AN EFFICIENT FPGA IMPLEMENTATION OF A CONSTANT MODULUS ALGORITHM EQUALIZER FOR WIRELESS TELEMETRY

Thesis

Submitted to

The School of Engineering of the

UNIVERSITY OF DAYTON

In Partial Fulfillment of the Requirements for

The Degree of

Master of Science in Electrical Engineering

By

Robert G. Schumacher

UNIVERSITY OF DAYTON

Dayton, Ohio

December, 2014

AN EFFICIENT FPGA IMPLEMENTATION OF A CONSTANT MODULUS ALGORITHM EQUALIZER FOR WIRELESS TELEMETRY

Name: Schumacher, Robert G.

APPROVED BY:

Eric Balster, Ph.D.
Advisor Committee Chairman
Professor, Department of Electrical and Computer Engineering

Guru Subramanyam, Ph.D.
Committee Member
Professor and Chairperson, Department of Electrical and Computer Engineering

Russell Hardie, Ph.D.
Committee Member
Professor, Department of Electrical and Computer Engineering

John G. Weber, Ph.D.
Associate Dean
School of Engineering

Eddy M. Rojas, Ph.D., M.A., P.E.
Dean, School of Engineering

ABSTRACT

AN EFFICIENT FPGA IMPLEMENTATION OF A CONSTANT MODULUS ALGORITHM EQUALIZER FOR WIRELESS TELEMETRY

Name: Schumacher, Robert G.
University of Dayton

Advisor: Dr. Eric Balster

In this manuscript, a real-time Field Programmable Gate Array (FPGA) implementation for a Constant Modulus Algorithm Equalizer is presented. In many wireless telemetry applications, the presence of multipath in the channel can obscure the intended message. One approach to reduce transmission errors is the application of equalization, with the objective of restoring the received information to an estimate of its original form, prior to demodulation. In situations where no *a priori* knowledge of the transmission exists, Constant Modulus Algorithm equalizers may be applied, leveraging only the constraint that the ideal transmitted signal exhibits a substantially constant amplitude. The application of the Constant Modulus Algorithm in an FPGA to high bit rate telemetry signals is analyzed, developed and tested. The research and development activity shows that the approach is practical to improve over-the-air bit error rate performance in airborne telemetry applications.

For Sherry

ACKNOWLEDGMENTS

I would like to express my appreciation to Dr. Eric Balster for his assistance and patience to make this thesis a reality. Also, my thanks to Dr. Russel Hardie, for his guidance in digital signal processing and review of the thesis material. And Dr. Guru Subrumanyam, for his gentle persuasion in getting me to come back to school, and time to review the material.

Mr. Terry Hill and Mr. Mark Geoghegan are instrumental in my education in digital signal processing and its application to telemetry. They have decades of combined experience in digital signal processing as it relates to communications. Mr. Geohegan spent many hours reviewing this manuscript and offering assistance in its development. Mr. Hill's generous flexibility regarding my work schedule and use of company resources has been incredibly valuable in obtaining a Master's Degree. Mark and Terry are both architects of telemetry technology and the key contributors to the design and development of the equalization approach discussed in this work.

And finally, a special appreciation to my family, especially my wife Sherry, for their continuous support of my career and return to school.

TABLE OF CONTENTS

LIST OF FIGURES

DEFINITIONS AND ABBREVIATIONS

AGC automatic gain control

AWGN additive white gaussian noise

BPSK binary phase shift keying

CMA constant modulus algorithm

DSP digital signal processing

FIR finite impulse response

Fmax maximum operating frequency as analyzed by the Altera compiler

IF Intermediate Frequency

IIR infinite impulse response

kHz kilohertz

LMS least mean squares

LTI linear time invariant

Mbps mega bits per second (10^6 bits per second)

MHz megahertz

MMIC monolithic microwave integrated circuit

Msym/sec mega symbols per second

NCO numerically controlled oscillator

PCM/FM pulse coded modulation, frequency modulation

ppm parts per million

PRNG pseudo-random noise generator

QPSK quadrature phase shift keying

RAM random access memory

RF radio frequency

ROM read only memory

SOQPSK shaped offset phase shift keying

VHDL VHSIC hardware description language

VHSIC very high speed integrated circuit

ZF zero forcing

CHAPTER I

INTRODUCTION

1.1 Background

In the transmission of wireless information, the effects of multipath are well documented, but persist as a major impediment to data transmission[1]. In analog video or voice communications, though multipath often produces unwanted artifacts, it may not render the signal unusable, for example, the ghosts in analog NTSC video. However in digital communications, especially the military telemetry market, it is generally necessary to obtain a nearly error-free link, and the presence of multipath makes this impossible without sufficiently restoring the original signal. The need to support higher data rates tends to aggravate the issue, as delays can span several symbols. One approach, which applies an equalization filter to correct the linear distortion caused by multipath, is presented here.

The application of equalization to repair multipath impaired channels has been studied and analyzed for many years[2][3]. Some wired systems, e.g.ethernet, have utilized equalization for some time now to correct distortions caused by the wired connection. Ample research has been presented on the topic, and many demonstrations of systems employing equalization have been shown[4]. Real-time processing has been successfully applied to many lower frequency applications, but the implementation of an equalizer to process samples of high data rate telemetry receivers is limited.

The advent of high density Field Programmable Gate Arrays with digital signal processing capability provides a platform for implementation of equalization approaches described in the literature. In this paper, we present a well-known approach first described by Godard [5]. This method, referred to as Constant Modulus Algorithm, leverages the fact that the transmitted waveform is a constant amplitude. The cases presented are continuous phase modulation, with constant envelope. The noteworthy advantage of the Constant Modulus Algorithm is that it can be applied to what is referred to as "blind" applications, where the receiver has no *a priori* knowledge of the data transmitted, and must therefore determine an equalization solution without this information[6][7].

1.2 Scope

One application of the equalization methods described is telemetry for the application of gathering range data from airborne test articles. The ranges are often over sea water, or mountainous areas, and the landings and takeoff areas are surrounded by many large buildings. This application, known as airborne telemetry, is a market served by a number of vendors who produce transmitters and receivers. To date, demonstration of receivers with working equalizers has been very limited, with inconsistent documented success in field applications.

In this paper, a method to implement a working Constant Modulus Algorithm equalizer is presented. It is noteworthy that the implementation will be tested in real telemetry applications, and therefore the design must be product ready, which is a higher standard than a typical academic experimental application. The Constant Modulus Algorithm is created in MATLAB Simulink, and then converted to VHDL using MATLAB's HDL Coder. The VHDL is compiled in Altera's Quartus tool, and downloaded to an Altera Stratix IV device. The target Altera device is installed in a receiver developed and produced by Quasonix, LLC. The receiver provides telemetry performance in L, S and C bands for data rates up to 46 Mbps. The waveforms supported are those specified in IRIG106, defined as Tier 0 (PCM/FM), Tier 1 (SOQPSK), and Tier 2 (MHCPM)[8].

A description of multipath and multipath scenarios is presented. The multipath analysis is performed assuming that the channel behaves as a linear, time-invariant (LTI) system over a suitable period of time. Through this analysis method, it is shown that the channel behaves like a linear filter, and can be analyzed in either time or frequency domains using LTI methods. A description of equalization approaches is presented, with background on equalizer types and performance comparisons. The CMA equalizer is presented in mathematical and structural detail, and simulation data are presented.

Finite precision digital signal processing methods pertinent to this implementation are presented with the design for this application. As the architecture requires the resources a large FPGA, a discussion of efforts to optimally utilize the available resources, and approaches to ensure timing compliance with high bit rate applications is presented. The architecture for the tested system is presented in detail, and associated support circuitry. Finally, test results are presented for the design, including eye diagrams, BER analysis and acquisition results.

CHAPTER II

THE MULTIPATH ISSUE

2.1 Multipath Overview

Multipath is generally described as a wireless communications phenomenon whereby the signal transmitted from an antenna arrives at the receiving antenna through a multiplicity of paths. These paths may have different attenuations, and also may exhibit different time delays. The classical example is the case where the line of sight (shortest) path is joined by a single longer delay path, for example, a reflection off of a mountain or building. The diagram in Figure 2.1 shows this case. The channel shown is not static, i.e. time-variant, as the aircraft is moving. However, at a given instant in time, the channel can be treated as static. This static case can be described mathematically using

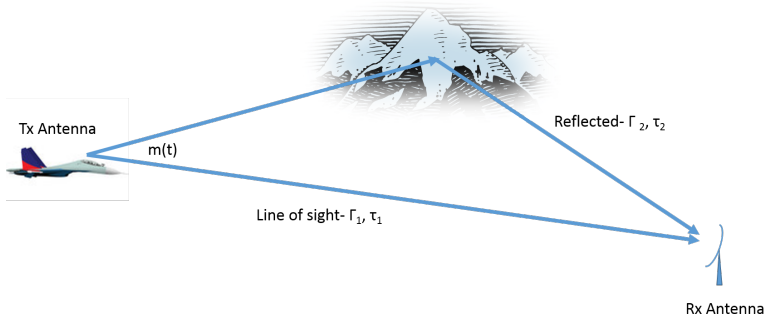

Figure 2.1: Multipath Example

$$r(t) = \Gamma_1 m(t - \tau_1) + \Gamma_2 m(t - \tau_2). \tag{2.1}$$

The simple relationship can be generalized by extending the channel to include an arbitrary number of alternate paths, each with different attenuations and delays. Further, for mobile, or non-static scenarios, each path may exhibit time-variance with respect to the attenuation and delay. This generalized form can be expressed as follows:

$$h(t) = \sum_{i=0}^{L-1} \Gamma_i(t)\delta(t - \tau_i(t)) \tag{2.2}$$

While this time-variant relationship is an effective way to fully describe a multipath channel, it is difficult to analyze. We will therefore make some observations about the system to simplify its analysis.

2.2 Linear Time Invariance

Although equation 2.2 clearly exhibits time-varying behavior in the channel attenuation and path delay as the object moves, we observe that the time-variance is slow with respect to the data rate over the information channel. For example, in a common telemetry application, the data rate through the information channel is 10Mbps, with a symbol rate of 5MSym/sec (2 bits per symbol). This corresponds to a symbol period of 200ns. For this example scenario, the receiving antenna is fixed, and the transmitting antenna is attached to an airborne platform moving less than mach 1 (761 mph). Even when the aircraft is flying directly at the receiving antenna at mach 1, in one bit period the aircraft has moved only 6.8e-5 meters. With operation in L-band (1500MHz), the resulting phase shift is 0.0049 degrees, or less than one-half degree over 100 bits. Treating this channel as an LTI case is acceptable, and convenient, though an equalizer to address the multipath must adapt at a rate high enough to track the changing multipath. In most cases, the test article

is moving much slower, and in fact, some of the most severe multipath is experienced when the aircraft is taxiing, on take-off, or landing. Therefore, over a finite, significant period of time, the channel can be viewed as stationary. In mobile applications such as airborne telemetry, we treat the channel as a series of static channels. This property is leveraged such that the computational benefits of linear, time-invariance can be utilized. Therefore, equation 2.2 is reduced to

$$h(t) = \sum_{i=0}^{L-1} \Gamma_i \delta(t - \tau_i). \tag{2.3}$$

It is this reduced form that we use for analysis, and treat the response as linear and time invariant. Although the telemetry application and lab environment includes the changing path attenuation and delay, test data confirms that these are slow enough that we can use LTI techniques.

2.3 A Multipath Example

The impact of summing multiple paths may not be intuitive. The simplest case, where multiple paths have the same delay, will simply result in positive reinforcement of the signal. If one of the two paths includes a 180 degree phase shift, the signal may cancel. Taking this thought experiment further, consider multiple paths with different delays, but with an unmodulated carrier. The summation of multiple paths at the receiver for this case would result in vector addition of the signal with different phases. The vector summation results in a vector modified in amplitude and phase. Therefore, a carrier described by

$$m(t) = Acos(2\pi f_c t), \tag{2.4}$$

simply becomes a copy of the unmodulated carrier, with a different phase and amplitude, or

$$r(t) = \Gamma_r Acos(2\pi f_c t + \phi_r). \tag{2.5}$$

with Γ_r and ϕ_r resulting from vector addition of the multiple paths.

This case is valid for unmodulated (zero bandwidth) transmissions only. However, since an unmodulated carrier is of virtually no communications value, we must address a case of a carrier with modulation. For this case, the transmission is represented by

$$m(t) = A(t)cos(2\pi f_c t + \phi(t)). \tag{2.6}$$

Information is carried by modulating either the value of $A(t)$ or $\phi(t)$ or both. However, in the multipath channel, combination of delayed versions of the transmitted signal results in intersymbol interference, which can make direct demodulation of the signal difficult or impossible. For example, take a simple case to illustrate the issue. A binary phase shift keyed (BPSK) signal is transmitted, where a carrier phase of +90 degrees represents a digital "1" ($Acos(2\pi f_c t + \pi/2)$ and a carrier phase of -90 degrees represents a digital "0" ($Acos(2\pi f_c t - \pi/2)$)). The multipath channel is composed of the incident ray, a second ray of $1/2$ amplitude with a delay of exactly one bit time, and a third ray of $1/2$ amplitude with a delay of exactly 2 bit times. The impulse response of the channel is shown in Figure 2.2, with a sample rate of 4 times per bit (Channel impulse response will be discussed in Section 2.4.1). Suppose further that the transmission sequence was the pattern '00010011010100101'. The baseband transmission in Figure 2.3 results. The bit states are marked for each transmission state. Ignoring the states of the first two received symbols, the third symbol received is "0", with reflected rays with one and two bits delay, respectively, giving a received signal of

$$r(3T) = A_r(Acos(2\pi f_c 3T - \pi/2) + \frac{1}{2}Acos(2\pi f_c 2T - \pi/2 + \frac{1}{2}Acos(2\pi f_c T - \pi/2)). \tag{2.7}$$

yielding

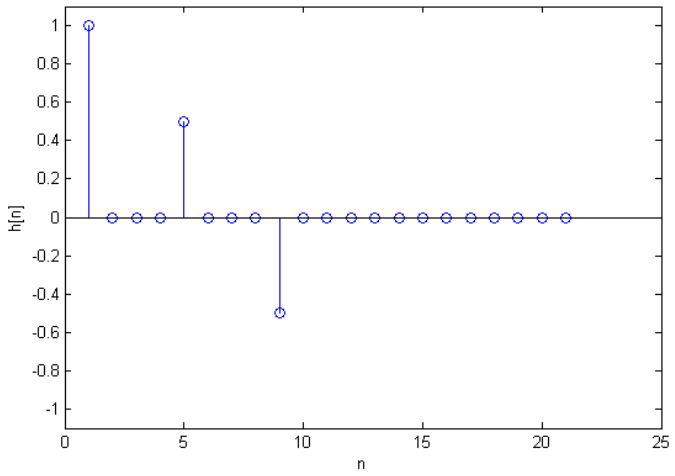

Figure 2.2: Example channel impulse response.

$$r(3T) = A_r(2Acos(2\pi f_c 3T - \pi/2)). \tag{2.8}$$

The following symbol received is "1", again with reflected rays of one and two bits delay, giving

$$r(4T) = A_r(Acos(2\pi f_c 4T + \pi/2) + \frac{1}{2}Acos(2\pi f_c 3T - \pi/2 + \frac{1}{2}Acos(2\pi f_c 2T - \pi/2)). \tag{2.9}$$

yielding

$$r(4T) = A_r(0) = 0. \tag{2.10}$$

In this very simple example, the multipath has the effect of doubling the bit magnitude for patterns 000 and 111, providing the correct bit magnitude for patterns 011, 101, 010 and 100, and completely canceling the information for patterns 001 and 110. The received baseband signal is presented in Figure 2.4. The detector in the receiver/demodulator will score a positive in-phase

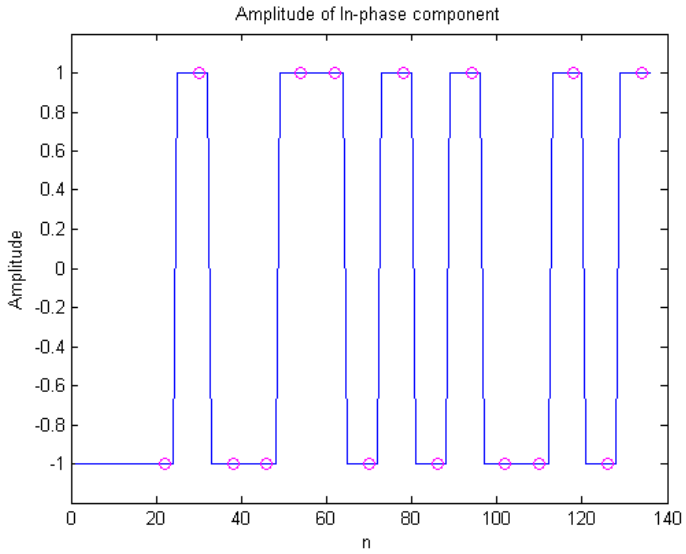

Figure 2.3: Example BPSK transmission (baseband).

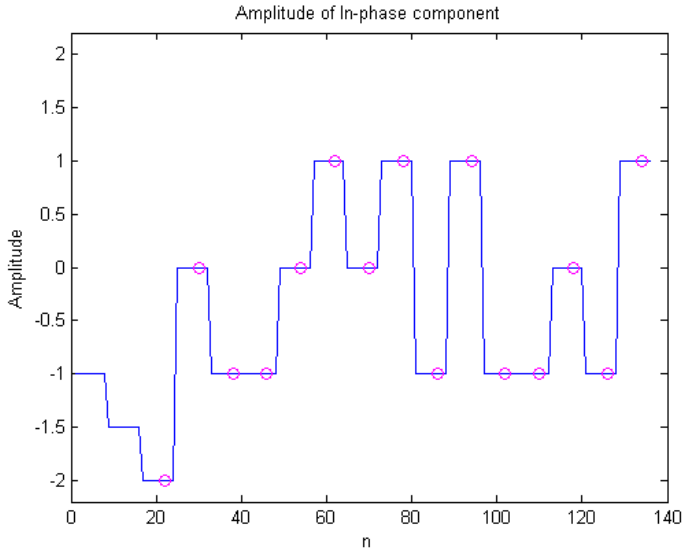

Figure 2.4: Example BPSK signal received after multipath channel (baseband).

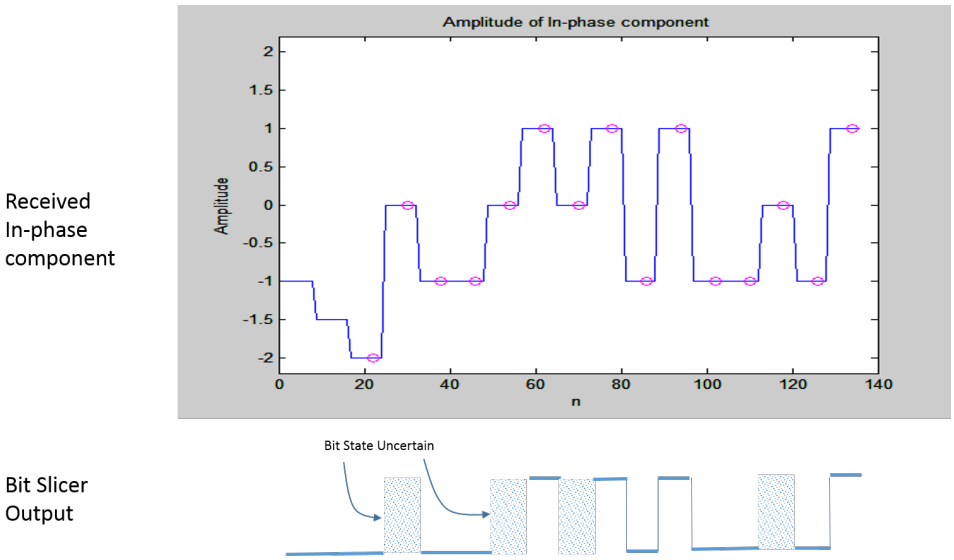

Figure 2.5: Detector output for BPSK signal demodulated after multipath channel (baseband).

value as a binary one, a negative in-phase value as a binary zero, and phase near zero is uncertain. The demodulator output shown in Figure 2.5 will produce errors at the uncertain regions in the recovered in-phase signal. Though this example is contrived, it demonstrates the deleterious effect of multipath, and further, a case where the most sophisticated equalization cannot recover lost information[1].

2.4 Channel Response

As we have established the basis for application of LTI analysis methods, the multipath channel response can be described as a linear system. The channel can be equivalently described in either time domain or frequency domain terms. In the time domain, the path is characterized by its impulse response, whereas in the frequency domain, the frequency response is used. These responses are complex valued.

[1]This is true in a single symbol detection system. A multisymbol detector may be be able to reconstruct the original sequence by observing the pattern in neighboring symbols.

2.4.1 Impulse Response

The most straightforward method to observe the effect of multipath is to characterize the impulse response of the channel. In digital signal processing, the discrete response of a system is given by the linear constant coefficient difference equation (LCCDE). For a finite impulse response filter (FIR) this is

$$y[n] = \sum_{m=0}^{M-1} b_m x[n-m]. \tag{2.11}$$

In this representation, the coefficients, b_m, form the impulse response. Clearly, a discrete representation of the multipath channel can be directly inferred if the path attenuations and delays are known. Since $y[n]$ is complex, the coefficients, b_m, are complex. Generally, it is convenient to choose the sample rate for 2.11 as a multiple of the information symbol rate. Although delays are continuous in time (and also dispersive), a sample rate of 4 times the symbol rate will yield an effective analysis. Choosing a sample rate as a multiple of the symbol rate is convenient for analyzing the waveform, and is coincidentally the approach used in the receiver.

As an example, consider a channel with a line of sight attenuation of 110 dB, and propagation delay of 16.0 microseconds. A second path has an attenuation of 113 dB, with a delay of 16.1 microseconds, and a third path has attenuation 117.5 dB and delay 16.4 microseconds. Since the channel description for multipath can be completely described using relative relationships for attenuation and delay, it is convenient to adjust the values for the three paths to

Path	Loss(dB)	Delay(μs)
1	0	0
2	3.0	0.1
3	7.5	0.4

Though this may seem to be a complete characterization, the transmission frequency wavelength is very short in comparison to multipath delays. For a particular carrier frequency, the path characteristic for a given delay results in a phase shift[2] given by

$$\phi_{path} = 2\pi f_c \tau. \tag{2.12}$$

which may be reduced to the interval $[-\pi, \pi]$ using

$$\phi_{path} = \phi_{path} - round(\frac{\phi_{path}}{2\pi})2\pi. \tag{2.13}$$

When the signal is translated to baseband, the resulting phase must be preserved. As a result, it is common practice to include a phase with the path description, or alternatively, the loss may be represented using a complex value. Assigning phases to the paths described aboved based on an arbitrary carrier frequency of 1485.5MHz, the table describing the channel becomes

Path	Loss(dB) α	Delay(μs) τ	Phase(radians) θ
1	0	0	0
2	-3	0.1	-2.827
3	-7.5	0.4	1.256

Using these values, the equation in 2.3 becomes

$$h(t) = 1 + 10^{\frac{\alpha_2}{20}}\delta(t-\tau_2)e^{j\theta_2} + 10^{\frac{\alpha_3}{20}}\delta(t-\tau_3)e^{j\theta_3}. \tag{2.14}$$

Suppose that the communications system is operated at a symbol rate of 5MSPS (200ns period), and that the received waveform was sampled at 4 times the symbol rate (sampling period is 50 nanoseconds). The equivalent sampled result given $x[n]$ transmitted through the multipath channel is given by

[2]The path may impart a constant phase change, although this is generally simply a 180 degree inversion due to a reflection off of a surface.

$$y[n] = x[n] + 10^{\frac{\alpha_2}{20}} x[n-2]e^{j\theta_2} + 10^{\frac{\alpha_3}{20}} x[n-8]e^{j\theta_3}. \tag{2.15}$$

And therefore,

$$h[n] = \delta[n] + 10^{\frac{\alpha_2}{20}} \delta[n-2]e^{j\theta_2} + 10^{\frac{\alpha_3}{20}} \delta[n-8]e^{j\theta_3}. \tag{2.16}$$

The stem plot in Figure 2.6 shows the system impulse response, clearly showing the 3 discrete paths. The stem plots are shown using magnitude and phase plots as the response is complex. It is common to focus on cases where the incident, or line of sight path, which by definition the shortest delay, is also the strongest. However, it is entirely possible, especially in mountainous areas or areas with structures, to have the shortest delay path be weaker in amplitude. This would occur, for example, when the line of sight path was blocked or heavily attenuated. The characterization above applies, but note that τ should be made zero for the shortest path, and the path losses adjusted accordingly.

