BLIND EQUALIZATION FOR DIGITAL COMMUNICATION SYSTEMS: ALGORITHM AND IMPLEMENTATION

A thesis submitted
to the College of Graduate Studies and Research
in Partial Fulfillment of the Requirements
for the Degree of Master of Science
in the Department of Electrical Engineering
University of Saskatchewan

by
Zhijun Zhang

Saskatoon, Saskatchewan, Canada
Spring 2003

© Copyright Zhijun Zhang, 2003. All rights reserved.
PERMISSION TO USE

In presenting this thesis in partial fulfillment of the requirements for a Postgraduate degree from the University of Saskatchewan, it is agreed that the Libraries of this University may make it freely available for inspection. Permission for copying of this thesis in any manner, in whole or in part, for scholarly purposes may be granted by the professors who supervised this thesis work or, in their absence, by the Head of the Department of Electrical Engineering or the Dean of the College of Graduate Studies and Research at the University of Saskatchewan. Any copying, publication, or use of this thesis, or parts thereof, for financial gain without the written permission of the author is strictly prohibited. Proper recognition shall be given to the author and to the University of Saskatchewan in any scholarly use which may be made of any material in this thesis.

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

Head of the Department of Electrical Engineering
57 Campus Drive
University of Saskatchewan
Saskatoon, Saskatchewan, Canada
S7N 5A9
I would like to express my gratitude to my supervisors, Professor Kunio Takaya and Professor David E. Dodds, for their support and guidance.

The Telecommunications Research Laboratories (TRLabs) is also gratefully acknowledged for providing financial support and equipment for this research. I would like to thank Mr. Jack Hanson, Mr. Garth Wells and all other faculty, staff, and students of TRLabs in Saskatoon for their support.

I also appreciate Prof. Ronald J. Bolton for allowing me use the simulation and design software in the VLSI laboratory, which significantly speeded up the progress of this project.
ABSTRACT

This thesis presents a study on the blind equalization for high speed digital communication systems. The thesis can be divided into two parts, the first part involves the theoretical issues about the blind equalization, the second part addresses the problem of practical implementation.

The thesis begins with some basic concepts and theories of the channel equalization, which are fundamentals for designing both trained equalizers and blind equalizers, and then introduces a blind equalization algorithm called the Constant Modulus Algorithm (CMA). Next presented in the thesis is a decision-directed algorithm that is suitable for blind equalizations. Its stability and convergence properties are analyzed in the thesis. Also, the thesis proposes a hybrid blind equalization scheme that combines the decision-directed algorithm and the CMA, which can take advantages of both them. Numerical simulations with QAM signals have been carried out to test the performances of the new algorithm.

The second part of the thesis presents a hardware equalizer design which has be
implemented in a FPGA chip. To simulate the work of the equalizer in real-time, I also designed a channel simulator and a channel simulator, which, together with a high-frequency signal generator, consists a test-bed for real-time simulation. Based on the results from the real-time simulation, the performance of the hardware equalizer and the test-bed is discussed. Finally, some suggestions for further work are included at the end of the thesis.
Table of Contents

PERMISSION TO USE i
ACKNOWLEDGMENTS ii
ABSTRACT iii
TABLE OF CONTENTS v
LIST OF FIGURES vi
ABBREVIATIONS AND ACRONYMS vii

1 INTRODUCTION 1
  1.1 Research Motivation ................................ 1
  1.2 Research Objectives ................................ 2
  1.3 Thesis Organization ................................ 2

2 Background 4
  2.1 Communication System and Signal Fundamentals .......... 4
    2.1.1 Modulation ...................................... 4
    2.1.2 Baseband Representation of Signals ............... 7
  2.2 The concept of inter-symbol interference (ISI) .......... 10
  2.3 A Summary of Equalization Techniques .................. 12
  2.4 Considerations on Implementation ....................... 13

3 Channel Equalization Theory 15
  3.1 A Baseband System Model ............................ 15
6.1.1 FPGA architecture ............................................. 62
6.1.2 FPGA Programming .......................................... 65
6.2 Arithmetic Circuit Element ..................................... 68
  6.2.1 Two’s Complement Number ................................ 68
  6.2.2 Digital Adder and Subtractor .............................. 70
  6.2.3 Digital Multiplier .......................................... 72
6.3 Hardware Design of Equalizer ................................. 76
  6.3.1 Overall Architecture ...................................... 76
  6.3.2 Channel Equalizer ......................................... 78
  6.3.3 Demodulator and Carrier Recovery Loop .................. 86
7 Real Time Simulation System Design ................................ 90
  7.1 Test-bed ...................................................... 90
  7.2 Channel Simulator .......................................... 91
8 Discussion on the Real Time Simulation Results .............. 94
9 Conclusion and Future Work ....................................... 102
List of Figures

