TAMPEREEN TEKNILLINEN YLIOPISTO
TAMPERE UNIVERSITY OF TECHNOLOGY

SEYED ALI HASSANI

DESIGN AND IMPLEMENTATION OF FPGA-BASED MULTI-RATE BPSK-QPSK MODEM WITH FOCUS ON CARRIER RECOVERY AND TIME SYNCHRONIZATION

Master of Science Thesis

Examiner: Prof. Jari Nurmi
Prof. Tapio Saramäki
Dr. Waqar Hussain
Examiner and topic approved by the Faculty Council of the Faculty of Computing and Electrical Engineering on 9th of Mar. 2016

ABSTRACT

SYED ALI HASSANI: Design and Implementation of FPGA-Based Multi-Rate BPSK-QPSK Modem with Focus on Carrier Recovery and Time Synchronization
Tampere University of Technology
Master of Science Thesis, 67 pages, 3 Appendix pages
February 2016
Master's Degree Programme in Information Technology
Major: Signal Processing
Examiner: Prof. Jari Nurmi, Prof. Tapio Saramäki , Dr. Waqar Hussain

Keywords: Software Defined Radio, Phase Shift Keying Modulation, Carrier Recovery, Clock and Data Recovery, Time Synchronization, Field-Programmable Gate Array.

Regarding the high performance and reconfigurability of Field Programmable Gate Arrays (FPGAs), many recent software defined radio (SDR) systems are currently being designed and developed on them. On the other hand, a wide variety of applications in communication systems benefits from Phase-Shift Keying (PSK) modulation. Therefore, with respect to practical constraints and limitations, design and implementation of a robust and efficient FPGA-based structure for PSK modulation is an attractive subject of study.

In practice, there is an unavoidable oscillator frequency difference between the transmitter and receiver which poses many challenges for designers. This frequency offset makes carrier recovery and time synchronization as two essential functions of every receiver. The possible solution lies in the closed loop control techniques. In other words, without feedback-based controllers, acceptable performance in a digital radio link is unachievable. The Costas Loop is one of the most effective methods for carrier recovery and its advantage over other methods is that the error signal in the feedback loop is twice as accurate. The Gardner time synchronization method is also introduced as a closed loop clock and data recovery technique and, regarding to its performance, is a potential candidate to be implemented on FPGA-based platforms.

The main body of this thesis work is related to the realization aspects of these methods on FPGA. The thesis spans from the design and implementation of a baseband digital transceiver to connecting it to a radio frequency device, forming a Binary/Quadrature PSK modem. The introduced platform is developed on National Instruments Universal Software Radio Peripheral (NI USRP) equipped with a Xilinx Kintex 7 FPGA.

Many case studies were conducted to evaluate the performance of similar systems considering Signal to Noise Ratio (SNR). In this study, in addition to SNR, the effectiveness of the implemented transceiver has been evaluated based on its ability to deal with the carrier and symbol rate frequency offsets. The introduced platform shows promising

results in its capability to resolve up to ±200 kHz carrier frequency offset and ±14 kHz symbol rate frequency offset (in 18 dB SNR). Furthermore, on the basis of the performed assessment, it is concluded that the introduced model is robust and potential to be applied in array-based or multi-channel networks.

PREFACE

This thesis was made as a part of the requirement for completing the Masters in Information Technology / Signal Processing Engineering. The ground study is about the design and implementation of an FPGA-based Software Defined Radio (SDR) system and hence, the introduced model is applicable to a wide variety of related fields. To facilitate any probable future development, I also stated both theoretical and practical points in the text.

I express my deep gratitude to my supervisor, Professor Jari Nurmi, who patiently supervised the thesis and his supports, comments and guidance led the work to success. Indeed, without his support, this work would not have been possible.

I am also thankful to Professor Tapio Saramäki, for his comments and valuable guidance.

I would also like to thank Dr. Ali Eshkevar (Shahid Behesti University, Iran), who shared his theoretical and practical knowledge of Digital Signal Processing and I am thankful to him for being an honest and lovely friend.

Finally, very special thanks go to my parents for all the support and showing me the value of hard work, and to my inspiring wife, Seyedeh Hamideh, for her patience, love and support.

Seyed Ali Hassani

Tampere, February 2016

CONTENTS

LIST OF FIGURES

LIST OF SYMBOLS AND ABBREVIATIONS

ADC	Analog to Digital Convertor
AWGN	Additive White Gaussian Noise
BER	Bit Error Rate
BPF	Band Pass Filter
BPSK	Binary Phase Shift Keying
CDR	Clock and Data Recovery
CR	Carrier Recovery
DDC	Digital Down Convertor
DDS	Direct Digital Synthesizer
DSP	Digital Signal Processing
DUC	Digital Up Convertor
ENOB	Effective Number of Bits
FPGA	Field Programmable Gate Array
IF	Intermediate Frequency
LFSR	Linear Feedback Shift Register
LPF	Low Pass Filter
NN	Nearest Neighbor
NRZ	Non-return to Zero
NI USRP	National Instrument Universal Software Radio Peripheral
PLL	Phase Lock Loop
PS	Phase Stabilizer
PSK	Phase Shift Keying
QPSK	Quadrature Phase Shift Keying
R&D	Research and Development
SDR	Software Defined Radio
SFDR	Spurious-Free Dynamic Range
SNR	Signal to Noise Ratio
SoC	System on Chip
SSG	Sinusoidal Signal Generator
VHDL	Very high speed integrated circuits Hardware Description Language
VI	Virtual Instrument

1. INTRODUCTION

Although the term "software defined radio" (SDR) was coined in 1995 by Stephen Blust, SDR systems have their origins in the defense sector since the late 1970s in both the U.S. and Europe [1]. To make the system reconfigurable, the ideal SDR system aims to reduce the analog components of the transceiver as much as possible and limit it to only the front-end parts like antenna, band-pass filter and low noise amplifier. In fact, SDR tends to migrate most of the involved signal processing tasks towards digital implementations. It enables the digital hardware to switch its functionality at run-time for different applications [2].

Nowadays, SDR refers to reconfigurable platforms that provide the high-performance infrastructure for realizing the rapidly expanding digital wireless communication. These systems not only must guarantee performance, but also due to the market, they must be adequately adaptable to follow the rapidly evolving standards and, as a viable solution, support multimode and multiband modes of operation. Despite some technical constraints due to the implementation aspects of SDR systems, their flexibility allows service providers an economical means of future development of these complicated and expensive systems [3]. A typical SDR system may perform many sophisticated signal processing tasks including channel estimation, equalization, forward error control, protocol management, carrier recovery and clock synchronization. Although there are many hardware platforms available to accomplish this wide variety of functions, the field programmable gate array (FPGA) is a reasonable solution to performing many of these tasks. In the early 1990s, FPGAs played a significant role in digital communication hardware, where they were utilized to provide state machines, bus interfacing and memory controlling [4].

To date, FPGA technology has undergone revolutionary changes. Particularly, the gate densities and clock speed of new generation FPGAs provide the communication system designers with a highly configurable logic framework that can be utilized for realizing advanced real-time signal processing functions. For instance, Xilinx Virtex-7 comprise nearly 2 million logic cells and 3600 Digital Signal Processing (DSP) slices, capable of operating up to 741 MHz [5]. In fact, the implementation of high-performance flexible systems has been made possible by advances in the emergence of FPGAs that has allowed the concept of SDR to become a reality. Thus, due to the key advantages of FPGAs like high-performance, reconfigurable features and low power and cost, they have found application in many areas, namely, Mobile systems (3G, 4G), Voice over IP, Multimedia and Radar systems.

On the other hand, Phase-Shift Keying (PSK) modulation is widely used in existing wireless technologies. For example, a current and important application of Quadrature PSK (QPSK) modulation is in standards like LTE & LTE-ADVANCE, IEEE 802.11b-1999, IEEE 802.11g-2003 and IEEE 802.15.4 [6]. Besides, in the past few years, binary phase-shift keying (BPSK) modulation has received a great deal of attraction in satellite communication and deep space network due to its simplicity, power efficiency and lower bit error rate [7]. Accordingly, a BPSK-QPSK demodulator is an inseparable component in many SDR systems.

Much of the current research on SDR systems pay particular attention to the implementation of digital PSK transceivers on FPGAs. Wen Wu [8] proposed a simplified DQPSK modulation and implemented it on an Altera Stratix II. In [9], S. O. Popescu and A. S. Gontine designed B-QPSK FGPA-based modulator and evaluated it on a Spartan 3E however; they have not addressed the effect of noise. In another survey, Nagaraj C. Shivaramaiah and coworkers [10] introduced a new method of time-multiplexing QPSK for the Global Navigation Satellite System (GNSS). They concluded that using an FPGA as the main processing core allows the use of a more general-purpose radio front end, which in turn enables the system to operate with a wider variety of settings. In [6], the authors used MATLAB / Simulink, ModelSim and Quartus II to implement a QPSK Modulator on Altera Cyclon IV. They presented a model-based approach and discussed its benefits over other methods. The authors in [27] and [28] implemented a similar system. However, they have not provided realistic performance analysis. Overall, these studies highlight the effectiveness of FPGA for a SDR-based PSK modem.

Of the many works that introduce FPGA-based PSK transceivers, few have combined the digital components with an actual radio, and most studies have only concentrated on baseband digital transceivers. Furthermore, far too little attention has been paid to the behavior of the implemented systems in the presence of carrier and clock frequency offset.

1.1. Research Objective

The design and implementation of a BPSK-QPSK transceiver, based on the concept of SDR is the specific objective of this thesis work together with evaluating the performance of the introduced model. The work differs from many previous studies by focusing on the actual implementation of an FPGA-based transceiver that utilizes a tunable RF transceiver and effectively resolves the symbol rate offset and carrier phase and frequency shift. Therefore, this study makes a major contribution to research on SDR systems by demonstrating a practical model.

In the beginning, the research work reviews some relatively general topics, i.e, theoretical understanding about PSK modulation and the ideal performance of BPSK and

QPSK modulations. The following chapters contain a detailed description of the design development and evaluation.

In the proposed model, the FPGA chip is a Xilinx Kintex 7 as the central processing core of NI USRP 2943R. In this design, the system components are adopted from Xilinx Core IP center, e.g. multipliers and Direct Digital Synthesizers (DDS). Additionally, a few application-specific blocks are constructed in Xilinx Very high speed integrated circuits Hardware Description Language (VHDL) design tool, Vivado and verified with ModelSim. The whole work is integrated into the LabVIEW FPGA module which employs Xilinx ISE compiler to generate the FPGA's configuration bit file. The LabVIEW module provides a convenient environment for assessing the design prior to the time-consuming task of synthesis, place and route.

1.2. Thesis Outline

The overall structure of the thesis takes the form of nine chapters. The second chapter begins by laying out the theoretical presentation of PSK modulation and looks at the mathematics behind it. The chapter also studies the Bit Error Rate (BER) estimation for BPSK and QPSK schemes in theory. Chapter 3 introduces NI USRP 2943 as the target platform. Chapter 4 and Chapter 5 explain the designed modulator and demodulator, respectively. The candidate carrier, clock and data recovery methods are also reviewed in Chapter 5. The remaining part of the work is as following:

Chapter 6 covers the system integration and also introduces the implemented Automatic Gain Controller (AGC) and presents the related simulation results.

Chapter 7 concerns the methodology used to evaluate the implemented transceiver and demonstrates the test results.

Chapter 8 concludes the work and suggests the future works.

The thesis also contains three appendixes including the VHDL descriptions and a view of the software interface that controls the modem.

2. PRINCIPLE OF PSK MODULATION

In the last years, due to the fast development of modern digital communication techniques, the demand for robust high data rate transmission has increased significantly. In fact, Digital Modulation is less complex, more secure and more efficient in long distance transmission. Besides, the noise detection and correction in a digital system are more effective than its analog counterpart [11]. In digital modulation, symbols are transmitted in the form of baseband pulses which modulate a high-frequency carrier [12]. There are three primary guidelines for choosing the most appropriate modulation scheme for a certain application [13]:

1) Power efficiency: reliable sending of data with minimal power requirements.
2) Bandwidth efficiency: the ability of the system to accommodate data within a prescribed bandwidth.
3) System complexity: the amount of circuits involved and the technical difficulty of the system, associated with the cost.

Phase Shift Keying (PSK) is one of the most efficient digital modulation techniques and is widely used in modern communication systems like satellite links and wideband microwave radio relay systems. In this scheme, the digital data is encoded in the phase property of a carrier signal [13]. PSK uses a finite number of phases, each assigned a unique pattern of binary digits. In other words, each pattern of bits forms the symbol that is represented by the particular phase. In order to reconstruct the original data, the demodulator determines the phase of the received signal and maps it back to the symbol it represents. The constellation points in this scheme are regularly positioned with uniform angular spacing. It provides maximum phase-separation between adjacent points and thus the best resistance to corruption. Since the data to be conveyed is normally in binary form, the PSK scheme is usually composed of the number of constellation points being a power of 2. Two common examples of PSK are BPSK and QPSK which respectively use two and four phases.

2.1. Binary Phase-Shift Keying

In BPSK, two chosen antipodal signals $s_1(t)$ and $s_2(t)$ in equation (1) represent the digital symbols. They are in the same frequency and carry the same amount of energy whereas they have a correlation coefficient of -1. This remarkable property provides the minimum error probability in BPSK modulation scheme.

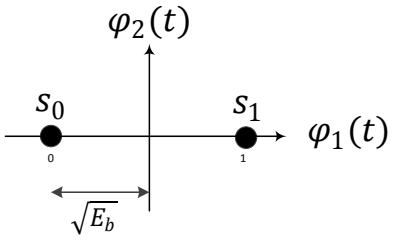

Figure 1. *Constellation of BPSK.*

for $0 \leq t \leq T_b$:

$$s_1(t) = A\cos(2\pi f_c t)\,, \text{Symbol} = 1 \tag{1}$$

$$s_2(t) = -A\cos(2\pi f_c t)\,, \text{Symbol} = 0$$

$$A = \sqrt{\frac{2E_b}{T_b}}$$

where E_b and T_b are, respectively, the energy per bit and the bit duration.

Figure 1 shows a linear combination of orthonormal basis functions $\varphi_1(t)$ and $\varphi_2(t)$ in equation (2). In this figure, $s_1(t)$ and $s_2(t)$ are graphically represented by two points and each antipodal signal has a finite energy of bit E_b given in equation (3) [13].

$$\varphi_1(t) = \sqrt{\frac{2}{T_b}}\cos(2\pi f_c t) \qquad \varphi_2(t) = -\sqrt{\frac{2}{T_b}}\sin(2\pi f_c t) \tag{2}$$

$$E_b = \frac{A^2 T_b}{2} \tag{3}$$

Equation (4) reveals how the modulator makes the bipolar data stream $a(t)$ from the binary data a_k. In this equation, $a_k \in \{+1, -1\}$, $p(t)$ is the rectangular pulse with unit amplitude defined on $[0, T_s]$ and T_s is symbol duration which is equal to the bit duration T_b.

$$a(t) = \sum_{k=-\infty}^{k=\infty} a_k\, p(t - kT_s) \tag{4}$$

Then, $a(t)$ is multiplied with a sinusoidal carrier $Aa(t)\cos(2\pi f_c t)$ to generate the BPSK signal $s(t)$ defined in equation (5).

$$s(t) = Aa(t)\cos(2\pi f_c t) \qquad -\infty < t < \infty \tag{5}$$

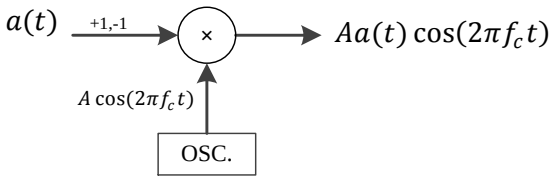

Figure 2. *BPSK Modulator.*

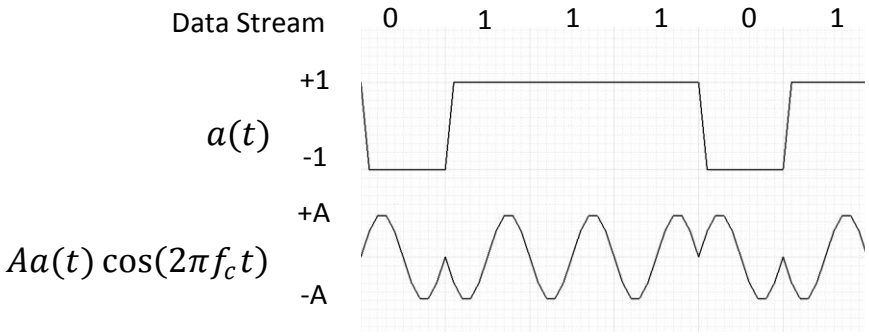

Figure 3. *BPSK Waveform.*

As shown in Figure 2, a typical BPSK modulator has a simple structure including a local oscillator and a multiplier. Figure 3 depicts the generated waveform for an instance data stream; {011101}.

To insure minimum bit error probability, f_c should be an integer multiple of the symbol rate R_s;

$$f_c = \quad m \times R_s \text{ , where m is an integer} \tag{6}$$

However, if $f_c \gg R_s$, this condition can be relaxed and the resultant performance degradation is negligible. In general, the phase is not continuous at the bit boundaries which means the bit timing is not necessarily synchronous with the carrier. Figure 4 presents coherent BPSK demodulator applying a correlator where the reference signal is the scaled-down version of the difference signal. The reference signal must be synchronous to the received signal $r(t)$ in frequency and phase [13]. In practice, carrier recovery block generates the reference signal.

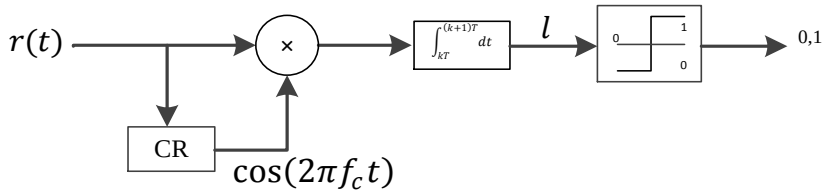

Figure 4. *BPSK Demodulator*

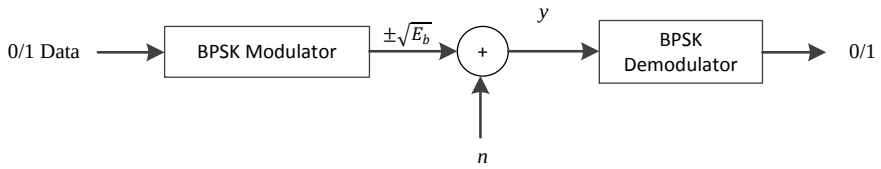

Figure 5. *BPSK Transceiver.*

In the absence of noise and setting $A = 1$, the output of the correlator at $t = (k+1)T_s$ is[15]:

$$l = \int_{kT_s}^{(k+1)T_s} r(t)\cos(2\pi f_c\, t)dt = \int_{kT_s}^{(k+1)T_s} a_k \cos^2(2\pi f_c\, t)dt \tag{7}$$

$$= \frac{T_s}{2}a_k + \frac{a_k}{8\pi f_c}[\sin(4\pi f_c(k+1)T_s) - \sin(4\pi f_c kT_s)]$$

The second term would be zero if R_s satisfies the condition in (6), thus the original signal $a(t)$ is perfectly reconstructed. Besides, as long as $f_c \gg R_s$, the second term is much smaller than the first and hence, its influence is negligible.

