An FPGA-Based 256-QAM Modem with Carrier Recovery

A Thesis Submitted to the College of

Graduate Studies and Research

in Partial Fulfillment of the Requirements

for the Degree of Master of Science

in the Department of Electrical Engineering

University of Saskatchewan

Saskatoon

by

Nikolaj Larionov

Fall 2002

PERMISSION TO USE

In presenting this thesis in partial fulfilment of the requirements for a degree of Master of Science from the University of Saskatchewan, the researcher agrees that the Libraries of this University may make it freely available for inspection. The researcher further agrees that permission for copying of this thesis in any manner, in whole or in part, for scholarly purposes may be granted by the professor who supervised this thesis work or, in his absence, by the Head of the Department or the Dean of the College of Graduate Studies and Research at the University of Saskatchewan. Any copying or publication or use of this thesis or parts thereof for financial gain without the researcher's written permission is strictly prohibited by law. Due recognition shall be given to the researcher and to the University of Saskatchewan in any scholarly use which may be made of any material in this thesis.

Requests for permission to copy or to make other use of material in this thesis in whole or part should be addressed to:

Head of the Department of Electrical Engineering

57 Campus Drive, University of Saskatchewan

Saskatoon, Saskatchewan, Canada

S7N 5A9

ABSTRACT

The ever-increasing demand for more access to digital information has forced data providers and data switching networks to offer higher data rates over the same bandwidth. The implementation of a high performance digital communication system is a significant task. The researcher has developed algorithms and modulation schemes that sustain rates of several megabits per second. However, these algorithms are quite complex and require large digital circuits. Therefore, they are unwieldy for direct digital implementation. In order to alleviate this effect, the researcher has rendered and reorganized these algorithms for effective implementation in order to improve the efficiency, simplicity, scalability, flexibility and gate count of the digital communications system design.

In this thesis the proposed design of a high-speed 256-QAM (Quadrature Amplitude Modulation) modem was developed for the effective implementation in FPGAs (Field Programmable Gate Arrays). Although FPGAs are less efficient in terms of the gate count than ASICs (Application Specific Integrated Circuits), their main advantage over ASICs is the in-the-field programming. Therefore, the architectures of FPGAs are designed to be very flexible and reusable. This thesis proposes solutions for efficient multirate pulse shaping and matched filters, the CORDIC (Coordinate Rotation Digital Computer) algorithm and the DPLL (Digital Phase Locked Loop) for carrier recovery.

ACKNOWLEDGEMENTS

The researcher's primary acknowledgement goes to his supervisor, Dr. Eric Salt. It was a privilege working with him. The knowledge and techniques acquired during the Communications 2, Verilog HDL and Random Variables in Engineering courses given by him were invaluable to the researcher. Without the guidance, support and wisdom of Dr. Eric Salt it would have been impossible to achieve the level of expertise that this project required.

The researcher would like to extend his appreciation and gratitude to the management and staff of Canada's communications technology research consortium - TRLabs (Saskatoon), particularly Mr. Garth Wells, for the invaluable technical support, and the modern facilities to complete the research. The researcher's thanks go to the Natural Sciences and Research Council of Canada (NSERC), as well as to the department of Electrical Engineering of the University of Saskatchewan for providing the researcher with generous scholarships.

The researcher would really like to thank his parents John and Olga Levasseur, his relatives and friends for their patience and encouragement during this challenging time. The researcher praises the Lord for the gifts, given to the researcher and the strength and persistence that He blessed the researcher with, in order to reach this high level of research.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ABREVIATIONS

A/D, ADC	Analog-to-Digital (Converter)
ADSL	Asymmetric Digital Subscriber Line
AGC	Automatic Gain Control
ASIC	Application Specific Integrated Circuit
CLB	Configurable Logical Block
CORDIC	Coordinate Rotation Digital Computer
D/A, DAC	Digital-to-Analog (Converter)
dB	decibel
DDS	Direct Digital Synthesis
DPLL	Digital Phase Locked Loop (Digital PLL)
DSP	Digital Signal Processing
FIR	Finite Impulse Response
FPGA	Field-Programmable-Gate-Array
HDL	Hardware Descriptive Language
IIR	Infinite Impulse Response
IF	Intermediate Frequency
I/O	Input/Output
ISI	Inter-symbol Interference
LUT	Look-Up Table
L	Number of Samples per Symbol

LE	Logical Element
L.O.	Local Oscillator
LP	Low-Pass
M	Decimation Rate
msd	mean square difference
MSymbols	Megasymbols
Mbps	Megabits per second
MSB	Most Significant Bit
NCO	Numerically Controlled Oscillator
PLL	Phase Locked Loop
PN	Pseudo-noise
ppm	Parts per million
QAM	Quadrature Amplitude Modulation
QPSK	Quadrature Phase Shift Keying
rad/samples	Radians per sample
RC	Raised-Cosine
RF	Radio Frequency
ROM	Read Only Memory
RX	Receiver
SAW	Surface Acoustic Wave (Filter)
S/P	Serial-to-Parallel (Converter)
SRRC	Square Root Raised Cosine (Filter)
TX	Transmitter

1 INTRODUCTION

To strive consciously for an object and to engage in engineering – that is, incessantly and eternally to make new roads, wherever they may lead.

Fyodor Dostoyevsky

1.1 Thesis Background

1.1.1 Cable System Fundamentals

The Cable TV industry has come a long way since 1948 when John Walson, for the first time, ran a twin-lead pair antenna line from the top of the hill to his appliance store, in order to bring television to the people who lived in the areas unable to receive conventional broadcast signals [1]. Since then the cable television industry has focused on becoming the major TV delivery medium by constantly expanding its subscriber base and covering larger geographical areas. The significant technological advancements in the Cable TV industry have allowed it to supply its subscribers with higher quality digital video and sound replacing the older analog technologies. Currently in North America, cable TV delivers a wide range of choice of information, news and entertainment to 105 million subscribers [1].

Cable TV uses large, high gain antennas to receive satellite broadcast signals. A headend cable termination system distributes these signals to the home environment. A simplified cable system is illustrated in Figure 1.1.

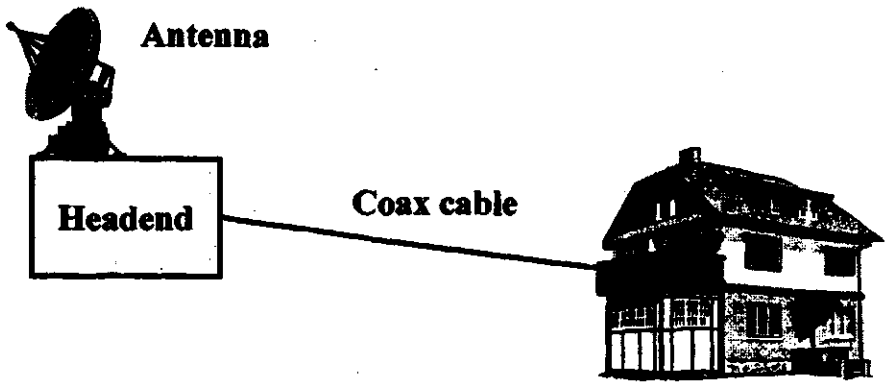


Figure 1.1 Simplified diagram of traditional cable system.

According to the report presented by the senior-level analysts of The Insight Research Corporation, a major telecommunications analysis firm [2], today the number of cable TV subscribers is reaching the saturation point, forcing cable operators to look elsewhere for revenues. Therefore, cable operators are now selling multimedia services, digital video, video-on-demand, pay-per-view, telephony, and high-speed Internet access services.

Internet access became extremely popular with the advent of the World Wide Web in 1995. At that time standard telephone modems supported digital data transmission rates far less than the maximum capacity of the telephone lines. The standard 4-KHz (kilohertz) bandwidth of a telephone channel was incapable of supporting high rate services for video and graphics applications. The slow data transmission traffic resulted in many frustrated subscribers. Since 1995, the basic broadband cable system has greatly improved the speed of Internet access, to a point where it is now hundreds of times faster

than voice band telephone lines. This high-speed transmission technology connects end users to the backbone network that runs between cities all over the world.

The two-way connection between the headend and the home environment is either a coaxial (coax) cable, a pair of twisted copper wires or modern hybrid fiber/coax systems. Digital signals cannot be sent directly over such media. They must be converted to high frequency analog signals, called symbols, for transmission. A device that performs this conversion is called a modulator. The device at the receiving end that converts the high frequency analog signals (symbols) back to digital signals is called a demodulator. In order to transmit and receive information a modulator and demodulator are required at each end of the physical wire link. A device that can modulate and demodulate is called a modem. When digital information is sent over a communication link it can be received error free and therefore does not experience degradation.

Today subscribers continue to simultaneously receive cable television service and high-speed Internet data on cable modems with the help of a one-to-two splitter [1]. The TV signal is directed to the set-top-box, which is connected to the TV-set while the Internet data is demodulated by the cable modem, which is in turn connected to a personal computer (PC). The set-top-box is necessary not only for the demodulation but also to convert (decode) the digital signal to appropriate video and audio analog signals, because most TV sets are still analog. The cable system architecture is illustrated in Figure 1.2.

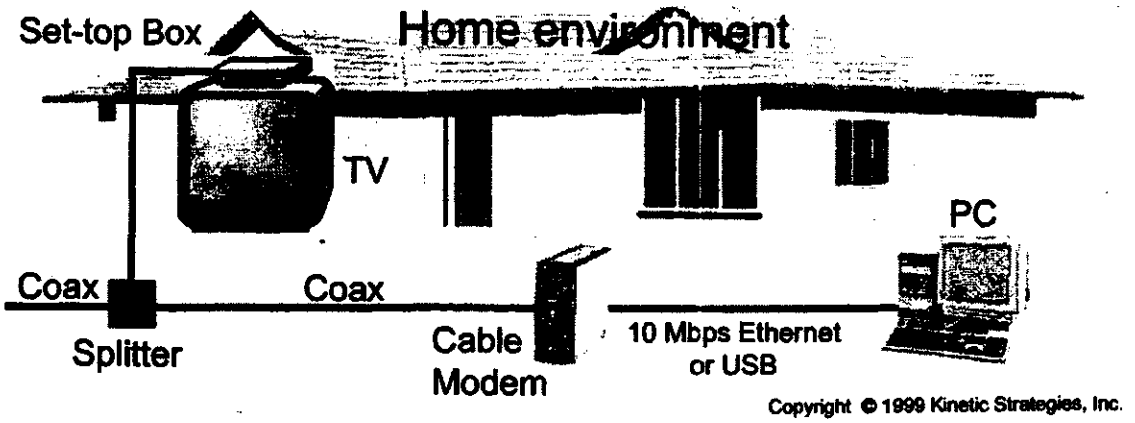


Figure 1.2 Cable system architecture – home environment (source **http://www.townisp.com/**).

The demand for high-speed transmission technologies such as the cable system is steadily increasing [2]. This demand as well as market competition is the driving force behind the development of the Cable TV industry. Digital television is poised to become a sales leader, while high speed Internet access is booming [3]. This emerging technology requires a particular modulation for the high speed transmission. Currently the most widely used modulation for the downstream (from the headend to the house) connection is 64-QAM (Quadrature Amplitude Modulation) that supports a 27-36 Mbps data rate. Currently Cable TV is distributed over 750-MHz of spectrum that is divided into ~6-MHz channels.

The Digital Subscriber Line (DSL), which provides a 1.5-52-Mbps data rate, is the main competitor for the cable companies. In order to stay competitive one of the possible strategies for the cable companies is to move from 64-QAM to 256-QAM, which would increase the data rate by a factor of 8/7 to approximately 42-Mbps. A 256-QAM signal,

in theory, can convey 8 bits of information per second per Hertz, often referred to in its shortened form as 8 bits per Hz, instead of 6 bits per Hz, transmitted in 64-QAM. Therefore, the 256-QAM modulation is favoured for the downlink in cable modems due to the greater amount of data being transmitted per time period using the same bandwidth as in Cable TV [3].

A cable data system consists of many different technologies and standards. In order to develop a mass market for cable modems, the products from different vendors have to be interoperable. The International Telecommunications Union (ITU) described the 256-QAM modulation in the J.83 standard [4], which regulates high-speed transmission in Europe and North America.

Applying 256-QAM modulation is an efficient solution to transmit information at a relatively high rate. Since the outdated analog design could not easily sustain the required high transmission rates and could not be efficiently controlled or modified, therefore the digital design became extremely popular.

1.1.2 FPGA Basics

Today, a digital design is usually implemented in an application specific integrated circuit (ASIC). An ASIC is an integrated circuit (IC) that is programmed at the time of manufacture in order to implement a particular function (e.g. cable modem design). Once there is a demand for the new digital design implementation the engineers design and build the first ASIC chip of a batch. This is usually a slow process that can last up to

6 months. Once the first ASIC chip comes back from the manufacturer it takes a few months to thoroughly test it. If, by that time, the produced design is still profitable, the ASIC specialized companies continue the production. However, once the ASIC chip is well tested it is very easy to replicate it in large volumes. Therefore, the design in ASIC is suitable for the high-volume production market [5] but its up-front, non-recurring, engineering costs are quite high.

Less than a decade ago the prime market for the cable companies was cable TV and later, Internet access services for the home environment. This required a mass production of relatively standard design components. Today cable companies are trying to penetrate the work environment. Different clients demand particular services, which may need some modification of the design. Companies that produce ASIC chips cannot satisfy this low volume, urgent demand. There is a constant need for innovative features that would improve the quality of service of the digital communication system. With the pressures of today's competitive environment, the time to market is of critical importance to the success of new products. The need for a new form of hardware, which is more flexible and can be easily and quickly updated, is therefore evident [5].

In a time of increasing design complexity along with shortened time-to-market requirements, Field Programmable Gate Arrays (FPGAs) are providing a timely solution.

FPGA technology was first suggested by Gerald Estrin from the University of California at Los Angeles in 1960. However, it became popular only in the last decade. Today

Altera, Xilinx, Actel and other companies are manufacturing modern FPGAs that are very competitive with ASICs.

FPGAs are implemented using a different concept than ASICs. Instead of performing one particular function FPGAs can be easily reprogrammed by the customer ("in the field") to suit many different applications. Currently, the FPGA chip fabrication is more expensive than the ASIC chip. However, the produced FPGA chip can be used for many different applications and allows the design to be quickly updated.

The FPGA chip consists of an array of configurable logical blocks (logical elements) - CLBs that are made up of gates, registers and Look-Up-Tables (LUTs).

An FPGA is comprised of an array of configurable logic blocks that perform the functions of logic gates. These gates can be considered as switches that implement basic logical operations - AND, OR, NAND, NOR, XOR, XNOR and NOT. By combining a different logical block function and the connections (routes) between the logical blocks, a different design is quickly developed.

Modern engineers do not have to spend a great deal of time specifying each element in the digital design. Instead, engineers develop behavioural descriptions that are synthesized into circuit by a compiler. The design behaviour can be described using a Hardware Descriptive Language (HDL) and synthesized by a compiler for that HDL. System designers can quickly program the solution of the particular design, compile it

and program an FPGA chip from files generated by the compiler. Using the FPGA, it is possible to test the design immediately after it has been compiled [6].

At the moment, it is usually impossible to optimize a reconfigurable chip (e.g. FPGA) to the same level of performance as a chip that has been specifically designed for a particular application (e.g. an ASIC). An FPGA requires a high level of programming to optimize it and its high maintenance costs also make it more suitable for a low output, high added value market [6].

1.2 Thesis Main Objectives

The prime interest of this project was an optimized algorithm for recovering the carrier of a 256-QAM modulated signal used in cable modems. The carrier is recovered as part of the demodulation process. It is important that the carrier be recovered with no phase error. This is a challenging task that requires reasonably complex circuits. The algorithm, which uses the Digital Phase Locked Loop (DPLL) function, must be suitable for implementation in an FPGA and yet not sacrifice performance. This algorithm was developed in the Matlab Simulink simulator and optimized for the implementation in FPGA.

The other task of this project was to design efficient cable modem blocks. The digital frequency converters, as well as the digital square root raised cosine (SRRC) pulse shaping and matched filters, were optimized for efficient implementation in FPGA. The optimization is measured by the area of the FPGA chip required.

1.3 Thesis Organization

The thesis contains 7 Chapters. Chapter 1 gives the introduction to this thesis and the rest of the thesis is divided into the following parts:

Some fundamental digital functions and basic principles of 256-QAM modulation as well as the I-Q constellation representation in the rectangular and polar coordinates are discussed in Chapter 2. Shaping / matched filters along with the polyphase structures for their hardware implementation are proposed in Chapter 3. The efficiency of the structure, which reduces the complexity of the filter's implementation, is discussed. The problem of frequency conversion and a method utilizing the Coordinate Rotation Digital Computer (CORDIC) algorithm are presented in Chapter 4. The carrier offset concept, its estimation and compensation, accomplished by applying a Digital Phase Locked Loop (DPLL) are given in Chapter 5. Finally, the results of the experiments are discussed in Chapter 6. The conclusions and a possible future research project are presented in Chapter 7.

2 BASIC PRINCIPLES OF QUADRATURE AMPLITUDE MODULATION

This chapter is a brief introduction to a type of carrier modulation known as Quadrature Amplitude Modulation (QAM). The data mapping as well as the QAM modulation constellation and a phasor model are explained. The theory of the raised cosine (RC) and square root raised cosine (SRRC) filters, essential for the shaping and matched filters, is discussed. Upsampling, downsampling and some other fundamental digital communication functions, which are necessary for implementing a 256-QAM digital modem with carrier recovery, are also described.

2.1 The Concept of Quadrature Amplitude Modulation

The demand for digital communication services has been steadily growing. This puts pressure on the service providers to supply higher transmission rates. In order to increase the information carrying capacity of the cable, a complex modulation scheme is used. For this purpose the modern digital cable system uses Quadrature Amplitude Modulation (QAM).

The basic principle of digital communication is the ability to transmit a high frequency sinusoidal carrier, whose parameters of amplitude, frequency or phase are modulated by

a digital baseband signal. At the receiver side, the transmitted digital information is recovered during the demodulation process. The QAM modulation allows the concurrent modulation of the amplitude $A(t)$ and phase $\theta(t)$ of the sinusoidal carrier [7]. The QAM signal can be represented as follows:

$$s(t) = A(t)\cos(\omega_c t + \theta(t)), \tag{2.1}$$

where ω_c is the carrier frequency, expressed in radians per second. The carrier is not modulated by the digital inputs but rather by a baseband signal represented by the inphase (I) and quadrature (Q) components. The inphase and quadrature components modulate the amplitudes of two sinusoidal carriers, orthogonal to each other. The QAM signal is produced by combining the amplitude modulated quadrature carrier signals:

$$s(t) = I(t)\cos\omega_c t - Q(t)\sin\omega_c t \tag{2.2}$$

The baseband signal is produced by shaping the digital input. The shaping is an essential part of the QAM modulation due to the following important restriction.

In North America most communication systems are regulated by the Department of Communications (Canada) or the Federal Communications Commission (US). These regulations restrict the transmission power and the bandwidth of the transmitted signals. One of the most severe restrictions is on the out-of-band power. For example, the signal on a cable system is limited to a specific bandwidth that defines the channel. The available bandwidth is shared with other channels, which occupy the spectrum near each other. The radio frequency (RF) channel is used as a pipe through which the signal carrying the data passes. In order to stay within the allocated channel the data sequence

must be shaped to reduce the out-of-band power. The shaping of the digital input can be explained as follows.

In 256-QAM a serial to parallel (S/P) operation converts every 8 bits of a serial data stream into two lower rate streams of I_m and Q_m words as shown in Figure 2.1. The I_m and Q_m words define the amplitudes of impulses converted to symbols by the pulse shaping filters. Using a delta function these impulses can be represented as:

$$\begin{cases} I_m \delta(t - mT) \\ Q_m \delta(t - mT) \end{cases} \tag{2.3}$$

where I_m and Q_m have values of $\pm 1, \pm 3, \pm 5, \pm 7, \pm 9, \pm 11, \pm 13, \pm 15$ and T is the time between symbol transmissions.

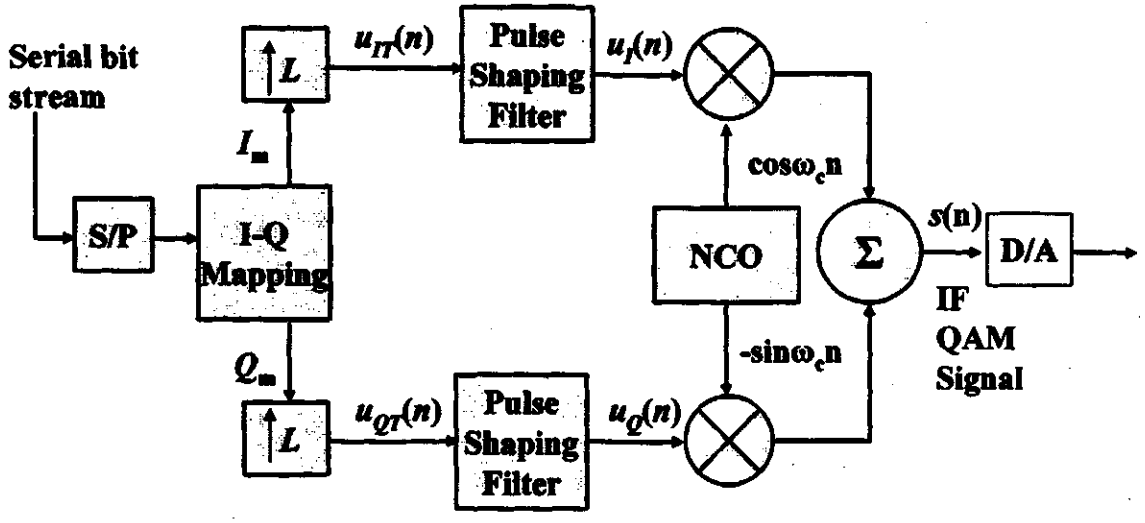


Figure 2.1 Schematic diagram of QAM modulator.

Figure 2.2 shows the 256-QAM I-Q constellation, which is formed by plotting I_m and Q_m against each over. Every point in this 256-QAM constellation represents a unique 8 bit word.

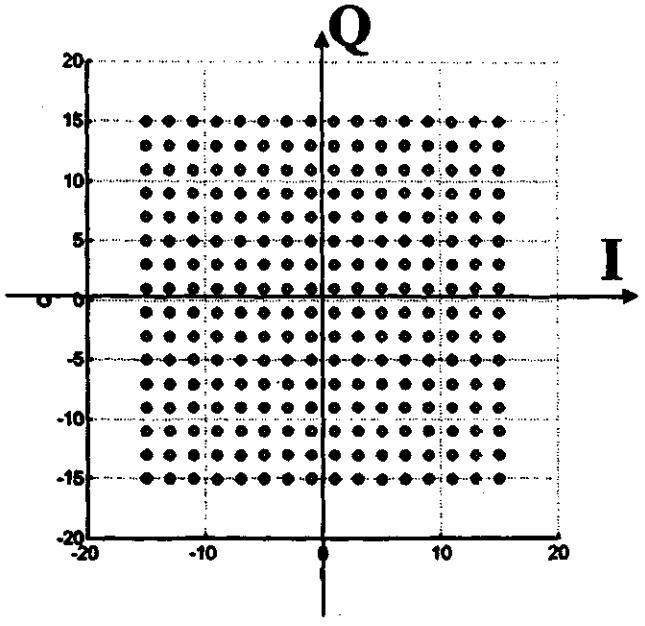


Figure 2.2 256-QAM I-Q constellation.

On the other hand, every point in the *I-Q* constellation can also be represented as a complex number, where I is the real and Q is the imaginary component as shown in Figure 2.3. The complex number can also be shown in a polar form with the radius A_m and the phase θ_m given by:

$$A_m = \sqrt{I_m^2 + Q_m^2} \text{ and}$$
$$\theta_m = \arctan\left(\frac{Q_m}{I_m}\right). \tag{2.4}$$

The polar representation of the complex number $I_m + jQ_m$ is

$$A_m e^{j\theta_m} \tag{2.5}$$

This is a phasor representing the amplitude (radius) and phase of the carrier. The phasor is rotating at an angular velocity ω_c, which is the frequency of the carrier expressed in radians per second [7].

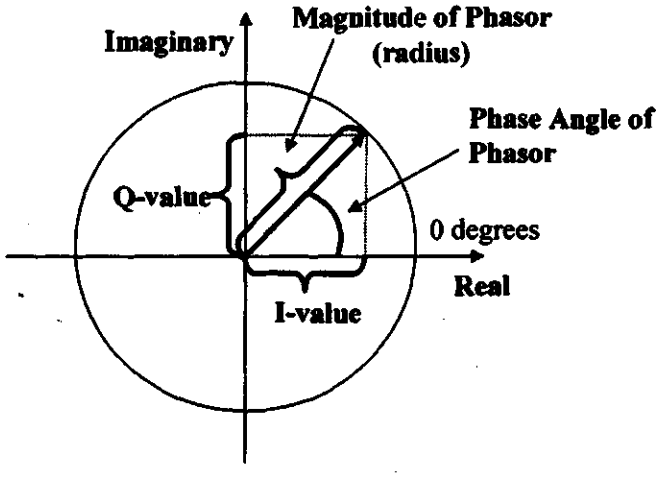


Figure 2.3 Phasor representation of point in constellation.

Pulse shaping is performed using pulse shaping filters. Before being sent to the shaping filters, the data sequences I_m and Q_m are upsampled, which means they are padded with zeros.

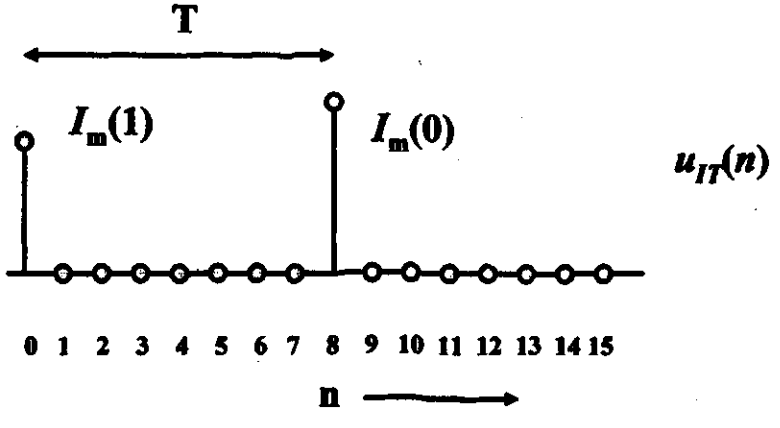


Figure 2.4 Zero padding.

The number of zeros stuffed between I_m (or Q_m) is L-1, where L is the number of samples per symbol. The upsampling process is shown in Figure 2.4 and can be represented as:

$$u_{IT} = \begin{cases} I_m\left(\frac{n}{L}\right) & n = mL \\ 0 & otherwise \end{cases}$$
$$u_{QT} = \begin{cases} Q_m\left(\frac{n}{L}\right) & n = mL \\ 0 & otherwise \end{cases} \tag{2.6}$$

In this thesis 8 samples per symbol are used, thus the number of stuffed zeros between I_m (or Q_m) is equal to 7. The corresponding symbol of the upsampling process is shown in Figure 2.5 and also in Figure 2.1.