2.1 Block diagram of a communication system .......................... 5
2.2 QAM modulator and demodulator ................................. 5
2.3 QAM signal constellation ......................................... 6
2.4 Impulse response of a raised cosine filter .......................... 8
2.5 Spectrums of bandpass and baseband signals ....................... 9
2.6 A multipath wireless channel: received signal is a sum of multiple signals from different propagation paths .................. 11

3.1 Frequency response of a bandpass channel ......................... 15
3.2 A baseband equivalent channel model .............................. 18
3.3 A baseband equivalent system model ................................ 19
3.4 A FIR filter structure ............................................. 20
3.5 A FSE system model .................................................. 25
3.6 An adaptive equalizer ............................................... 31
3.7 An adaptive FIR filter ............................................... 31

4.1 The CM circle of 16QAM signal .................................... 36
4.2 The magnitude of the channel impulse response ................... 42
4.3 The constellation of the original received signal .................. 43
4.4 The eye diagram of the original signal ............................. 44
4.5 The convergence curve of the CMA-FSE in terms of ISI .......... 45
4.6 The convergence curve of the CMA-FSE in terms of normalized MSE

4.7 The magnitude the combined channel and equalizer impulse response

4.8 The eye diagram of the equalized signal

4.9 The constellation of the equalized signal

4.10 The convergence curve of the CMA-TSE in terms of ISI

4.11 The convergence curve of the CMA-TSE in terms of normalized MSE

5.1 Auto-feedback loop of convergence

5.2 A hybrid equalization scheme

5.3 Convergence curves of the algorithms

5.4 Signal constellations: (a) original received signals, (b), (c), and (d) equalized by the CMA, the decision-directed and the hybrid algorithms respectively.

6.1 The architecture of FPGA

6.2 The structure of CLB

6.3 The interconnection between CLB (CLE)s

6.4 FPGA design flow

6.5 The architecture of ripple carry adder

6.6 The symbol and the truth table of full adder

6.7 The architecture of carry lookahead adder

6.8 A 4-bit unsigned array multiplier

6.9 A carry save multiplier
The signal constellation of the equalizer output during \((15000/4)\)th \((20000/4)\)th symbols

The signal constellation of the equalizer output after \((20000/4)\)th symbols
<table>
<thead>
<tr>
<th>Abbreviation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AGC</td>
<td>Automatic Gain Controller</td>
</tr>
<tr>
<td>ASIC</td>
<td>Application Specific Integrated Circuit</td>
</tr>
<tr>
<td>BER</td>
<td>Bit Error Rate</td>
</tr>
<tr>
<td>CLB</td>
<td>Configurable Logic Block</td>
</tr>
<tr>
<td>CM</td>
<td>Constant Modulus</td>
</tr>
<tr>
<td>CMA</td>
<td>Constant Modulus Algorithm</td>
</tr>
<tr>
<td>DSP</td>
<td>Digital Signal Processor</td>
</tr>
<tr>
<td>FIR</td>
<td>Finite Impulse Response</td>
</tr>
<tr>
<td>FPGA</td>
<td>Field Programmable Gate Array</td>
</tr>
<tr>
<td>FSE</td>
<td>Fractionally Spaced Equalizer</td>
</tr>
<tr>
<td>HOS</td>
<td>Higher Order Statistics</td>
</tr>
<tr>
<td>IIR</td>
<td>Infinite Impulse Response</td>
</tr>
<tr>
<td>ISI</td>
<td>Inter-Symbol Interference</td>
</tr>
<tr>
<td>LMS</td>
<td>Least Mean Square</td>
</tr>
<tr>
<td>MMSE</td>
<td>Minimized Mean Squared Error</td>
</tr>
<tr>
<td>MSE</td>
<td>Mean Squared Error</td>
</tr>
<tr>
<td>NMSE</td>
<td>Normalized Mean Squared Error</td>
</tr>
<tr>
<td>QAM</td>
<td>Quadrature Amplitude Modulation</td>
</tr>
<tr>
<td>RTL</td>
<td>Register Transfer Level</td>
</tr>
<tr>
<td>SNR</td>
<td>Signal to Noise Ratio</td>
</tr>
<tr>
<td>SOS</td>
<td>Second Order Statistics</td>
</tr>
<tr>
<td>TSE</td>
<td>T-Spaced Equalizer</td>
</tr>
<tr>
<td>VLSI</td>
<td>Very Large Scale Integration</td>
</tr>
</tbody>
</table>
1. INTRODUCTION

1.1 Research Motivation

High-speed digital communication systems are usually very sensitive to the channel distortion that arises from the non-ideal characteristics of the transmission media, e.g. multi-path fading. A severe problem associated with channel distortion is inter-symbol interference (ISI), which corrupts the signals that arrive at the receiver and hence increases the bit error rate (BER) and imposes limitations on the channel capacity (or bandwidth). In order to remove ISI, certain kinds of channel equalization technologies must be used in receivers to counteract the negative effects of the channel distortion.

