# Implementation of a BPSK Transceiver for use with the University of Kansas Agile Radio

by

**Ryan Reed**

Bachelor of Science, University of Kansas, Lawrence, Kansas, 2004

Submitted to the Department of Electrical Engineering and Computer Science and the Faculty of the Graduate School of the University of Kansas in partial fulfillment of the requirements for the degree of Master of Science in Electrical Engineering

Thesis Committee

______________________________
Chairperson: Dr. Gary Minden

______________________________
Dr. David Petr

______________________________
Dr. Alexander Wyglinski

Date Accepted: May 1, 2006

The Thesis Committee for Ryan Reed certifies

That this is the approved version of the following thesis:

Implementation of a BPSK Transceiver

In a Xilinx FPGA for use with KUAR

Committee:

____________________________
Chairperson

____________________________

____________________________

Date approved: ____________________________

A designer knows he has achieved perfection not when there is nothing left to add, but when there is nothing left to take away.

- Antoine de Saint-Exupéry, *Wind, Sand and Stars*

# Abstract

The purpose of this study was to design a binary phase-shift-keying (BPSK) transceiver specifically for use in the University of Kansas Agile Radio (KUAR). The resulting transmitter communicates 1 Mbaud of data over a 5 MHz carrier, sampling at 80 Msps. The receiver compensates for frequency and phase errors presented by Doppler shift and bit-time errors. Simulations show an error rate performance of 0.002835 at 10 dB of Eb/No. The resulting design could be used with other transceivers by making slight alterations.

# Acknowledgements

First, I'd like to thank Dr. Gary Minden, my advisor and mentor during my graduate career at the University of Kansas. His ingenuity and guidance are so valuable to everyone who works on the Flexible Wireless Systems for Rapid Network Evolution project. I'd also like to thank Dr. Joseph Evans, who developed the proposal for this project and received Grant AN-0230786 from the National Science Foundation.

Dr. Glenn Prescott, who was the first person to ignite my interest in engineering, inspired me to learn more about digital signal processing and communication systems. It was my pleasure to have him as an advisor during my undergraduate career.

Dr. David Petr showed me how to take signals and create communications systems. Because of his teaching, I decided to become a communications engineer. His creativity and inspiration both in the classroom and out were very encouraging to me.

I owe a very large thank you to Dr. Alex Wyglinski. His personality, encouragement, and wisdom were inspiring. Our meetings were invaluable to my work. I'd also like to thank Dr. Erik Perrins, who gave me a good starting point for my research.

My experiences as an undergraduate student at the University of Kansas showed me the excellence of the program and helped me to appreciate the great opportunities that this wonderful school provides to all of its students. I would like to thank the National Science Foundation for funding Dr. Minden and the Flexible Wireless Systems for Rapid Network Evolution Wireless Project.

Finally, thanks to my co-workers in the lab: Jordan Guffey, Ted Weidling, Rory Petty, Leon Searl, Dan DePardo, Tim Newman, Brian Cordill, Levi Pierce, Megan Lenherr, Rakesh Rajbanshi, Qi Chen, Anu Veeragandham, and Wes Mason who were always helpful with the project. Thanks to my mom for her constant support and love;

thanks to my dad for being a wonderful role model; thanks to my sister for bearing the burden of older sibling; thanks to Jennifer for her support and assurance; and thanks to Kodi, the greatest dog in the world.

# Table of Contents

# List of Figures

## List of Tables

# List of Symbols and Acronyms

ADC – Analog-digital converter

ASIC – Application-specific integrated circuit

AFC – Automatic frequency control

AM – Amplitude modulation

BER – Bit-error rate

BPSK – Binary phase shift-keying

CPH – Control Processor Host

CRC – Communications Research Center

DARPA – Defense Advanced Research Projects Agency

DSP – Digital signal processing

FCC– Federal Communications Commission

FIR – Finite impulse response

FPGA – Field programmable gate array

GHz – Gigahertz

GUI – Graphical user-interface

IF – Intermediate frequency

IIR – Infinite-impulse response

IP – Intellectual property

JTRS – Joint Tactical Radio System

KUAR – University of Kansas Agile Radio

Mbaud – Megabaud

MHz - Megahertz

MSPS – Mega-samples per second

NCO – Numerically-controlled oscillator

PC – Personal computer

PCMCIA – Personal Computer Memory Card International Association

PLL – Phase-locked loop

PSK – Phase-shift-keying

RAM – Random access memory

RC – Resistor, capacitor, inductor components

RF – Radio frequency

SCA – Software Communications Architecture

SDR – Software-defined radio

SFDR – Spurious-free dynamic range

SNR – Signal-to-noise ratio

SSB – Single side-band

SSB-AM – Single side-band amplitude modulation

SSL – Symbol synchronization loop

UNII – Unlicensed National Information Infrastructure

XG – Next Generation

# CHAPTER 1 A BPSK TRANSCEIVER

## *1.1 Introduction*

The thesis describes the design of a binary phase-shift-keying (BPSK) transceiver specifically for use in the University of Kansas Agile Radio (KUAR), a test-bed for software-defined radio (SDR) development. The design is simulated using Matlab Simulink and implemented using Xilinx ISE targeted for the Xilinx Field Programmable Gate Array (FPGA), a reconfigurable processor. The scope of the research is the following.

This receiver design is based on the single side-band (SSB) receiver proposed by Costas in 1956 [1]; the majority of the research in this thesis is in the area of improvements to his original design. An overview of the research in the area of digital communications and digital signal processing is presented with respect to the Costas loop and bit-time recovery algorithms.

The thesis shows the requirements to build a BPSK transceiver with the design constraints as follows:

- The transmitter must communicate 1 MBaud of data over a 5 MHz carrier, sampling at 80 Msps. The receiver must compensate for two problems: the frequency and phase errors presented by Doppler shift and bit-time errors. The thesis will examine approaches to solve for carrier frequency and phase errors. Furthermore, existing bit-time recovery algorithms will be explored for use within an SDR. This design will not cover other problems such as multi-path, fading, channel estimation, or gain control.

- Use as few resources as possible, while maintaining performance. Components are being developed by other KUAR researchers and there must be enough resources for the implementations to run concurrently.

- Other modulation schemes are yet to be implemented on the KUAR. The research presented shows how different transceivers might be implemented by making slight alterations to this design.

The thesis shows the steps that were required to port the design from a mathematical simulation to a firmware implementation. While most of the porting process was one-to-one, some algorithms were modified for an improvement in FPGA resource efficiency.

The thesis provides the reader with an overview of the progress made in the field of SDR. The implementation of a BPSK transceiver in the KUAR is one step toward developing a small form factor SDR.

## *1.2 Thesis Organization*

The thesis is organized such that in Chapter 1 there is a description of the thesis goals and an outline of the political and engineering developments that oversee and propel the innovations in SDR. In Chapter 2, there is an overview of various implementations of SDR. Also there is a discussion of the basic components of an SDR and the technologies for synchronization. It also examines known algorithms for solving the problems described in the scope of this research. In Chapter 3 there is a review of the KUAR system as a test bed for research in SDR. It includes a description of the hardware and a brief overview of the system's characteristics and specifications.

Chapter 4 contains a summary of the simulation of the selected algorithms used in the research of this problem. The simulations were completed in Matlab Simulink to predict how the hardware implementation will behave. It concludes with a prediction of the signal-to-noise ratio versus bit-error rate (SNR vs. BER) plot.

Chapter 5 describes the method of porting the previously simulated algorithms to the firmware construction tool, Xilinx ISE. Finally, Chapter 6 describes the results in detail and explains how this research met the scope of requirements. It also describes the steps to expand on this research for future work.

## *1.3 Motivation to Build Software-Defined Radios*

The current spectrum allocation structure implemented by the Federal Communications Commission (FCC) alloates the entire spectrum for specific activities, including un-regulated regions for WiFi, protected regions for mobile phones, as well as protected regions for municipal services. However, recent findings by the Spectrum Policy Task Force indicate that the FCC will need to revise its policies in order to meet the growing demand for spectrum. The task force met with several entities in 2002 to promote innovations for using the spectrum more efficiently, more effectively, and more responsibly. [2]

The task force suggested that the following changes need to occur: the majority of the radio spectrum could be used much more effectively, outdated policies should be replaced with more efficient dynamic systems, and the given command and control regulations should be replaced by regulations which encourage flexibility, robustness, efficiency, and incorporation of better technologies. [2]

Rather than doing away with the current “command and control” regulations, the task force would rather see sections of it reserved for special circumstances, such as public works or treaty compliance. The majority of the spectrum would be broken into two groups: exclusive use and commons. Exclusive spectrum would assign a range of frequencies in a given geographic area to an entity and would protect these transmissions from interference. The commons spectrum would be an area for any user to transmit a given set of technology standards and etiquette, but would provide no assurances to the user. [3]

Under the Defense Advanced Research Projects Agency (DARPA) Next Generation (XG) program, researchers have found that even within these coveted regions of spectrum, the power ratio is very small, meaning that most frequencies are unused. Researchers perform these measurements dynamically, such that a section of spectrum is sampled over a long period of time to form a large data set. It is easy to see from this data set that large sections of spectrum are unused for a large period of time. Thus, the focus of the following research should not only go towards finding narrowband methods of communications, but also towards time-sharing the spectrum. [4]

One of the goals of the spectrum research project is to document these regions of unused spectrum with time to form a database. If a Software Defined Radio (SDR) system were incorporated with this database, spectral resources could be allocated for both time and bandwidth.

One of the goals of the XG program is to develop a hardware platform that would enable the dynamic access of spectrum, allowing users to maintain communication links by changing radio operating parameters and not interfering with other users, such as the license holders. [5]

While DARPA is primarily focused on creating the means for the military to communicate anywhere in the world, the FCC's Spectrum Policy Task Group is attempting to implement similar technologies in the United States. In a similar fashion to DARPA XG, flexible communications systems would allow users to identify unused portions of spectrum and possibly borrow them from an assigned user. This leads to the concept of an agile radio, which is a device capable of quickly changing its operating parameters, such as carrier frequency, bandwidth, or modulation scheme. The device also needs to be capable of selecting a transmission style appropriate for a given channel characteristic set. [3]

The SDR Forum has outlined its vision of the capabilities of these next generation radios into four tiers. Most of these radios have not been created yet; however, this is a good vision of what will likely be developed chronologically. The most basic radio is the “Tier 0 Hardware Radio”, which is capable of changing system parameters by user operation with switches, dials, buttons, or physical interaction with the hardware. A “Tier 1 Software-Controlled Radio” would be capable of changing its characteristics by user interaction with software. Instead of dials and switches, it would have soft dials and switches, perhaps in a graphical user-interface (GUI). A “Tier 2 Software Defined Radio” has a broad operational radio frequency (RF) bandwidth and performs the majority of its Intermediate frequency (IF) operations in a processor. It would be capable of operating over a wide bandwidth using a large variety of modulations and modify a subset of its software to perform the user’s commands. This would be done without reloading the entire system and without handling any hardware aside from the RF front end. The “Tier 3 Ideal Software-Defined Radio” improves on the Tier 2 radio by implementing more aspects of the RF front end in software, such as gain control or mixing. The final “Tier 4 Ultimate Software Radio” has a very futuristic set of capabilities: uses very little power, requires no external antenna, can operate at any frequency, performs all transactions by a single connector, switches operational parameters in milliseconds, uses GPS, stores money in the smartcard format, provides video via satellite or terrestrial broadcasts, and stores several programs. [6]

The goal of these theoretical radios is to work under the guidelines proposed by the Spectrum Policy Task Force and to borrow unused sections of spectrum. Through current research, researchers are recording and statistically analyzing spectrum measurements to provide a potential database of unused spectrum at given times of day.

The KUAR is designed to meet the criteria of a Tier 2 software-defined radio and was created to research wireless communications. It is capable of communicating using "many modulation schemes, media access protocols, and adaptation mechanisms." [7] As an experimental device, future implementations could provide valuable information to DARPA or the Spectrum Task Force.

# CHAPTER 2 Background

## *2.1 Overview of Implementations of Software-Defined Radios*

Faster microprocessor speeds have enabled the telecommunications industry to push radio frequency (RF) hardware further back in the receiver chain and also has allowed processors to handle more of the reception and detection process. These advances in microprocessor architecture also drive the advances in alternative processing, such as Application Specific Integrated Circuit (ASIC) processors, Field Programmable Gate Array (FPGA) processors, and digital converters so that this RF integration is possible. The technology in the field of FPGA processors has advanced to enable the innovation of the SDR. [8]

The integration of RF algorithms in microprocessors allows the functionality of the microprocessor to be integrated with the RF section. The advantage of using microprocessors in a software-defined radio (SDR) is that multiple operations can be performed within a flexible architecture. For example, instead of simply using the processor to take over some of the intermediate frequency (IF) conversion and detecting a signal, the processor can change algorithms in order to detect several types of signals with different modulation schemes and frequency characteristics. Secondly, FPGA and Digital Signal Processing (DSP) processors have a chance of being forward compatible. Their firmware can be updated if an unforeseen change occurs in the device's operating requirements, so that it is not necessary to replace the whole unit. For example, assume that all the mobile phone companies agreed to use one common modulation scheme. An SDR would be capable of downloading the new algorithms and would still be able to operate in those bands. Hardware would likely not have to be replaced in an SDR. [9]

Until now, most development in SDR has been in base station improvements. [52] This is primarily due to the immense power necessary to operate an SDR. Several processors, amplifiers, and converters are necessary to create an SDR. Some of the hardware innovations are the following. The University of California at Berkeley has developed a system called the Berkeley Emulation Engine (BEE), which uses 20 FPGA processors and is capable of DSP algorithms. It fits in a box about the size of a single-drawer filing cabinet. [10] Vanu, Inc. has developed a fully operational base station for Cingular Wireless. Furthermore, they have integrated an SDR with very few applications into a Personal Data Assistant (PDA). [11, 12, 13] Portland State University has recently developed a software defined radio, which is about the size of a laptop computer, based on GNU Radio. Their modulation sets include GPS, 802.11, and FM. [14] Spectrum Signal Processing has also developed a software-defined radio platform. Offering a variety of systems, the smallest is about the size of a desktop computer and the largest is about the size of a mini-refrigerator. Designed for base-station operations, they house several processors, digital converters, and FPGA processors giving the user a very broad operating bandwidth. [15] Finally, ISR Technologies has developed a unit about the size of a desktop computer for Joint Tactical Radio System (JTRS) communications. Using Xilinx Virtex-4 FPGA processors, it complies with the Communications Research Center (CRC) Software Communications Architecture (SCA) and Radio Manager Platform. It is primarily a tool for engineers to develop JTRS-oriented firmware. [16]

## *2.2 Similar Work*

This thesis will examine several analog and digital algorithms for detecting BPSK signals, locking on to the carrier and phase, and synchronizing the symbol. Many of these previously analog designs have been translated to the digital domain. [17, 18] These digital designs vary on their intended implementations: some are for digital circuits, some are for ASIC, and some are for FPGA. These are outlined in the Tables 1 and 2.

The Costas loop takes RF input from a complex source and analyzes the real and imaginary parts separately. In the traditional design, a PLL analyzes the data from to two parts to make a decision about how to adjust the phase and frequency. This decision affects both arms in an attempt to synchronize to the carrier. In the case of the KUAR, the selected algorithm will compensate for frequency and phase errors.

**Table 1: PLL filter algorithms**

| **Reference** | **Basic Method** | **Intended Implementation** |
|---|---|---|
| Berner [23] | Second-order integrator | Digital system |
| Cahn [22] | Automatic Frequency Control | RC components |
| Mirabbasi [21] | Third-order Bessel IIR filter | RC components |
| Rice [19] | Traditional second-order IIR filter | Digital system |
| Statman [20] | Two second-order IIR filters | Digital and RC components |

Just as the carrier needs to be synchronized, the data also needs to be synchronized to ensure the maximum probability of estimation. This will be handled by a symbol timing algorithm.

**Table 2: Symbol timing algorithms**

| **Reference** | **Basic Method** | **Intended Implementation** |
|---|---|---|
| Gardner [27] | Interpolation and timing circuit | Digital system |
| Georghiades [25] | Early-late gate method | Digital system |
| Gervargiz [17] | Several integrate and dump filters | Digital system |
| Hang [29] | Two Gardner algorithms | Digital system |
| Holmes [18] | Filter, square, and filter | RC components |
| Hwang [30] | Interpolator and recursive Costas loop | Digital system |
| Judd [26] | Modified early-late gate algorithm | Digital system |
| Liu [28] | Interpolation and rate conversion | Digital system |
| Pomalaza-Ráez [24] | Tree-search algorithm | Digital system |

With the recent emergence of the SDR as a viable communications tool, there have also been several implementations of SDR hardware besides KUAR. The thesis will examine some of these devices for comparison. These radios are discussed in greater detail in Section 2.1, but a summary is shown in Table 3.

**Table 3: Survey of SDR hardware systems**

| Device | Key feature | Approximate size |
|---|---|---|
| JTRS SDR Kit [16] | JTRS Testbed | Desktop computer |
| KUAR [7] | Extremely mobile | Alarm clock |
| PSU SDR [14] | Rocket telemetry system | Laptop PC |
| SDR-3000 [15] | Compact-PCI, SCA compliant | One-drawer filing cabinet |
| UC-BEE [10] | Several FPGA processors | One-drawer filing cabinet |
| USRP [54] | Open source project | Laptop PC |
| Vanu Basestation [11] | Currently being used by Cingular | Unknown |
| Vanu iPaq [12, 13] | Handheld | PDA |

## 2.3 Technologies for Synchronization

Synchronization occurs when the timing of the transmitter is known to the receiver. This is an important step in communications as unsynchronized transmissions can have detrimental affects on the estimation process, which will be shown in Section 4.3.1.1. In the scope of this research, two synchronization algorithms need to be implemented. One will recover the carrier from frequency and phase errors; the other will recover the symbol from the transmission and is discussed later, in Section 2.4.

This design is based on the Costas loop, first implemented in 1956 to "take full advantage of AM communications." [1] There was a movement in the military to move from AM to SSB-AM due to the capability of SSB-AM to nearly double the number of channels in a given bandwidth.

Although his design was primarily for synchronous AM systems, the same design has been proven effective for demodulating binary phase shift-keying (BPSK) signals, as BPSK is essentially synchronous AM with only two amplitudes. Costas went on to show that it would be very difficult to force his receiver to see a null, a lack of signal power caused by a phase misalignment between the receiver and transmitter. Phase lock is achieved through a loop filter, which drives the local oscillator to the correct frequency and phase. Costas showed that this receiver design was significantly more robust than a SSB system in the scope of difficulty to jam, signal-to-noise ratio (SNR) required to accurately detect a message, and receiver complexity.