Figure 5 depicts the simplified block diagram of a BPSK transceiver. The binary digits 1 and 0 are represented through the analog levels $+\sqrt{E_b}$ and $-\sqrt{E_b}$, respectively. In the presence of noise, the probability of error is a function of Signal to Noise Ratio (SNR).

In order to estimate this probability, consider n as the additive white Gaussian noise:

$$n = \frac{1}{\sqrt{2\pi\sigma^2}}\exp\left(\frac{-(x-\mu)^2}{2\sigma^2}\right) \text{ where } \mu = 0 \text{ and } \sigma^2 = \frac{N_0}{2} \tag{8}$$

In Figure 5, y represents the noisy signal as it is given by equation (9).

$$y = \begin{cases} s_0(t) + n & \text{for } bit = 0 \\ s_1(t) + n & \text{for } bit = 1 \end{cases} \tag{9}$$

Equation (10) demonstrates the conditional probability distribution function (PDF) of y for the two cases in equation (9):

$$P(y|s_0) = \frac{1}{\sqrt{\pi N_0}} \exp\left(\frac{-\left(y+\sqrt{E_b}\right)^2}{N_0}\right), P(y|s_1) = \frac{1}{\sqrt{\pi N_0}} \exp\left(\frac{-\left(y-\sqrt{E_b}\right)^2}{N_0}\right) \quad (10)$$

Assuming that s_0 and s_1 are equally probable i.e. $\mathrm{P}(S_1) = P(S_2) = \frac{1}{2}$, the threshold at zero forms the optimal decision boundary;

$$y > 0 \rightarrow s_1 \text{ and } y \leq 0 \rightarrow s_0 \quad (11)$$

With accordance to the threshold, the probabilities of error given"s_i is transmitted" are [15]:

$$P(e|s_0) = \frac{1}{\sqrt{\pi N_0}} \int_0^{\infty} \exp\left(\frac{-\left(y+\sqrt{E_b}\right)^2}{N_0}\right) dy = \frac{1}{2} erfc\left(\sqrt{\frac{E_b}{N_0}}\right) \quad (12)$$

$$P(e|s_1) = \frac{1}{\sqrt{\pi N_0}} \int_{-\infty}^{0} \exp\left(\frac{-\left(y-\sqrt{E_b}\right)^2}{N_0}\right) dy = \frac{1}{2} erfc\left(\sqrt{\frac{E_b}{N_0}}\right)$$

where $erfc(x)$ is complementary error function as written in (13).

$$erfc(x) = \frac{2}{\sqrt{\pi}} \int_x^{\infty} \exp(x^2)\, dx \quad (13)$$

Figure 6 presents an illustrative view of error probabilities in red and blue regions. In this figure, the blue area is equivalent to $P(e|s_0)$, while the red region determines the conditional error probability of $P(e|s_1)$. According to the equations (11) - (13), the total probability of error or the BER for BPSK can be written as [15]:

$$P_b = \sum_{i=0}^{1} P(s_i) P(e|\, s_i) = \frac{1}{2} erfc\left(\sqrt{\frac{E_b}{N_0}}\right) \quad (14)$$

Figure 7 portrays the BER curve for BPSK modulation. This plot helps the evaluation of the implemented demodulator in Chapter 8.

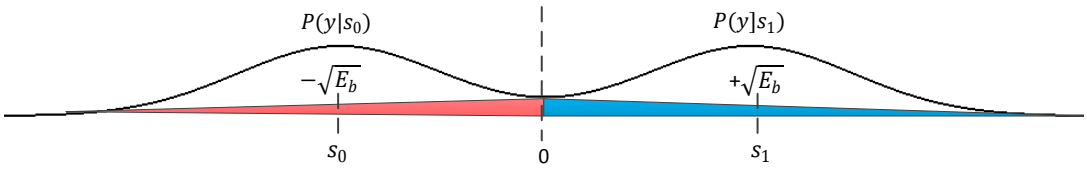

Figure 6. *Conditional probability density functions, BPSK modulation; the blue and red regions are equivalent to $P(e|s_0)$ and $P(e|s_1)$, respectively.*

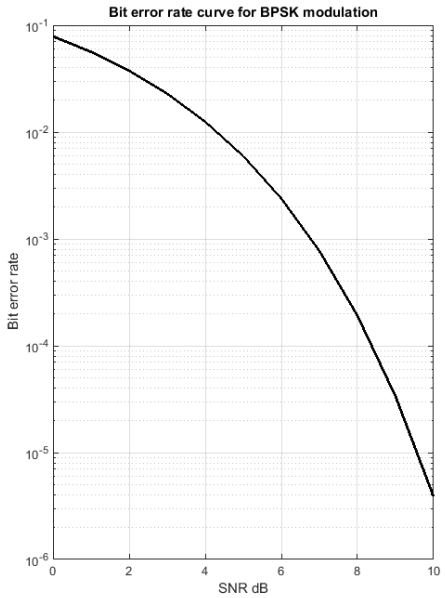

Figure 7. *Bit Error Rate Curve for BPSK Modulation.*

2.2. Quadrature Phase-Shift Keying

Among all schemes in PSK family, QPSK is the most often used scheme since it does not suffer from the BER degradation while the bandwidth is increased. In fact, other PSK schemes increase bandwidth efficiency at the expenses of the BER performance [13]. Equation (15) presents corresponding s_i signals for this modulation.

$$S_i(t) = Acos(2f_c t + \theta_i), 0 \le t \le T_s,\ \theta_i = \frac{(2i-1)}{4}\pi,\ i = 1,2,3,4 \tag{15}$$

$$A = \sqrt{\frac{2\,E_s}{T_s}},\ E_s = 2\,E_b, T_s = 2T_b$$

where, E_s is the energy per symbol and T_s is the symbol duration.

The initial signal phases are $\frac{k\pi}{4}$; $k = 1, 2, 3, 4$. If the carrier frequency is chosen as an integer multiple of the symbol rate, in any symbol interval $[kT_s, (k+1)T_s]$ the signal initial phase is also one of the four phases.

To achieve a linear combination of $\varphi_1(t)$ and $\varphi_2(t)$, the former expression of S_i can be rewritten as equation (16);

$$S_i(t) = Acos(\theta_i)\cos(2\pi f_c t) - Asin(\theta_i)\sin(2\pi f_c t) = s_{i1}\varphi_1(t) + s_{i2}\varphi_2(t) \tag{16}$$

where s_{i1} and s_{i2} are given in (17);

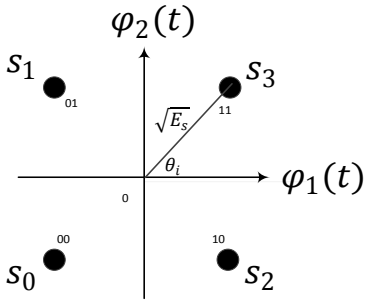

Figure 8. *QPSK Signal Constellation*

$$s_{i1} = \sqrt{E_s}cos(\theta_i),\ s_{i2} = \sqrt{E_s}\sin(\theta_i) \tag{17}$$

In this equation, $E_s = \frac{T_s A^2}{2}$ is the symbol energy and $\varphi_1(t)$ and $\varphi_2(t)$ are given in equation (2). On this linear coordinate, the constellation points are represented by four vectors so that the angle of vector S_i with respect to the horizontal axis is the signal initial phase θ_i .

As illustrated in Figure 8, in the QPSK scheme, data bits are divided into groups of two bits, called *Dibits*. The assignment of the *Dibits* to the signals could be arbitrary as long as the mapping is one to one. The coordinates of the QPSK symbol points are tabulated in Table 1.

Table 1. *QPSK Signal Coordinates*

Symbol: Dibits	$\boldsymbol{\theta_i}$	$\boldsymbol{s_{i1}}$	$\boldsymbol{s_{i2}}$
3 : 11	$\pi/4$	$+\sqrt{E_s/2}$	$+\sqrt{E_s/2}$
1 : 01	$3\pi/4$	$-\sqrt{E_s/2}$	$+\sqrt{E_s/2}$
0 : 00	$-3\pi/4$	$-\sqrt{E_s/2}$	$-\sqrt{E_s/2}$
2 : 10	$-\pi/4$	$+\sqrt{E_s/2}$	$-\sqrt{E_s/2}$

In this table, each point in the QPSK constellation refers to a certain digital symbol and for the convenience of modulator structure, logic 1 and 0 are mapped to $+\sqrt{E_s/2}$ and $-\sqrt{E_s/2}$ respectively.

According to the data in Table 1, odd-numbered bits are mapped to s_{i1} and even-numbered bits to s_{i2}. Therefore, the QPSK signal in equation (16) can be simplified as:

$$S(t) = \frac{A}{\sqrt{2}}I(t)cos(2\pi f_c t) - \frac{A}{\sqrt{2}}Q(t)sin(2\pi f_c t),\ \ -\infty < t < \infty \tag{18}$$

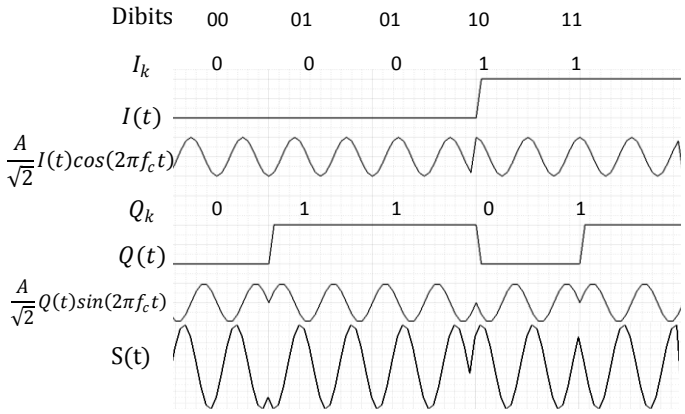

Figure 9. *QPSK Waveform.*

In this equation, $I(t)$ and $Q(t)$ are pulse trains determined by the odd- and even-numbered bits:

$$I(t) = \sum_{k=-\infty}^{\infty} I_k p(t - kT_s), \quad Q(t) = \sum_{k=-\infty}^{\infty} Q_k p(t - kT_s) \tag{19}$$

where, $I_k = \pm 1$, $Q_k = \pm 1$ and $p(t)$ is a rectangular pulse function defined on [0,T_s]. The QPSK waveform using the signal assignment in the above equations is shown in Figure 9.

Figure 10 depicts the structure of QPSK modulator. In a similar way, the QPSK waveform has a constant envelope and discontinuous phases at symbol boundaries. If the transmission rate of the symbols is the same in QPSK and BPSK, it is intuitively clear that QPSK transmits data twice as fast as BPSK. Moreover, it is evident that the distance between the adjacent points of QPSK constellation is shorter than that of the BPSK. Consequently, in equal conditions, this modulation is more vulnerable to noise than BPSK.

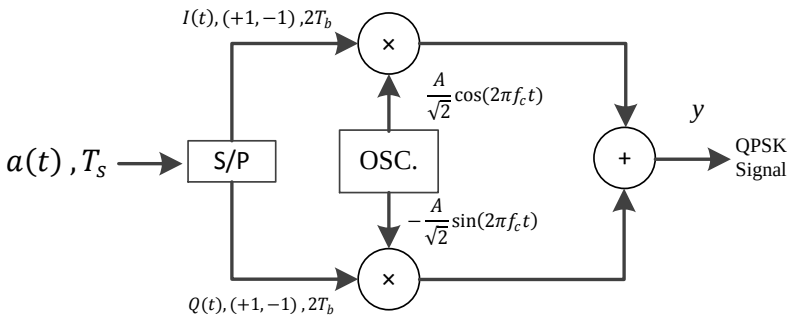

Figure 10. *QPSK Modulation.*

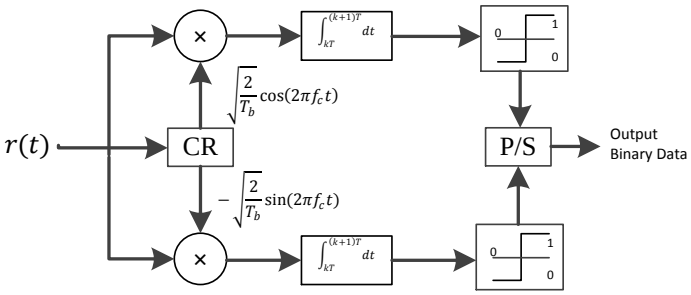

Figure 11. *QPSK Demodulator.*

Figure 11 shows a QPSK demodulator which comprises two individual BPSK demodulators in a complex channel. In this model, the parallel-to-serial converter (P/S) is applied to combine two bit sequences into a single sequence of symbols.

In a noisy channel, the conditional PDF of y given ‘‘$S_2 = (s_{21}, s_{22})$ was transmitted’’ is defined in equation (20).

$$P(y|S_2)) = \frac{1}{\sqrt{\pi N_0}} \exp\left(\frac{-(y+\sqrt{E_s})^2}{N_0}\right) \tag{20}$$

where the scaling factor of $\sqrt{E_s/2}$ normalizes the average energy of the transmitted symbols to 1, assuming that all the constellation points are equally likely and y is:

$$y = \begin{cases} s_{i1}(t) + n \\ s_{i2}(t) + n \end{cases} \qquad \text{for Dibits} : 00, 01, 10, 11 \tag{21}$$

As can be seen in Figure 12, the symbol $s_2 = (s_{21}, s_{22})$ is decoded correctly only if y falls in the first region of the coordinates:

$$P(c|s_2) = P(R_y > 0|s_2)P(Q_y > 0|s_2) \tag{22}$$

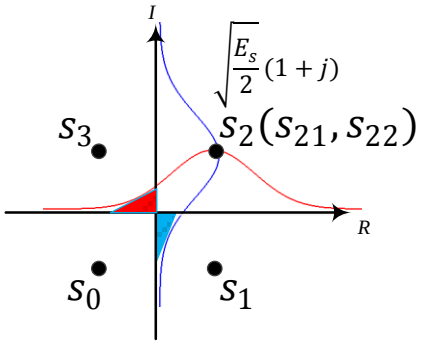

Figure 12. *Conditional probability density functions for QPSK modulation. The red and blue regions are equivalent to $P(e|s_{21})$ and $P(e|s_{22})$, respectively.*

On the other hand, the probability of real and imaginary components of $y > 0$, given ''s_2 was transmitted'' can be written as [15]:

$$P(R_y > 0|s_2) = 1 - \frac{1}{\sqrt{\pi N_0}} \int_{-\infty}^{0} \exp\left(\frac{-\left(R_y - \sqrt{\frac{E_s}{2}}\right)^2}{N_0}\right) dy = 1 - \frac{1}{2}\, erfc\left(\sqrt{\frac{E_s}{2N_0}}\right) \quad (23)$$

$$P(Q_y > 0|s_2) = 1 - \frac{1}{\sqrt{\pi N_0}} \int_{-\infty}^{0} \exp\left(\frac{-\left(Q_y - \sqrt{\frac{E_s}{2}}\right)^2}{N_0}\right) dy = 1 - \frac{1}{2}\, erfc\left(\sqrt{\frac{E_s}{2N_0}}\right)$$

Thus, with respect to equations (22) and (23) the probability of s_2 being decoded correctly is;

$$P(c|s_2) = \left[1 - \frac{1}{2}\, erfc\left(\sqrt{\frac{E_s}{2N_0}}\right)\right]^2 = 1 - erfc\left(\sqrt{\frac{E_s}{2N_0}}\right) + \frac{1}{4}\, erfc^2\left(\sqrt{\frac{E_s}{2N_0}}\right) \quad (24)$$

The symbol will be in error, if at least one of the symbols is decoded incorrectly. Equation (25) gives the probability of symbol error.

$$P_s = 1 - p(c|s_2) = erfc\left(\sqrt{\frac{E_s}{2N_0}}\right) - \frac{1}{4}\, erfc^2\left(\sqrt{\frac{E_s}{2N_0}}\right) \quad (25)$$

For higher values of E_s/N_0, the second term in this equation becomes negligible and the probability of error can be approximated as following[15]:

$$P_b = \frac{1}{2} P_s \cong \frac{1}{2} erfc\left(\sqrt{\frac{E_s}{2N_0}}\right) = \frac{1}{2} erfc\left(\sqrt{\frac{E_b}{N_0}}\right) \quad (26)$$

As a result, the probability of bit-error for QPSK is the same as for BPSK. However, to achieve the same bit-error probability, QPSK uses twice the power (since two bits are transmitted simultaneously). Otherwise, with the same power for each symbol in QPSK and BPSK, the BER follows the given curves in Figure 13.

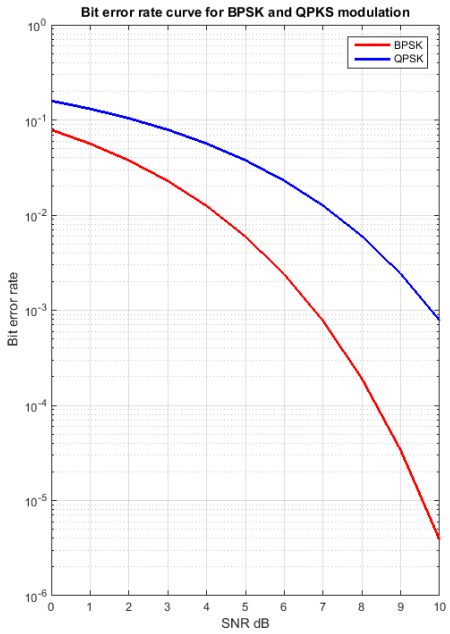

Figure 13. *Bit Error Rate Curve for BPSK (red) and QPSK (blue) Modulations.*

3. NI USPR2943 AS THE TARGET PLATFORM

As mentioned in the introduction, the transceiver is developed on an NI USRP2943 [18] device designed to form an integrated environment with LabVIEW FPGA module. It provides a powerful tool for FPGA-based system developers. Although the software suffers from a number of limitations, it offers a wide variety set of tools. Figure 14 illustrates how LabVIEW makes use of the Xilinx ISE compiler to generate an appropriate programming file for the FPGA.