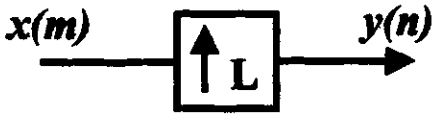


Figure 2.5 Upsampler.

In general, shaping can be described by the convolution summation:

$$\begin{cases} u_I(n) = \sum_{k=0}^{N-1} h_{TX}(k) u_{IT}(n-k) \\ u_Q(n) = \sum_{k=0}^{N-1} h_{TX}(k) u_{QT}(n-k), \end{cases} \tag{2.7}$$

where $h_{TX}(k)$ is the impulse response of the shaping filter that must be low-pass to limit the spectrum of the symbol. While any type of low-pass filter can be used to limit the bandwidth, most types will cause intersymbol interference (ISI). ISI degrades the performance of the receiver due to the symbol transmitted at mT. If appropriate filters are used in the transmitter and receiver there will be no ISI.

The impulse response of the ideal low-pass filter is normally expressed as sinX/X, i.e.:

$$h(t) = \frac{\sin(\pi f_s t)}{\pi f_s t}, \tag{2.8}$$

where f_s is the sampling frequency. The main advantage of the ideal low-pass filter is that its impulse response is zero at all mT except the zero one, i.e.

$$h(mT) = \begin{cases} 1, & mT = 0 \\ 0, & mT = \pm 1, \pm 2, \pm 3 \ldots \end{cases}, \tag{2.9}$$

where T is the time between symbol transmissions (symbol spacing). From Equation (2.9) it can be concluded that no ISI will be introduced by the pulse shaping filter [8].

An ideal low-pass pulse shaping filter has a flat passband limited to $|f| \leq f_{cut\text{-}off}$, where $f_{cut\text{-}off}$ is the cut-off frequency, expressed in hertz (Hz). However, the impulse response of the ideal low-pass shaping filter is non-causal, which is impossible to implement.

As discussed in the introduction, the digital communication systems are standardized. According to the J.83 standard, the practical pulse shape can be produced using a filter with a raised cosine (RC) frequency response. Contrary to the ideal filter, the energy in the response of the RC filter is more centered, i.e. its second central moment is lower than the second central moment in the response of the ideal filter. Thus, the truncation of the impulse response of the RC filter removes less energy from the response. Such filters use a shorter finite impulse response and therefore are much easier to implement than the ideal filter with truncated finite impulse response [8]. The frequency response $H_{RC}(j\omega)$ of the RC filter is real and even, while the transition band has a cosine shape [4, 8, 9]. The frequency response of the RC filter is given by:

$$H_{RC}(j\omega)=\begin{cases}1 & for\ \dfrac{|\omega|}{\omega_{cut-off}}\le 1-r\\ \dfrac{1}{2}\left(1+\cos\left[\dfrac{\pi}{2r}\left(\dfrac{\omega}{\omega_{cut-off}}-(1-r)\right)\right]\right) & for\ 1-r\le\dfrac{|\omega|}{\omega_{cut-off}}\le 1+r\\ 0 & for\ \dfrac{|\omega|}{\omega_{cut-off}}\ge 1+r,\end{cases} \quad (2.10)$$

where $\omega_{cut-off}$ is the cut-off frequency expressed in radians per second and r is a parameter called the roll-off factor. The parameter r, restricted to the interval $0 < r \le 1$, defines the excess bandwidth of the designed RC filter.

The impulse and frequency response of the RC filter with different roll-off factor values are illustrated in Figure 2.6. When r is equal to zero the response is that of the ideal low-pass filter. The pulse shape produced by the RC filter retains the characteristics given in Equation (2.9).

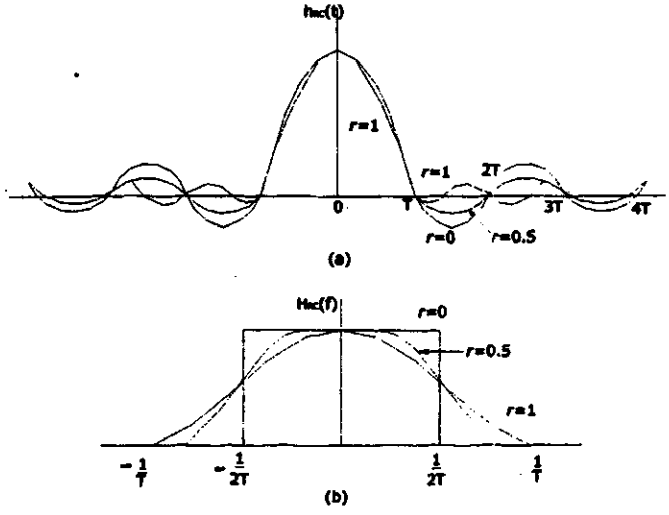


Figure 2.6 (a) RC filter impulse response, (b) Magnitude frequency response with different roll-off factor, r, values.

The 6-dB bandwidth of the baseband signal is equal to $1/2T=R_S/2$, where T is the symbol spacing, while R_S is the symbol rate, defined in symbols per second. The baseband bandwidth is given by:

$$B = \frac{1+r}{2} R_S \tag{2.11}$$

The absolute RF (or IF) signal bandwidth, which contains all the signal power, is twice the baseband bandwidth, i.e.:

$$B_{RF} = (1+r) R_S, \tag{2.12}$$

This relationship can also be used to define the maximum supported symbol rate:

$$R_S = \frac{2B}{1+r} = \frac{B_{RF}}{1+r} \tag{2.13}$$

The RF channel of the cable system is generally considered to be an additive white Gaussian noise (AWGN) channel. In order to achieve the optimal receiver from an AWGN channel perspective, it is common to evenly split the overall raised cosine pulse shaping between the transmitter shaping filter and the receiver matched filter. In other words, a pair of identical filters at the receiver and the transmitter sides is designed in such a way that their cascaded response is the RC filter.

The square root of the non-negative RC filter frequency response is the response needed for both pulse shaping and matched filters. A filter with this response, defined in Equation (2.14), is called a square root raised cosine (SRRC) filter.

$$H_{SRRC}(j\omega)=\begin{cases}1 & for\ \dfrac{|\omega|}{\omega_{cut-off}}\le 1-r\\ \cos\left[\dfrac{\pi}{4r}\left(\dfrac{\omega}{\omega_{cut-off}}-(1-r)\right)\right] & for\ 1-r\le\dfrac{|\omega|}{\omega_{cut-off}}\le 1+r\\ 0 & for\ \dfrac{|\omega|}{\omega_{cut-off}}\ge 1+r,\end{cases} \qquad (2.14)$$

where r is the roll-off factor and $\omega_{cut-off}$ is the cut-off frequency, expressed in radians per second.

In Figure 2.1 the output of the pair of shaping filters, $u_I(n)$ and $u_Q(n)$, is sent to the digital frequency upconverter in order to modulate the orthogonal carriers. As shown in Figure 2.1 the orthogonal carriers can be generated by a numerically controlled oscillator (NCO), which is covered in Chapter 4. The produced discrete intermediate frequency (IF) QAM signal can be expressed as:

$$s(n)=u_I(n)\cos\left(\frac{2\pi f_c}{f_s}n\right)-u_Q(n)\sin\left(\frac{2\pi f_c}{f_s}n\right), \qquad (2.15)$$

where f_s is the sampling frequency and f_c is the carrier frequency.

The discrete QAM signal $s(n)$ is then sent to the digital-to-analog (D/A) converter and can be represented as an analog signal in Equations (2.1) and (2.2).

2.2 Demodulation

At the receiver side the transmitted information is recovered during the demodulation process. The received IF (or RF) discrete QAM signal comes from the analog-to-digital (A/D) converter and can be represented as:

$$v(n) = v_I(n)\cos(\omega_c n + \varphi_m) - v_Q(n)\sin(\omega_c n + \varphi_m) + n_{AWGN}(n), \qquad (2.16)$$

where $n_{AWGN}(n)$ is the zero-mean, additive, white Gaussian noise and φ_m is the carrier offset that will be covered in Chapter 5.

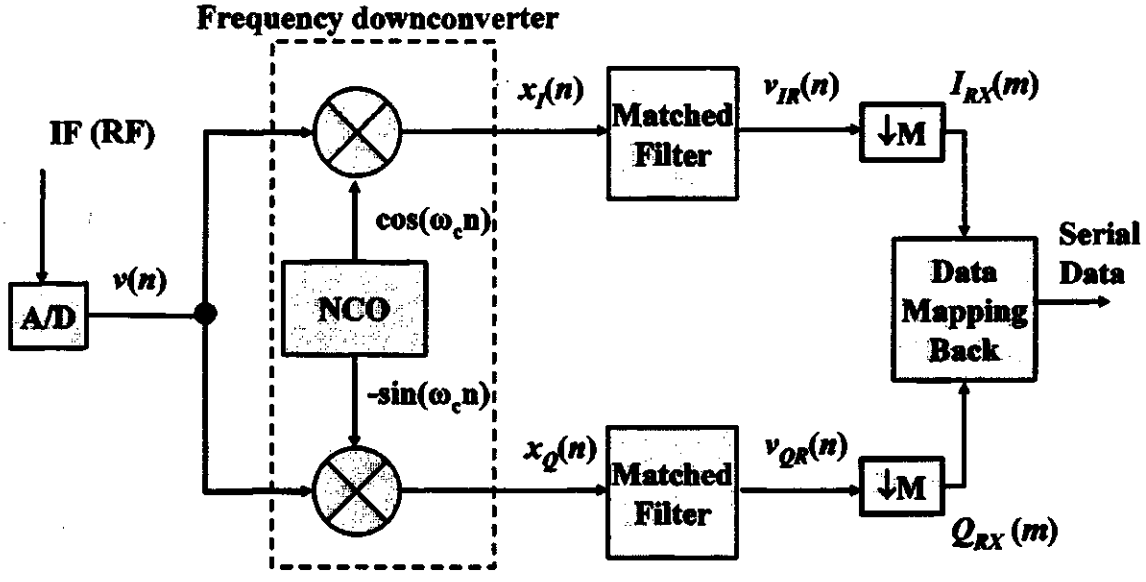


Figure 2.7 Schematic diagram of QAM demodulator.

In Figure 2.7 the IF (or RF) signal is downconverted to the baseband using complex multiplication by the orthogonal sinusoids. Similar to the transmitter, the sinusoids with frequency ω_c can be generated by the NCO. The spectrum of the IF QAM signal $v(n)$, centered at $\omega_c = \frac{\pi}{2}$, is shown in Figure 2.8.

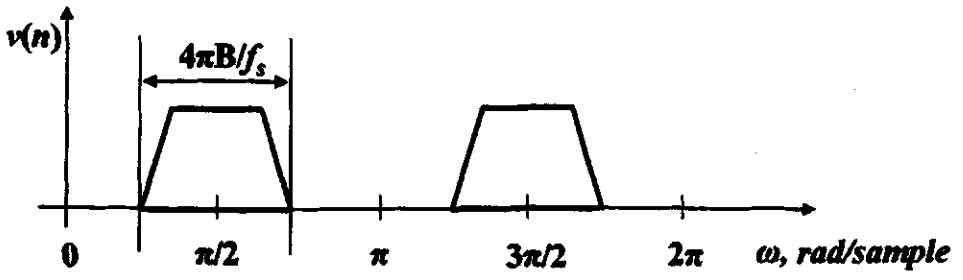


Figure 2.8 Spectrum of bandpass IF signal $v(n)$.

Assuming no noise, the demodulation can be represented using some simple trigonometric derivations:

$$\begin{aligned} x_I(n) &= v(n)\cos\omega_c n \\ &= \frac{1}{2}v_I(n)\cos\varphi_m - \frac{1}{2}v_Q(n)\sin\varphi_m \\ &+ \frac{1}{2}v_I(n)\cos(2\omega_c n+\varphi_m) - \frac{1}{2}v_Q(n)\sin(2\omega_c n+\varphi_m) \end{aligned} \tag{2.17}$$

and similarly,

$$\begin{aligned} x_Q(n) &= -v(n)\sin\omega_c n \\ &= \frac{1}{2}v_I(n)\sin\varphi_m + \frac{1}{2}v_Q(n)\cos\varphi_m \\ &- \frac{1}{2}v_I(n)\sin(2\omega_c n+\varphi_m) - \frac{1}{2}v_Q(n)\cos(2\omega_c n+\varphi_m). \end{aligned} \tag{2.18}$$

According to Equations (2.17) and (2.18) the output of the frequency downconverter consists of the low-pass and highpass components as shown in Figure 2.9.

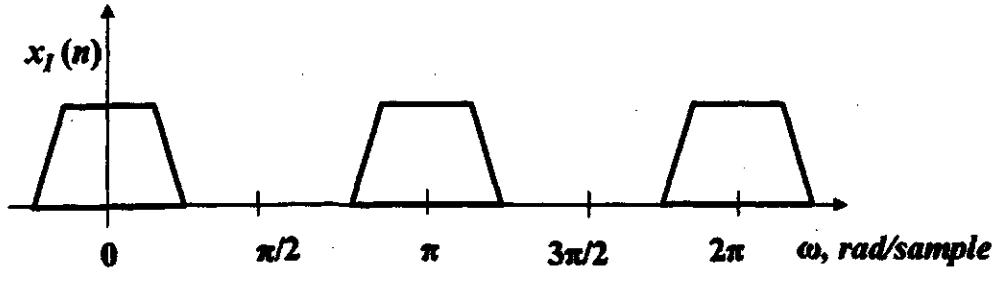


Figure 2.9 Spectrum of signal $x_I(n)$.

A pair of low-pass, matched, SRRC filters remove the high frequency components. Thus, the output of the SRRC filter, assuming a gain of 2, can be expressed as:

$$\begin{aligned} v_{IR}(n) &= v_I(n)\cos\varphi_m - v_Q(n)\sin\varphi_m \\ v_{QR}(n) &= v_I(n)\sin\varphi_m + v_Q(n)\cos\varphi_m, \end{aligned} \tag{2.19}$$

In order to map the received inphase and quadrature components back to the serial data stream the output of the matched filters is downsampled to the symbol rate by taking only the Mth value of the output as shown in Figure 2.10 [10].

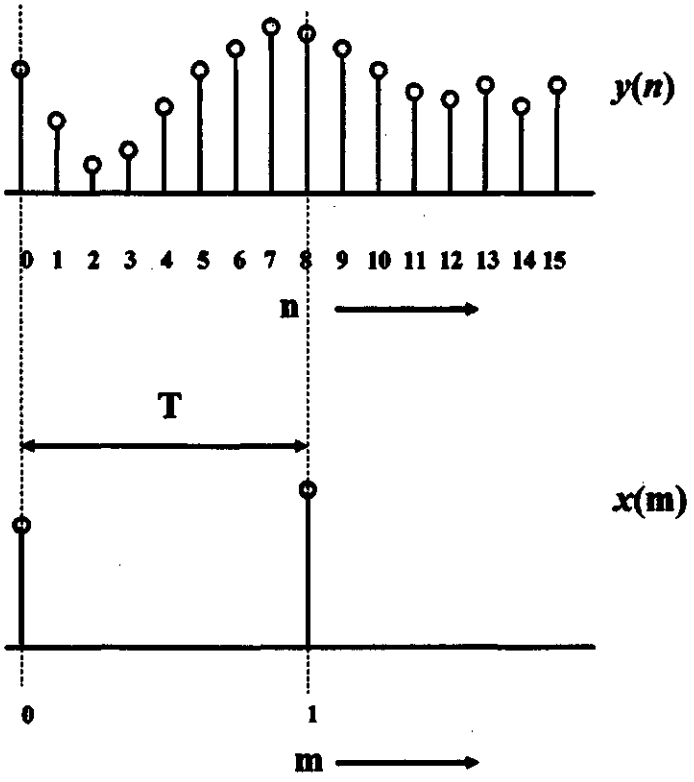


Figure 2.10 Downsampling process.

According to Figures 2.10 and 2.11 the downsampled signal $x(m)$ is expressed by:

$$x(m) = y(m \cdot M) \tag{2.20}$$

Figure 2.11 Downsampler.

Thus, ideally the downsampled values of the inphase and quadrature components, I_{RX} and Q_{RX}, of the received signal should be equal to the transmitted ones I_m and Q_m. In this case the received data sequence is equal to the transmitted data sequence. However, because of the carrier offset φ_m the received components can be represented as:

$$\begin{cases} I_{RX}(m) = I_m \cos\varphi_m - Q_m \sin\varphi_m \\ Q_{RX}(m) = I_m \sin\varphi_m + Q_m \cos\varphi_m \end{cases} \tag{2.21}$$

The final Equations (2.21) will be important in Chapters 4 and 5, once the CORDIC algorithm and the carrier recovery are described. While the effect of the carrier offset is discussed in Chapter 5, the challenges of the shaping (and matched) filter implementation are investigated in the following chapter.

3 SHAPING AND MATCHED FILTER IMPLEMENTATION

One of the main tasks of this research was the design of cable modem components suitable for implementation in a Field Programmable Gate Array (FPGA). In this chapter the implementation of the square root raised cosine (SRRC) filters in the FPGA is discussed. The polyphase structure that reduces the complexity of the SRRC implementation is also presented in detail. The polyphase filter structure is compared to the transversal filter structure.

3.1 SRRC Filter Specifications

As discussed in the previous chapter, the data to be transmitted is first mapped to inphase and quadrature components. These I_m and Q_m components run at the symbol rate R_S, which is also expressed as *clk_symbol*. The inphase and quadrature components are upsampled by inserting L -*1* zeros, where L is equal to the number of samples per symbol. Finally, two upsampled streams are shaped using SRRC (square root raised cosine) shaping filters. Usually these SRRC filters are finite impulse response (FIR) type. The FIR filters are causal and non-recursive filters.

Given finite duration input values $y(n)$ the FIR filter will produce finite duration output values $u(n)$ [11] (Figure 3.1).

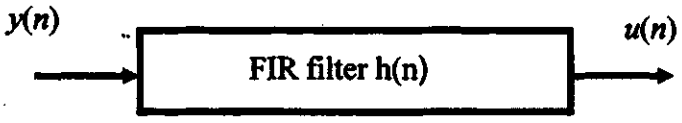


Figure 3.1 Digital FIR filter.

Since the impulse response of the filter is represented by its coefficients $h(0)$, $h(1)$, $h(2)$, ... $h(N-1)$, the Nth-order, FIR, filter transfer function can be expressed as:

$$H(z) = \sum_{k=0}^{N-1} h(k)z^{-k} = h(0) + h(1)z^{-1} + h(2)z^{-2} + \ldots + h(N-1)z^{-(N-1)}. \qquad (3.1)$$

The output of a FIR filter is given by the convolution of the input with the impulse response $h(k)$, $k = 0, 1, \ldots N-1$:

$$u(n) = h(k) * y(n). \qquad (3.2)$$

As mentioned in Chapter 1, cable modems are standardized. In order to achieve the optimal design of the SRRC filter, the minimum requirements for the SRRC filter are given in the J.83 standard for 256-QAM modulation. The template of the band-limited SRRC filter is illustrated in Figure 3.2. The value of the in-band ripple r_m in the pass-band as well as at the Nyquist frequency f_N ($f_N = 1/2T = R_S/2$) should be lower than 0.4 dB. The out-of-band rejection should be greater than 43 dB. Also the filter has to be phase linear [4].

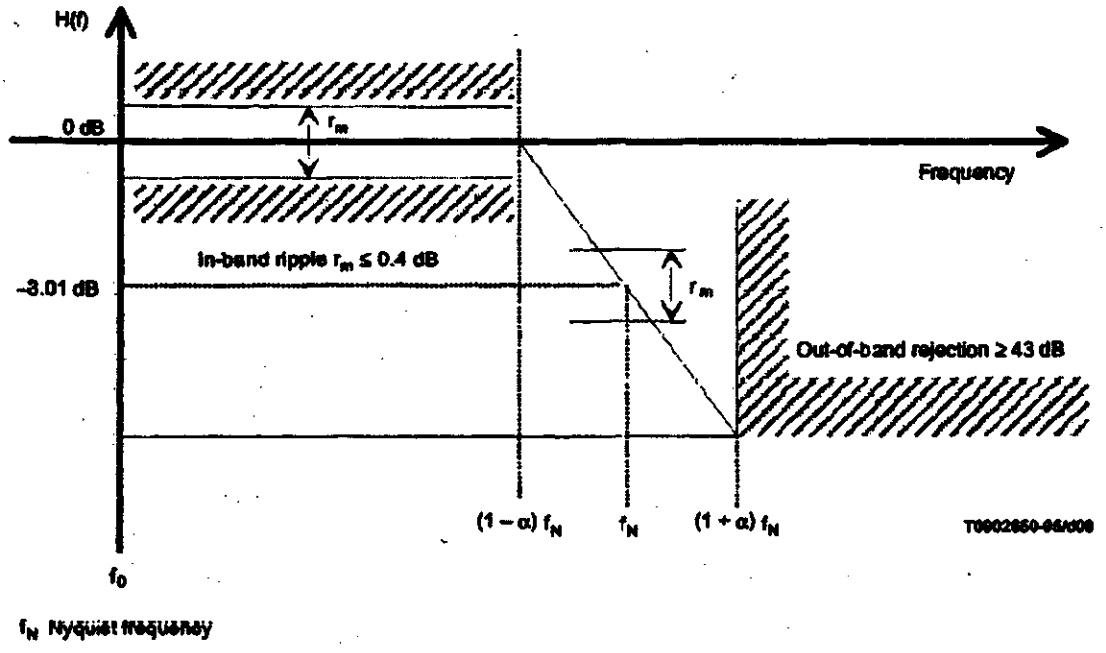


Figure 3.2 SRRC filter amplitude characteristics (ITU J.83, page 14).

In order to make the filter practical for implementation, the impulse response of the SRRC filter is truncated. Therefore, it is important to determine the length of the filter that would satisfy the template given in the J.83 standard. At the same time it is necessary to determine how many bits per coefficient, i.e. what word length of the coefficients, should be used. The farther the sidelobes are from the main lobe the closer to zero the heights of the sidelobes are. Therefore, the maximum length of the filter is determined by the point at which the sidelobes can no longer be accurately represented by the word length of the coefficient. This maximum length can also be specified in symbol spacings.

Using a Matlab simulation, it was empirically determined that in order to make the SRRC filter design satisfy the J.83 standard specifications, the truncated impulse

response of the SRRC filter should have 16 symbol spacings (the time between symbol transmissions) with 8 samples per symbol and a coefficient word length of 10 bits. In this case the filter has 128 coefficients. The main trade-off is between the sufficient sidelobe attenuation and the complexity of the multipliers. The filter specifications are given in Table 3.1.

Table 3.1 SRRC filter specifications.

Parameter	Value
Frequency response	SRRC (square root raised cosine) filter
Type	Linear phase FIR (finite impulse response)
Roll-off factor	0.35
Number of samples per symbol	8
Number of coefficients	128
Length of impulse response in symbol spacings	16
Number of bits per coefficient	10

In this project the filter coefficients are 10-bit wide integer numbers that were determined in the following way. First, the coefficients of the filter were produced using a Matlab "rcosfir" function. Unfortunately, these coefficients are floating point numbers. In order to make it suitable for the FPGA implementation the floating-point coefficients were then scaled and rounded to the nearest integer. The normalized mean square difference (msd) between the floating-point impulse response, $h_{floating}(n)$, and the scaled

integer response, $h_{integer}(n)$, depends on how many bits are used for the filter coefficients, which is determined by the scaling factor.

$$msd = \frac{\sum_{n=-\infty}^{\infty} \left(h_{floating}(n) - h_{integer}(n)\right)^2}{\sum_{n=-\infty}^{\infty} \left(h_{floating}(n)\right)^2} \tag{3.3}$$

If the word length of the coefficients is 10 bits, then the mean square difference is

$$msd = 8.7888 \cdot 10^{-6} \tag{3.4}$$

The truncated impulse response for a SRRC filter is shown in Figure 3.3.

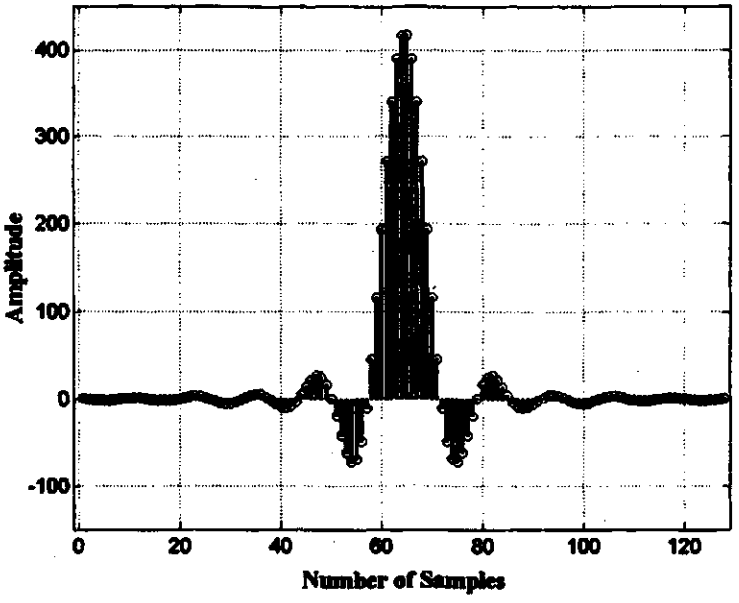


Figure 3.3 Impulse response of SRRC filter.

The magnitude response of this particular SRRC filter and the corresponding RC filter is shown in Figure 3.4. A vertical line helps to verify the ripple at the 3-dB point for the SRRC filter or at the corresponding 6-dB point for the RC filter. The ripple r_m (see also Figure 3.2) at that point, as well as the ripple in the passband, shown in Figure 3.4 (b), is less than 0.4 dB, as specified in the J.83 standard. The first sidelobe's attenuation is

close to the required – 43-dB attenuation, whereas the attenuation of the other sidelobes is well beyond the – 43-dB level.

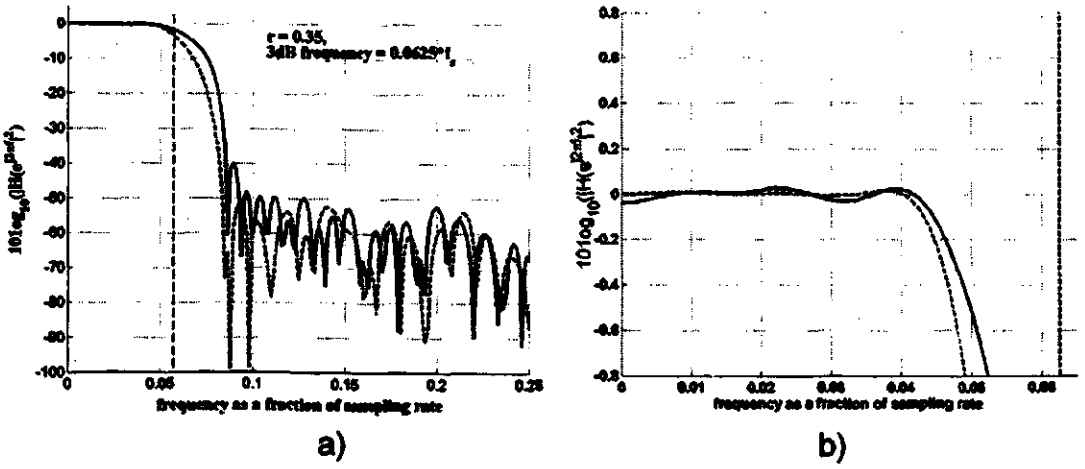


Figure 3.4 a) Magnitude response of RC (dashed line) and SRRC (solid line) filters, b) ripple in the passband < 0.4dB (logarithmic scale).