Rice proposed a digital version of the Costas Loop using a traditional second-order PLL and two frequency synthesizers. The algorithm requires two synthesizers due to the necessity to synchronize the carrier. The first modulates the carrier to baseband. The second is constantly being updated by the result of the loop filter. With the write enable always active, the second synthesizer can only perform fine tuning as the frequency value will never increment. Using write enable always sets the frequency starting point to zero. This makes it appropriate for designs with small frequency offsets. [19]

While the overall structure of the Costas loop is fairly simple, the loop filter has been the center of debate ever since the design's publication. Several engineers have proposed methods of improving various aspects of the filter's performance. Statman and Hurd propose using two oscillators and two filters. In this way, one filter acts as an estimator. In their experiment, a least-squares estimator was used. This is coupled with a predictor, which is used to remove the time delay associated with these digital filters. They suggest that their approach for solving for the loop filter is better than the trial-and-error approach. However, the design is more appropriate for an ASIC than an FPGA as it assumes certain components are built in hardware and others in software. Implementing this design in an FPGA is equivalent to building two third-order infinite-impulse response (IIR) filters. [20]

Mirabbasi and Martin propose a method of designing third-order or higher IIR filters that have similar bandwidths to the lower order filters. This is accomplished by using Bessel filter coefficients, which makes the design expandable to higher-order filters. [21]

Cahn proposes a method of increasing the error detection and correction by using two separate filters and calls the system automatic frequency control (AFC). This new system expands the bandwidth of the Costas error detection by tracking the frequency even if it leaves the operating bandwidth of the Costas filter. Cahn adjusts his filter to compensate for false lock on a sideband. [22]

The most promising alternative to the traditional loop filter proposes implementing a second-order IIR filter in the parallel form and provides the flexibility of being reconfigurable. The PLL filter proposed by Berner, Layland, and Kinman is targeted for spacecraft. [23] Another implication of being spacecraft based is the high mobility, but the Doppler shift in this system is potentially much higher than the scope of the KUAR system. Both perfect and imperfect integrators are examined and the performance is compared under the presence of Gaussian noise. Their findings show that a perfect integrator will offer better tracking while an imperfect integrator will drift away from the best-lock frequency less. This implementation looks promising as it uses one less coefficient (multiplier) than the traditional implementation. However, they do not compare a traditional IIR loop filter to any of their proposed filters.

A summary of the designs researched is provided in Table 4.

**Table 4: Tradeoffs of discussed PLL filters**

| Reference | Advantages | Disadvantages |
|---|---|---|
| Berner [23] | Small | Very wide bandwidth |
| Cahn [22] | Good pull-in range | Third-order, large bandwidth |
| Mirabbasi [21] | Easy design | Third-order |
| Rice [19] | Easy design, small | Mediocre performance |
| Statman [20] | Very good performance | Large, uses external components |

The smallest designs researched are the Rice and Berner proposals as they are both of the second-order. However, the Rice algorithm has a much smaller bandwidth, thus it will pull in less noise and be less susceptible to errors. The Berner algorithm was designed for spacecraft, where speeds will have a much larger effect on the Doppler shift. If there is a desire to place the KUAR on a spacecraft, the Berner design should be reconsidered.

## 2.4 Technologies for Bit-Time Recovery

Bit-time recovery algorithms analyze streams of data in order to estimate the transmitted message. Two algorithms on which this research focuses are the Gardner algorithm and the early-late algorithm. The data-transmission tracking loop was not considered as it has no benefits over the early-late algorithm. [24] A few less common algorithms were also examined, such as the tree search algorithm and the filter and square bit synchronizer. [24, 18] It is with one of these algorithms that the data will be synchronized to provide the best estimate of the transmitted message.

The Early-Late Gate algorithm determines if the algorithm has sampled the bit stream too early, on time, or too late. The algorithm uses three chronologically sequential samples to form the limit approaching the center point from both sides. If the two limits differ, the algorithm assumes it has sampled in the correct position. However, if the two limits are the same (both increasing or both decreasing), then the algorithm uses the center point to gate one sample faster or slower. [25]

One proposal to improve this method of bit synchronization has been made by Judd. In this method, the first half of a bit period is integrated on one arm and the second half of a bit period is integrated on a second arm. By comparing the energy in the two arms, the algorithm can determine if the sampling needs to occur faster or slower. Furthermore, the integration of the late gate becomes the bit estimate. [26]

The Gardner method is another means of estimating the transmitted bit. The result of an integration of the sampled bits becomes the input to an interpolator. The filtered output of this interpolator becomes the bit estimate. The output of the filter is analyzed by a timing circuit. This is done to determine how to better estimate the value of the bit. The result of this calculation becomes the control for the interpolator. [27]

This method has been the topic of several papers since Gardner first proposed it in 1993. Some investigators analyze interpolation methods to determine which method is the best to use. [28] Others propose filtering methods for better estimating the symbol. [29] Although this method and proposed changes to it have the advantage of resolving errors very quickly, the timing circuit is very complicated and difficult to build.

Liu and Willson propose a method of reducing digital resources using the Gardner method by altering the standard interpolator structure, a “Farrow” cubic interpolator. In their proposal, the bit samples are up-sampled and linearly interpolated. The up-sampling is accomplished in two stages. First, the samples are interpolated by two; next, the samples are interpolated at a variable rate, which is controlled by the feedback loop. This method saves two multipliers. They claim that performance is increased because errors are reduced by a factor of five. This method was geared towards implementation in

ASIC, where multipliers might be considered more costly than implementation in FPGA. [28]

Hang and Renfors propose a method of adjusting the gain in the loop filter to reduce jitter while increasing the time to convergence. This is accomplished by using an adaptive filter. When the gain is large, convergence is fast but jitter is high. Likewise, when the gain is small, convergence is slow but jitter is low. Thus, the filter gain starts high and decreases once the filter senses convergence has occurred. [29]

There can also be differences within the Costas Loop. Hwang and Chu propose using a Gardner algorithm to recover the symbols before passing them into the Costas Loop for phase recovery. The bits from the analog-digital converter (ADC) pass through a matched filter before being interpolated and decimated with a Cubic Farrow Interpolator. The results then are frequency and phase corrected with a digital Costas Loop. [30]

As an alternative to both the Gardner and Early-Late Gate algorithms, Gevargiz proposes using a Symbol Synch Loop (SSL) to synchronize his symbol timing. This loop uses samples out of a matched filter as well as samples out of a subsequent integrate and dump filter. The result of this loop triggers the integrate and dump filter to provide the best estimate for the rest of the system, a Costas Loop and decoder. The loop attempts to reduce errors produced by the Costas Loop sample frequency, Costas Loop sample phase, and estimated SSL timing error. This is done through differentiation, integrate and dump, a loop filter, a numerically-controlled oscillator (NCO), and a windowed integrate and dump filter. [17]

Another alternative to the Gardner and Early-Late Gate algorithms is proposed by Pomalaza-Ráez and Mohan. This method applies tree search algorithms, which they claim will reach steady state faster than the Early-Late Gate algorithm. They also determine that the *(M, L)* search algorithm is a better choice than either the Viterbi or

Ungerboek algorithms. The algorithm dictates that *M* samples of the data set are further broken into *L* subsets. The *L* subsets are analyzed and *M* samples are chosen as the best candidates; the remaining samples are discarded. This process continues until the smallest data set has been analyzed. [24]

The final alternative to the Gardner and Early-Late Gate algorithms, called the Filter and Square Bit Synchronizer, is proposed by Holmes. The goal of this algorithm is to reduce resources consumed while providing performance comparable to the Early-Late Gate algorithm and Digital Transition Tracking loop. It consists of a one-pole RC filter, a mixer, and a PLL and performs only 2 percent worse than the Early-Late Gate algorithm. He admits that this algorithm could have trouble tracking in multi-rate systems. [18]

A summary of the advantages and disadvantages of each synchronization algorithm is shown below in Table 5.

**Table 5: Symbol timing algorithms**

| Reference | Advantages | Disadvantages |
|---|---|---|
| Gardner [27] | Accurate and fast response | Very difficult to design |
| Georghiades [25] | Simple design, few resources | Average performance |
| Gervargiz [17] | Very simple to build | Uses several resources |
| Hang [29] | More accurate Gardner algorithm | Uses several resources |
| Holmes [18] | Very few resources | Sub-optimal performance |
| Hwang [30] | More efficient than Gardner algorithm | Difficult to design |
| Judd [26] | Easier than early-late algorithm | Performance not quantified |
| Liu [28] | More efficient than Gardner algorithm | Clock rate unattainable |
| Pomalaza-Ráez [24] | All digital design | Sub-optimal performance |

The early-late gate algorithm proposed by Georghiades will be implemented in this design. It was selected due to the ease of construction and manipulation between different data rates. This will be discussed further in Section 5.2. The algorithm also uses few resources and provides performance better than the filter and square bit synchronizer and tree search algorithms. However, the Gardner methods provide better performance than the early-late gate algorithm at the expense of resources and complexity.

# CHAPTER 3 KUAR – Experimental Test-Bed

## 3.1 Introduction

The University of Kansas Agile Radio (KUAR) consists of five primary units: the RF section, the digital board, the Control Processor Host (CPH), the battery board, and two antennas. All of the units are contained in a metal box which measures 7 inches high, 5 inches deep, and 2 inches wide. The antennas are attached on either side of the metal box by a boom. The RF front end is responsible for accurately moving a section of bandwidth from 5.25-5.85 GHz down to baseband and vise versa for modulation. The primary function of the digital board is to receive and detect signals at baseband from 0 to 30 MHz bandwidth. The CPH unit's primary function is to control the digital board and to provide external communication to the user. The battery board's primary function is to provide power to the other units while managing the charge of the batteries. Finally, the antenna radiates transmissions and receives electromagnetic power. [7]

## 3.2 RF Front End

The current RF board is capable of modulating between baseband and a radiation frequency range of 5.25-5.85 GHz. Through a two-stage mixing process, complex modulation and demodulation is performed to translate to the baseband range of 0-30 MHz. A voltage controlled local oscillator, controlled by the CPH, is used to select the 30 MHz out of the radiated frequency range and an automatic gain control mechanism (AGC) sets the power to the optimal levels.

Physically, there are multiple RF sections thatinteract with the KUAR. This is necessary because the baseband bandwidth of the KUAR is only 30 MHz. Thus, in order for the KUAR to interact with other radios operating at other frequencies, different modules must be present, such as a 2.4 GHz module for communicating with WiFi or cordless phones; or a 5 GHz module, shown in Figure 1, for communicating in the

Unlicensed National Information Infrastructure (UNII) band of WiFi. However, for operating with no up-conversion, a shortwave section is being developed. The RF section is shielded as much as possible from the rest of the radio to prevent interference, especially from the digital board. A cutaway of the radio is shown in Figure 3.

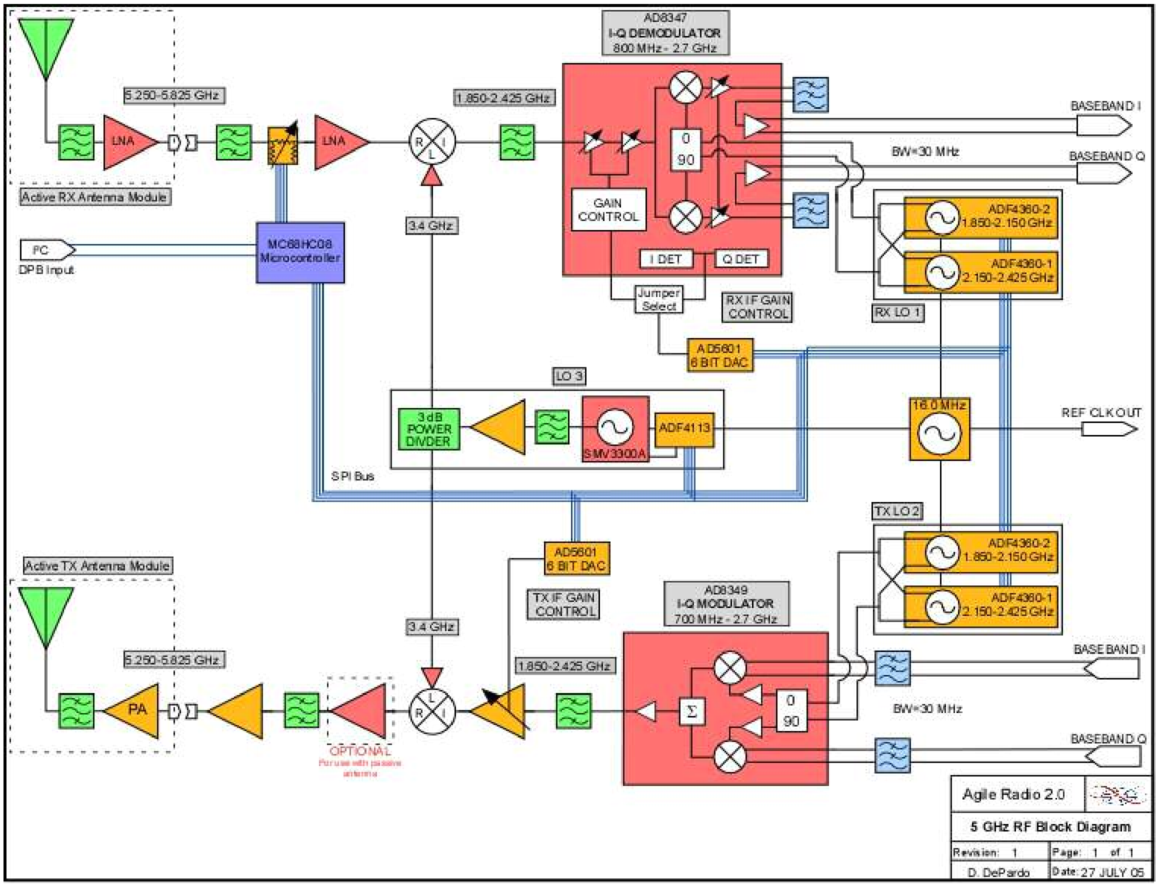


**Figure 1: Simplified block diagram of the 5 GHz RF section [32]**

Since the analog digital converters are located on the digital board, the connection between the digital and RF boards is the analog. Transmitted messages are converted from the digital domain to the analog domain by a 16-bit digital-analog converter, operating at 80 MHz. Received signals are filtered with a low-pass filter to 30 MHz before being converted to the digital domain by two 14-bit analog-digital converter (ADC) circuits, operating at 80 MSPS. The entire system operates with in-phase and quadrature-phase components before detection occurs.

## *3.3 Digital Board*

The digital board performs a great number of functions, including programming certain aspects of the RF section, modulating, demodulating, and detecting signals, storing data in RAM, and performing analog digital conversion.  The goal of the digital board is to perform several modulation schemes, which are provided by the CPH, and to switch very quickly between them.

The Xilinx Virtex-II Pro FPGA is component of the digital board.  This component provides the RF front end with the analog transmitted signal via two analog-digital converters and digitizes the received signal from the RF front end via one digital-analog converter, has a bank of 4 MB of RAM for storage, and holds the smaller control processor board, along with a few other specific functions.  All digital signal processing is handled in the FPGA, including modulating signals, demodulating signals, analyzing spectra, filtering, and estimating.  Most functions are implemented in logic slices, the primary fabric of FPGA processors; however, some functions are under development for implementation in the internal Power PCs.  This board generates a lot of heat, which must be monitored and controlled.  This is done via an on-board thermometer, which reports to the CPH, and by heat sinking.

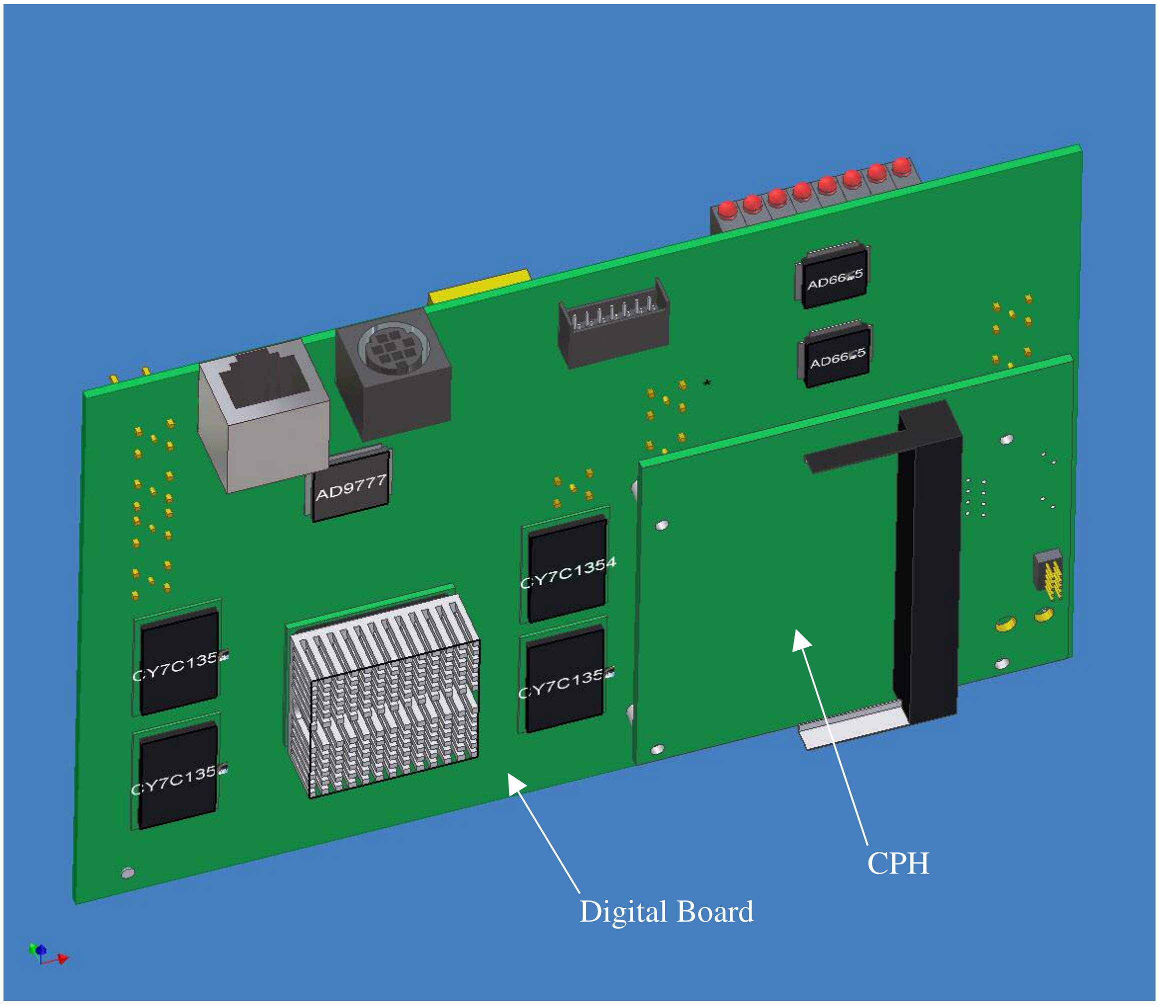


**Figure 2: Illustration of the CPH attached to the digital board [33]**

## *3.4 Control Processor Host*

The CPH is a Power PC which runs Linux. Its limited memory, 32 MB of SDRAM and 32 MB of flash, is primarily used to store different modulation schemes with which to load on to the FPGA on the digital board. "The CPH is designed to perform four basic functions: radio control; signal processing configuration management; execute adaptive algorithms; and interface with conventional networks." [7] This board physically snaps on to the digital board and connects through a 32-bit wide bus, allowing for several input/output connections. A diagram of these connected boards is shown in Figure 2. This processor, which is approximately 4 cubic inches in size, was designed to be a very small stand-alone computer. It has several connections, including Ethernet (10/100), serial, and a PCMCIA card. It is capable of storing around 20 bit files, which are used to program the FPGA. Also, it makes several digital connections to the digital board for communication with the FPGA, RAM, and other components.

## 3.5 Battery board

The battery board uses two Lithium-Ion battery packs to provide both digital and analog power to all components in the system. All components on the RF front end require analog power; the digital board uses a mix of both; and the control processor uses only digital power. Estimates show a battery lifespan of three hours from fully charged to empty. The batteries have a capacity of approximately 4.4 A-Hr at 11 Volts.