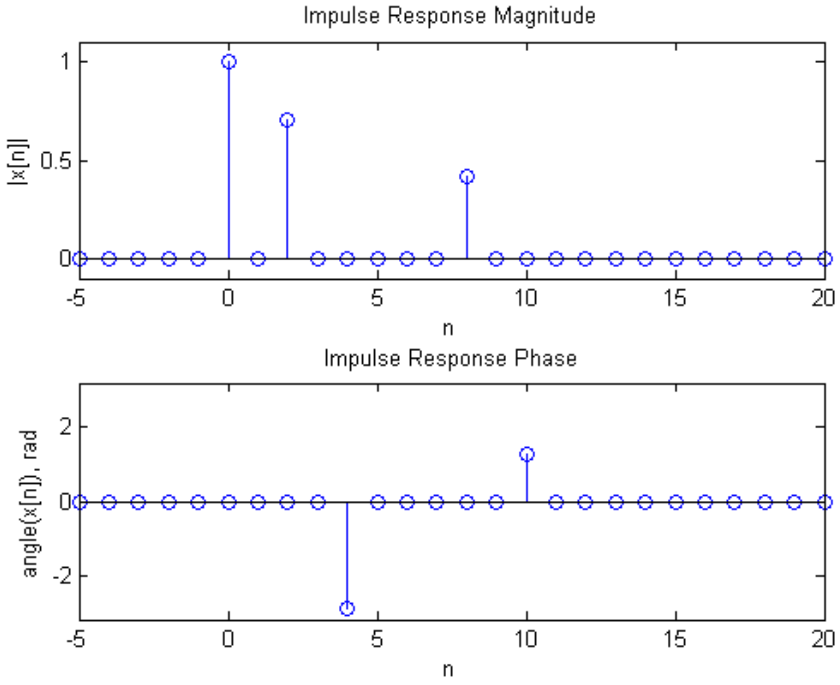

Figure 2.6: Multipath Example– Channel Impulse Response

2.4.2 Frequency Response

An equivalent description of the multipath channel is given by the channel frequency response. The response must be complex, in order to describe the delays in the channel. This familiar way of measuring a system response is similar to the response calculated for a linear filter. In fact, the LTI channel behaves like a linear filter and, as we shall discuss later, is normalized by equalization using the inverse filter response. The response in Figure 2.7 is the frequency response of the channel given by the impulse response in Figure 2.6. Note that the phase response in Figure 2.7 is not a straight line, indicating that the response is not a constant delay for all frequencies.

The frequency response also provides some intuition into the difficulty to equalize the channel. For example, consider the case with a substantially long second path with little attenuation relative to the incident, shown in Figure 2.8. This forms a frequency response with a deep notch. The

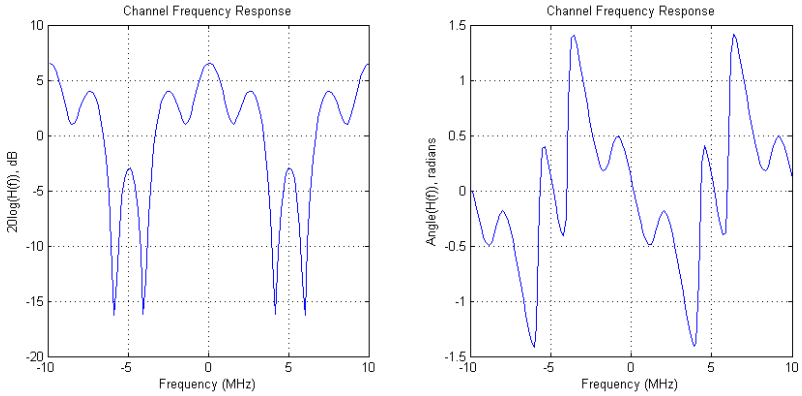

Figure 2.7: Multipath Example– Baseband Frequency Response

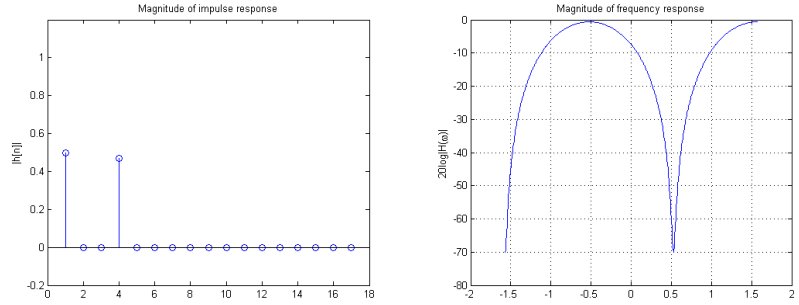

Figure 2.8: Multipath Example– Impulse and frequency response for a deep notch.

notch in the frequency response removes the energy at that frequency, and intuitively, the inverse filter will have nearly infinite gain at this frequency. Even if this were practical, the noise in the communications channel would most likely conceal the energy in the notch, and the gain would cause excessive amplification of the noise at the equalizer output.

2.5 Multipath Models

In order to simulate and test the performance of the equalizer implemented, it is necessary to develop multipath models. Generally, test equipment that simulates a multipath environment will

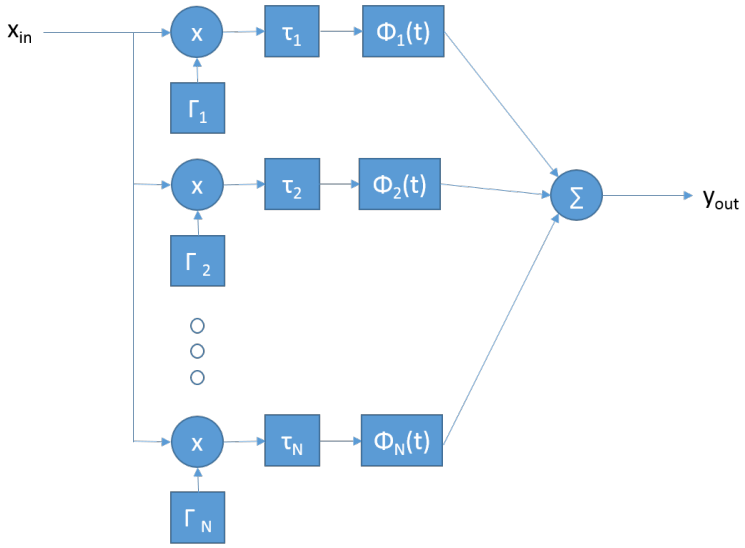

Figure 2.9: Multipath Example– Multipath Test System

have a fixed number of ”rays” that represent the fundamental and delayed path components of a multipath scenario. The diagram in Figure 2.9 shows the general layout for a multipath test emulator. This test equipment is manufactured by a number of vendors, such as Agilent, and is also included as a capability with the Quasonix Receiver Test set, pictured in Figure 2.10. This equipment was used to characterize the performance of the receiver/demodulator used in this development.

Although static multipath is appropriate for analysis and test of the equalizer as an LTI system, the true measure of an *adaptive* equalizer is using a dynamic channel. This is done by varying w, τ, and ϕ in Figure 2.9. Often, test equipment will have a setting for each path, *doppler frequency*, that modulates the angle, ϕ, at the set frequency. This parameter simulates a mobile target, such as an airborne test article, as it traverses many wavelengths of the carrier frequency over a short period of time. Simulations, such as those built in MATLAB, use the same mathematical representation for multipath. In fact, it is convenient to construct a model in MATLAB using identical parameters as the test system, to ease the transition to the laboratory.

Figure 2.10: Photo of the Quasonix Receiver Test set, containing multipath emulation capability. Photo courtesy of Quasonix, LLC.

CHAPTER III

EQUALIZATION TECHNIQUES

3.1 Equalization

The implementation of an equalizer requires two main elements. The first is a method to determine the channel characteristic, either directly or indirectly. The second is to develop a transfer function (e.g. an inverse filter to the channel transfer function) to modify the received waveform to allow the demodulator to recover the intended data. Many implementations have been proposed and tested, either by simulation or in hardware[9]. In some implementations, the two processes are combined, for example, in a gradient search algorithm, where a cost function drives the shape of the transfer function iteratively.

The first part, determining the channel characteristic, can exist in various forms, but most algorithms focus on either characterizing the frequency or impulse response of the channel, which is then converted to an inverse channel filter, or direct determination of the inverse filter. An example of this approach is a system that utilizes a training sequence to estimate the inverse channel response using a Wiener solution. The second part of the equalizer implementation is the creation of a transfer function to attempt to restore the information signal. Again, various implementations have been proposed and constructed. Probably the most popular is a finite impulse response (FIR) filter that utilizes variable tap weights to construct the inverse filter. Another approach uses a frequency domain implementation that applies a filter to the frequency domain representation of a signal. These

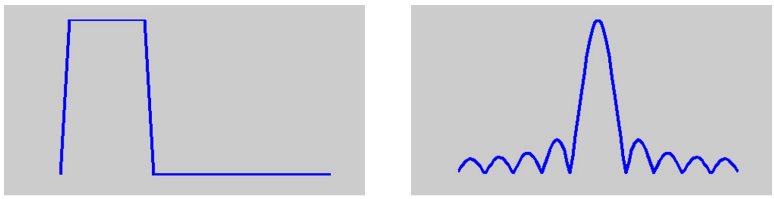

Figure 3.1: Response of rectangular filter in frequency domain

filters approximately invert the channel response, in an effort to obtain the original signal. The multipath signal is applied to the filter, and at the equalizer output, the signal should resemble the original transmitted signal.

If we consider that the distortion imparted by the channel is convolution, or linear application of an impulse response to the signal being transmitted over the channel, the equalizer is performing deconvolution. In most cases, complete deconvolution is not possible even if the exact channel impulse response is known. For example, a common filter applied in signal processing is rectangular in the time domain, given by the response

$$y[n] = \frac{1}{N} \sum_{k=0}^{N-1} x[n-k]. \tag{3.1}$$

This rectangular response in the time domain can be converted to a frequency domain response as shown in Figure 3.1. The frequency response has zeros at frequencies related to the inverse of the filter length. The information at frequencies where zeros exist cannot be recovered, even with infinite gain.

Although perfect deconvolution is normally unachievable, solutions generally exist that are acceptable, or more importantly, useful. While data transmission systems strive for error free communications, unlike analog systems, they are often relatively robust in the presence of noise and

distortion. Effectively, a signal distorted by multipath is not required to be restored to original condition, as long as the data can be recovered. This will be demonstrated in later sections, as eye diagrams from equalized signal are presented.

3.2 Data Aided versus Blind Equalizers

The method to determine either the channel or the inverse channel can be broadly grouped into two approaches, data-aided and blind[10]. The data-aided implementation requires some sort of known transmission, which is received after being distorted by the channel. Various approaches are then utilized by comparing the received, or observed, signal against a local copy in the receiver to determine the channel response or directly determine an inverse filter. This leads ultimately to a channel inverse transfer function that can then be applied to equalize the received signal. The blind equalization approach is far more difficult in that there is no *a priori* knowledge at the receiver of the intended transmission. The equalizer must attempt to create the inverse channel transfer function by leveraging hypotheses of the most likely transmission One of example of this is termed decision feedback, whereby the best guess at the intended state of the digital symbol is made, and fed back to compare against the transmitted signal. The comparison yields a cost, which is used to update the transfer function. The focus of the work in this manuscript is an approach that is called the Constant Modulus Algorithm. This algorithm uses a cost function based on the distance of the received waveform amplitude from a constant radius. The Constant Modulus Algorithm approach is a gradient search approach, closely related to a Minimum Least Squares algorithm.

In the process of equalization of a communications channel, the task of identifying either the channel characteristic, or the direct solution to the impaired channel is the most daunting. The application of an LTI solution thereafter is straightforward (albeit requiring complex mathematics). A number of techniques will be presented in this section.

3.3 Simulation Environment

In this chapter, we will compare a number of standard equalizer implementations. These implementations are simulated in MATLAB, and results provided graphically. Note that it is not the intent to quantitatively compare performance, rather to build a reference backdrop for the CMA implementation synthesized in the FPGA. First, we establish a stimulus for the equalization techniques, using a PCM/FM waveform at 5Mbps. The waveform is transmitted at high signal to noise and convolved with a static multipath case. This simulation environment is described in more detail below, and will be used for the comparison of a number of equalizer implementations.

3.3.1 Transmission waveform

The application of equalizers to range telemetry, as described in the introduction, provides an opportunity rich environment for test scenarios. The telemetry standard, IRIG106[8], defines three primary waveforms for telemetry communications. The oldest and still the most widely used today, PCM/FM (pulse coded modulation, frequency modulation), is an excellent case to demonstrate the effects of multipath, and the benefits of equalization, as the the eye pattern may be visually observed.

The PCM/FM transmitter takes a serial stream of bits as its input, and outputs a frequency based on the bit state. The IRIG standard defines PCM/FM parameters, including a peak deviation of 0.35 times the bit rate. This specifies that the frequency of a mark (binary 1) will be 0.35 times the bit rate above frequency, and a space (binary 0) 0.35 times the bit rate below the center frequency. The modulator shapes the bit transitions using a premodulation filter as shown in Figure 3.2, in order to maintain a reasonable transmission bandwidth. This has the effect of smoothing the transitions between symbols.

When the signal is received at the receiver, the demodulator functions to recover the binary information. Many demodulators have been proposed and implemented, but for our purposes we will consider a simple FM discriminator based on change in angle of the complex signal. Since

$$f = \frac{\Delta\phi}{\Delta t}, \tag{3.2}$$

we can measure the change in complex phase on each new sample to determine frequency. In the digital signal processing implementation, the arctangent of the complex signal is computed, unwrapped, and compared to the previous sample, giving

$$\phi[n] = unwrap(atan(y[n])), \tag{3.3}$$

and,

$$f[n] = \frac{\phi[n] - \phi[n-1]}{T_s}. \tag{3.4}$$

The magnitude of the discriminator output is independent of signal amplitude, as it is scaled by the modulation index of the transmitted waveform. However, as the apparent modulation index of the receiver is impacted by multipath and other intersymbol interference, it is necessary to maintain headroom in the discriminator output.

The IRIG106 standard for PCM/FM prescribes a modulation index of 0.7, which corresponds to a peak deviation of ± 0.35 times the bit rate. Since we are sampling at 4 times the bit rate in the receiver/demodulator, $f_s = 4f_b$, and $f_{dev} = 0.35 f_b$, yielding

$$f_{dev} = \frac{0.35}{4} f_s, \tag{3.5}$$

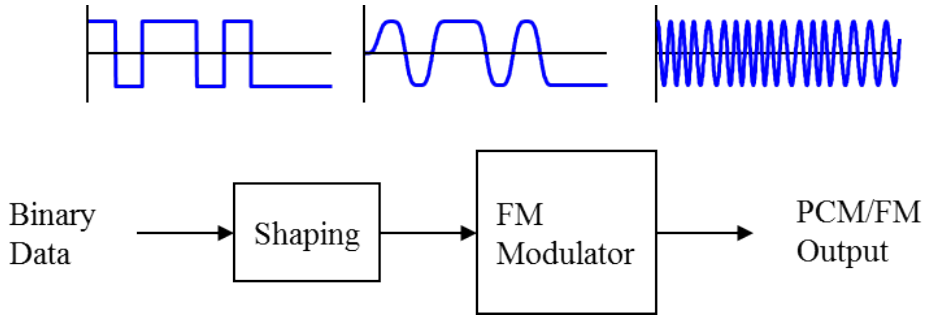

Figure 3.2: Notional block diagram of a PCM/FM modulator. The equivalent system is done mathematically in a DSP implementation.

meaning that the complex phase of the signal will advance or diminish 0.175π radians on each sample. Because this type of discriminator develops its output on each sample, and does not consider information from adjacent symbols, it is called a single symbol detector. The block diagram in Figure 3.3 shows the notional block diagram for the discriminator. Note that the output from the discriminator are binary values (0, 1), but the frequency versus time signal is recovered as an intermediate step as shown in Figure 3.3. This information is plotted for a number of symbols on a persistent graph. This is known as the eye diagram, or eye pattern, as the persistent collection of demodulated symbols resembles an eye. In general, the more open the ”eye”, the easier it is to detect the binary information.

The eye pattern of the PCM/FM signal is observed after the received signal has been demodulated by an FM discriminator. The transmission, when demodulated without noise, multipath or any other impairments will produce an eye diagram as shown in Figure 3.4. Note the sharp zero crossings, and open eye.

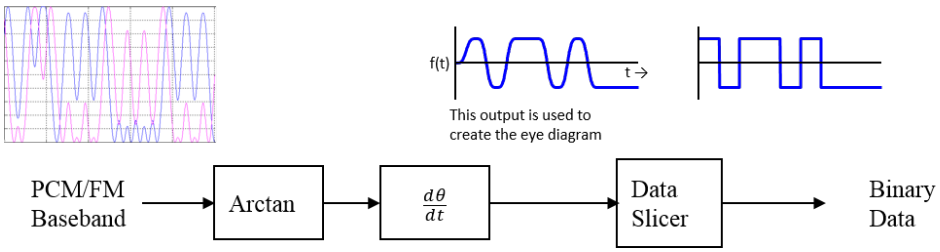

Figure 3.3: Notional block diagram of a PCM/FM discriminator. The equivalent system is done mathematically in a DSP implementation.

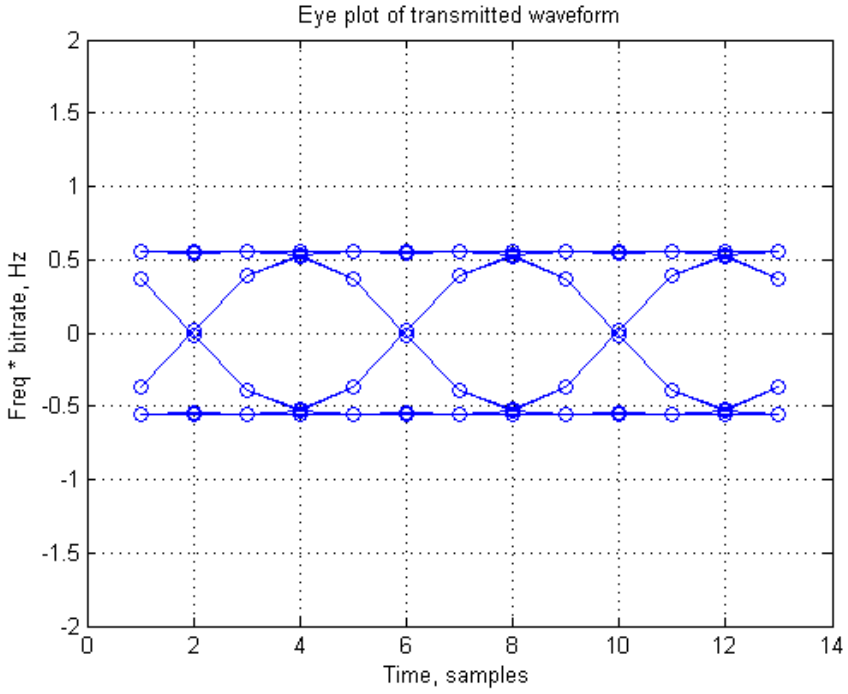

Figure 3.4: Eye diagram of an ideal PCM/FM signal.

3.3.2 Multipath Channel

Next, we create a simulation channel. The analyses presented will attempt to show convergence of the various equalization approaches, so a static channel is used. The multipath channel is a 3-ray scenario, with the channel response given by

$$h(t) = \delta(t) + 0.7\delta(t - \frac{5}{f_s})e^{-j\pi/2} + 0.1\delta(t - \frac{8}{f_s})e^{-j\pi/4}. \tag{3.6}$$

The response in Equation 3.6 is normalized such that the main (incident) path is a magnitude of 1, with no delay. The additional paths have magnitudes of 0.7 and 0.1, delays of 5 samples and 8 samples, and phase shifts of $\frac{\pi}{2}$ and $\frac{-\pi}{4}$, respectively. In discrete terms, Equation 3.6 is written as

$$h[n] = \delta[n] + 0.7\delta[n - 5]e^{-j\pi/2} + 0.1\delta[n - 8]e^{-j\pi/4}. \tag{3.7}$$

The stem plots for this response are given in Figure 3.5. The stem plots are shown as real/imaginary, and magnitude/phase, which are equivalent representations.

This impulse response represents a static multipath case with a strong reflection at $1\frac{1}{4}$ symbols, and a weaker reflection at 2 symbols. The impulse response can be transformed to the frequency domain equivalent shown in Figure 3.6.

When the transmitted signal is convolved with the multipath channel described in Equation 3.6, with the impulse response in Figure 3.5, delayed versions of the signal are summed with the incident signal, producing an eye diagram like that in Figure 3.7. In this diagram, we can observe the time domain distortion caused by the multipath channel, and we can clearly see that a single symbol detector, such as the simple FM discriminator in Figure 3.3, would be unable to correctly recover the information bits.

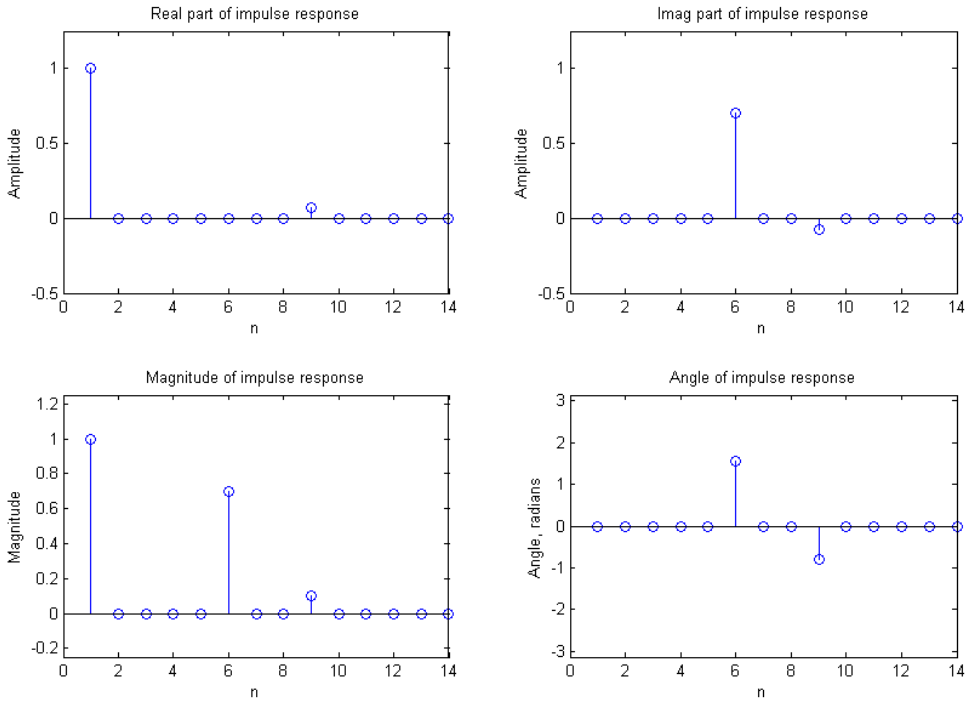

Figure 3.5: Example channel impulse response. The upper stem plots show the impulse response in real/imaginary format. The lower stem plots show the impulse response in magnitude/phase format.