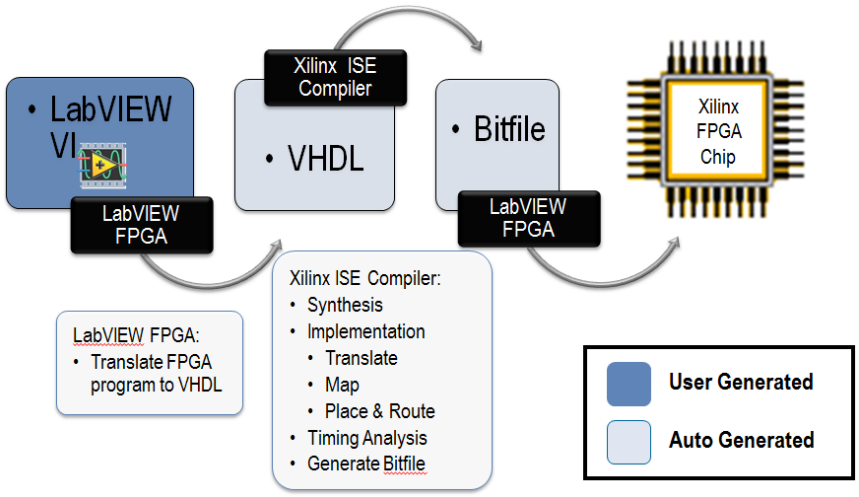

Figure 14. *LabVIEW; steps to generating a FPGA Bitfile. [17]*

As can be seen from the figure above, the software allows performing a desired system on the FPGA. The key idea behind LabVIEW FPGA module is to obtain a working FPGA-based model, regardless of integrating and interfacing difficulties.

The NI USRP2943 device (see Figure 15) can be connected to a computer via Peripheral Component Interconnect (PCI) Express bus. It allows the computer to control the Radio Frequency (RF) block and either read or write FPGA registers and FIFOs. Besides, the software offers other components to monitor and analyze the captured data. The components can be employed in a host Virtual Instrument (VI), operating on the computer; the host VI firstly loads the Bitfile into the FPGA to start operation and then, frequently send commands to the device or read its memories to monitor data.

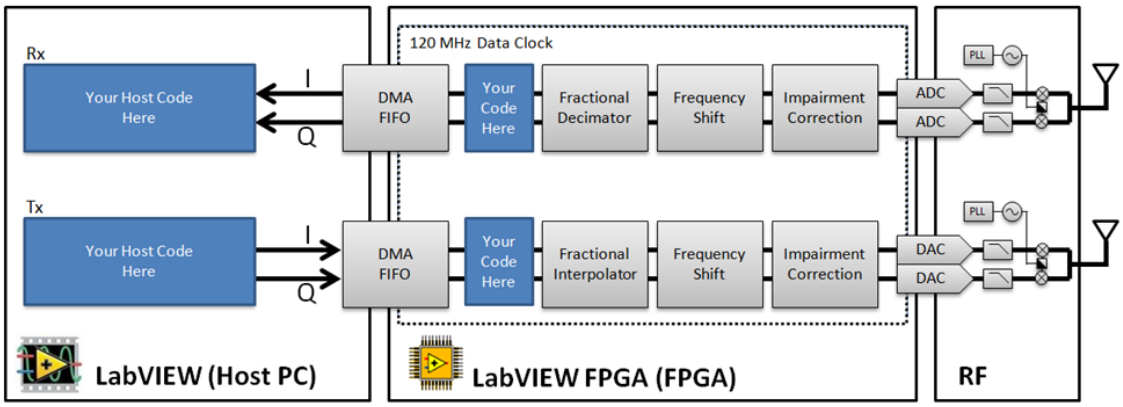

Figure 15. *Block Diagram of NI USRP2943.[17]*

For evaluating the introduced system in this study, the receiver and transmitter are implemented in one NI USRP device. However, in order to simulate non-ideal condition, practical constraints such as the noise and the frequency offsets are taken into account. Figure 16 shows how channel 1 and channel 2 are considered to simulate a practical radio link. Moreover, in the "All Digital Transceiver" testbed, the RF block is bypassed and the modulator sends the signal directly to the demodulator, all inside the large reconfigurable Xilinx Kintex-7 FPGA. In fact, this highly efficient and affordable FPGA enables designers to address connectivity and throughput requirements while minimizing part counts. The following table shows the maximum capability of the Kintex-7 FPGA.

Table 2. *Xilinx Kintex-7 Maximum Capability[18].*

Logic Cells	478 K	**Peak Serial Bandwidth (Full Duplex)**	800 Gb/s
Block RAM	34 Mb	**Memory Interface**	1,866 Mb/s
DSP Slices	1,920	**I/O Pins**	500
Transceivers	32	**I/O Voltage**	1.2V, 1.35V, 1.5V, 1.8V, 2.5V, 3.3V
Peak Transceiver Speed	12.5 Gb/s	**Package Options**	Low-Cost, Lidless Flip-Chip and High-Performance Flip-Chip

Figure 16. *NI USRP2943, channel 1: transmitter, channel 2: receiver.*

4. BPSK-QPSK MODULATOR

Chapter 2 presented the concept of BPSK-QPSK modulations in a nutshell. This chapter describes the structure of the modulator and concentrates on its implementation dimensions in detail. In this modulator design three aims are mainly taken into account:

1) Supporting both BPSK and QPSK modulations.
2) Generating deterministic data.
3) Precise symbol rate adjustment.

The third item allows the systematic assessment of the performance of the demodulator. Figure 17 depicts an overview of the designed modulator. In this schematic, the generated polar Non-Return to Zero (NRZ) signal passes through a Nearest Neighbor (NN) resampler and then an FIR filter bank is applied to shape the pulse train with respect to the desired data rate. The shaped signal either feeds a Digital Up Convertor (DUC) in the radio module or is up-converted by a Direct Digital Synthesizer (DDS). The internal up-conversion allows the modulator to be as a full digital transmitter in an "All Digital Transceiver" testbed which is a classical solution for evaluation purposes regardless of analog-related circuits. However, in combination with a radio, the DDS is not required and can be bypassed in FPGA synthesis; hence it does not consume any logic resource. It should be noted that, in order to save more logic resources in all designed blocks, the arithmetic (signed) left/right shift operator is applied to amplify or attenuate the signal. Furthermore, to obtain maximum compatibility with Xilinx FPGAs, the DDSs, Multipliers and Filters are adopted from Xilinx IP Core library.

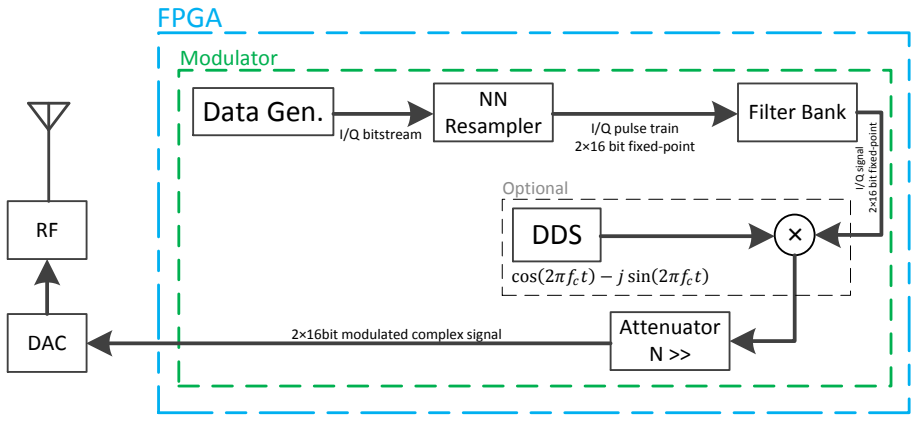

Figure 17. *Block Diagram of The Implemented Modulator.*

4.1. Data Generator

Traditionally, the performance of a radio link has been assessed by measuring BER. In this approach, the receiver approximates correlation between the decoded symbols and reference data. Consequently, the origin of transferred data should be known to the receiver. To achieve this ability, the introduced Data Generator contains a pair of 9-bits Linear Feedback Shift Registers (LFSR) which generates a deterministic bit-stream. In its elementary form, an LFSR is a shift register whose input bit is driven by the XOR of some bits of the overall shift register value. The initial value of the LFSR is called the seed and the rightmost bit of the LFSR is the output bit. The operation of LFSR is deterministic because the stream of values produced by the register is completely determined through its current or previous state. Likewise, because the register has a finite number of possible states, it must eventually enter a repeating cycle [19]. From this point of view, LFSR is a reasonable option that facilitates BER measurement.

Figure 18 illustrates the structure of the performed LFSR and its implementation in VHDL design is included in Appendix 1. The block generates a 1023-bit data stream successively. Although the reset capability is not usually applicable to ordinary systems, it plays a key role in more advanced applications. Therefore, to meet the required synchronizing ability in these applications, the implemented LFSR holds certain input gates for the reset, seed and external clock.

Figure 19 represents the block diagram of the implemented data generator. In this model, certain seed values initialize the LFSRs and form I/Q data streams. It should be pointed out that the Q channel is disabled in the BPSK operating mode.

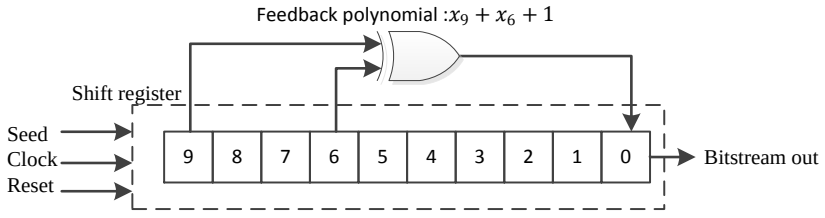

Figure 18. *Implemented 10-bit LFSR.*

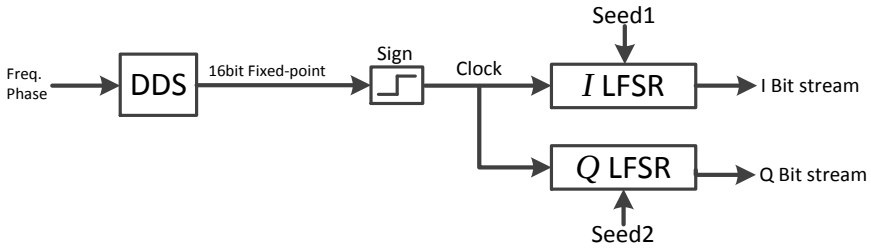

Figure 19. *Implemented Data Generator Module.*

As shown in Figure 19, this block is equipped with a DDS as the clock source, providing accurate symbol rate adjustment. The DDS can even apply a precise amount of time delay to the output data stream. In addition to the aperiodic autocorrelation characteristic of the generated 1023-bit pseudo-random data by LFSRs and the reset pin, this advantage makes the implemented modulator a potential alternative in the array-based communication systems. Furthermore, from this thesis viewpoint, the DDS facilitates the assessment of the implemented demodulator because it can impose a precise amount of shift to the symbol rate.

4.2. Shaping Filter

Due to the sharp transitions at the pulse edges, unfiltered rectangular bit pulses have theoretically infinite bandwidth. Figure 20 reveals how the spectral side lobes continue to infinite frequency, however, the power decreases gradually. Thus, the baseband information signal must be filtered to limit the bandwidth. It is also called 'pulse shaping' since the filter smooths out the pulses to achieve a signal with more desirable spectral properties than a rectangular pulse. Raised-cosine filter is a popular filter for this purpose because it can minimize Inter-Symbol Interference (ISI) [20].

As can be seen in Figure 20, the shaping filter has preserved the main spectral lobe and in contrast, suppressed the side-lobes to reduce the Occupied Bandwidth (OBW) of the transmitting signal. Equation (27) defines the bandwidth of the filtered signal.

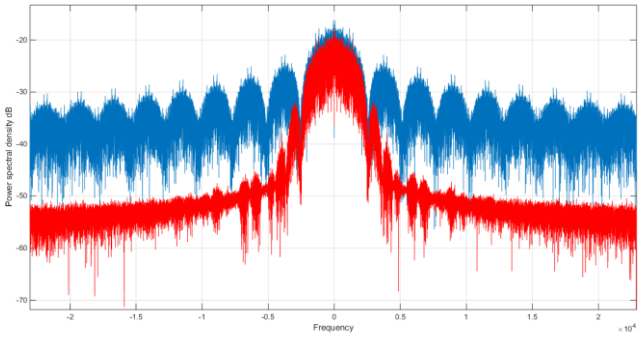

Figure 20. *Power Spectrum of Unshaped (blue) and Shaped (red) Pulse Train.*

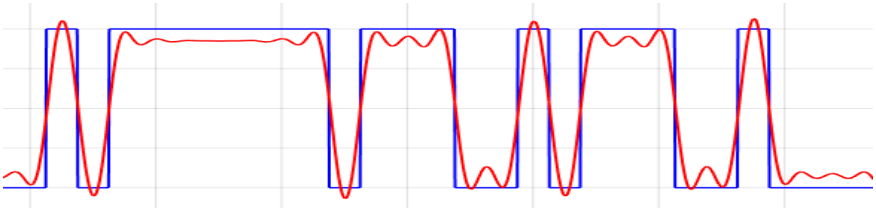

Figure 21. *Baseband Pulse Shaping. Rectangular Pulse Train (blue), Raised-Cosine Shaped Pulse (red).*

$$BW = f_{symbol}(1 + a) \quad (27)$$

where f_{symbol} is the symbol rate and a is the roll-off factor of the raised-cosine filter.

Preferably, the roll-off factor should be as small as possible. However, a too small factor yields a signal from which it is difficult to recover symbol timing information. This is due to the following facts. First, the most of symbol timing recovery techniques rely on abrupt transitions in the baseband signal in order to obtain timing synchronization. Second, such a small value of the roll-off factor makes the symbol transitions so smooth that they are impossible to identify. Reasonable filter roll-off factors for this purpose are within 0.25 to 0.5 [20]. The selected factor is 0.25 in the implemented raised-cosine shaping filter.

Figure 21 shows the rectangular pulse and the filtered signal. As illustrated in this plot, the transition edges are smoother in the filtered pulse and thereby, its bandwidth is less than that of the rectangular pulse.

4.3. The Implemented Modulator In LabVIEW

Figure 22 presents the implemented modulator in LabVIEW. As mentioned earlier, it is in the VI-form, including many other sub-VIs. A 120 MHz Data clock drives the whole block. In this thesis work, all VHDL designs are imported utilizing “IP Integration” component from LabVIEW FPGA library.

Figure 22 shows that the data generator produces a deterministic impulse-like pseudo-random sequence and the DDS inside this block regulates the symbol rate. As can be seen in this structure, with the same DDS frequency, the bit rate is two times faster in the QPSK operating mode.

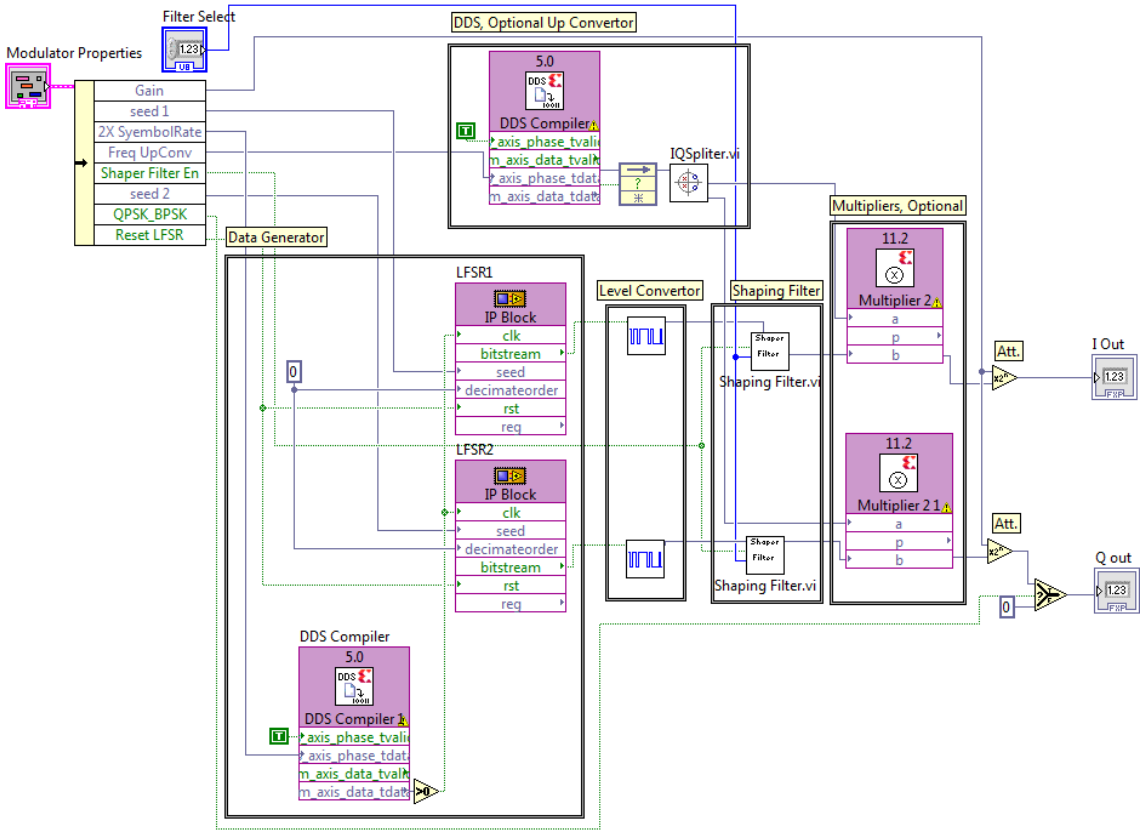

Figure 22. *The Implemented BPSK-QPSK Modulator in LabVIEW.*

The Level Convertor transforms unipolar impulse data to bipolar signals $I(t)$ and $Q(t)$ in (19). To minimize OBW, both parts of complex data pass through the shaping filter bank. In order to up-convert the filtered data, it could either be sent to the RF block or be multiplied by a sinusoidal wave. For evaluation purposes, the full-range complex output is attenuated by the signed shift operation on the output ports.

Figure 23 illustrates the baseband and the modulated signals in BPSK "All Digital Transceiver" testbed. This plot reveals how the shaping filter smooths a pulse train and influences the modulated signal.