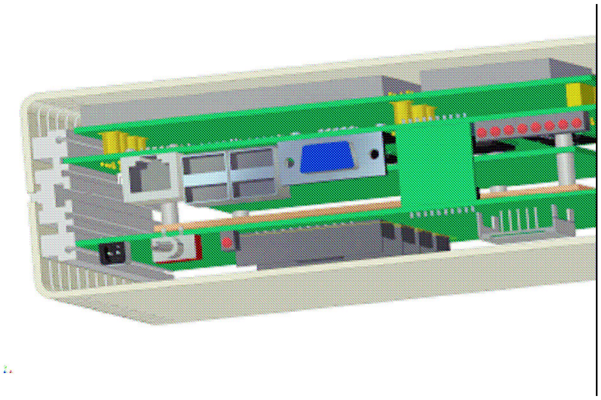


**Figure 3: Diagram of boards assembled in the KUAR case [33]**

The drive towards mobility constrains the design of the digital section. Designs must attempt to conserve power, and any modulation scheme implemented must be able to handle the Doppler shifts associated with the mobile system. Although estimated to be minimal these frequency shifts, described in section 4.3.1.3, could have detrimental effects on more complicated modulation schemes.

Since the batteries have a limited lifetime before exhaustion, the battery board has been designed with as much charging flexibility as possible. It can be charged from a wall socket or a car battery.  The ability to quickly switch between power sources is due to a bank of capacitors that keeps charge while a switch toggles.

# CHAPTER 4 SIMULATION OF BPSK TRANSCEIVER

## 4.1 Introduction

The first step in building the transceiver for the KUAR is to show how it operates mathematically. The transceiver is modeled in Matlab Simulink and consists of a BPSK transmitter, an additive white Gaussian noise (AWGN) channel, and a receiver. The system is designed with Xilinx intellectual property (IP) cores because this is the ultimate destination for the design. The receiver design is based on a modified Costas loop proposed by Dr. Michael Rice. [19]

For review, the goal of this research is to design a 1 MBaud BPSK transceiver with a carrier of 5 MHz and use algorithms to synchronize the carrier and symbol.

## 4.2 Transmitter

In the analog domain, multiplying the carrier by a positive or negative DC voltage is the best method of creating a BPSK signal for this design as a square symbol will use the fewest resources. While this is an option in the digital domain, a better approach for the KUAR is to change the lookup address of a direct digital synthesizer (DDS). The DDS is an IP Core provided by Xilinx that uses two registers to determine the address of a look-up table containing the value of the sine and cosine of the argument. The two registers used to calculate the address are the phase increment and phase accumulator. The phase increment is proportional to the desired output frequency and inversely proportional to the input clock. The phase accumulator is the primary factor in the output width of the signal. [31] The two methods are illustrated in Figure 4 and Figure 5, where $m(t)$ is the transmitted binary message.

Since a DDS is already necessary for the waveform output, few extra resources are required. The message bit serves as the address bit for a ROM containing the phase value, which is loaded into the DDS on every bit change. This method uses an equal number of slices as multiplier method. A ROM would still be necessary to generate a negative one for the symbol zero. The phase design also has the advantage of being easily expandable for M-ary Phase Shift Keying (PSK) designs. Thus, it is possible that the same load of the FPGA can be used for multiple versions of PSK modulation. Since expandability is one of the primary goals of the KUAR, this design uses the ROM method. That is, in order to add more phases, expand the ROM to contain more angles. Until the other simultaneously running operations are known to this transceiver in the FPGA, multipliers are considered a more valuable resource than slices.

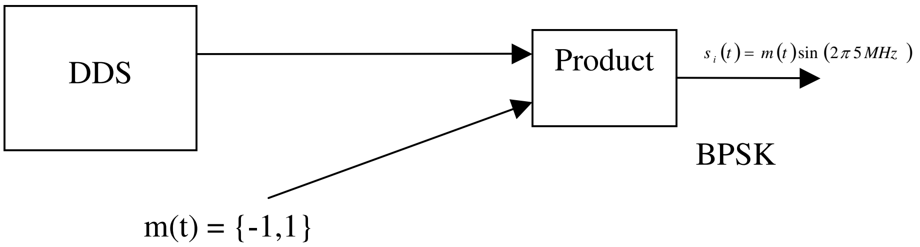


**Figure 4: Simplified block diagram of a BPSK transmitter created through multiplication**

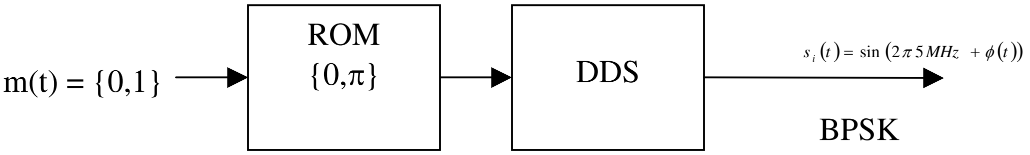


**Figure 5: Simplified block diagram of a BPSK transmitter created through ROM lookup**

The DDS outputs both the sine and cosine waveforms simultaneously. Any change to the addressing affects both waves. Thus it is the DDS method is expandable to M-ary PSK by adding more phases to the ROM while using the same DDS. The redundancy is also an advantage as it is not necessary to build a Hilbert transform, which would be necessary in M-ary PSK without this feature. If the transmitter and receiver are out of phase, the signal will still be received. This is due to the rotation feature of the loop filter. The receiver will attempt to align itself to the transmitter based on the result. It follows that the final waveform is a BPSK signal, aligned on one of the axes, as shown in Figure 6, and an example of the output waveforms is shown in Figure 7.

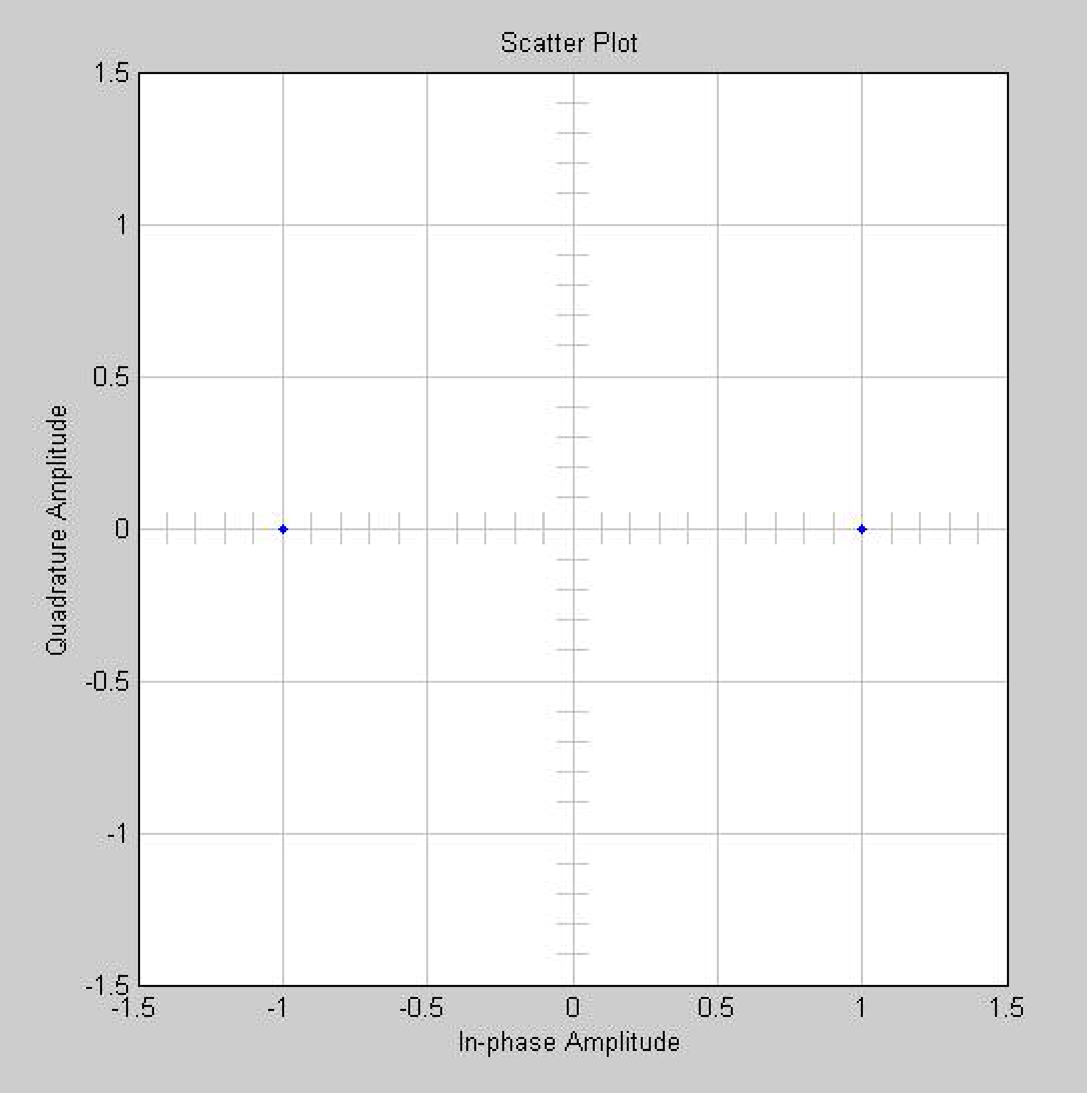


**Figure 6: Ideal constellation of the BPSK transmitter**

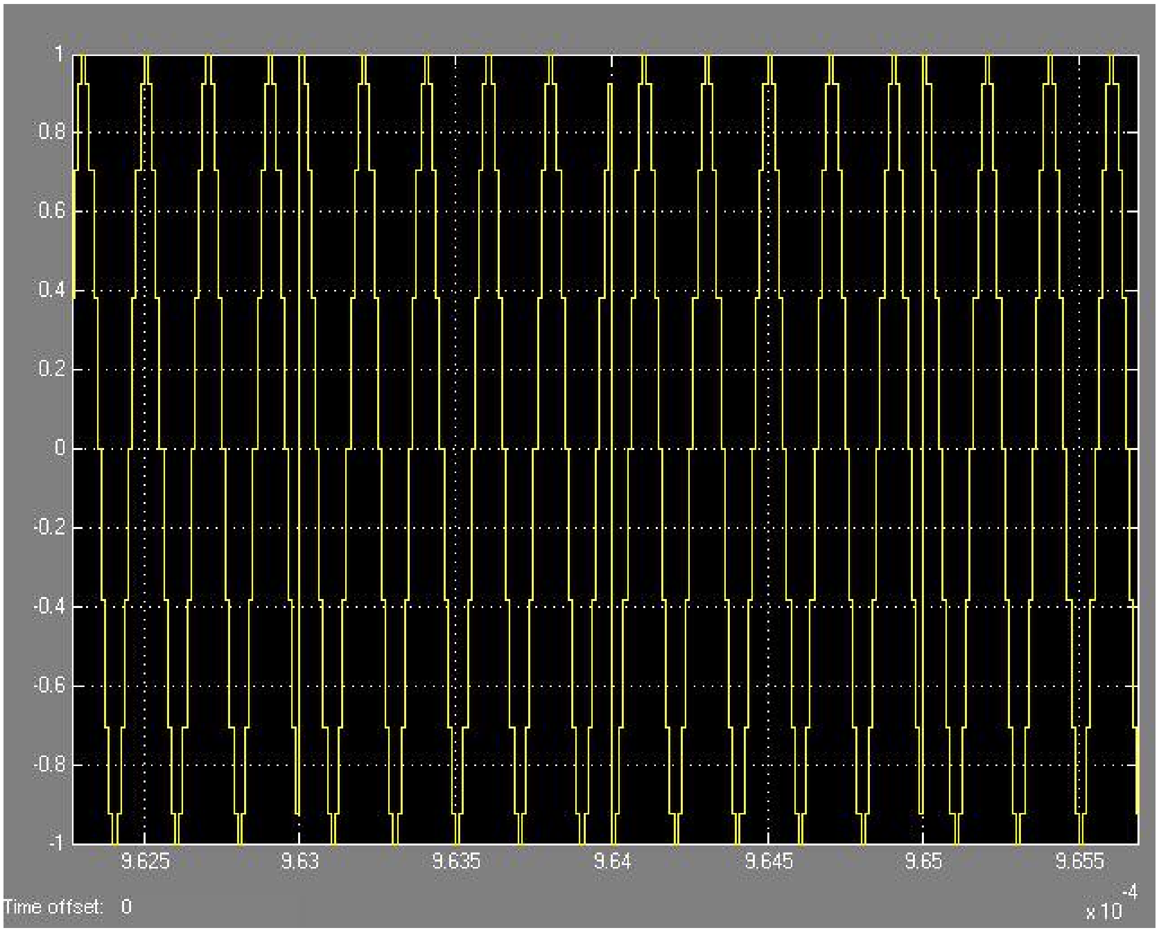


**Figure 7: Output waveforms of the transmitter**

In either implementation, the ideal final output of the transmitter is given in Equation 1, where *m*(*t*) is the message, {-1, 1}, and $f_c$ is the carrier frequency. The Simulink model is shown in Figure 8.

**Equation 1: The ideal output of a quadrature BPSK transmitter**

$$x_i(t) = m(t) \cdot \cos(2\pi f_c t)$$
$$x_q(t) = m(t) \cdot \sin(2\pi f_c t)$$

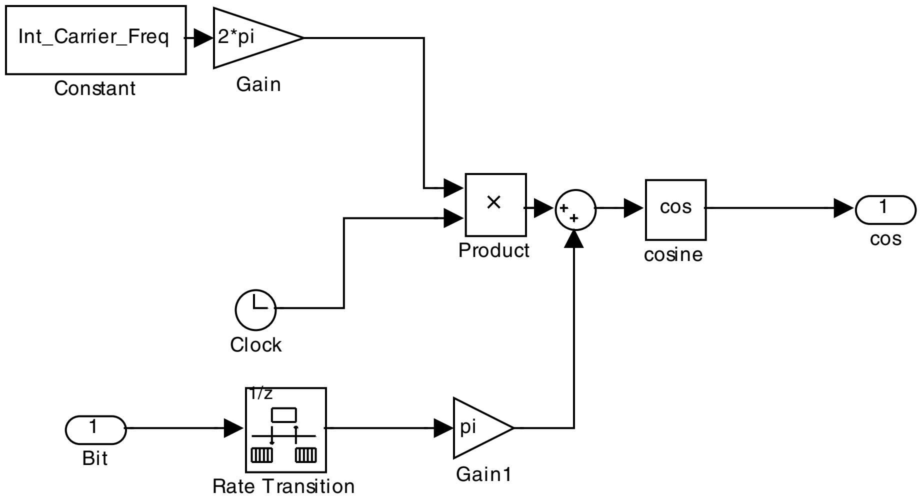


**Figure 8: Simulink model showing the modified schematic of the BPSK transmitter**

## *4.3 Receiver*

Given that the preferred method of receiving phase-shift-keying (PSK) signals is through a Costas loop, the analog design must be ported to the digital domain. [22] This process involves converting mixers to multipliers, analog filters to digital filters, and using DDS modules instead of local oscillators. However, the theory remains the same as his original design in 1956, which is displayed in Figure 9. [1] The first step is to demodulate the signal. Next, the signal is filtered and integrated. Then, phase tracking techniques are used to lock the receiver's carrier and phase to the transmitted signal. Simultaneously, the estimated bit pattern is aligned to the system clock. The modifications are shown in Figure 10 demonstrate the similarities between the original and modified receiver designs.

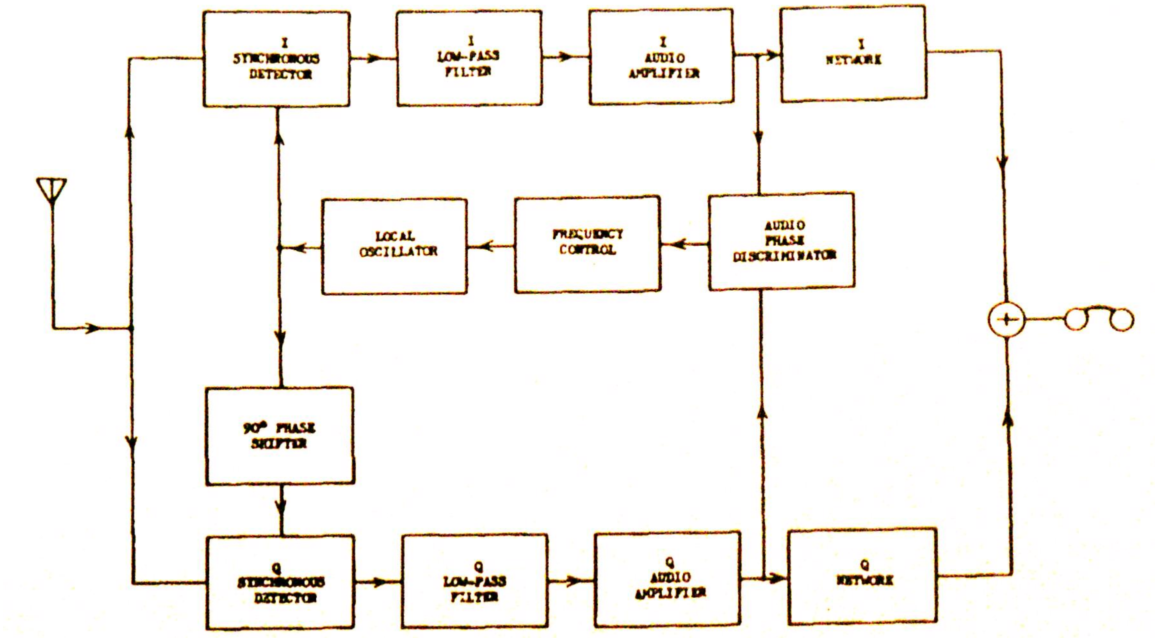


**Figure 9: Original synchronous receiver as proposed by Costas in 1956 [1]**

### *4.3.1 Receiver Overview and Scope of Problems*

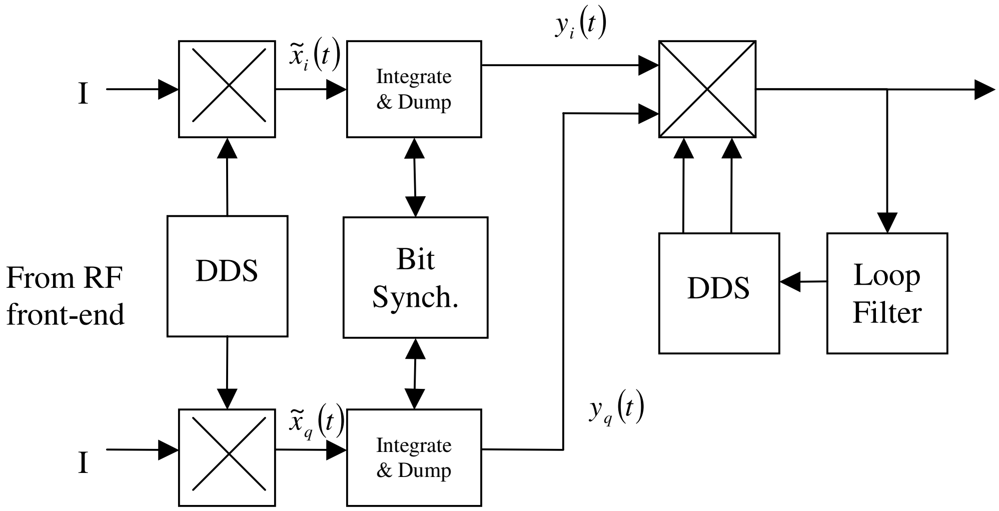


**Figure 10: Modified Costas loop implemented in the KUAR [19]**

In the KUAR BPSK design, the first step modulates the received signal to baseband. Mathematically, the DC component should therefore be the message, as shown in Equation 2.

