)) \tag{A.13}
\end{aligned}$$

The mixed signal on the quadrature path also contains high-frequency part and low-frequency part. The subsequent low-pass filter (LPF) removes the part with high frequency $f_{c1} + f_{c2}$ and passes the low-frequency part through. Then, the quadrature component $Q'(t)$ of information signal is reconstructed near the baseband with half amplitude and small frequency offset Δf .

$$\begin{aligned}
Q'(t) &= \frac{1}{2} \sin(2\pi(f_{c1} - f_{c2})t + \phi(t)) \\
&= \frac{1}{2} \sin(2\pi\Delta f t + \phi(t)) \tag{A.14}
\end{aligned}$$

The recovered signal $y(t)$ is composed of its in-phase part $I'(t) = \text{Re}\{y(t)\}$ and its quadrature part $Q'(t) = \text{Im}\{y(t)\}$.

$$\begin{aligned}
y(t) &= I'(t) + j \cdot Q'(t) \\
&= \frac{1}{2} \cos(2\pi\Delta f t + \phi(t)) + j \cdot \frac{1}{2} \sin(2\pi\Delta f t + \phi(t)) \\
&= \frac{1}{2} e^{j(2\pi\Delta f t + \phi(t))} \\
&= \frac{1}{2} e^{j2\pi\Delta f t} \cdot e^{j\phi(t)} \\
&= \frac{1}{2} e^{j2\pi\Delta f t} \cdot x(t) \tag{A.15}
\end{aligned}$$

Compared with the BPSK-modulated signal $x(t)$ before IQ modulation, the reconstructed signal $y(t)$ after IQ demodulation is reduced to half in its magnitude and rotated by the frequency offset $e^{j2\pi\Delta f t}$ in constellation diagram. The recovered signal $y(t)$ is the BPSK-modulated signal. It will be demodulated by the BPSK demodulator into binary bits. However, the frequency offset term $e^{j2\pi\Delta f t}$ in $y(t)$ could cause the BPSK demodulator to detect $x(t)$ incorrectly and output erroneous bits. To eliminate the negative effect of the frequency offset term $e^{j2\pi\Delta f t}$ on the BPSK demodulator, we adopted double differential coding method to encode $x(t)$ after BPSK modulation and decode $y(t)$ before BPSK demodulation.

The working principle with regard to IQ modulation and demodulation can also be understood in frequency domain. For clarity, let us define the real and imaginary parts of the input signal $x(t)$ to the IQ modulator as

$$x_I(t) = \text{Re}\{x(t)\} \quad (\text{A.16})$$

$$x_Q(t) = \text{Im}\{x(t)\} \quad (\text{A.17})$$

The in-phase (real) part and quadrature (imaginary) part of the input signal $x(t)$ have their own Fourier transforms as follows:

$$\mathcal{F}\{x_I(t)\} = X_I(f) \quad (\text{A.18})$$

$$\mathcal{F}\{x_Q(t)\} = X_Q(f) \quad (\text{A.19})$$

For convenience, let us assume that the carrier frequency f_{c1} in the IQ modulator equals the carrier frequency f_{c2} in the IQ demodulator, that is, $f_{c1} = f_{c2} = f_c$. The Fourier transforms for the carriers $\cos 2\pi f_c t$ and $\sin 2\pi f_c t$ are as follows:

$$\mathcal{F}\{\cos 2\pi f_c t\} = \frac{1}{2}[\delta(f + f_c) + \delta(f - f_c)] = \frac{1}{2}\delta(f + f_c) + \frac{1}{2}\delta(f - f_c) \quad (\text{A.20})$$

$$\mathcal{F}\{\sin 2\pi f_c t\} = \frac{j}{2}[\delta(f + f_c) - \delta(f - f_c)] = \frac{j}{2}\delta(f + f_c) - \frac{j}{2}\delta(f - f_c) \quad (\text{A.21})$$

It is noteworthy that the mixer performs multiplication in time domain and convolution in frequency domain. In the subsequent derivation, the sifting property of δ -function $X(f) * \delta(f \pm f_c) = X(f \pm f_c)$ is applied. After being mixed with the carrier along the in-phase path in the IQ modulator, the in-phase component in frequency domain becomes:

$$\begin{aligned} & \mathcal{F}\{x_I(t) \cdot \cos 2\pi f_c t\} \\ &= X_I(f) * \left[\frac{1}{2}\delta(f + f_c) + \frac{1}{2}\delta(f - f_c) \right] \\ &= \frac{1}{2} \cdot X_I(f) * \delta(f + f_c) + \frac{1}{2} \cdot X_I(f) * \delta(f - f_c) \\ &= \frac{1}{2} \cdot X_I(f + f_c) + \frac{1}{2} \cdot X_I(f - f_c) \end{aligned} \quad (\text{A.22})$$