The magnitude responses of RC and SRRC filters on a linear scale are plotted in Figure 3.5.

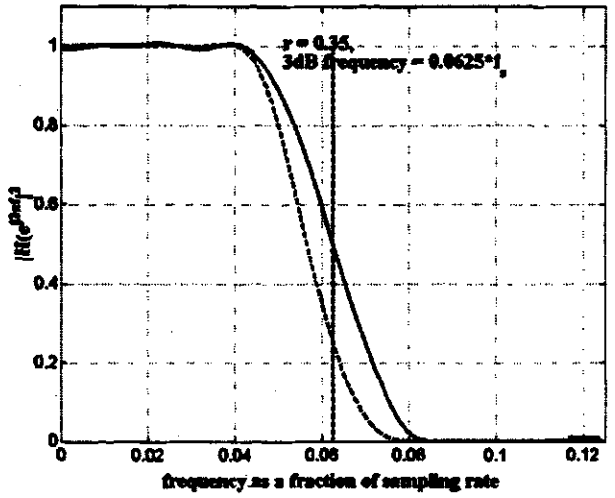


Figure 3.5 Magnitude response of RC (dashed line) and SRRC (solid line) filters (linear scale).

The phase responses of both the RC and the SRRC filters are linear in the passband as shown in Figure 3.6, where the passband is from 0 to 0.0625 cycles/ sample.

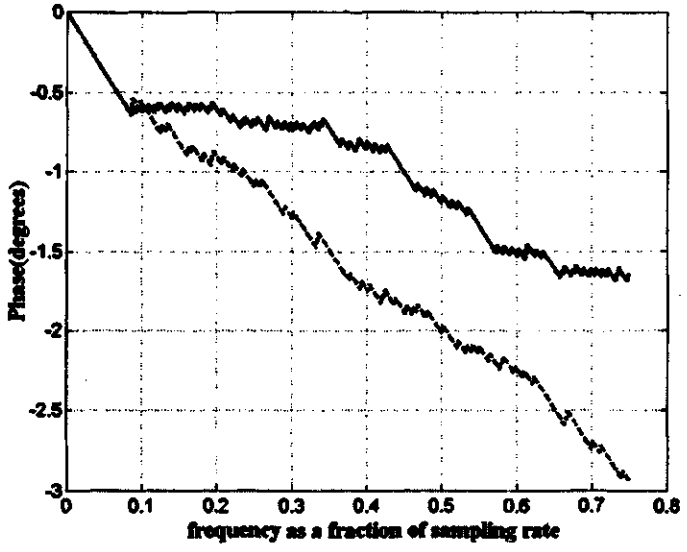


Figure 3.6 Phase response of RC (dashed) and SRRC (solid) filters.

In Figures 3.4 and 3.5 it is shown that the 3-dB cut-off frequency is equal to $0.0625 \cdot f_s$, where f_s=16.47 MHz is the sampling frequency, used in this project. This means that the 3-dB absolute bandwidth is 2.0588 MHz, which is equal to the symbol rate R_S=2.0588 Msps (Megasymbols per second). As previously shown in Chapter 2, the absolute bandwidth is $B_{RF} = (1+r)\, R_S = 2.7$ MHz, due to the chosen roll-off factor $r = 0.35$.

3.2 Implementation of Linear Phase Filter in Hardware

In this project the digital design implementation in a FPGA (Field Programmable Gate Array) chip was considered. The implementation should be optimal for the available FPGA chip resources, i.e. the digital design must use as few available logical elements (LE) of the FPGA chip as possible.

There are several different types of the FIR filter structure. It is important to select the most suitable type for the FPGA implementation of the particular design. A common structure for the FIR filter, called the *direct form*, is illustrated in Figure 3.7. Here z^{-1} represents registers (delays) that are clocked simultaneously at the sample rate clock - $clks = \frac{1}{f_s}$.

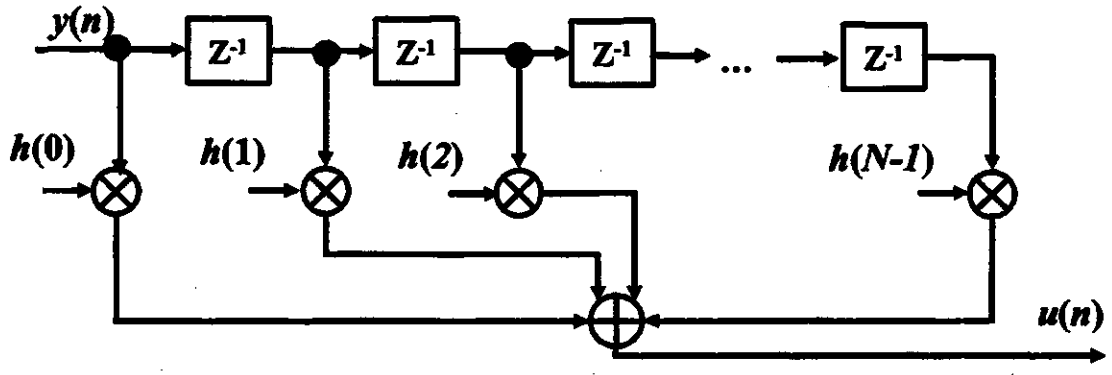


Figure 3.7 Direct form of FIR filter.

The direct form implementation requires a multiplier per each coefficient. Multipliers are expensive, so filters with a long impulse response are very costly. This is a significant disadvantage of the direct FIR filter structure.

The RC and SRRC filters belong to the class of linear phase FIR filters. Thus, the impulse response of the RC or the SRRC filter is symmetric about its midpoint, i.e.

$$h(n) = h(N-1-n) \tag{3.5}$$

This property can be applied in order to reduce the number of expensive multipliers by a factor of 2. Therefore, in this so-called *transversal structure* only $\frac{N}{2}$ multipliers are needed instead of N multipliers, as required in the direct structure. This symmetry

property allows the taps with the same weights to be added before being multiplied. The transversal structure of the symmetrical FIR filter with an even number of coefficients is shown in Figure 3.8.

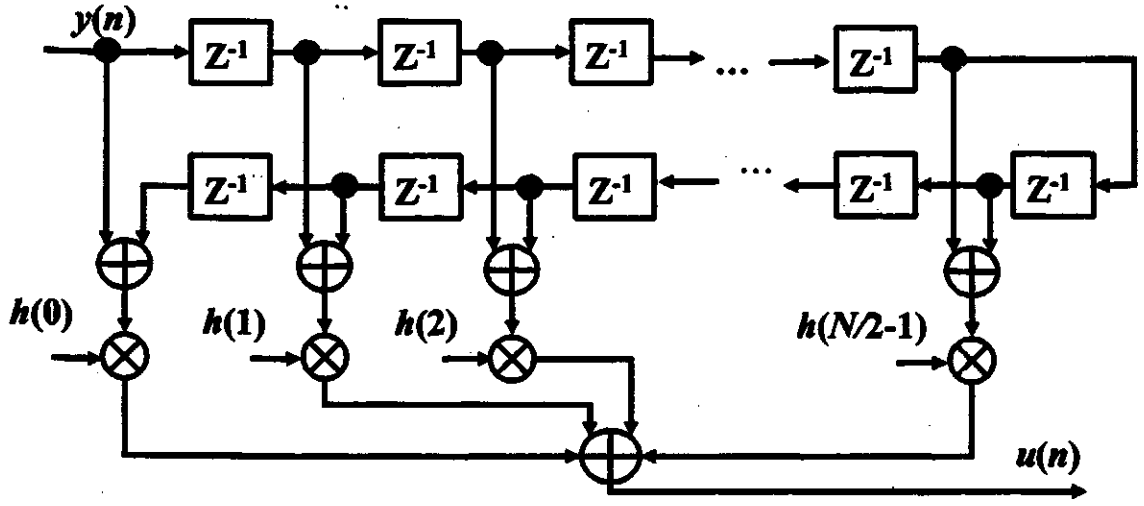


Figure 3.8 Transversal structure of symmetrical FIR filter.

Although the typical transversal structure of the FIR filter is quite an improvement in the filter implementation, still it is not the best solution in the case of a very long filter. The main disadvantage of the transversal structure is that it requires not only N/2 multipliers but also N/2 adders. These blocks that perform the addition and multiplication operations occupy most of the FPGA chip area. Therefore, attention should be paid to how many multipliers and adders are necessary for the particular task and what kind of multipliers and adders they should be.

Generally, there are two types of multipliers. In the first type, one of the inputs is constant while the other input is variable. In the other type, both inputs are variable [12]. Different types of adders and multipliers were tested using Altera Quartus II software

simulations. The results of these tests and the corresponding numbers of logical elements (LEs) that are required for their implementation are listed in Table 3.2.

Table 3.2 Adder and multiplier area required.

Function	Logic elements (LE) required
8 bit adder	13
10 bit adder	17
16 bit adder	25
10 x 10 variable coefficient multiplier	162
16 x 16 variable coefficient multiplier	226
10 x 10 constant coefficient multiplier	50
16 x 16 constant coefficient multiplier	84

From Table 3.2 it is evident that a multiplier with both inputs variable would occupy approximately 2-3 times more FPGA chip area than the multiplier with one variable input and one constant input. The transversal structure of the SRRC filter with constant coefficients appears optimal for the FPGA implementation. Unfortunately, the number of multipliers and adders required for this particular SRRC filter with 128 coefficients is relatively big. In the transversal structure of the SRRC filter there are N / 2 =64 multipliers and $N / 2 = 64$ adders before the final summation of the entire convolution process. The SRRC filter with the transversal structure was tested using Altera Quartus II software. According to the compilation report approximately 5710 LEs (logical elements) or about 14 % of the FPGA chip were required.

Generally, there are 2 SRRC filters in the modulator, one for the inphase component and one for the quadrature component, and correspondingly, 2 matched filters in the demodulator. Therefore, if the direct form of the SRRC filters were applied, the total number of multipliers in the modem would reach 512. If the property of symmetry and the transversal structure is used, 256 multipliers and 256 adders are still required. These huge numbers of expensive multipliers and adders show that the direct and transversal filter structures are not an effective use of the FPGA design strategy for this particular design. As an effective alternative to the conventional direct or transversal structure, a polyphase filter structure is presented in the following sections.

3.2.1 Multirate Filters and Polyphase Structure

3.2.1.1 Shaping Filters

As discussed in Chapter 2 the SRRC filter is used for shaping. The shaping consists of first stuffing zeros between the data samples (upsampling). In this case the SRRC filter has to work at a sampling rate that is L times higher than the symbol rate. The common transversal structure of the SRRC filter, discussed in the previous section, reduces the number of multipliers to $N/2$, where N is the number of taps. The shaping filter has to perform $N/2$ multiplications per symbol, while the non-zero upsampled data sequence coming to the filter was only N/L as shown in Figure 2.4. This disadvantage can be removed by using a *polyphase filter structure*, instead of a transversal one [9]. The polyphase structure reduces the number of multipliers to N/L and allows reusing the same multipliers for a different set of coefficients.

The shaping process should be examined thoroughly at this point. The 128-tap SRRC filter, that is shaping the upsampled data by a factor of $L = 8$, is shown in Table 3.3. (Only 12 coefficients are shown for simplicity.) At every sample rate clock "clks" rising edge the upsampled data $y(n)$ (where $y(n) = x(n/L)$, when $n = mL$, and 0 otherwise) moves from left to right along the filter delay line and the new output $u(n)$ is calculated by the convolution procedure. The corresponding filter coefficients are shown in the first row in Table 3.3.

Table 3.3 Shaping process.

Sampling clock tick	SRRC filter impulse response												
	h(0)	h(1)	h(2)	h(3)	h(4)	h(5)	h(6)	h(7)	h(8)	h(9)	h(10)	h(11)	...
First	x(1)	0	0	0	0	0	0	0	x(0)	0	0	0	...
Second	0	x(1)	0	0	0	0	0	0	0	x(0)	0	0	...
Third	0	0	x(1)	0	0	0	0	0	0	0	x(0)	0	...
Fourth	0	0	0	x(1)	0	0	0	0	0	0	0	x(0)	...
Fifth	0	0	0	0	x(1)	0	0	0	0	0	0	0	...
Sixth	0	0	0	0	0	x(1)	0	0	0	0	0	0	...
Seventh	0	0	0	0	0	0	x(1)	0	0	0	0	0	...
Eighth	0	0	0	0	0	0	0	x(1)	0	0	0	0	...

In the direct filter structure each coefficient is consecutively multiplied by all the corresponding input values, including those cases when the incoming data is equal to zero. In $L - 1$ out of L input samples the multiplication result is known to be zero and consequently has no effect on the filter output sequence. Thus, there is no need to perform the redundant multiplications. Since only Lth data is non-zero, therefore, the output of the filter could be represented by the following convolutions:

$$
\begin{aligned}
u(0) &= x(1)\cdot h(0) + x(0)\cdot h(8) + x(-1)\cdot h(16) \div \ldots + x(-14)\cdot h(120) \\
u(1) &= x(1)\cdot h(1) + x(0)\cdot h(9) + x(-1)\cdot h(17) + \ldots + x(-14)\cdot h(121) \\
&\ldots \\
u(7) &= x(1)\cdot h(7) + x(0)\cdot h(15) + x(-1)\cdot h(23) + \ldots + x(-14)\cdot h(127)
\end{aligned}
\tag{3.6}
$$

The entire convolution $h(n) * y(n)$ at the higher sampling rate can be substituted with the independent convolutions at the lower rate [9]. For example, in order to calculate $u(0), u(8), u(16) \ldots u(nL)$ the input sequence y(n) must be convolved with a subfilter h_0(n) impulse response, where h_0(n) = h(0), h(8), h(16), h(24), h(32), h(40), h(48), h(56), h(64), h(72), h(80), h(88), h(96), h(104), h(112), h(120). Similarly, in order to calculate $u(1), u(9), u(17) \ldots u(nL+1)$ the input sequence y(n) must be convolved with a subfilter h_1(n) impulse response, where h_1(n) = h(1), h(9), h(17), h(25), h(33), h(41), h(49), h(57), h(65), h(73), h(81), h(89), h(97), h(105), h(113), h(121) and so forth. Therefore, in the shaping filter there are L branches (subfilters) that are given by:

$$
h_\lambda(n) = h(nL + \lambda), \tag{3.7}
$$

where $\lambda = [0 : L-1]$ is the number of the subfilter.

At every sampling rate clock "clks" rising edge only the particular subfilter, e.g. $h_0(n)$, should be used because all the rest of the subfilters will not have any effect on the output of the filter. At the second clks rising edge only the subfilter $h_1(n)$ should be applied and consequently at the $(L-1)$th sampling rate clock 'clks' rising edge only the subfilter $h_{L-1}(n)$ gives a non-zero output. On the Lth sampling clock tick the $h_0(n)$ must be used again and so forth.

Therefore, there is no need to use all N (or $N/2$) multipliers at every sampling clock tick. Instead, only N/L multipliers with periodically changing coefficients are necessary to achieve the same result. The proposed block diagram for the shaping filter is given in Figure 3.9. This type of multirate filter has a polyphase structure due to its coefficient separation to L subfilters. The L subfilters are rearranged in such a way that their coefficients are stored in separate LUTs (Look-Up Tables), also called ROMs (Read Only Memory), and are periodically reapplied to the multipliers as illustrated in Figure 3.9. Here N/L=128/8=16 multipliers are periodically reused by the time-varying coefficients in order to calculate the output of the shaping filter.

At the positive edge of the sampling rate clock "clks" the coefficients are periodically reapplied to the data, coming in at the lower symbol clock, called "clk_symbol". The coefficients, stored in the ROMs, are organized in order that every first coefficient of each ROM belongs to the $h_0(n)$ subfilter and all these coefficients are applied to the multipliers simultaneously. At the next "clks" rising edge all the coefficients of the subfilter $h_1(n)$ are applied and so forth for the remaining subfilters. All these iterations are performed while the current data is present. On the next "clk_symbol" rising edge the new data arrives and the cycle repeats starting with $h_0(n)$.

Since the same data is used for multiplications by coefficients from different subfilters there is no need to insert zeros. Therefore, the upsampling is eliminated from the entire shaping process. All the multiplications are iterated at L times higher rate while the input data, coming in to the filter at a lower rate, is present.

Therefore, the computation of the output in Figure 3.9 is performed exactly as it was shown in Equation (3.6) eliminating unnecessary operations (i.e. multiplications by zero). The truncation operation at the output of the filter results in taking the 12 most significant bits (MSBs). In this project a 12-bit wide DA (digital-to-analog) converter was considered, which limited the number of bits left after the truncation.

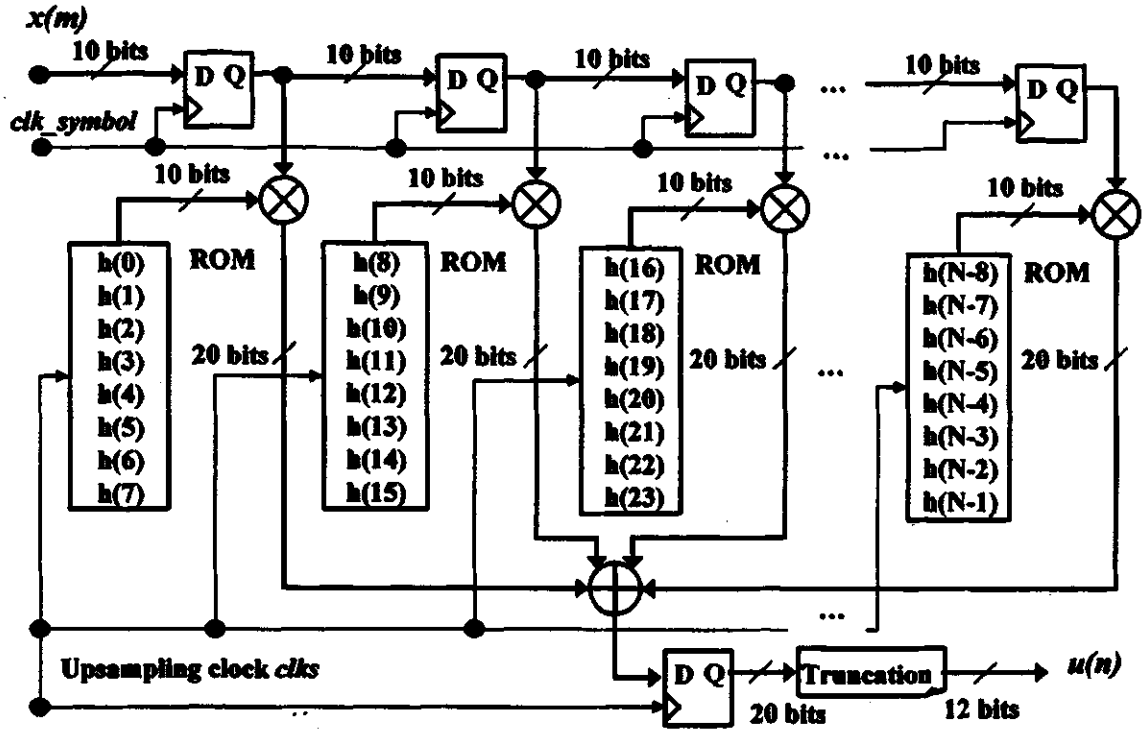


Figure 3.9 Logic block diagram of polyphase shaping filter.

3.2.1.2 Matched Filters

According to the discussion in Chapter 2, in order to obtain optimal detection the RC response is split between the transmitter shaping filter and receiver matched filter. Once the received QAM signal is downconverted to the baseband, it is filtered by a pair of matched SRRC filters – a filter for both the inphase and quadrature components. The

output of the matched filter is then downsampled to the symbol rate as shown in Figure 2.7 [7, 10]. The downsampling by a factor of M is the process of taking only every Mth value of the discrete signal, which was shown in Equation (2.20) [9].

When $n \neq m \cdot M$ (where M=8 is the downsampling rate), the calculation of all values of y(n) is unnecessary due to the following downsampling by M. In other words, only the multiplications of $h(k)$ by $v(-Mn-k)$ contribute to the downsampled outputs. There is no need to perform the remaining computations because they are ignored by the following downsampling process. Therefore, the entire convolution process $v(n) * h(n)$ can be considered as a combination of M independent convolutions that are added at the end to give the output sequence $x(m)$. Table 3.4 illustrates the independent convolutions processing:

Table 3.4 Polyphase matched filter with downsampling process.

Sampling clock number	Filter Coefficients	Input Samples	Independent convolutions	Output sequence
0	h(Mn+7)	v(-Mn-7)	$x_0(m)$	$x(m)$
1	h(Mn+6)	v(-Mn-6)	$x_1(m)$	
2	h(Mn+5)	v(-Mn-5)	$x_2(m)$	
3	h(Mn+4)	v(-Mn-4)	$x_3(m)$	
4	h(Mn+3)	v(-Mn-3)	$x_4(m)$	
5	h(Mn+2)	v(-Mn-2)	$x_5(m)$	
6	h(Mn+1)	v(-Mn-1)	$x_6(m)$	
7	h(Mn)	v(-Mn)	$x_{M-1}(m)$	

In this project the matched filter is designed as a polyphase filter and implemented in a similar fashion to the shaping filter, discussed in the previous section. The design shown in Figure 3.10, does not perform any unnecessary calculations of $y(n)$, which would be removed by the following downsampling. Similar to the shaping process discussed in the previous section, at the first sampling rate clock "clks" rising edge $x_0(m)$ is calculated by convolving $h(Mn+7)$ coefficients with $v(-Mn-7)$ samples, where M =8 is the downsampling rate. At the second "clks" rising edge $x_1(m)$ is produced by convolving $h(M+6)$ coefficients with $v(-Mn-6)$ samples. Consequently, at the $(M-1)$th "clks" rising edge $x_{M-1}(m)$ is calculated by convolving the $h(Mn)$ coefficients with the $v(-Mn)$ samples. The produced results of the independent convolutions are accumulated and then finally added at the symbol clock rate "clk_symbol".

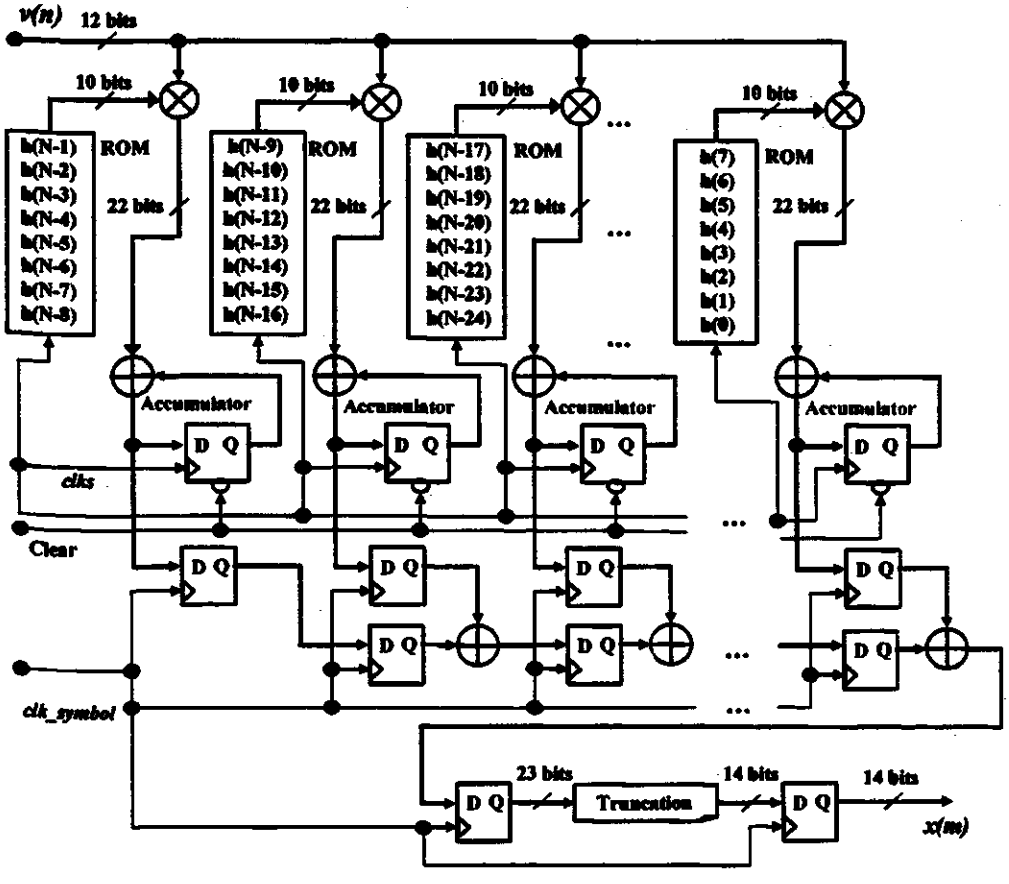


Figure 3.10 Logic block diagram of polyphase matched filter.

Since the polyphase matched filter works on two different clocks (i.e. "clks" and "clk_symbol"), it is also called a multirate filter. The output of a pair of matched filters is a downsampled stream of inphase and quadrature components, running at the symbol clock. Therefore, there is no need for downsampling.

3.3 Comparison of Polyphase and Transversal Structure for Shaping Filter and Matched Filter Design

In conclusion, the following advantages and disadvantages of the polyphase structure are enunciated. The conclusions were made after several tests of different filter structures.

According to the Altera Quartus II compilation report, the FPGA shaping SRRC filter, with the polyphase structure and 16 variable multipliers, required 2645 logical elements (LEs) out of 38400 LEs available, which is about 6% of the FPGA chip area. The matched SRRC filter, with the polyphase structure and 16 variable multipliers occupied 11% (4432 LEs) of the available chip area. This difference could be explained by a more complex structure of the matched filter with supplementary adders required in the design. Nevertheless, the proposed architecture of the SRRC shaping and matched filters with the polyphase structure and variable multipliers required less FPGA chip resources than a transversal structure with 64 constant multipliers and 64 adders (see Table 3.5).

Table 3.5 FPGA resources for different filter structures.

Filter structure	Number of logical elements (LE)	FPGA chip area (%)
Transversal with constant coefficients	5710	14
Polyphase with variable coefficients for shaping filter	2645	6
Polyphase with variable coefficients for matched filter	4432	11
Total available FPGA chip resources	38400	100

Since only N/L multiplications (instead of N) are performed in order to calculate each output of the SRRC filter, the overall computational savings are:

$$\left(N-\frac{N}{L}\right)=\left(128-\frac{128}{8}\right)=112 \text{ out of } 128, \tag{3.8}$$

where N is the number of filter taps, while L (or M) is equal to the number of samples per symbol.

The derived computational savings function of the number of samples per symbol, assuming a constant value of taps N, is shown in Figure 3.11. Here, when $L = 2$ the computational savings are the same for both the transversal and the polyphase structure. As L increases, the computational savings of the polyphase structure grow, while the computational savings of the transversal structure remain the same (50%). Although, when L is increased significantly, a ~100% computational savings of the original cost is reached theoretically. However, in practice, as soon as L (where $L < N$) is more than 8, a longer filter must be used, i.e. N must be increased in order to get a sufficient sidelobe attenuation. Therefore, the computational savings of the polyphase filter structure approach a constant level of 87.5% rather than increasing (see Figure 3.11).