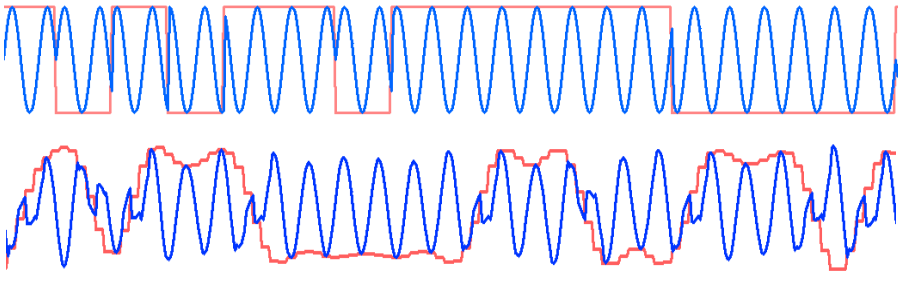

Figure 23. *The Baseband Data (red) and Modulated Signal (blue). Unshaped (top) and Shaped Data (bottom). BPSK Modulation. Full-range, 16 bits fixed-point.*

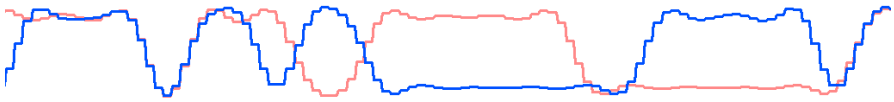

Figure 24. *The Baseband Shaped Complex Data. QPSK Modulation. Full-range, 16-bits Fixed-point.*

The shaped QPSK signal is also depicted in Figure 24. The signal is ready for up-conversion in the RF module.

It should be noted that, in this implementation, the data is represented by a 16-bit two's complement in fixed-point format. Therefore, the possible dynamic range is [-32768, +32767].

5. BPSK-QPSK DEMODULATOR

Chapter 5 is dedicated to propose the block diagram of the demodulator. This block could be either fed by the RF module or an internal Digital Down-Convertor (DDC). As shown in Figure 25, the DDC down-converts the modulated signal inside the FPGA and, the designer can bypass it in the radio-based testbed. In the proposed demodulator, the complex baseband data passes through three sub-components; the Carrier Recovery (CR) block compensates the carrier frequency offset. The Phase Stabilizer (PS) block enhances the signal and eliminates the stationary residual carrier phase offset and finally the Clock and Data Recovery (CDR) extracts the complex bitstream to establish symbols. Additionally, it generates a synchronizing trigger signal for the PCI-e interface to capture symbols accurately, with accordance to the recovered clock. It should be noted that the modulation type (BPSK / QPSK), the symbol rate, the filter bandwidth and other system parameters are adjustable in any operating state and no initialization is required.

In the following sections, the structure of each block is explained. Besides, in order to make a better perspective, the waveforms of critical signals are presented.

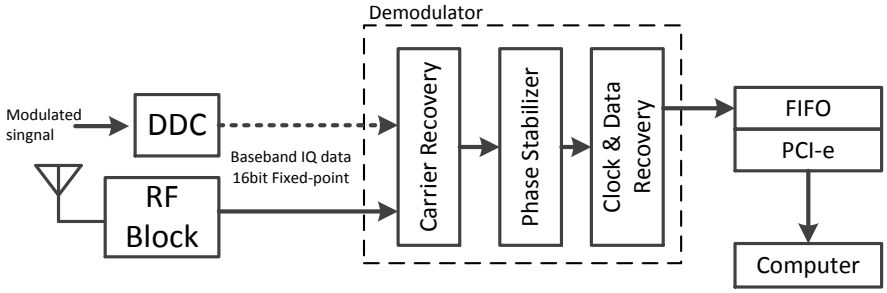

Figure 25. *The Integrated Demodulator in NI USRP2943*

5.1. Digital Down-Convertor

The DDC converts a real signal at Intermediate Frequency (IF) to a complex baseband signal and typically consists of four sub-blocks: DDS, Multipliers, Low-Pass Filters (LPF) and down-sampler.

As depicted in Figure 26, the DDS generates a complex sinusoid at the intermediate frequency f_{IF}. The multipliers create the sum $f_{IF} + f_c$ and difference $f_{IF} - f_c$ signals. In order to extract the baseband data, f_{IF} is selected to be as close as possible to f_c. Then, the LPFs preserve the difference term while rejecting the sum image.

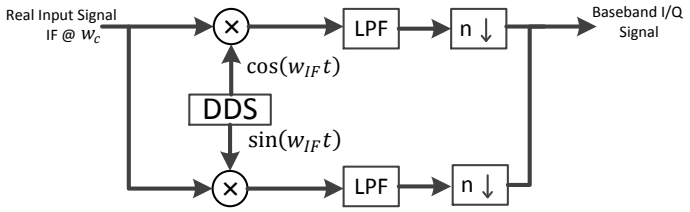

Figure 26. *Digital Down-Convertor.*

This term comprises two signals; a complex baseband representation of the original desired data and the residual sinusoidal signal at $\Delta f = f_{IF} - f_c$. In addition to down-conversion, the DDC decimates the input signal to a lower sample rate, allowing follow-on signal processing by lower speed [21].

Figure 27 presents the implemented DDC in LabVIEW as an aggregation of Xilinx IP cores. The down-conversion is accomplished by filters, and the DDC frequency is finely adjustable, providing operation in a wide frequency range. The DDC receives real input data as 16-bits two's complement fixed-point samples and generates baseband I/Q data in the same format.

Figure 28 illustrates how Δf rises in the down-converted signal. As depicted in this figure, for a QPSK input signal, the output is purely the desired pulse trains when the offset frequency Δf is zero. In contrast, non-zero offset appears as a sinusoid, enveloping the complex baseband data.

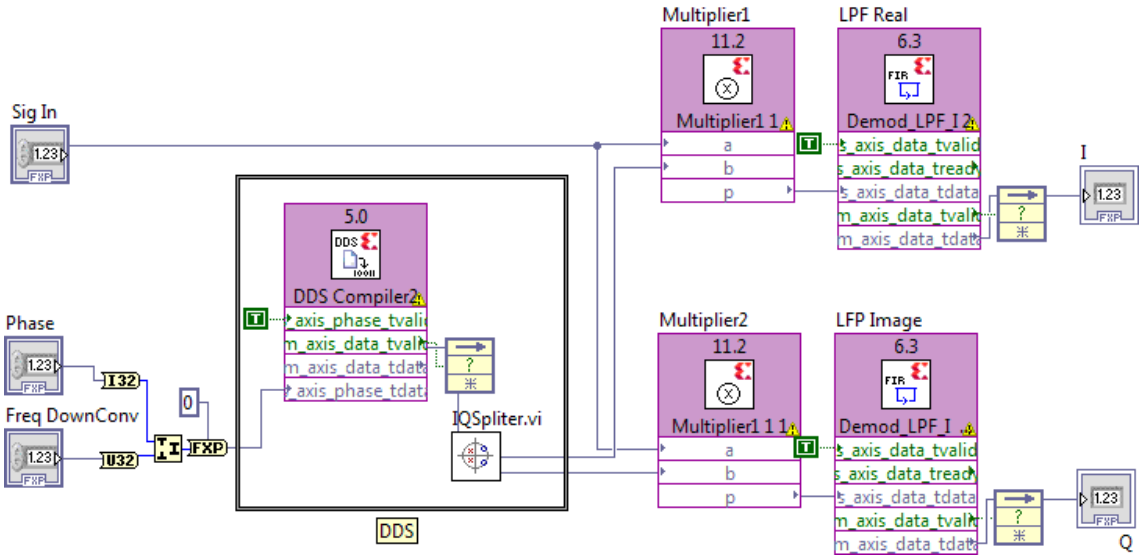

Figure 27. *Implemented DDC in LabVIEW.*

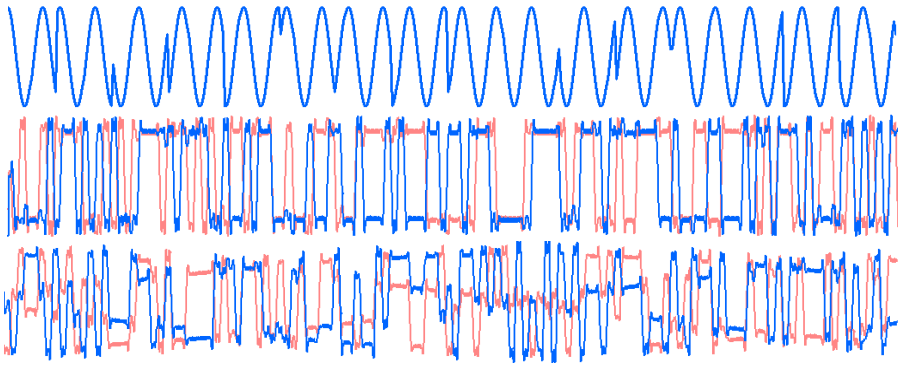

Figure 28. *Top: QPSK signal. Middle: baseband signal where* $\Delta f = 0$. *Bottom: baseband signal where* $\Delta f \neq 0$.

Similarly, the frequency offset influences the recovered symbols; it rotates the constellation counterclockwise or clockwise depending on its sign. For instance, with a frequency offset of 1Hz, the constellation turns once in a second. Thus, it reveals the important role of the CR block which is mainly discussed in the next section.

5.2. Carrier Recovery

In practice, the transmitter and receiver rarely use synchronized oscillators. Instead, they have independent oscillators with phase and frequency offsets and instabilities. Additionally, in mobile radio communications, systems may also suffer from the frequency difference due to Doppler phenomenon, where the receiver is in motion relative to the transmitter. The difference frequency Δf causes the rotation of the received symbol constellation (see Figure 29) and hence, the symbols would be unrecognizable to the demodulator. Basically, the CR block estimates the frequency difference between a received signal's carrier wave and the receiver's local oscillator to allow perfect demodulation.

In this thesis work, two closed-loop CR approaches were considered as a potential option to be implemented on FPGA; squaring loop and Costas loop. Both methods are based on the concept of Phase-Locked Loop (PLL) described in the following section.

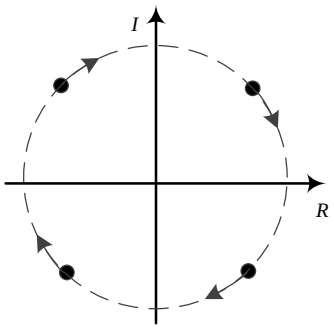

Figure 29. *The Effect of Carrier Frequency Offset on Constellation.*

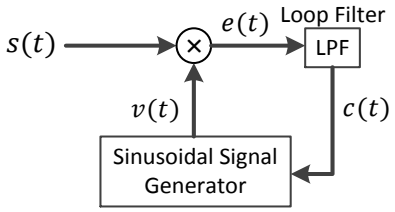

Figure 30. *Typical PLL.*

5.2.1. Carrier Recovery Methods

Phase-Locked Loop

As portrayed in Figure 30, a simple PLL typically contains three main components: an error estimator, a loop filter, and a Sinusoidal Signal Generator (SSG). In the figure above, the signal $s(t)$ is presented in equation (28) at radian frequency of w_c and the arbitrary phase of α is applied to the input of the receiver and multiplied by the local reference $v(t)$ given in equation (29).

$$s(t) = \cos(w_c t + \alpha) \tag{28}$$

$$v(t) = -sin(w_c t + \hat{\alpha}) \tag{29}$$

Also, the error signal $e(t)$ could be determined as:

$$e(t) = v(t) \times \mathrm{s}(t) \tag{30}$$

$$e(t) = -si\,n(w_c t + \hat{\alpha}) \times \cos(w_c t + \alpha) = -\frac{1}{2}\sin(2w_c t + \alpha + \hat{\alpha}) + \frac{1}{2}\sin(\alpha - \hat{\alpha})$$

Considering $\hat{\alpha}$ as a time-varying estimate of α, the input and reference frequencies would be identical. The narrow band LPF rejects the double frequency term, generating the control signal $c(t)$ to drive the SSG;

$$c(t) = \frac{1}{2}\sin(\alpha - \hat{\alpha}) \tag{31}$$

The design of the loop filter, the specification of its order and related parameters, requires the most attention as the discussion of PLL design progresses.

For a quiet carrier or a signal containing a dominant carrier spectral line, carrier recovery can be accomplished with the simple presented PLL. Whereas, many modulation schemes make this uncomplicated approach ineffective because most signal power is devoted to modulation, where the information is present, and not to the carrier frequency. Therefore, different methods must be applied to recover the carrier in practical conditions[22].

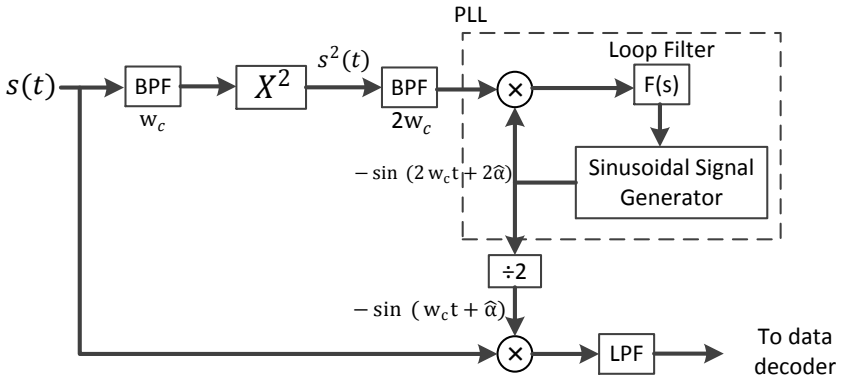

Figure 31. *Squaring Loop Carrier Recovery Method*

Squaring Loop Carrier Recovery

The key idea in this method is to convert the major power of the input signal $s(t)$ to a carrier-dependent signal. It is achievable through applying a direct squaring operation on the input signal $s(t)$;

$$s(t) = \sqrt{2P}\mathrm{d(t)}\cos(w_c t + \alpha) \tag{32}$$

where $d(t) \in \{-1, +1\}$ denotes the transmitted binary signal at time t. The output of the squaring block is given in equation (33).

$$s^2(t) = Pd^2(\mathrm{t})\sin(2w_c t + 2\alpha) \tag{33}$$

Figure 31 illustrates how a quiet-like carrier signal drives a PLL. In this structure, the Band-Pass Filter (BPF) at w_c is considered for noise immunity. Since $d^2(\mathrm{t}) = 1$, the double frequency term in $s^2(t)$ passes through the second BPF. This signal feeds a PLL tuned to $2w_c$. The local reference may then be divided to provide the required signal to down-convert $s(t)$ to baseband [20].

The squaring loop structure may easily be extended to higher-order modulations. For M-ary PSK signal sets, the squarer can be replaced simply with an Mth-law device, and the divide-by-2 can be replaced by a divide-by-M block.

Costas Loop Carrier Recovery

Fundamentally, a Costas loop applies a PLL to compensate the carrier frequency offset and phase error. As can be seen in Figure 32, a quadrature decision-directed phase detector controls the SSG.

Considering $s(t)$ as the transmitted signal in equation (32), the filtered signals $i(t)$ and $q(t)$ in the complex channel are given in (34).

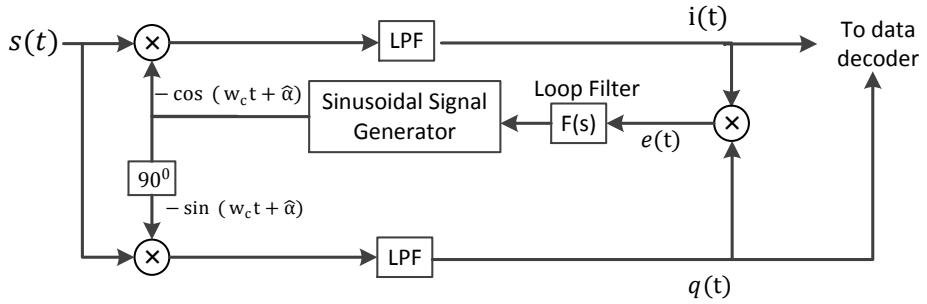

Figure 32. *Costas Loop CR Model.*

$$i(t) = \sqrt{\frac{P}{2}}\mathrm{d(t)}\cos(\alpha - \hat{\alpha})\,,\ q(t) = \sqrt{\frac{P}{2}}\mathrm{d(t)}\sin(\alpha - \hat{\alpha}) \tag{34}$$

The key idea in this method is to provide some mechanism for driving the PLL with a data-independent control signal. In other words, the loop should control the SSG independent of the data $\mathrm{d(t)}$. To achieve this aim, the error signal is the multiplication of the signals $i(t)$ and $\mathrm{q}(t)$:

$$e(t) = \frac{P}{2}\mathrm{d^2(t)}\sin(\alpha - \hat{\alpha})\cos(\alpha - \hat{\alpha}) = \frac{P}{4}\sin(2\varphi),\ \varphi = \ (\alpha - \hat{\alpha}) \tag{35}$$

where φ denotes the phase difference between input and reference carriers. Also, $\mathrm{d^2(t)}$ equals unity which means the modulation is removed via squaring the received signal, providing a PLL-like error signal in $e(t)$. When the error is small, the linearizing approximation in equation (36) can be made, and the Costas loop operates like a typical PLL [20]:

$$e_{small}(t) = \frac{P}{4}\sin(2\varphi)\ \approx 2\varphi \tag{36}$$

5.2.2. The Implemented Carrier Recovery Block

Over the past sections, the Squaring Loop and Costas Loop have been introduced as the possible alternatives for carrier recovery. The appropriate method can be selected by considering the advantages and drawbacks of these approaches concerning the constraints imposed by the hardware platform.

Squaring Loop VS Costas Loop

The Squaring Loop requires two band-pass filters in addition to the Loop filter and two low-pass filters in the demodulation I/Q branches. Besides, the SSG generates a signal at $2W_c$ which should be divided to down-convert the IF signal. Furthermore, as it mentioned earlier, the platform is an NI USRP that provides base-band data which means the carrier offset appears in the form of a low-frequency envelope over the data. Hence, in lower symbol rates, the performance is highly influenced by the band-pass filters.

On the other hand, the Costas Loop utilizes only one filter to estimate the error signal and, as proved in equation (36), it is two times more sensitive to the phase error than the counterpart methods. Moreover, its structure is more consistent with the target hardware because the SSG, as the actuator in the controlling system, could be simply replaced by a phase shifter block.

Therefore, in comparison with the Squaring Loop approach, the Costas Loop can be implemented more effectively.

Adapted Costas Loop

This section introduces a practical form of the Costas Loop and explains its realization dimension in detail. As it was mentioned above, the model is quite appropriate to be applied to the base-band signal and the actuator is a phase shifter. Figure 33 shows the proposed CR block which uses a cross product phase detector instead of an arctangent estimator.