Traditionally, channel equalization is achieved by an adaptive equalizer whose parameters are estimated with the training data sequences sent from the transmitters. In most cases, this form of equalization can perform very well in terms of mean square error (MSE), robustness and convergence speed. However, the burden of transmitting training sequences decreases the bandwidth efficiency of the system. This drawback becomes particularly intolerable in the case of the rapidly changing channels for which the periods between training sequences must be very short to ensure that the equalizers work.

Blind equalizations are techniques to equalize the channel without employing training sequences. Blind equalizations are desirable because in many circumstances it is either too expensive or impossible to send training signals, for example in multi-point communication networks or broadcasting systems. In recent years, blind equalizations have gained more and more practical usage, for example, in wireless LAN, xDSL, Cable and DTV systems [3].
1.2 Research Objectives

The objective of this research is to design a blind equalizer and implement it in hardware for high speed digital communication systems. The targeted applications of the equalizer may have a symbol rate of 6 to 15 MS/s (Mega symbol per second), which is the typical symbol rate of broadband access systems such as cable modem and wireless LAN. Because the targeted application works at such a high symbol rate, the equalization algorithm should be simple and fast, and also easy to implement. For the hardware implementation, we need to explore a suitable hardware architecture which can achieve the functions of our design effectively and economically. We also need to build a low-cost experimental bench to test the performance of the hardware implementation.

1.3 Thesis Organization

This thesis is organized into 9 chapters.

Chapter 1 states the motivation and the objectives of this research.

Chapter 2 provides some background information for this research, which includes a brief introduction to signals and communication systems and, a concise explanation of the concept of inter-symbol interference (ISI), a brief summary of some existing channel equalization techniques and some considerations for implementation.

Chapter 3 provides the basic mathematical framework of channel equalization. We will discuss a baseband channel model and two different types of linear equalizer structures: baud-spaced or T-spaced equalizer (TSE) and fractionally-spaced equalizer (FSE). We will specify conditions and constraints on the channel and the length of the equalizer for perfect channel equalization. Chapter 3 also includes a brief description of the adaptation algorithms for channel equalizations.

An introduction of the constant modulus algorithm (CMA) is given in Chapter 4, where the convergence property of the CMA is discussed. Initialization plays a very important role in the convergence of the CMA equalizer; in Chapter 4 we also discuss
the conditions imposed on the initialization of CMA equalizer. There is a numerical example in Chapter 4, which shows the performance of the CMA equalizer in simulation for a digital wireless channel.

In Chapter 5, a hybrid equalization scheme is suggested to improve the convergence of a blind equalizer. The hybrid scheme first uses the CMA to initialize the equalizer and then switches to a decision directed algorithm, which has faster convergence and smaller residual errors than the CMA under certain conditions. The stability and robustness of this decision directed algorithm are also analyzed in Chapter 5.

Chapter 6 describes the detailed implementation of the blind equalizer. Since a channel equalizer is usually integrated into the baseband demodulator in a receiver, a QAM demodulator that works with the equalizer is also designed. The VLSI architecture and the RTL code of the hardware design are given.

Chapter 7 describes a test bench for the hardware implementation. A bandpass channel simulator is designed to simulate the real channel for our experimental system.

Chapter 8 presents the test results of the experimental system described in Chapter 7. The results are then discussed.

Chapter 9 summarizes the results of the research and suggestions for future work.
2. Background

2.1 Communication System and Signal Fundamentals

A communication system consists of three basic components, a transmitter, a transmission medium or channel, and a receiver as shown in Fig. 2.1. At the transmitter side, the source signal is first processed by the modulator and then up-converted to a high frequency RF signal that is finally sent into the channel. At the receiver side, the received signal is first down-converted to a low frequency or intermediate frequency signal and then processed by the demodulator to recover the source signal. The channel may be any wire-line or wireless transmission medium, e.g. copper, fiber or radio frequency over air.