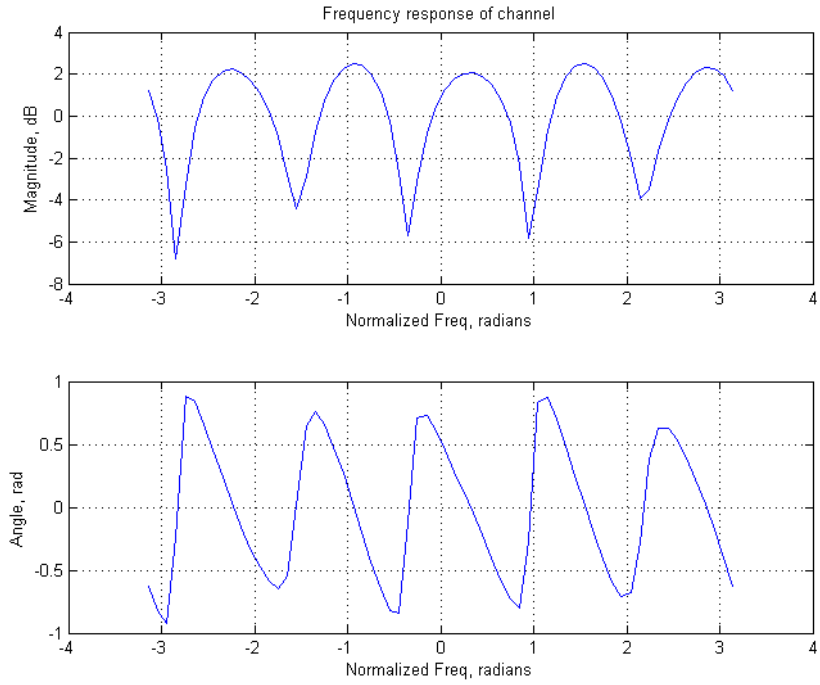

Figure 3.6: Example channel frequency response.

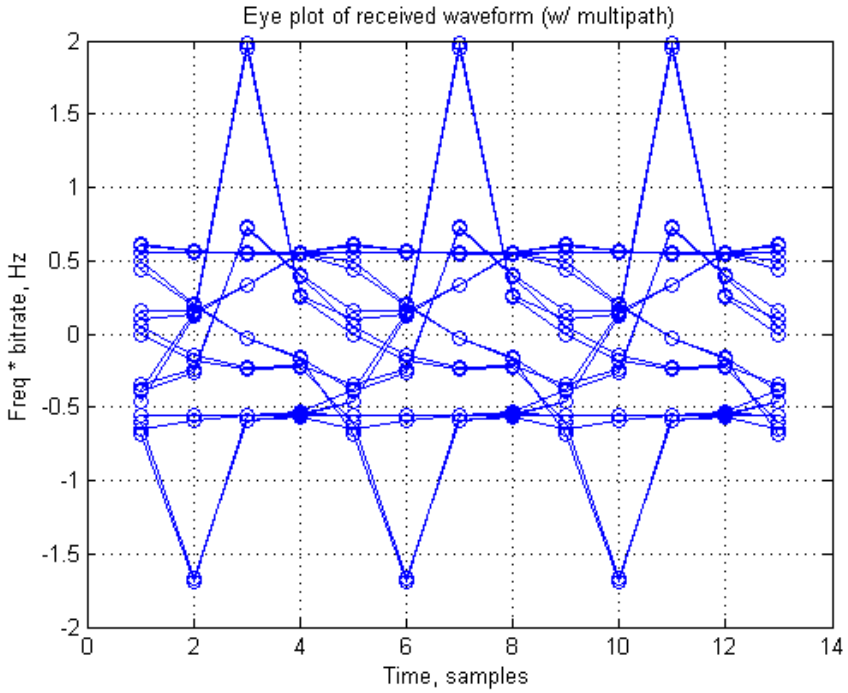

Figure 3.7: Eye diagram of an PCM/FM signal after convolution with a multipath channel.

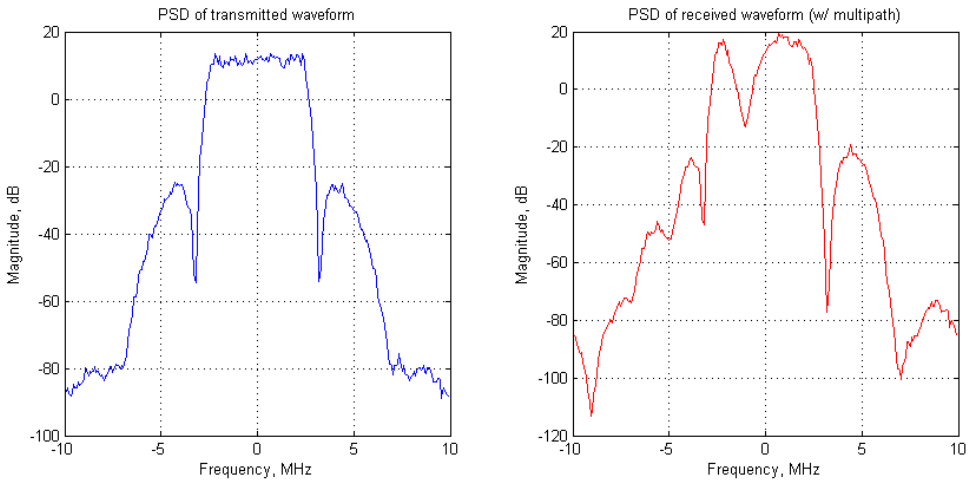

Figure 3.8: PSD of an ideal PCM/FM signal, and received signal after multipath channel is applied.

In addition, the signal can be observed in the frequency domain. While this may not provide the intuitive assessment of distortion that the time domain affords, it demonstrates the loss of energy at some frequencies that complicates the inversion filter, and in cases renders complete recovery impossible. Figure 3.8 show power spectral density (PSD) plots of the transmitted signal and the received signal after convolution with the multipath channel. Note how the frequency response in Figure 3.6 results in the PSD at the right (convolution in the time domain is multiplication in the frequency domain).

3.4 Data-aided Equalizers

In the design of data-aided equalizers, a sequence of known bits must be transmitted, so that when the sequence is received, the data can be used to ”train” the equalizer. Using one of several methods, the equalizer uses this information to determine the equalization filter characteristics. In an application where the multipath is fixed, such as a building-to-building link, the training may be accomplished during setup of the system, where the equalizer taps will be permanently established. However, in the airborne telemetry application, the target platform is mobile, and as a result, the

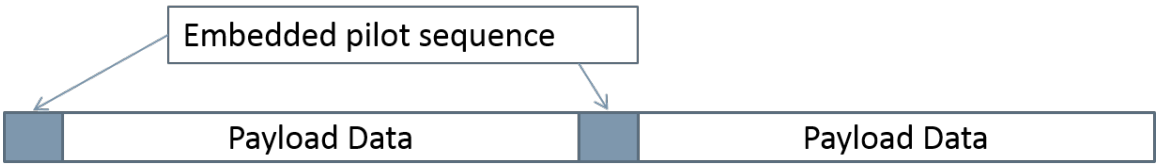

Figure 3.9: Pilot sequence inserted in block formatted transmission.

multipath characteristic is dynamic. In this application, a method must be utilized to continuously update the equalizer settings. In most data-aided implementations, this is done by inserting a number of ”pilot” bits occasionally, so that the equalizer may adjust its settings. Clearly, the pilot bits must occur at a rate sufficiently high enough that the equalizer can adapt to the multipath channel.

In Figure 3.9, a block based transmission begins each block with a fixed pilot sequence. The remaining bits in the block are ”payload” bits, used to transmit telemetry data. The pilot bits are chosen such that their position in the received signal can be extracted using a correlator. The correlator operates by locating a maximum response when the complex samples are correlated against a local copy of the known pilot sequence, modulated to form a complex baseband representation of the transmitted signal. The correlator function

$$C[n] = \sum_{k=0}^{L-1} S_{rx}[n-k] S_{pilot}[L-k], \tag{3.8}$$

must provide the ability to locate and extract the pilot in the received data with low signal to noise ratio, with frequency offsets, and in the presence of impairments such as multipath. The design of effective correlators is a science unto itself and will not be covered in detail here. After the pilot sequence position has been determined, the samples comprising the pilot sequence are extracted, and used in one of the methods described below to determine the equalizer response.

3.4.1 Zero Forcing Equalizer

The data-aided equalizer approach known as zero forcing (ZF) uses the known transmission characteristic to estimate the channel impulse response, which is then inverted to create the equalizer filter. This approach was proposed by Robert Lucky [11], and in a mathematical sense, the zero forcing filter is the ideal solution for equalization. However, the zero forcing approach suffers in practical applications for the following reasons:

1. Due to zeros in the frequency response of the channel, the inverse filter may exhibit very high gain at these frequencies. In a practical implementation, this results in processing overflows, excessive noise due to the high gain, and some frequencies that are not invertible (i.e. divide by zero).

2. Although the channel impulse response may be finite, the impulse response of the inversion is generally infinite. Truncation of the impulse response in any practical implementation results in an imperfect solution. While no implementation results in a perfect solution, approaches that utilize a cost function that minimizes the error may provide a better solution.

3. The "known" data must have sufficient frequency content to provide information at all frequencies within the data bandwidth. Frequencies that are not well represented in the frequency response of the known data may cause erroneous gains in the inversion filter.

To demonstrate the zero forcing approach, we utilize the stimulus described in section 3.3.1. There are two methods of computing the transversal filter tap weights. The first method utilizes the frequency response of the channel as shown in in Figure 3.6. Procedurally, we extend the complex impulse response, $\mathbf{h}$, to N samples, where N is the number of taps we intend to use on the transversal filter. Then,

$$\mathbf{H} = FFT(\mathbf{h}). \tag{3.9}$$

The inverse of $\mathbf{H}$ is found by inverting each frequency point in $\mathbf{H}$,

$$\mathbf{H}_{inv} = \frac{1}{\mathbf{H}}. \tag{3.10}$$

The inversion yields the response shown in Figure 3.10. Note the high gain peaks which attempt to compensate the low gain in the channel response notches. We then convert the inverse filter response, $\mathbf{H_{inv}}$ to its impulse response in the time domain,

$$\mathbf{h}_{inv} = IFFT(\mathbf{H}_{inv}). \tag{3.11}$$

The complex values in $\mathbf{h}_{inv}$ are mapped directly to the equalizer weights, $\mathbf{w}$. The impulse response is presented in Figure 3.11 and Figure 3.12. This impulse response maps directly to the equalization filter tap weights. The impulse response spans the entire set of filter taps, but the magnitude has decayed considerably at the final tap. This would suggest that the truncation of the infinite response will not have a large effect on the result. In a later section, we will discuss the impact of the filter length on the solution.

The received signal is then convolved with the impulse response of the zero forcing filter. The result is the eye diagram shown in Figure 3.13. The filter does a nearly perfect job of opening the eye, and should result in error free reception of the original signal.

An alternative method for finding the coefficients operates entirely in the time domain, and gives the process its name[12]. Effectively, we are seeking a set of coefficients, $\mathbf{c}$, to force the convolution of $\mathbf{c}$ and $\mathbf{h}$ to the delta function, δ. Therefore,

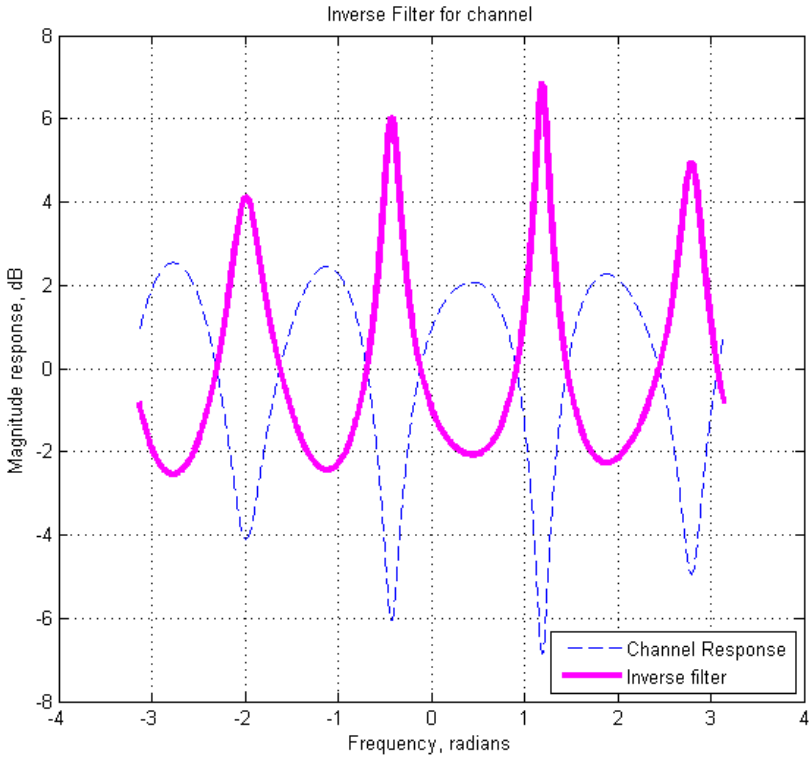

Figure 3.10: Zero forcing filter response, with channel response shown for reference.

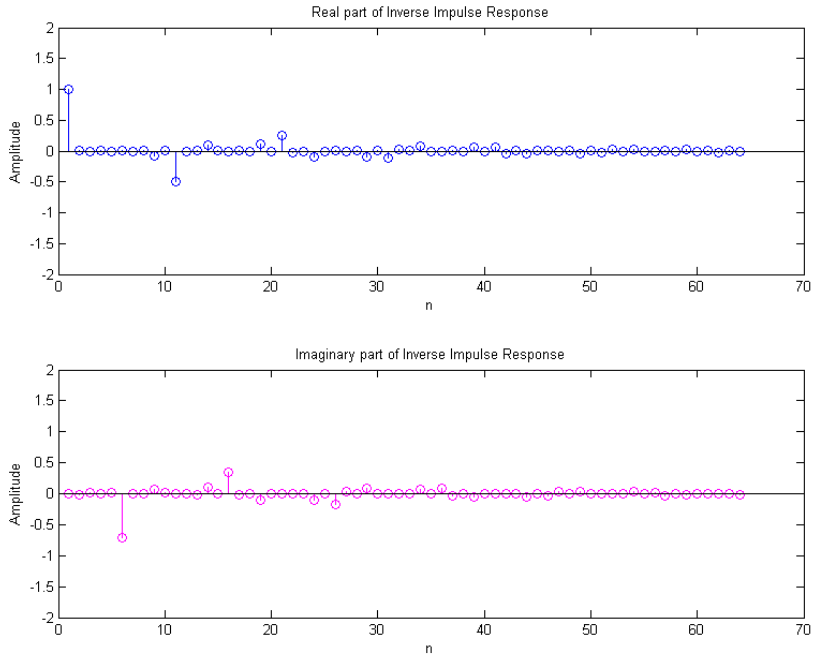

Figure 3.11: Zero forcing impulse response showing the real and imaginary parts of the complex response.

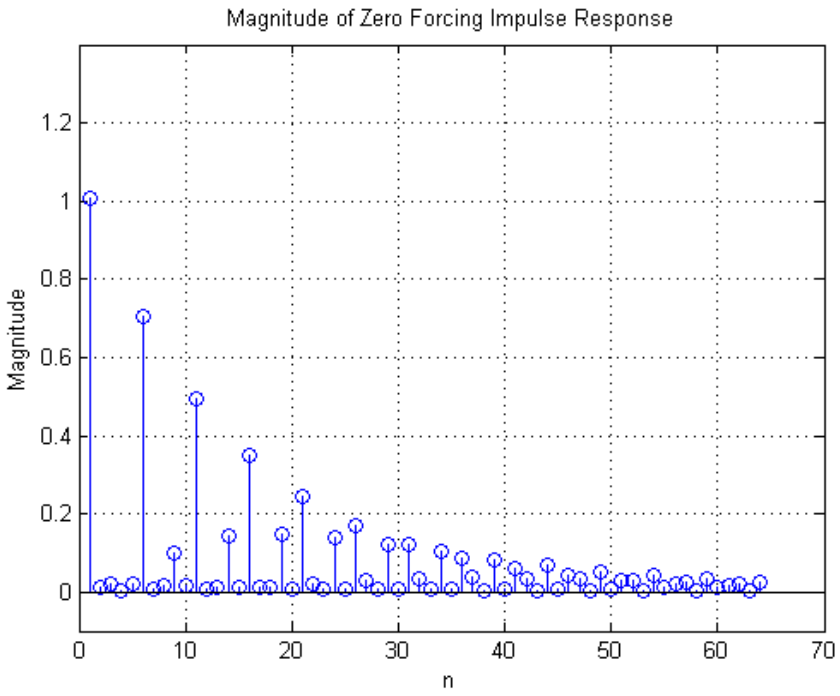

Figure 3.12: Zero forcing impulse response. The top two plots show the real and imaginary parts, and the third shows the magnitudes of the taps.

$$\mathbf{c} * \mathbf{h} = \delta. \tag{3.12}$$

Writing this in matrix form,

$$\begin{bmatrix} h_1 & h_2 & h_3 & ... & h_N \\ 0 & h_1 & h_2 & ... & h_{N-1} \\ 0 & 0 & h_1 & ... & h_{N-2} \\ ... & ... & ... & ... & ... \\ 0 & 0 & 0 & ... & h_1 \end{bmatrix} \begin{bmatrix} c_1 \\ c_2 \\ c_3 \\ ... \\ c_N \end{bmatrix} = \begin{bmatrix} 0 \\ ... \\ 1 \\ ... \\ 0 \end{bmatrix} \tag{3.13}$$

and solving for $\mathbf{c}$,

$$\begin{bmatrix} c_1 \\ c_2 \\ c_3 \\ ... \\ c_N \end{bmatrix} = \begin{bmatrix} h_1 & h_2 & h_3 & ... & h_N \\ 0 & h_1 & h_2 & ... & h_{N-1} \\ 0 & 0 & h_1 & ... & h_{N-2} \\ ... & ... & ... & ... & ... \\ 0 & 0 & 0 & ... & h_1 \end{bmatrix}^{-1} \begin{bmatrix} 0 \\ ... \\ 1 \\ ... \\ 0 \end{bmatrix}. \tag{3.14}$$

The inversion in Equation 3.14 is not trivial, especially in an FPGA implementation. After solving for $\mathbf{c}$, these complex values are directly mapped to the equalizer weights, $\mathbf{w}$, and will produce an eye diagram nearly identical to that in Figure 3.13.

Note that in the development of the equations in this section, it was assumed that the channel impulse response, $\mathbf{h}$, was known. In a data-aided implementation, either the frequency or impulse response must be derived by comparing the received pilot to its expected value. This process is similarly non-trivial, especially in the case where the multipath is dynamic and the filter must adapt quickly.

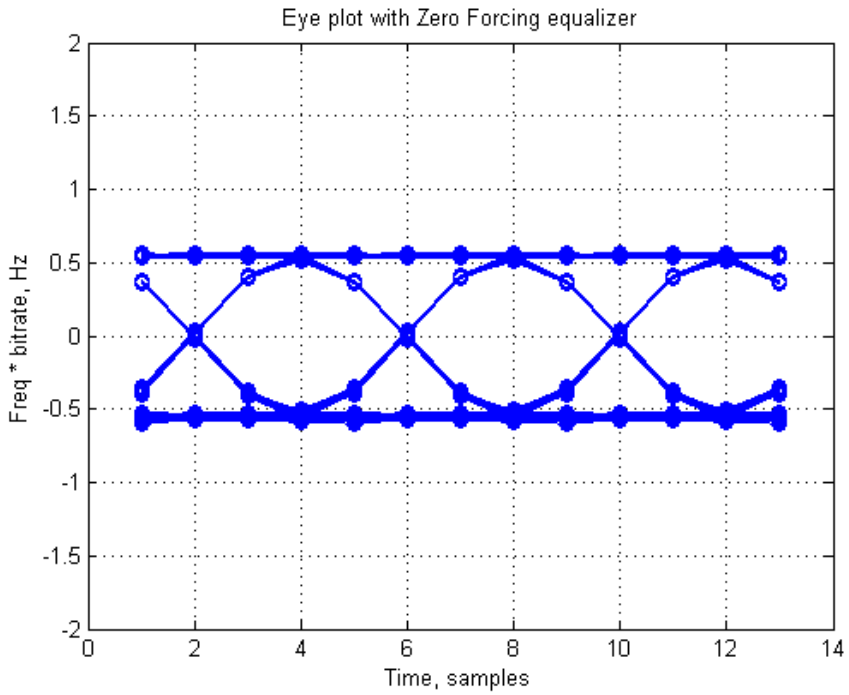

Figure 3.13: Eye diagram for system with Zero forcing equalizer.

3.4.2 Wiener Filter

A popular adaptive filter approach invented by Norbert Wiener[11] provides the optimal solution in the least squares sense[13]. This filter, known by the inventor's name, uses the known and observed data sets to create an overdetermined system. A matrix manipulation of the data forms an autocorrelation matrix, $\mathbf{R}$, and a cross-correlation matrix, $\mathbf{p}$. The solution for the channel inverse filter weights is given by

$$\mathbf{w} = \mathbf{R}^{-1}\mathbf{p}. \tag{3.15}$$

The block diagram in Figure 3.14 shows a top level view of the Wiener adaptive equalizer. This data-aided approach requires an embedded pilot sequence of known data. The correlator identifies

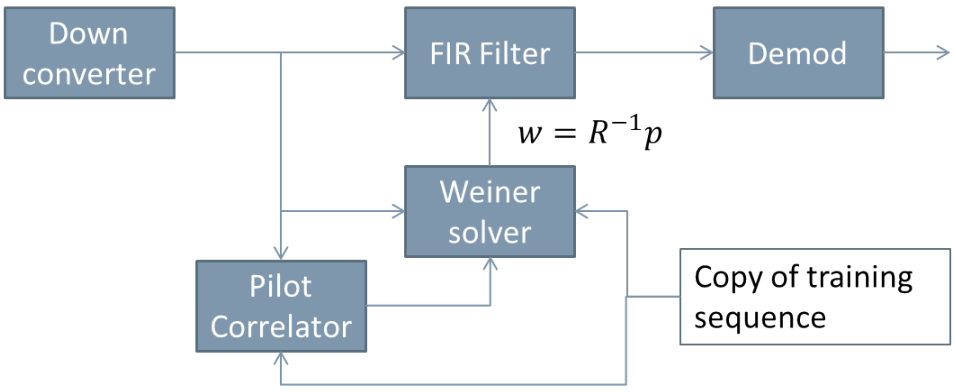

Figure 3.14: Block diagram of the Wiener adaptive equalizer.

the position of the pilot and the samples, $\mathbf{x}$, are fed to to the Wiener solver. The solver uses a stored record of the complex pilot samples, $\mathbf{d}$, made up of samples, $d[n]$, and forms the autocorrelation and correlation matrices.