**Equation 2: The ideal result of demodulation on a BPSK signal**

$$\begin{aligned}\tilde{x}_i(t) &= x_i(t)\cdot\cos(2\pi f_c t) = 0.5m(t) + 0.5m(t)\cos(4\pi f_c t)\\ \tilde{x}_q(t) &= x_i(t)\cdot\sin(2\pi f_c t) = 0.5m(t)\sin(4\pi f_c t)\end{aligned}$$

However, real-life implementation shows that there are multiple scenarios for which the receiver must accommodate according to the given requirements. The Simulink model of the entire receiver is shown in Figure 11.

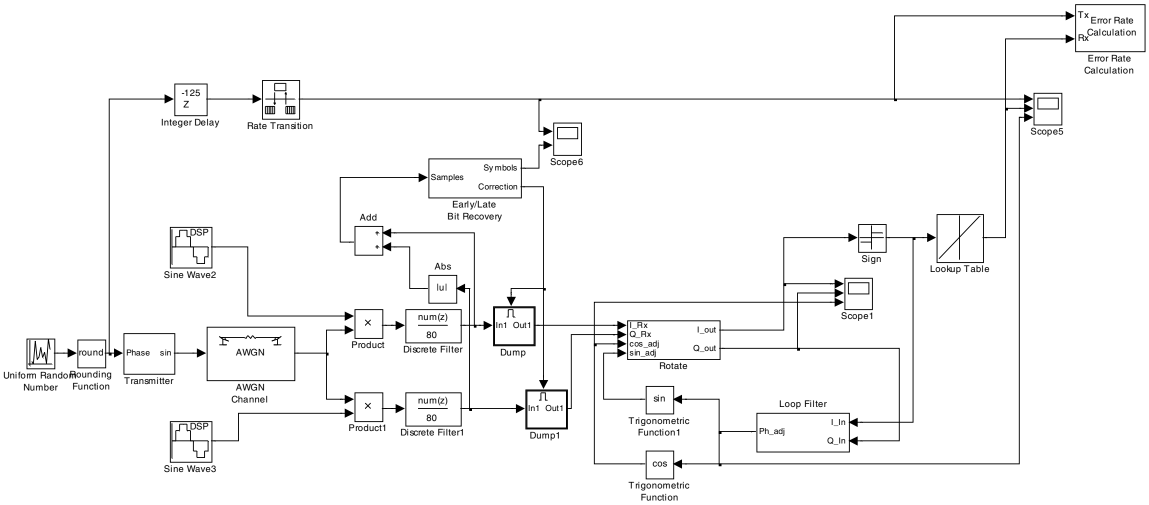


**Figure 11: Simulink model of the receiver**

#### 4.3.1.1 Transmitter – Receiver phase offset

First, if the receiver's and transmitter's clocks are misaligned, this phase shift could affect reception on either arm of the receiver. This scenario is described mathematically in Equation 3.

**Equation 3: The result of demodulation on a BPSK signal with a phase misalignment**

$$\begin{aligned}\tilde{x}_i(t) &= x_i(t+\phi)\cdot\cos(2\pi f_c t) = m(t)\cos(2\pi f_c t)[\cos\phi\cos(2\pi f_c t) - \sin\phi\sin(2\pi f_c t)] \\ &= m(t)[\cos\phi(0.5+0.5\cos(4\pi f_c t)) - \sin\phi(0.5\sin(4\pi f_c t))] \\ &= 0.5m(t)\cos\phi + 0.5m(t)\cos\phi\cos(4\pi f_c t) - 0.5m(t)\sin\phi\sin(4\pi f_c t)\end{aligned}$$

and similarly,

$$\tilde{x}_q(t) = x_i(t+\phi)\cdot\sin(2\pi f_c t) = 0.5m(t)\cos\phi + 0.5m(t)\sin\phi\cos(4\pi f_c t) + 0.5m(t)\cos\phi\sin(4\pi f_c t)$$

Therefore, if $\phi$ is $\pi$, the power at baseband will be reduced.

#### 4.3.1.2 Analog filter not present

Secondly, this design assumes there is an analog filter immediately before the ADC. If the filter is not in place, conversion will cause signals at 75 MHz, 85 MHz, 155 MHz, 165 MHz, etc. to also translate to baseband. Sampling at 80 MSPS causes aliasing at every 80 MHz interval. This analog filter will eliminate signals with frequencies above 30 MHz and thus prevent signals from being translated to 5 MHz and then, consequently to baseband.

#### 4.3.1.3 Doppler shift due to mobility

Third, this system assumes at least one user is mobile. According to the Doppler-Shift Principle, this will result in a frequency offset according to the formula given in Equation 4, where $f_0$ is the transmitter's carrier frequency, $v$ is the wave speed, and $v_{s,r}$ is the transmitter's relative velocity. It can be shown that the frequency will increase if the receiver and transmitter are moving towards each other, and likewise the frequency will decrease if the receiver and transmitter are moving away from each other. This will result in a continuously varying rotation of the constellation, as shown in Figure 12.

**Equation 4: The resulting frequency due to Doppler shift, assuming a moving transmitter relative to a stationary receiver [34]**

$$f = f_0\left(\frac{v}{v + v_{s,r}}\right)$$

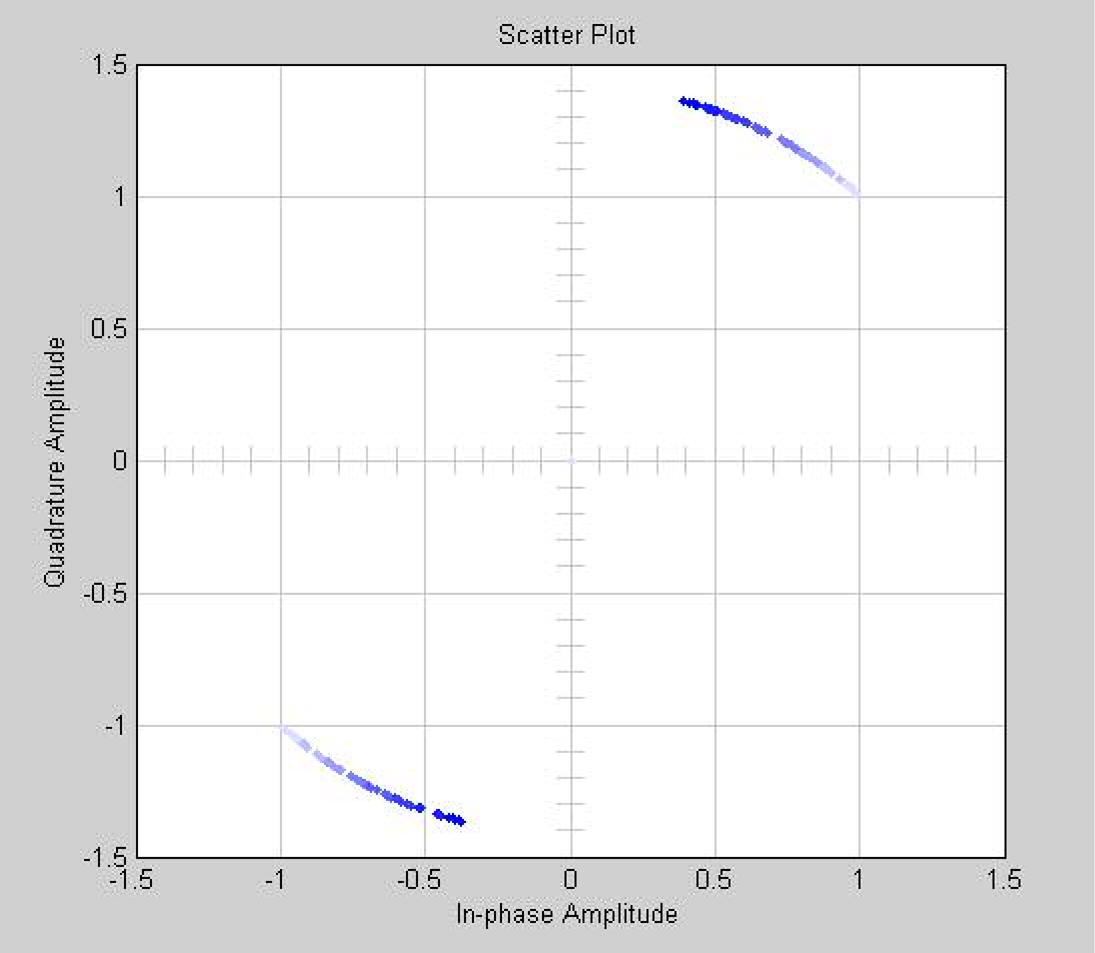


**Figure 12: A constellation demonstrating the effect of Doppler shift**

If both users are moving away from each other at a rate of 300 m/s, roughly Mach 1, according to Equation 4, the corresponding Doppler shift will be 10 Hz. Thus, the filter will synchronize for frequency errors in a range of -10 to 10 Hz.

#### 4.3.1.4 Unsynchronized bits

The receiver needs to detect the transition between bits in order to tell the processor when a new bit has been detected. If the receiver has some time offset, *T*, compared to the transmitter, the final integrator will integrate over a portion of two bit periods, if one bit period is called $\tau$. This will detract from performance since noise, *n*(*t*), also in the system, increases the likelihood of an error in the estimation, $\hat{m}(t)$, of the transmitted message. This is described in Equation 5.

**Equation 5: The bit estimation algorithm**

$$\hat{m}(t) = \sum_{0}^{\tau} m(t+T) + n(t)$$

Thus, an algorithm is necessary to reduce *T* to 0 and the integration will reduce the affects of the noise.

All of these problems will be handled later by blocks in the detector chain. The problems for the analog filter need to be handled externally to the FPGA.

After demodulation to baseband, both the in-phase and quadrature signals pass through a matched filter, as seen in Figure 10, labeled “integrate & dump.” This matched filter suppresses signals outside of the message bandwidth. This stage is easier since the message baud rate is known, as $\tau$ can be “hardwired” into the algorithm. The ideal result is a square-wave version of the message signal, as calculated in Equation 6, and shown in Figure 13.

**Equation 6: The result of integrating the demodulated signal**

$$y_q(t) = \sum_{t}^{t+\tau} 0.5m(t+T)\cos\phi + 0.5m(t)\sin\phi\sin(4\pi f_c t) - 0.5m(t+T)\cos\phi\cos(4\pi f_c t) + n(t)$$

$$= \sum_{t}^{t+\tau} 0.5m(t+T)\cos\phi$$

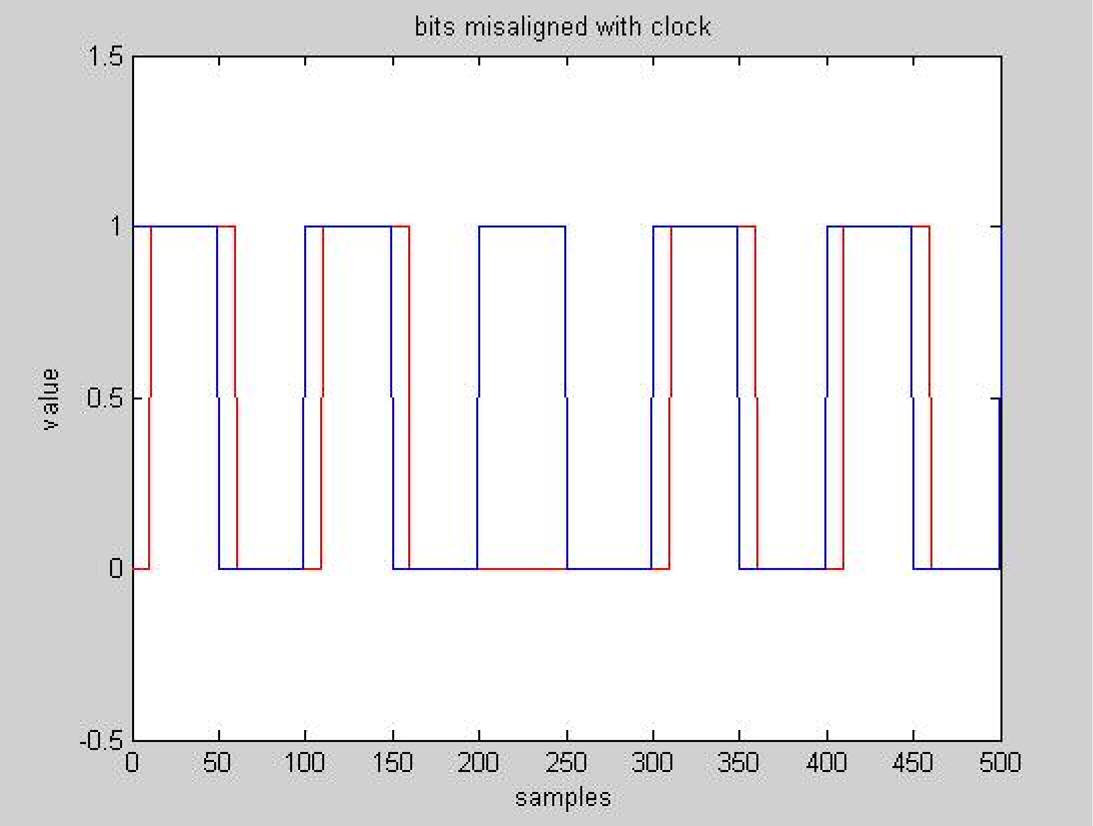


**Figure 13: A clock not synchronized with bit samples**

The next block in the detector attempts to eliminate frequency and phase errors from the signal, and is labeled loop filter in Figure 10. This tracking is performed in a second-order infinite-impulse response (IIR) filter. The result is a complex signal, which is translated into trigonometric form via another DDS. The estimated message is the sign of the result of the in-phase result of this rotation.

Finally, the bit-time recovery algorithm uses data from the integrate and dump block to synchronize the symbol. The data is sampled internally to the early-late algorithm and triggers the integrator to dump. The result is an integration of the transmitted symbol and a clock pulse indicating to the rest of the receiver that a new symbol has been estimated.

### 4.3.2 Details of the Matched Filter

As was previously shown in Equations 6, at baseband, the ideal signal is a square wave envelope of the bit. After sampling the 1 MBaud now square-wave at 80 MSPS, it becomes a sequence of eighty ones or negative ones and represented in Equation 7. Thus, the goal of the matched filter is to attempt to create a symbol identical to the transmitted one and compare to the symbol. The matched filter is an integration of the previous eighty samples. This integration is performed with a boxcar finite-impulse response (FIR) filter. Mathematically, a boxcar FIR filter follows the format of Equation 8, and has a frequency response shown in Figure 14.

**Equation 7: The ideal sampled, demodulated received signal**

$$S(z) = \pm\left(1 + z^{-1} + z^{-2} + z^{-3} + \ldots + z^{-79}\right)$$

**Equation 8: Impulse response of the matched boxcar filter**

$$H(z) = 1 + z^{-1} + z^{-2} + z^{-3} + \ldots + z^{-79}$$

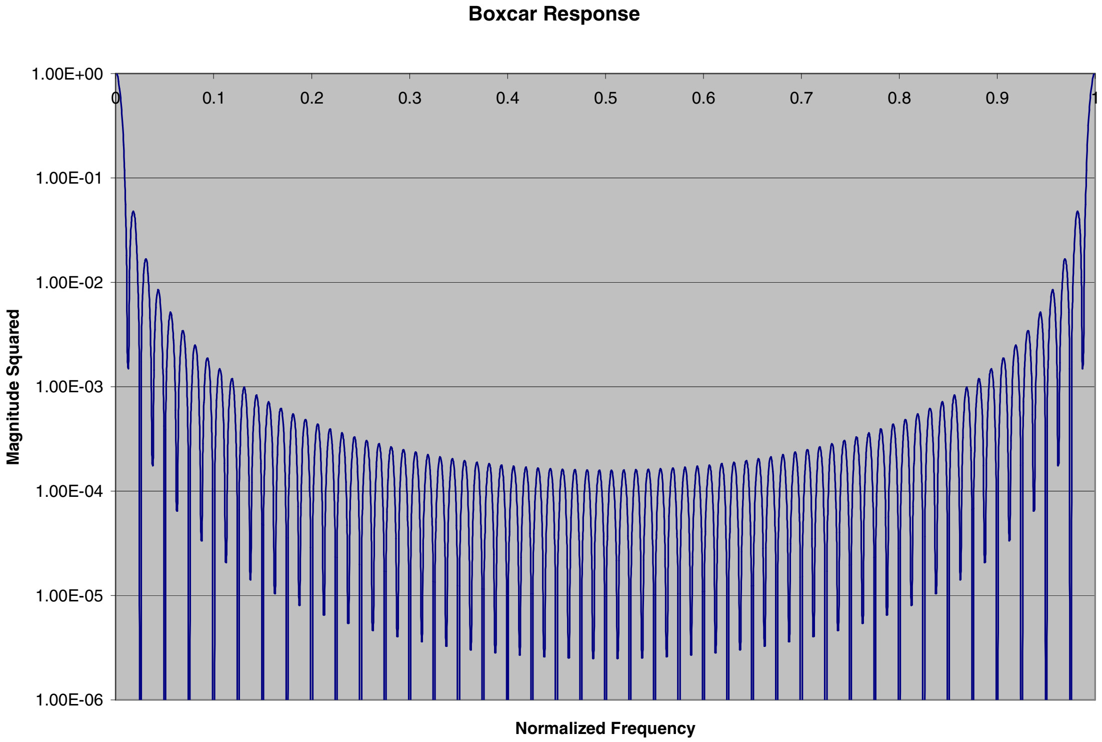


**Figure 14: Frequency response of an 80-tap boxcar filter**

According to Shanmugan, based on the performance, ease of construction, and on-the-fly adaptability, the boxcar FIR filter is the best choice for a matched filter. In fact, he suggests that any filter placed in this position must match the symbol in order to yield the maximum probability of estimating the bit correctly. [35] Since the boxcar filter follows this rule, it is the best choice for a matched filter. The ease of construction will be discussed in the following section; it will be shown that this filter can be adapted to handle several different lengths easily and can be modified quickly to handle a different message rate.

This filter is triggered to dump its result after 79, 80, or 81 samples. This is determined by the result of the early-late gate algorithm. It works in parallel to the matched filter and corrects for bit-time errors. This will be discussed in greater detail later.

## *4.3.3 Frequency and phase tracking*

### 4.3.3.1 Complex multiplication

The result of the boxcar FIR filter passes into a complex multiplier. This multiplication follows the format shown in Equation 9.

**Equation 9: Complex multiplication**

$$I_{out} + jQ_{out} = (I_{Rx} + jQ_{Rx})(I_{Loop} + jQ_{Loop}) = (I_{Rx}I_{Loop} - Q_{Rx}Q_{Loop}) + j(I_{Rx}Q_{Loop} + Q_{Rx}I_{Loop})$$

The result of this multiplication rotates the input so that the phase and frequency is steady with the constellation. This operation is necessary as both the loop and the received lines are complex.

The in-phase result of this multiplication should resemble the transmitted bit stream. The quad-phase result should be driven to zero by the loop filter.

### 4.3.3.2 Loop filter

This message signal passes into the loop filter, which determines how to rotate the signal around the constellation. The loop filter is a second-order IIR filter, designed in the traditional method. The traditional method provides an adequate response while using few resources. Although a FIR is easier to design in the DSP world, it does not have the same latency as that of an IIR filter with a comparable frequency response. For example, a comparable FIR filter would require a substantially larger number of taps to achieve the same degree of performance relative to this IIR filter. This excess of latency drastically affects the ability of the loop. For instance, if one user moves suddenly and stops, this creates a brief frequency deviation. If a FIR filter is used, this latency will "update" the rotation significantly later than is necessary. Either this will allow an error to pass through the system or it will modify a correct estimation to an incorrect value.