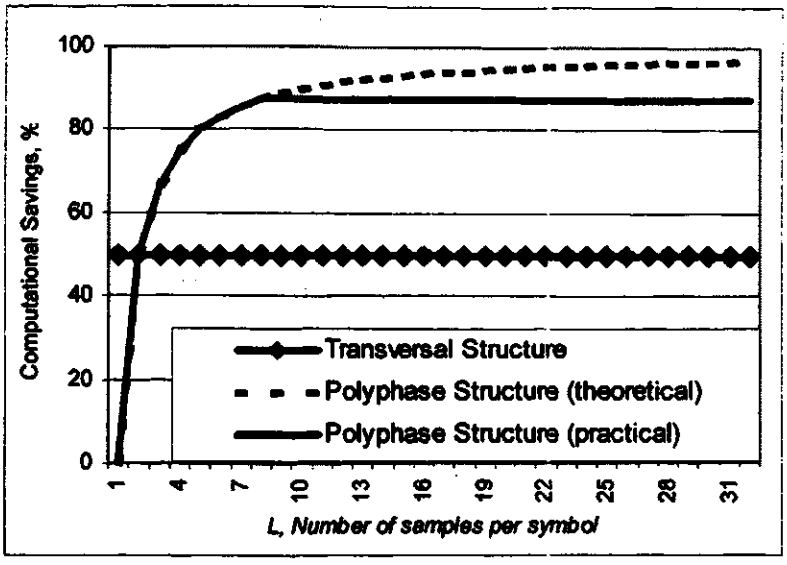


Figure 3.11 Computational savings for transversal (diamond line) and polyphase (solid or dashed line) filter structure.

Since $L-1$ values per input sample (padded with zeros) are not stored, then the memory is saved by a factor of [13]:

$$(L-1)/L=(8-1)/8=0.875=87.5\% \tag{3.9}$$

The computational and memory savings as well as the size and type of the multipliers and adders discussed above are important in determining which strategy to apply to the filter implementation. In some cases the transversal structure with constant coefficient multipliers is more effective. In other cases the polyphase structure with variable coefficient multipliers may be more suitable. Here, the main tradeoff is between the required number of multipliers and adders and the complexity of these computational blocks.

In conclusion, it can be predicted that the transversal structure would be more efficient for a small filter with a short impulse response and a relatively low sampling rate. However, if this is a long filter with a high sampling rate (more than ~ 4 samples per symbol), it can be predicted that the polyphase filter would be more efficient.

There is also another tradeoff between the area of the FPGA chip and the maximum supported sampling frequency. The higher the frequency, the more suitable the transversal structure (with constant coefficient multipliers) is.

4 FREQUENCY CONVERTER DESIGN USING NCO AND CORDIC

This chapter explores the efficiency of a digital frequency converter design. The digital complex frequency converters are considered instead of expensive and relatively imprecise analog frequency converters.

Due to the similar challenges discussed in Chapter 3, the digital design of the frequency converter should be optimized in order to make it suitable for the implementation in FPGAs. As a possible strategy for this task, the CORDIC algorithm was proposed and compared with an NCO (numerically controlled oscillator). Some techniques to reduce the implementation costs were also investigated.

4.1 Digital Frequency Converter

As discussed in previous chapters, the data is first shaped to ensure that after modulation the RF (or IF) 256-QAM signal does not leak into adjacent channels. During the modulation process the digital baseband 256-QAM signal is upconverted to occupy a particular channel according to cable system specifications. The appropriate channel can be considered as a "pipe", through which the signal is transmitted. Correspondingly, in the receiver, the 256-QAM signal is downconverted back to baseband during the demodulation process.

As discussed in Chapter 2, the frequency conversion is performed using a complex multiplication by sinusoids. A digital frequency converter is preferred because it is more precise and easier to implement and control than the expensive analogue frequency converter [14].

The common structure of the digital frequency converter is the digital Numerically Controlled Oscillator (NCO), which controls the phase and the frequency of the sinusoids, and a couple of multipliers that multiply the input signal by these sinusoids. The sinusoids are of the same frequency but with the phase difference equal to 90^0. Figure 4.1 shows the typical design of the NCO, which consists of a phase generator, a LUT (look-up table) and a pair of multipliers.

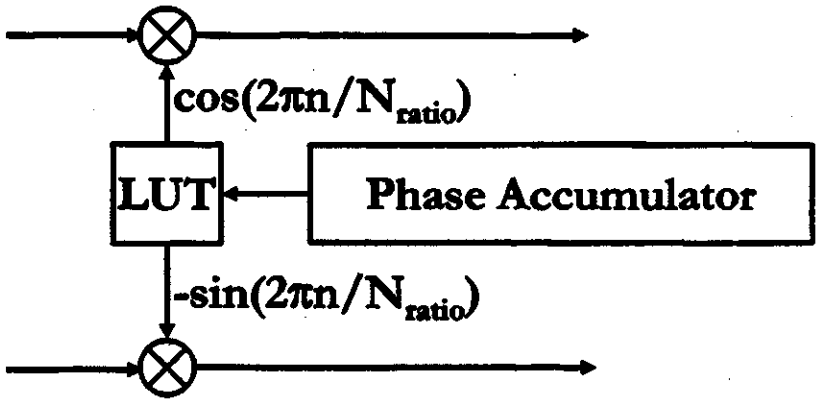


Figure 4.1 Numerically controlled oscillator (NCO).

The main part of a digital frequency converter is the phase generator (phase accumulator) illustrated in Figure 4.2:

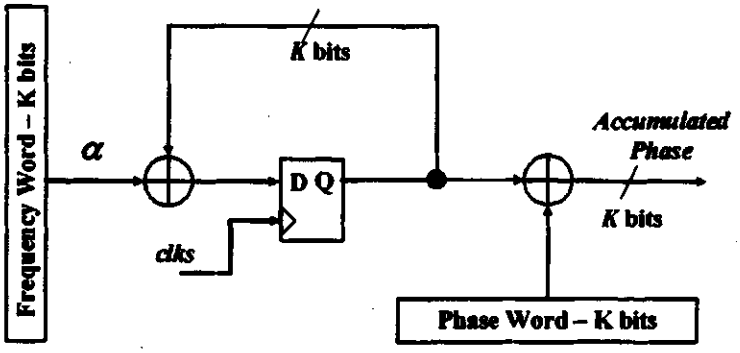


Figure 4.2 Phase accumulator.

The K-bit value that the accumulator holds is a fraction of a cycle, with all zeros corresponding to 0 radians and all ones corresponding to $(1-2^{-K})$ cycles which is $(1-2^{-K})2\pi$ radians. At every sampling rate clock "clks" a constant K-bit word, representing the frequency, is added to the K-bit accumulated value to produce the phase of the sinusoids. The phase is accumulated as a ramp function. The frequency depends on the step, α, of the ramp function. This process continues until rollover occurs and then starts again. The rollover occurs at 2π as shown in Figure 4.3. If the frequency word is determined by a constant value α, then the accumulated phase is expressed as $2\pi\alpha/2^K$. An additional constant K-bit phase word can be added for constant phase shift control.

The accumulated phase is then applied as an address for the LUT (look-up table) that stores the values of the sine and cosine functions as shown in Figure 4.1. The size of the LUT depends on how many bits represent the phase. If the accumulated phase is K bits wide then there are 2^K values of sinusoids stored in the LUT. Typically though, the number of bits going to the LUT is less than K, depending on what precision is required.

The produced orthogonal sinusoids are then multiplied by the input signal for up (or down) frequency conversion.

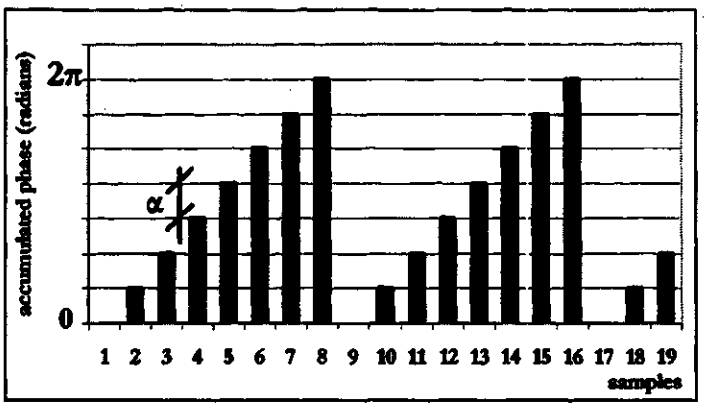


Figure 4.3 Accumulated phase as function of sample number.

As discussed in Chapter 3, two NCO multipliers with variable inputs along with a big LUT may require a significant amount of FPGA area. Therefore, the digital frequency converter design must also be optimized for a suitable implementation in FPGA.

4.2 Frequency Converter at Transmitter Side

As mentioned in Chapter 2 the modulation process is given by the following equation:

$$u_I(n)\cos\left(\frac{2\pi n}{N_{ratio}}\right) - u_Q(n)\sin\left(\frac{2\pi n}{N_{ratio}}\right), \tag{4.1}$$

where $u_I(n)$ and $u_Q(n)$ represent the inphase and quadrature components of the baseband 256-QAM signal, while $N_{ratio}=f_s/f_c$ is the sampling to carrier frequency ratio. If $N_{ratio}=4$, then the upconversion process can be simplified, because the sinusoids become periodic sequences of just 1, 0 and -1 as shown in Equation (4.2) [11].

$$\cos\frac{2\pi n}{4}=\begin{cases}(-1)^{\frac{n}{2}} & \text{for } n=0,2,4...\\ 0 & \text{for } n=1,3,5...\end{cases}=\ldots\ 1,\ 0,\ -1,\ 0,\ 1,\ 0,\ \ldots$$

$$-\sin\frac{2\pi n}{4}=\begin{cases}0 & \text{for } n=0,2,4...\\ (-1)^{\frac{n+1}{2}} & \text{for } n=1,3,5...\end{cases}=\ldots\ 0,\ -1,\ 0,\ 1,\ 0,\ -1,\ \ldots \qquad (4.2)$$

In Figure 4.4 the triangles and quadrants represent the values of 4 samples of the sinusoids at the points 0, π/2, π and 3π/2 radians per cycle.

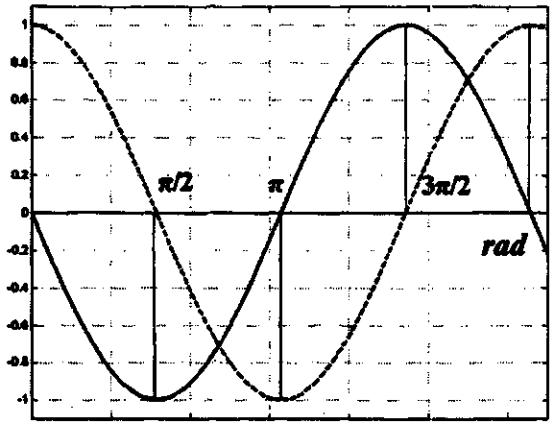


Figure 4.4 Orthogonal -sine (solid) and cosine (dashed) functions sampled at 2πn/4.

Since sine and cosine functions are orthogonal to each other, the value of sine (cosine) at 2πn/4 is equal to 1 or -1 when cosine (sine) is equal to zero. Obviously, there is no need for multipliers to perform a multiplication by 1 or –1.

Therefore, the frequency converter at the transmitter side can be considered to be a commutator (MUX- multiplexer) or a data selector that selects either the inphase component $u_I(n)$ or the quadrature component $u_Q(n)$ or their negative values as shown in Figure 4.5.

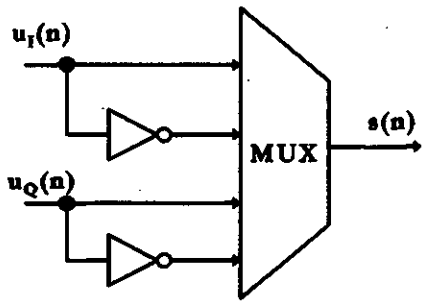


Figure 4.5 MUX-based frequency upconverter.

Thus the output of the digital frequency upconverter is given by:

$$s(n) = ...u_I(0), -u_Q(1), -u_I(2), u_Q(3), u_I(4), -u_Q(5), ... \tag{4.3}$$

Since the MUX-based digital frequency upconverter does not contain multipliers, adders, a relatively big LUT or any other expensive elements of the digital design, it requires only 63 LEs (logical elements), which is less than 1% of the available FPGA chip area. Therefore, the MUX-based digital frequency upconverter is highly suitable for the implementation in FPGAs.

4.3 Frequency Converter at Receiver Side

In order to solve the problem of frequency downconverter efficiency at the receiver side, the conventional structure of the NCO design can also be avoided. Unfortunately, due to the unknown phase of the received QAM signal the design strategy used in the modulator cannot be applied to the demodulator. Nevertheless, the design of a digital frequency converter can still be optimized eliminating expensive multipliers and a big look-up table (LUT). The CORDIC algorithm is an alternative to the NCO technique for digital frequency downconverter implementation.

4.3.1 Brief Introduction to CORDIC

CORDIC – Coordinate Rotation Digital Computer – is a hardware effective method for computing trigonometric, logarithmic, exponential and hyperbolic functions, as well as performing approximations of the complex multiplication, division, square root calculation and even discrete signal processing. The first time Jack E. Volder described this algorithm for the trigonometric functions approximation was in 1959 [15]. At that time CORDIC was considered to be an interesting mathematical algorithm and had little practical importance. In the late 1980s to early 1990s digital design became more popular than analog design. Since then, the CORDIC algorithm has become an extremely important, cheap and reliable tool used to accomplish numerous digital design tasks. For example, Hewlett Packard calculators use the CORDIC algorithm for the approximation of different mathematical functions [16]. The summary of various functions computed by the CORDIC algorithm can also be found in [17, 18].

In this thesis project the CORDIC algorithm was used for the frequency converter and the carrier recovery design. It will now be discussed in more detail.

4.3.2 CORDIC Algorithm

The complex multiplication at the digital frequency converter is performed by rotating the input signal by the accumulated phase. As discussed in Chapter 2, the QAM signal can be represented not only by its inphase and quadrature components but also as a phasor with a particular phase and magnitude (radius) [19]. A phasor with coordinates

(I_j, Q_j) can be rotated to the new coordinates (I_{j+1}, Q_{j+1}) by a phase angle φ as shown in Figure 4.6.

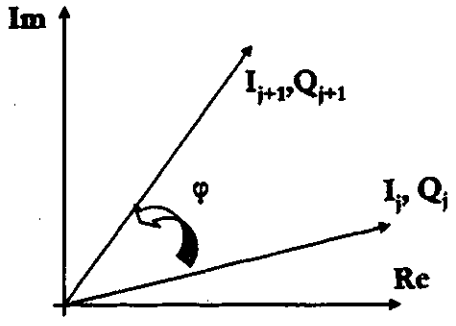


Figure 4.6 Phasor rotation by phase angle φ.

This rotation can be accomplished by using the rotational matrix [10, 18, 19]:

$$\begin{bmatrix} I_{j+1} \\ Q_{j+1} \end{bmatrix} = \begin{bmatrix} \cos\varphi & -\sin\varphi \\ \sin\varphi & \cos\varphi \end{bmatrix} \begin{bmatrix} I_j \\ Q_j \end{bmatrix} \tag{4.4}$$

The straightforward implementation of the rotational matrix does not have any advantage because it requires four variable input multipliers and a LUT for the sinusoids. The CORDIC algorithm can simplify the rotational matrix calculations by the following strategy.

First, $\cos\varphi$ is taken out of the rotation matrix:

$$\begin{bmatrix} I_{j+1} \\ Q_{j+1} \end{bmatrix} = \cos\varphi \begin{bmatrix} 1 & -\tan\varphi \\ \tan\varphi & 1 \end{bmatrix} \begin{bmatrix} I_j \\ Q_j \end{bmatrix} \tag{4.5}$$

Then, the entire phase angle φ is divided into m angle steps φ_w in such a way that the tangents of the selected angle steps represent a series of powers of 2, i.e.

$$\begin{cases} \tan\varphi_w = \sigma_w 2^{-w} \\ \varphi_w = \arctan(1/2^w) \\ \sigma_w = \pm 1, \end{cases} \tag{4.6}$$

where σ_w is the sign of the angle step. The sum of the angle steps φ_w must be equal to the entire rotation phase angle φ:

$$\sum_{w=0}^{m-1} \sigma_w \varphi_w = \varphi \tag{4.7}$$

Thus the entire rotation by a particular phase angle φ is divided into the m CORDIC stages of the elementary rotations by the phase step angles φ_w. The CORDIC angle steps as well as the appropriate tangents are listed in Table 4.1.

Table 4.1 CORDIC phase steps.

Phase step φ_w	Tangent of the phase step	Number of bits to shift
45^0	$2^{-0}=1$	0
26.56505^0	$2^{-1}=1/2$	1
14.03624^0	$2^{-2}=1/4$	2
7.12502^0	$2^{-3}=1/8$	3
3.57633^0	$2^{-4}=1/16$	4
1.78991^0	$2^{-5}=1/32$	5
0.89517^0	$2^{-6}=1/64$	6
...	...	...

The additional rotations by $\pm 90^0$ and $\pm 180^0$, which are possible to accomplish without the CORDIC algorithm, will be covered in the next chapter.

The angle step selection is performed assuming that the multiplication or division by a power of 2 can be implemented easily in hardware using only an inexpensive shift operation. For example, if inphase and quadrature components are multiplied by $\frac{1}{2}$, a binary value of the corresponding component is shifted by one bit position to the right. Similarly, in the case of the multiplication by $\frac{1}{4}$, a shift by two bit positions to the right is

performed and so forth. The number of bits to shift at each CORDIC stage is shown in the third column of Table 4.1.

The elementary CORDIC rotations are applied successively one after another. The entire complex multiplication is substituted by m consecutive shift and add operations as shown in Equation (4.8).

$$\begin{bmatrix} I_{w+1} \\ Q_{w+1} \end{bmatrix} = \cos\varphi_w \begin{bmatrix} 1 & -\sigma_w \frac{1}{2^w} \\ \sigma_w \frac{1}{2^w} & 1 \end{bmatrix} \begin{bmatrix} I_w \\ Q_w \end{bmatrix}, \tag{4.8}$$

where σ_w is the sign of the rotation step angle at CORDIC step w.

The phasor can be rotated clockwise or counter-clockwise at each stage of the CORDIC algorithm by checking the sign σ_w. There are two different methods of determining the sign of the angle step rotation. Depending on what method is applied, the CORDIC algorithm can be employed in two different modes, which are given as [15, 18]:

- *Rotation Mode*, in which the phasor is rotated to the new coordinates by the rotation phase angle φ.
- *Vectoring Mode*, in which the phasor magnitude and phase are computed from the known coordinates I_j and Q_j.

The rotation and vectoring modes with corresponding examples are discussed in the following sections.

4.3.2.1 Rotation Mode

In the rotation mode the phasor should be rotated by the phase angle φ. This is accomplished by m CORDIC rotations by the phase steps φ_w listed in Table 4.1. At each stage of the CORDIC rotation the direction of the rotation is determined by the sign σ_w. In the rotation mode the sign $\sigma_w = \pm 1$ for each angle step rotation is determined by the remaining phase Z_w, the part of the angle φ that has not been rotated yet. Therefore, σ_w must first be verified in order to decide whether to perform a clockwise or counterclockwise rotation:

$$\sigma_w = \begin{cases} +1 & \text{if } Z_w \geq 0 \\ -1 & \text{otherwise} \end{cases} \tag{4.9}$$

Finally, the three main equations of the CORDIC rotation can be represented as [15, 19]:

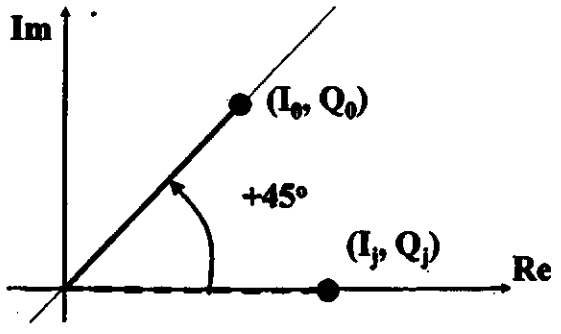
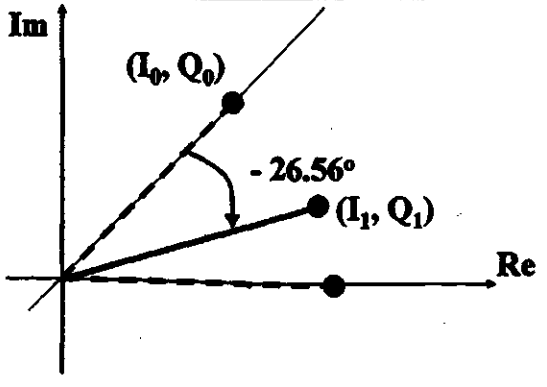
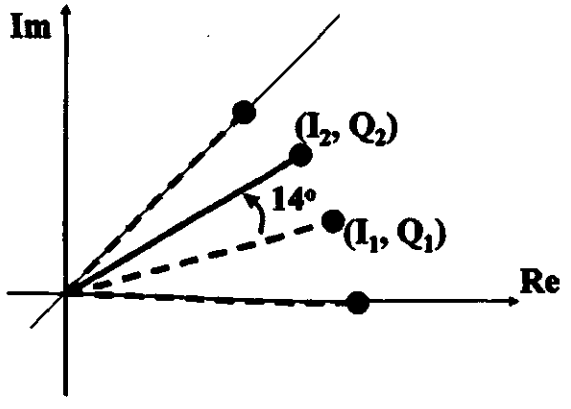
$$\begin{cases} I_{w+1} = I_w - Q_w \cdot \sigma_w \cdot 2^{-w} \\ Q_{w+1} = Q_w + I_w \cdot \sigma_w \cdot 2^{-w} \\ Z_{w+1} = Z_w - \sigma_w \cdot \arctan(2^{-w}) \end{cases}, \tag{4.10}$$

where σ_w is defined in Equation (4.9).

An example of the CORDIC in rotation mode is given in Table 4.2. Suppose there is a phasor with coordinates (I_j, Q_j) and suppose $Q_j = 0$ for simplicity reasons. The phasor should be rotated by a phase $\varphi = +30^0$. At CORDIC stage $w = 0$, the sign of φ is verified. Since $\varphi = +30^0$ is positive, the phasor is rotated by $\varphi_0 = 45^0$ while the remaining phase $Z_0 = 30^0 - 45^0 = -15^0$. The appropriate CORDIC formulas are given in the third column of Table 4.2. At CORDIC stage $w = 1$, since Z_0 is negative, $\sigma_0 = -1$ and the phasor is rotated by $\varphi_1 = -26.56^0$ while the remaining phase $Z_1 = -15^0 + 26.56^0 = 11.56^0$. Similarly, at

CORDIC stage w=2, since Z_1 is positive, $\sigma_1 = 1$ and the phasor is rotated by $\varphi_2 = 14^0$ while the remaining phase $Z_2 = 11.56^0 - 14^0 = -2.44^0$ as shown in Table 4.2.

Table 4.2 Example of CORDIC algorithm in rotation mode.

Stage w	CORDIC rotations	CORDIC rotation formulas
0		$I_0 = I_j - Q_j$ $Q_0 = Q_j + I_j$ $Z_0 = 30^0 - 45^0 = -15^0$
1		$I_1 = I_0 + Q_0/2$ $Q_1 = Q_0 - I_0/2$ $Z_1 = Z_0 + 26.56^0 = 11.56^0$
2		$I_2 = I_1 - Q_1/4$ $Q_2 = Q_1 + I_1/4$ $Z_2 = Z_1 - 14^0 = -2.44^0$
...	...	...

During the first three CORDIC rotations the phasor was rotated to the new coordinates (I_2, Q_2). At these coordinates the phase is equal to $\varphi_0 - \varphi_1 + \varphi_2 = +45^0 - 26.56^0 + 14^0 = 32.44^0$, which is already close to the desired 30^0 degrees. Obviously, the more CORDIC stages are involved the closer the phase angle gets to the desired φ and the smaller is the remaining phase Z_w as shown in Figure 4.7.

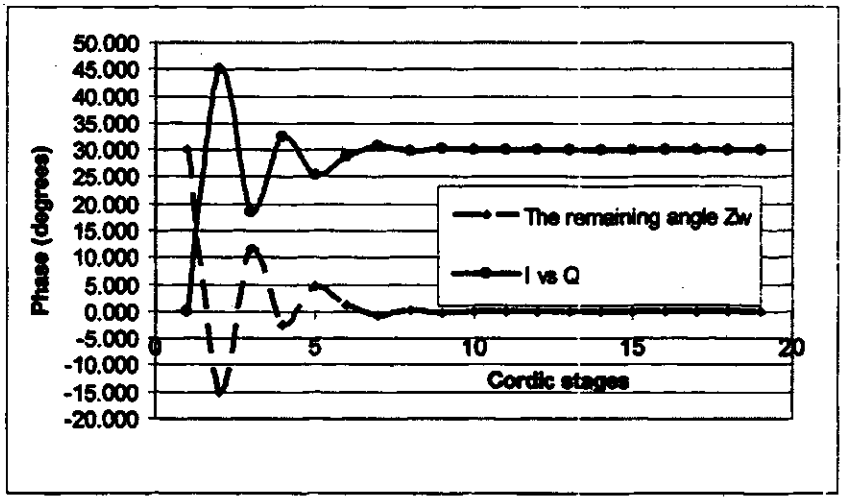


Figure 4.7 Remaining angle and I vs. Q angle.

The Rotation Mode structure of the CORDIC algorithm is shown in Figure 4.8. In the rotation mode the CORDIC stage has two outputs I_{w+1} and Q_{w+1} and three inputs I_w, Q_w and Z_w, where the latter is the remaining phase angle. At each CORDIC stage w the addition or subtraction operations are controlled by the sign of the remaining angle Z_w. The LUT shown in Figure 4.8, stores the values of CORDIC step phase angles listed in Table 4.1.

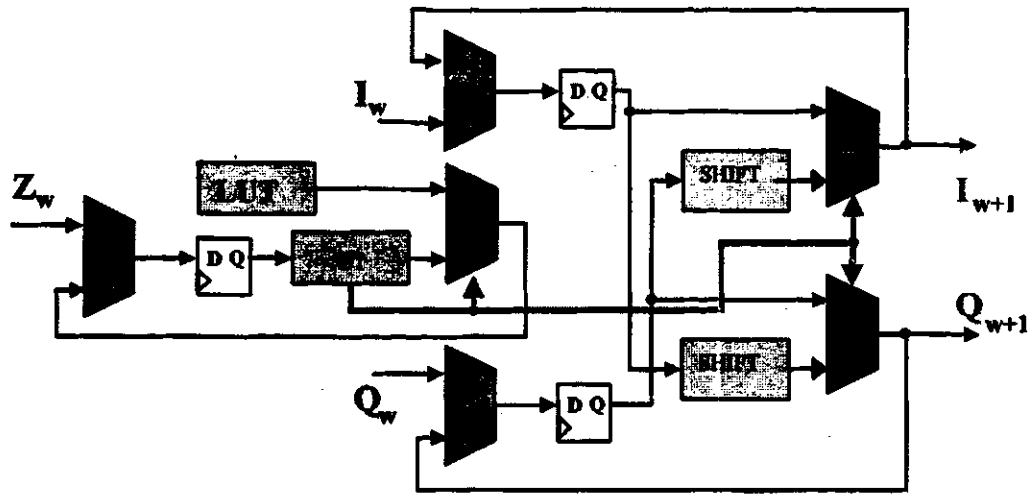


Figure 4.8 CORDIC algorithm structure for rotation mode.