2.1.1 Modulation

In a digital communication system, the information we want to send and receive is represented in binary numbers i.e. '0' and '1'. One or more bits of a binary number may constitute a symbol. A modulator then converts the symbols into a form suitable for transmission over the channel. This is usually a signal waveform that has a one-to-one correspondence with the symbols. If a symbol only has one bit, the bits '0' and '1' are mapped to two waveforms $P_0(t)$ and $P_1(t)$ respectively, it is called the binary modulation signalling. If there are $K$ bits per symbol resulting in a total set of $M = 2^K$ symbols, each mapped to a distinct waveform $P_i(t)$, it is called M-ary modulation signalling.

One widely-used M-ary modulation is the pulse amplitude Modulation (PAM), which uses pulses of regularly spaced amplitude as modulating waveform $P$. In PAM modulation, the pulse interval is the same as the symbol interval $T$ ($T = 1/R$, where $R$ is the symbol rate), while the pulse amplitude changes with different symbols.
Figure 2.1 Block diagram of a communication system

Figure 2.2 QAM modulator and demodulator
There is one to one mapping from the symbol set to the pulse amplitude set, which is a set of real numbers located symmetrically about the origin, e.g.

\[ A = -(M - 1)/2, \ldots, -1/2, 1/2, \ldots, (M + 1)/2 \]

Another very popular M-ary modulation scheme is the quadrature amplitude modulation (QAM) as shown in Fig. 2.2, which combines two amplitude-modulated (PAM) signals into a single channel and hence doubles the effective bandwidth. One signal in QAM modulation is called the I (In-phase) signal, and the other is called the Q (Quadrature) signal. To modulate these two signals, we need two carrier signals that are orthogonal to each other. These two orthogonal signals can be two cosine waves of the same frequency, but there must be a phase difference of 90 degrees between them. Because the transmitted signal in QAM is a combination of two orthogonal signals, the transmitted symbols can be regarded as complex numbers or as real 2-tuples in real 2-space \( \mathbb{R}^2 \). A standard \( M \times M \) order QAM signal set can form a square array of \( M \times M \) signal points located symmetrically about the origin, as illustrated in Fig. 2.3. The signal point array as shown in Fig. 2.3 is also called signal constellation.
2.1.2 Baseband Representation of Signals

A QAM baseband signal can be defined as a complex pulse sequence as follows

\[ u(t) = \sum s(n)g(t - nT), \]  

(2.1)

where \( s(n) \) is the complex source symbol that consists of an In-phase signal \( I(n) \) and a quadrature signal \( Q(n) \): \( s(n) = I(n) + jQ(n) \), \( u(t) \) is also a complex signal: \( u(t) = u_I + ju_Q \), \( g(\cdot) \) represents a certain pulse waveform. The \( g(\cdot) \) is actually achieved by a pulse-shaping filter, so the \( u(t) \) can be regarded as the result of applying a pulse-shaping filter to the source symbol sequence. Because the channel is band-limited, the pulse-shaping filter must ensure that \( u(t) \) is band-limited. In practice, often a low-pass filter with a cut-off frequency of \( 1/2T \) is used for pulse shaping. A filter of such a kind is usually referred to as a Nyquist Filter. Practical considerations dictate a gradual roll-off where the half voltage gain point occurs at frequency \( 1/2T \). The most common pulse-shaping filter is raised cosine roll-off filter, which has an impulse shape as shown in Fig. 2.4. With pulse shaping, the power spectrum of \( u(t) \) only consists of low-pass components centred around \( 0Hz \) (Fig. 2.5), which are called baseband components.

According to Fig. 2.2, the QAM signal after modulation is represented by

\[ \tilde{u}(t) = u_I(t) \cos(w_c t) - u_Q(t) \sin(w_c t) \]  

(2.2)

where \( w_c \) is the carrier frequency. After modulation, the complex baseband signal, \( u(t) = u_I(t) + ju_Q(t) \), is converted to a real bandpass signal (Fig. 2.5). A bandpass signal is a signal whose one-sided energy spectrum is centred at a non-zero frequency, say \( w_c \) and does not extend in frequency to \( 0Hz \) (DC). A bandpass signal is real valued and consequently the power spectrum will always be symmetric around zero.

Eq. 2.2 can also be written as

\[ \tilde{u}(t) = \Re\{u(t)e^{j\omega_c t}\} \]  

(2.3)

\[ = \frac{1}{2}(u(t)e^{j\omega_c t} + u^*(t)e^{-j\omega_c t}), \]  

(2.4)
Figure 2.4  Impulse response of a raised cosine filter

where $\Re$ denotes the real part of a complex signal. Using the following properties of Fourier Transform:

$$F[x^*(t)] = X^*(-w)$$  \hspace{1cm} (2.5)

and

$$F[x(t)e^{j\omega t}] = X(w - w_c)$$  \hspace{1cm} (2.6)

we can obtain the Fourier Transform of the equation 2.3:

$$\tilde{U}(w) = \frac{1}{2}(U(w - w_c) + U^*(-w - w_c))$$  \hspace{1cm} (2.7)