After being mixed with the carrier along the quadrature path in the IQ modulator, the quadrature component in frequency domain becomes:

$$\begin{aligned}
& \mathcal{F}\{x_Q(t) \cdot (-\sin 2\pi f_c t)\} \\
&= X_Q(f) * \left[-\frac{j}{2}\delta(f+f_c) + \frac{j}{2}\delta(f-f_c)\right] \\
&= -\frac{j}{2} \cdot X_Q(f) * \delta(f+f_c) + \frac{j}{2} \cdot X_Q(f) * \delta(f-f_c) \\
&= -\frac{j}{2} \cdot X_Q(f+f_c) + \frac{j}{2} \cdot X_Q(f-f_c)
\end{aligned} \tag{A.23}$$

The signal to be transmitted $s(t)$ is the sum of the mixed in-phase and mixed quadrature components and its Fourier transform is $S(f)$.

$$s(t) = x_I(t) \cdot \cos 2\pi f_c t + x_Q(t) \cdot (-\sin 2\pi f_c t) \tag{A.24}$$

$$\begin{aligned}
S(f) &= \mathcal{F}\{s(t)\} \\
&= \mathcal{F}\{x_I(t) \cdot \cos 2\pi f_c t\} + \mathcal{F}\{x_Q(t) \cdot (-\sin 2\pi f_c t)\} \\
&= \frac{1}{2} \cdot X_I(f+f_c) + \frac{1}{2} \cdot X_I(f-f_c) - \frac{j}{2} \cdot X_Q(f+f_c) + \frac{j}{2} \cdot X_Q(f-f_c)
\end{aligned} \tag{A.25}$$

For the purpose of simplification, let us assume that the signal $s(t)$ is transmitted through an ideal channel without any noise and interference so that the received signal $r(t)$ and its Fourier transform $R(f)$ are equal to the transmitted signal $s(t)$ and its Fourier transform $S(f)$ respectively.

$$r(t) = s(t) \tag{A.26}$$

$$\begin{aligned}
R(f) &= S(f) \\
&= \frac{1}{2} \cdot X_I(f+f_c) + \frac{1}{2} \cdot X_I(f-f_c) - \frac{j}{2} \cdot X_Q(f+f_c) + \frac{j}{2} \cdot X_Q(f-f_c)
\end{aligned} \tag{A.27}$$

Now, let us explain how the IQ demodulator is working in frequency domain. Along the in-phase path in the IQ demodulator, the received signal $r(t)$ is mixed with the cosine carrier and becomes:

$$\begin{aligned}
& \mathcal{F}\{r(t) \cdot \cos 2\pi f_c t\} \\
&= R(f) * \left[\frac{1}{2}\delta(f+f_c) + \frac{1}{2}\delta(f-f_c)\right] \\
&= \frac{1}{2} \cdot R(f+f_c) + \frac{1}{2} \cdot R(f-f_c) \\
&= \frac{1}{4} \cdot X_I(f+2f_c) + \frac{1}{4} \cdot X_I(f) - \frac{j}{4} \cdot X_Q(f+2f_c) + \cancel{\frac{j}{4} \cdot X_Q(f)} + \\
&\quad \frac{1}{4} \cdot X_I(f) + \frac{1}{4} \cdot X_I(f-2f_c) - \cancel{\frac{j}{4} \cdot X_Q(f)} + \frac{j}{4} \cdot X_Q(f-2f_c) \\
&= \frac{1}{4} \cdot X_I(f+2f_c) + \frac{1}{2} \cdot X_I(f) + \frac{1}{4} \cdot X_I(f-2f_c) \\
&\quad - \frac{j}{4} \cdot X_Q(f+2f_c) + \frac{j}{4} \cdot X_Q(f-2f_c)
\end{aligned} \tag{A.28}$$

Then, the mixed signal $r(t) \cdot \cos 2\pi f_c t$ along the in-phase path passes through a low-pass filter (LPF) and all of its high-frequency contents $\frac{1}{4} \cdot X_I(f+2f_c)$, $\frac{1}{4} \cdot X_I(f-2f_c)$, $-\frac{j}{4} \cdot X_Q(f+2f_c)$, $\frac{j}{4} \cdot X_Q(f-2f_c)$ are removed. The output of the low-pass filter (LPF) $y_I(t)$ on the in-phase path and its Fourier transform $Y_I(f)$ are as follows:

$$\begin{aligned}
Y_I(f) &= \mathcal{F}\{y_I(t)\} \\
&= \frac{1}{2} \cdot X_I(f)
\end{aligned} \tag{A.29}$$