This circuit could be used iteratively or replicated and stacked to make a pipeline. The trade off is between FPGA area and speed.

There is a significant difference between the LUT in the NCO and the LUT of the CORDIC algorithm shown in Figures 4.1 and 4.8 respectively. In Figure 4.8 a special LUT has to store only *m* phase angle values represented as K bit words while the LUT in the NCO has to store 2^K phase values. Thus, the digital frequency downconverter at the receiver side is implemented by applying the phase from the phase generator to the CORDIC algorithm with the rotation mode. The CORDIC-based digital frequency downconverter is illustrated in Figure 4.9.

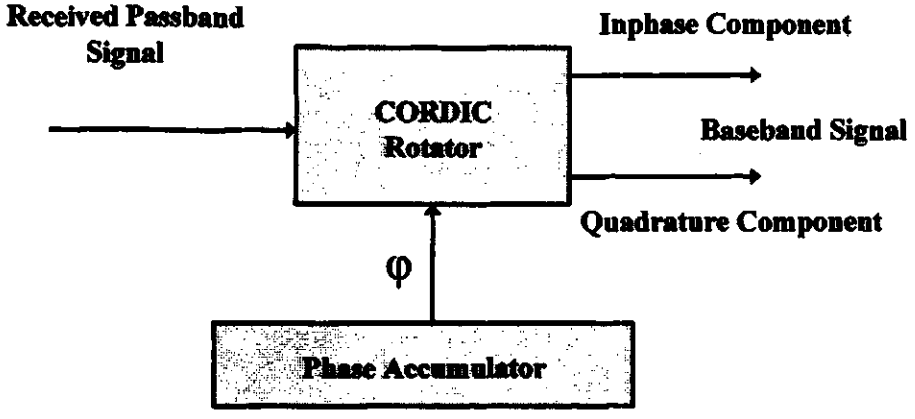


Figure 4.9 CORDIC-based frequency downconverter.

In the beginning of the CORDIC algorithm derivation the $\cos\varphi$ was taken out of the rotation matrix. Therefore, the multiplications by $\cos\varphi_w$ at every w CORDIC stage were omitted. Due to this, the CORDIC gives an imprecise calculation of the new coordinates. On the other hand, since $\cos(\varphi_w) = \cos(-\varphi_w)$ the permanent multiplication by $\cos\varphi$ results in the multiplication by a number A_m [19]. Fortunately, this number tends to reach a constant value, as the number of CORDIC stages goes to infinity, i.e.:

$$\begin{aligned} A_m &= \prod_{w=0}^{\infty} \cos\left(\arctan\left(\frac{1}{2^w}\right)\right) \\ &= \prod_{w=0}^{\infty} \frac{1}{\sqrt{1+\tan^2\varphi_w}} \\ &= \prod_{w=0}^{\infty} \frac{1}{\sqrt{1+2^{-2w}}} \\ &\approx 0.607253 \end{aligned} \tag{4.11}$$

In practice, these cosine multiplications by the angle steps can be considered a constant after approximately 7 or 8 CORDIC stages as shown in Figure 4.10. Obviously, more stages should be considered in order to achieve a more precise value of the constant.

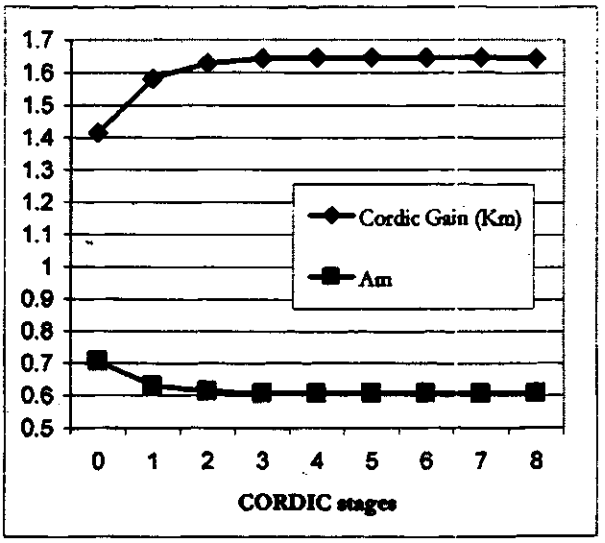


Figure 4.10 CORDIC gain K_m and inverse gain A_m.

Since the cosφ was taken out of the rotation matrix the multiplication by $A_m \approx 0.607253$ was omitted. Therefore, the algorithm has a CORDIC gain [19] of

$$K_m = \frac{1}{A_m} = \prod_{w=0}^{\infty} \sqrt{1+2^{-2w}} = 1.647. \tag{4.12}$$

The CORDIC gain may either be neglected by adjusting the receiver for slightly bigger values of I and Q, or compensated elsewhere in the receiver. Therefore, in the rotation mode the phasor with coordinates (I_j, Q_j) is rotated to the new coordinates (I_{j+1}, Q_{j+1}) by the entire angle φ, divided into the series of the angle step rotations [19]:

$$\begin{cases} I_{j+1} = K_m \left[I_j \cos\varphi - Q_j \sin\varphi \right] \\ Q_{j+1} = K_m \left[I_j \sin\varphi + Q_j \cos\varphi \right] \\ Z_{j+1} = 0 \\ K_m = \prod_{w=0}^{m-1} \sqrt{1+2^{-2w}} \approx 1.647 \end{cases} \tag{4.13}$$

Here m represents the number of CORDIC rotation stages and K_m is the CORDIC gain of the magnitude of the phasor.

Since the remaining angle Z_{j+1} is equal to zero, the rotation is considered to be complete. In practice though, $Z_{j+1} \neq 0$, it only approaches the zero value. Therefore, it is purely a designer's choice of what precision to apply. (The precision for this project is discussed in Chapter 6.)

4.3.2.2 Vectoring Mode

The CORDIC algorithm can also be used in the vectoring mode. In this mode the algorithm converts the coordinates of a rectangular system to the coordinates of a polar system. The polar coordinates help in the carrier recovery, a topic which is discussed in the following chapter. In vectoring mode, the phasor magnitude (radius) and the phase angle are established by the CORDIC successive step angle rotations. Contrary to the rotation mode, the CORDIC rotations are performed by instantly verifying the sign of the imaginary part (quadrature component).

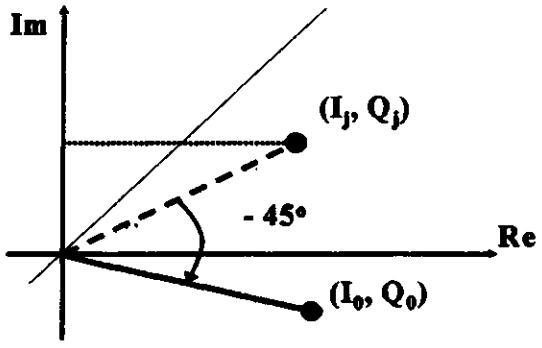
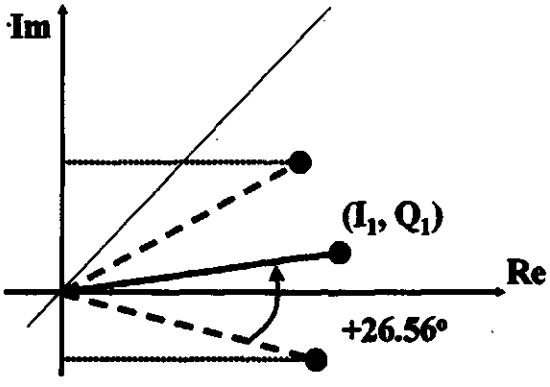
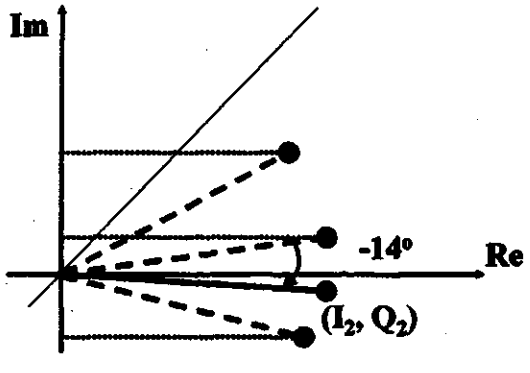
The elementary CORDIC rotations in the vectoring mode are performed in such a way that the quadrature component approaches zero. In the vectoring mode the CORDIC rotations are performed according to the same set of CORDIC equations given in Equation (4.10). However, the sign σ_w in these equations is determined by the sign of the quadrature component:

$$\sigma_w = \begin{cases} +1 & \text{if } Q_w \leq 0 \\ -1 & \text{otherwise} \end{cases} \tag{4.14}$$

In the vectoring mode, Z_w is considered not as the remaining phase, but as the cumulative phase angle. During the CORDIC algorithm, Z_w accumulates the phase value of the phasor.

For example, in Table 4.3, a phasor, with original coordinates (I_j, Q_j), has a phase angle equal to 30^0, which the CORDIC algorithm has to determine. At stage 0 of the CORDIC algorithm the sign of Q_j is checked. Since Q_j is positive the first elementary rotation by -45^0 is performed. The phasor is rotated to the new coordinates (I_0, Q_0). The appropriate formulas for this rotation are given in the third column in Table 4.3. The rotation phase of -45^0 is stored in the cumulative phase Z_0. At CORDIC stage 1 the sign of Q_0 is verified. Since Q_0 is negative, the phasor is rotated to the new coordinates (I_1, Q_1) by $+26.56^0$. At this stage, the cumulative phase Z_1 is equal to -18.44^0. Similarly, at CORDIC stage 2, the sign of Q_1 is verified. Since Q_1 is positive the next elementary rotation by -14^0 is performed. Thus, at stage 2, the cumulative phase Z_2 is equal to -32.44^0 as shown in Table 4.3.

Table 4.3 Example of CORDIC algorithm in vectoring mode.

Cordic stage w	CORDIC rotations	CORDIC rotation formulas
0		$I_0=I_j + Q_j$ $Q_0=Q_j - I_j$ $Z_0= -45^0$
1		$I_1=I_0 - Q_0/2$ $Q_1=Q_0 + I_0/2$ $Z_1= Z_0+26.56^0$ $= -18.44^0$
2		$I_2=I_1 + Q_1/4$ $Q_2=Q_1 - I_1/4$ $Z_2=Z_1 -14^0$ $= -32.44^0$
...	...	...

Depending on the number of CORDIC stages, the quadrature component gradually decreases to zero, and correspondingly, the angle between the inphase and quadrature component also reaches zero, as shown in Figure 4.11. At the same time, the cumulative

phase gradually reaches the negative value of the phase angle of the original phasor (I_j, Q_j), as shown in Figure 4.11.

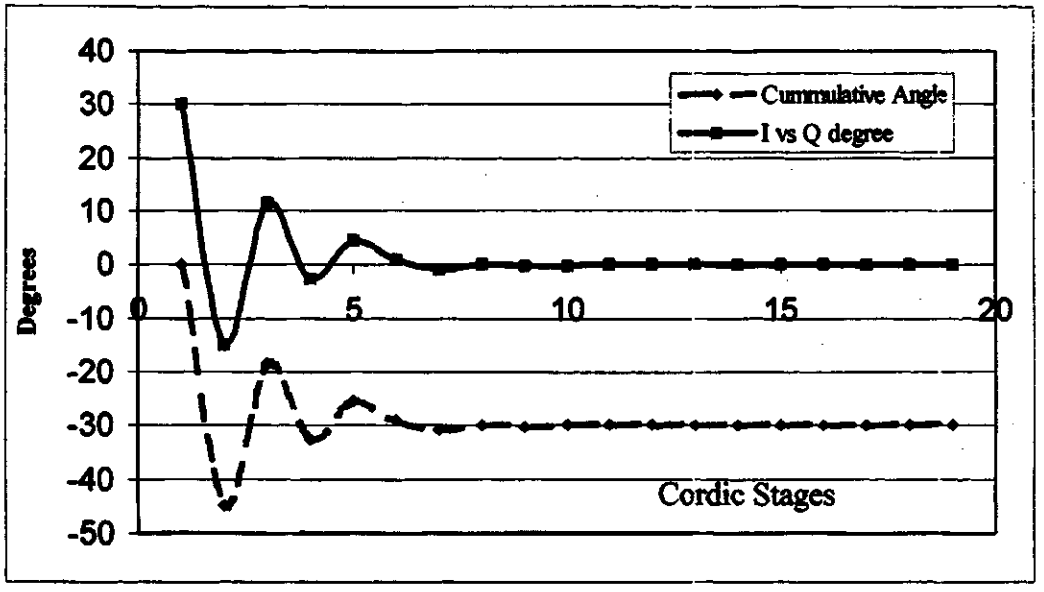


Figure 4.11 *I* vs. *Q* and cumulative angles.

The quadrature component can be considered as the imaginary component of the complex number, as discussed in Chapter 2. Since by the end of the CORDIC algorithm the quadrature component is equal to zero, therefore the magnitude (radius) of the complex number is equal to the inphase component (real component). Therefore, similar to Equation (4.13), by the end of the vectoring mode of the CORDIC algorithm:

$$\begin{cases} I_{j+1} = K_m \left[\sqrt{I_j^2 + Q_j^2} \right] \\ Q_{j+1} \approx 0 \\ Z_{j+1} = -\arctan \dfrac{Q_j}{I_j} \\ K_m = \prod_{w=0}^{m-1} \sqrt{1 + 2^{-2w}} \approx 1.647 \end{cases} \tag{4.15}$$

where Z_{j+1} is the cumulative phase angle at the end of the CORDIC algorithm, while K_m is the CORDIC gain. The CORDIC gain must be considered here once more. Therefore, the value of I_{j+1} is slightly bigger than the actual value of the radius. In this project all the thresholds were adjusted assuming the CORDIC gain.

The main advantage of the CORDIC algorithm, is that the phase and the radius of the complex signal, can be determined at the same time, using the Vectoring mode of the CORDIC algorithm. (The implications of this advantage will be discussed in the next Chapter, when the carrier recovery will be presented.)

In Figure 4.12 the conventional structure of the CORDIC algorithm in the vectoring mode is displayed. For simplicity reasons, only one CORDIC stage is shown. In Figure 4.12 there are two inputs I_w and Q_w and three outputs I_{w+1}, Q_{w+1} and Z_{w+1}, where the latter stands for the cumulative angle. Similarly to Figure 4.8 the LUT stores the values of the CORDIC phase angles, listed in Table 4.1. The main difference between Figures 4.8 and 4.12 is the method of determining the sign of the next angle step rotation. In Figure 4.8 the direction of the rotation at each CORDIC stage is determined by the remaining phase while in Figure 4.12 it is determined by the sign of the quadrature component Q_w.

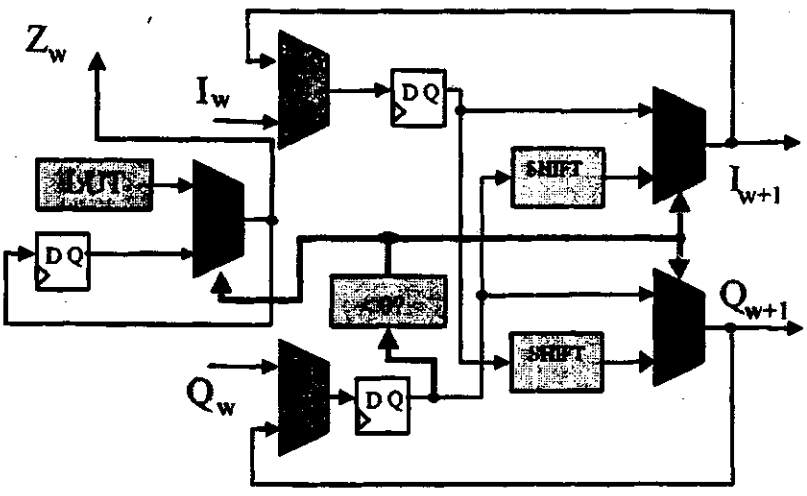


Figure 4.12 CORDIC algorithm structure for vectoring mode.

Unfortunately, if only a shift operation is applied, then the precision of the rotation decreases with every shifted bit. Therefore, in order to retain an acceptable precision, the shift operation is combined with a sign extension. The precision applied in this design, as well as the required size of the FPGA area, are covered in Chapter 7.

5 DIGITAL CARRIER RECOVERY

The carrier offset concept, its estimation and compensation, accomplished by applying a Digital Phase Locked Loop (DPLL) and CORDIC, are the main topics of this chapter.

5.1 Concept of Carrier Offset

According to the discussion in the previous chapters, cable systems use radio frequency (RF) channels for QAM signal transmission. The QAM signal is formed as a complex baseband signal and then translated to a radio frequency to fit into the allocated RF channel. The frequency translation or upconversion is usually done in two stages. First, the translation is done digitally to an intermediate frequency (IF) and second, the translation to the radio frequency (RF) is done using analog mixers. Similarly, the translation from RF back to the baseband is also usually done in two stages. The first stage uses an analog mixer to translate the signal to an IF frequency. The intermediate frequency signal is sampled to produce a digital IF signal. The digital IF signal is then translated to the baseband using a digital frequency downconverter, discussed in Chapter 4.

The translation from RF to IF is accomplished by mixing the RF QAM signal with a local oscillator (L.O.) as shown in Figure 5.1. A carrier offset problem arises when the downconverter in the receiver does not translate the RF spectrum by the exact amount.

The local oscillators are free running with their frequency governed by an electro-mechanical resonator (e.g. quartz crystal), which means the frequency will not be exactly the same [7, 8].

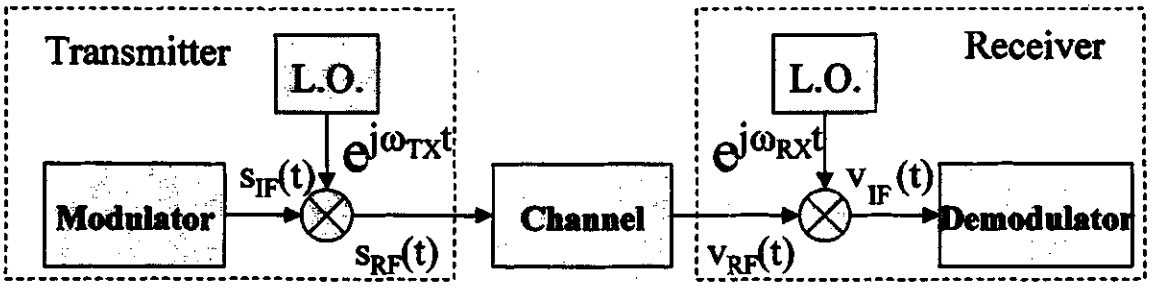


Figure 5.1 256-QAM mixing scheme.

Ideally, when the transmitter and receiver are first turned on, it can be considered that the mixers are working at the same frequency $\omega_{TX} = \omega_{RX}$ but there could be some constant phase offset φ_{const} between them. In reality though, the frequencies generated at the transmitter and the receiver side are not equal to each other due to the different clock sources. The difference between the receiver and the transmitter mixer frequencies can be defined as,

$$\begin{aligned} \Delta\omega &= \omega_{TX} - \omega_{RX} \\ &= 2\pi \cdot \Delta f, \end{aligned} \tag{5.1}$$

where $\Delta\omega$ has units radians per second. In the presence of the frequency offset, sampled at the symbol rate, the carrier offset can be represented as:

$$\varphi_{frequency}(m) = \Delta\omega \cdot mT, \tag{5.2}$$

where $\varphi_{frequency}(m)$ has units radians per symbol and T has units seconds per symbol and is the time between the symbol transmission, which is referred to as symbol spacing.

The entire carrier offset, which will drift from the mth symbol to the $(m+1)$th symbol, can be expressed as the sum of a constant phase offset and the frequency offset:

$$\begin{aligned}\varphi(m) &= \varphi_{frequency}(m) + \varphi_{const} \\ &= \Delta\omega \cdot mT + \varphi_{const} \\ &= 2\pi \cdot \Delta f \cdot mT + \varphi_{const},\end{aligned} \tag{5.3}$$

where $\varphi(m)$ has units radians per symbol.

According to Equation 2.21, the demodulated baseband signal can be represented by its inphase and quadrature components I_{RX} and Q_{RX}. Because of the carrier offset φ_m, these components are not equal to transmitted I_m and Q_m, as mentioned in Chapter 2. Equation 2.21 can also be represented as the rotation matrix form given in Equation 4.4. Thus, the received inphase and quadrature components can be considered as rotated, i.e.:

$$\begin{cases} I_{rotated}(m) = I_m \cos\varphi_m - Q_m \sin\varphi_m \\ Q_{rotated}(m) = I_m \sin\varphi_m + Q_m \cos\varphi_m, \end{cases} \tag{5.4}$$

where I_m and Q_m are the inphase and quadrature components of the transmitted signal.

Because of the carrier offset φ_m the entire 256-QAM constellation is either rotated by a constant angle or spinning, depending on whether φ_m is constant or increases with m.

For example, suppose that the transmitter constantly sends the same symbol, which is represented as a point in the 256-QAM constellation with inphase and quadrature components equal to 1, i.e. $I_m + jQ_m = 1+j1$. Ideally, assuming no carrier offset, the same symbol will be detected at the receiver. However, if $\varphi_{frequency}(m) = 0$ but φ_{const} is equal to -30^0, then the received symbol will be rotated by -30^0 to another position as illustrated in

Figure 5.2 (a), i.e. $(I_m + jQ_m)e^{j\varphi_{const}}$. In Figure 5.2 the transmitted symbol is shown as a circle, while the received symbol is represented by a dot. Finally, if $\Delta\omega \neq 0$, then the frequency offset $\varphi_{frequency}(m) \neq 0$ and the received symbol (represented as a point in the constellation) will start to spin from one position in the constellation to another. This can be represented as a point spinning at rate $\Delta\omega T$, which has units radians per symbol. The effect of carrier offset, when $\varphi_m = m\Delta\omega T$ is illustrated in Figure 5.2 (b).

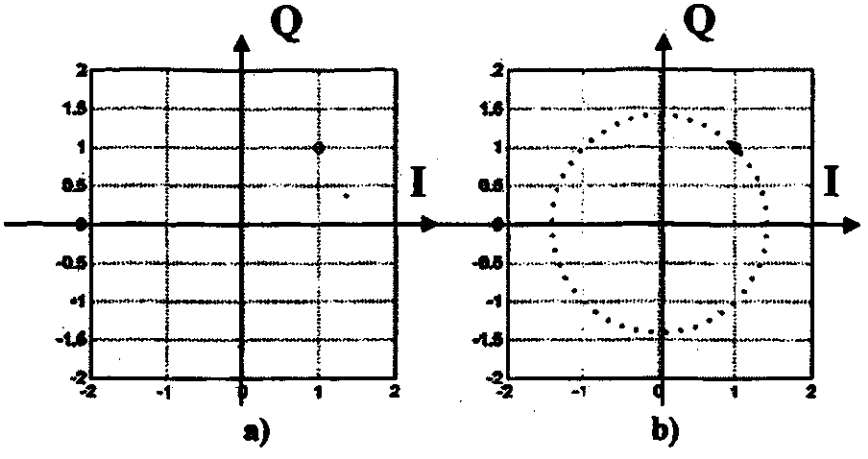


Figure 5.2 Carrier offset effect: a) rotation in presence of constant phase offset, b) spinning in presence of frequency offset (transmitted symbol - circle, received - dot).

In the given example, only one point in the constellation was considered. In the 256-QAM constellation there are 256 different points. In the presence of the phase offset the downconversion is inaccurate, thus the entire 256-QAM constellation is rotated by $\varphi(m) = \varphi_{const}$. In the presence of frequency offset, the downconversion is also inaccurate and causes the entire constellation to spin. Figures 5.3 and 5.4 illustrate the rotated and spinning 256-QAM constellations. (The transmitted symbols are represented as circles in the constellation, while crosses represent the rotated constellation.)

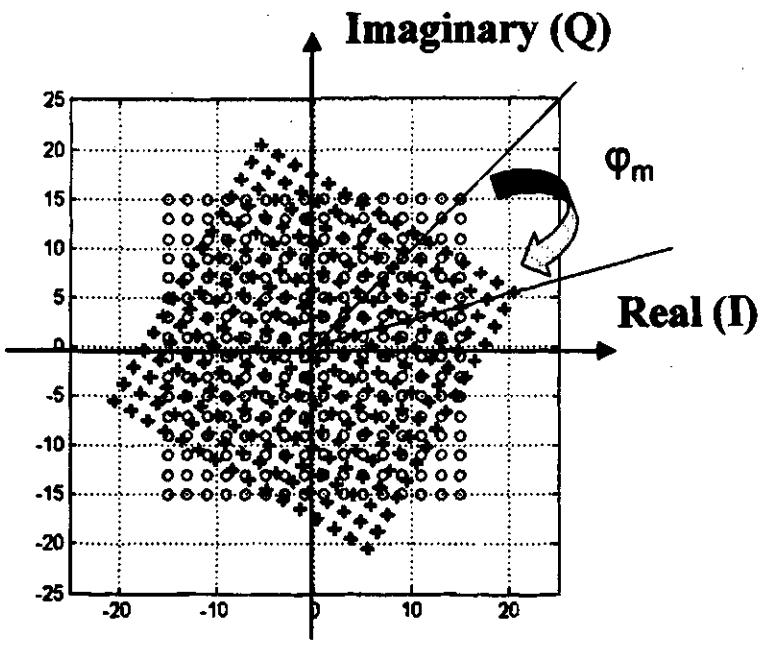


Figure 5.3 Rotated 256-QAM *I-Q* constellation (transmitted symbols (circles) and received symbols (crosses)) in presence of constant phase offset.

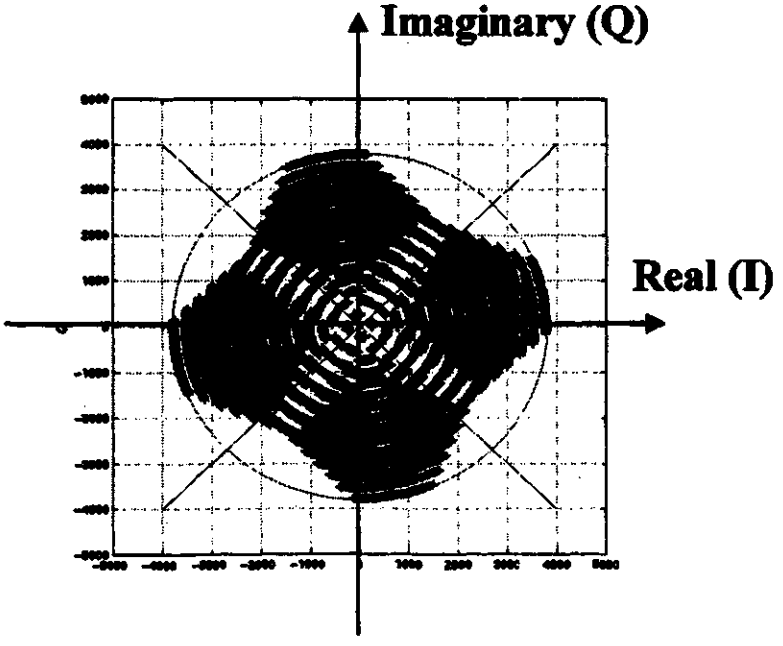


Figure 5.4 Spinning 256-QAM *I-Q* constellation of received symbols in presence of frequency offset.