Assuming sufficient length in $\mathbf{x}$ and $\mathbf{d}$, for a transversal filter length of N, and M observations, the received data vector is broken into M observation vectors of length N,

$$\mathbf{x_0} = [x[0], x[1], x[2], ...x[N-1]]\,,\ \mathbf{x_1} = [x[1], x[2], x[2], ...x[N]]\,, etc. \tag{3.16}$$

The autocorrelation matrix, $\mathbf{R}$, is given by

$$\mathbf{R} = \frac{1}{M}\sum_{i=0}^{M-1} \mathbf{x_i^T x_i}, \tag{3.17}$$

and the crosscorrelation matrix, $\mathbf{p}$, is given by

$$\mathbf{p} = \frac{1}{M}\sum_{i=0}^{M-1} d[i]\mathbf{x_i^T}. \tag{3.18}$$

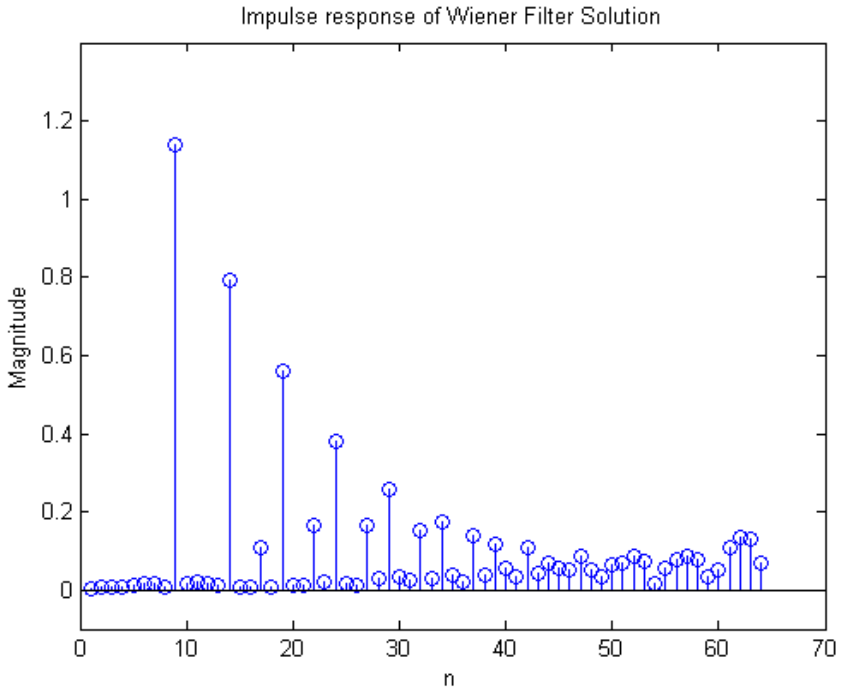

Figure 3.15: Magnitude tap weights of the Wiener adaptive equalizer.

From these we find the tap weights for the transversal filter, using Equation 3.15. Applying this process to the test signal, and using $M = 1000$ observations, the weights are calculated as shown in Figure 3.15. The corresponding eye diagram is shown in Figure 3.16.

Clearly, this solution has opened the eye, and should provide error free communication. In an FPGA implementation, especially for high speed signaling, this approach is plagued by the need to invert the autocorrelation matrix, which is a very computation intensive operation. Further, in order to constrain overhead, the pilot bit observations will normally be spread over multiple blocks.

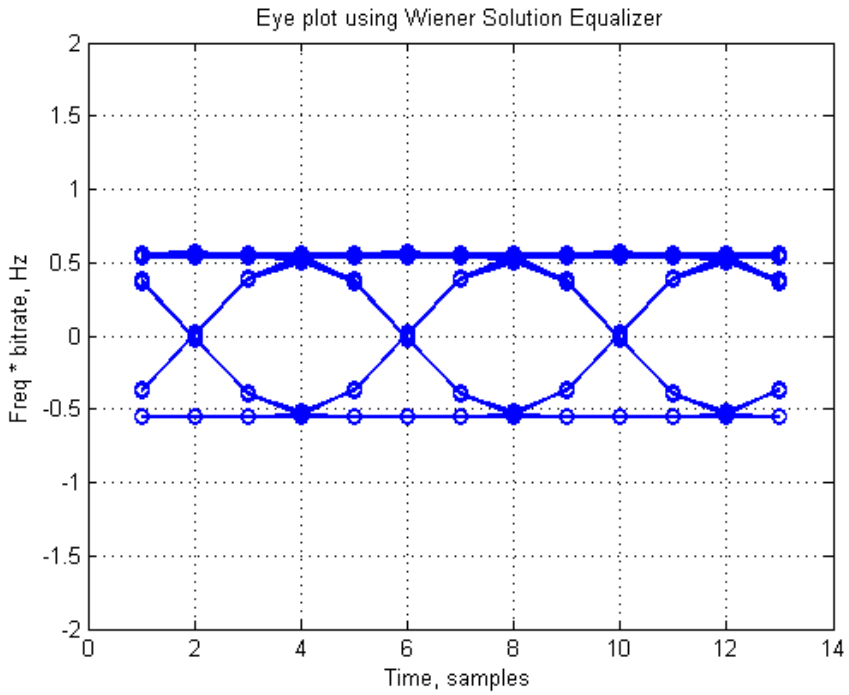

Figure 3.16: Eye diagram after application of the Wiener adaptive equalizer.

3.4.3 LMS Algorithm

The LMS algorithm, or Least Mean Squares algorithm, iteratively increments the taps of the transversal filter, according to a cost function given by the equation

$$e[n] = d[n] - y[n]. \tag{3.19}$$

The block diagram in Figure 3.17 shows a high level view of the architecture. As in the other data-aided approaches, a correlator is used to determine the position of the pilot sequence, $\mathbf{d}$. The output of the FIR filter, $y[n]$, is compared against the pilot sample, and forms the cost, $e[n]$, as described in Equation 3.19. This cost is used to update the taps according to

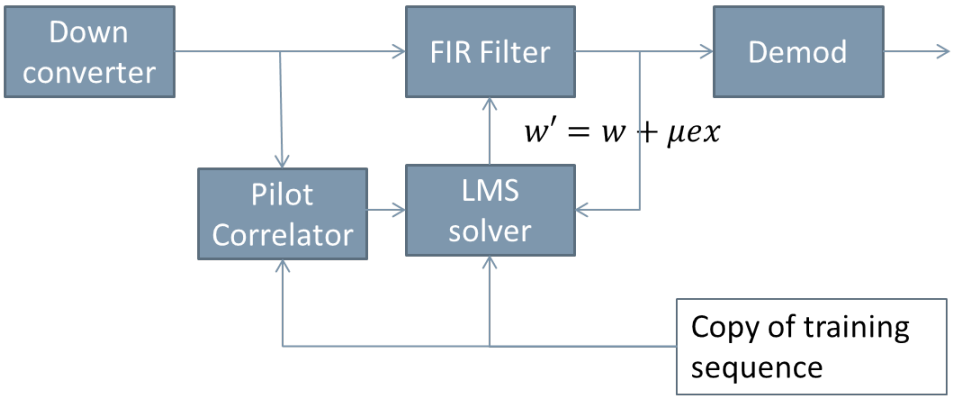

Figure 3.17: Block diagram of the LMS adaptive equalizer.

$$\mathbf{w}_n = \mathbf{w}_{n-1} + \mu e[n]\bar{\mathbf{x}}. \tag{3.20}$$

The constant, μ, is a gain constant, and adjusts the rate at which the taps update based on the cost function.

The LMS Algorithm is a stochastic gradient descent algorithm, generally attributed to Bernard Widrow[11], of Stanford University. Given sufficient iterations to converge, and a stationary channel, the LMS equalizer will converge on a solution that approaches the Wiener solution[13]. The rate of convergence is slow, and therefore begs the question as to why the Wiener (more exact) solution would not be applied. The computational complexity of the Wiener solution, which is based on autocorrelation and correlations, and requires a matrix inversion, is difficult to realize in hardware such as an FPGA. The LMS solution uses an iterative approach that is computationally very light compared to the Wiener solution.

The LMS algorithm was used to equalize the test stimulus, with $\mu = 0.005$, and resulted in the tap weights shown in Figure 3.18. The application to the test signal resulted in the eye diagram shown in Figure 3.19.

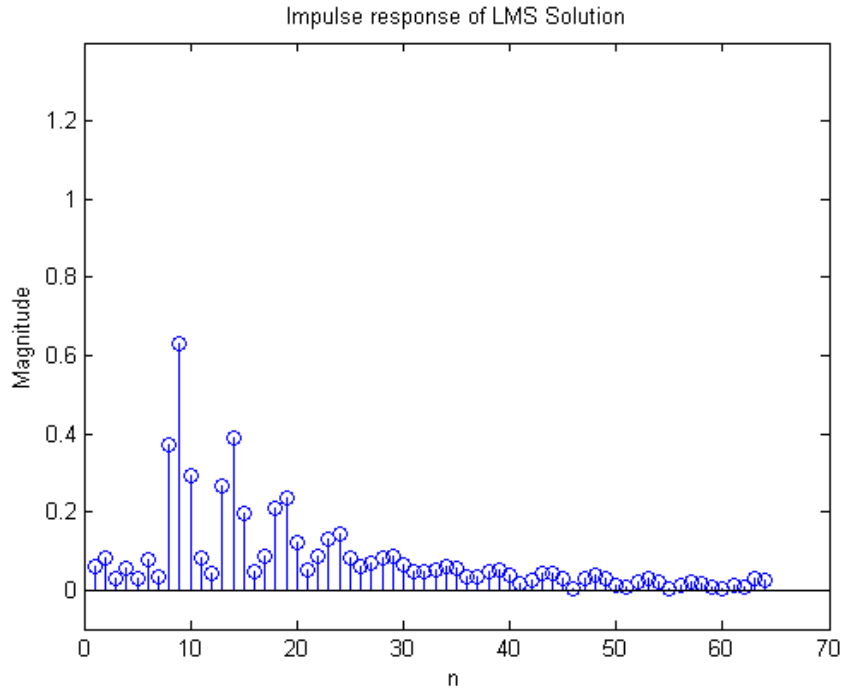

Figure 3.18: Magnitude tap weights of the LMS adaptive equalizer.

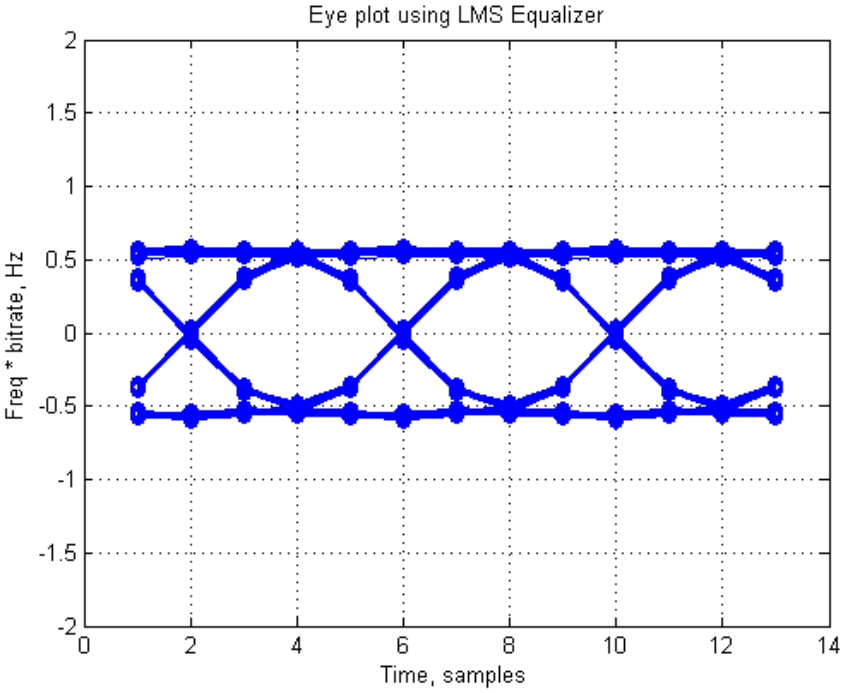

Figure 3.19: Eye diagram after application of the LMS adaptive equalizer.

Again, this approach has opened the eye, and with significantly lower processing load on the equalizer system. This algorithm is closely related to our subject approach.

3.5 Blind Equalizers

3.5.1 Decision Feedback Equalizer

One form of blind equalization is an approach termed decision feedback. In this approach, the best guess for the bit decisions are fed back to a Wiener, LMS, or other adaptive equalizer. The block diagram in Figure 3.20 shows an implementation using the LMS algorithm. Conceptually, if most of the guesses are correct, the solution will begin to converge, and the decisions will improve. Once a satisfactory inverse filter is attained, the decisions will be error-free or nearly so, and the equalizer will perform similar to a data-aided equalizer. A problem with this type of equalization is that if insufficient correct decisions are made, the equalizer may never converge on an acceptable solution.

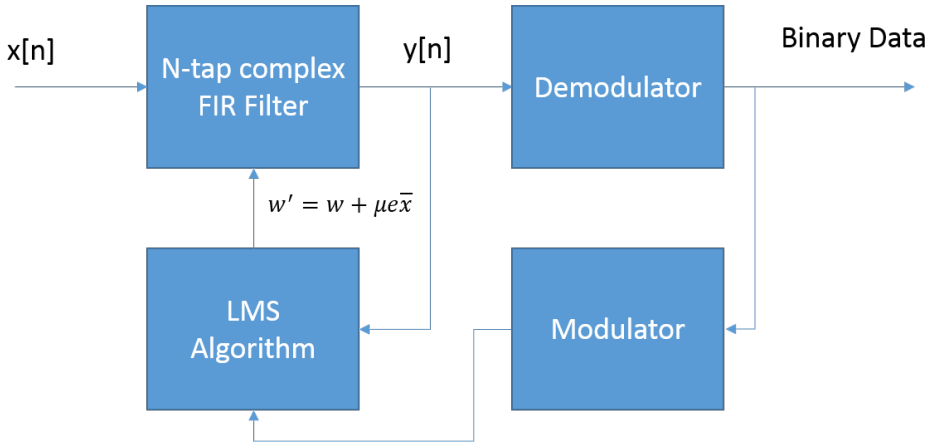

Figure 3.20: Block diagram for a decision feedback implementation of an LMS adaptive equalizer.

3.5.2 CMA Equalizer

The Constant Modulus Algorithm is similar to the LMS Algorithm in that it is a stochastic gradient descent algorithm. In fact, the tap weight update approach is identical, differing only in the cost function. In this algorithm, the cost function utilizes the expectation of constant envelope power to develop the direction of the gradient. Although beyond the scope of this work, it is interesting to note that the CMA algorithm can be used to provide some improvement even with non-constant envelope waveforms, assuming that the waveform displays a constant power over some reasonable interval. An example of this is its application to Quadrature Amplitude Modulation (QAM).

The block diagram in Figure 3.21 shows the equalizer connections in the receiver system. The CMA equalizer does not require known data, so the correlator is not needed to extract the pilot. The cost function for CMA is relatively simple to compute, although the application of the result to update the tap weights is not particularly intuitive. The cost function, e, is given by

$$e[n] = (1 - |y[n]|^2)y[n]. \tag{3.21}$$

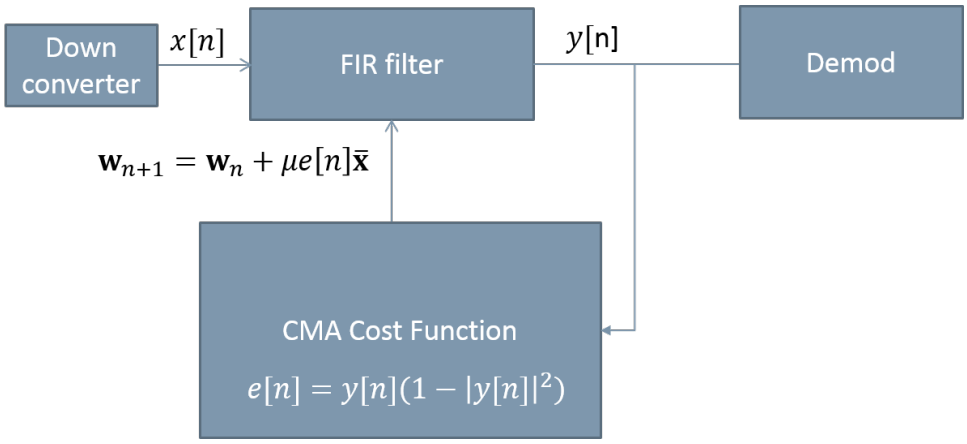

Figure 3.21: Block diagram of the CMA adaptive equalizer.

This cost function is then used to update the tap weights in exactly the same fashion as the LMS algorithm, using

$$\mathbf{w}_n = \mathbf{w}_{n-1} + \mu e[n]\bar{\mathbf{x}}. \tag{3.22}$$

Again, this computation is relatively light on resources as equalizers go, making it a good choice for a high-speed FPGA implementation. The CMA equalizer was utilized with the test input and $\mu = 0.001$, resulting in the tap weights in Figure 3.22. The eye diagram in Figure 3.23 is clearly opened with the CMA equalizer, and would certainly yield an error free link.

CMA Issues

So, clearly the CMA equalizer provides a number of significant benefits over data-aided equalizers. Examples include

1. The blind approach does not require a pilot. This eliminates the overhead associated, included expanded bandwidth, and more complicated block generation in the transmitter.

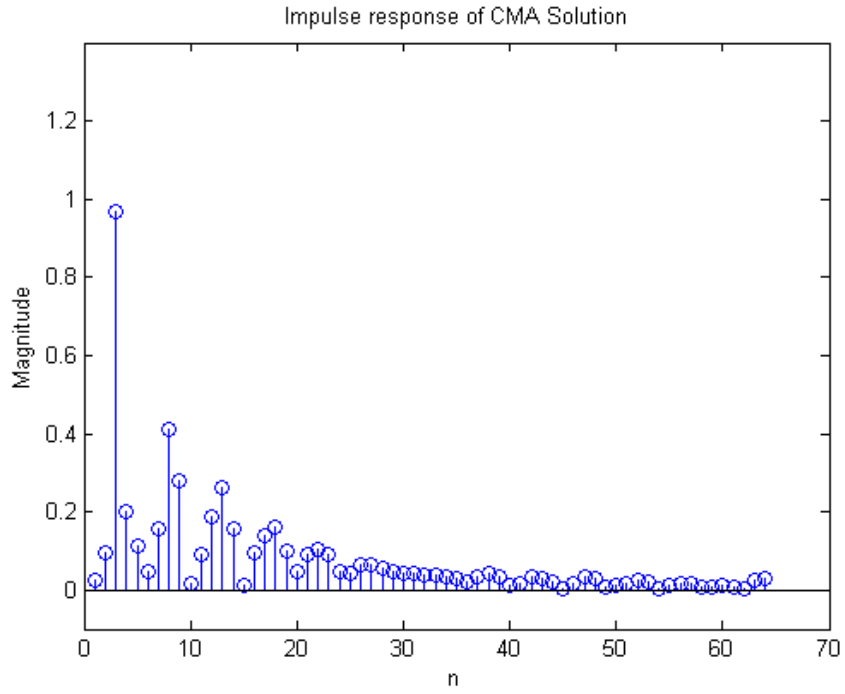

Figure 3.22: Magnitude tap weights of the CMA adaptive equalizer.

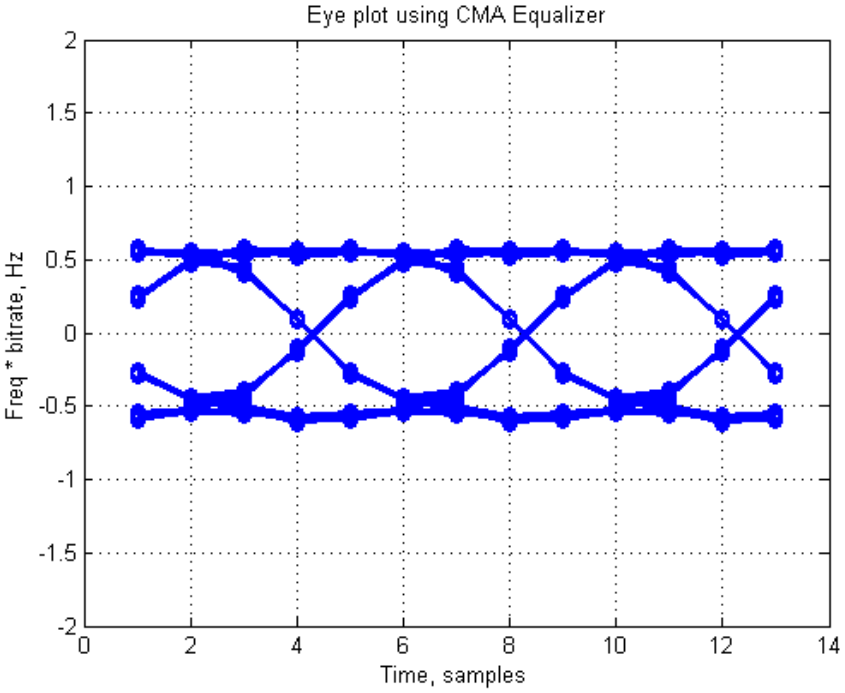

Figure 3.23: Eye diagram after application of the CMA adaptive equalizer.

2. As there is no pilot to recover, the receiver does not require a correlator or associated block timing to recover and process the pilot.

3. Blind equalization is preferred in applications where existing equipment is used, again, as the transmitters were generally designed without pilot sequences.

4. The relatively light processing load is well-suited for FPGA implementation.

However, the CMA has a number of weaknesses, including:

1. Unlike the Wiener and the LMS approaches, the CMA equalizer does not have a data reference. As a result, the delay through the system is uncertain. This will be discussed in greater detail later.

2. The CMA equalizer will tend to converge on a solution, but the length of time and path taken can vary widely. Acquisition times and tracking rates are reasonably predictable, but statistical in nature.

3. The CMA equalizer will occasionally converge on a poor or even unusable solution. Although this is sometimes attributed to an unsolvable channel, it may occasionally occur as a result of the path taken.

CHAPTER IV

FPGA IMPLEMENTATION

4.1 FPGA Synthesis

The FPGA synthesis is a multi-step process using sophisticated applications from MathWorks® and Altera®. The block diagram in Figure 4.1 gives a top-level overview of the design synthesis flow. The process starts with a design in MATLAB Simulink®, VHDL coding using a MATLAB tool called HDL Coder, compilation with the Altera Quartus® tool, and then programming into a target Altera device. Each of these steps will be described in greater detail below.