The above equation states that the Fourier transform of a bandpass signal is simply derived from the spectrum of the complex baseband signal. Fig. 2.5 shows the spectrum of a bandpass signal, the spectrum for positive frequency, say $|U(w - w_c)|$, is obtained by shifting its baseband equivalent spectrum $|U(w)|$ by $w_c$ and scaling the amplitude by $1/2$, the spectrum for negative frequency is the mirror image of
Figure 2.5 Spectrums of bandpass and baseband signals
\[ |U(w - \omega_c)| \] with respect to \( w = 0 \). Since the complex exponential function in Eq. 2.3 only determines the center frequency, all the information in \( \hat{u}(t) \) is derived from the complex baseband signal \( u(t) \), so it is possible to analyze a communications system based on this complex baseband signal. Analysis of a system in baseband is easier than in bandpass, because it does not need to deal with the high frequency signal components contained in the bandpass signal.

### 2.2 The concept of inter-symbol interference (ISI)

As a signal travels in the channel, it is inevitably affected by various kinds of noise and fading in the channel. For example, in a wireless system whose channel is the free space between the transmitter and the receiver (Fig. 2.6), objects in propagation path can create multiple echoes on the transmitted signal. These echoes arrive at the receiver at different times and overlap to form a distorted version of the original signal. This effect of multiple echoes is referred to as multi-path fading. In frequency domain, multi-path fading results in a non-uniform frequency response of channel, i.e. some frequency components in the transmitted signal are enhanced and other frequency components are attenuated.

One problem associated with a multi-path fading channel is inter-symbol interference (ISI), which refers to the interference to the current symbol by preceding symbols. Inter-symbol interference, which degrades the error-rate performance of the receiver, is a significant problem in data communication systems. Mathematically, ISI can be illustrated by using a linear system model as below:

\[
x(t) = s(t) * c(t) + n(t), \tag{2.8}
\]

where \( x(t) \) is the received signal, \( c(t) \) is the impulse response of the channel, \( s(t) \) is the source signal, \( n(t) \) is the noise, which is usually regarded as a Gaussian white noise signal. This system model is a simplified baseband model that represents a total system from the source to the receiver. It includes the effects of the modulator, the propagation channel, and also the demodulator.
Figure 2.6 A multipath wireless channel: received signal is a sum of multiple signals from different propagation paths.

If $c(t)$ is an ideal or non-dispersive channel, i.e., an all-pass filter or a $\delta$ function in time domain, there is no ISI introduced to the received signal because a convolution of any signal with a $\delta$ function is the signal itself. However, many practical channels have multi-path transmission that introduces ISI to the received signal due to the memory effect of the channel impulse response. To understand this, let us consider a digital PAM transmission system. Based on the channel model described by Eq. 2.8, the received signal is given by

$$x(t) = \sum_{n=0}^{\infty} s(nT)c(t - nT) + n(t),$$

where $T$ is the symbol space (interval). The received analog signal waveform is sampled periodically and converted to digital form. We are interested in the signal value at $mT$, where, again, $T$ is the symbol space. Replacing $t$ with $mT$ in Eq. 2.9, we have

$$x(mT) = s(mT)c(0) + \sum_{n \neq m} s(nT)c((m - n)T) + n(mT).$$
From the above formula, we can see the received signal is a weighted sum of the desired signal component from the symbol $s(m)$ (1st term) and the contributions from the other symbols (2nd term). The 2nd term is unwanted inference to the current symbol from the other symbols, which is just what we call inter-symbol interference.

2.3 A Summary of Equalization Techniques

One method to resolve the problem of ISI is to apply a special filter to the received signal as a compensator for the dispersive channel. Such a filter is called the channel equalizer. If the channel impulse response is already known to the receiver, an equalizer can be built directly from the inverse of the channel [2]. In most cases, the channel is not priori known and is possibly time-varying, so adaptive techniques need to be used in practical communication systems. A classical adaptive technique for channel equalization is using an adaptive filter with a training signal sent periodically from a transmitter [1]. The training signal is a priori known special data sequence which is also stored in the receiver. When the transmitter is sending training signal, the receiver compares the received signal with a locally stored replica of the training signal and uses the error between these two signals to adjust the coefficients of the adaptive filter. A channel equalizer of this type is called a trained equalizer. Another type of equalizer, which is called a blind equalizer, does not use any training signals. The blind equalizer is the focus of this study.