As defined in equation (37), the error detector removes the effect of modulating data via multiplying the sign of real and imaginary data to the imaginary and real part, respectively:

$$e = Image(x) \times sign\big(Real(x)\big) - Real(x) \times sign\big(Image(x)\big) \tag{37}$$

In this design, the loop filter is a combination of two IIR filters H_1 and H_2. To obtain the frequency and phase response (see Figure 34), its transform function is driven as following:

$$(X_1 + Y_1)z^{-1} = Y_1 \rightarrow\ H_1 = \frac{Y_1}{X_1} = \frac{z^{-1}}{1-z^{-1}}\ (-X_2 + Y)z^{-1} = Y \rightarrow\ H_2 = \frac{Y}{X_2} = \frac{-z^{-1}}{1-z^{-1}} \tag{38}$$

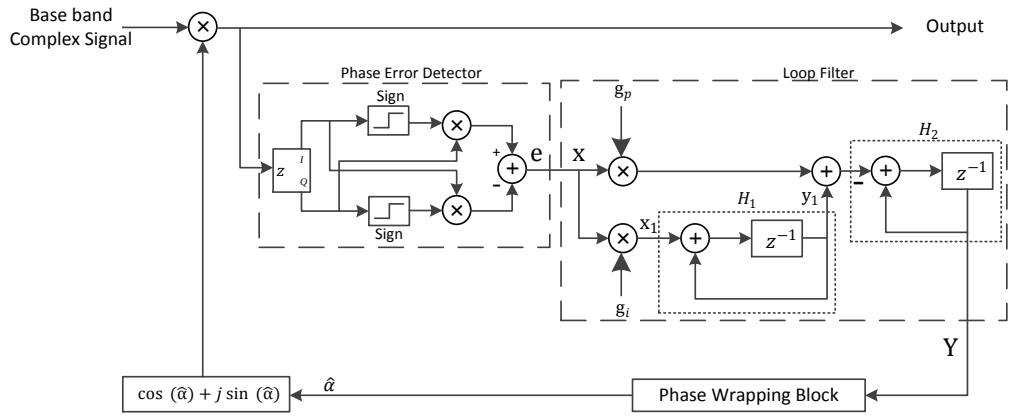

Figure 33. *Structure of The Implemented Carrier Recovery Block.*

$$X_1 = g_i X,\ X_2 = X_1 H_1 + g_p X,\ Y = X_2 H_2 \rightarrow \tag{39}$$

$$H_{LF} = \left(g_p + g_i H_1\right) H_2 = \frac{-g_p z^{-1}}{1-z^{-1}} + \frac{-g_i z^{-2}}{\left(1-z^{-1}\right)^2} = -\frac{g_p z^{-1} + (g_i - g_p) z^{-1}}{1 - 2z^{-1} + z^{-2}}$$

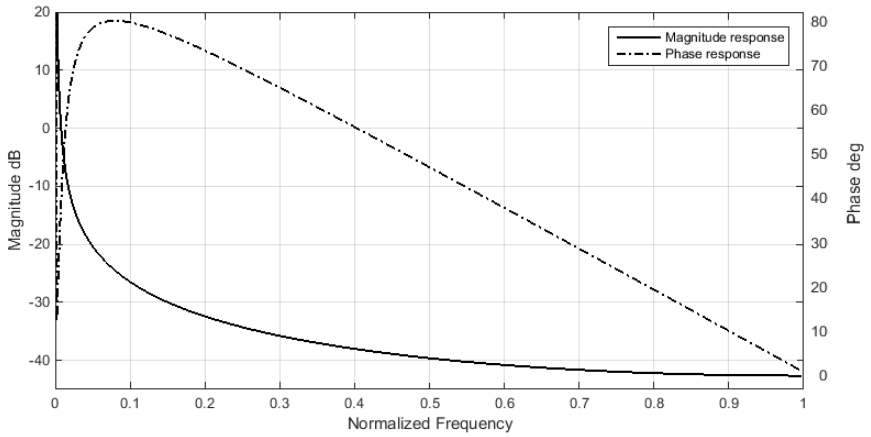

Figure 34. *Loop Filter Magnitude and Phase Response. (g_p=32×g_i=0.015).*

As it can be seen in Figure 33, to keep the estimated phase in a confined range, the wrapping block wraps the filtered signal within [-π,π];

$$y = \begin{cases} x & |x| \le \pi \\ x - 2\pi & x > \pi \\ x + 2\pi & x < \pi \end{cases} \tag{40}$$

In order to study the adapted Costas Loop more precisely, it is realized in MATLAB Simulink (see Figure 35). In this simulation, the CR model is fed by a BPSK signal that carries 40 bps binary data at 200 Hz.

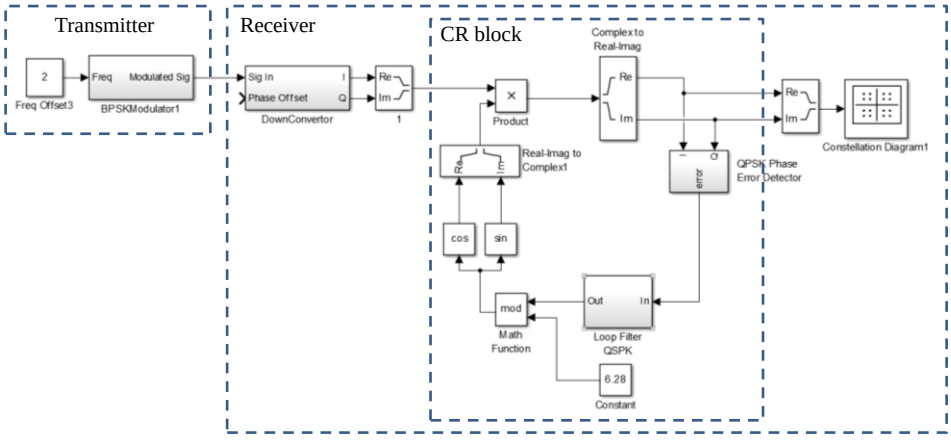

Figure 35. *The Simulation of The Adapted Carrier Recovery Block in MATLAB Simulink. F_s=1 kHz, F_c=200 Hz, BPSK Symbol Rate=40 bps, Frequency Offset = 2 Hz.*

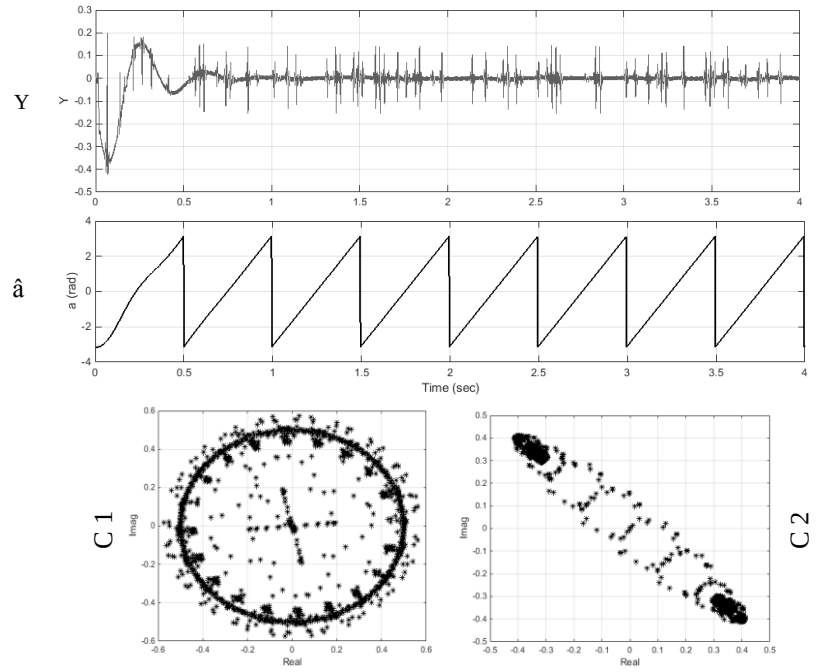

Figure 36. *Carrier Recovery Control Signals (Frequency Offset 2 Hz); Y: The Loop Filter's Output, $\hat{\alpha}$: The Compensator Phase, C1: The Rotating Constellation due to The Frequency Offset 20 Hz, C2: The Resultant Constellation by RC Block.*

The plots in Figure 36 explain how the loop filter estimates $\hat{\alpha}$ to resolve 2 Hz frequency difference between the receiver and transmitter. As shown in this figure, the wrapping block generates a saw-tooth signal within [-π , π], following the phase of the carrier. The signal is applied to a phase shifter to compensate the momentary phase changes. In this example, the loop filter locks in 0.6 sec and the phase $\hat{\alpha}$ raises from -π to π two times in a second.

The represented carrier recovery model in LabVIEW is depicted in Figure 37; it involves the phase shifter, the phase error estimator which is given in Figure 38, and the wrapper block presented in Figure 39.

The phase shifter includes a Xilinx Cordic IP core that generates $\sin(\hat{\alpha})$ and $\cos(\hat{\alpha})$, a complex multiplier and a VHDL-described block that converts data to the desired format. To save more logic blocks and avoiding multiplication round-off error, the gain parameters $g_1$and g_2 are also applied by arithmetic right shift in SigAtt16bit VI.

The wrapped signal in equation (40) is also realized by LabVIEW components in Figure 39. Due to some limitation in LabVIEW, the block utilizes two adders and two subtractors to make the desired output. However, it could be carried out by a VHDL description in a more efficient way.

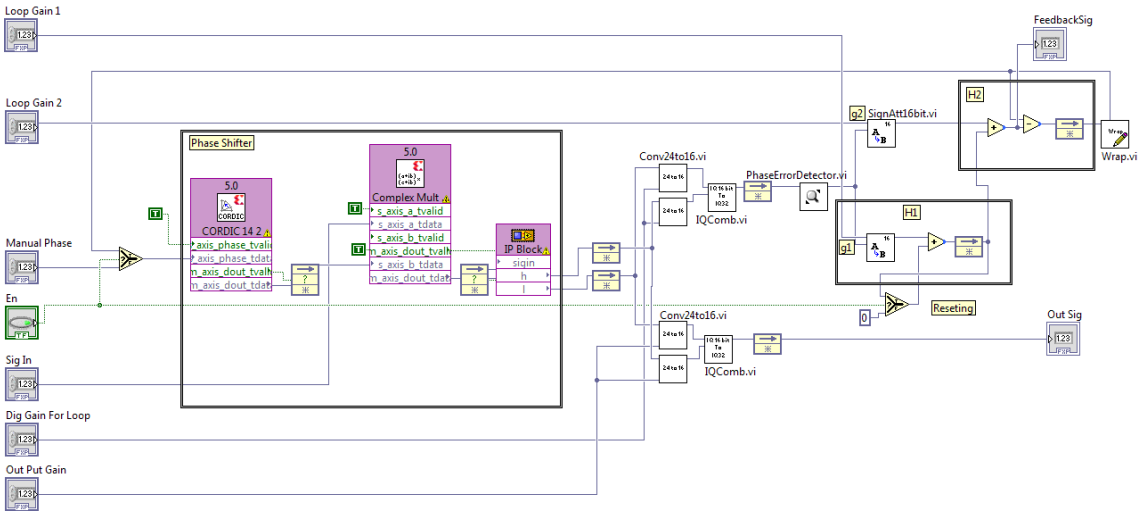

Figure 37. *LabVIEW Implementation of The Carrier Recovery Block.*

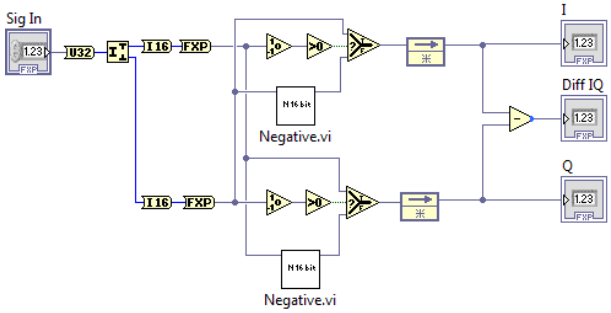

Figure 38. *LabVIEW Implementation of The Phase Error Estimator.*

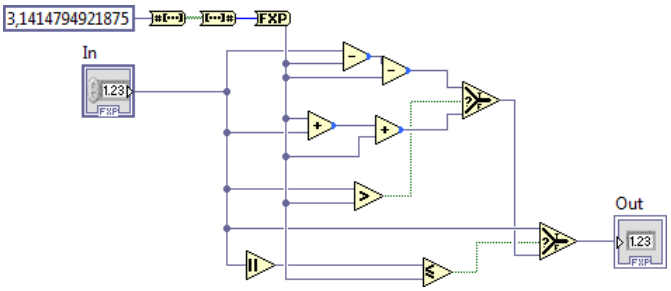

Figure 39. *The Implemented Wrapping Sub-block in LabVIEW.*

5.2.3. Phase Stablizer

The proposed Carrier Recovery block in the previous section compensates the carrier frequency offset in both BPSK and QPSK schemes. Hence, it imposes the 45^0 symbol rotation in BPSK. Furthermore, as shown in Figure 40 (right), when the amount of carrier frequency offset is substantial, in the steady state, the CR block is inadequate for rotating symbols to their desired angle. However, it locks the constellation and prevents

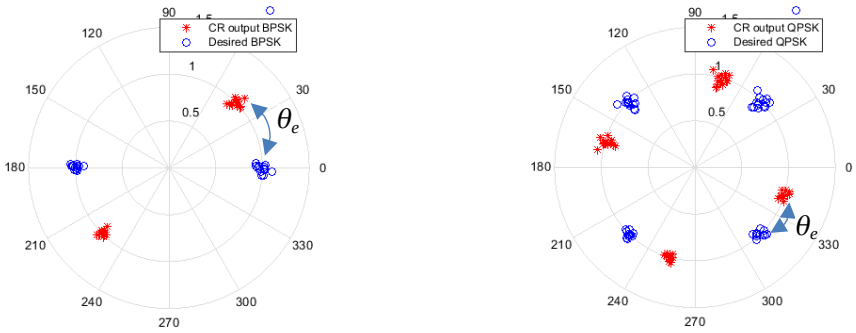

Figure 40. *Carrier Recovery Output in BPSK (left), Carrier Recovery Output when The Carrier Frequency Offset Is Considerable (right).*

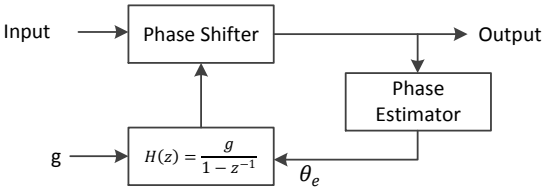

Figure 41. *Phase Stabilizer.*

rotation. As a possible solution, boosting the gain factors enhances the constellation. Instead, in lower SNRs, it suppresses the performance because high feedback gain leads to instability. A realistic alternative is a phase stabilizer that plays a complementary role beside the CR block and resolves the residual carrier phase offset.

Figure 41 depicts the block diagram of the introduced phase stabilizer and Figure 42 shows its realization in LabVIEW. It involves a Xilinx Cordic IP core for phase rotation, a phase estimator block that was demonstrated earlier in Figure 38 and other LabVIEW components to form the loop filter. In this block, the control loop diminishes the error signal θ_e. In fact, this error signifies the phase difference from the desired constellation (Figure 40, blue patterns) which is selected according to the PSK scheme.

Figure 43 reveals how effectively the implemented block enhances the constellation patterns. In this example, the carrier is at 10 kHz offset relative to the receiver frequency. The CR block compensates the carrier offset although there is an undesirable rotation in the resultant symbols. Figure 43 (right) shows the enhanced signal by the phase stabilizer block.

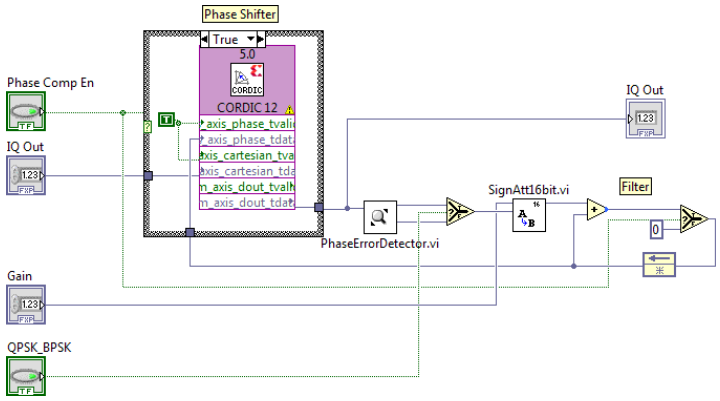

Figure 42. *LabVIEW Implementation of The Phase Stabilizer.*

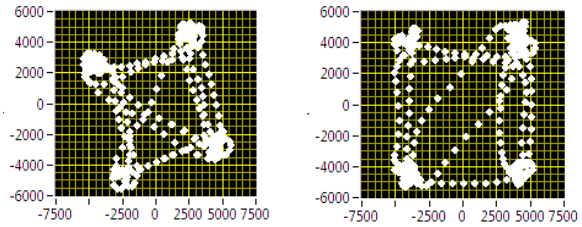

Figure 43. *The Affected Constellation by Large Frequency Offset (left). The Enhanced Constellation by The Phase Stabilizer (right).*

5.3. Symbol Time Recovery

Similarly to the CR block, another technique is required to compensate the symbol rate frequency and phase offset. In other words, to recover correct data, the sampling clock has to be synchronized with the symbol frequency and phase of the received signal. In this synchronization state, the receiver generates a sampling clock from an approximated frequency reference, and then phase-aligns to the transitions in the data stream by a closed loop controller [23].

Practically, it is desirable to utilize the same oscillator for the data down-conversion and sampling. However, to generalize the study, this thesis work considers two independent adjustable frequency offsets for each. Since the performance of the CR block influences the efficiency of the CDR block, this feature gives a confident evaluation of either the CR or the CDR blocks.

Figure 44 illustrates the block diagram of a typical CDR. In this schematic, the sampled symbols are sent to a timing error estimator which can make use of many different techniques to generate a timing error signal $e(t)$. To apply a Proportional-Integral (PI) controller, the error passes through the loop filter and controls the clock generator.