4.2 MATLAB Design

The FPGA implementation is achieved using MATLAB as the synthesis tool. MATLAB provides an entire host of engineering analysis and simulation capabilities. In recent years, MathWorks has introduced a toolbox integrated within MATLAB called HDL Coder. This toolbox takes a design from Simulink, and translates the design to equivalent VHDL (VHSIC Hardware Description Language) code. VHDL is commonly used to synthesize logic and digital signal processing design in Field Programmable Gate Arrays (FPGA's), and is universally compatible with FPGA compilers. While VHDL is a powerful syntax within its own right, the fixed precision power, simulation capabilities and array/matrix handling capabilities make MATLAB an extremely powerful tool to design

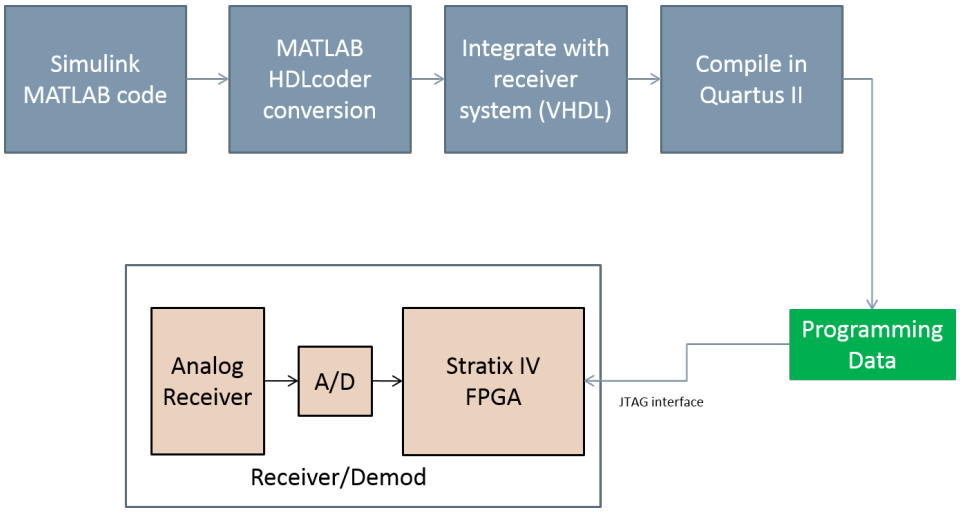

Figure 4.1: Block diagram showing the steps in synthesizing the equalizer in an Altera FPGA.

digital signal processing machines. HDL Coder then provides an efficient means to create FPGA programming firmware.

4.3 Simulink

Simulink provides a graphical interface to efficiently develop a digital signal processing system, and a powerful simulator to perform ”bit true” simulations for that design. Although Simulink was initially intended to provide an environment for graphical system design, a generic block is available that permits the use of a subset of MATLAB code to implement special functions. Although these operate similar to ordinary MATLAB functions, some special considerations are necessary for Simulink use. Additionally, the function block synthesizes a graphical block with input and output pins to use in the Simulink environment. As the HDL Coder product has matured, it has become more tightly integrated with MATLAB, and the use of MATLAB m-functions is now supported within the Simulink environment.

The use of MATLAB to create a system design provides a number of advantages.

1. MATLAB is an extremely powerful analysis tool for mathematical analysis and simulation, and its highly optimized engine provides the ability to perform millions of complex calculations very quickly.

2. The syntax and highly optimized matrix/array handling are well suited for many digital signal processing tasks, for example the FIR filter.

3. MATLAB's extensive offering of toolboxes for specialized processing are essential productivity enhancers. For this task, these include:

   (a) HDL Coder to generate VHDL from the Simulink design.

   (b) Fixed Point processing toolbox.

4.4 HDL Coder Toolbox

The HDL Coder takes a Simulink project as its input, and generates VHDL files. HDL Coder is configurable, but for the most part, the operation of the tool is as simple as a mouse click. The generated VHDL files are then used as input to the Altera Quartus II tool that compiles the VHDL for a specific FPGA device.

4.5 Fixed Point Processing

The FPGA design for this equalizer implementation necessarily utilizes a fixed point implementation, which MATLAB supports with the Fixed Point Processing Toolbox. In fixed point processing, the single and double precision variables generally encountered in numerical analysis software like MATLAB are replaced by variables of finite precision. For example, 8, 16 and 32 bit variations are commonly used. A 16 bit unsigned integer includes the range from 0 to $2^{16} - 1$. A 16 bit signed integer spans -2^{15} to $2^{15} - 1$. Although integers are simple to understand, to convert and manage all variables as integers would require a significant level of bookkeeping. Instead, fixed precison

with a fractional part is used. For example, a signed 16 bit with 12 bits fractional spans the range -2^3 to $2^3 - 1$ in increments of 2^-12.

Ordinarily, in digital signal processing that includes multiplication steps, it is advantageous to normalize signals to magnitudes of approximately one. This approach provides products after multiplication that will similarly have magnitudes of approximately one, thus allowing a uniform fixed point declaration. This approach keeps the bookkeeping of precision to a minimum. In some instances, however, some processes requiring smaller or larger values will require some manipulation of fixed point results. For example, the equalization cost function gain value, μ, is used to set the rate of convergence, and is generally on the order of $\mu = 0.001$. Suppose we intend to represent this value using a 16 bit number. If the standard variable type of signed 16 bit with 12 bits fractional $numerictype(1, 16, 12)$, the resulting representation is

```
fi(.001,1,16,12) = 9.7656e-04
```

whereas,

```
fi(.001,1,16,16) = 0.0010.
```

To further emphasize the point, suppose that $\mu = 0.001$ is multiplied by a small value, $b = .01$, also represented as a fixed point using $numerictype(1, 16, 12)$. It is instructive to look at the full 32 bit result as compared to when recast to 16 bits.

```
mu = fi(0.001,1,16,12);
b = fi(0.01,1, 16,12);

a = mu * b;

fi(a,1,32,24) = 9.7752e-06
fi(a,1,16,12) = 0
```

Although at times it is acceptable, or necessary, to keep the 32 bit result, it is common to recast results to the smallest width possible to conserve resources and improve computation speed in the FPGA. So, reconsider the example above, with some optimization to the fixed point casts,

```
mu = fi(0.001,1,16,16);
b = fi(0.01,1,16,28);

a = mu * b;

fi(a,1,32,32) = 1.0065e-05
fi(a,1,16,28) = 1.0066e-05.
```

Clearly, the accuracy of the result is reasonably well maintained, though the designer must take great care in analyzing the range of input and output variables when creating a mathematical process.

A design observation for the CMA equalizer is that the tap weight is an accumulator (integrator), and due to small values for μ, may accumulate very small increments. As the FPGA implementation is fixed point, it is necessary to keep a larger number of bits in the tap weight registers, though it is not necessary to use the entire register when executing the FIR. For example, the received complex samples, $x[n]$, are designed to be on the order of one. As the solution converges, the error function, $e[n]$, becomes increasingly smaller. Suppose that the power error is approximately 15 dB below the signal value, or a ratio of 0.03. In the tap update equation,

$$\mathbf{w}_n = \mathbf{w}_{n-1} + \mu e[n]\bar{\mathbf{x}}. \tag{4.1}$$

this value is multiplied by μ and the conjugate of $x[n]$. If $\mu = .001$, and $x[n]$ is approximately unity, the fixed point result

```
mu = fi(0.001,1,16,16);
b = fi(0.03,1,16,28);
```

```
a = mu * b;

fi(a,1,32,32) = 3.0195e-05
fi(a,1,16,12) = 0
```

So, in order to support the convergence to these levels, the tap weights must be accumulated with more precision.

One of the challenges historically of working with finite precision variables is managing overflow and underflow conditions. The default treatment in the Simulink fixed point processing is saturation. A signed, 16-bit, with 12-bit fractional variable accepts numbers in the range of -8.00 to +7.9998. Any attempt to store a number greater than 7.9998 will saturate to the maximum value, and similarly, a value less than -8.0 will saturate to the lower limit. Although good design practice avoids overflow and underflow, a system that processes signals with additive white gaussian noise will occasionally overflow or underflow, and saturation is preferable over wrapping.

4.6 Hardware Consideration

One consideration when managing fixed point values is the type of operation it will be used in. For example, the Altera Stratix device DSP blocks have four 18 bit by 18 bit high-speed hardware multipliers, as shown in Figure 4.2. Earlier, it was shown that the tap weight accumulators would perform better at more than 16 bits. Suppose that 24 bits were used. In the operation of the FIR filter, the tap weights are multiplied by the contents of the shift register. While the device will support 24 bit multiplication, it will significantly impede the clock speed, and consume a large quantity of resources. Further, the extended precision does not measurably improve the output fidelity, since the system must operate in a noisy environment.

So, to coordinate the precision of variables, the designer may need to recast variables. This is very easy to do using the MATLAB fixed point notation. For example, if y is a signed, 32-bit, 24-bit fractional and x is a signed, 16-bit, 12-bit fractional, y is recast to x by

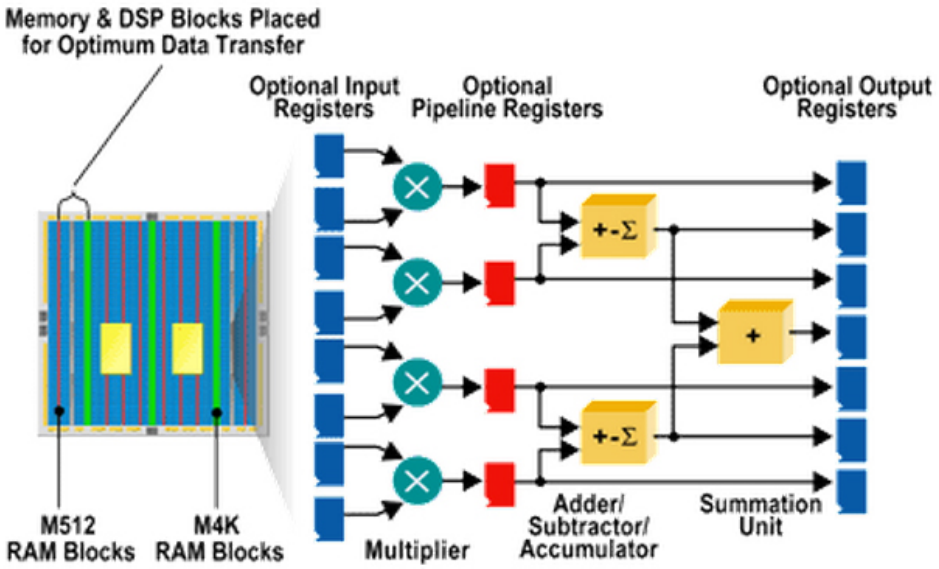

Figure 4.2: Block diagram of an Altera Stratix DSP block.

```
x(:) = y;.
```

In this example, x simply takes on the numeric value of y. If y is too large to fit in x, it saturates at the largest value of x; if it is less than the smallest value x can hold, it saturates at the smallest value of x. it Some care must be taken in recasting, as Simulink has some unique variable handling quirks.

Although not part of the current equalizer design, the device contains random access memory (RAM), which can also be configured as read only memory (ROM). These are arranged in blocks of 9 kilobits and blocks of 144 kilobits. When designing to use the RAM or ROM, the designer should consider the address and data bus sizes to optimize use of the memory.

4.7 Pipelined Architecture

When designing high speed functions within an FPGA, it is crucially important to understand the flow of the processing with respect to timing. The FPGA provides highly parallel processing,

which is very helpful in attaining high speed, but often complex mathematical operations cannot be accomplished in one clock cycle. A straightforward example of this is the finite impulse response (FIR) filter structure used in the equalizer, with the block diagram in Figure 5.1. For this implementation, a 64-tap complex FIR filter is realized. Ideally, at each tick of the clock, the FIR would produce a complex output for its present set of complex inputs. In MATLAB, the statement

```
y = sum(x .* h);
```

seems simple enough, but breaks down to 64 complex multiplications (four scalar multiplications plus two scalar additions for each complex multiplication), plus 63 complex summations (two scalar additions for each complex addition). The 64 complex multiplications can be done in one clock cycle using parallel DSP blocks, but a summation with 64 complex inputs would consume a mammoth amount of resources, and would require a substantial length of time for the solution to ripple to the output.

In order to accomplish this function and maintain high throughput, pipelining is used. In the pipelined approach, the function is broken into smaller pieces, and each piece is performed on subsequent clock cycles. This method has the negative effect of delaying the answer, but maintains a high rate of throughput, which is generally more important. Consider the diagram in Figure 4.4. The multiplications are done in parallel, resulting in 64 complex values, which are stored in registers. On the following clock cycles, the 64 values are reduced to 32 by adding adjacent samples. During the following cycle, these 32 are reduced to 16 and so forth until on the seventh cycle, the final summation, y_{out}, is performed. The timing diagram in Figure 4.5 shows how the FIR result progresses through the pipeline on each clock cycle.

In order to successfully implement the equalizer algorithm, careful accounting of pipeline delays must be accomplished. It is noteworthy that pipeline delays are generally not an issue for linear

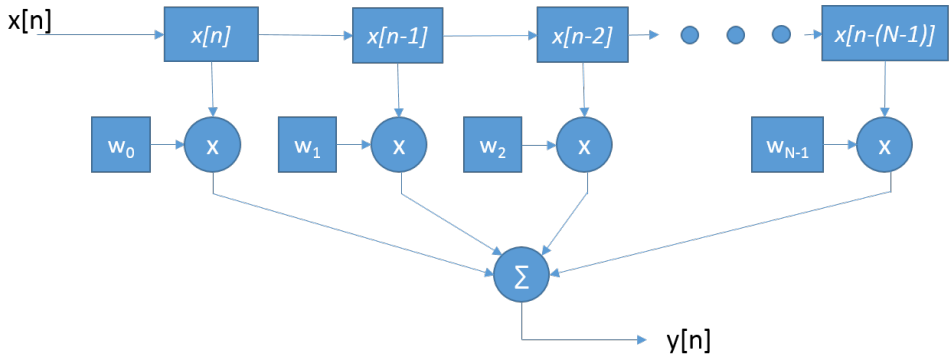

Figure 4.3: Block diagram of a finite impulse response filter.

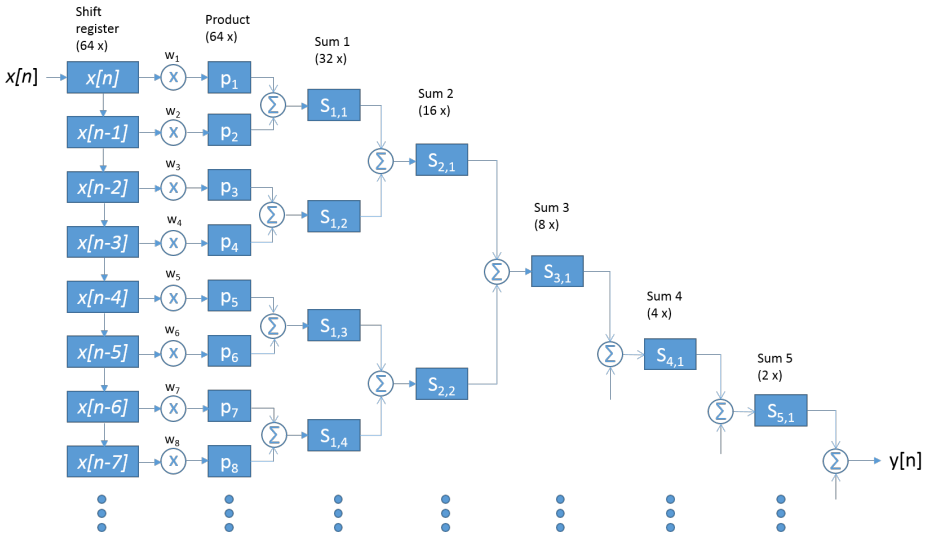

Figure 4.4: Block diagram showing pipelined FIR filter.

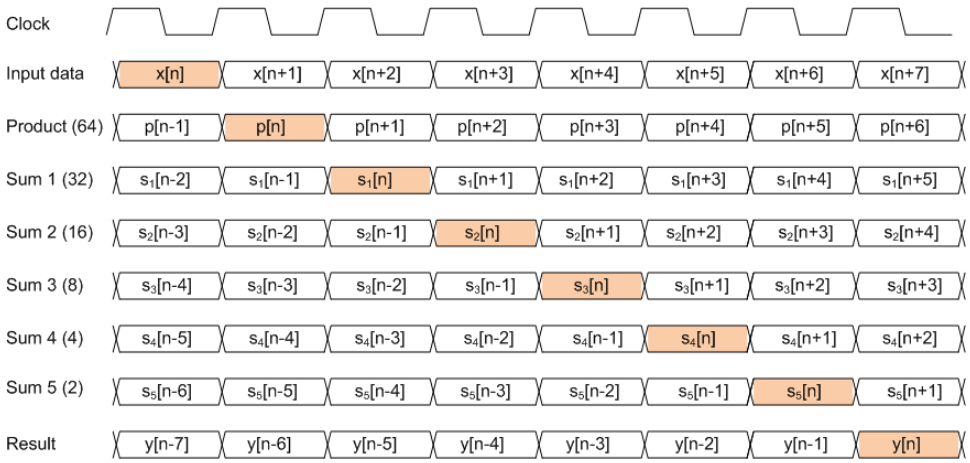

Figure 4.5: Timing diagram showing the progression of the FIR filter result through the pipeline.

processing, as the result is simply delayed. Pipelining can be an issue for processes that utilize feedback, as the delay must be accommodated.

4.8 Resource Planning

The FPGA target for this design is very large, with over 180,000 registers and nearly 1300 multipliers. While this is a large canvas to paint on, careless design practices can quickly consume resources. Early design rough-in generally includes a coarse estimate of resources needed to accomplish the overall design. In this application, the digital demodulator consumes a large percentage of the available FPGA resources. Within the equalizer, the FIR filter coupled with the tap update circuitry consumes the largest percentage of the remaining resources.

The FIR structure is a common digital signal processing tool, but in order to implement a high speed filter in an FPGA, a significant number of resources are consumed. The design implemented as shown in Equation 4.1 used 16 bit complex values in the complex shift registers, 32 bits in the tap weight accumulators, and 32 bits in each sum to generate the FIR filter output. Next, the resources for the tap update are included. There are 64 complex registers to store $\mu\bar{\mathbf{x}}$, 64 registers to contain

Description	Values	Registers (bits)	Registers (ext)
Shift Reg	64	32	2048
$\mu\bar{x}$	64	32	2048
Tap Delta	64	64	4096
Tap Weights	64	64	4096
Mult	64	64	4096
Sum 1	32	64	2048
Sum 2	16	64	1024
Sum 3	8	64	512
Sum 4	4	64	256
Sum 5	2	64	128
Result	1	64	64
Power	1	32	32
Delta	1	32	32
e[n]	1	32	32
Total			20512

Table 4.1: Table showing resource usage for core CMA implementation.

the pipeline value for $\mu\bar{\mathbf{x}}e[n]$, or the tap delta, and 3 registers to develop $e[n]$. Table 4.1 shows the estimate of expected registers in the equalizer core, estimated at 20,512.

In a device with 182,000 registers, this would not seem to be an overwhelming demand, but the digital demodulator consumes over half the available resources. Assuming approximately 40 percent of the devices total resources available, the core of the equalizer utilizes a little less than a third of the allocated registers. Register count is one measure of expected resource usage. As the design is placed by the compiler, logical functions will utilize Adaptive Look-Up Table (ALUT) elements [3], for example, to implement the adders in the filter summation steps. Estimation of the number of ALUT's is much more difficult, and for this type of logic, a value of 2 times the register count is a reasonable rough estimate.

[3] An Adaptive Look-Up Table (ALUT) is a logical construct that represents what can be implemented by the combinational logic hardware of an Adaptive Logic Module (ALM) in supported device families. *Taken from Altera website.*

To some extent, rough planning can be helpful at the start of a project, but given the complexity of DSP functions, and the wide array of resources in a high-density FPGA, the best information is derived from compiling to investigate the resource usage and maximum operating frequency.

4.9 Device Programming

The Quartus tool creates programming files for the Altera FPGA. These files are combined with compiled code for the Nios processor, which is instantiated within the FPGA. The converted programming file is then downloaded into EEPROM on the receiver. For the CMA equalizer, no changes to the Nios processor code were necessary, except to provide a switch to disable or enable the equalizer.

CHAPTER V

EQUALIZER ARCHITECTURE

5.1 Finite Impulse Response Filter

The main element of the equalizer that creates the inverse filter to repair the channel is a finite impulse response (FIR) filter. The FIR filter uses variable weights as determined by the CMA cost function. Referring to 5.1, the FIR filter is comprised of an N-length shift register, N tap weight registers, and summation circuitry to accumulate the result. The output of a FIR filter is given by[14]

$$y[n] = \sum_{k=1}^{N} w[k]x[n-k], \tag{5.1}$$

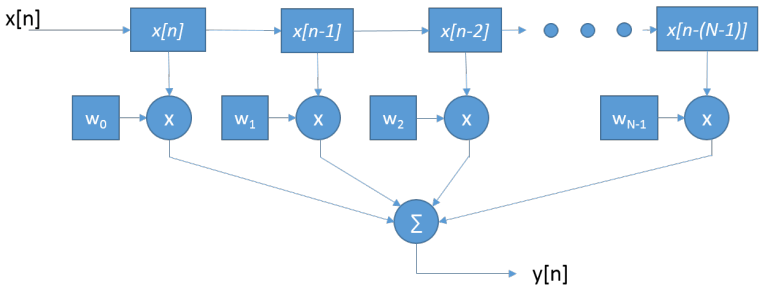

Figure 5.1: Block diagram of Finite Impulse Response (FIR) filter

It would seem intuitive that the filter length in samples, or alternatively, the length of the impulse response would be simply long enough to span the maximum delay in the multipath channel. However, the inverse response to the channel is generally an infinite response, so to approach an ideal solution, the FIR filter should be as long as possible. Clearly, in a hardware implementation, the number of taps are constrained by available resources, and the ability to meet timing requirements. In the tested implementation, the filter contained 64 delay elements, or ”taps”, each with a corresponding tap weight. The weights for these taps are determined by the Constant Modulus Algorithm cost function, which is described in section 3.5.2 in more detail. The 64-tap implementation is a reasonable compromise between hardware constraints in the FPGA and performance. To demonstrate this, the tap weight stem plots for 16-tap, 32-tap, 64-tap and 128-tap architectures are given in Figure 5.2, and eye diagrams are given in Figure 5.3. Observation of the different eye diagrams shows marked improvements from 16-tap to 32-tap and from 32-tap to 64-tap filters. The difference from 64-tap to 128-tap for the case shown, would not seem to justify the more than doubling of resources necessary to accommodate the additional taps.

The FIR filter design in the FPGA consists of a 64-stage shift register that stores the complex data samples. On each sample clock, the values in the shift register are shifted, with a new value being stored, and the oldest value discarded. Another array of 64 registers holds the complex values of the tap weights. The output of the filter is pipelined, in order to produce an output on each sample. The pipelining process is described in more detail in Section 4.7. The delay due to pipelining is 7 samples. It is worth noting that the finite impulse response filter implementation is the most popular for practical equalizer designs, but many other architectures are possible, but outside the scope of this thesis.