Most blind equalization techniques use higher-order statistics (HOS) of the source signals, i.e. 4th-order moment, 3rd-order and 4th-order cumulants [4][5][6][9]. There are also some new techniques developed based on second order statistics (SOS)[7][8], but currently they are seldom used in practice because, so far, theory on the convergence of SOS-based techniques has not been established. The most popular blind equalization technique may be the constant modules algorithm (CMA) [12], which already has been realized in hardware for practical applications such as HDTV receivers [3]. The CMA can be classified as a HOS technique, for it actually makes an indirect use of the higher-order statistical information of source signals. A number
of in-depth studies have proven the convergence and robustness properties of CMA [13][14][15][17].

2.4 Considerations on Implementation

Generally speaking, a signal processing algorithm can be implemented in two forms: software and hardware. The choice of which form being taken for implementation depends on the requirements of cost, flexibility, speed for the targeted application. Sometimes, it is hard to find a solution that perfectly satisfies all the above mentioned requirements, so people have to seek a trade-off in making a choice.

Software implementation is often done by using a digital signal processor (DSP), which is an enhanced microprocessor specially designed for digital signal processing. The advantages of the DSP are high flexibility and low cost. It is also easy to use, because a DSP chip can be programmed in assembly language and/or C just like a general purpose microprocessor. An example of a DSP is Texas Instrument's TMS320C6x DSP processor, which is widely used in general signal processing and digital communication systems. Some newest DSP chips can provide a operation speed up to 4,000 MIPS, which is already high enough for most narrow bandwidth (below 1Mbps) communications applications, such as a cellular phone base station or a voice codec.

However, for broadband systems, most existing DSP processors are not fast enough. This is because of the sequential nature of these DSP processors. Only having a single CPU core, a DSP processor can perform only one operation at a time. In order to achieve a high processing speed beyond the limits of DSP processors, we need to implement algorithms in dedicated hardware. Hardware implementation is based on hardware logic and does not suffer from the sequential nature of the software-based processing. It allows applications to run in parallel so that a 64-tap filter can run as fast as a 1-tap filter. Hardware DSP systems can offer 10 to 1000 times faster performance than the most advanced digital signal processor at similar or even lower costs.
For hardware implementation, there are mainly two choices: ASIC (Application Specific Integrated Circuit) and FPGA (Field Programmable Gate Array).

An ASIC is an integrated circuit chip designed for a special application, it is custom manufactured by a semiconductor manufacturer. ASICs are cheap in large volumes and can achieve the best performance as determined by the semiconductor manufacture technologies, but they can not be reconfigured for new applications once manufactured.

A FPGA is an integrated circuit that can be programmed in field after it is manufactured. A FPGA’s function is defined by a program written by the designer not the device manufacturer. The designer can also reprogram a FPGA to change the hardware function it performs, so the FPGA is a reconfigurable device that is very suitable for those applications that require a high degree of flexibility. Since the architectures of FPGAs are optimized for reconfigurability but not for speed, FPGAs are somewhat slower than ASICs. Many hardware designers use FPGAs as prototypes for their ASIC designs before the designs are error-free for manufacturing.

In this study, we implement the equalizer in a FPGA chip. The hardware is designed by using an industry standard hardware description language called Verilog. The resulting design can also be implemented in an ASIC chip with little modification.
3. Channel Equalization Theory

3.1 A Baseband System Model

In a communications system, a source signal is usually first transferred by the modulator into a bandpass signal with a higher center frequency (carrier frequency) and then is sent into the channel. In most cases, due to the limited frequency spectrum resource for transmission, a bandpass signal must be transmitted in a bandpass channel, i.e. a band-limited channel with passband centred around the carrier frequency as shown in Fig. 3.1.

![Figure 3.1 Frequency response of a bandpass channel](image)

Since a bandpass channel has a real-valued impulse response, the Fourier transform of a bandpass channel must be conjugate symmetrical about the origin. That means
the Fourier transform of a bandpass channel can be expressed in the form:

$$C_p(w) = \frac{1}{2}(C_L(w-w_c) + C_L^*(-w-w_c)), \quad (3.1)$$

where $C_L(w)$ is the Fourier transform of a low-pass channel with center frequency at zero. Transforming Eq. 3.1 from frequency domain to time domain, we can obtain the impulse response of a bandpass channel:

$$c_p(t) = \frac{1}{2}(c_L(t)e^{j\omega_c t} + c_L^*(t)e^{-j\omega_c t})$$

$$= \Re\{c_L(t)e^{j\omega_c t}\}, \quad (3.3)$$

where $c_L(t)$ is the impulse response of the low-pass equivalent channel $C_L(w)$. Very similar to the baseband representation of Eq. 2.3, the bandpass channel $C_p(w)$ can be described by its equivalent low-pass channel impulse response $C_L(w)$. The carrier frequency $w_c$ only affects the position of the spectrum.