With only two orders, the IIR filter can update the loop significantly faster than the FIR filter.

The traditional method is a second-order system because there are no third order errors. A phase error is compensated with a first-order system; a frequency error is compensated with a second-order system. This is described mathematically in Equations 10 and 11.

**Equation 10: A phase error is corrected with a step response phase correction, which is a first-order system [19]**

$$A\cos(\omega_0 t + \theta(t)) = A\cos(\omega_0 t + \Delta\theta u(t))$$
$$\theta(t) = \Delta\theta u(t)$$
$$\Theta(s) = \frac{\Delta\theta}{s}$$

**Equation 11: A frequency error is corrected with a ramp response frequency correction, which is a second-order system [19]**

$$A\cos((\omega_0 + \Delta\omega)t) = A\cos(\omega_0 t + (\Delta\omega)t(t))$$
$$\theta(t) = (\Delta\omega)tu(t)$$
$$\Theta(s) = \frac{\Delta\omega}{s^2}$$

A PLL loop filter, designed in the traditional method, is described below, in Figure 15.

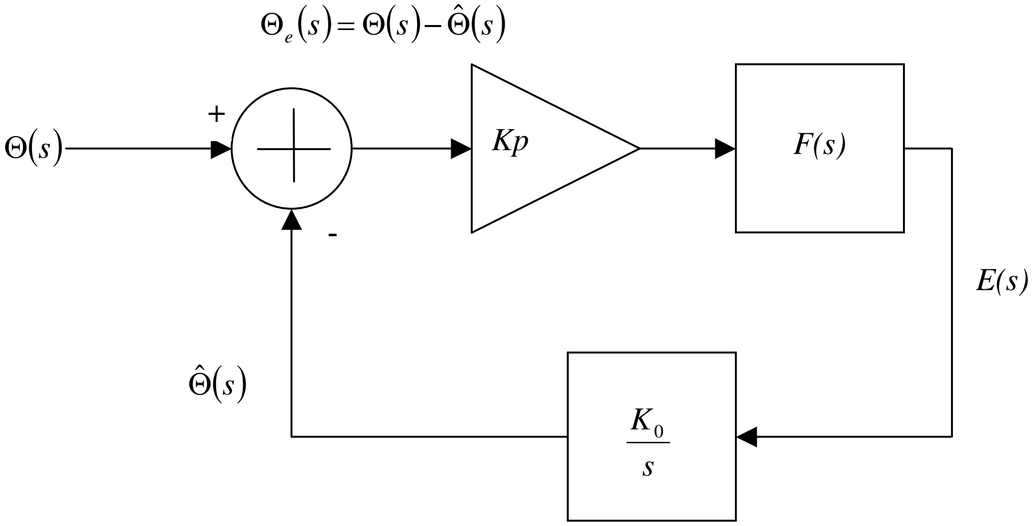


**Figure 15: Block diagram of a traditional PLL loop filter [19]**

It can be shown that the phase error, *Θ<sub>e</sub>(s)*, for the step and ramp responses is given by Equation 12. Furthermore, the boundary conditions of F(0) are also given in Equation 12.

**Equation 12: The phase error responses for a step and ramp, used to determine the boundary conditions [19]**

$$\Theta_e(s)=\frac{\Delta\theta}{s+K_0K_pF(s)}=\lim_{s\to 0}\left\{\frac{s\Delta\theta}{s+K_0K_pF(s)}\right\}$$
$$=0\,\text{if }F(0)\neq 0$$

$$\Theta_e(s)=\frac{\Delta\omega}{s^2+K_0K_psF(s)}=\lim_{s\to 0}\left\{\frac{\Delta\omega}{s+K_0K_pF(s)}\right\}$$
$$=0\,\text{if }F(0)=\infty$$

A Proportional and Integrator filter, as described in Equation 13, fulfills the boundary conditions, as $F(0)=\infty$.

**Equation 13: The response of a Proportional and Integrator filter [19]**

$$F(s) = K_1 + \frac{K_2}{s}$$

Inserting the filter in the system gives the transfer function as follows in Equation 14.

**Equation 14: Closed-loop transfer function of the PLL loop filter [19]**

$$H(s) = \frac{\hat{\Theta}(s)}{\Theta(s)} = \frac{K_0 K_p \frac{F(s)}{s}}{1 + K_0 K_p \frac{F(s)}{s}} = \frac{K_0 K_p K_1 s + K_0 K_p K_2}{s^2 + K_0 K_p K_1 s + K_0 K_p K_2} = \frac{2\zeta\omega_n s + \omega_n^2}{s^2 + 2\zeta\omega_n s + \omega_n^2}$$

$$\omega_n = \sqrt{K_0 K_p K_2}$$

$$\zeta = \frac{K_1}{2}\sqrt{\frac{K_0 K_p}{K_2}}$$

Given the closed-loop transfer function, it can be shown that the equivalent noise bandwidth of the loop is described by Equation 15.

**Equation 15: Equivalent noise bandwidth of the closed-loop transfer function [19]**

$$B_N = \frac{1}{|H(0)|^2}\int_0^\infty |H(j\omega)|^2 d\omega = \frac{\omega_n}{2}\left(\zeta + \frac{1}{4\zeta}\right)$$

The equivalent noise bandwidth of the filter is used to describe characteristics of the loop, such as the pull-in range and acquisition time, which are described in Equations 16 and 17.

**Equation 16: Pull-in range of the loop filter [19]**

$$\Delta f \approx 6B_N$$

**Equation 17: Acquisition time of the loop filter [19]**

$$T_{LOCK} \approx 4\frac{(\Delta f)^2}{B_N^3} + \frac{1.3}{B_N}$$

Moving to the digital domain allows for the solution to the loop filter, given design parameters. The solution equations are shown in Equation 18.

**Equation 18: Solution equations for the digital loop filter [19]**

$$K_0 K_p K_1 = \frac{4\zeta}{\zeta + \frac{1}{4\zeta}}(B_N T)$$
$$K_0 K_p K_2 = \frac{4}{\left(\zeta + \frac{1}{4\zeta}\right)^2}(B_N T)^2$$

The solution equations determine the filter coefficients given the design parameters $B_N$, $T_s$, and $\zeta$. The final design parameter is given in Equation 19, where $N$ is the number of samples per symbol.

**Equation 19: The final design parameter, the critical phase of the loop filter [19]**

$$\theta_n = \frac{B_N T}{N\left(\zeta + \frac{1}{4\zeta}\right)}$$

Following the design procedure, the loop filter will be designed such that

$$K_0 K_p K_1 = 0.025$$
$$K_0 K_p K_2 = 0.0003125$$

Finally, the closed-form digital loop filter is solved in Equation 20.

**Equation 20: The solution of the closed-loop digital filter [19]**

$$H(z)=\frac{\hat{\Theta}(z)}{\Theta(z)}=\frac{K_0K_p(K_1+K_2)z^{-1}-K_0K_pK_1z^{-2}}{1-2\left(1-\frac{1}{2}K_0K_p(K_1+K_2)\right)z^{-1}+(1-K_0K_pK_1)z^{-2}}$$

$$=\frac{0.0253125z^{-1}-0.025z^{-2}}{1-2z^{-1}+z^{-2}}$$

This output becomes the phase input to a DDS module. In effect, this translates the exponential complex number to a trigonometric complex number. The entire loop is shown in Figure 16.

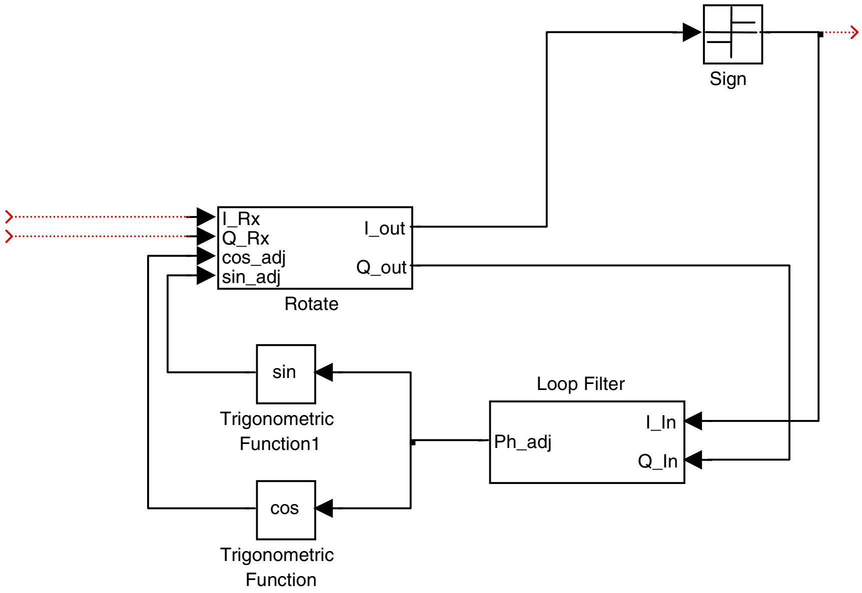


**Figure 16: Simulink model of the loop filter**

The first test of the loop filter will be a near optimal case where the transmitter is in phase with the receiver and there is a 20 dB SNR AWGN channel. The test schematic is shown in Figure 17. Each of the following figures shows three things: the transmitted message, the estimated message, and the output of the loop filter, respectively. Due to the nature of BPSK, a unique symbol set will need to be transmitted in order to differentiate between the phases. Thus, the second line in the results may be inverted with respect to the transmitted signal.

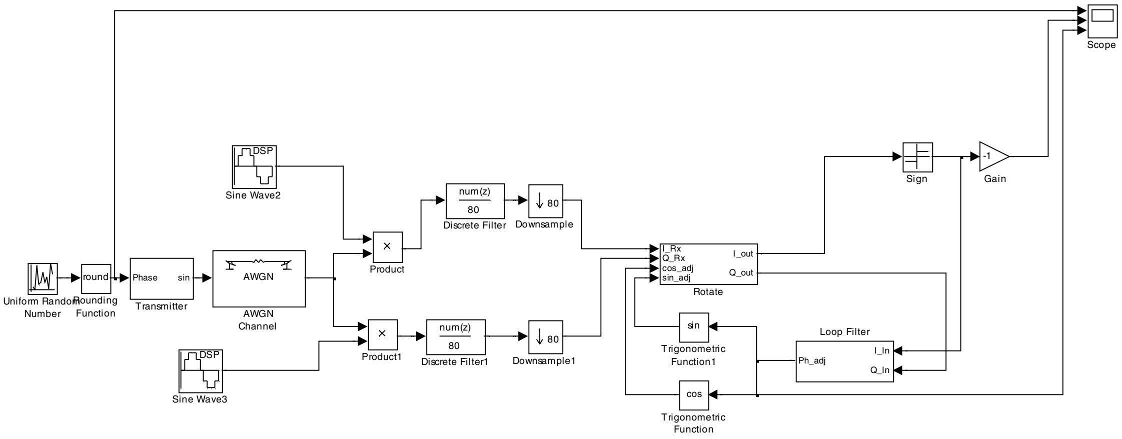


**Figure 17: The test schematic for carrier synchronization**

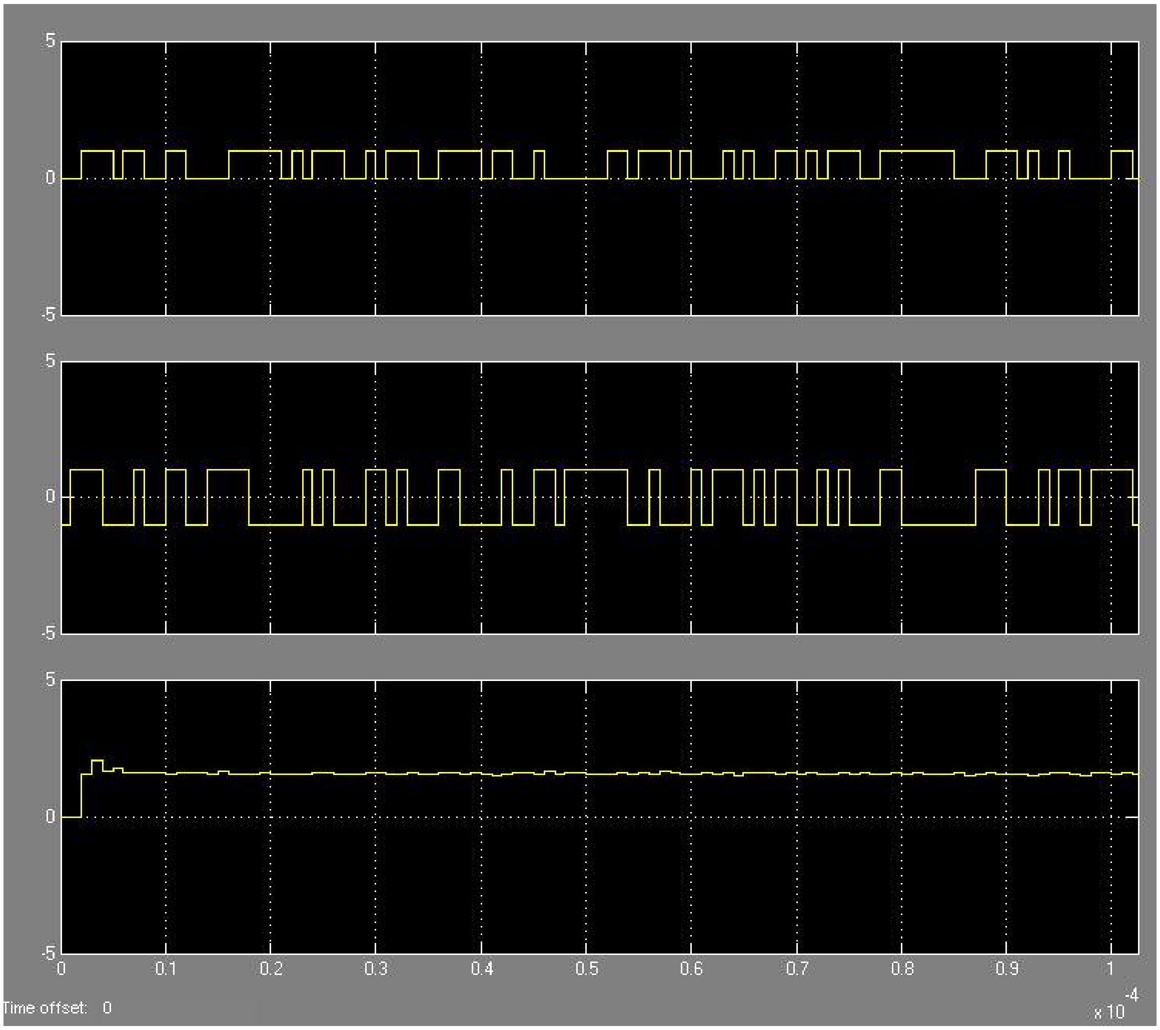


**Figure 18: The ideal case for the loop filter**

Figure 18 shows that the filter has compensated for the noise and shows that the filter comes to steady state in about 10 μs. The next case will show how the filter reacts to a constant phase misalignment in Figure 19.

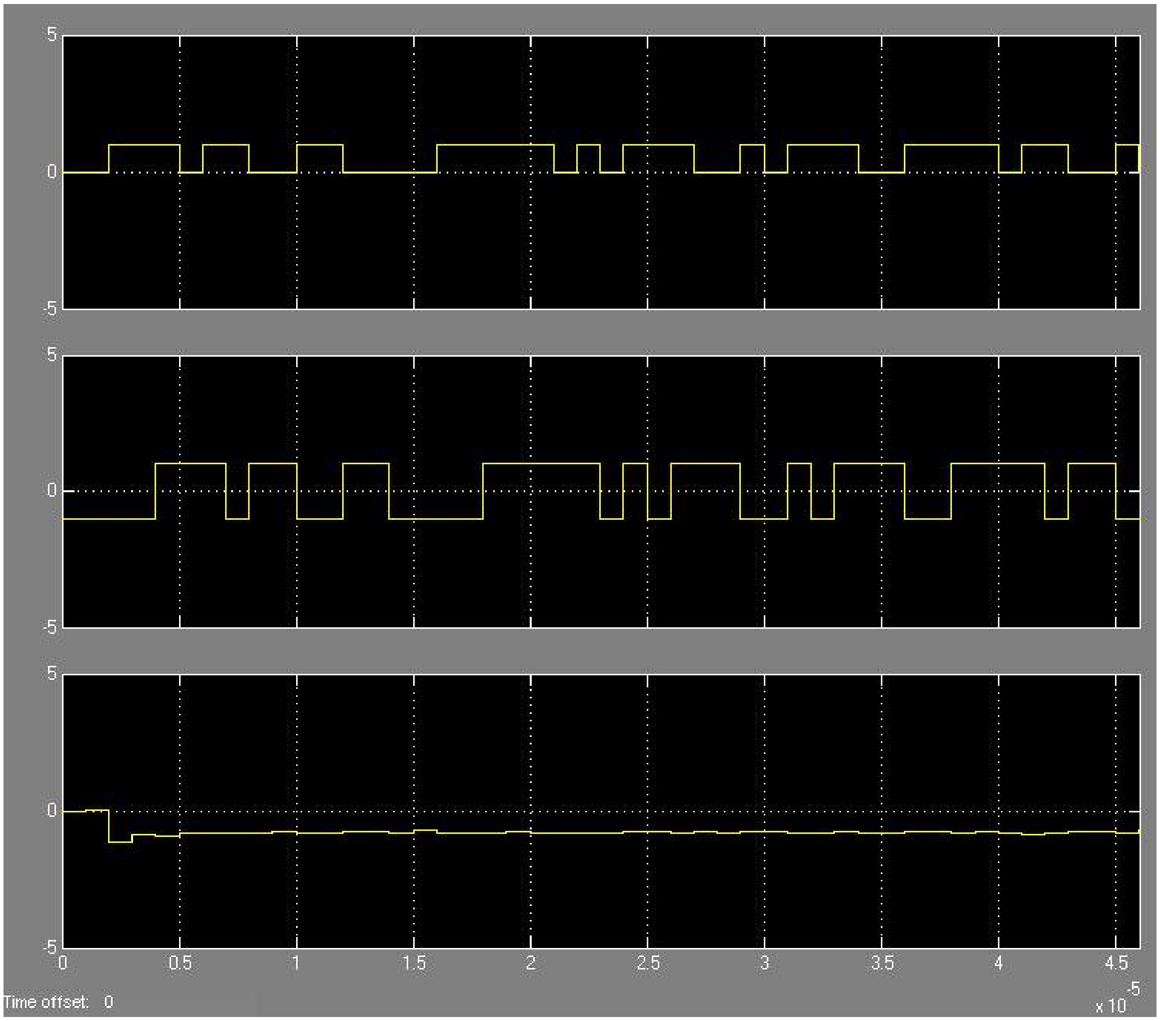


**Figure 19: The loop filter's response to a constant phase error**

Once again, the filter compensates for the error within 10 μs. The next case shows how the filter will react to a 20 Hz deviation. Figure 20 shows the expected, that the output of the filter is a sloped line, while Figure 21 shows that the bits are correct and that the error is compensated within 60 μs.