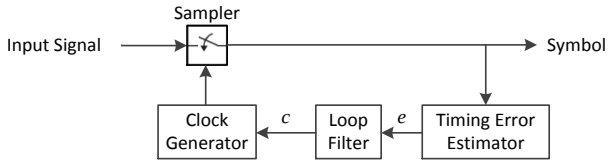

Figure 44. *Clock and Data Recovery Block Diagram.*

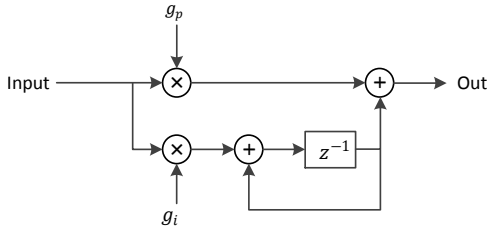

Figure 45. *Loop Filter in CDR Block.*

It can be seen in Figure 45 that the loop filter is to some extent similar to the loop filter in the CR block and is adjustable with proportional and integral gain factors g_p and g_i. In following, this section explains three more frequently used timing error estimation approaches in brief and considers their benefits and drawbacks to make a comparative perspective.

5.3.1. Early-Late Approach for Clock Recovery

This straightforward strategy is based on the preceding and following samples. It means that the slope of the line between these two samples provides a reasonable criterion for compensating the frequency offset. The Early-Late error signal is defined in equation (41).

$$e_{EL}[n] = (y[n+1] - y[n-1]) \times sign(y[n]) \tag{41}$$

Figure 46 demonstrates distinct timing situations; in the correct timing, $e_{EL}[n]$ equals zero and on the other hand, when the error is positive/negative, the sampling is early/late.

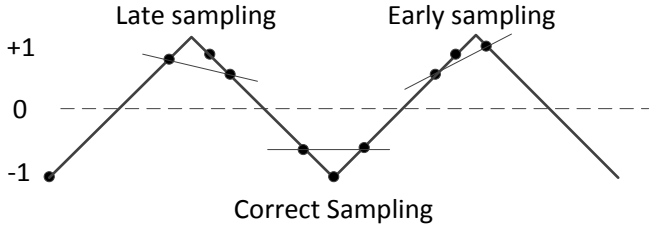

Figure 46. *Early-Late Time Recovery Technique.*

The method is relatively simple to be performed because it only requires a subtractor and a logic switch. Instead, it is sensitive to noise and pulse shaping. Furthermore, when the present sample is close to zero, the controller may lock in an ambiguous state and a lateral mechanism is needed to resolve the ambiguity.

5.3.2. Mueller and Müller Timing Error Estimation

Equation (42) illustrates how the Mueller and Müller method [20] utilizes one sample per symbol to determine the timing error. This technique computes the error term considering a sample from current symbol $y[n]$ and three previous symbols.

$$e_{MM}[n] = (y[n] \times y[n-3]) - (y[n-1] \times y[n-2]) \tag{42}$$

Based on Figure 47, $e_{MM}[n]$ is nearly zero when the sampling clock and the symbol rate are perfectly synchronized. Otherwise, the error value provides a sensible estimation of the sampling frequency deviation.

Since the method needs only one sample per symbol, it is comparatively sensitive to carrier offsets. Besides, it requires two multipliers and a subtractor to compute the error.

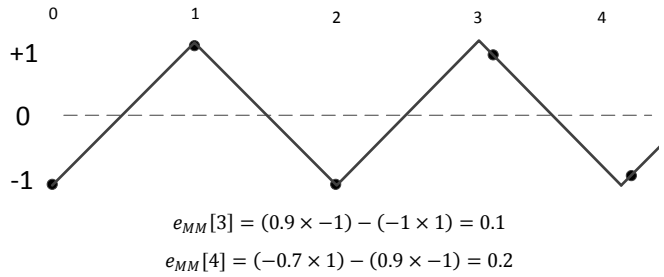

Figure 47. *Mueller and Müller Timing Error Estimation.*

5.3.3. Gardner Timing Error Algorithm

Among all the timing error estimators, the one introduced by Gardner [25] is widely used. This is because, in contrast with many other methods, it is insensitive to carrier offsets. The method estimates the timing error based on three consecutive samples as given in equation (43).

$$e_G[n] = (y[n-2] - y[n]) \times y[n-1]) \tag{43}$$

Figure 48 reveals how the method computes the symbol rate error given three continuing samples. In this model, the extracted symbols would be the all odd sampled values. The method needs one multiplier and a subtractor, and therefore, less hardware to be implemented.

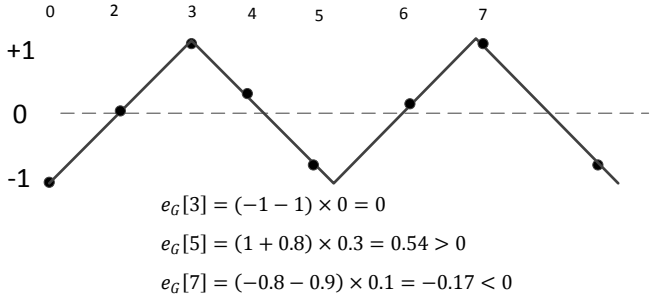

Figure 48. *Gardner Timing Error Estimation.*

5.3.4. The Implemented Clock and Data Recovery Block

Figure 49 depicts the implemented CDR module that utilizes a Gardner timing error estimator to provide the control signal for a DDS. To overcome some practical implementation constraints related to the platform, the DDC generates a sinusoidal signal at the quadruple rate of the reference symbol rate SR and the filtered error signal feeds its phase input. To extract sampled symbols y[n-k], a VHDL-based sampler is implemented. This block down-samples the complex input signal to the rate of 2×SR. However, it latches the symbol outputs by the rate of the recovered symbol rate. The implemented sampler is involved in Appendix 2. As shown in Figure 49, the sampler comprises a Schmitt-level controller that makes a proper clock from the DDS output; the adaptiveness of the clock generator is the causative factor of the clock fluctuation when the DDS signal is around zero. It leads to resample a symbol several times, creating fake symbols. The Schmitt-level controller prevents the sampling clock from fluctuating and hence, plays a critical role in the sampler block. The definition of this part is given in equation (44).

$$y[n] = \begin{cases} 0 & x[n] \le L_1 \\ 1 & x[n] \ge L_2 \\ y[n-1] & \text{otherwise} \end{cases} \tag{44}$$

where $L_1$and L_2 are the minimum and the maximum level to expose 0 and 1.

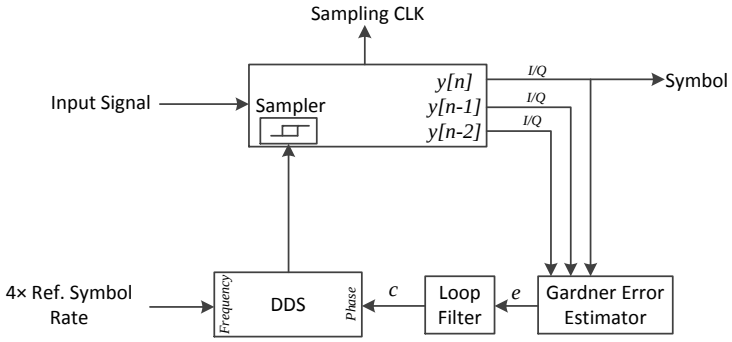

Figure 49. *Implemented CDR Block.*

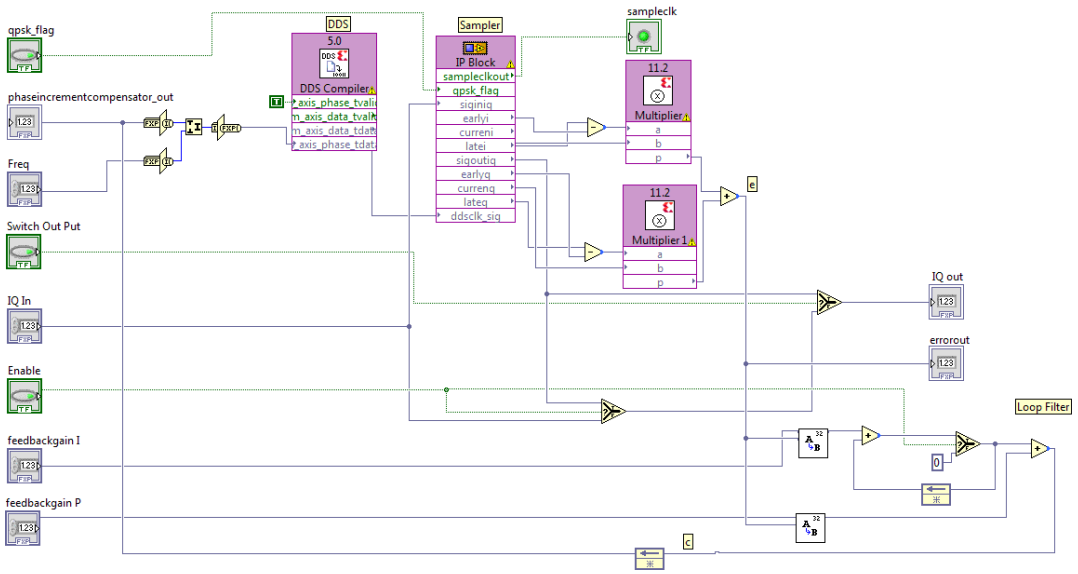

Figure 50. *Implemented CDR Block in LabVIEW.*

Figure 50 shows the presented CDR block in LabVIEW. In this figure, the error term $e_G[n]$ is accomplished by multipliers and subtractors for both real and imaginary signals. Additionally, an operating mode flag is considered in order to disable the imaginary branch in BPSK mode.

6. SYSTEM INTEGRATION

The previous chapters specified the essential elements of a BPSK-QPSK modem and also expanded each block in detail. Chapter 6 covers the system integration on the target hardware and depicts the realized model as the combination of the designed sub-systems. Besides**,** this chapter explains the importance of the AGC block in the digital communication systems and develops the structure of the proposed AGC.

6.1. Automatic Gain Control

As mentioned in Chapter 4, the stream data in the implemented demodulator is in fixed-point format. Furthermore, all the gain factors are applied by the signed shift operation to preserve more FPGA logic resources. In fact, the input signal should be stronger than a certain level to drive feedback loops in the CR block, phase stabilizer, and CDR block. On the other hand, it is desirable to increase the ADC's dynamic range which indicates the ratio between the largest and smallest possible inputs [26]. Therefore, it is essential to apply a gain stage to amplify the input signal up to an acceptable level. In general, amplifiers induce a negligible amount of noise to the signal which is the cost of dynamic range.

The platform NI USRP2943 possesses an analog gain circuit to provide 50 dB adjustable gain. In this thesis work, the AGC is realized on the PC software interface. The AGC amplifies the input signal to reach the desirable level and also protects the IF block from over-amplification. Equation (45) shows how a simple mechanism in the presented AGC in Figure 51 selects the feedback gain. In this procedure, when the signal is stronger than the desirable level, the decision function $f(x)$ imposes a large loop gain to attenuate the input signal quickly and in contrast, when the input is a weak signal, the decision block applies a small value to raise the straight gain factor gradually. In fact, the system is more agile in attenuation than amplification.

$$f(x) = \begin{cases} g_1 & x \leq -\Delta l \\ g_2 & x \geq \Delta l \\ 0 & \text{Otherwise} \end{cases} \tag{45}$$

where $g_1 \ll g_2$ and $-\Delta l < e < +\Delta l$ is the desirable operation region.

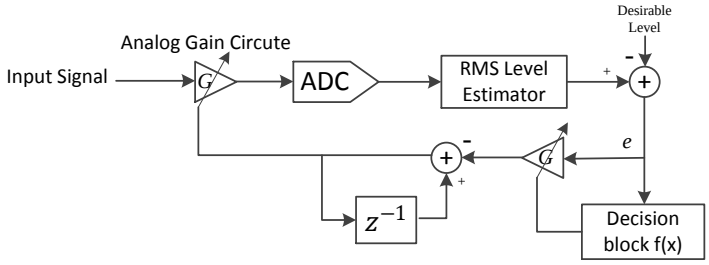

Figure 51. *Implemented AGC.*

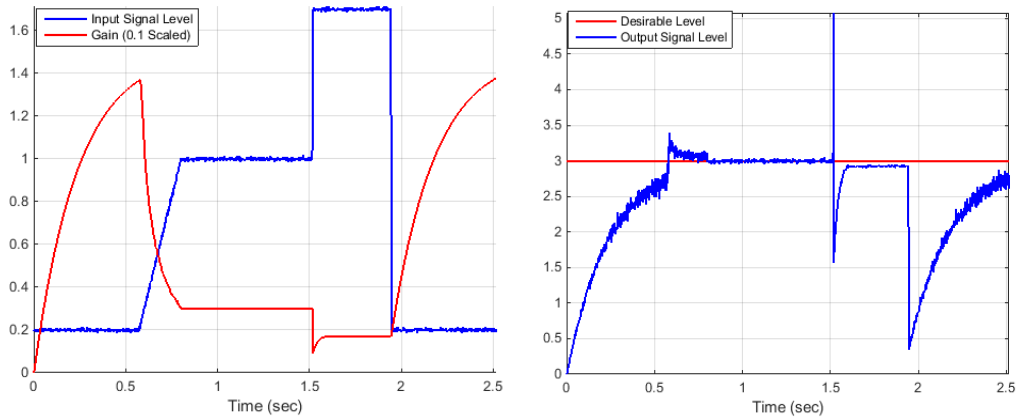

Figure 52. *AGC Response in MATLAB Simulation. AGC Parameter:* $g_1 = 0.03$, $g_2 = 1$, *Desirable Level = 3,* $\Delta l = 0.1$, *The Input Signal Level and Gain Factor (left), The Output Signal Level (right)*

In order to investigate more about the introduced AGC, it is simulated in MATLAB. As shown in Figure 52 (left), the test signal forces the AGC to move between the three states of the decision function $f(x)$. The output signal in this figure reveals that the AGC adaptively amplifies the signal to meet the minimum required level, and it is also resistant to instant level overshoot. Typical, the commercial radio equipment provides fast, slow, medium and manual AGC operating modes. In fact, the gain parameter g_1 determines the operating mode and intuitively $g_{1_{fast}} > g_{1_{medium}} > g_{1_{slow}}$.

6.2. Proposed Testbeds

As mentioned earlier, two testbeds are proposed; the All-Digital Transceiver realized in Figure 53 and the Radio-Based Transceiver in Figure 54. The former is considered for Research and Development (R&D) phase because its synthesis time is significantly short. In contrast, the latter structure is considered in order to evaluate the system in a more realistic way; it involves the accessories to drive the radio and thereby the synthesis time is comparatively long.

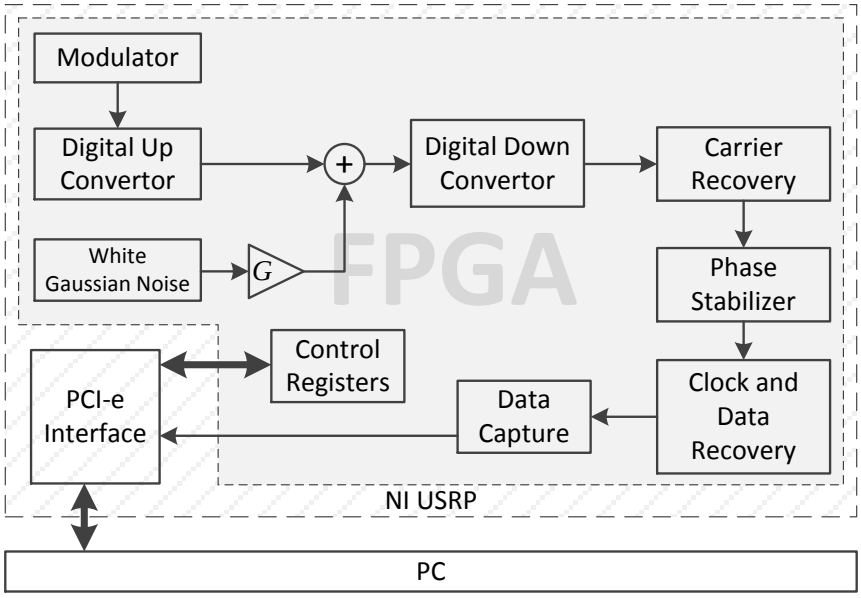

Figure 53. *All Digital Transceiver Testbed.*

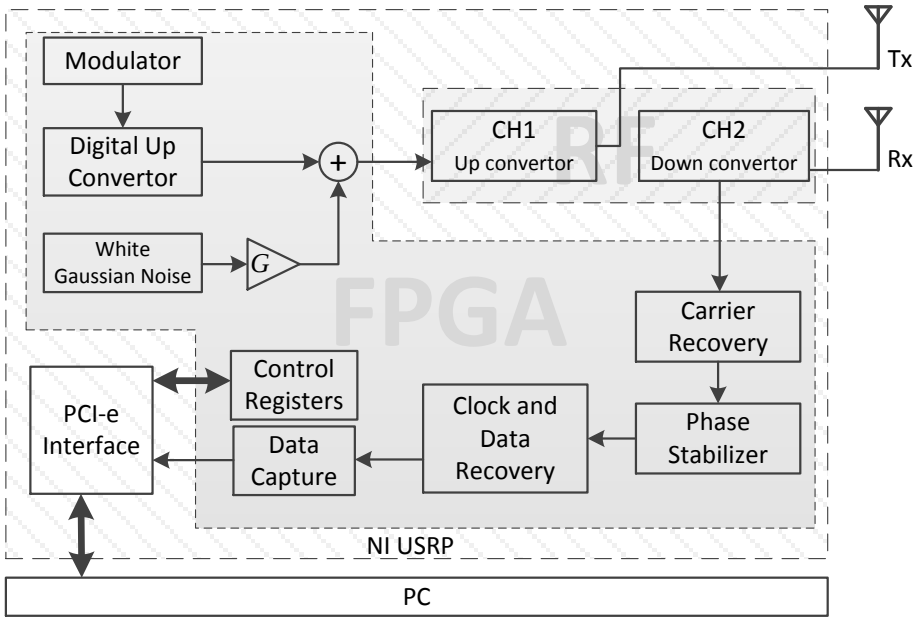

Figure 54. *Radio-Based Transceiver Testbed.*

Table 3 shows the compilation result for these two models. It reveals the advantage of utilizing the all-digital transceiver testbed in the debugging and development procedure; It can be seen from the data in the table that the first architecture requires less than an hour to be ready while the compiler needs roughly three hours to build a working radio-based testbed.