For a bandpass channel without noise, a received signal can be expressed as

$$x_r(t) = \tilde{u}(t) * c_p(t), \quad (3.4)$$

where $\tilde{u}(t)$ is the transmitted bandpass signal as defined in Eq. 2.3. Then the Fourier Transform of the received signal $x_r(t)$ is given by

$$X_r(w) = \tilde{U}(w)C_p(w) = (U(w-w_c) + U^*(-w-w_c))(C_L(w-w_c) + C_L^*(-w-w_c)), \quad (3.5)$$

where the $\tilde{U}(w)$ and $U(w)$ are the transmitted bandpass signal and its baseband equivalent signal respectively. According to the properties of bandpass signals and channels, $U(w-w_c)$ and $C_L(w-w_c)$ are zero for $w \leq 0$, $U(-w-w_c)$ and $C_L(-w-w_c)$ are zero for $w \geq 0$. Therefore, Eq. 3.5 can be changed to

$$X_r(w) = U(w-w_c)C_L(w-w_c) + U^*(-w-w_c)C_L^*(-w-w_c), \quad (3.6)$$

and, in time domain, the received signal is given by

$$x_r(t) = \Re\{u(t) * c_L(t)e^{j\omega_c t}\} \quad (3.7)$$
Demodulation of the received signal is equivalent to multiplying the received signal by $e^{-j\omega t}$ and then put it through a matched filter (which is also a low-pass filter) for equalization, say $h(t)$. It is easy to obtain a mathematical expression of the demodulated signal as below:

$$x(t) = u(t) \ast c_L(t) \ast h(t)$$

$$= s(t) \ast (g(t) \ast c_L(t) \ast h(t)), \quad (3.8)$$

where $s(t)$ and $g(t)$, as defined in Chapter 2, are the source signal and the pulse-shaping filter, respectively. There is no carrier signal involved in Eq. 3.8 so that the effect of the channel on the demodulated signal $x(t)$ only lies in the low-pass equivalent channel $c_L(t)$. As shown in Fig. 3.2, the entire system from $s(t)$ to $x(t)$ can be described by a combination of pulse-shaping filter, low-pass equivalent channel and matched filter:

$$c(t) = g(t) \ast c_L(t) \ast h(t) \quad (3.9)$$

As the low-pass channel, $c(t)$ involves only baseband signals, it is often referred to as the baseband equivalent channel model of communication systems. Compared with a bandpass channel model, the baseband equivalent channel model is easier to use for the analysis and simulation of communication systems, because it is carrier frequency independent and works at lower frequency.

To study how an equalizer that works with its targeted channel, we need to build a baseband system model that includes both channel and equalizer. An equalizer is usually implemented at baseband in a communication receiver, because at baseband the equalizer only needs to process a low frequency baseband signal instead of a high frequency bandpass signal. Although a signal is transmitted in a bandpass channel, the effect of the bandpass channel on the received signal is fully reflected to its baseband equivalent channel model, an equalizer designed for the baseband equivalent
Figure 3.2 A baseband equivalent channel model
channel should have the same effect as a equalizer designed for the bandpass channel.

Fig. 3.3 illustrates a single-channel digital communication system with additive noise \( n(t) \). The channel \( c(t) \) is a baseband equivalent channel that represents a total system that consists of carrier subsystems, propagation channel and any filters at the transmitter that may or may not be known at the receiver (for example, pulse shaping filters). Also shown in Fig. 3.3 is a channel equalizer that operates on samples of the received signal.

The signal at the receiver end may be sampled at baud spaced intervals or fractionally spaced intervals (over-sampled). The scalar \( N \) in Fig. 3.3 is the ratio between the fraction spacing \( T/N \) and the baud spacing \( T \). Though a non-integer value may be used for \( N \) in the implementation, in practice \( N \) is usually an integer and typically \( N = 2 \). We always assume \( N \) to be an integer in the following parts of the thesis.

The received signal sequence \( z(k) \) is a distorted version of the source signal sequence \( s(k) \), due to the non-ideal channel characteristics and an additive noise signal \( n(t) \), which is typically assumed to be an independently and identically distributed (i.i.d.) Gaussian random process. The equalizer can be regarded as a special filter
Figure 3.4  A FIR filter structure

that compensates for the signal distortion introduced by the channel.