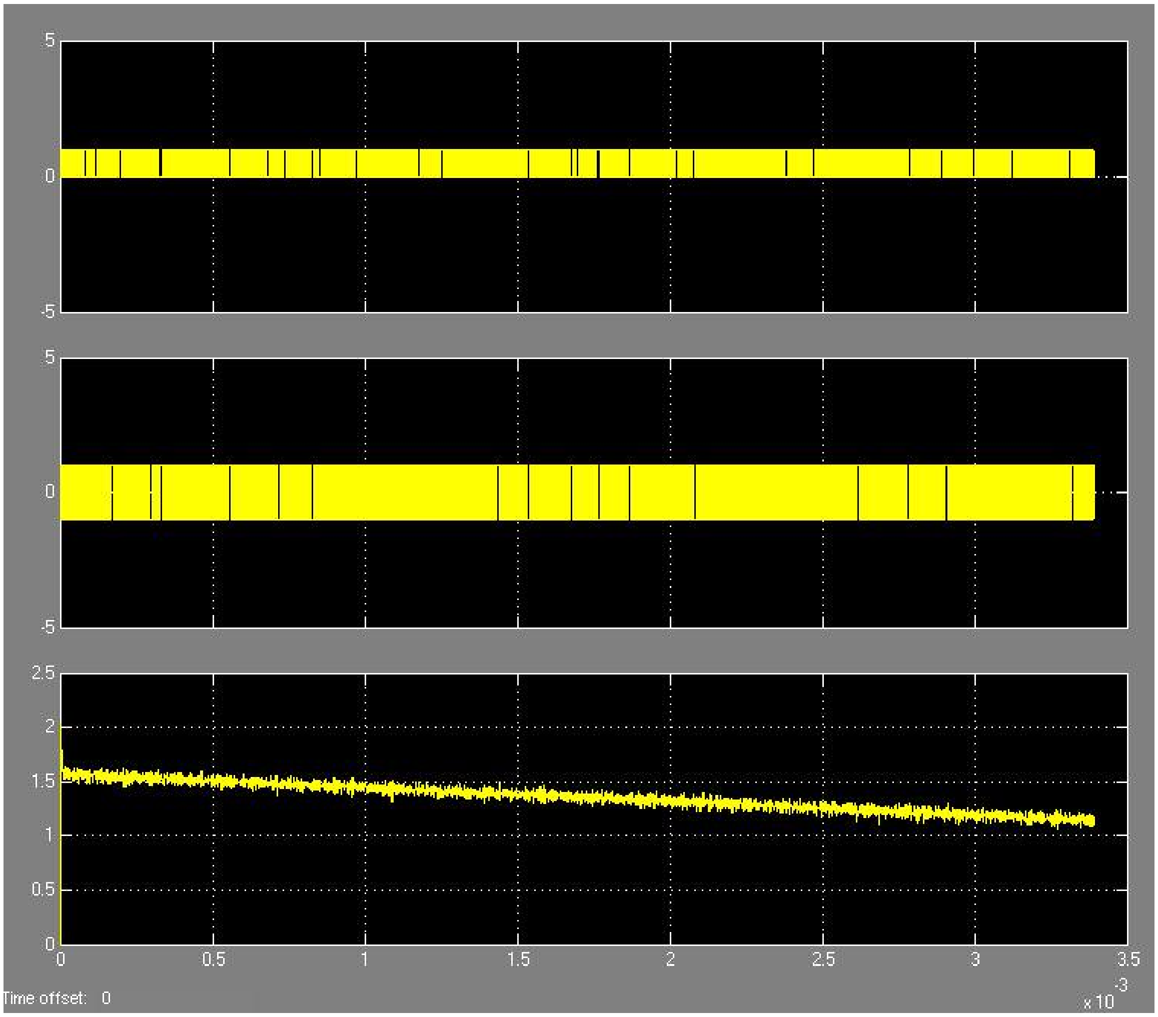


**Figure 20: The loop filter's response to a Doppler Shift**

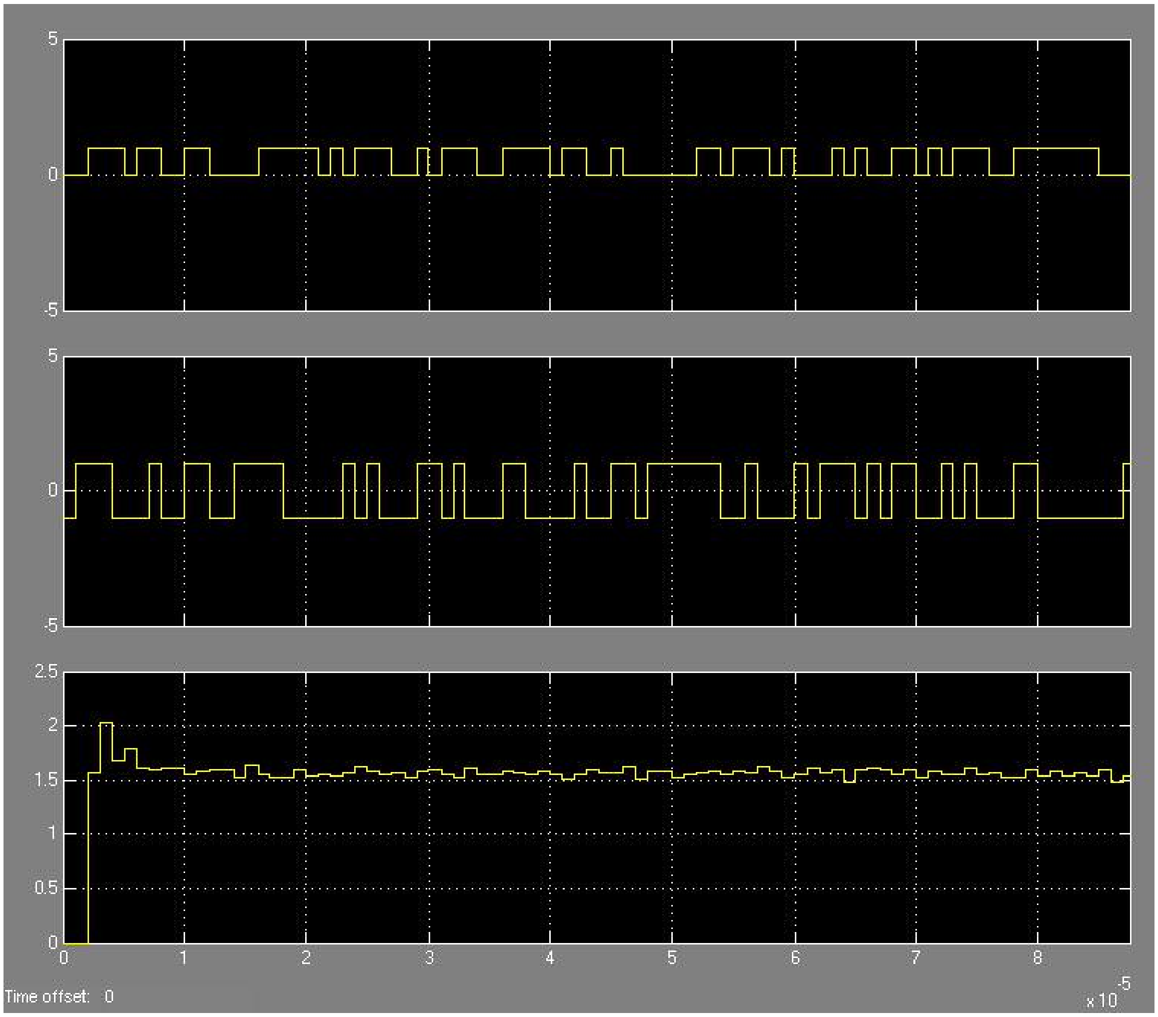


**Figure 21: Verification that the loop filter compensates for the designed shift**

This shows that the filter is capable of synchronizing to the designed frequency deviations.

#### 4.3.3.3 Bit-time recovery

The bit-time recovery block estimates the location of the symbol clock with respect to the estimated symbol. It also does a final estimation of the transmitted symbol. The simplicity of the early-late gate algorithm made it a very good choice compared to other algorithms. [25] While the other algorithms discussed in the background section claim to have faster response time or estimation accuracy, these algorithms use more resources. The tradeoff of resources to performance led to the determination that the early-late gate algorithm is the best choice.

This algorithm uses another eighty-tap boxcar FIR filter to average over an arbitrary bit interval. The result then splits into three branches: one has no delay, one has one cycle of delay, and the final has two cycles of delay. In the middle of these three branches is a latch, which gates based on the result of the next section. The final stage of the three branches is the math, which determines both when to gate the latch and what the bit estimate is. The slope of the input is determined by subtracting the top branch from the lower branch. If the slope is zero, the clock is locked on the bit interval and the gate will latch every eighty cycles. The center branch is multiplied by the slope to determine if the slope is positive or negative. Thereby, the algorithm determines if the gate needs to latch every seventy-nine or eighty-one cycles. The result appears as if the integration window shifts towards the correct bit interval.

The center branch also serves as the bit estimate. Assuming the clock is locked, the integration over the previous eighty samples provides the most accurate estimate of the message. The downside of this algorithm is the lock time. If the transmitter and receiver are one half bit-cycle off and the transmitted message is alternating on every bit, it will take forty cycles to attain lock. If the message does not alternate on every bit, it will take longer to lock. This does not mean that the bit estimate will be incorrect until lock, but it will have little confidence. Although not coded, this is a possible output if desired in the future. It would be preferable to have the algorithm output a locked line, indicating if the algorithm is gating every eighty cycles.

This algorithm may be selected for implementation in software if space becomes an issue. The advantage of performing this operation in logic is the parallel computation versus the single thread of a microprocessor. However, since the algorithm runs primarily at the symbol time, the fast speed of the microprocessor should be able to handle this algorithm without diverting much from its other tasks. The Simulink model of the Early-Late Gate algorithm is shown in Figure 22, and an example of an output is shown in Figure 23.

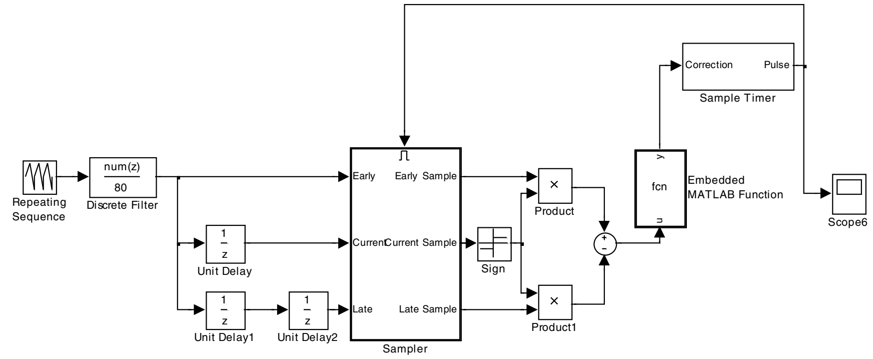


**Figure 22: Simulink model of the Early-Late Gate bit recovery algorithm**

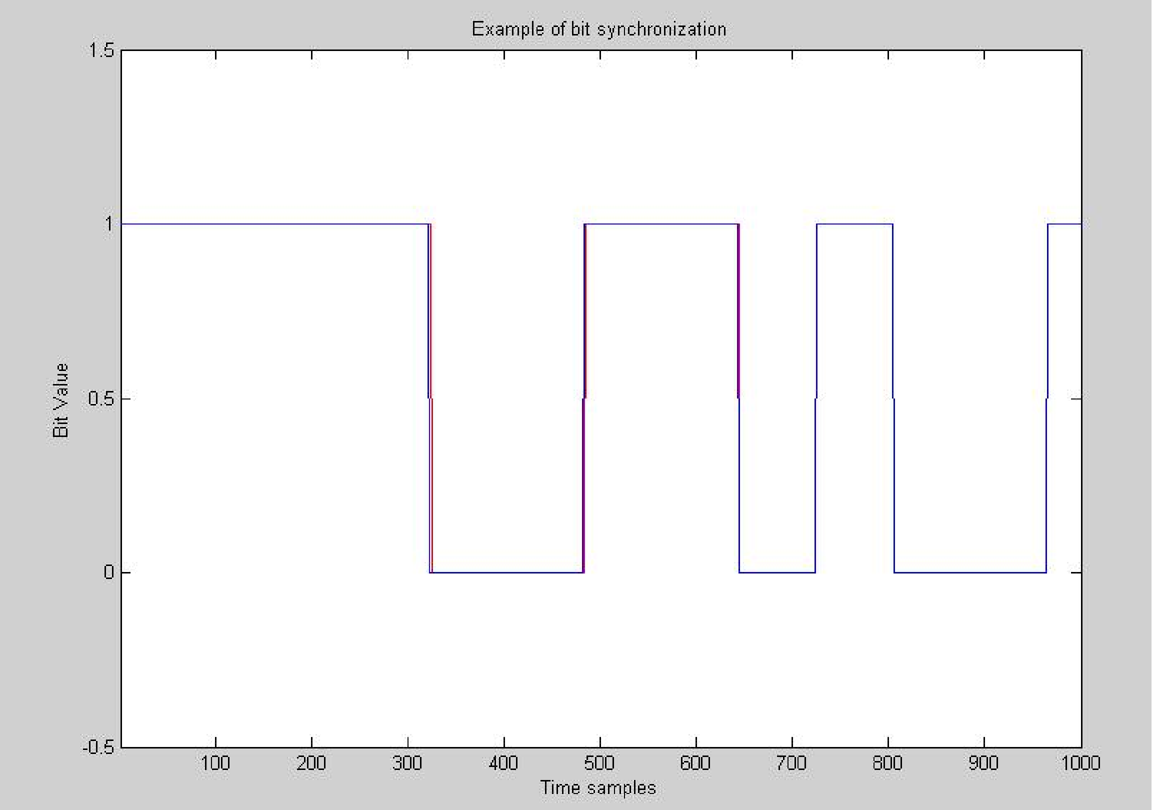


**Figure 23: An Early-Late Gate Algorithm locking to a signal**

The results, seen in Figure 24, demonstrate that this receiver is capable of estimating the message. Although other algorithms may realize a better SNR vs. bit-error rate (BER) curve, that the novelty of this receiver is the implementation in an FPGA for a mobile SDR.

With the combination of error-control and error-correction algorithms, the BER performance should improve. These algorithms are proposed to run concurrently in the FPGA's embedded processor.

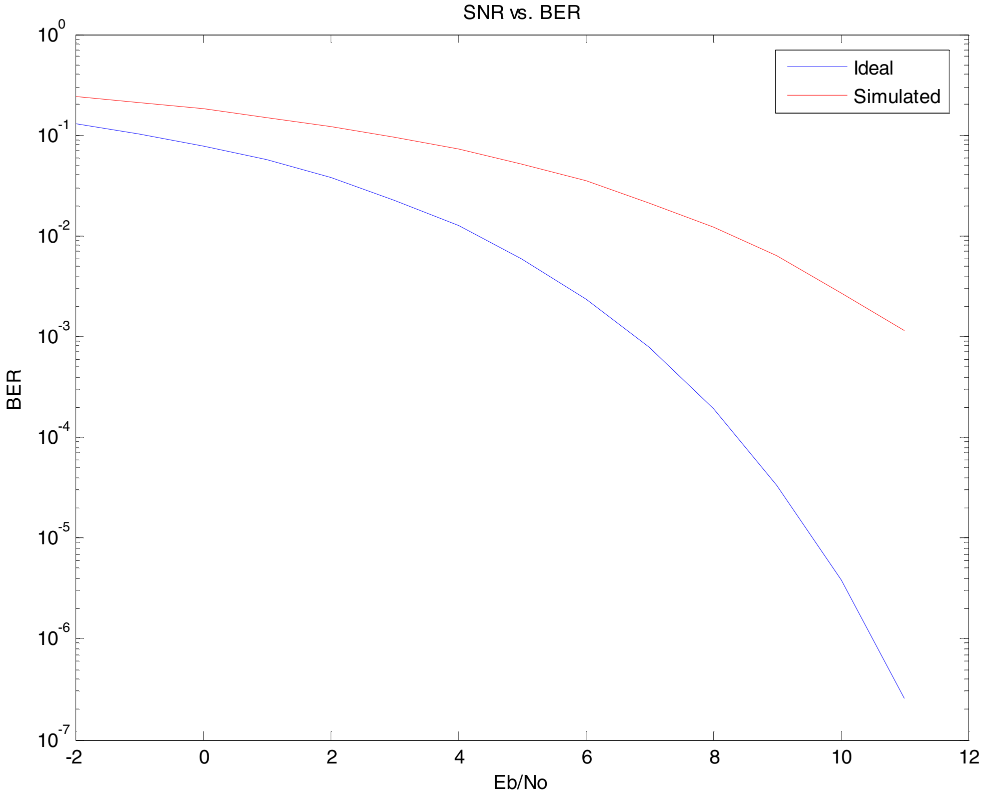


**Figure 24: Simulated SNR vs. BER for the BPSK Transceiver after carrier and bit synchronization.**

# CHAPTER 5 HARDWARE IMPLEMENTATION

## 5.1 Introduction

The previous simulation shows that the transceiver is capable of communicating a message between the transmitter and receiver mathematically, and in simulation. However, the transceiver needs to be built into firmware. The design is ported into Xilinx ISE, a program that Xilinx sells for use with their FPGA processors. It provides primarily three methods of creating designs to a user: programming in VHDL, programming in Verilog, and programming through schematic. If the user chooses to type code in the VHDL or Verilog languages, ISE will interpret this code and virtually wire the processor to perform the algorithms. However, if the user chooses to draw the algorithms in schematic format, the user will also have the option of coding in VHDL or Verilog and produce modules that the schematic will interpret. The final schematic, similarly, can be loaded in the FPGA processor. The design for KUAR was created in schematic, and a few modules were designed in VHDL.

Since the simulation was created with as many simple blocks as possible, the port to Xilinx ISE is nearly one to one. Xilinx provides the ability to create most digital and math structures through their Intellectual Property Cores (IP Cores). These cores provide the best optimization of selectable speed or resources, using the look-up table (LUT)-based hardware in the Virtex-II Pro FPGA. Should a designer decide to not use an IP Core, it would still be possible to create most algorithms from digital basics (e.g. Flip Flops, logic gates, hardware multipliers, or block RAM) through a graphical schematic approach, a programming VHDL approach, or a combination of the two.

Under these constraints, the Simulink design is ported to Xilinx ISE. In the transition, mixers become multipliers, filters become delays, multipliers, and accumulators, and delays become shift registers. Only two modules did not port easily in this transition: the boxcar FIR filters and the IIR loop filter. The IP Core did not efficiently build the boxcar FIR and there is no IP Core for an IIR filter.

Local oscillation is provided through a Direct Digital Synthesizer IP Core. This module constructs sinusoids by use of LUT hardware. The DDS module's size is determined by the frequency accuracy and desired SFDR. The frequency is determined through the pace at which the index increments through the LUT. Since these sinusoids are created through LUT hardware, they also create a great deal of harmonic resonance. This noise is counter-acted in one of two ways: Taylor Series Correction or Phase Dithering. Taylor Series Correction is accomplished by using otherwise discarded bits in an attempt to increase spurious-free dynamic range (SFDR). The result pushes the noise floor very low, but leaves spectral harmonics throughout the working frequency range. The Phase Dithering actually adds noise to the least significant bits in the phase slope. The randomness thereby nearly eliminates harmonic components, but increases the noise floor slightly. [31] Phase dithering was used in all local oscillators in this project as harmonic interference is considered a bigger problem than noise floor. An example of the GUI used to construct a DDS is shown in Figure 27.

Another component used in the simulation that does not port one-to-one is the Boxcar Filter. Instead of using either of the Xilinx-provided FIR creation modules, this filter was implemented more abstractly. Since the algorithm performs a non-weighted moving average, only two modules are necessary. The two modules are an 80-tap shift register and a block to perform the mathematical operations. This is discussed in greater detail later in Section 5.2.

The final consideration is the switch from floating point in simulation to fixed point in implementation. The data input to the FPGA comes from two analog-digital converters running at a sampling frequency of 80 MSPS with a width of 14 bits. Thus, all modules use at least 16 bits in an attempt to negate this problem. Sign extension and truncation are used wherever necessary.