$$y_I(t) = \frac{1}{2} \cdot x_I(t) \tag{A.30}$$

Along the quadrature path in the IQ demodulator, the received signal $r(t)$ is mixed with the sine carrier and becomes:

$$\begin{aligned}
&\mathcal{F}\{r(t) \cdot (-\sin 2\pi f_c t)\} \\
&= R(f) * \left[-\frac{j}{2}\delta(f + f_c) + \frac{j}{2}\delta(f - f_c)\right] \\
&= -\frac{j}{2} \cdot R(f + f_c) + \frac{j}{2} \cdot R(f - f_c) \\
&= -\frac{j}{4} \cdot X_I(f + 2f_c) - \cancel{\frac{j}{4} \cdot X_I(f)} - \frac{1}{4} \cdot X_Q(f + 2f_c) + \frac{1}{4} \cdot X_Q(f) + \\
&\quad \cancel{\frac{j}{4} \cdot X_I(f)} + \frac{j}{4} \cdot X_I(f - 2f_c) + \frac{1}{4} \cdot X_Q(f) - \frac{1}{4} \cdot X_Q(f - 2f_c) \\
&= -\frac{1}{4} \cdot X_Q(f + 2f_c) + \frac{1}{2} \cdot X_Q(f) - \frac{1}{4} \cdot X_Q(f - 2f_c) \\
&\quad - \frac{j}{4} \cdot X_I(f + 2f_c) + \frac{j}{4} \cdot X_I(f - 2f_c)
\end{aligned} \tag{A.31}$$

The mixed signal $r(t) \cdot (-\sin 2\pi f_c t)$ along the quadrature path also passes through a low-pass filter (LPF) and all of its high-frequency contents $-\frac{1}{4} \cdot X_Q(f + 2f_c)$, $-\frac{1}{4} \cdot X_Q(f - 2f_c)$, $-\frac{j}{4} \cdot X_I(f + 2f_c)$, $\frac{j}{4} \cdot X_I(f - 2f_c)$ are eliminated. The output of the low-pass filter (LPF) $y_Q(t)$ on the quadrature path and its Fourier transform $Y_Q(f)$ are as follows:

$$\begin{aligned}
Y_Q(f) &= \mathcal{F}\{y_Q(t)\} \\
&= \frac{1}{2} \cdot X_Q(f)
\end{aligned} \tag{A.32}$$

$$y_Q(t) = \frac{1}{2} \cdot x_Q(t) \tag{A.33}$$

The IQ demodulator output $y(t)$ is equal to the IQ modulator input $x(t)$ with half magnitude. Thus, the signal is recovered.

$$\begin{aligned}
y(t) &= y_I(t) + j \cdot y_Q(t) \\
&= \frac{1}{2} \cdot x_I(t) + j \cdot \frac{1}{2} \cdot x_Q(t) \\
&= \frac{1}{2} \cdot x(t)
\end{aligned} \tag{A.34}$$

DOUBLE DIFFERENTIAL CODING

This appendix will present the detailed mathematical principle behind the double differential encoder and decoder. In order to clarify the working principle of the double differential encoding and decoding, Figure B.1 is used, where all of the important notations that appear in the appendix are marked in corresponding positions to facilitate understanding.

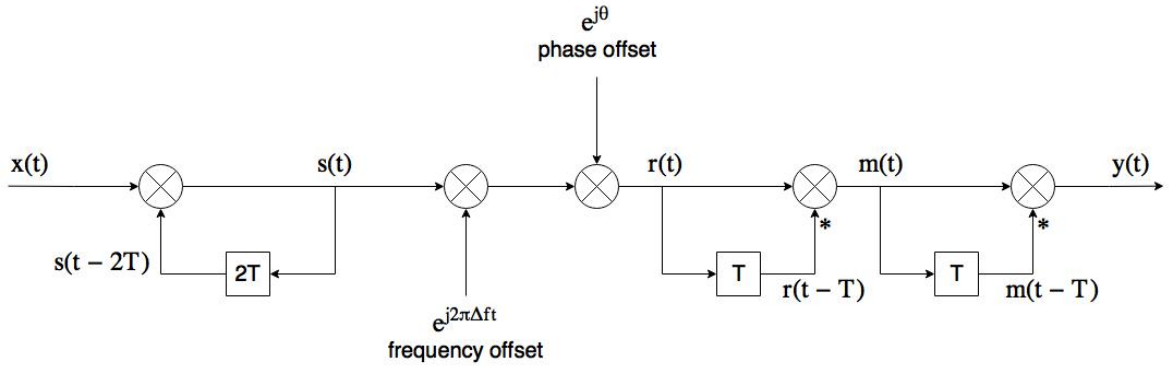


Figure B.1: Double differential coding

$x(t)$ is defined as the modulated complex signal immediately after the BPSK modulator. In order to simplify mathematical derivation, it is assumed that $x(t)$ has unit magnitude.