Table 3. *Compilation Time, Device Utilization and Timing for The Presented Testbeds.*

		All-Digital Transceivers	Radio-Based Transceivers
Device Utilization (%)	**Total Slices** *Out of 63550*	**0.7%**	**33.1%**
	Slice Registers *Out of 508400*	**3.9%**	**12.3%**
	Slice LUTs *Out of 254200*	**6.2%**	**17.6%**
	DSP48s *Out of 1540*	**14.9%**	**31.6%**
	Block RAMs *Out of 795*	**6.7%**	**21.1%**
Timing	**40 MHz Onboard Clock**	**60,64 MHz**	**109.6 MHz**
	Data Clock	**171,14 MHz**	**118.84 MHz**
	DRAM Clock	**709,72 MHz**	**709.72 MHz**
Compilation Time(min)	**Time compiling**	**50**	**186**
	Plan Ahead	**1**	**2**
	Core Generator	**0**	**0**
	Synthesis - Xst	**11**	**65**
	Translate	**10**	**62**
	Map	**9**	**277**
	Place & Route	**7**	**21**
	Bitgen	**3**	**10**

7. EVALUATION

Chapter 8 presents a test methodology to assess the implemented system. The method measures the SNR and the performance to form the BER curve. Comparing the resultant graphs and the theoretical ideal curve in Chapter 2 illustrates the effectiveness of the implemented system.

To measure the BER, similar LFSRs generate the reference bitstreams on the PC regarding the seed numbers. The control interface, which is running on PC, correlates the reference bitstreams and the receiving data stream to find the number of incorrectly recovered bits. The expectation of this number determines the value of BER. It should be pointed out that, in BPSK scheme the BER and the Symbol Error Rate are identical while in QPSK the BER is double.

In next step, an estimation of SNR in equation (46) is needed.

$$SNR = \frac{P_x}{P_{noise}} = \frac{\sum x[n]^2}{\sum N[n]^2} \tag{46}$$

where, $x[n]$ is the informative signal and $N[n]$ is the additive Gaussian noise.

Applying a zero gain to the output of the modulator is a simple way to obtain the power of noise. In contrast, the estimation of the signal power P_x is not equally. Practically, what can be measured is the power of summation of the noise and the informative signal $P_{x+noise}$;

$$P_{x+noise} = \sum(x[n] + N[n])^2 = \sum x[n]^2 + \sum N[n]^2 + \sum 2N[n]x[n] \tag{47}$$

Since the noise and signal are uncorrelated, the last term is zeros and hence;

$$P_{x+noise} = \sum x[n]^2 + \sum N[n]^2 = P_x + P_{noise} \tag{48}$$

For a large amount of data, equation (48) gives an acceptable approximation of P_x and thus, the SNR is achievable as driven in equation (49) and (50);

$$SNR = \frac{P_x}{P_{noise}} = \frac{P_{x+noise} - P_{noise}}{P_{noise}} = \frac{P_{x+noise}}{P_{noise}} - 1 \tag{49}$$

$$SNR_{dB} = 10\log(\frac{P_{x+noise}}{P_{noise}} - 1) \tag{50}$$

As a summary, the following procedure is carried out for the evaluation of the BER curves:

1) Radio Gain adjustment.
2) Signal and noise power $P_{x+noise}$ estimation.
3) Noise power P_{noise} measurement (Applying zero gain to the modulator).
4) SNR estimation (as defined in equation (49)).
5) BER measurement.

7.1. Results and Discussion

Comparing the resultant and the ideal BER graphs illustrates the performance of the presented transceiver. This section mainly demonstrates the resultant comparative plots based on the radio-based testbed. The objective of this experiment is to assess the performance of the CR, PS and CDR blocks dealing with frequency offsets. Therefore, the frequency offsets are imposed by the considered facilities in the modulator and a coaxial cable is applied to play the role of an ideal communication channel, avoiding the impacts of the air channel. As explained earlier, the radio-based testbed employs the first channel as the transmitter and the second as the receiver. Figure 55 shows how the channels are connected via a short cable.

Figure 56 and Figure 57 show the BER plots for BPSK and QPSK at two data rates. In this experiment, the demodulator is configured with fix parameters to recover an unshaped transmitting signal. Besides, two terminals are perfectly synchronized; however, all blocks are included in the demodulation procedure.

Considering these results, it can be seen that the model works acceptably in ideal condition, when there are no carrier and symbol rate offsets. In fact, the experimental curves follow the ideal ones in both schemes and symbol rates.

Figure 55. *Radio-based Testbed.*

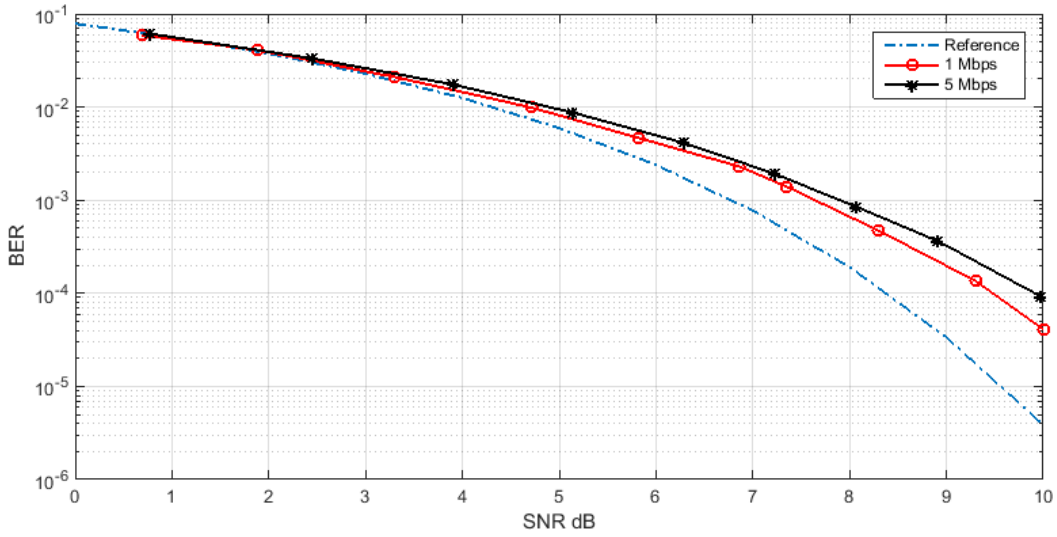

Figure 56. *BPSK Test @ 2 GHz. BER Curve*

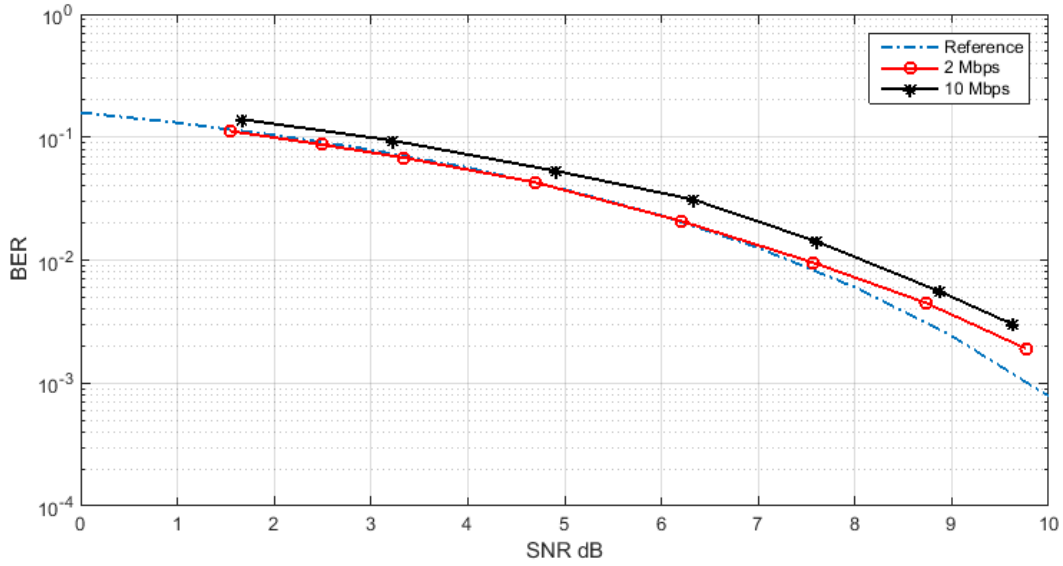

Figure 57. *QPSK Test @ 2 GHz., BER Curve*

The main focus in this work is on carrier recovery and clock synchronization. Therefore, the next tests consider practical conditions.

Figure 58 and 59 present the SNR VS maximum tolerable carrier frequency offset in linear and logarithmic scales. The test indicates that the combination of CR and PS blocks effectively compensates the carrier frequency and phase offset, and the BER performance raises with the increasing SNR. Since the PS loop gain factor is rather small, the block comes to work in higher frequency shifts; where the CR merely locks on the carrier frequency and there is still a residual phase offset.

The test is carried out for three different CR loop gains g_i to investigate the optimum value. Comparing the resultant plots shows that a higher gain obtains more compensa-

tion range in higher SNR while it reduces the performance for noisy signals. Furthermore, the CR mechanism is more effective with BPSK scheme. Besides, as can be seen in Figure 60, the efficiency of the CR block is almost independent of the symbol rate.

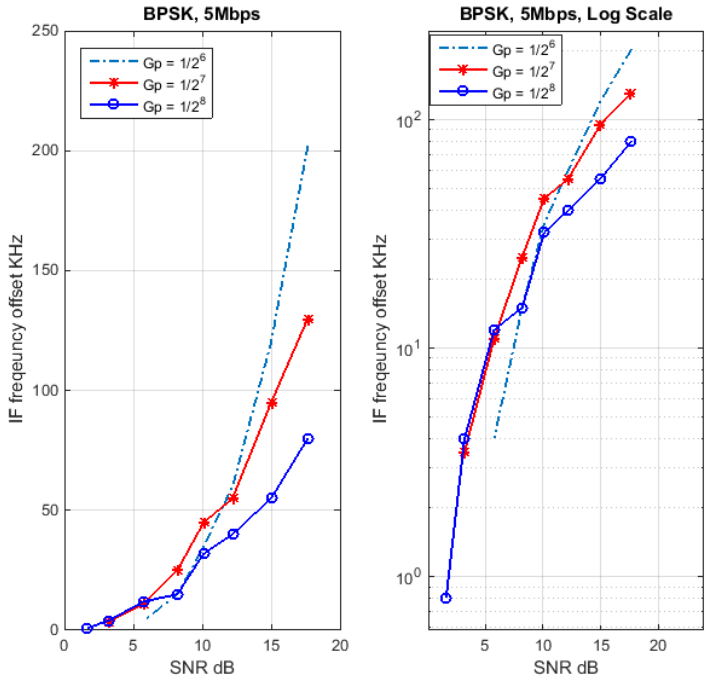

Figure 58. *Carrier Recovery Test, BPSK Scheme @ 2 GHz, 5 Mbps.*

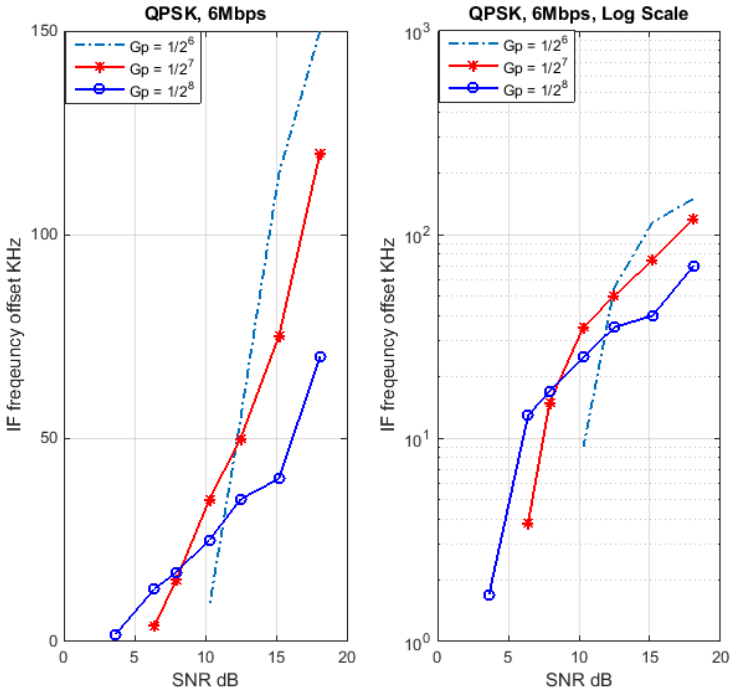

Figure 59. *Carrier Recovery Test, QPSK Scheme @ 2 GHz, 6 Mbps.*

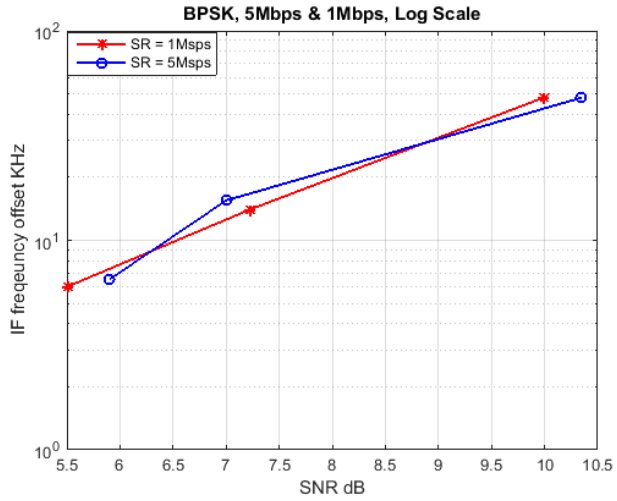

Figure 60. *Carrier Recovery Test, BPSK , @ 2 GHz, 1 and 5 Mbps*

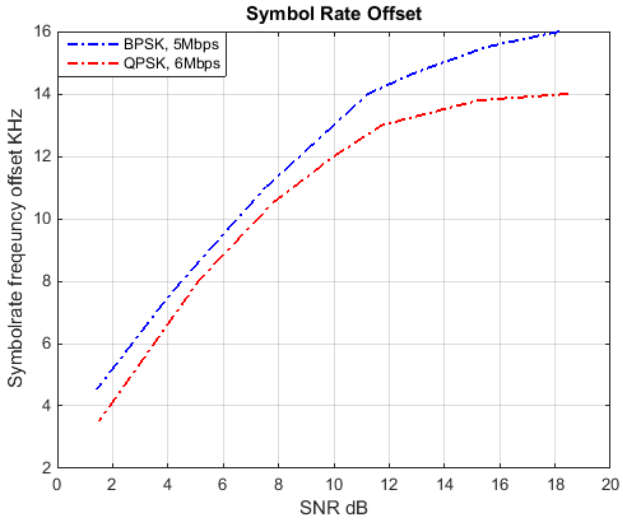

Figure 61. *Maximum Tolerable Symbol Rate Offset VS SNR, @ 2 GHz.*

In the following tests, the receiver is tuned exactly at the carrier to assess the CDR block individually. The examination concerns the maximum tolerable symbol rate offset to achieve the same BER with no offset. Figure 61 shows the test for BPSK and QPSK schemes. The results indicate that the designed block resolves the offset effectively, and the BER performance rises in higher SNRs. Additionally, according to this data, the symbol timing recovery function is accurate enough to lock on a clock at 4 kHz offset and 2 dB SNR. Also, the overall performance is better in the BPSK scheme.

As it was mentioned earlier, there is more difficulty accompanied with a shaped signal to recover the clock and performing time synchronization. The next evaluation provides an illustrative comparison; Figure 62 shows the result of the same test for BPSK for both shaped and unshaped 1 Mbps signals. As it can be seen in this demonstration, the shaping filter decreases the lock range up to 3 kHz.

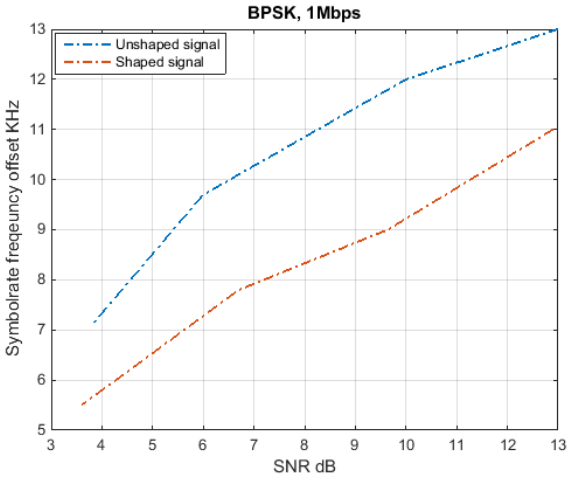

Figure 62. *Maximum Tolerable Symbol Rate Offset VS SNR, @ 2 GHz. Shaped Signal (red), Unshaped Signal (blue)*

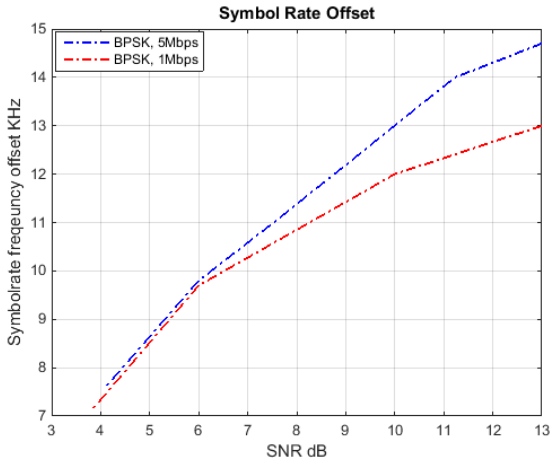

Figure 63. *Maximum Tolerable Symbol Rate Offset VS SNR, @ 2 GHz. BPSK 1 Mbps(red) and 5 Mbps(blue)*

In the next examination, the effect of symbol rate on the operation of the CDR block is studied. Figure 63 shows the plots for BPSK in 1 and 5 Mbps. Objectively, the implemented CDR block works better in the higher symbol rate, especially when the offset frequency is more than 6 dB. The reason lies in the speed of error estimator block; since the block computes the Gardner error as fast as the symbol rate, when the frequency offset is very high, it is not agile enough to track the clock frequency offset. Therefore, the maximum permissible drift increases with raising symbol rates.

8. SUMMARY AND CONCLUSION

This thesis work proposes a practical FPGA-based BPSK-QPSK transceiver which is robust against the carrier frequency offset and accurate in the symbol timing recovery. To adopt the best methods which fit the target platform and provide the highest robustness, many documents were studied and several CR and CDR approaches were reviewed. Due to their effectiveness, the Costas loop carrier recovery and the Gardner time synchronization techniques were simulated and implemented on a Xilinx Kintex 7 FPGA as the processing core of NI USRP 2943 device. The constituent blocks in the final architecture were evaluated independently and the performance was assessed experimentally. The implemented modulator utilized the proposed LFSRs to produce a modulating data stream at the desired data rate and a shaper filter enhanced the spectral features of the stream to make it ready to be transmitted by the RF module. To simulate the Additive White Gaussian Noise (AWGN) channel the noise signal was generated by a white Gaussian noise generator inside the modulator. Additionally, in order to generalize the assessment, most of the tests were accomplished with distinct symbol rates at 2 GHz and an attenuator adjusted the power of noise to achieve the desirable SNR. On the receiver side, the realized demodulator employed the proposed carrier recovery, phase stabilizer and clock and data recovery components to be synchronized with the transmitter and reconstruct the transmitted bitstream. To measure the BER performance, a PCI-e interface sent the recovered data to a controlling computer.