Because of the carrier offset the receiver will detect an incorrect symbol. Thus, if nothing is done to recover the carrier, the end user at the receiver side will not receive the transmitted information.

5.2 Digital Phase Locked Loop

In order to compensate for the carrier offset, the digital receiver must recover the carrier. During the carrier recovery process the receiver must detect the carrier offset, compensate it and then constantly keep track of any carrier offset change. The common name for such a control device is the phase locked loop (PLL) [20, 21]. As previously stated, the main intent of this thesis is to eliminate, where feasible, an analog design. Therefore, preference is given to the Digital PLL (DPLL). The DPLL used for the carrier recovery, is part of the demodulator. A DPLL is a circuit that follows the matched filters. This is done in order to perform the carrier recovery at the lower symbol rate instead of the higher sampling rate. Also placing the DPLL after the matched filters eliminates the large delay line of the SRRC matched filters.

The structure of the DPLL, illustrated in Figure 5.5, consists of three components:

- carrier phase error detector,
- loop filter and
- carrier offset compensator or derotator.

The circuit that does the derotation is labeled a "derotator". The derotator has two inputs: the complex signal to be derotated and the amount by which it is to be derotated. The

carrier phase error detector measures the phase error of the derotated baseband signal. This error is added to the previously accumulated error to produce an estimate of carrier offset as shown in Figure 5.5.

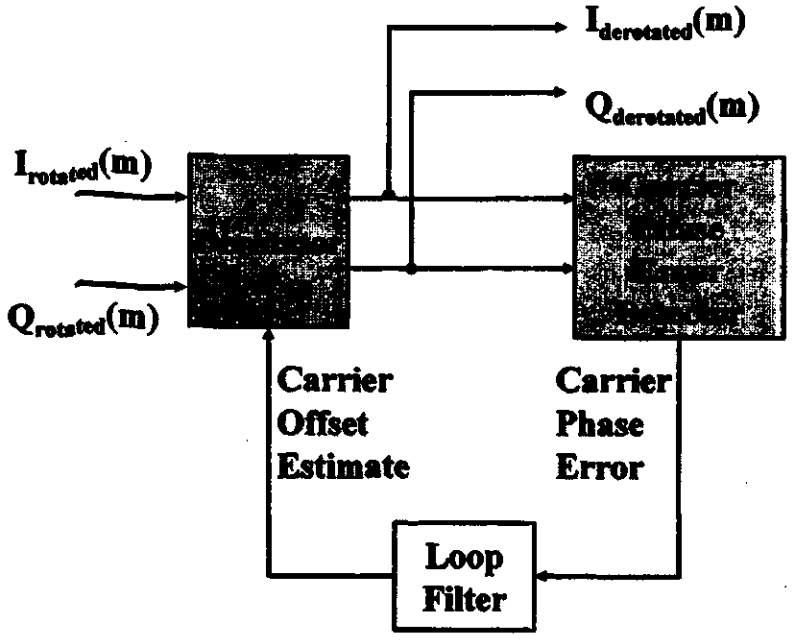


Figure 5.5 DPLL main components.

5.2.1 Model for DPLL

The typical structure of the linearized control loop model of the DPLL is shown in Figure 5.6 [20, 22].

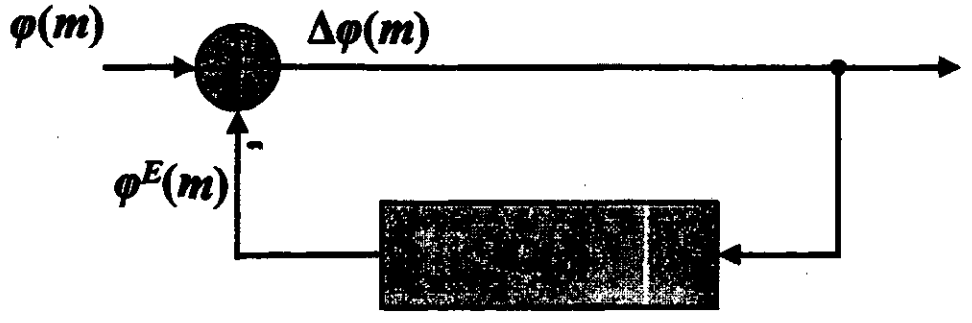


Figure 5.6 Linearization of loop model of DPLL.

The main purpose of the DPLL is to make the value of the carrier offset estimate, $\varphi^E(m)$, equal to the incoming carrier offset, $\varphi(m)$. Ideally, if $\varphi^E(m) = \varphi(m)$, then there is perfect synchronization between the receiver and the QAM signal carrier. On the contrary, if the carrier offset estimate is not equal to the carrier offset, then there will be some carrier phase error:

$$\Delta\varphi(m) = \varphi(m) - \varphi^E(m) \tag{5.5}$$

The carrier phase error is then averaged or accumulated by the loop filter $G(z)$ to estimate the carrier offset for the next incoming symbol. In the DPLL the carrier offset estimate value will eventually reach and track the value of the carrier offset. At the same time, the carrier phase error will decrease to zero and stay there. If $\varphi(m)$ is constant, the derotator derotates each symbol by the same amount. If $\varphi(m) = m\Delta\omega T$, then the derotator derotates the first symbol by $\Delta\omega T \bmod(2\pi)$, the second by $2\Delta\omega T \bmod(2\pi)$ and so on. This process leads to the carrier recovery and consequently, to the synchronization between the receiver and the carrier of the QAM signal.

5.3 Proposed Carrier Recovery Algorithm

The main intent of this algorithm is to perform carrier recovery not using a known training sequence. Thus, the carrier recovery is called a blind carrier recovery with random data.

The first challenge in this algorithm is the carrier offset detection. The carrier offset detection can be accomplished by permanently verifying the phase of the points in the 256-QAM constellation. There is a problem distinguishing which point is which, in the

presence of arbitrary rotations. As mentioned in previous chapters the points can be distinguished by the phase and the radius. The diagonal points could be used as the reference value for the correct carrier offset estimation in any quadrant of the 256-QAM constellation. All of the diagonal points are placed at the "diagonal angles" equal to: $+45^0$ -45^0 $+135^0$ and -135^0 degrees as shown in Figure 5.7. Thus, it can be assumed that if the diagonal point is placed at the diagonal angle then all the rest of the points in the constellation should be at the correct place as well.

There is an ambiguity determining in which of the four quadrants a diagonal point belongs. Thus, in this thesis it is assumed that the carrier phase error should be less than $\pm 45^{\circ}$, i.e. the system must lock before the error reaches the value where the point changes the quadrant.

When applying this method for the 256-QAM constellation several issues should be addressed. First, the radii of the points in the 256-QAM are extremely close to each other, therefore it may be necessary to use a significant number of bits in order to determine the difference between them. The other problem is that the difference between the radii of the diagonal points and the radii of the adjacent points is different for various diagonal points. Figure 5.7 shows the 256-QAM constellation with 32 diagonal points, while Figure 5.8 shows the radii of the diagonal points (black diamonds) plotted among the radii of all the rest of the points (grey circles) in the 256-QAM constellation.

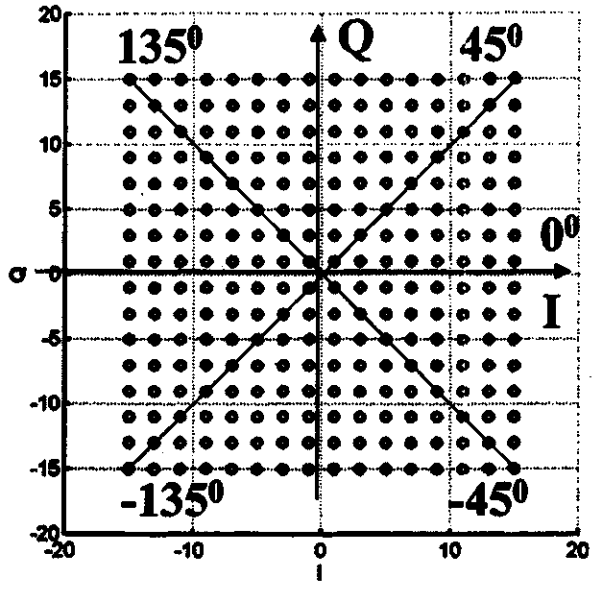


Figure 5.7 Diagonal points and diagonal angles in 256-QAM constellation.

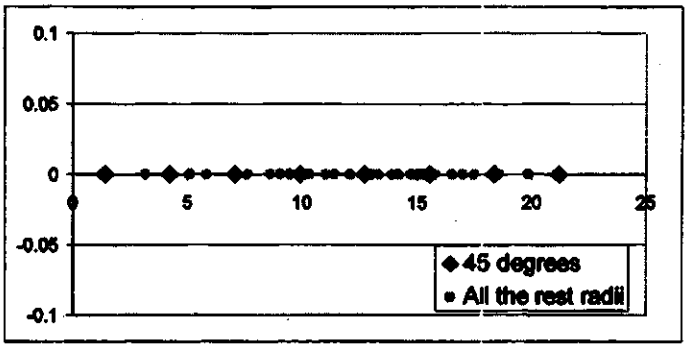


Figure 5.8 Radii of points in 256-QAM constellation (diagonal points - diamonds).

From Figure 5.8 it can be concluded that the only diagonal points, whose radius is different from the radius of any other point in the 256-QAM constellation, are the four corner points and the four points with the smallest radius. The outer corner points have the biggest radius and the highest value of I and Q, hence, there is more energy sent for these points than for any others. Therefore, the outer corner points are preferred for the carrier offset detection. Once the established radius indicates the corner point, the

calculated phase is compared with the phase of the diagonal point. The detected difference gives the carrier phase error $\Delta\varphi(m)$, defined in Equation (5.5).

By calculating the radius, it is possible to determine the corner point. The radius calculation of the complex number is quite a challenge in hardware implementation. Fortunately, the CORDIC algorithm in vectoring mode, discussed in Chapter 4, can simultaneously detect the phase and the radius of the received phasor and the corresponding point in the 256-QAM constellation. In vectoring mode the CORDIC algorithm rotates the point to the zero degree phase position establishing the phase of the point. For the carrier recovery task it is necessary to make CORDIC calculate not the phase of the point but the difference between the phase of the point and the phase of the corner point.

In order to use all the corner points, regardless of what quadrant they are in, it is important to rotate the point to the first quadrant. This can be done applying *initial rotations* by the initial rotation angles. The direction of the initial rotation is determined from the signs of inphase and quadrature component values. In Table 5.1 the initial values of inphase and quadrature components are shown as $I_{initial}(m)$ and $Q_{initial}(m)$ while the initially rotated components are shown as $I_{quad}(m)$ and $Q_{quad}(m)$.

Table 5.1 Initial rotations.

Sign of $I_{initial}$	Sign of $Q_{initial}$	Initial Rotation Angle (degrees)	Rotated I_{quad}	Rotated Q_{quad}
+	+	0^0	$I_{initial}$	$Q_{initial}$
+	-	90^0	$-Q_{initial}$	$I_{initial}$
-	+	-90^0	$Q_{initial}$	$-I_{initial}$
-	-	$(\pm)\,180^0$	$-I_{initial}$	$-Q_{initial}$

In order to make CORDIC calculate the difference between the phase of the point and the phase of the corner point, an *additional rotation* is applied. The additional rotation by -45^0 is performed once the initial rotation is done. This operation is taken from the CORDIC algorithm and is performed by the addition and subtraction operation:

$$\begin{cases} I_{cordic}(m) = I_{quad}(m) + Q_{quad}(m) \\ Q_{cordic}(m) = Q_{quad}(m) - I_{quad}(m) \end{cases} \tag{5.6}$$

The following example is given to illustrate the initial and additional rotations. In Figure 5.9 the point represents the received symbol. Suppose this is a corner point with the phase -150^0. As mentioned before, the carrier phase error should not be more than $\pm 45^0$. This means that the carrier phase error should be calculated between the phase of the point and the phase of the closest corner point, which is in the same quadrant. Thus, the carrier offset at symbol m is $-150^0 - (-135^0) = -15^0$. In order to determine this carrier phase error, the point is rotated to the first quadrant by the initial rotation angle listed in Table 5.1. Since the inphase and quadrature components are both negative, thus the initial rotation by $\pm 180^0$ is performed by inverting the signs, i.e.

$I_{quad}(m) = -I_{initial}(m)$ and $Q_{quad}(m) = -Q_{initial}(m)$ as shown in Figure 5.10 (a).

Figure 5.10 (b) illustrates the following additional rotation by -45^0 performed according to Equation (5.6).

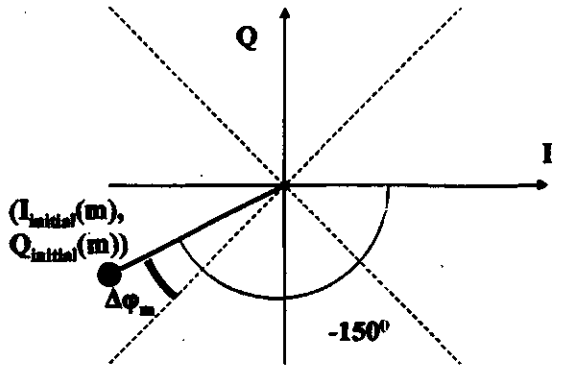


Figure 5.9 Corner point in presence of carrier offset.

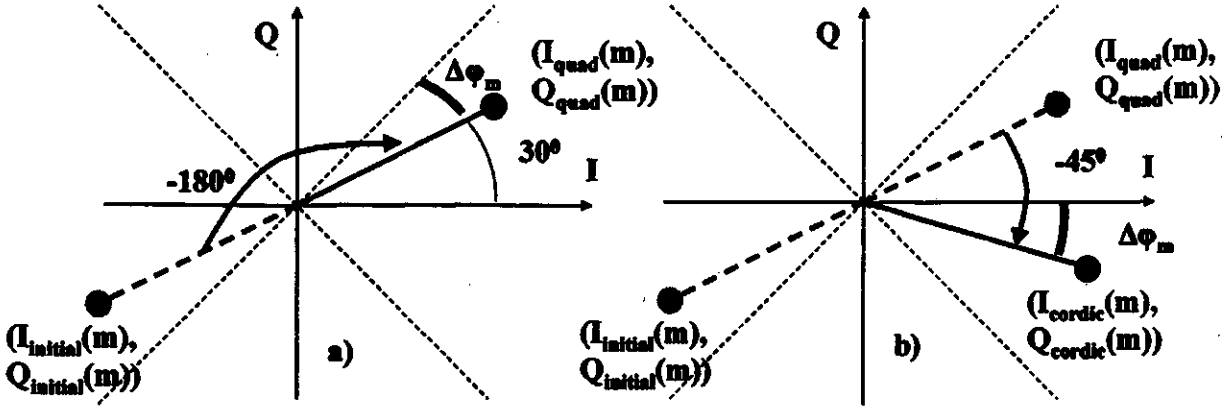


Figure 5.10 a) Initial rotation by $\pm 180^0$, b) additional rotation by -45^0.

The purpose of the initial and the additional rotations is to make the angle between the point and the zero reference (I axis) equal to the carrier phase error, $\Delta\varphi(m)$, as shown in Figure 5.10 (b). Because of the initial and additional rotations, the following CORDIC algorithm in vectoring mode does not determine the phase of the particular point, but the difference between the phase of the point and the diagonal phase angle, i.e. the carrier phase error value. As discussed in Chapter 4, the CORDIC algorithm in vectoring mode can simultaneously determine the carrier phase error value, $\Delta\varphi(m)$, and the radius. Once

the radius is in the region of the corner point radius, the carrier phase error value, $\Delta\varphi(m)$, is sent to the loop to update the carrier offset estimate, $\varphi^{E}(m)$.

Using this technique the carrier offset should be less than $\pm 45^{0}$, therefore, the CORDIC algorithm in vectoring mode can start from the stage w=1 with a 26.56^{0} CORDIC phase angle step.

The more detailed, proposed carrier recovery algorithm, illustrated in Figure 5.11, consists of three main components:

- After the initial (quad flip) and the additional rotations, the CORDIC algorithm in vectoring mode should constantly determine the carrier phase error $\Delta\varphi^{1}(m)$ and the radius(m).
- If this is a corner point, the carrier phase error should then be sent to the loop, i.e. $\Delta\varphi^{11}(m) = \Delta\varphi^{1}(m)$. Before coming to the loop filter, $\Delta\varphi^{11}(m)$ could be sent through the low-pass additional filter, which will attenuate the high frequency noise of the CORDIC algorithm. The output of the additional filter is $\Delta\varphi(m)$ which is finally sent to the loop filter, in order to update the value of the carrier offset estimate, $\varphi^{E}(m)$.
- The CORDIC based algorithm in the rotation mode, discussed in the previous chapter, will keep derotating the rotated inphase and quadrature components by a carrier offset estimate, $\varphi^{E}(m)$, phase angle. The output of the CORDIC derotator,

$I_{derotated}(m)$ and $Q_{derotated}(m)$, is the inphase and quadrature components after the carrier recovery.

In order to constantly track the carrier offset, the $I_{derotated}(m)$ and $Q_{derotated}(m)$ components are sent back to the loop, so that $I_{initial}(m) = I_{derotated}(m)$ and $Q_{initial}(m) = Q_{derotated}(m)$ as shown in Figure 5.11.

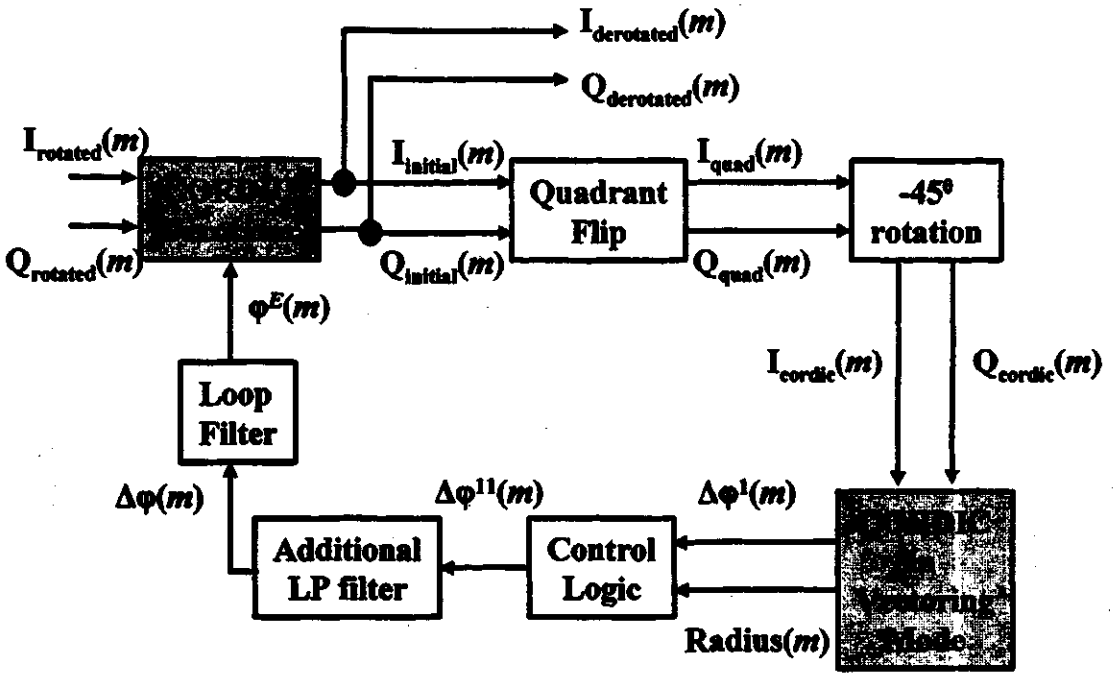


Figure 5.11 Block diagram of proposed carrier recovery.

The entire carrier recovery is represented as an algorithm, illustrated in Figure 5.12. The important assumption in this algorithm is that the carrier offset estimate is updated only if a corner point is detected.

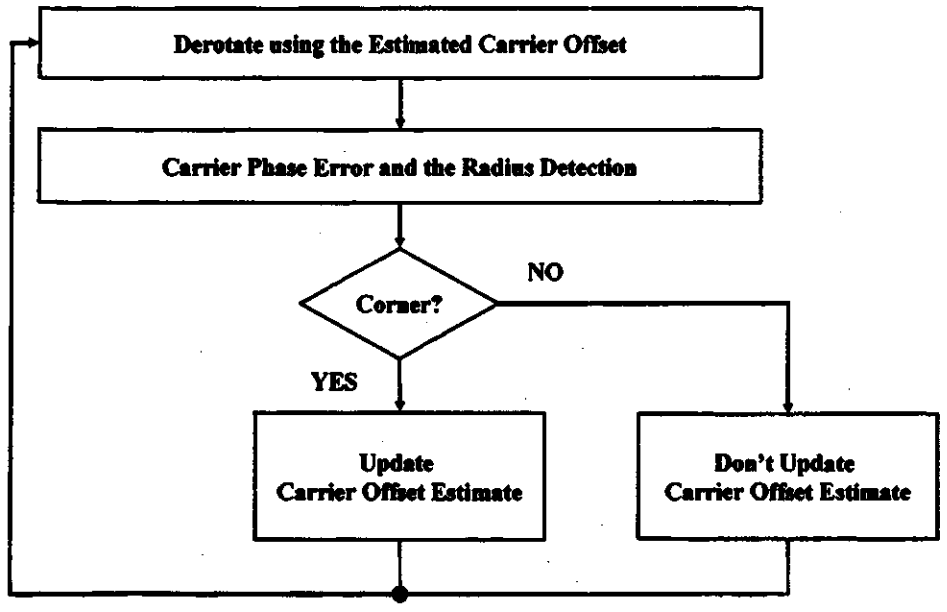


Figure 5.12 Proposed carrier recovery algorithm

5.4 Loop Filter

The operation of the system and the loop filter can be explained in the time domain or in the z-domain. Taking the z-transform is the most convenient approach of representing how the DPLL works. According to Figure 5.6, the z-transform of the carrier phase error at the symbol m can be represented as follows:

$$\begin{aligned} &\Phi(z) - \Phi^E(z) = \Delta\Phi(z) \\ &\Phi(z) - \Delta\Phi(z) \cdot G(z) = \Delta\Phi(z) \\ &\Phi(z) = (1 + G(z))\Delta\Phi(z) \end{aligned} \tag{5.7}$$

Therefore, the DPLL can be expressed by two different transfer functions:

$$\Psi(z) = \frac{\Delta\Phi(z)}{\Phi(z)} = \frac{1}{1+G(z)}$$

or (5.8)

$$H(z) = \frac{\Phi^E(z)}{\Phi(z)} = \frac{G(z)}{1+G(z)},$$

where $G(z)$ is the loop filter transfer function.

In order to compensate for the carrier offset, the transfer function, $\Psi(z)$, has to gradually eliminate the carrier phase error, while the transfer function $H(z)$ must force the carrier offset estimate to sequentially track the carrier offset [20, 21, 22].

The simplest loop filter for the DPLL is the first order infinite impulse response (IIR) filter, consisting of a unit delay and an integrator, which has a typical structure shown in Figure 5.13.

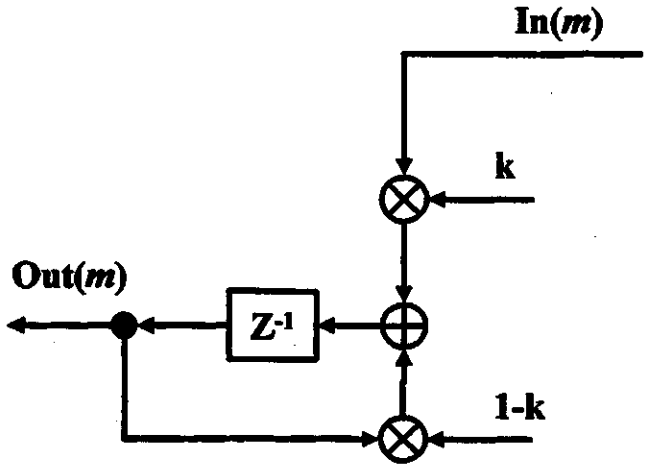


Figure 5.13 Typical first order loop filter for DPLL

Here k must take values (0, 2) to ensure the poles are inside the unit circle, which is the criterion for stability [23].

There are many different techniques of carrier recovery in the presence of constant phase offset. However, the main objective of the proposed algorithm is to perform the carrier recovery in the presence of frequency offset. In Appendix A, using the final value theorem it is shown that the first order IIR filter can be used for the constant phase offset recovery but is not sufficient for the frequency recovery. To help visualize the problem,

the first order loop filter effect is shown in Figure 5.14 (a), where the dashed-dotted line represents the carrier offset, $\varphi(m)$, ramp function with a step $\Delta\omega$, while the dotted line represents the carrier offset estimate, which should track the carrier offset. The solid line, which represents the carrier phase error, should gradually reach the zero level. Figure 5.14 (b) shows the first 35 symbol spacings, T, of Figure 5.14 (a) on a larger scale. In Figure 5.14 (c) it is quite clear that the carrier offset estimate attempts to reach the carrier offset value but in practice it is never equal to the carrier offset. A similar conclusion is made by observing Figure 5.14 (d), where the carrier phase error does not decrease to zero but keeps its value at a constant level. The bigger k (in Figure 5.13) is the more the carrier phase error is attenuated, but will never be equal to zero.

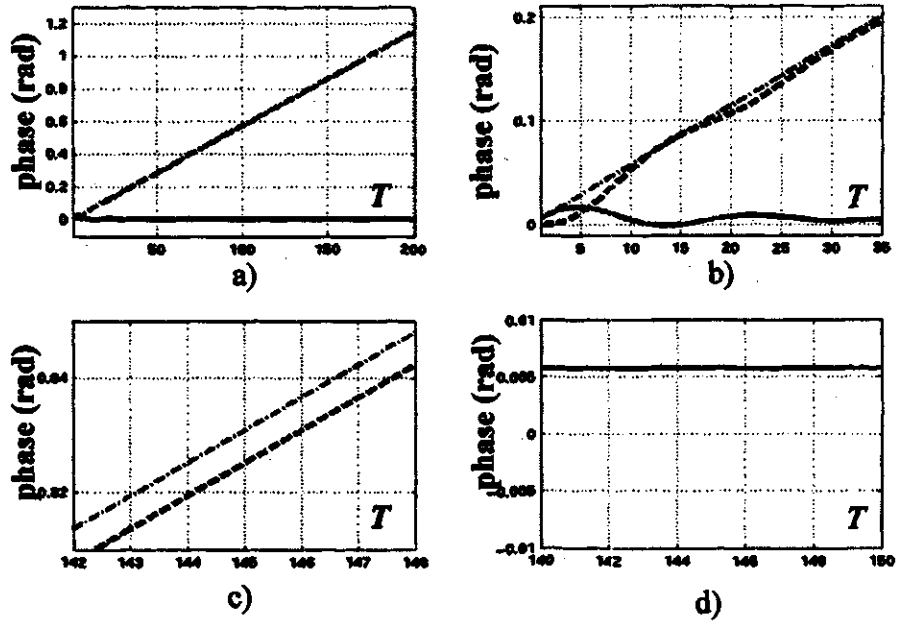


Figure 5.14 Effect of first order loop filter: a) carrier offset (dashed dotted), carrier offset estimate (dashed) and carrier phase error (solid), b) first 35 symbol spacings, c) estimated value is not exactly equal to carrier offset, d) carrier phase error does not reach zero level.