$$x(t) = e^{j \cdot \phi(t)} \quad (\text{B.1})$$

The binary bit $i(t)$ is modulated into binary phase $\phi(t)$ with the modulation order $M = 2$.

$$\phi(t) = \frac{2\pi \cdot i(t)}{M} = \pi \cdot i(t) = \begin{cases} 0 & i(t) = 0 \\ \pi & i(t) = 1 \end{cases} \quad (\text{B.2})$$

$s(t)$ is the signal to be sent to TX RF front-end and it is also the output of the double differential encoder at the transmitter side.

$$s(t) = s(t - 2T) \cdot x(t) \quad (\text{B.3})$$

$r(t)$ is the signal received from RX RF front-end. The frequency offset term $e^{j2\pi\Delta f t}$ is introduced because of the frequency mismatch between the TX RF front-end oscillator and the RX RF front-end oscillator. The phase offset term $e^{j\theta}$ results from some unknown channel characteristics and IQ imbalance.

$$r(t) = s(t) \cdot e^{j2\pi\Delta f t} \cdot e^{j\theta} \quad (\text{B.4})$$

The received signal $r(t)$ then passes through the 1st differential decoder, in which $r(t)$ is delayed by one symbol period T and then converted into corresponding complex conjugate.

$$r(t - T) = s(t - T) \cdot e^{j2\pi\Delta f(t-T)} \cdot e^{j\theta} \quad (\text{B.5})$$

$$r^*(t - T) = s^*(t - T) \cdot e^{-j2\pi\Delta f(t-T)} \cdot e^{-j\theta} \quad (\text{B.6})$$

The intermediate signal $m(t)$ is the output of the 1st differential decoder and meanwhile the input of the 2nd differential decoder.

$$\begin{aligned} m(t) &= r(t) \cdot r^*(t - T) \\ &= s(t) \cdot e^{j2\pi\Delta f t} \cdot e^{j\theta} \cdot s^*(t - T) \cdot e^{-j2\pi\Delta f(t-T)} \cdot e^{-j\theta} \\ &= s(t) \cdot s^*(t - T) \cdot e^{j2\pi\Delta f T} \end{aligned} \quad (\text{B.7})$$

The intermediate signal $m(t)$ indicates that the phase offset term $e^{j\theta}$ is eliminated by the 1st differential decoder and meanwhile the frequency offset term $e^{j2\pi\Delta f t}$ is converted into a phase offset term $e^{j2\pi\Delta f T}$. Then, the intermediate signal $m(t)$ passes through the 2nd differential decoder, where $m(t)$ is delayed by one symbol period T and turned into its complex conjugate.

$$m(t - T) = s(t - T) \cdot s^*(t - 2T) \cdot e^{j2\pi\Delta f T} \quad (\text{B.8})$$

$$\begin{aligned} m^*(t - T) &= s^*(t - T) \cdot s^{**}(t - 2T) \cdot e^{-j2\pi\Delta f T} \\ &= s^*(t - T) \cdot s(t - 2T) \cdot e^{-j2\pi\Delta f T} \end{aligned} \quad (\text{B.9})$$

The signal $y(t)$ is the output of the 2nd differential decoder and meanwhile the input of the BPSK demodulator.

$$\begin{aligned} y(t) &= m(t) \cdot m^*(t - T) \\ &= s(t) \cdot s^*(t - T) \cdot e^{j2\pi\Delta f T} \cdot s^*(t - T) \cdot s(t - 2T) \cdot e^{-j2\pi\Delta f T} \\ &= s(t) \cdot s^*(t - T) \cdot s^*(t - T) \cdot s(t - 2T) \\ &= s(t) \cdot s(t - 2T) \\ &= s(t - 2T) \cdot x(t) \cdot s(t - 2T) \\ &= s(t - 2T) \cdot s(t - 2T) \cdot x(t) \\ &= x(t) \end{aligned} \quad (\text{B.10})$$

where $s^*(t - T) \cdot s^*(t - T) = 1$ and $s(t - 2T) \cdot s(t - 2T) = 1$ because the signal $s(t)$ is BPSK-modulated and always equal to +1 or -1. In addition, $s(t) = s(t - 2T) \cdot x(t)$ because $s(t)$ is already double-differential encoded. The output signal $y(t) = x(t)$ indicates that the 2nd differential decoder not only removes the phase offset term $e^{j2\pi\Delta f T}$ that stems from the frequency offset term $e^{j2\pi\Delta f t}$ but also recovers $x(t)$, which is the uncoded BPSK-modulated signal at the transmitter side.