In contrast to the conducted researches in [3],[9],[12],[27], and [28], the presented evaluation is not confined to the ideal condition where the receiver is perfectly tuned to the carrier frequency or the transmitter generates an accurate data-rate. Even the proposals like what is conducted in [31], concerned only a fixed carrier frequency offset, in their assessment phase. To make a distinction in this investigation, this thesis has studied the behavior of the proposed model in a more realistic way and considered the following operating ranges:

1) SNR: [1,18] dB
2) Carrier Frequency Offset: [-200, 200] kHz
3) Symbol Rate Offset: [-15, 15] kHz
4) Symbol Rate: 1 Mbps and 5 Mbps for BPSK, 6Mbps and 10 Mbps for QPSK.

As another remarkable advantage, the target platform in this thesis work makes use of a dual-channel radio to provide an RF link while, the most relevant studies have been

conducted in baseband. In fact, the proposed design has adapted a baseband CR component to be applicable in IF.

In Chapter 7, the evaluation results reveal that in the case of a strong signal (18 dB SNR) the carrier frequency lock range and the maximum tolerable symbol rate deviation are up to ±200 kHz and ±14 kHz, respectively. Conversely, for a weaker receiving signal (5 dB SNR), the lock ranges decrease to ±5 kHz and ±8 kHz. In contrast to the SNR, the symbol rate influences the CDR block's performance inversely. In fact, due to the timing error estimation mechanism, the CDR block operates more accurately at higher data rates.

Based on the obtained results, it is concluded that the principal aims of the thesis are accomplished as following:

1) The modulator generates the QPSK-BPSK signal in the desirable data-rates and it is also capable of being synchronized with the other similar modulators to form an array-based transmitter.
2) The demodulator effectively overcomes the frequency, phase and timing offsets and reconstructs the modulated data in the presence of noise.
3) The designed components are evaluated independently and their effectiveness is assessed with respect to the model parameters like loop gain factors, symbol rate and SNR.

8.1. Future Works

Due to the adaptability of the implemented system, it can find application in a wide variety of areas. Furthermore, the assessment in Chapter 8 opens a new landscape to enhance and optimize the presented architecture. Hence, the following items can be drawn for future research:

1) As explained in Chapter 4, the modulator can induce fine time shift to the transferring data. Therefore, multi transmitters could be synchronized to maximize the power in a long distance link such as Deep Space Network. On the other hand, the demodulator is adaptive enough to be synchronized with other similar blocks to perform calibration in such a system.
2) Realistically, the communication channel imposes distortion on the traveling signal. There is a wide range of approaches to channel estimation. However, a few techniques can be realized on FPGA efficiently. A further study with more focus on the FPGA-based channel equalizer is therefore suggested.
3) In the proposed system, the AGC is a mounted software module to the control interface. Realization of an FPGA-based AGC is one of the possible future developments. The advantage is that the system is protected against over-

amplification even if it is not connected to the computer. In fact, the gain controller must work regardless of external interrupts.

4) The result in the previous chapter indicates that the frequency compensation dynamic range, in both CR and CDR stages, is reliant on the feedback gains. Hence, an adaptive mechanism could be applied to adjust the gain according to the signal to noise ratio. Realization of a noise estimator block on FPGA is the first step to achieve this aim. Therefore, further studies, which take these variables into account, will need to be undertaken.
5) The sampling clock plays an important role in the BER performance of the demodulator. Hence, the parameter of "sample per symbol" is interesting to be studied in the future works.
6) To obtain the pure effectiveness of the proposed system, there is a need to consider the characteristics of hardware like the Superior Free Dynamic Range (SFDR) and the Effective Number of Bits (ENOB) of the Analog to Digital Convertor (ADC).

REFERENCES

[1] *1st First International Workshop on Software Radio, Proceeding of the ACTS Mobile Summit* (Rhodes, Greece: European Commission, June 1998)

[2] M. W. Hussain, Design and Development from Single Core Reconfigurable Accelerators to a Heterogeneous Accelerator-Rich Platform. *Thesis for the degree of Doctor of Science in Technology*, Tampere University of Technology, Tampere, Finland, 2014.

[3] P. Zicari, E. Sciagura, S. Perri, and P. Corsonello, "A programmable carrier phase independent symbol timing recovery circuit for QPSK/OQPSK signals," *Journal of Microprocess & Microsystems*, vol. 32, no. 8, pp. 437–446, Nov. 2008.

[4] C. Dick, F. Harris and M. Rice, "Synchronization in software radios - Carrier and timing recovery using FPGAs," Symposium on *Field-Programmable Custom Computing Machine. IEEE 2000*, pp. 195–204, Apr. 2000.

[5] Xilinx.com, 'Virtex-7 FPGA Family', 2015. [Online]. Available: http://www.xilinx.com/products/silicon-devices/fpga/virtex-7.html. [Accessed: 10- Nov- 2015].

[6] T. Kazaz, M. Kulin, and M. Hadzialic, "Design and Implementation of SDR Based QPSK Modulator on FPGA," *IEEE 36th International Convention of Information and Communication (MIPRO 2013)*, pp. 513–518, May 2013.

[7] Y. Li, M. Li, Y. Poo, J. Ding, M. Tang, and Y. Lu, "Performance analysis of OOK, BPSK, QPSK modulation schemes in uplink of ground-to-satellite laser communication system under atmospheric fluctuation," *Optic Communication*, vol. 317, pp. 57–61, 2014.

[8] W. Wu, "A FPGA-based 5 Gbit/s D-QPSK Modem," *Thesis for the degree of Master of Science in Technology, Department of Signal and System, Chalmers University of Technology*, 2011.

[9] S. O. Popescu, a. S. Gontean, and D. Ianchis, "QPSK Modulator on FPGA," *SISY 2011 - 9th Int. Symp. Intell. Syst. Informatics, Proc.*, pp. 365–370, 2011.

[10] N. C. Shivaramaiah, A. G. Dempster, and C. Rizos, "Time-multiplexed offset-carrier QPSK for GNSS," *IEEE Transaction on Aerospace and Electronic Systems*, vol. 49, no. 2, pp. 1119–1138, Apr. 2013.

[11] G. S. Chandel and P. K. *Singh, "International Journal of Advanced Research in Computer Science and Software Engineering," vol. 4, no. 2, pp*. 382–386, 2014.

[12] N. A. Ranabhatt, S Agarwal and P. P. Gandhi, "RTL Design and Implementation of BPSK Modulation at Low Bit Rate," *International Journal of Engineering Research & Technology (IJERT)*, Vol. 2 Issue 2, pp.1-6, February 2013.

[13] F. Xiong, *Digital modulation techniques*. Boston, MA: Artech House, 2006.

[14] H. Hguyeu, E. Shwedyle, "A first course in Digital Communications, Cambridge University Press", New York, 2009.

[15] J. G. Proakis, M. Salehi, *DIGITAL COMMUNICATION*, 5th Edition. McGraw-Hill, 2008,pp. 161.

[16] H. Stern & S. Mahmoud, "*Communications Systems"*, Pearson Prentice Hall, 2004, pp. 283.

[17] Ni.com, "Overview of the NI USRP RIO Software Defined Radio - National Instruments", 2015. [Online]. Available: http://www.ni.com/white-paper/52119/en/. [Accessed: 10- Nov- 2015].

[18] "USRP-2943R - National Instruments", *Sine.ni.com*, 2016. [Online]. Available: http://sine.ni.com/nips/cds/view/p/lang/fi/nid/213002. [Accessed: 17- Mar- 2016].

[19] A. Klein, "*Stream ciphers," London, Springer*, 2013, pp. 17-19.

[20] J. E. Gilley, "Digital Phase Modulation, a Review of Basic Concepts" *Transcript International Inc.*, Aug. 2003.

[21] Designing Efficient Digital Up and Down Converters for Narrowband Systems Available: http://www.xilinx.com/support/documentation/application_notes/xapp1113.pdf. [Accessed: 12- Nov- 2015].

[22] S. P. Nicoloso, "An Investigation of Carrier Recovery Techniques for PSK Modulated Signals in CDMA and Multipath Mobile Environments," *Thesis for the degree of Master of Science in Technology, University of Blacksburg, Virginia,* June 1997.

[23] J. Costas, "Synchronous communications", *Proceedings of the IEEE*, vol. 90, no. 8, pp. 1461-1466, Aug. 2002.

[24] K. Mueller and M. Muller, "Timing Recovery in Digital Synchronous Data Receivers," *IEEE Transactions on Communication*, vol. 24, no. 5, pp. 516–531, May 1976.

[25] F. M. Gardner, "A BPSK/QPSK Timing-Error Detector for Sampled Receivers," *IEEE Transactions on Communication*, vol. 34, no. 5, pp. 423–429, May 1986.

[26] M. Mienkina, H. Lin, and A. Guzman, “Understanding the 16-bit ADC PGA in Kinetis K series,” Automotive and Industrial Solution Group, Freescale Semiconductor, *Application Note AN4568* , Sep. 2012.

[27] M. Rise, C. Dick, and f. Harris, “Maximum Likelihood Carrier Phase Synchronization in FPGA-Based Software Defined Radio,” *IEEE International Conference on Acoustic, Speech, and Signal Processing,* vol. 2, pp. 889892, May 2001.

[28] P. Rapaka, V. Babu, and G. J. Chitra, “FPGA Implementation of A BPSK Modem,” *International Journal of Engineering Research and Application,* vol. 3, no. 6, pp. 1976–1985, Dec. 2013.

[29] M. Xiao and T. Cheng, “Improved Implementation of Costas Loop for DQPSK Receivers Using FPGA,” *International Journal of Modern Engineering Research,* Vol. 3, pp. 1748-1755, May-June 2013.

[30] W. Xin, “Optimization of FPGA Design and Implementation of Timing Recovery in DVB-S2,” *International Conference on Communication, Circuits and Systems.* pp. 1265–1269, 2008.

[31] M. Ali, “Implementing Carrier Recovery for LTE 20MHz on Transport Triggered Architecture,” *Master Thesis , Tampere University of Technology*, December, 2011.

APPENDIX A: LFSR VHDL IMPLEMENTATION

```
library IEEE;
use IEEE.std_logic_1164.all;
use IEEE.numeric_std.all;
use IEEE.std_logic_unsigned.all;

-- Uncomment the following library declaration if using
-- arithmetic functions with Signed or Unsigned values
-- use IEEE.NUMERIC_STD.ALL;
-- Uncomment the following library declaration if instantiating
-- any Xilinx leaf cells in this code.
--library UNISIM;
--use UNISIM.VComponents.all;

entity LFSR is
    Port ( CLK : in STD_LOGIC;
           RST : in STD_LOGIC;
           Seed : in STD_LOGIC_VECTOR (15 downto 0);
           DecimateOrder : in STD_LOGIC_VECTOR (7 downto 0);
           Reg : out STD_LOGIC_VECTOR (15 downto 0);
           BitStream : out STD_LOGIC);
end LFSR;

architecture Behavioral of LFSR is
signal Dec_Clk: std_logic := '0';
signal TempReg: std_logic_vector(9 downto 0):= ( others => '0');
signal Initialized : std_logic:='0';
begin

process(CLK)
variable CNT: std_logic_vector(31 downto 0):= ( others => '0');
begin
if rising_edge(CLK) then
    CNT := CNT + 1;
    Dec_Clk <= CNT( conv_integer(DecimateOrder));
end if;
end process;

process(Dec_Clk)
begin
if rising_edge(Dec_Clk) then
    Reg(9 downto 0) <= TempReg;
    BitStream <= TempReg(0);
    if Initialized = '1' and RST = '0' then
        TempReg(9 downto 1) <= TempReg(8 downto 0);
        TempReg(0) <= TempReg(9) xor TempReg(6);
    else
       Initialized <= '1';
       TempReg <= Seed(9 downto 0);
    end if;
end if;
end process;

end Behavioral;
```

APPENDIX B: GARDNER SAMPLER VHDL IMPLEMENTATION

```
library IEEE;
use IEEE.std_logic_1164.all;
use IEEE.numeric_std.all;
use IEEE.std_logic_signed.all;
--use IEEE.std_logic_arith.all;

-- Uncomment the following library declaration if using
-- arithmetic functions with Signed or Unsigned values
--use IEEE.NUMERIC_STD.ALL;

-- Uncomment the following library declaration if instantiating
-- any Xilinx leaf cells in this code.
--library UNISIM;
--use UNISIM.VComponents.all;

entity GardnerPreBlock is
    Port ( CLK : in STD_LOGIC;
           DDSCLK_Sig: in STD_LOGIC_VECTOR (15 downto 0);
           SampleCLKOut : out STD_LOGIC;
           QPSK_Flag : in STD_LOGIC;
           SigInIQ : in STD_LOGIC_VECTOR (31 downto 0);
               EarlyI : out STD_LOGIC_VECTOR (15 downto 0);
               CurrenI : out STD_LOGIC_VECTOR (15 downto 0);
               LateI : out STD_LOGIC_VECTOR (15 downto 0);
               EarlyQ : out STD_LOGIC_VECTOR (15 downto 0);
               CurrenQ : out STD_LOGIC_VECTOR (15 downto 0);
               LateQ : out STD_LOGIC_VECTOR (15 downto 0);
           SigOutIQ : out STD_LOGIC_VECTOR (31 downto 0));
end GardnerPreBlock;

architecture Behavioral of GardnerPreBlock is
signal DecCLK,DecCLKFine,DDSCLK,SampleCLK:  STD_LOGIC:='0';
signal TempEarlyI,TempCurrenI,TempLateI :  STD_LOGIC_VECTOR(15 downto 0)
:=(others => '0');
signal TempEarlyQ,TempCurrenQ,TempLateQ :  STD_LOGIC_VECTOR(15 downto 0)
:=(others => '0');
begin

--Schmitt trigger
process(CLK)
variable State : STD_LOGIC :='0';
begin
    if rising_edge(CLK) then
            if DDSCLK_Sig < X"C000" then
                  DDSCLK <= '0';
            elsif DDSCLK_Sig > X"4000" then
                  DDSCLK <= '1';
            else
                  DDSCLK <= DDSCLK;
            end if;
    end if;
end process;

process(DDSCLK)
variable CNT_CLK:STD_LOGIC_VECTOR (7 downto 0) := (others => '0');
begin
    if rising_edge(DDSCLK) then
       CNT_CLK := CNT_CLK + 1 ;
       DecCLK <= CNT_CLK(1);-- to sample out
       SampleCLK <= CNT_CLK(1);
       DecCLKFine <= CNT_CLK(0); -- to separate early late samples
    end if;
end process;
```

```
SampleCLKOut <= SampleCLK;
process(SampleCLK)
begin
    if rising_edge(SampleCLK) then
            SigOutIQ <= SigInIQ;
            EarlyI <= TempEarlyI;
            CurrenI <= TempCurrenI;
            LateI <= TempLateI;
            if QPSK_Flag = '1' then
                  EarlyQ <= TempEarlyQ;
                  CurrenQ <= TempCurrenQ;
                  LateQ <= TempLateQ;
            else
                  EarlyQ <= (others => '0');
                  CurrenQ <= (others => '0');
                  LateQ <= (others => '0');
            end if;
      end if;
end process;

process(DecCLKFine)
begin
    if rising_edge(DecCLKFine) then
        TempEarlyI <= SigInIQ(31 downto 16);
        TempCurrenI <= TempEarlyI;
        TempLateI <= TempCurrenI;
        TempEarlyQ <= SigInIQ(15 downto 0);
        TempCurrenQ <= TempEarlyQ;
        TempLateQ <= TempCurrenQ;
    end if;
end process;

end Behavioral;
```

APPENDIX C: SOFTWARE INTERFACE

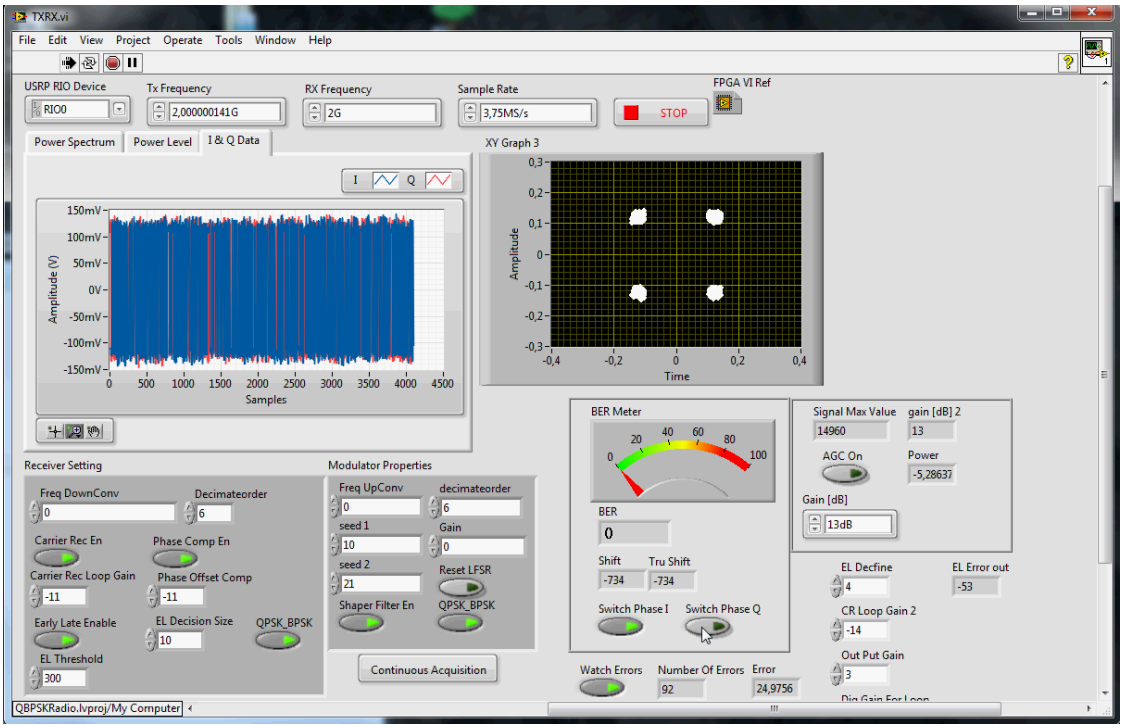

Figure 64. *The Designed Software Interface.*