In most modules, this never becomes a problem. For example, the boxcar filter sums over eighty samples, which causes high bit growth. However, this bit growth is fixed and can never grow more than this known quantity of bits. The only block in which this poses a problem is the IIR filter, where a result is accumulated. The accumulation register is expanded to 32 bits wide to compensate for the bit growth. Furthermore, the filter is driving numbers to zero; thus, the issue should not become a problem. The top level of the receiver schematic is shown below in Figure 25, and the transmitter is in Figure 26.

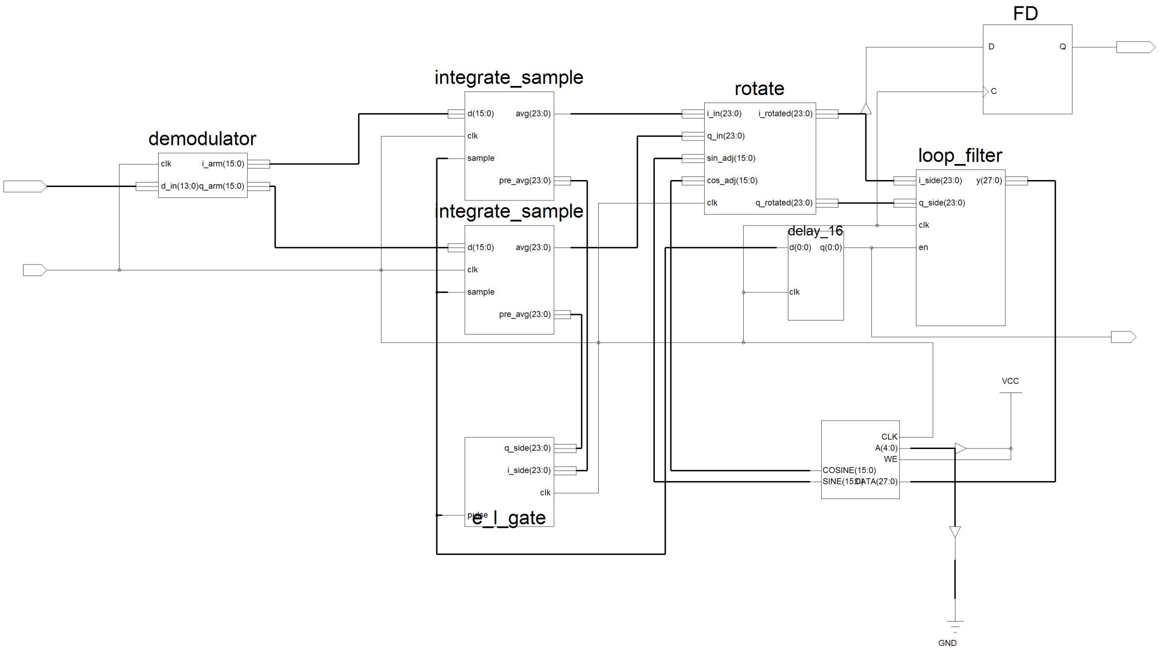


**Figure 25: Schematic of the receiver, created in Xilinx ISE**

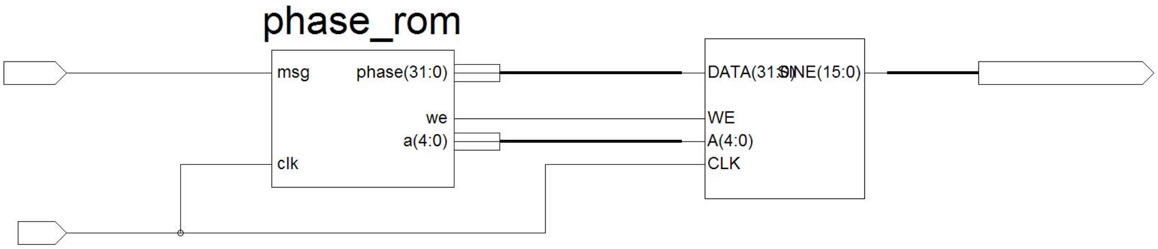
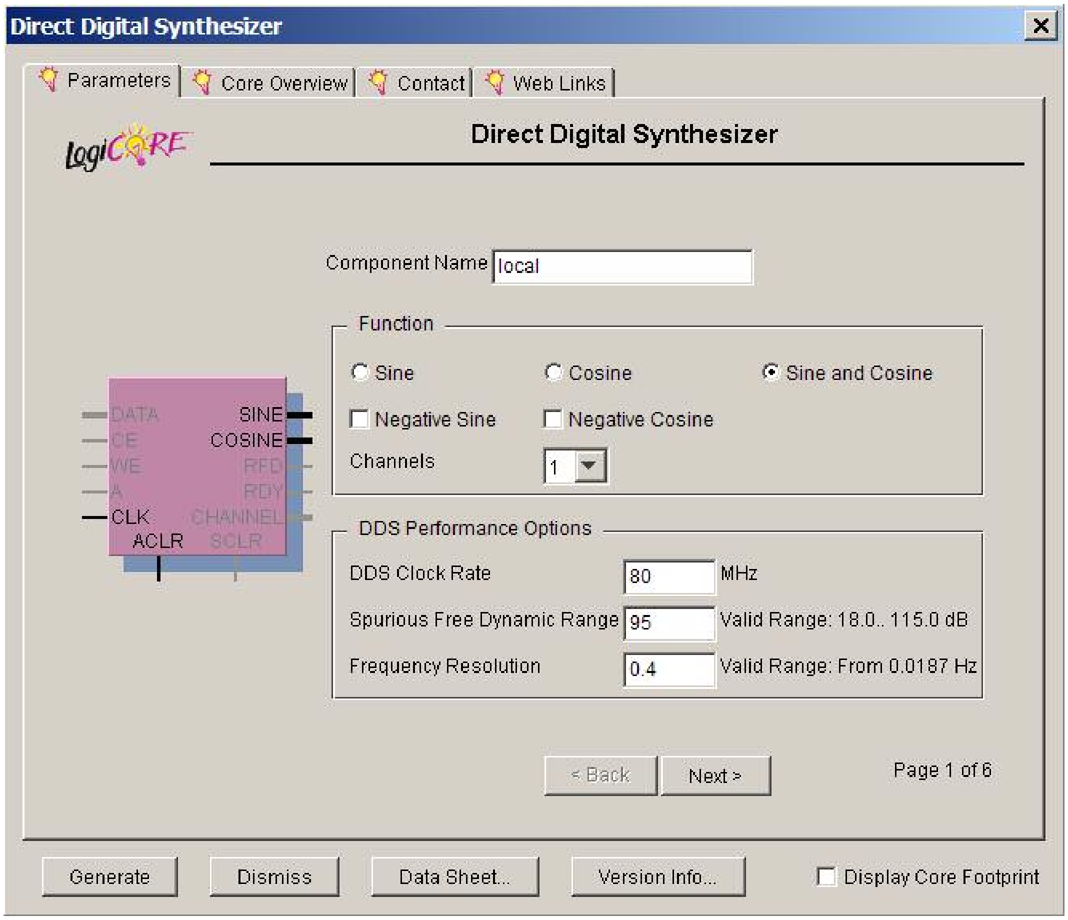


**Figure 26: Schematic of the transmitter, created in Xilinx ISE**



Component Name local

Function

○ Sine ○ Cosine ● Sine and Cosine

□ Negative Sine □ Negative Cosine

Channels 1

DDS Performance Options

DDS Clock Rate 80 MHz

Spurious Free Dynamic Range 95 Valid Range: 18.0.. 115.0 dB

Frequency Resolution 0.4 Valid Range: From 0.0187 Hz

□ Display Core Footprint

**Figure 27: Xilinx IP Core used to create a DDS for use as a local oscillator**

## 5.2 Boxcar filter implementation

Several different designs for the boxcar filter module were explored before the final design was selected. The goal was not only to perform the operations necessary to the algorithm, but also to minimize the consumption of the FPGA resources. The first method of designing this filter was to use one of the Xilinx FIR IP Cores. This process included telling the core GUI how many taps to include, 80 in this case; what bit resolution to use (16 in this case), the symmetry of the taps (symmetric), whether the bits are signed or unsigned (signed), and if the coefficients need to be reloaded at any point (false). Following this procedure, and loading the coefficients (eighty ones) into the core, the core generates a filter, designed in Direct Form, which performs the desired operation. The problem with this algorithm is the overuse of FPGA resources. Instead of eliminating the multipliers it used forty multipliers. Due to the coefficient's symmetry, multipliers can be reused. Finally, since adders only have two inputs, this design will use several in hierarchy to produce a result.

The next implementation involved using an adder, a subtractor, and several flip flops. The input split as one input into the adder and the input into a chain of 79 flip flops. The adder result was stored into a single flip flop. This flip flop output provided the second input to the adder; thus, incoming samples are accumulated. Only 79 flip flops were necessary on the flip flop chain due to the delay through the math chain. This caused the eightieth sample to reach the subtractor at the same time as the input. The output of the flip flop chain was subtracted from the math chain flip flop. This reduced the resource consumption compared to the FIR IP Core. However, now, the algorithm was incorrect. Since the single flip flop in the math chain is before the subtractor, this flip flop will overflow if the input is not alternating ones and zeros. Furthermore, shift registers consume fewer resources than flip flops, allowing for one more size optimization.

The final and best algorithm replaces the flip flop chain with an 80-length, 16-deep shift register. Inputs are stored here and pass into the math block after 80 cycles. The shift register can also be configured for variable length. This is an advantage for flexibility as the ratio of sampling rate to symbol rate determines the integration period. Thus, the boxcar filters can be reused for different M-ary PSK designs.

Instead of using IP Cores to build the necessary adders and subtractors, the same operation can be performed in two lines of VHDL code. First, a signal is instantiated to zero to serve as the memory register. Secondly, the math is performed and, the result is assigned to the output. This code serves the same purpose as blocks designed to add and subtract. The result is the sum of the previous eighty input samples on every clock cycle. The same design techniques are used on the boxcar filter in the early-late-gate operation.

This boxcar filter, however, integrates widths of one bit instead of sixteen. The boxcar schematic is shown, in Figure 28.

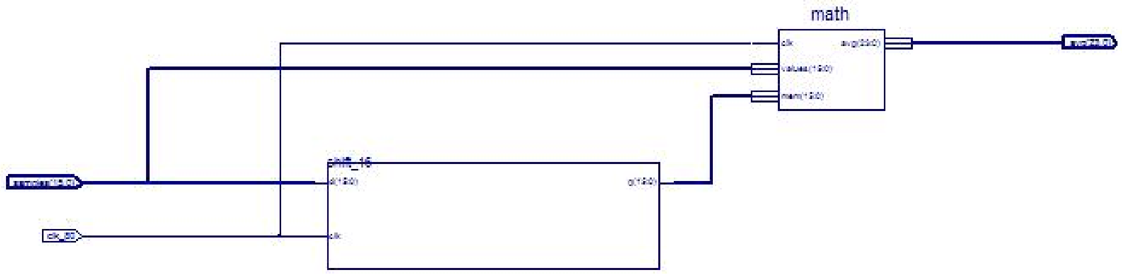


**Figure 28: Schematic of the boxcar filter implementation.**

## 5.3 Loop filter implementation

Since the IIR loop filter is of such a low order, only a few stages are necessary. Two methods of implementing an IIR filter have been considered. The first and most efficient method is the parallel form. This form takes advantage of the parallel processing capabilities of the FPGA. In order to design the filter in this method, first, the coefficients need to be in the transfer function form. An example of converting a second-order form (the default output form of Matlab) to transfer function form is shown in Equation 21.

**Equation 21: Converting a second-order equation to a transfer function.**

$$H(z)=\frac{\left(\beta_1+\beta_2 z^{-1}+\beta_3 z^{-2}\right)}{\left(1-\alpha_1 z^{-1}-\alpha_2 z^{-2}\right)}\frac{\left(\beta_4+\beta_5 z^{-1}+\beta_6 z^{-2}\right)}{\left(1-\alpha_3 z^{-1}-\alpha_4 z^{-2}\right)}$$
$$=\frac{\beta_A+\beta_B z^{-1}+\beta_C z^{-2}+\beta_D z^{-3}+\beta_E z^{-4}}{1-\alpha_A z^{-1}-\alpha_B z^{-2}-\alpha_C z^{-3}-\alpha_D z^{-4}}$$

Using partial fraction expansion, the transfer function will ideally break into several single-order sections. An example of partial fraction expansion is shown in Equation 22.

**Equation 22: An example of using partial fraction expansion on a transfer function.**

$$H(z)=\frac{\beta_A+\beta_B z^{-1}+\beta_C z^{-2}+\beta_D z^{-3}+\beta_E z^{-4}}{1-\alpha_A z^{-1}-\alpha_B z^{-2}-\alpha_C z^{-3}-\alpha_D z^{-4}}$$

$$=k_0+\frac{r_0}{1-p_0 z^{-1}}+\frac{r_1}{1-p_1 z^{-1}}+\frac{r_2}{1-p_2 z^{-1}}+\frac{r_3}{1-p_3 z^{-1}}$$

This would be the ideal structure for any IIR implemented in an FPGA if the partial fraction expansion yields single-order sections. It is ideal because the entire operation could be performed in one cycle. However, with the coefficients used in this design, the expansion yields a double pole equation, thereby nullifying the reason for choosing this structure, as the algorithm would take more than one cycle. The implementation of the filter in this design would yield one branch with a single-order section and one with a second-order section. This is shown in Equation 23.

**Equation 23: The result of using partial fraction expansion on a system with a double pole.**

$$H(z)=k_0+\frac{r_0}{1-p_0 z^{-1}}+\frac{r_1}{\left(1-p_0 z^{-1}\right)^2}$$

The final structure explored in this research is Biquad Direct Form II Transposed. In this form, the second-order structure is used to create the filter. This structure has more latency than the parallel form; however, timing is met since the data rate is equal to the symbol rate, which is much slower than the clock rate. Therefore, several operations can be performed before the result must be known. This schematic is shown in Figure 29.

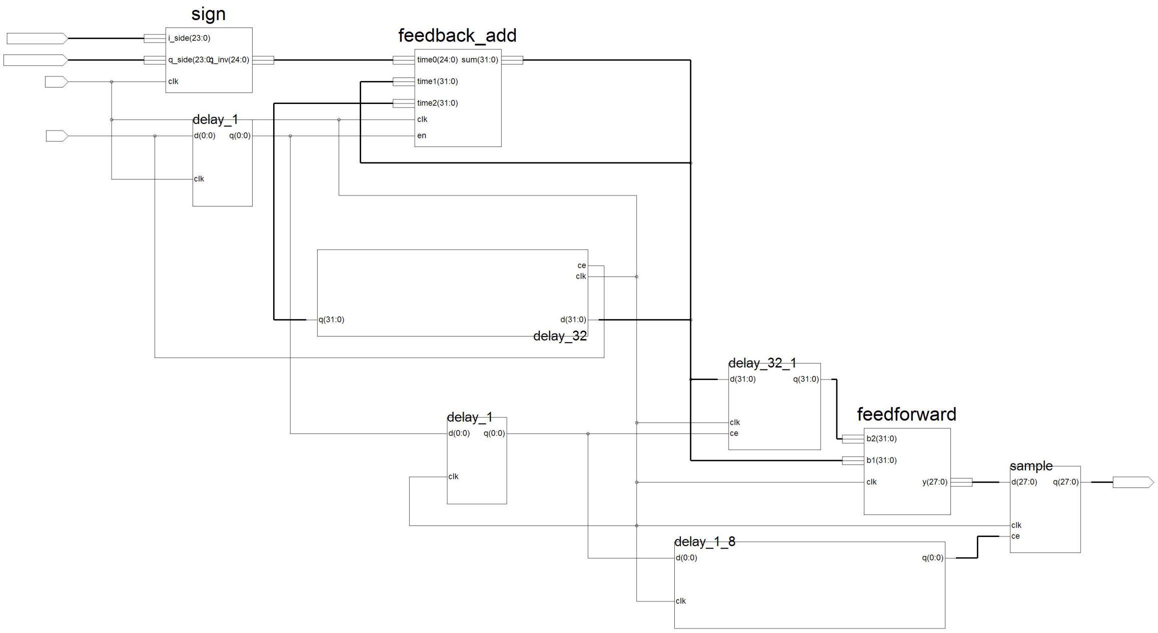


**Figure 29: Schematic of the loop filter implemented in single-order cascaded form.**

The block labeled sign performs the initial operation of multiplying the quad-phase line by the sign of the in-phase line. The next block, labeled "feedback_add," performs the second-order feedback operations. The result is fed into both itself and a delay block such that the result is $H_{fb}(z) = \dfrac{1}{1 - 2z^{-1} + z^{-2}}$. The result of this operation is passed into the feed-forward loop to perform the operations in the numerator. Therefore, the result is the desired $H(z) = \dfrac{0.0253125z^{-1} - 0.025z^{-2}}{1 - 2z^{-1} + z^{-2}}$.

Results are shown below in Figure 30. The output of the filter algorithm differs from the predicted output typically by 1 due to rounding error.

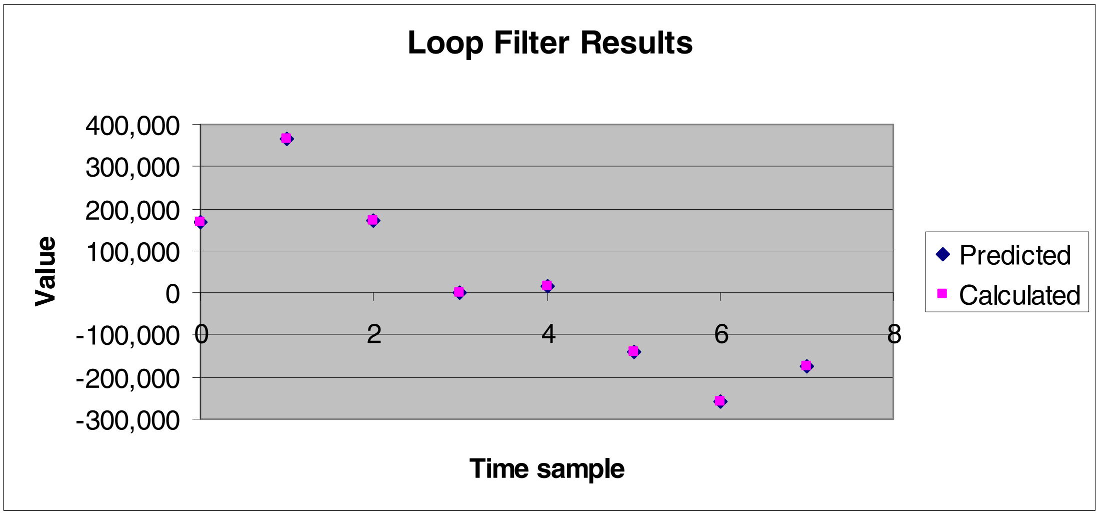


**Figure 30: Results of the loop filter**

## *5.4 Early-Late Gate Implementation*

The early-late gate algorithm takes the un-sampled results of the integration and determines when the sample of the symbol should occur to maximize estimation results. If the algorithm detects that the sampling is occurring on an edge (in between samples), it will sample either earlier or later until the ideal sampling time is determined. If the derivative of the inputs is flat, the ideal sampling time has been reached. These cases are shown in the following figures.