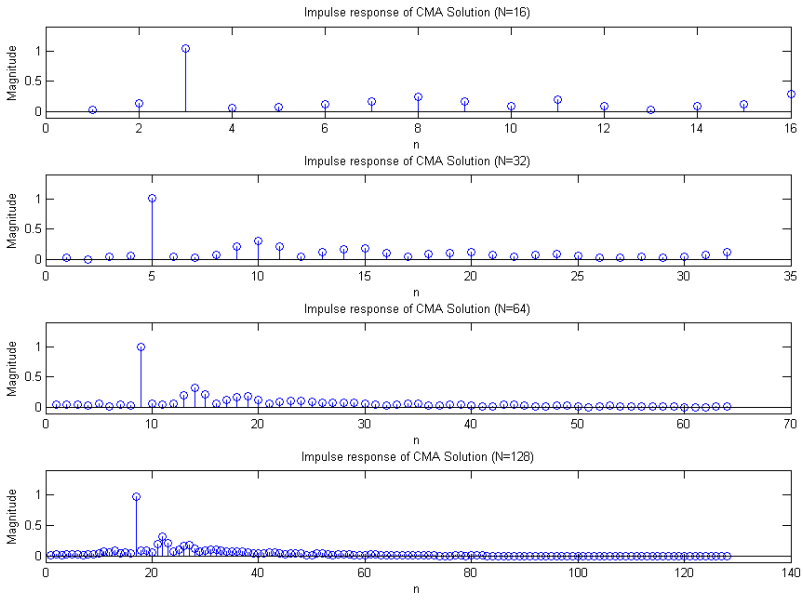

Figure 5.2: CMA Tap Weights for various length FIR filters

5.2 CMA Tap Calculation

The next significant processing function is the CMA tap update function. The cost function as described in Section 3.5.2 is realized in a pipeline architecture. Recall that the cost function

$$e[n] = (1 - |y[n]|^2)y[n], \tag{5.2}$$

is based on the filter output power, $|y[n]|^2$, which is computed using

$$P[n] = |y[n]|^2 = y[n]\bar{y}[n]. \tag{5.3}$$

Thereafter, the error magnitude is calculated by subtracting from a target power of 1, and then multiplying by the filter output to produce a complex error, $e[n]$. This process is accomplished in

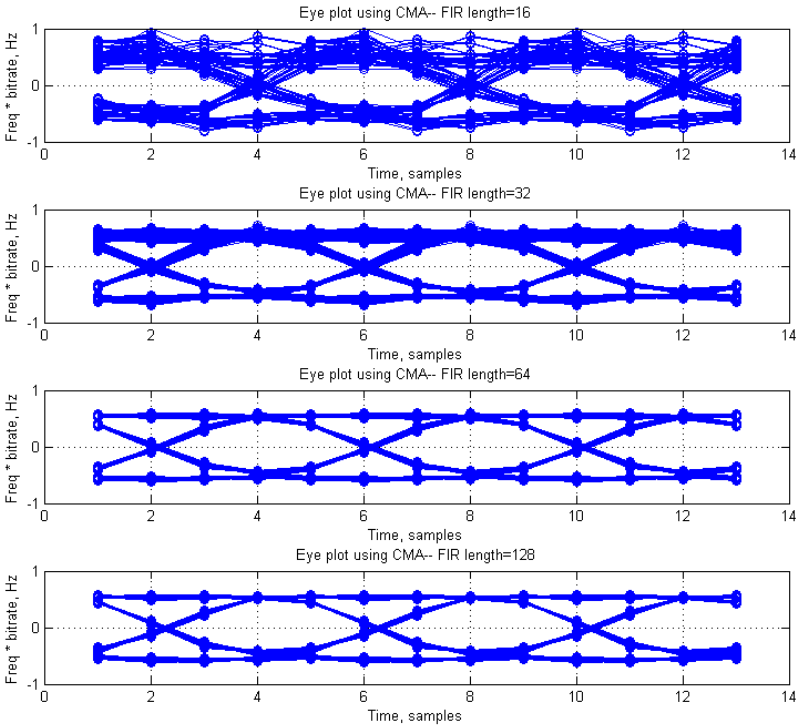

Figure 5.3: CMA Eye Diagrams for various length FIR filters

3 pipeline steps, extending the delay from the FIR to 10 samples. The final step to update the tap weights is to complete the update equation

$$\mathbf{w}_n = \mathbf{w}_{n-1} + \mu e[n]\bar{\mathbf{x}}. \tag{5.4}$$

Observing Equation 5.4, the term $\mu e[n]\bar{\mathbf{x}}$, requires two multiplications. However, to conserve multipliers, and acknowledging that the pipelining process will incur registers anyway, and that μ is constant, a resource optimization is accomplished by creating a second 64-stage shift register. This shift register will contain $\mu\bar{\mathbf{x}}$, delayed by the appropriate number of cycles to compensate for the pipeline delay. In addition to using only two multipliers (saving 126), the fixed point format of the register can be designed to minimize quantization loss. Since μ is generally small, on the order of $\mu = 0.001$, the register can be designed to hold the appropriate fixed point numeric range for the variable. So, the remaining steps of Equation 5.4 require multiplication by the error function , $e[n]$, and addition to the existing tap weight array. Note that this multiplication and addition produce complex results.

5.3 Convergence Monitor

Although the Constant Modulus Algorithm tends to converge for many channels, the algorithm does not guarantee convergence, and more importantly, the initial conditions play a role in attaining convergence. Therefore, there is some probability that for some channels, under certain initial conditions, that a solution will not develop. This condition is further aggravated by the fixed precision implementation. Since a tap weight is bounded by the maximum value of the register, sometimes convergence is impeded because a tap value has saturated, and a converging path to the solution does not develop.

Although an incorrect solution for a given channel may not bring severe consequences (certainly, some channels have no solution), these invalid solutions are often ”latched”, wherein the machine cannot recover, even as the channel multipath characteristic improves. This is certainly an unacceptable state for the machine to rest in, as thereafter no valid data is recovered, even in non-multipath situations.

Whatever the cause, for a practical, useful implementation, the algorithm must have a means to avoid, or remedy an invalid solution. Test observation of the conditions under which an invalid solution was encountered yield two conditions. First, the power distribution of the taps can be monitored for a particular signature. For most cases, this is sufficient information to determine that the system is stuck on an invalid solution. The second condition discovered is that when an invalid state is encountered, the data recovered by the demodulator will not be sufficiently random, given a random input, due to a characteristic of the demodulator. These methods are considered proprietary by Quasonix, and the technique is therefore not disclosed in detail.

If an invalid state is detected, the CMA machine is simply reset by applying a $1 + j0$ to the primary tap, and zero to all others. This has proven to give the initial condition most likely to result in convergence.

5.4 Tap Centering

Another challenge of the CMA approach is managing the delay through the FIR equalization filter. Consider the case where two FIR filters have the tap values shown in the two topmost graphs in Figure 5.4. These solutions provide identical outputs in magnitude frequency response, but they exhibit significantly different delays. In the third graph, a reference signal, V_{in} is convolved with the responses, yielding the outputs, V_1 and V_2, in the lower two graphs in the figure. Clearly, V_1 and V_2 are identical time domain responses, but have different delays.

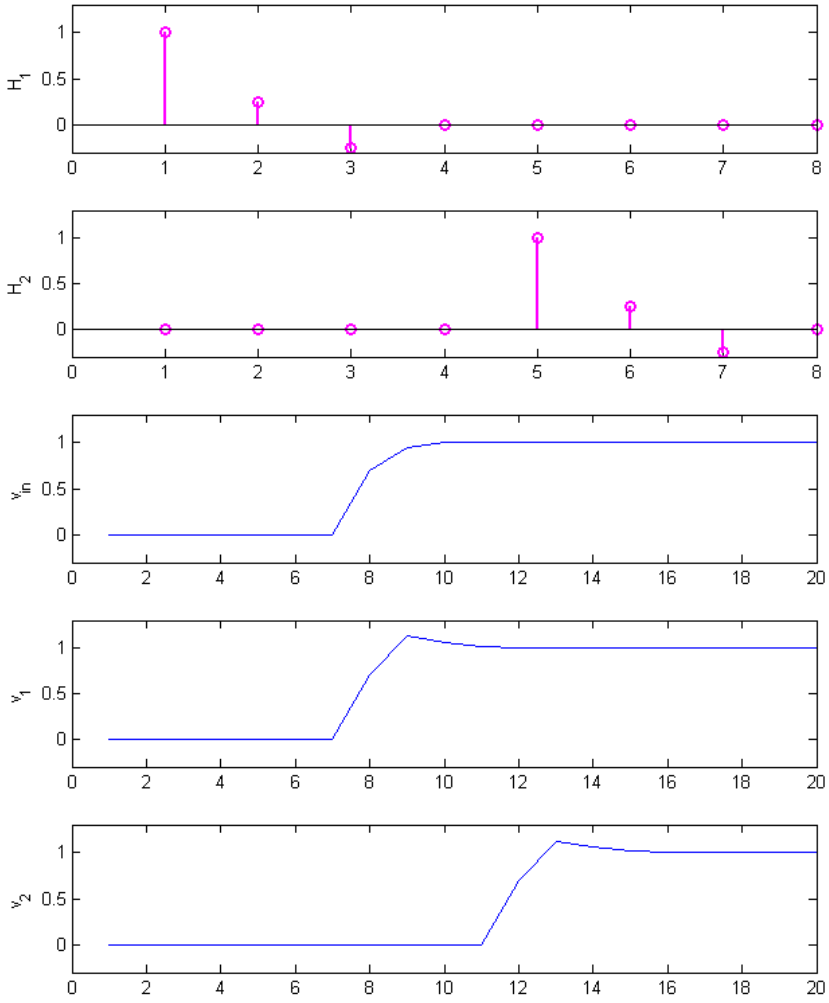

Figure 5.4: This series shows how H_1 and H_2 have identical magnitude responses in the frequency domain, but exhibit different delays.

Although a receiver by its fundamental design must adapt to different delays (e.g. the path delay for a mobile system changes constantly), as the rate of change of the delay approaches the symbol rate, the receiver will struggle to maintain proper timing. Further as the non-zero taps approach one end or the other of the equalizer FIR filter, the equalizer will be constrained potentially by the abrupt truncation of the response. The CMA design is a stochastic gradient search which generally converges to a solution, because there is no reference the algorithm has no knowledge of delay or the boundaries of the FIR filter. For identical path parameters, the algorithm can settle on slightly different tap arrangements, and delays. An approach was implemented to force the "center of gravity" of the taps to a set position in the FIR. The center of gravity is given by

$$COG = \frac{\sum_i |W_i| i}{\sum_i |W_i|}. \tag{5.5}$$

The approach to shift taps without discontinuity in the filter output is proprietary and is not disclosed here in detail.

5.5 FM Discriminator

5.6 Noise Generator

In channels that result in deep notches in the frequency domain, the solution tends toward large gain in the equalizer to correct those notches. This is one of the weaknesses of a zero-forcing approach. The CMA approach will migrate toward large gain solutions as well. A method that was tested and implemented intentionally added white noise to the input signal[4] in order to aid the convergence in the finite math implementation. The noise assists in two cases:

[4] this approach is also considered proprietary

1. The noise provides a method to force a truer average in the finite math processing. As the quantization of the signal may have a bias in value, an appropriate level of noise will help to mask biases.

2. In a path response that exhibits notches or deep zeros, the noise may help to constraint the solution to finite values. If left alone, the solution may attempt to produce an infinite response.

The noise was generated using a Park-Miller linear congruential pseudo-random noise generator (PRNG). The PRNG is approximately white in nature, with a uniform distribution in the time domain. The linear congruential generator is of the form $X_i = (aX_{i-1} + b) \bmod m$, where a, b, and m are constants.

Figure 5.5 shows the frequency spectrum and distribution of values. Although the distribution is uniform, it is postulated that the distribution is not critical to the function of the noise generator, as the noise is added far below the point where it would contribute errors. It is further postulated that possibly if AWGN were used, the gaussian "tails" contain very large values, that although infrequent may cause sporadic errors, whereas the uniform distribution is constrained. Future testing may explore performance comparisons with different generators, and with gaussian shaping (more like AWGN. Gaussian shaping can be accomplished by applying the Box–Muller transform where

$$n_1 = \sqrt{-2ln(u_1)}\cos(2\pi u_2), \tag{5.6}$$

and

$$n_2 = \sqrt{-2ln(u_1)}\sin(2\pi u_2), \tag{5.7}$$

where u_1 and u_2 are random variables with uniform distribution.

The scaling for the noise is controlled automatically by the equalizer. The power in the taps given by

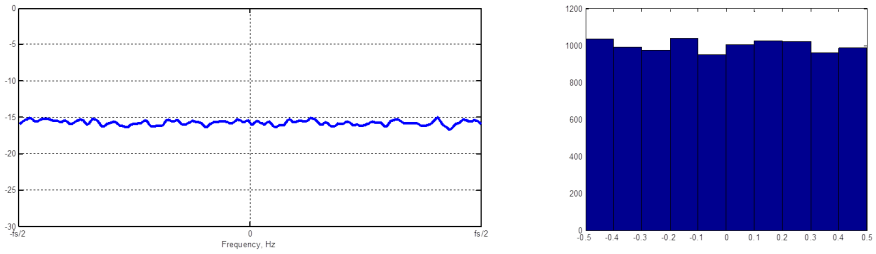

Figure 5.5: PSD and distribution for noise generator

$$P_{taps} = \sum_i |W_i|, \tag{5.8}$$

is compared to a threshold, and the gain of the noise generator is set in a feedback loop. The system Laplace form is given by

$$G_n(s) = (P_{targ} - P_{taps})(\frac{k_i}{s} + k_p). \tag{5.9}$$

This was found empirically to work very well to stabilize the tap magnitudes, without adding excessive noise when the system signal to noise decreased.

CHAPTER VI

SYSTEM ARCHITECTURE

6.1 Digital Receiver Overview

The advent of high speed, high density FPGA's (Field Programmable Gate Arrays) and ASIC's (Application Specific Integrated Circuits) has enabled the realization of high performance digital receivers. High profile military programs, such as Joint Tactical Radio System (JTRS), have incurred vast research and funding for the development of digital transmitters, receivers and transceivers. Generally, the receiver front end is constructed in the form of a heterodyne, as has been done for many years. The availability of miniature components, especially complex, high performance MMIC's (Monolithic Microwave Integrated Circuits) has led to highly integrated, densely packaged designs. Two methods are commonly employed in digital receivers, either conversion to baseband IF (intermediate frequency), or conversion to a non-zero IF.

The receiver used in this investigation is a dual conversion type, with a final analog IF at 70MHz. The level controlled IF is directly sampled by an analog to digital converter, with a sampling rate of 93.3333MHz. The receiver is designed and produced by Quasonix, in West Chester, Ohio. A photo of the receiver is as shown in Figure 6.1. The samples are delivered to the FPGA on a parallel bus, at 93.3333 Ms/sec, in scalar form. In the FPGA, the samples are downconverted to zero IF, or baseband, and are therefore of complex form. The equalizer is a complex implementation of a finite impulse response filter, and is followed by a sophisticated demodulator, standard to the product line.

Figure 6.1: Quasonix Digital Receiver. Photo courtesy Quasonix, LLC.

Because of the mathematical realization of the equalizer structure, the digital receiver implementation provides an ideal approach. In this chapter, the receiver functions will be described in greater detail, as the interactions of the various blocks are critical to understand for proper operation of the equalizer.

6.2 Receiver Functions

6.2.1 RF Downconversion

Although the receiver/demodulator is classified as a software defined radio, the function to convert a received RF signal at L, S or C band is done via conventional means, albeit in an extremely compact package. The RF is filtered using a tunable preselector, converted to a first Intermediate Frequency (IF) at 374 MHz, then converted to a second IF at 70 MHz. The block diagram in Figure 6.2 shows the progression of the dual conversion from the antenna to the FPGA. In addition to filtering and variable gain stages, the IF section contains several SAW filters at different bandwidths, so that an optimum configuration can be selected depending on modulation type and bit rate. The selection of intermediate frequencies to maintain selectivity, and optimize image and spurious performance, is a specialized skill, and beyond the scope of this manuscript.

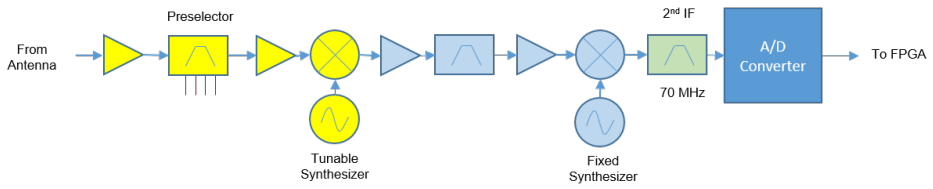

Figure 6.2: Block diagram of the receiver, showing dual conversion approach.

6.2.2 Automatic Gain Control

The receiver has an advanced Automatic Gain Control, or AGC, which receives feedback from the digital receiver to adjust gain in the RF and IF sections. The AGC characteristic is programmable according to optimum parameters based on mode, modulation type and bit rate. The AGC function is critical to optimizing operation of the digital receiver. First, analog to digital conversion technology is limited to approximately 14 bits at the conversion rate used in this implementation. The A to D converter in this receiver operates at 12 bits. At approximately $6dB$ per bit, this gives a dynamic range of about $72dB$, whereas the receiver is required to operate over a range in excess of 120dB. The AGC sets the gain to a level that optimizes the A to D conversion. Secondly, many of the specialized waveforms require a stable amplitude for demodulation. The AGC is key to this performance.

In order to preserve dynamic range through the analog receiver, the automatic gain control stages are distributed through the cascade of gain, filter, and mixing stages. In general, gain is reduced from the back end toward the front end for optimal performance.

6.2.3 Conversion to Baseband

The analog to digital converter samples the 70MHz IF input at 93.3333 MHz. Although the signal is above the Nyquist rate, it is sufficiently bandlimited, so the response in the second Nyquist region is returned, as shown in Figure 6.3. This is equivalent to sampling a response centered at an IF

of 23.3333MHz (with the spectrum inverted). An advantage of sampling at 93.3333MHz is that the signal is very easily converted to baseband by simply consecutively multiplying the samples by the sequence $1, j, -1, -j, 1, j, -1, -j$, etc., which describes a carrier of amplitude one at 23.3333MHz. This simple multiplication does not even require a hardware multiplier, as the IF samples are simply multiplied by zero, 1 or -1 as they are converted to complex baseband. For example, consider a stream of IF samples

$$s_1, s_2, s_3, s_4, s_5, s_6, s_7, s_8, s_9, s_{10}, etc.$$

The mix to complex baseband yields the following samples

$$s_1 + j0, 0 + js_2, -s_3 + j0, 0 - js_4, s_5 + j0, 0 + js_6, -s_7 + j0, 0 - js_8, s_9 + j0, 0 + js_{10}, etc.$$

This process is shown graphically in Figure 6.4. The complex baseband result contains an image at 46.6667MHz, which must be removed by filtering. This is done using digital filters in the FPGA.

6.2.4 Decimation (Resampling)

The digital demodulators require 4 samples per symbol, so the baseband signal is decimated, or more accurately, resampled to 4 samples per symbol. The decimation subsystem creates a clock using a numerically controlled oscillator driven by the 93.3333MHz reference. The resampled output rate can be any frequency up to 93.3333MHz. Because the decimation clock uses the 93.3333MHz as its reference, the output rate is an average rate, but does not guarantee that the clock enables are evenly spaced in time. The resampling is performed with interpolation, and does provide evenly spaced samples in time. Figure 6.5 shows the decimation operation, with interpolation between samples to provide time accurate samples. Decimation is a critical process in the FPGA implementation, as it significantly decreases resources in the signal processing of the received signal, without violating Nyquist criterion for sampling rate.

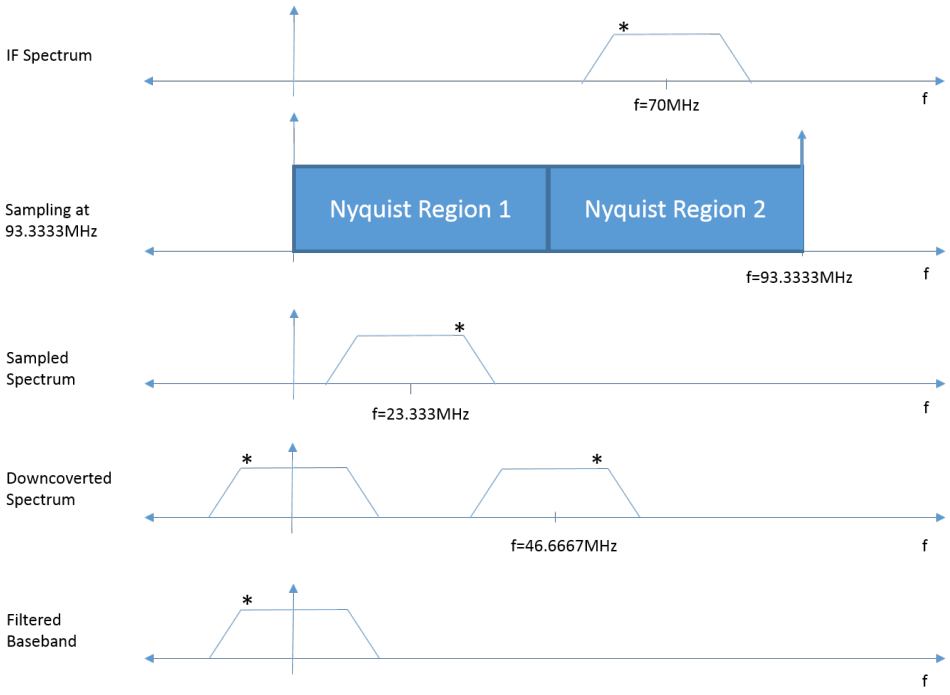

Figure 6.3: Receiver Sampling approach. The IF is sampled at 93.3333MHz, mixed to baseband and filtered.

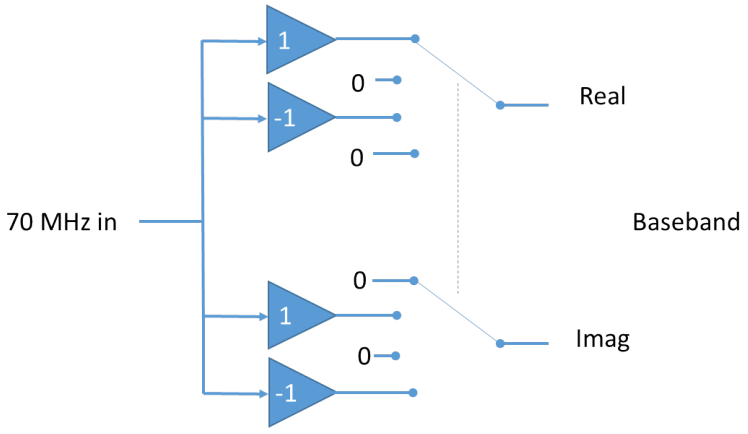

Figure 6.4: Digital Downconverter

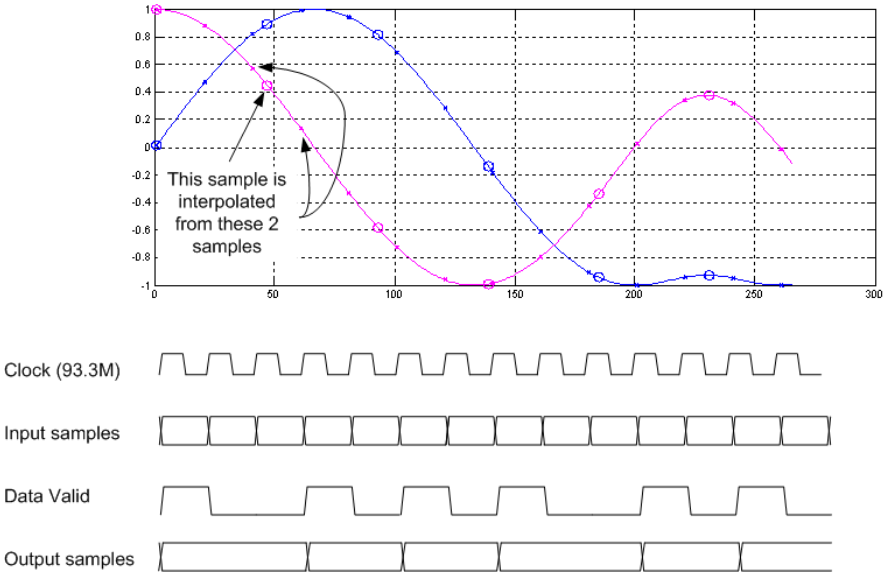

Figure 6.5: Example of decimation function interpolating samples synchronously.