In order to compensate the frequency offset, the loop filter may be represented, not as a single integrator, but as a double integrator design [23, 24]. The structure of the loop filter consisting of two integrators with coefficients k_1 and k_2 is shown in Figure 5.15.

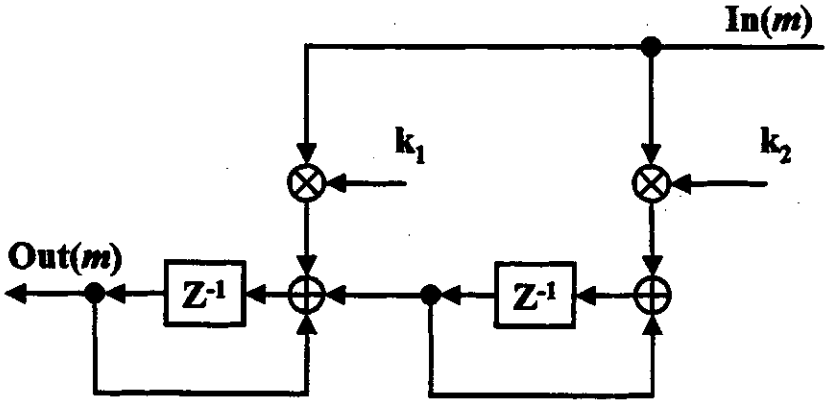


Figure 5.15 Structure of second order loop filter

The final value theorem, given in Appendix A, shows that by using the second order IIR loop filter, the steady-state carrier phase error approaches zero for both frequency and phase offset.

The effect of using a second order IIR loop filter is shown in Figure 5.16. The carrier offset estimate tracks the carrier offset in 5.16 (c), while in 5.16 (d) the carrier phase error approaches the zero level.

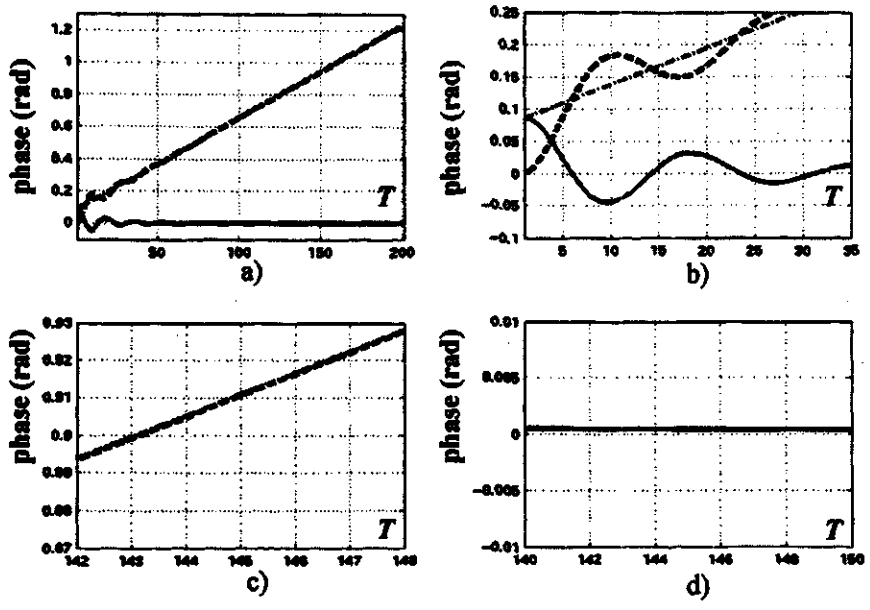


Figure 5.16 Effect of second order loop filter: a) carrier offset (dashed dotted), carrier offset estimate (dashed) and carrier phase error (solid), b) first 35 symbol spacings, d) estimated value is exactly equal to carrier offset, e) carrier phase error reaches zero level.

In order to select the coefficients of the loop filter it is important to determine in what frequency range the frequency offset is. The frequency offset can be represented as follows.

First, the precision of the L.O., expressed in parts per million (ppm), is considered. The precision of a cheap and widely used L.O. is usually equal to ±50 ppm. Although in cable systems the frequency, used for the downstream, ranges from 50 MHz to 750 MHz [3], in this project the highest L.O. frequency is limited to 74 MHz. Thus, the highest frequency offset is equal to $74 \cdot 10^6 \cdot 100/10^6 = 7400$ Hz. Since the carrier recovery is performed at the symbol rate, R_S, it is useful to represent the frequency offset in cycles per symbol. Thus, in the worst case scenario the highest frequency offset assumed in this

project is equal to $7400/R_S = 7400/2.0588 \cdot 10^6 \approx 0.00359$ cycles per symbol or 1.29^0 per symbol.

Therefore, the frequency offset is considered to be in a low frequency region. Keeping that in mind, the coefficients of the filter, shown in Figure 5.15, were adjusted using Matlab simulations for 256-QAM. It was empirically determined that the filter coefficients should be equal to $k_1 = 1$ and $k_2 = 2^{-12}$.

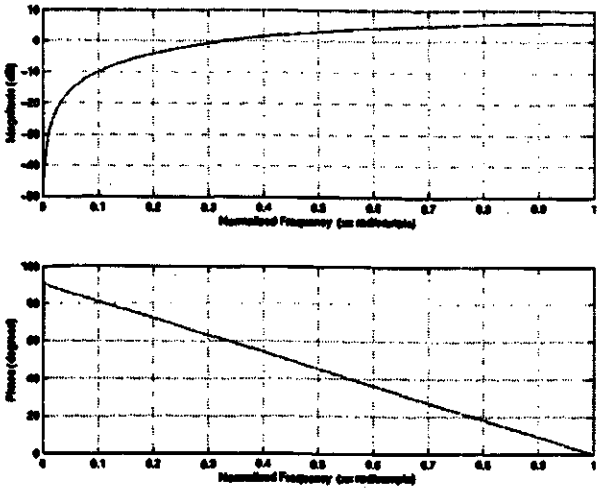


Figure 5.17 Magnitude (top) and phase frequency response of $|\Psi(e^{j\omega})|_{dB}$.

The magnitude and phase frequency response of the loop transfer functions are given in Figures 5.17 and 5.18. In Figure 5.17 (top), as expected, the frequency response of $\Psi(z)$ attenuates the low frequency component where the carrier offset is, while $H(z)$ tracks the low frequency component in Figure 5.18 [top].

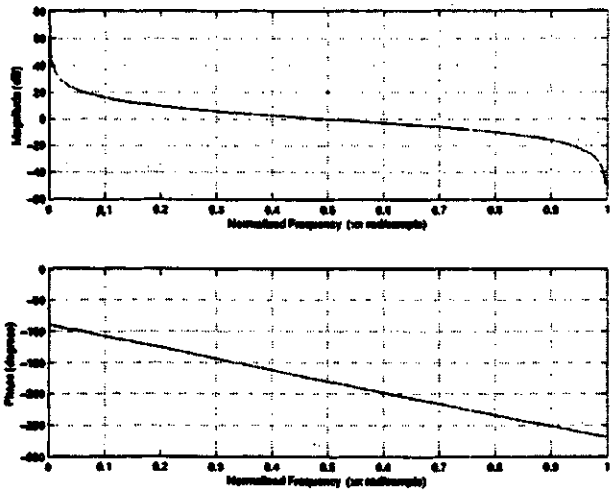


Figure 5.18 Magnitude (top) and phase frequency response of $|H(e^{j\omega})|_{dB}$.

The experiment and its result are discussed in the following chapter.

6 SOFTWARE AND HARDWARE SIMULATION RESULTS

This thesis addresses several interesting topics in digital cable system design. First, the project was conceived and simulated in Matlab. The Matlab design was optimized for hardware implementation.

The hardware design was performed using behavioral description in Verilog HDL (Hardware Descriptive Language) and synthesized using LeonardoSpectrum and Quartus II software [25]. Parts of the project were simulated using Altera Quartus II and then implemented in an FPGA Altera chip. A two's complement numbering system was used throughout the project.

6.1 General Test Set-up

The modem design was implemented using a TR*L*abs board with a local oscillator (L.O.) and an Altera Apex 20 KE FPGA chip. The general test set-up block diagram is shown in Figure 6.1. The modem design was evaluated using a spectrum analyzer and a logic analyzer. The detailed explanation of the modem design is given in the following sections.

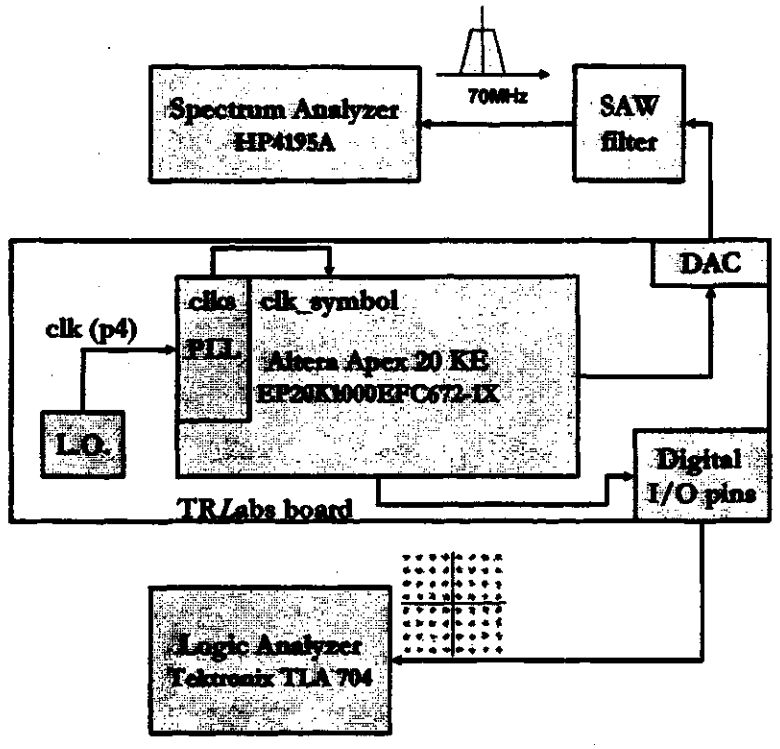


Figure 6.1 General test set-up block diagram.

6.2 Modem Design

6.2.1 PN Generator

One of the main ideas in this project was not to apply any data aided sources. Thus, a random data sequence was assumed. For this purpose the PN (pseudo-noise) random data generator, shown in Figure 6.2, was implemented in the modulator.

In this project the PN generator consists of a 16-bit long shift register with the feedback, based on a parity polynomial. The periodic binary sequence, generated by the shift register, has a period equal to:

$$b = 2^d - 1, \tag{6.1}$$

where d is the integer number, which represents the length of the shift register. Since $d = 16$, the binary sequence recycles at every $2^{16} - 1 = 65535$ time – the largest possible period [7].

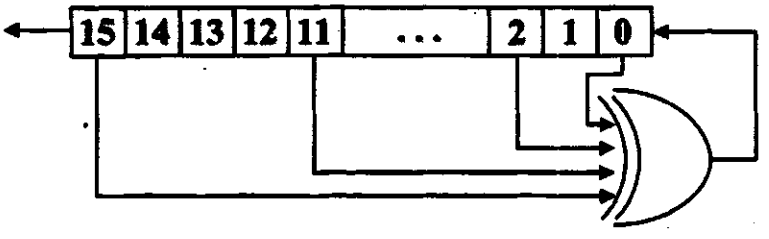


Figure 6.2 PN sequence generator.

6.2.2 System Clock Organization

The second task was the system clock organization. The timing is extremely important for reliable modulation and demodulation. The Altera Apex 20KE series EP20K1000EFC672-1X FPGA chip, used in this project, has 4 embedded PLLs shown in Figure 6.3. These PLLs are used for the clock control. The dedicated clock lines reduce clock delay and skew within the device, which results in the minimization of the clock-to-output, setup and hold times.

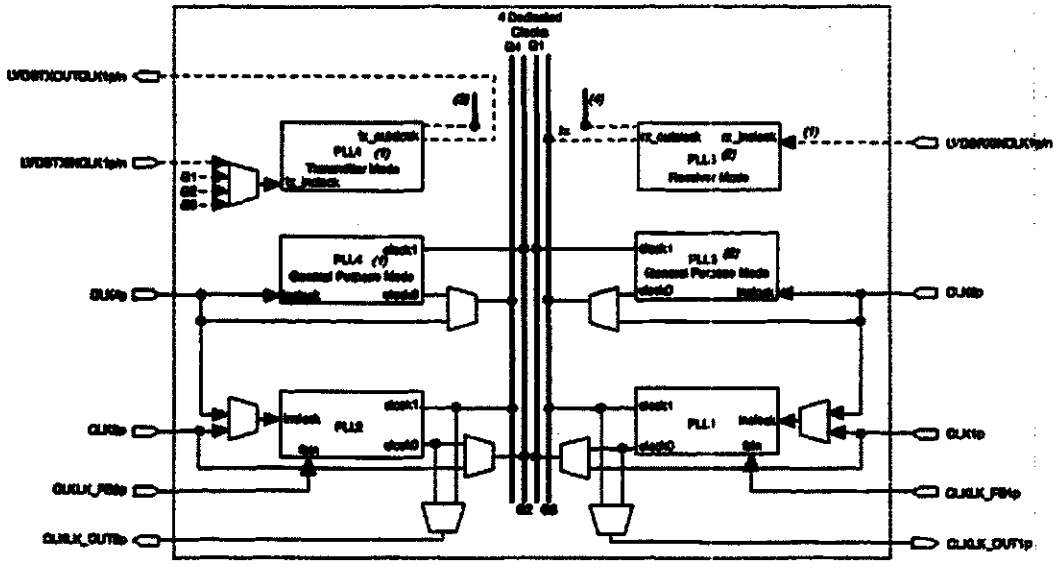


Figure 6.3 Overview of PLLs in APEX 20KE FPGA chip

(Source: www.altera.com).

The local oscillator generates a constant clock for the Altera board, while PLLs can decrease or increase this clock according to the design requirements. Unfortunately, a single master clock cannot drive every flip-flop in a multirate project. Therefore, all the remaining clocks are derived from the global one using a structure based on a synchronous counter design shown in Figure 6.4.

In Figure 6.4, "p4" is the clock that arrives from the local oscillator or an external clock source. The output "clks" is the sampling clock and a serial bit stream clock equal to 16.47 MHz. The symbol clock "clk_symbol", equal to 2.0588 MHz, is derived from the sampling clock "clks".

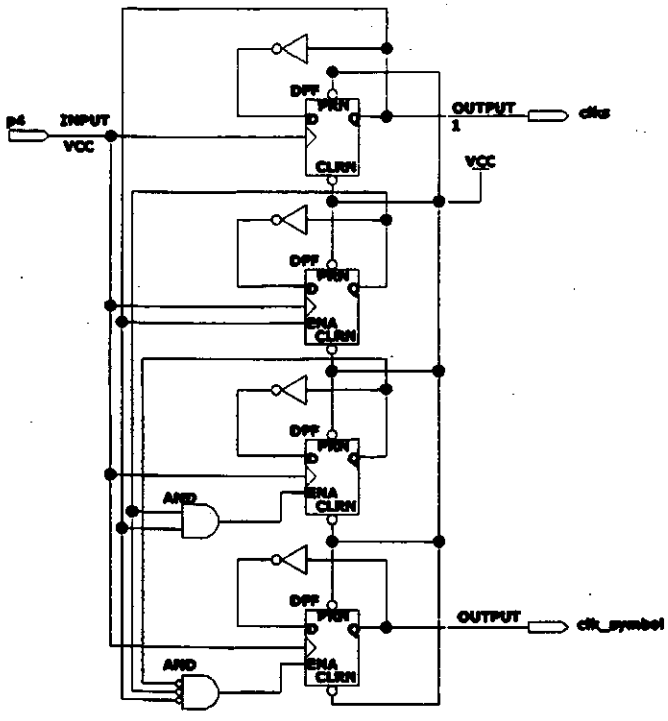


Figure 6.4 Modem clocks.

The block diagram of the modem clocks was developed in Quartus II software. The timing diagram for the symbol clock (clk_symbol) as well as the sampling clock (clks), which is 8 times faster than the clk_symbol, is represented in Figure 6.5.

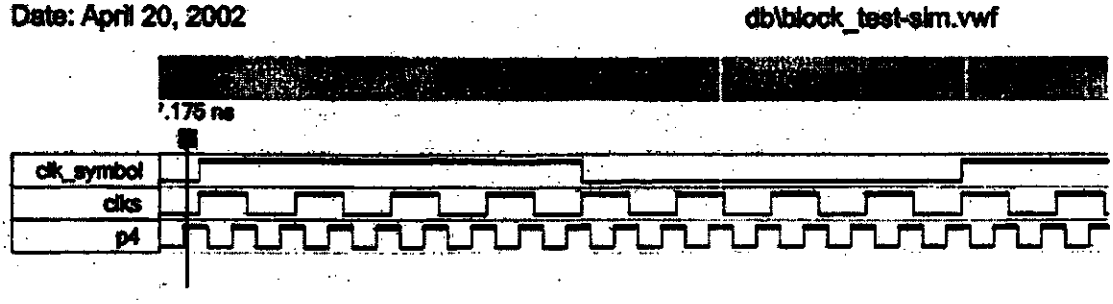


Figure 6.5 Symbol clock (clk_symbol) and sampling clock (clks) as derived from local oscillator (p4).

6.2.3 SRRC Filters

The third task was to design the SRRC filters. According to the discussion in Chapter 3, the digital shaping and matched filters were implemented to satisfy the J.83 standard. As stated in Chapter 3, the SRRC shaping filter with a polyphase structure required 2645 LEs (logical elements) out of 38400 available LEs, which is about 2.1 times less than the 5710 LEs needed for the conventional filter with a transversal structure. The matched SRRC filter with the polyphase structure required 4432 LEs, which is still less than the 5710 LEs required for the corresponding filter with the transversal structure. Therefore, the shaping and matched filters with the polyphase structure were used in this project.

6.2.4 Modulator and Demodulator

The entire digital modulator is illustrated in Figure 6.6. The digital modulator consists of a serial to parallel converter, an IQ mapping block, a pair of shaping filters for the inphase and quadrature components and a frequency upconverter. A PN generator provided a test input sequence.

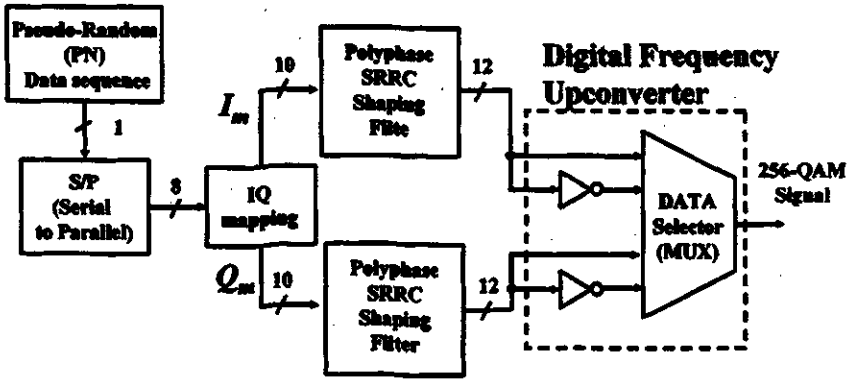


Figure 6.6 Modulator.

The modulator was evaluated using 256-QAM, at a symbol rate of 2.0875 Msymbols per second, and an IF frequency of 4.1125 MHz. The produced 256-QAM signal with ~2.7-MHz absolute bandwidth was then sent to the D/A (digital-to-analog) converter. Since the sampling to carrier frequency was $f_s/f_c = 4$, $f_c = \pi/2$ rad/sample and f_s=16.47 MHz, thus one of the spectra was centered at

$$f_c = 4f_s \pm \frac{f_s}{4} = 70 \text{ MHz}. \tag{6.2}$$

At TRlabs an analog SAW (surface acoustic filter) with a bandwidth of 3 MHz was used as a reconstruction filter to attenuate the spectrum images, except the one placed at 70 MHz. Figure 6.7 shows the spectrum of the 256-QAM signal. The designed modulator was tested with the HP4195A spectrum analyzer. In Figure 6.7, the output of the spectrum analyzer shows the power spectrum of the 256-QAM signal, placed at a 70MHz center frequency with ~2.78 MHz absolute bandwidth and about 43-dB sidelobe attenuation, as required in the J.83 standard.

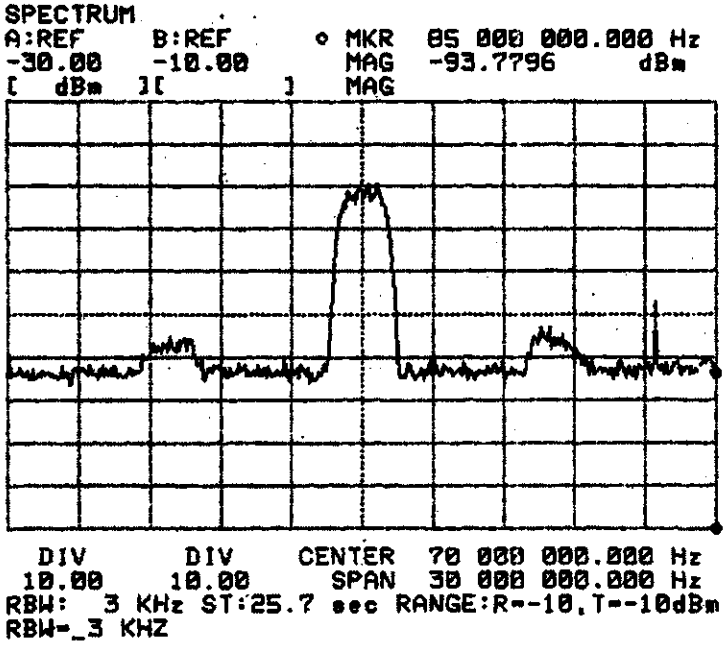


Figure 6.7 Power spectrum of quadrature amplitude modulated signal.

The modulator was then connected to the demodulator back to back. The digital demodulator, shown in Figure 6.8 consists of the frequency downconverter and a pair of matched filters.

The digital frequency downconversion from IF to the baseband was implemented using the CORDIC algorithm in the rotation mode. The CORDIC algorithm uses 14 stages, with shift and sign extension providing a phase resolution of $\sim 0.001^0$. This CORDIC was unfortunately impossible to run iteratively reusing one CORDIC stage for the required sampling clock, clks =16.47 MHz. This rate was achieved for several CORDIC stages by pipelining one after another, i.e. the first CORDIC stage performs the rotation by $\pm 90^0$, the second by $\pm 45^0$, the third by $\pm 26.56^0$ and so on. The pipelining involves separating the stages with registers. Thus, every stage was used as a separate block.

However, this structure of the CORDIC algorithm required ~2000LEs, while the conventional NCO would have required only ~600 LEs. The conclusion is that the CORDIC algorithm is not always the best solution for hardware implementation. The higher the required clock rate, the less it is possible to reuse the same CORDIC stage for different rotations and consequently, the more logical elements would be necessary to apply the pipelining.

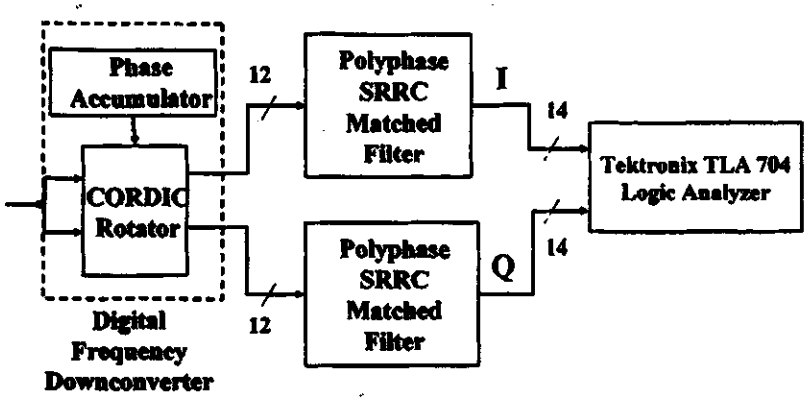


Figure 6.8 Demodulator.

The modem required ~16000 LEs, i.e. ~41% of the available FPGA area. The PLLs, used for the timing clock control, as well as the filters` multipliers were constructed using the Altera library and the megafunctions.

The inphase and quadrature components of the received 256-QAM signal were connected to the Tektronix TLA 704 Logic Analyzer. The output of the logic analyzer is the 256-QAM constellation, shown in Figure 6.9. The result indicates a timing offset problem in the design - the points smear along the *I* axis. In Figure 6.9 one of the points is missing. This point should have represented the all-zero combination, which is prohibited by the PN sequence generator.

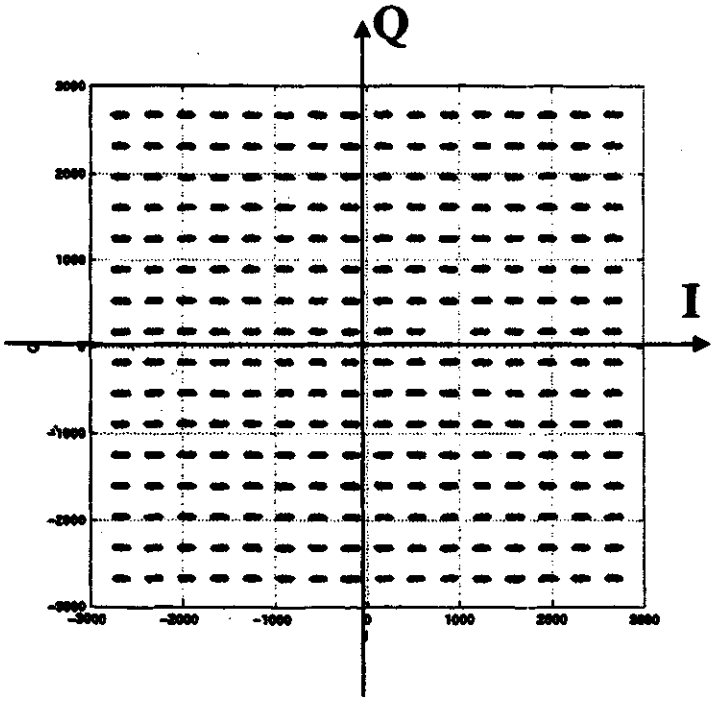


Figure 6.9 256-QAM constellation (logic analyzer output).

6.3 Carrier Recovery Matlab Simulation

The carrier recovery algorithm was simulated in Matlab, assuming a random data source. The results of the Matlab simulation are shown in Figures 6.10 and 6.11. Figure 6.10 illustrates the spinning constellation of the received 256-QAM signal. Figure 6.11 shows the constellation of transmitted (circles) and recovered (points) data. (The diagonals and the maximum radius are also shown to help verify the position of the points in the constellation.)