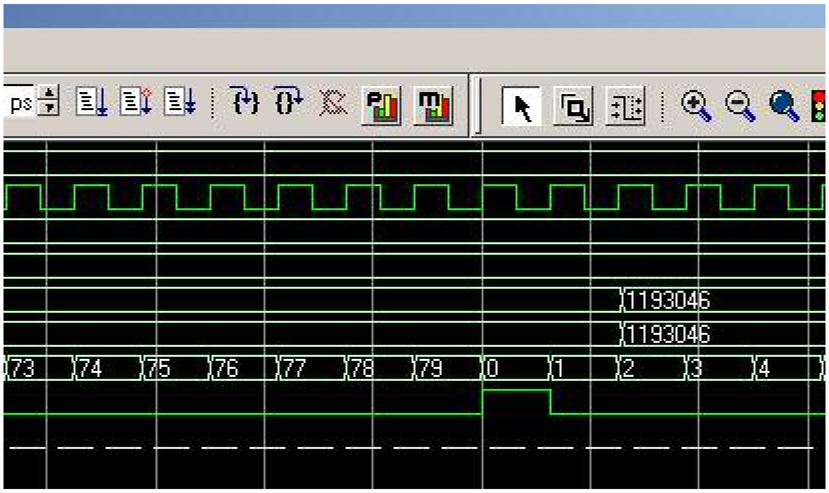
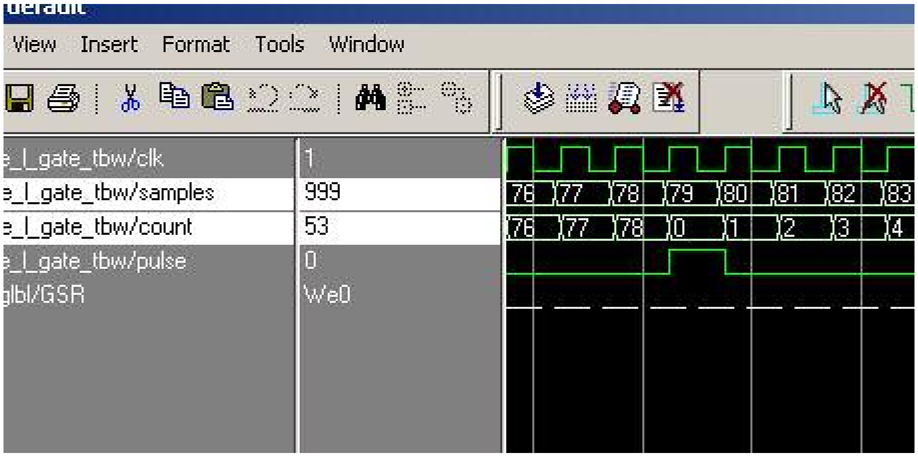
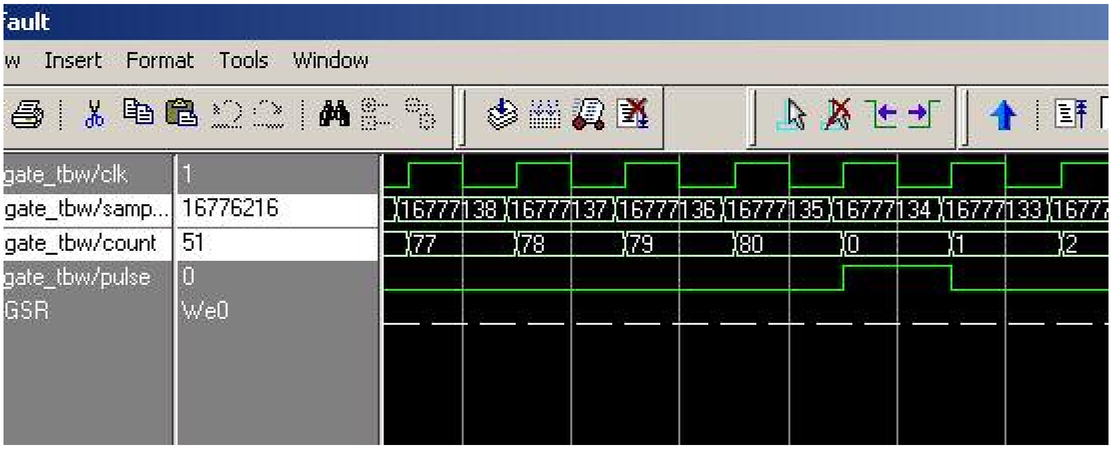


**Figure 31: The result of a synchronized symbol**



```
e_l_gate_tbw/clk       1
e_l_gate_tbw/samples   999
e_l_gate_tbw/count     53
e_l_gate_tbw/pulse     0
glbl/GSR               We0
```

**Figure 32: Sampling faster due to a rising edge**



```
gate_tbw/clk      1
gate_tbw/samp...  16776216
gate_tbw/count    51
gate_tbw/pulse    0
GSR               We0
```

**Figure 33: Sampling slower due to a falling edge**

## *5.5 Transmitter*

The transmitter was implemented exactly as described in the simulation section. The output waveform is shown below in Figure 34.

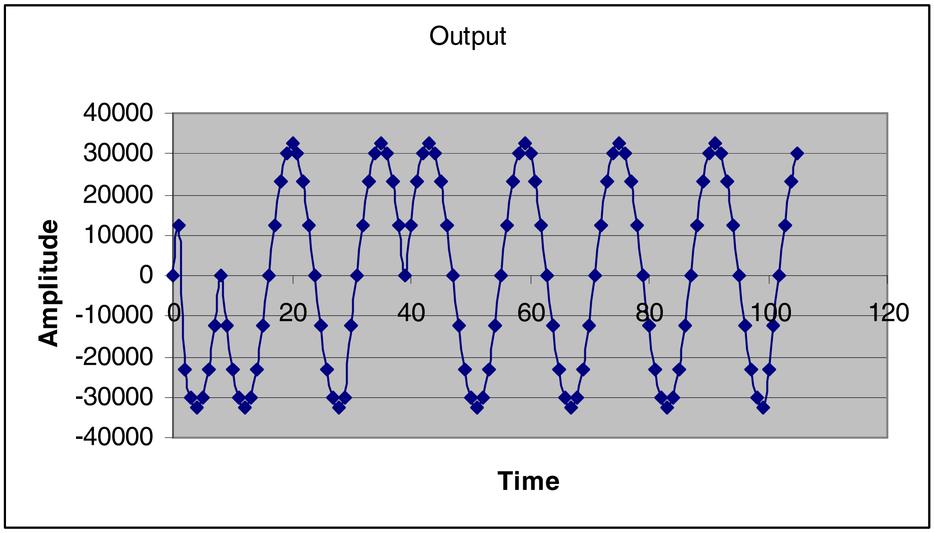
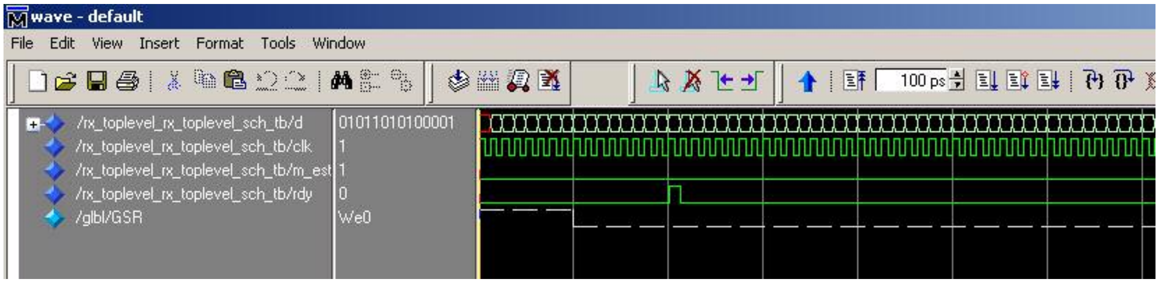


**Figure 34: Simulated waveform of the transmitter from Modelsim**

# CHAPTER 6 CONCLUSION

The results shown above validate that the firmware implementation is a precise realization of the computer simulation.  Given that all the components of the firmware match those of the simulations, the SNR vs. BER curve should also meet those of the simulation.  A sample simulation of the entire receiver is in Figure 35, showing that the output is the estimate and a pulse dictating that the estimate is complete.



```
/rx_toplevel_rx_toplevel_sch_tb/d       01011010100001
/rx_toplevel_rx_toplevel_sch_tb/clk     1
/rx_toplevel_rx_toplevel_sch_tb/m_est   1
/rx_toplevel_rx_toplevel_sch_tb/rdy     0
/glbl/GSR                               We0
```

**Figure 35: Sample simulation of the receiver**

The thesis proposes larger algorithms which should provide better performance.  It also demonstrates means of using this design as the basis of M-PSK transceivers.  The transceiver is capable of communicating 1 Mbaud of data at the provided SNR vs. BER ratio.  Finally, the selected algorithms provide an adequate means of solving for frequency, phase, and bit-time errors.

Algorithms were chosen such that errors are reduced while maintaining resource efficiency.  The resource consumption by the receiver and transmitter is shown in Table 6.

**Table 6: Resource consumption**

| | Receiver | Transmitter | Total |
|---|---|---|---|
| Slices | 1481/9280 | 158/9280 | 1639/9280 |
| Multipliers | 10/88 | 0/88 | 10/88 |
| BRAMs | 5/88 | 4/88 | 9/88 |
| Maximum Freq. | 151.469 MHz | 250.062 MHz | 151.469 MHz |

## 6.1 Future Work

This transceiver is one of the first designed for use with KUAR. It is intended to be a starting point for future designs in the radio. The thesis provides several points where the design can be expanded to more complicated transceivers, such as M-PSK. As it was the original intention of Costas, this design could also be expanded for use as a SSB-AM transceiver.

This transceiver is intended to be a module of a larger design. Since the phase-tracking loop could lock on to an inversion of the signal, the system will need to use a unique symbol for data synchronization. This transceiver also provides a research tool for channel sounding, equalization, multi-path, fading, and other communications issues.

# CHAPTER 7 REFERENCES

[1] J. Costas, “Synchronous Communications,” *Proceedings of the IEEE*, vol. 44, p. 1713-1718, 1956.

[2] Spectrum Policy Task Force, “Spectrum Policy Task Force Report ET Docket No. 02-135,” U. S. Federal Communications Commission, 2002.

[3] J. Mitola, “Cognitive Radio for Flexible Mobile Multimedia Communications,” in *IEEE International Workshop on Mobile Multimedia Communications*, 1999, p. 3-10.

[4] F. Weidling, D. Datla, V. Petty, P. Krishnan, and G. J. Minden, “A Framework for R.F. Spectrum Measurements and Analysis,” in *Proceedings of IEEE Dynamic Spectrum Access Networks 2005*, Baltimore, Maryland, 2005.

[5] DARPA XG Working Group, “XG Policy Language Framework, Request for Comments,” version 1.0, prepared by BBN Technologies, Cambridge, Massacusetts, USA, April 2004.

[6] C. Serra, “SDR Forum,” 2006, http://www.sdrforum.org/.

[7] G. J. Minden, “KU Agile Radio Overview,” technical report, University of Kansas, Lawrence, Kansas, 2005.

[8] C. Dick, “The Platform FPGA: Enabling the Software Radio,” in *Proceedings of the 2002 Software Defined Radio Technical Conference and Product Exposition*, 2002.

[9] J. Mitola, “Software Radios Survey, Critical Evaluation and Future Decisions,” *IEEE Aerospace and Electronic Systems Magazine*, vol. 8 p. 25-36, 1993.

[10] BEE Home Page, Berkeley Wireless Research Center, University of California, Berkeley, 2006, http://bwrc.eecs.berkeley.edu/Research/BEE/.

[11] J. Steinheider, V. Lum, J. Santos, “Field Trials of an All-Software GSM Basestation,” Vanu, Inc, 2003.

[12] A. Chiu and J. Forbess, “A Handheld Software Radio Based on the iPAQ PDA: Software,” in *Proceedings of the 2003 Software Defined Radio Technical Conference*, 2003.

[13] J. Forbess and M. Wormley, “A Handheld Software Radio Based on the iPAQ PDA: Hardware,” in *Proceedings of the 2003 Software Defined Radio Technical Conference*, 2003.

[14] B. Massey, NWACC/PSU SDR, “WebHome – sdr,” 2006, http://wiki.cs.pdx.edu/~sdr/.

[15] “SDR-3000 Series Software Defined Radio Transceiver Platform,” Spectrum Signal Processing, 2005.

[16] “JTRS SDR Kit,” ISR Technologies, 2006.

[17] J. Gevargiz, “Performance Analysis of an all Digital BPSK Demodulator,” in *Proceedings of IEEE Global Telecommunications Conference*, vol. 3, p. 1670-1676, 1993.

[18] J. Holmes, “Tracking Performance of the Filter and Square Bit Synchronizer,” *IEEE Transactions on Communications*, vol. 28 (8), p. 1154-1158, 1980.

[19] M. Rice, “Introduction to Digital Communication Theory,” 2004, http://www.ee.byu.edu/class/ee485public/ee485.fall.04/.

[20] J. Statman and W. Hurd, “An Estimator-Predictor Approach to PLL Loop Filter Design,” *IEEE Transactions on Communications*, vol. 38 (10), p. 1667-1669, 1990.

[21] S. Mirabbasi, and K. Martin, “Design of Loop Filter in Phase-Locked Loops,” *IEEE Electronics Letters*, vol. 35 (21), p. 1801-1802, 1999.

[22] C. Cahn, “Improving Frequency Acquisition of a Costas Loop,” *IEEE Transactions on Communications*, vol. 25 (12), p. 1453-1459, 1977.

[23] J. Berner, J. Layland and P. Kinman, “Flexible Loop Filter Design for Spacecraft Phase-Locked Receivers,” *IEEE Transactions on Aerospace and Electronic Systems*, vol. 37 (3), p. 957-964, 2001.

[24] C. Pomalaza-Ráez and S. Mohan, “Application of Tree Search Algorithms to Bit Synchronization,” in *Proceedings of the 33rd Midwest Symposium on Circuits and Systems*, vol. 2, p. 1026-1029, 1991.

[25] C. Georghiades, “Synchronization,” *The Communications Handbook*, 2nd ed., Ed. J. Gibson, Boca Raton: CRC Press, 2002.

[26] D. Judd, “Data Synchronization Simulation Using the Mathworks Communications Toolbox,” *IEEE International Conference on Communications*, vol. 2, p. 706-710, 1996.

[27] F. Gardner, “Interpolation in Digital Modems – Part I: Fundamentals,” *IEEE Transactions on Communications*, vol. 41 (3), p. 501-507, 1993.

[28] X. Liu and A. Willson, “An New Interpolated Symbol Timing Recovery Method,” *IEEE International Symposium on Circuits and Systems*, vol. 2, p. 569-572, 2004.

[29] Z. Hang and M. Renfors, “A New Symbol Synchronizer with Reduced Timing Jitter for QAM Systems,” in *Proceedings of the IEEE Global Telecommunications Conference*, vol. 2, p. 1292-1296, 1995.

[30] J. Hwang and C. Chu, “FPGA Implementation of an All-Digital T/2-Spaced QPSK Receiver with Farrow Interpolation Timing Synchronizer and Recursive Costas Loop,” *Proceedings of 2004 IEEE Asia-Pacific Conference on Advanced System Integrated Circuits*, p. 248-251, 2004.

[31] Xilinx LogiCore, “Direct Digital Synthesizer, v. 5.0,” Xilinx, DS246, Apr. 2005.

[32] D. DePardo, “5 GHz Block Diagram,” technical report, University of Kansas: Agile Radio Project, Lawrence, Kansas, 2005.

[33] L. Searl, “Radio Digital Board Assembly Bottom,” technical report, University of Kansas: Agile Radio Project, Lawrence, Kansas, 2005.

[34] Wikipedia, “Doppler Effect,” 2006, http://en.wikipedia.org/wiki/Doppler_effect.

[35] K. Shanmugan and A. Breipohl, *Random Signals: Detection, Estimation and Data Analysis*, New York: Wiley, 1988.

[36] C. Bergstrom, S. Chuprun, S. Gifford and G. Maalouli, “Software Defined Radio (SDR) Special Military Applications,” in *Proceedings of the IEEE Military Communications Conference*, vol. 1, p. 383-388, 1988.

[37] S. Blust, “Modular Multifunction Information Transfer System Forum on Sofware Defined Radio,” 1998, in *Proceedings of the 8th Annual International Symposium on Advanced Radio Technologies*, http://www.its.bldrdoc.gov/isart/art98/slides98/blust /blus_s_all.pdf.

[38] S. Blust, “Software Based Radio,” *Software Defined Radio*, Ed. W. Tuttlebee, West Sussex: John Wiley & Sons, Ltd., 2002, pp. 3-22.

[39] S. Blust, “Stephen Blust Presentation to SDR Workshop in Tokyo on 17 October 2001,” 2002, *Report on Global Regulatory Views on SDR and Radio Software Download for RF Reconfiguration (Working Paper)*, http://sdrforum.org/MTGS /mtg_27_feb02/02_i_0008_v0_00_dl_reg_01_25_02.pdf.

[40] A. Cinquino and Y. Shayan, "A Real-Time Software Implementation of an OFDM Modem Suitable for Software Defined Radios," in *Proceedings of the 2004 IEEE Canadian Conference on Electrical and Computer Engineering*, vol. 2, p. 697-701, 2004.

[41] L. Erup, F. Gardner and R. Harris, "Interpolation in Digital Modems – Part II: Implementation and Performance," *IEEE Transactions on Communications*, vol. 41 (6), p. 998-1008, 1993.

[42] A. Haghighat, "A Review on Essentials and Technical Challenges of Software Defined Radio," in *Proceedings of the IEEE Military Communications Conference*, vol. 1, p. 377-382, 2002.

[43] E. Lee and D. Messerschmitt, "Synchronous Data Flow," *Proceedings of the IEEE*, vol. 75 (9), p. 1235-1245, 1987.

[44] J. MacLeod, T. Nesimoglu, M. Beach and P. Warr, "Enabling Technologies for Software Defined Radio Transceivers," in *Proceedings of the IEEE Military Communications Conference*, vol. 1, p. 354-358, 2002.

[45] D. Messerschmitt, "Synchronization in Digital System Design," *IEEE Journal on Selected Areas in Communications*, vol. 8 (8), p. 1404-1419, 1990.

[46] J. Mitola, "SDR Architecture Refinement for JTRS," in *Proceedings of the IEEE Military Communications Conference*, vol. 1, p. 214-218, 2000.

[47] Wikipedia, "Moore's Law," 2006, http://en.wikipedia.org/wiki/Moore's_law.

[48] S. Rajagopal, S. Rixner and J. Cavallaro, "A Programmable Baseband Processor Design for Software Defined Radios," in *Proceedings of the IEEE Midwest Symposium on Circuits and Systems*, vol. 45 (3), p. 413-416, 2002.

[49] A. Shah, "An Introduction to Software Radio," Cambridge: Vanu, Inc, 2002.

[50] T. Shono and M. Matsui, "Software Defined Radio Prototype (II) – Implementation and Evaluation of IEEE 802.11 Wireless LAN," *NTT Technical Review*, vol. 1 (4), p. 24-30, 2003.

[51] S. Srikanteswara, R. Palat, J. Reed and P. Athanas, "An Overview of Configurable Computing Machines for Software Radio Handsets," *IEEE Communications Magazine*, vol. 41 (7), p. 134-141, 2003.

[52] J. Steinheider, V. Lum and J. Santos, "Field Trials of an All-Software GSM Basestation" in *Proceedings of the 2003 Software Defined Radio Technical Conference*, Orlando, 2003.

[53] A. Veeragandham, *OFDM Testbed Analysis and Implementation Framework*, M.S. Thesis, The University of Kansas, Lawrence, Kansas, 2005.

[54] GNURadio, “UniversalSoftwareRadioPeripheral,” technical report, GNU Radio, <http://comsec.com/wiki?UniversalSoftwareRadioPeripheral>.