6.2.5 Frequency Loop

When the transmitted message is created and applied to a carrier, the source reference clock is a high precision reference, often with an accuracy of $\pm 1ppm$. Similarly, the receiver nominal center frequency is determined by the receiver reference clock, which is also on the order of $\pm 1ppm$. Even with that precision, with use in C-band, this equates to over 8 kHz possible total error. Further, with airborne test articles, doppler frequency shift adds algebraically. The doppler shift is given by

$$f_{shift} = \frac{v}{v+c} f_c cos\theta, \tag{6.1}$$

where v = magnitude of velocity of test article,

c = speed of light,

f_c = carrier frequency, and

θ = angle between test article path and receiver.

For example, consider an aircraft flying at a velocity of Mach 1 (761 miles per hour) directly at the receiver, the doppler shift at C band is over 2 kHz. The doppler shift adds algebraically, which for this example produces a frequency offset of up to 10 kHz. This offset must be resolved by the demodulator, and corrected by multiplying the baseband signal by the inverse of the offset, given by

$$x'[n] = x[n]e^{j\theta[n]}, \tag{6.2}$$

where $\theta[n] = \theta[n-1] + 2\pi\frac{-f_{offset}}{f_{sample}}$,

The multiplication at baseband is done using a numerically controlled oscillator (NCO) as the other multiplicand. The NCO is made up of a phase accumulator and a sine/cosine look-up table. The accumulator is configured as a wrapping register, so an overflow or underflow is wrapped (i.e. overflow or underflow value truncated to the register size). The block diagram in Figure 6.6 shows a block diagram of the NCO. In Figure 6.7, the NCO phase accumulator is depicted as a circle, indicating its circular continuity as a wrap accumulator. The input frequency is essentially a phase increment to the accumulator, leveraging the relationship

$$f = \frac{\Delta\phi}{\Delta t}. \tag{6.3}$$

The accumulator output, generally rounded to fewer bits, is input as an address to a sin/cos lookup table, which outputs $a + jb$. The complex received baseband signal is then multiplied by this complex value to offset the frequency error. An NCO can generate a very precise frequency output. For example, a 10MBPS signal is samples at 4 times per symbol, yielding $f_s = 40MSPS$. If an NCO is designed with a 32 bit accumulator, the NCO precision is given by

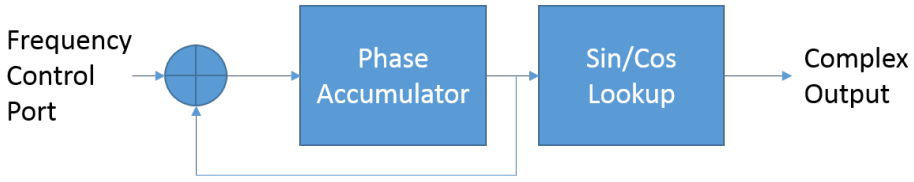

Figure 6.6: Block diagram of numerically controlled oscillator (NCO).

$$f_{incr} = \frac{1}{2^{32}} f_s = 0.009Hz. \tag{6.4}$$

The frequency control loop utilizes a detection circuit to detect the offset frequency in the demodulated data. This frequency is fed back in a control loop to drive the NCO in the opposite direction. Although, many frequency detection methods do not provide an exact value for frequency error, the loop iteratively drives the frequency offset the proper direction, until the error is zero. The frequency control loop is relatively sophisticated in its implementation, which is beyond the scope of this thesis.

6.2.6 Timing Loop

A similar function of the receiver is to determine the exact bit rate, so the demodulator can be synchronized to the bit rate of the transmitter. The timing loop determines timing from the bit synchronizer, and adjusts a numerically controlled oscillator to exactly match the correct rate. Due to the trellis demodulation, the timing loop in the test receiver is somewhat sophisticated. For our purposes, the function can be simply described as an operation that detects zero crossing in the demodulated output. A numerically controlled oscillator generates a bit clock that is aligned to the data using a servo loop, generally consisting of a proportional and an integral term. A common implementation called an early/late gate structure is depicted in Figure 6.8. In this approach a first integrator integrates samples for one-half of the bit period prior to an anticipated edge and a second

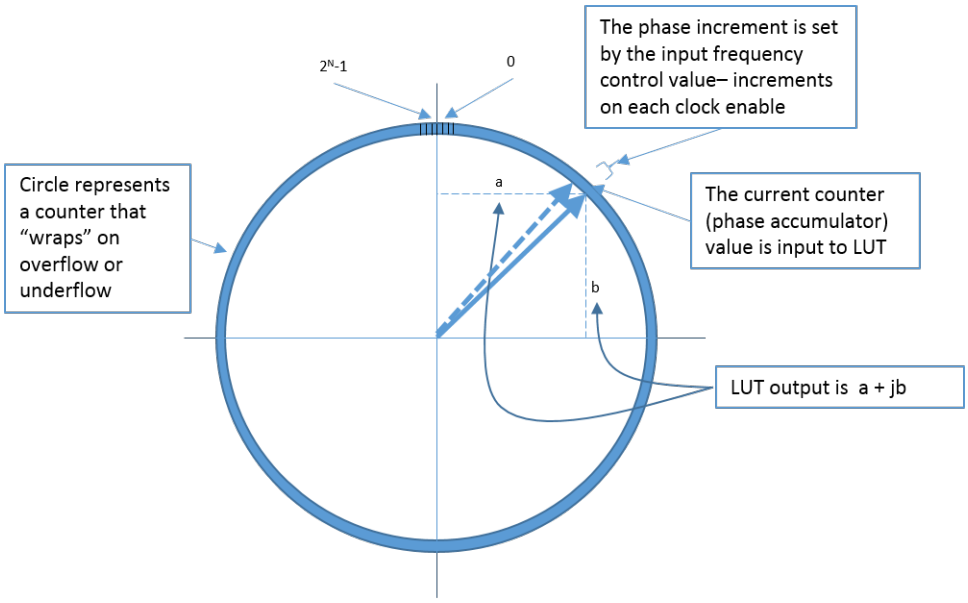

Figure 6.7: Graphical description of NCO operation.

integrator integrates samples for one-half of a bit period after the anticipated edge. The absolute values of these integrator are subtracted. If the transition is centered between the integration periods, the error is zero. An unbalance in the integrator will create an error that drives a servo loop to force an NCO to the proper frequency and phase, operating similarly to a conventional phase locked loop.

6.2.7 Demodulation

The most important function in the receiver subsystem is the demodulator. The demodulator converts the complex baseband signal to information bits. Demodulators appear in many forms, trading complexity and resource usage for robustness and acquisition speed, among other features. The demodulators in the test receiver use an advanced technique termed trellis demodulation. Trellis demodulators seek to wring every last bit of information out of the received waveform, especially in additive, white, Gaussian noise (AWGN). The trellis designs for the 3 ARTM waveforms operate on continuous phase modulations, and in simple terms, attempt to match the waveform over several

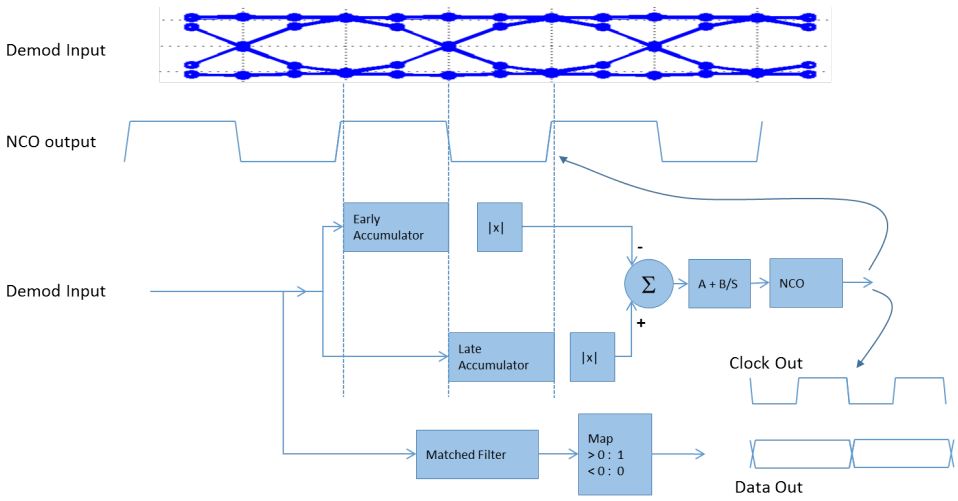

Figure 6.8: Bit synchronization using early-late gate approach.

symbols to work back to the most probable sequence of bits. These general provide significantly better performance than single symbol detectors.

To understand a simplistic view of the trellis, consider the PCM/FM waveform discussed in Section 3.3.1. The waveform is clearly frequency modulated, and a simple demodulator would detect the frequency versus time, to yield the data states. This straightforward approach works well in a low noise, undistorted channel, as shown in the eye pattern for the transmitted signal in Figure 3.4. However, consider another method to analyze the waveform. The diagram in Figure 6.9 shows the baseband waveform plotted on the complex plane versus time. Since the waveform is a constant amplitude, the result is a cylinder. The path traced on the cylinder shows the phase progression. This can alternatively be shown in a phase trajectory plot in Figure 6.10. The phase is ”unwrapped”, so as not be be discontinuous at $\pm\pi$.

In Figure 6.11, using the same phase trajectory, a sequence of five symbols is highlighted to show the ideal trajectory for that combination of bits. The frequency discriminator is shown to the right, showing perfect demodulation, as would be expected in a noise free signal. In Figure 6.12,

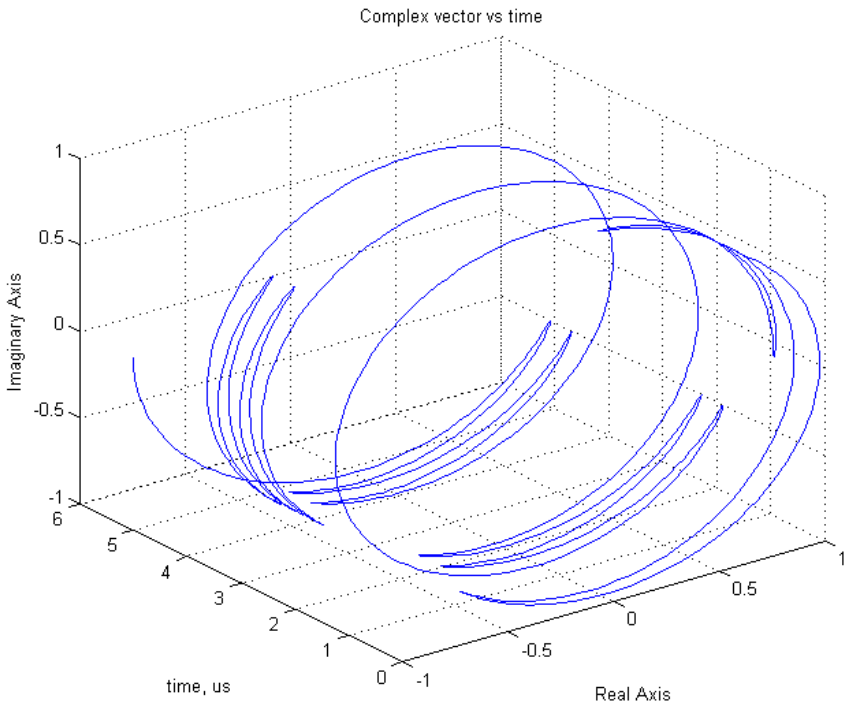

Figure 6.9: Graphical view of PCM/FM on complex plane versus time.

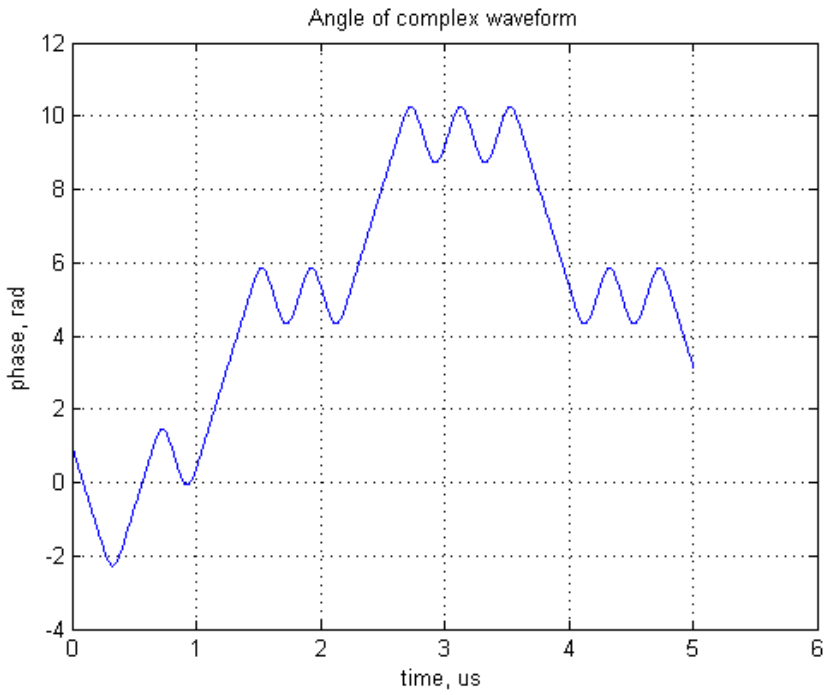

Figure 6.10: Graphical view of PCM/FM on complex plane versus time.

noise has been added to the signal. The phase trajectory at the left is noisy, but it it easy to visualize that the trajectory would correlate strongly with the ideal segment, while the discriminator output at the right would certainly cause occasional errors. Simplistically speaking, the trellis demodulator compares the phase trajectory over a number of symbols to ideal trajectories, to find the most probable match. Highly optimized algorithms exist, such as Viterbi decoders, in order to realize the trellis decoder in an FPGA implementation.

Although this is a qualitative comparison, the trellis demodulator clearly has an advantage in AWGN over a single symbol detector. Empirical data has shown that the trellis demodulator also performs better than a single symbol detector in light multipath channels.

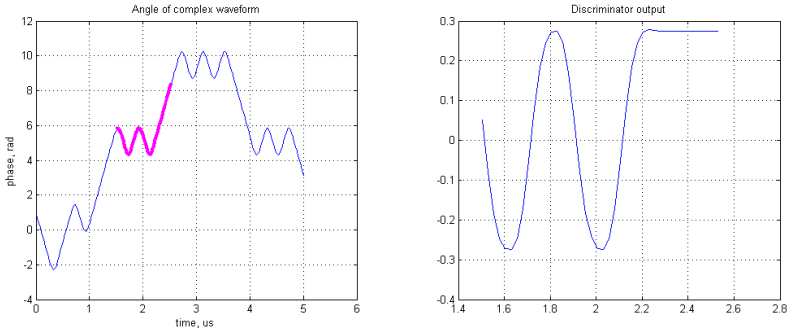

Figure 6.11: Phase trajectory and frequency discriminator plots for a PCM/FM signal without noise.

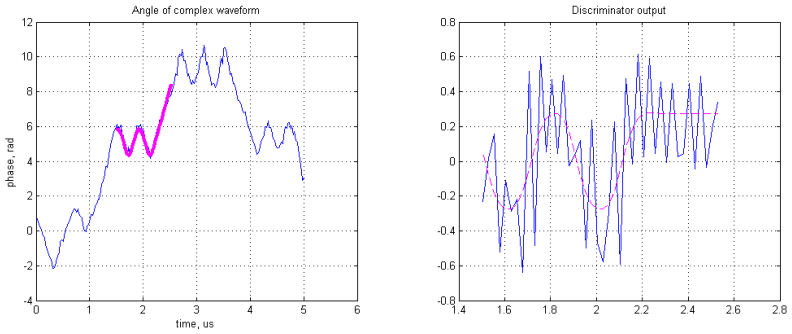

Figure 6.12: Phase trajectory and frequency discriminator plots for a PCM/FM signal with noise.

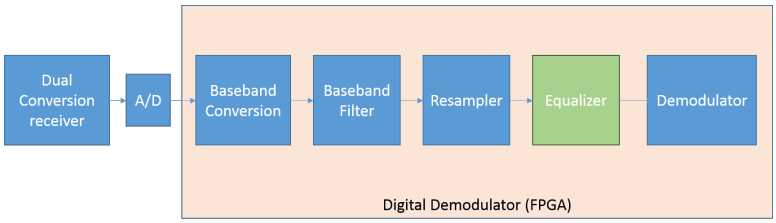

Figure 6.13: Receiver System block diagram showing Equalizer insertion.

6.3 Equalizer Insertion

The equalizer operates on baseband information, attempting to drive samples to a fixed radius on the complex plane. For this reason, the equalizer is best suited prior to demodulation. If the equalizer were positioned before the decimation stage, equalization would need to occur at the full sample rate, regardless of bit rate. This limits the length of the finite impulse response filter, which optimally spans as many symbols as possible. Therefore, the equalizer is situated between the decimation function and the demodulation function. The equalizer does not require the frequency of the received signal to be exactly resolved, but the best performance is realized with very small offsets, so it is advantageous to place the equalizer function within the frequency loop. Similarly, the equalizer does not require exact resolution of the bit timing, but more stable operation is obtained with the timing resolved, as the samples will fall in sync with bit timing. Further, timing resolution is achieved by minor changes to the decimation rate, which necessarily precedes the equalizer, as described above. The block diagram in Figure 6.13 shows the receiver subsystem with the equalizer inserted.

With the equalizer inserted, the samples in the 70MHz IF are converted to the equivalent IF of 23.3333MHz. This is converted to complex baseband and the image filtered. The samples are resampled to 4 times the symbol rate in the decimation block. These samples are presented to

the CMA equalizer. Assuming multipath free transmission, and well-constructed receiver front-end, these complex samples should lie on a circle of constant radius if plotted on the IQ plane. When multipath is introduced, amplitude variation appears on the received waveform, and the linear, transversal filter in the equalizer attempts to adapt the filter weights to achieve a constant radius. The demodulator then operates on equalized samples.

With the equalizer placed as shown in the block diagram, the receiver operates as it would normally, and is essentially unaware of the presence of the equalizer, and its attempt to improve the signal quality. As such, the equalizer is applicable to all constant envelope mode, including Tier 0, Tier 1, and Tier 2 waveforms as defined in the IRIG 106 specification.

CHAPTER VII

TEST RESULTS

7.1 Test Setup

The test setup shown in Figure 7.1 was used to exercise the receiver/demodulator with the CMA equalizer installed. The test source shown in the block diagram is a Quasonix Receiver Analyzer. For the test results presented, the Receiver Analyzer was configured to output a base signal with the characteristics shown in Table 7.1. The receiver/demodulator produces an analog output from a frequency discriminator before the equalizer, and a second analog output from a frequency discriminator after the equalizer. These signals are available on connectors on the receiver, and are displayed on the oscilloscope. Alternatively, the analog outputs can by configured to present a near real-time representation of the FIR tap weight magnitudes. This output may also be displayed on the oscilloscope. The spectrum analyzer is connected to the 70MHz intermediate frequency output on the receiver. This output is spectrally identical to the spectrum at the RF frequency, and translated to 70MHz by the dual conversion receiver.

Although a wide range of test conditions were exercised, three are presented in this section.

1. As a baseline, the test system produced the signal described in Table 7.1 with no multipath.

Waveform	Tier 0 (PCM/FM)
Frequency	1485MHz
Data rate	5Mbps
Ouput Power	-60dBm
Data Pattern	PN15

Table 7.1: Table showing test parameters for Receiver Analyzer test source.

Ray	Loss	delay (ns)	phase (degrees)	Doppler (Hz)
Ray 1	0	388	276	0
Ray 2	0.423	0	66	0
Ray 3	4.86	48	66	0

Table 7.2: Table showing test parameters for static multipath channel.

2. To demonstrate the system with a static multipath channel, the channel characteristics in Table 7.2 were utilized. The multipath characteristic is taken from a paper detailing observations made for telemetry channels at Edwards Air Force Base in California[10].

3. To demonstrate the adaptive capability of the equalizer, the channel characteristics in Table 7.3 were utilized. Note that the highest doppler frequency is 0.14 Hz. The slow rate is chosen to allow visual observation of the spectrum display and the tap weight adaptation.

Ray	Loss	delay (ns)	phase (degrees)	Doppler (Hz)
Ray 1	0	0	0	0
Ray 2	4.5	200	0	0.035
Ray 3	15.0	400	0	0.072
Ray 3	8.4	500	0	0.143

Table 7.3: Table showing test parameters for dynamic multipath channel.

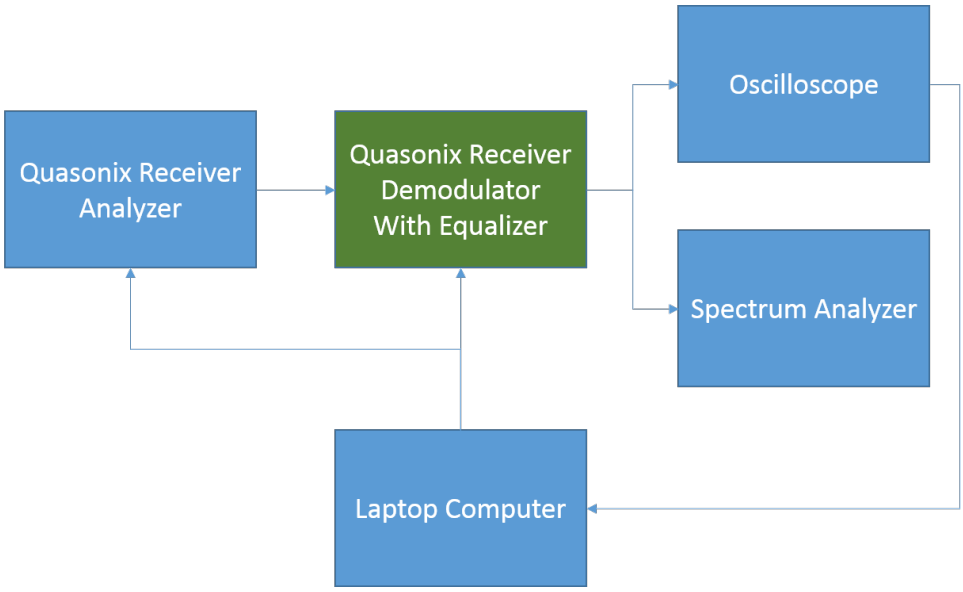

Figure 7.1: Block diagram showing equipment used to test the receiver with equalization.

7.2 Spectral Plots

In order to observe the multipath impact to the channel, the spectrum was viewed at the receiver IF output on a spectrum analyzer. In Figure 7.2 the transmitted spectrum is shown with no multipath. In Figure 7.3 and Figure 7.4 the spectrum analyzer display is shown for the dynamic multipath channel, at two different times, to demonstrate the changing spectrum of the dynamic multipath.