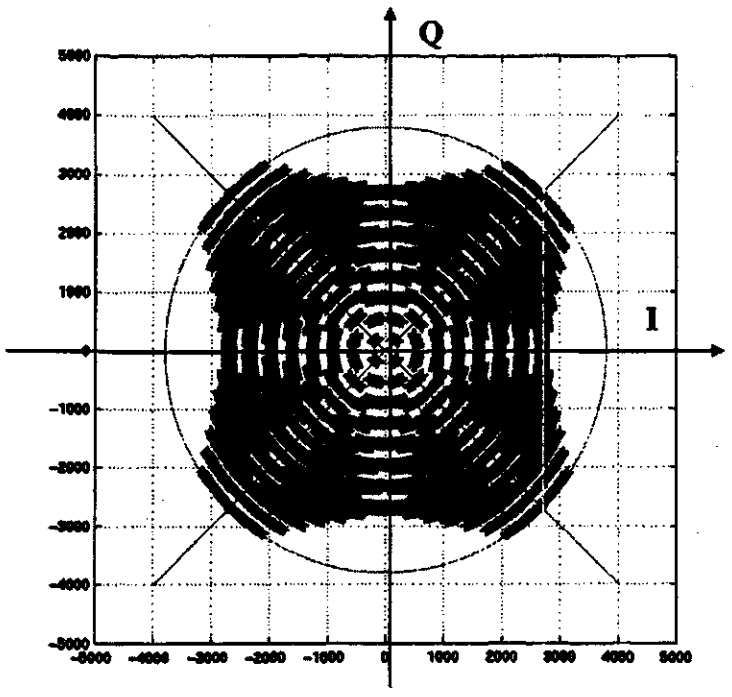


Figure 6.10 Matlab simulation results of 1/100 degrees per symbol frequency offset (recovered data – black dots).

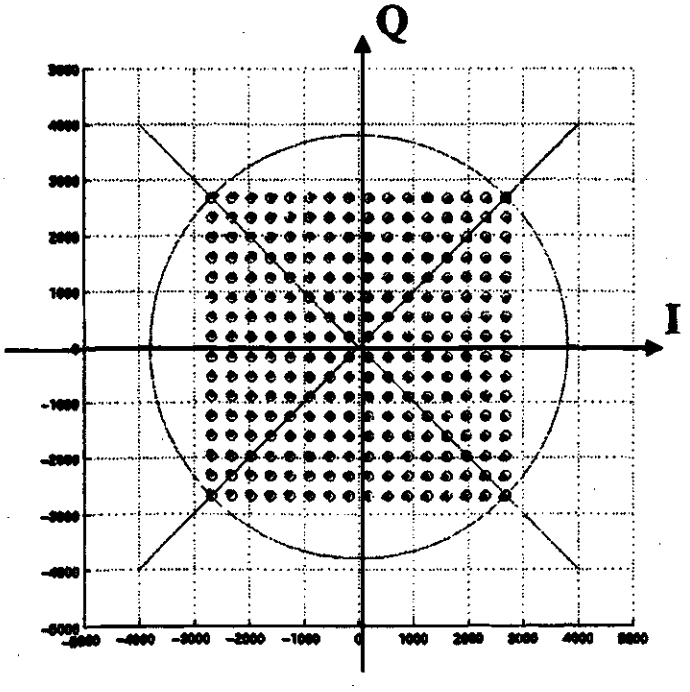


Figure 6.11 Matlab simulation results of 1/100 degrees per symbol frequency offset (sent (circles) and recovered (dots) data).

In this project the RF channel was centered at 70 MHz. In this case using a Matlab simulation, it was empirically established that the highest frequency offset of 50 Hz (~1/100 degrees per symbol) could be recovered using 2 Msymbols. Since the symbol clock was equal to 2.0588 Msymbols per second, the time for carrier recovery to occur could be approximately 1 second. The carrier offset estimate is shown in Figure 6.12.

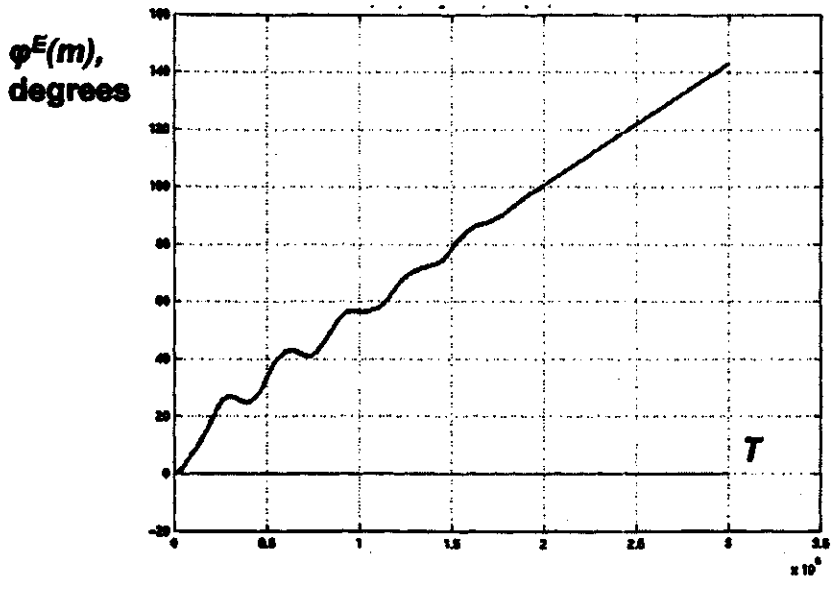


Figure 6.12 Carrier phase offset estimate.

The precision of the local oscillator in this case should be ~1 ppm (or less). A more expensive L.O. usually guarantees a precision of 10 ppm. Thus, using this method of blind carrier recovery for 256-QAM, it requires a very high precision of the L.O., which is not easy to achieve in practice.

For comparison purposes, a QPSK signal, that has only 4 points in the constellation, was tested with the highest frequency offset equal to 4.575 KHz. The testing results are shown in Figure 6.13. Figures 6.13 (a) and (b) show the derotation of the inphase and

quadrature components, while (c) illustrates the derotated points of a QPSK signal. (The circle and the diagonals are shown to help visualize the position of the 4 QPSK points.) Figure 6.13 (d) illustrates the carrier offset rollover at 360 degrees. The established carrier recovery time took ~0.1 Msymbols. In this case the required precision of the L.O. can be ~62 ppm, which is easy to achieve using a cheap local oscillator.

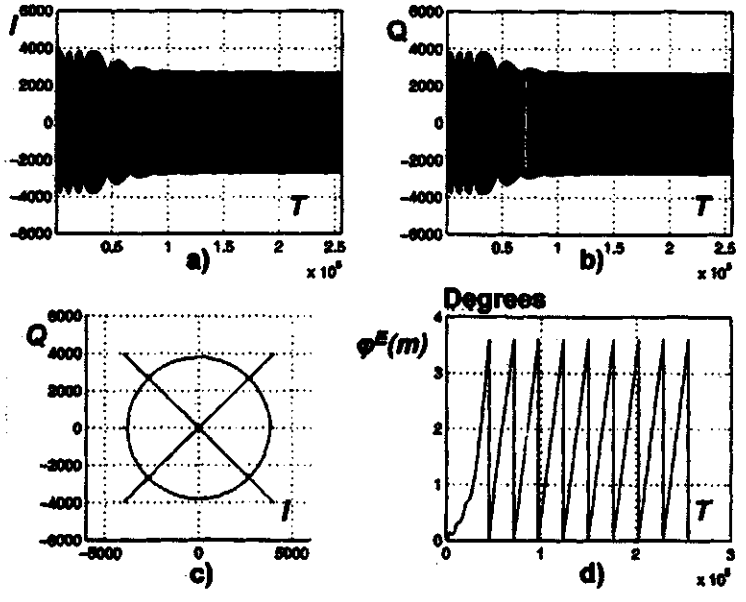


Figure 6.13 Carrier recovery for QPSK signal: a) derotation of *I* component, b) derotation of Q component, c) *I-Q* constellation of derotated components with radius and diagonals, d) carrier offset estimate.

7 CONCLUSIONS AND POSSIBLE FUTURE RESEARCH

This chapter gives a summary of the results obtained in this thesis. Recommendations for potential future research are also discussed.

7.1 Conclusions

The Cable TV industry is moving from the old analog TV to the new digital TV and high speed internet access using a broadband connection. The demand for these high speed digital services is growing steadily. The digital TV set-top boxes and high-speed cable modems are used to meet these demands. The broadband systems are standardized, which gives different companies an opportunity to build compatible products. In order to increase the amount of transmitted data, a 256-QAM modulation is applied in modern cable systems. The 256-QAM modulation used in the cable system is standardized in the J.83 standard.

In this thesis several interesting ideas for the hardware implementation of selected parts of a digital cable modem were proposed. The design optimization for the implementation in FPGAs (Field Programmable Gate Arrays) was one of the prime goals. This was achieved by reducing the number of expensive multipliers.

The conventional implementation of the proposed design would have required 128 multipliers in each of the 4 SRRC filters at the transmitter and the receiver side, 2 multipliers for the digital frequency upconverter and 2 for the digital frequency downconverter. In addition, 4 multipliers would have been necessary for the derotation operation for the carrier recovery, resulting in a total of at least 520 multipliers. The proposed technique of the 256-QAM digital modem implementation using polyphase filters and the CORDIC algorithm includes a total of only 64 multipliers, which are required for the implementation of the multirate SRRC filters. Thus, compared to the conventional design, a savings of approximately 87% is attained for the expensive multipliers.

A solution for the carrier recovery problem was proposed, and the simulation results in Matlab were given. The main advantage of the algorithm is that it can be used for both frequency and phase offset compensation. Applying CORDIC in vectoring and rotation modes reduces the complexity of the DPLL. The expensive multipliers are eliminated from the carrier phase error and radius detection as well as from the derotation. It is a universal algorithm that can be applied not only to 256-QAM but to other QAM or QPSK modulations as well.

Unfortunately, the proposed carrier recovery algorithm could only recover a very small frequency offset after a long time period (up to several Msymbols). If a cable system with an RF frequency of ~70 MHz were used, the required precision of the L.O. for 256-QAM is ~1ppm, which unfortunately could only be achieved using an expensive L.O. The proposed carrier recovery algorithm can be used for the same high rates, assuming

QPSK modulation, or for 256-QAM lower rate communication systems with a maximum frequency of ~1 MHz. In this case the required L.O. precision would be only 50 ppm. A L.O. with a precision of 50 ppm is relatively inexpensive.

Assuming a random data source, the carrier phase error is not updated frequently, thus the carrier phase offset estimate should grow extremely slowly, in order to reach the value of the frequency offset. The delay of acquisition is the main disadvantage of this method. Another concern is that the carrier offset detection and corrections are done at the symbol rate, while the offset is generated at the sampling rate.

The total area of the cable modem design (without DPLL) took approximately 41% of the FPGA chip capacity. Matlab and Quartus II, simulations showed that the CORDIC-based, phase offset detector should have a significant number of stages (14) and bits per stages to support an extremely high resolution of 0.001^0. Based on the area needed for the CORDIC algorithm it is predicted that the required DPLL area would occupy an additional 20% of the FPGA area.

7.2 Future Work

In the proposed carrier recovery algorithm only 4 outer corner points were used for the carrier recovery. Assuming random data, the probability of receiving a corner point among all the possible 256 points is low. In order to increase the carrier offset estimate update time other diagonal points could have been used. In this case the required precision of the L.O. would have been lower. At the same time the required precision of

the CORDIC algorithm would also have been lower. Thus, fewer CORDIC stages would have been required and consequently fewer FPGA logic elements (LEs) would have been used.

The proposed algorithm for the carrier recovery did not assume a data-aided testing sequence, but blind recovery with random data. A research project could be performed in the area of increasing the number of the corner points using preambles. In a cable system a QPSK signal can be used as a preamble. This would, however, reduce the user data throughput. Finally, the training sequence of a combination of known symbols could have been used in order to compare the results. The training sequence would definitely decrease the acquisition time for the DPLL to lock, but the payload would be decreased. These tradeoffs would be interesting for a future investigation.

Timing recovery is one of the problems that was beyond the scope of this thesis. Fractional delay challenges are waiting to be addressed as well. Since it was possible to calculate the radius using the CORDIC algorithm, the AGC and/or equalizer design would also be an interesting addition to the complete digital modem implemented in hardware.

The sidelobe attenuation of the 256-QAM signal spectrum was close to the J.83 standard specification. A tapered window could have been used for better sidelobe attenuation. The proposed polyphase multirate filters could support a maximum sampling frequency clock rate of up to 20 MHz. Higher frequencies were supported only by a filter with a transversal structure using constant coefficient multipliers. The frequency

downconversion was implemented using CORDIC in order to avoid complex multiplications by the sinusoids. However, the area required for the CORDIC was bigger than for the conventional NCO. This project was designed for the Altera Apex 20KE FPGA chip. However, by the end of the project Altera produced a new FPGA Stratix chip, which has embedded complex multipliers. For future research it would be interesting to compare the efficiency of the digital design using embedded complex multipliers.

REFERENCES

[1] P. Stranahan, "Cable Modems." Posted on the Jones Telecommunications & Multimedia Encyclopedia website, Apr. 2001. http://www.digitalcentury.com/encyclo/update/cmodem.html

[2] "Market research report 2000." Posted on the Insight Research Corporation website, Apr. 2001. http://www.insight-corp.com/reports.html

[3] Kendell Musgrove, "A Brief History of Cable Television," Optical Fiber Corning Inc., 2001.

[4] International Telecommunications Union, ITU-T Recommendation J.83: "Digital Multi-programme Systems for Television, Sound and Data Services for Cable Distribution," 1997.

[5] "Field Programmable Gate Arrays (FPGAs) and Enabling Technology." Posted on the Virtual Computer Corporation website, Sep. 2001. http://bard.eu.cornell.edu/downloads

[6] Peter L. Levin & Reinhold Ludwig, "Crossroads for Mixed Signal Chips", *IEEE Spectrum*, vol. 39, pp. 38-43, March 2002.

[7] J. G. Proakis, *Digital Communications*, Toronto: McGraw Hill, 2001.

[8] Leon W. Couch, *Digital and Analog Communication Systems*, Tom Robbins, Prentice Hall, 2001.

[9] N.J. Fliege, *Multirate Digital Signal Processing*, Toronto: John Wiley & Sons, Inc., 1994.

[10] Fred Harris, "Multirate DSP Course, for TRLabs and the University of Saskatchewan," May 1997.

[11] Richard G. Lyons, *Understanding Digital Signal Processing*, Reading, Massachusetts: Addison-Wesley Publishing Company,1996.

[12] Gregory Ray Goslin, "A Guide to Using Field Programmable Gate Arrays (FPGAs) for Application Specific Digital Signal Processing Performance," Xilinx Inc., 1995.

[13] "Multirate FAQ: Interpolation." Posted in the public domain on the dspGuru website, January 2000. http://www.dspguru.com

[14] E. Grayver and B. Daneshrad, "Direct Digital Frequency Synthesis Using a Modified CORDIC," Integrated Circuits and Systems Laboratory, UCLA, Apr. 2002.

[15] J. Volder, "The CORDIC Trigonometric Computing Technique," IRE Trans. Electronic Computing, Vol. EC-8, pp. 330-334, Sept. 1959.

[16] C. Dick and F. Harris, "Configurable Logic Digital Communications it's about time," Xilinx Inc., Jan. 2002.

[17] J.S. Walther, "A Unified Algorithm for Elementary Functions," Proc. AFIPS Spring Joint Computer Conference, pp. 379-385, 1971.

[18] T. Vladimirova and H. Tiggeler, "FPGA Implementation of Sine and Cosine Generators Using the CORDIC Algorithm," Surrey Space Center, University of Surrey, Jan. 2001.

[19] R. Andraka, "A Survey of CORDIC Algorithms for FPGA Based Computers," Andraka Consulting Group Incorporated, 1998.

[20] J. L. Stensby, *Phase-Locked Loops*, CRC Press, New York, 1997.

[21] L. Charles, H. Philips and T. Nagle, *Digital Control System Analysis*, Prentice Hall, New Jersey, 1995.

[22] H. Meyer and G. Ascheid, *Synchronization in Digital Communications*, John Wiley & Sons, Toronto, 1990.

[23] A. V. Oppenheim and R. W. Schafer with J. R. Buck, *Discrete-Time Signal Processing*, New Jersey: Prentice-Hall Inc., 1999.

[24] Y.R. Shayan and T. Le-Ngoc, "All Digital Phase-Locked Loop: Concept Design and Applications," IEE proceedings, Vol. 136, Pt. F. No. 1, pp.53-56. Feb. 1989.

[25] M. D. Ciletti, *Modeling, Synthesis, Rapid Prototyping with the Verilog HDL*, Prentice Hall, Upper Saddle River, 1999.

APPENDIX A. LOOP FILTER

Based on the discussion in Chapter 5 the DPLL can be expressed by two different transfer functions:

$$\begin{cases} \dfrac{\Delta\Phi(z)}{\Phi(z)} = \Psi(z) = \dfrac{1}{1+G(z)} \\ \dfrac{\Phi^E(z)}{\Phi(z)} = H(z) = \dfrac{G(z)}{1+G(z)} \end{cases} \tag{A.1}$$

In order to compensate the carrier offset, the transfer function $\Psi(z)$ has to gradually eliminate the carrier phase error, $\Delta\varphi(m)$, while the transfer function $H(z)$ must force the carrier offset estimate, $\varphi^E(m)$, to sequentially track the carrier offset, $\varphi(m)$.

A.1 Loop Filter for Phase Offset Compensation

A slightly less than ideal situation, where only the phase offset exists and no frequency offset is considered, is discussed first. Since the phase offset has a constant value and it is causal, i.e. $\varphi(m) = \varphi_{const}$ for $m \geq 0$ and zero for m < 0, it can be represented as a step function u(m). The z-transform of the step function is equal to [23]:

$$\Phi(z) = \Phi_{const}(z) = \frac{1}{1-z^{-1}} \tag{A.2}$$

The simplest loop filter for the DPLL is the first order Infinite Impulse Response (IIR) filter, which has the following transfer function:

$$G(z) = \frac{kz^{-1}}{1-z^{-1}} \tag{A.3}$$

Therefore, substituting G(z) in equation (5.8) (repeated in (A.1)) with the expression in (A.3), the loop transfer function is given as:

$$\begin{aligned}\Psi(z) &= \frac{1}{1+G(z)} \\ &= \frac{1}{1+\dfrac{kz^{-1}}{1-z^{-1}}} \\ &= \frac{1-z^{-1}}{1-(1-k)z^{-1}}\end{aligned} \tag{A.4}$$

Here k must take values (0, 2) to ensure the poles are inside the unit circle, which is the criterion for stability.

The z-transform of the carrier phase error, $\Delta\varphi(m)$, is given as a product of the transfer function $\Psi(z)$ and the z-transform of the carrier offset $\Phi(z)$:

$$\begin{aligned}\Delta\Phi(z) &= \Phi(z)\cdot\Psi(z) \\ &= \frac{1}{1-z^{-1}}\cdot\frac{1-z^{-1}}{1-(1-k)z^{-1}} \\ &= \frac{1}{1-(1-k)z^{-1}}\end{aligned} \tag{A.5}$$

In order to calculate the residual carrier phase error, $\Delta\varphi(m)$, the final value theorem of the z-transform is applied [21, 23]:

$$\begin{aligned}\lim_{m\to\infty}\Delta\varphi(m) &= \lim_{z\to 1}\left(\frac{1-z^{-1}}{z^{-1}}\right)\cdot\Delta\Phi(z) \\ &= \lim_{z\to 1}\left(\frac{1-z^{-1}}{z^{-1}}\right)\cdot\frac{1}{1-(1-k)z^{-1}} \\ &= 0\end{aligned} \tag{A.6}$$

Therefore, using the first order IIR causal filter it is possible to perform carrier recovery in the presence of phase offset.

A.2 Loop Filter for Frequency Offset Compensation

Assuming that $\varphi_{frequency}(m) = \Delta\omega T \cdot m$ for $m \geq 0$ and zero for m < 0, the frequency offset can be represented as a ramp function with the scope of 2π. Therefore, the z-transform of the carrier offset, considering that frequency offset, is given as a ramp function [21, 23]:

$$\Phi(z) = \frac{\Delta\omega T z^{-1}}{\left(1-z^{-1}\right)^2} \tag{A.7}$$

The z-transform of the carrier phase error, $\Delta\varphi(m)$, assuming the frequency offset, $\varphi_{frequency}(m)$, is given as the product of the transfer function $\Psi(z)$ and the z-transform of the carrier offset $\Phi(z)$:

$$\begin{aligned}\Delta\Phi(z) &= \Phi(z)\cdot\Psi(z) \\ &= \frac{\Delta\omega T z^{-1}}{(1-z^{-1})^2}\cdot\frac{1-z^{-1}}{1-(1-k)z^{-1}} \\ &= \frac{\Delta\omega T z^{-1}}{(1-z^{-1})}\cdot\frac{1}{1-(1-k)z^{-1}}\end{aligned} \tag{A.8}$$

The residual carrier phase error $\Delta\varphi(m)$ for the frequency offset $\varphi_{frequency}(m)$ is found by applying the final value theorem [21]:

$$\begin{aligned}\lim_{m\to\infty}\Delta\varphi(m) &= \lim_{z\to 1}\left(\frac{1-z^{-1}}{z^{-1}}\right)\cdot\Delta\Phi(z) \\ &= \lim_{z\to 1}\left(\frac{1-z^{-1}}{z^{-1}}\right)\cdot\frac{\Delta\omega T z^{-1}}{\left(1-z^{-1}\right)}\cdot\frac{1}{1-(1-k)z^{-1}} \\ &= \lim_{z\to 1}\frac{\Delta\omega T z^{-1}}{1-(1-k)z^{-1}} \\ &= \frac{\Delta\omega T}{k}\end{aligned} \tag{A.9}$$

The residual carrier phase error in equation (A.9) is not equal to zero, but rather a constant that is proportional to the frequency offset. The first order IIR loop filter (a single integrator with a unit delay) is not sufficient for the carrier recovery in the presence of the frequency offset.

The reason why the first order loop IIR filter is not sufficient for the frequency recovery, can also be observed from the z-transform of the frequency offset (A.7) and the first order loop filter (A.3): there are two poles in the $\Phi(z)$, while there is only one pole in $G(z)$. Therefore, in order to compensate the second pole in $\Phi(z)$ by prediction, the second pole should exist in the loop filter transfer function $G(z)$. In order to have two poles in the filter transfer function, it may be represented not as a single integrator but as a double integrator design [24].

The transfer function of the double integrator shown in Figure 5.15 is derived as follows:

$$Out(z) = \left(Out(z) + In(z)k_1 + In(z) \cdot \frac{k_2 z^{-1}}{1 - z^{-1}} \right) z^{-1} \tag{A.10}$$

Then, after several mathematical manipulations of equation (A.10):

$$Out(z)\left(1 - z^{-1}\right) = In(z)\left(\frac{k_1 z^{-1} - k_1 z^{-2} + k_2 z^{-2}}{1 - z^{-1}} \right) \tag{A.11}$$

Finally, the transfer function of the loop filter is given as:

$$G(z)=\frac{Out(z)}{In(z)}$$
$$=\frac{k_1 z^{-1}+(k_2-k_1)z^{-2}}{(1-z^{-1})^2} \qquad (A.12)$$
or
$$=\left(\frac{k_2 z^{-1}}{1-z^{-1}}+k_1\right)\cdot\frac{z^{-1}}{1-z^{-1}}$$

Inserting $G(z)$ from (A.12) into (5.8) (or A.1), the loop transfer function is expressed as:

$$\Psi(z)=\frac{(1-z^{-1})^2}{1-(2-k_1)z^{-1}+(1+k_2-k_1)z^{-2}} \qquad (A.13)$$

Therefore, assuming equation (A.2) in case of the phase offset only, the z-transform of the carrier phase error is expressed as:

$$\Delta\Phi(z)=\Phi(z)\cdot\Psi(z)$$
$$=\frac{1}{1-z^{-1}}\cdot\frac{(1-z^{-1})^2}{1-(2-k_1)z^{-1}+(1+k_2-k_1)z^{-2}} \qquad (A.14)$$
$$=\frac{(1-z^{-1})}{1-(2-k_1)z^{-1}+(1+k_2-k_1)z^{-2}}$$

Similar to the derivations done above, the residual carrier phase error, $\Delta\varphi(m)$, is found by applying the final value theorem of the z-transform [21]:

$$\lim_{m\to\infty}\Delta\varphi(n)=\lim_{z\to 1}\left(\frac{1-z^{-1}}{z^{-1}}\right)\cdot\Delta\Phi(z)$$
$$=\lim_{z\to 1}\left(\frac{1-z^{-1}}{z^{-1}}\right)\cdot\frac{(1-z^{-1})}{1-(2-k_1)z^{-1}+(1+k_2-k_1)z^{-2}} \qquad (A.15)$$
$$=0$$

Assuming the z-transform of the carrier phase error, given in the case of the frequency offset derived in (A.7), the z-transform of the phase error is expressed as:

$$\begin{aligned}\Delta\Phi(z) &= \Phi(z)\cdot\Psi(z) \\ &= \frac{\Delta\omega T z^{-1}}{(1-z^{-1})^2}\cdot\frac{(1-z^{-1})^2}{1-(2-k_1)z^{-1}+(1+k_2-k_1)z^{-2}} \\ &= \frac{\Delta\omega T z^{-1}}{1-(2-k_1)z^{-1}+(1+k_2-k_1)z^{-2}}\end{aligned} \tag{A.16}$$

Finally, the residual carrier phase error, $\Delta\varphi(m)$, for the frequency offset, $\varphi_{frequency}(m)$, is calculated by applying the final value theorem, assuming the second order IIR loop filter [21]:

$$\begin{aligned}\lim_{m\to\infty}\Delta\varphi(m) &= \lim_{z\to 1}\left(\frac{1-z^{-1}}{z^{-1}}\right)\cdot\Delta\Phi(z) \\ &= \lim_{z\to 1}\left(\frac{1-z^{-1}}{z^{-1}}\right)\cdot\frac{\Delta\omega T z^{-1}}{1-(2-k_1)z^{-1}+(1+k_2-k_1)z^{-2}} \\ &= 0\end{aligned} \tag{A.17}$$

Therefore, by using the second order IIR loop filter, the final value theorem proves that the steady-state carrier phase error approaches zero, for both frequency and phase offset.

APPENDIX B. MATLAB, SIMULINK AND VERILOG HDL FILES

This appendix contains the list of the Matlab, Simulink and Verilog HDL files that are stored on the attached CD.

Matlab files:

- start.m and start1.m generate a random data sequence.
- polytafacos.m, intera16.m and differa16.m contain Matlab codes to design a SRRC filter with a polyphase structure and organize the coefficients for the proper use in the shaping and matched filters.
- sc_maker.m generates a sequence of 1s, 0s and -1s for the frequency upconverter.

Simulink file:

- aaa16cor_april360_19CCZoldie.mdl contains the design of the entire modem with a CORDIC-based frequency downconverter, shaping and matched filters with polyphase structure and the CORDIC-based DPLL.

Verilog HDL files:

- tx16_la_new.v is the top level Verilog HDL code of the modem that also includes: a) the PN sequence generator, the SP converter and a data mapping block (module mapping), b) shaping filter (module baran32), c) matched filter (module differenciator2) d) clock control (modules pll2a and ppl4) and e) the CORDIC-based frequency downconverter (module cordic_nco).
- corulez.v contains Verilog HDL code for the DPLL.