7.3 Eye Diagrams

The oscilloscope in Figure 7.1 is connected to two analog outputs on the receiver. These outputs can be configured to display eye diagrams for equalized and unequalized demodulated data, or alternatively, to display a near real-time representation of tap weight magnitudes in the finite impulse response filter in the equalizer. The baseline signal (no multipath) produced an eye pattern as shown in Figure 7.5. The corresponding tap weights are shown in Figure 7.6. The eye pattern indicates no detectable distortion or noise caused by the equalizer. The tap display shows a cluster of larger

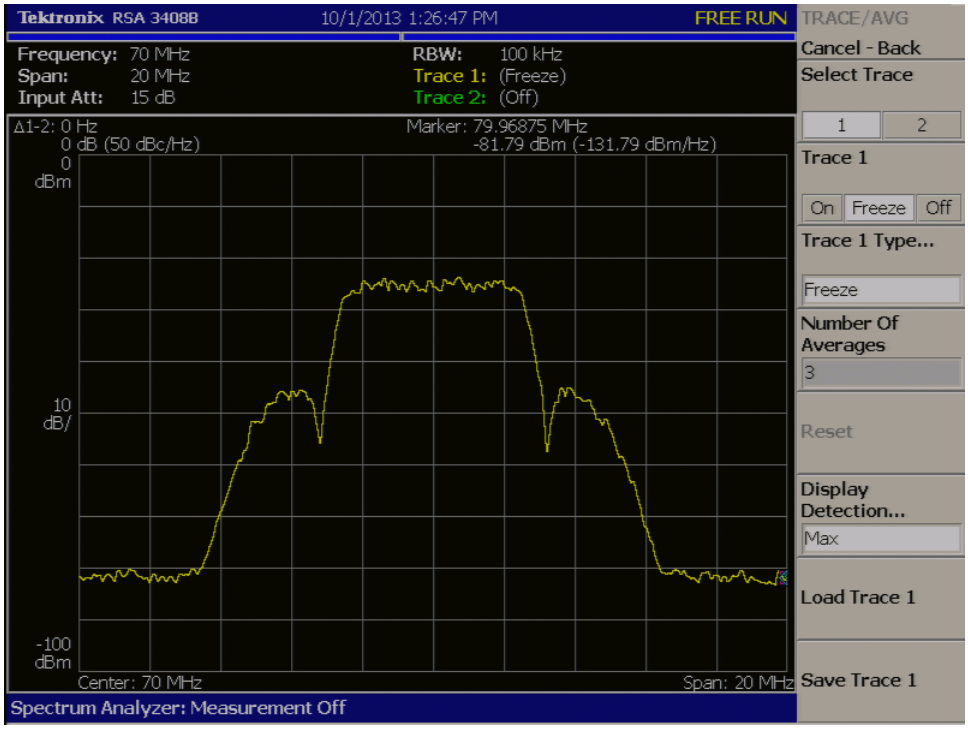

Figure 7.2: Spectrum analyzer display for transmit signal with no multipath.

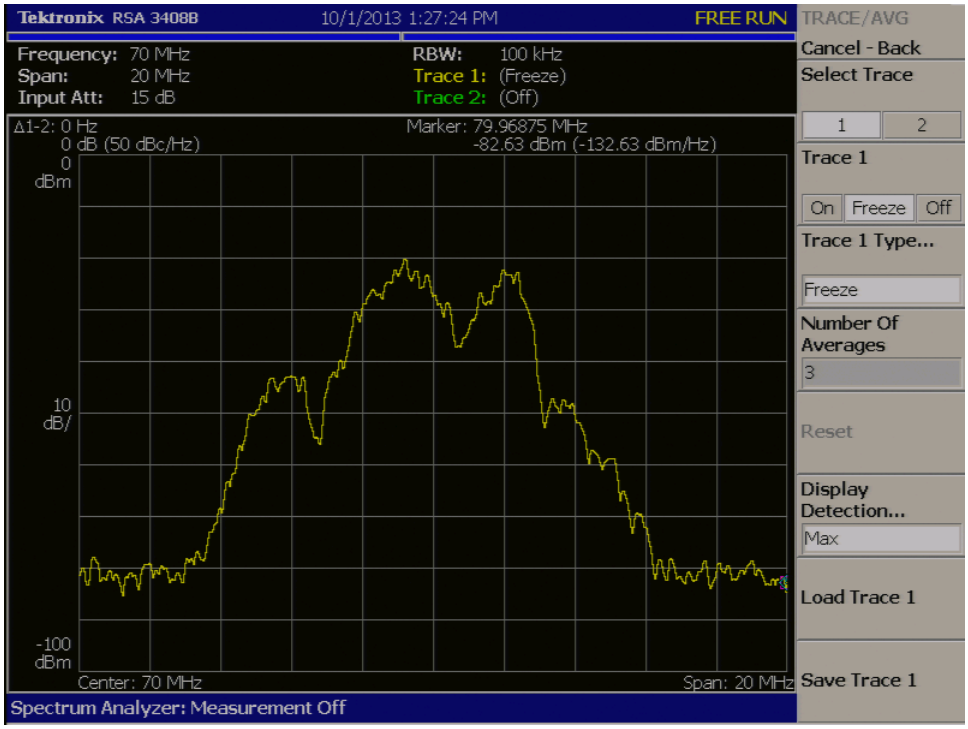

Figure 7.3: Spectrum analyzer display for transmit signal with multipath.

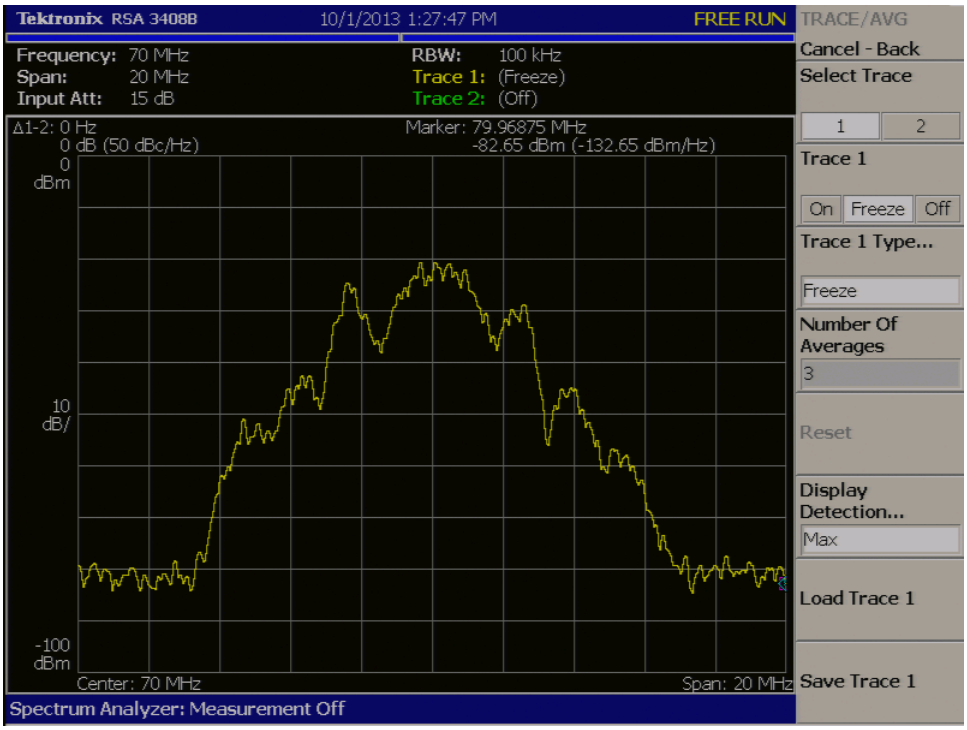

Figure 7.4: Spectrum analyzer display for transmit signal with multipath.

tap magnitudes around the center. Though the anticipated response is simply $1 + j0$, the cluster of non-zero weights are best explained as the equalizer attempting to correct other path distortions, such as IF and baseband filtering. Figure 7.7 and Figure 7.8 are eye diagrams and tap weights for the static multipath case. The reference eye diagram in red (bottom trace) shows the demodulated output with no equalization, demonstrating the severe distortion experienced with a single symbol detector. The equalized case in blue (top trace) shows the effectiveness of the equalizer in deconvolving the channel distortion, and opening the eye pattern.

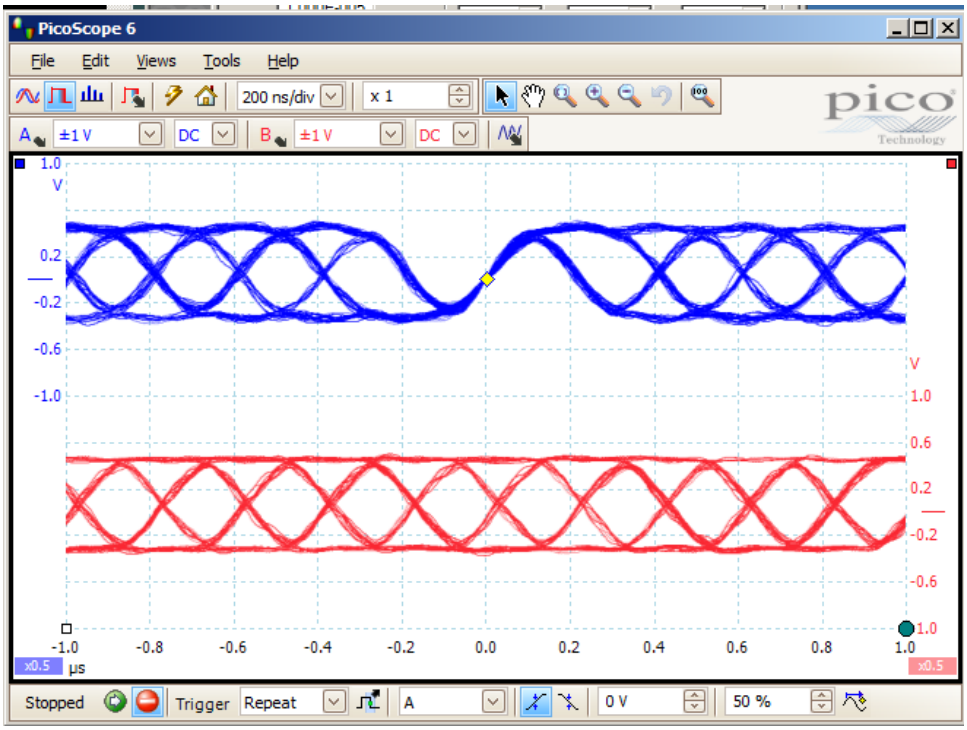

Figure 7.5: Oscilloscope eye pattern display before and after equalizer with no multipath.

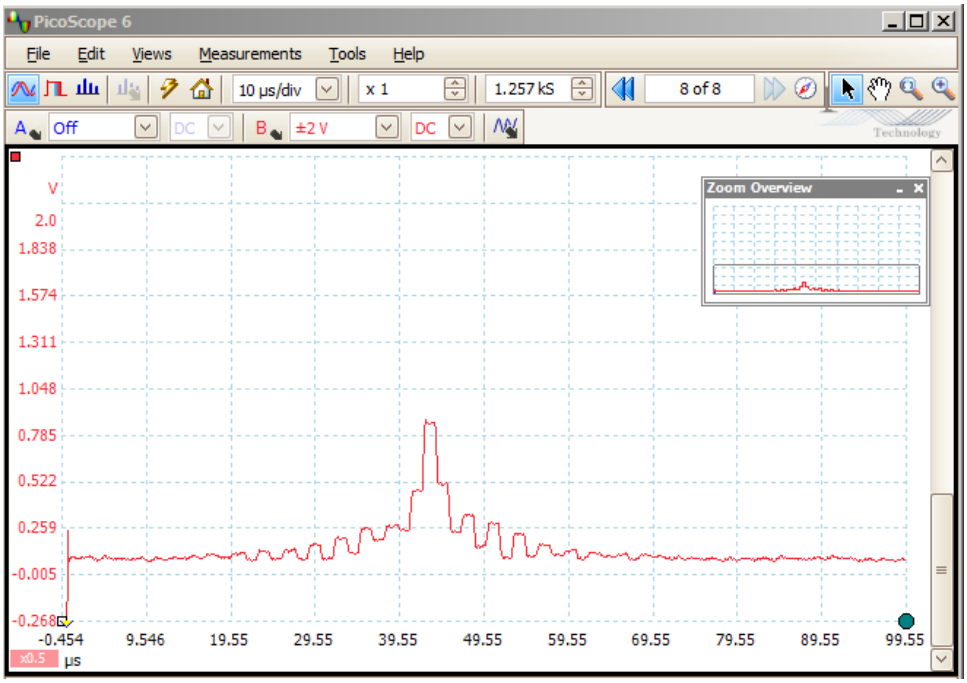

Figure 7.6: Oscilloscope tap display from equalizer with no multipath.

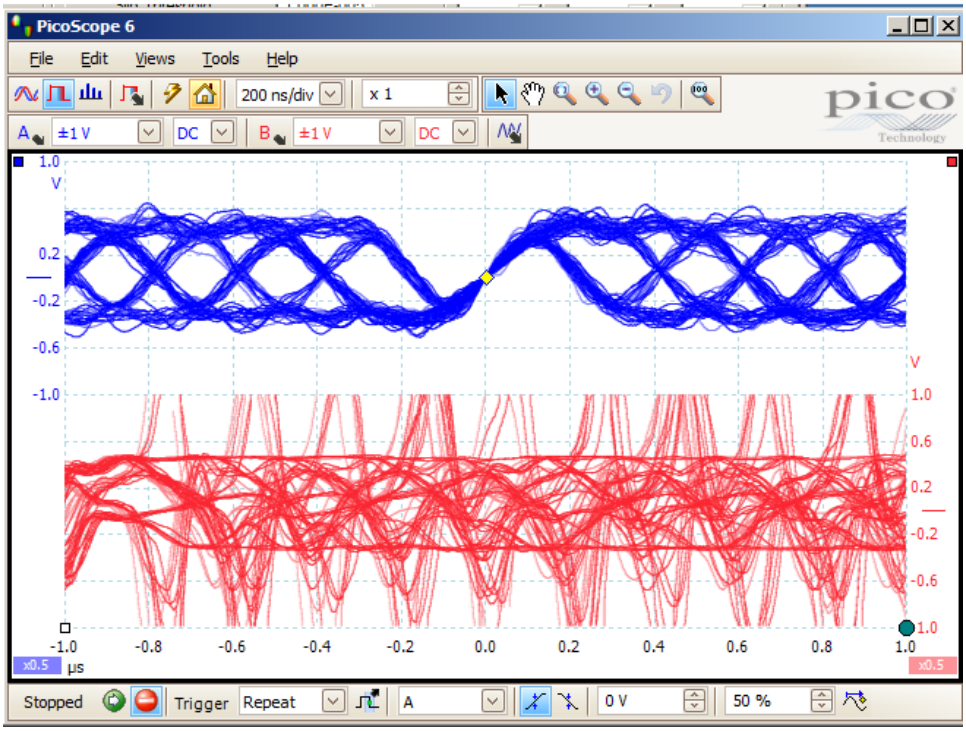

Figure 7.7: Oscilloscope eye pattern display before and after equalizer with multipath.

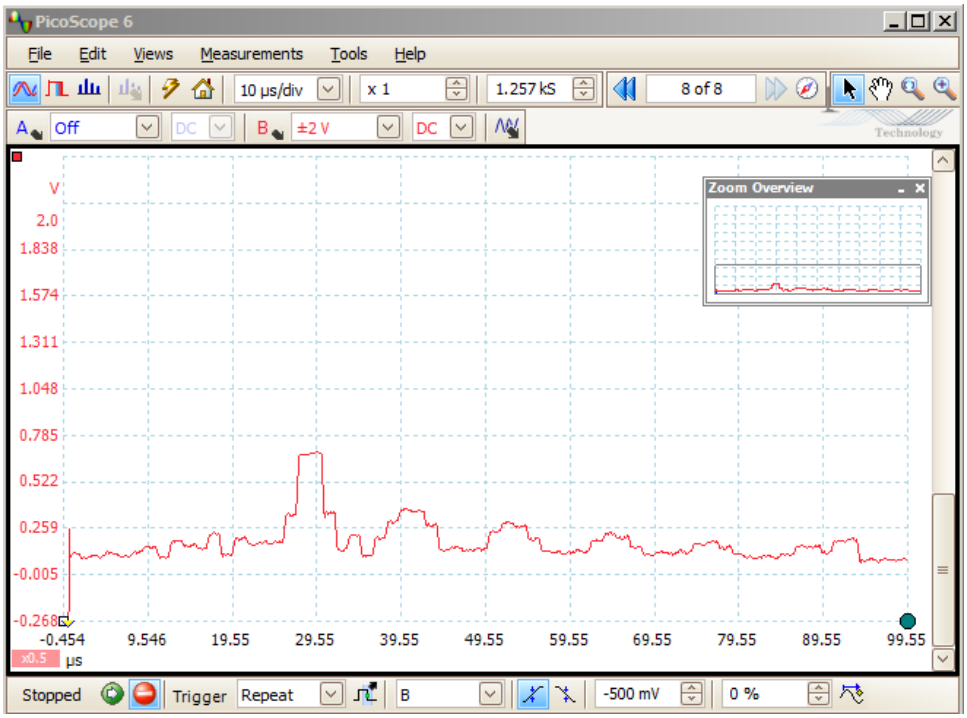

Figure 7.8: Oscilloscope tap display from equalizer with multipath.

7.4 Bit Error Rate Testing

A very common characterization of a receiver/demodulator's performance is the bit error rate curve. The BER curve x-axis is $\frac{E_b}{N_0}$, which is a normalized measurement of signal to noise ratio. The y-axis is bit error rate, expressed as a ratio. The graph of BER is typically drawn with the x-axis linear in decibels, and the y-axis logarithmic. The bit error rate curve in Figure 7.9 shows the measured BER values for the receiver/demodulator with the equalizer bypassed, and with the equalizer engaged. When the receiver/demodulator is operated with the equalizer in bypass, the unit deviates from the theoretical curve by less than 0.1 dB. When the equalizer is engaged, less than 1.0 dB degradation occurs due to the equalizer. This is due to the equalizer filter increasing the noise in the demodulator bandwidth.

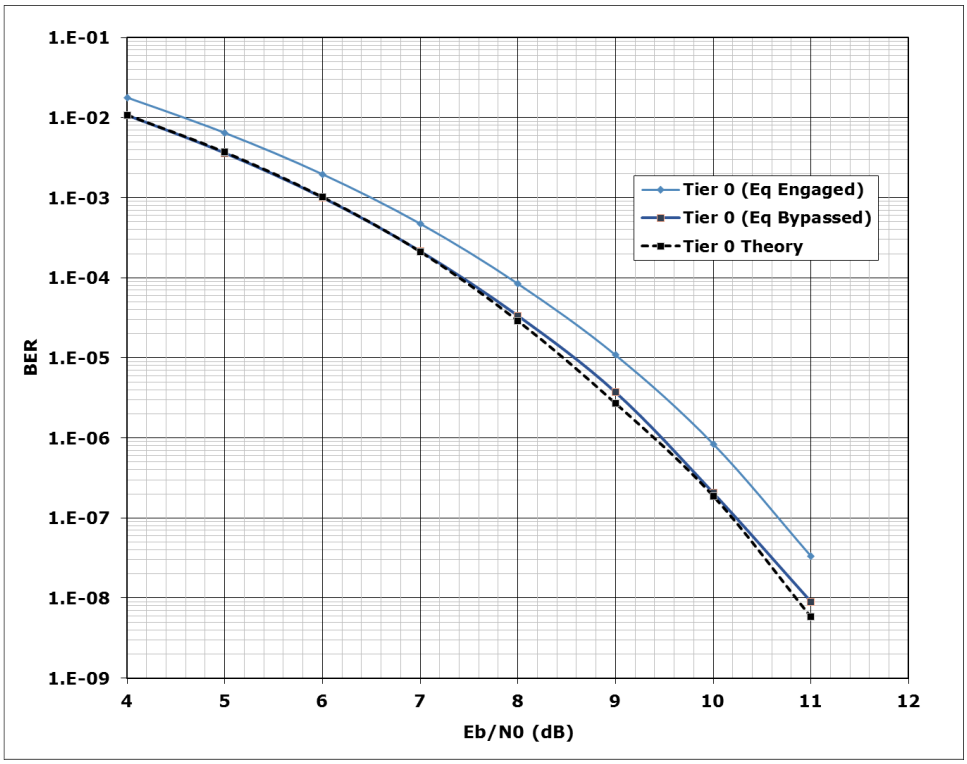

Figure 7.9: Bit Error Rate curve for the receiver with equalization.

7.5 Fmax and Data Rate

Fmax is a measure of the maximum clocking rate, and is reported by the Altera Compiler. The demodulator FPGA is clocked at 93.3333MHz, and therefore the compiled result must meet this value of Fmax as a minimum. The reported Fmax is 96.3MHz. As a result, the equalizer will support the full data rate capability of the receiver/demodulator, which is 23 Mbps for IRIG106 Tier 0, and 46 Mbps for Tier 1. Tier 2 was not tested.

CHAPTER VIII

CONCLUSIONS

Equalization has been proposed in many wireless systems and successfully deployed. Constant Modulus Algorithm equalization, which is a form of blind equalization, leverages the expectation that the signal envelope is constant, without the benefit of known data transmissions. In this project, it was shown that an FPGA implementation of a Constant Modulus Algorithm equalizer works well to restore constant envelope modulation formats described in IRIG 106, for airborne telemetry applications. An equalizer was designed and developed for a high density FPGA, and operated in lab and field environments. The equalizer was able to sufficiently restore a signal corrupted with a dynamic profile to result in an error-free link. The implementation is production ready, requiring no further development for field application.

The FPGA implementation was designed and developed using MATLAB Simulink, and downloaded to an Altera Stratix IV device, in an existing production digital receiver. The design was simulated and tested and met the full data rate capability of the receiver.

BIBLIOGRAPHY

[1] M. Geoghegan, “”Experimental Results for PCM/FM, SOQPSK, and multi-H CPM with CMA equalization”,” *Mobile Dev Design*, vol. 29, no. 1, pp. 511–546, November 2003.

[2] T. Hill and M. Geoghegan, “”A comparison of adaptively equalized PCM/FM, SOQPSK, and multi-h CPM in a multipath environment”,” *Proceedings of the International Telemetering Conference*, vol. 29, no. 1, pp. 511–546, October 2002.

[3] E. Law, “How well does a blind adaptive CMA equalizer work in a simulated telemetry multipath environment?” in *Proceedings of the International Telemetering Conference*, San Diego, CA, October 2004.

[4] M. Rice and M. Jensen, “A comparison of L-band and C-band multipath propagation at Edwards AFB,” in *Proceedings of the International Telemetering Conference*, Las Vegas, NV, October 2011.

[5] D. N. Godard, “Self-recovering equalization and carrier tracking in two-dimensional data communication systems,” *Proceedings of the IEEE*.

[6] R. Johnson *et al.*, “Blind equalization using the constant modulus criterion: A review,” *Proceedings of the IEEE*, vol. 86, no. 10, pp. 1927–1950, October 1998.

[7] M. Rice and E. Satorius, “A comparison of MMSE and CMA equalization techniques for ARTM Tier-1 waveforms,” in *Proceedings of the International Telemetering Conference*, San Diego, CA, October 2004.

[8] *IRIG Standard 106-09: Telemetry Standards*, Secretariat, Range Commanders Council, White Sands Missile Range, New Mexico, 2009, (Available on-line at http://www.irig106.org/docs/106-09).

[9] M. Rice and E. Satorius, “Equalization techniques for multipath mitigation in aeronautical telemetry,” in *Proceedings of the IEEE Military Communications Conference*, Monterey, CA, November 2004.

[10] M. Rice, “Phase 1 report: Preamble assisted equalization for aeronautical telemetry (PAQ),” Brigham Young University, Tech. Rep., 2014, submitted to the Spectrum Efficient Technologies (SET) Office of the Science & Technology, Test & Evaluation (S&T/T&E) Program, Test

Resource Management Center (TRMC). Also available on-line at http://hdl.lib.byu.edu/1877/3242.

[11] S. Haykin, *Adaptive Filter Theory*. New Jersey: Prentice-Hall, 1996.

[12] F. G. Stremler, *Introduction to Communications Systems*. Massachusetts: Addison-Wesley, 1992.

[13] R. Hardie, "Class notes, ece561–digital signal processing," University Lecture, Dayton, Ohio, 2013.

[14] A. V. Oppenheim and R. W. Schafer, *Digital Signal Processing*. New Jersey: Prentice-Hall, 1975.