The structure of most equalizers is very similar to a linear transversal filter, which is made of a tapped delay line as shown in Fig. 3.4. Assuming that the delay period (interval) of each unity delay element is the same as the sampling period (interval), the transfer function of a linear transversal filter can be written in the form of $z$-transform, i.e. a function of the unity delay operator $z^{-1}$, such as

$$F(z) = f(0) + f(1)z^{-1} + f(2)z^{-2} + ...$$  \hspace{1cm} (3.10)

The filter coefficient or tap weight sequence $f(k) = f(0), f(1), f(2), ...$ is equivalent to the impulse response of the filter. If this sequence is of finite length, the filter is called a finite impulse response (FIR) filter, otherwise the filter is an infinite impulse response (IIR) filter. Since it is hard to guarantee the stability of IIR filters in adaptive systems, IIR filters are not used as widely as FIR filters in channel equalizations. In this study, only a FIR filter structure is used for our equalizer design.
### 3.2 Baud Spaced Equalizer Design

When \( N = 1 \) in Fig. 3.3, the received analog signal waveform is assumed to be sampled at baud spaced interval \( T \), i.e., at the same rate as the source signal sequence. The received signal sequence, \( x(k) = \{x(0T), x(1T), ..., x(kT), ...\} \), can be defined as the convolution of the source signal sequence \( s(k) \) and the discrete channel response sequence \( c(k) = \{c(0T), c(1T), ..., c((L_c - 1)T)\} \) of length \( L_c \). The received signal sequence is then

\[
x(k) = c(k) \ast s(k)
\]

and the output signal sequence of equalizer is

\[
y(k) = f(k) \ast x(k) = f(k) \ast c(k) \ast s(k).
\]

Let \( h(k) = f(k) \ast c(k) \) be the combined response of channel and equalizer. A zero ISI system is the ideal channel that means a pure delay, so a perfect channel equalizer or a zero-forcing equalizer should satisfy [2]:

\[
f(k) \ast c(k) = \delta(k - d),
\]

or in frequency domain,

\[
|F(w)C(w)| = 1.
\]

This means the combined system of channel and equalizer must be an ideal channel. The Eq. 3.13 can also be rewritten in the \( z \)-transform as

\[
H(z) = F(z) \cdot C(z) = z^{-d}.
\]

There is no such a finite-length \( z \)-transform \( F(z) \) that satisfies the above equation for non-trivial \( C(z) \). The solution, if it exists, tends to be of infinite length. In
general, a perfect or zero-forcing equalization solution for a baud spaced system is not achievable via a finite-length linear equalizer or FIR equalizer. Usually a truncated zero-forcing equalizer of finite of length is used in practice. However, a zero-forcing equalizer only performs well for those channels that have very little noise. This is because, if a channel has high attenuation at certain frequencies within its spectrum, the zero-forcing equalizer may excessively amplify the noise at these frequencies. This point can be easily justified by the definition of zero-forcing equalizer in frequency domain (Eq. 3.14).

A more practical solution for channel equalization is the Minimum Mean Squared Error (MMSE) equalizer, which takes the channel noise into consideration. To obtain a MMSE equalizer, we need to express the channel and the equalizer by their impulse response vectors: \( c = [c(0), c(1), ..., c(L_c - 1)]^T \) and \( f = [f(0), f(1), ..., f(L_f - 1)]^T \) respectively, where \( L_f \) is the length of the equalizer. The combined channel-equalizer impulse response vector, \( h = [h(1), h(2), ..., h(L_c + L_f - 1)]^T \), can be expressed as the inner product of a channel convolution matrix \( C \), which is associated with the channel vector \( c \), and the equalizer vector \( f \), i.e.

\[
h = C f, \tag{3.16}
\]

where

\[
C = \begin{bmatrix}
  c(0) \\
c(1) & c(0) \\
  \vdots & \ddots & \ddots \\
c(L_c - 1) & \vdots & c(0) \\
c(L_c - 1) & c(1) \\
  \vdots \\
c(L_c - 1)
\end{bmatrix} \tag{3.17}
\]

In the presence of a white Gaussian noise and a zero mean i.i.d. source signal sequence, the variance power of the error between the source and equalizer output can be expressed as
\[ \sigma_d^2 = (u_d - Cf)^H(u_d - Cf)\sigma_n^2 + f^Hf\sigma_n^2, \tag{3.18} \]

where \( u_d = [0, ..., 0, 1, 0, ..., 0]^T \) is a unit vector with zeros in all positions except index \( d \). The variance powers of the source signal and noise are represented by \( \sigma_d^2 \) and \( \sigma_n^2 \) respectively. The superscript \( 'H' \) means the conjugate transform of a matrix.

There are two contributors to the MSE in Eq. 3.18. The first term represents the error power contributed by ISI, while the second represents the error power contributed by noise, which is amplified by \( f^Hf \) (the squared \( l_2 \)-norm of the equalizer vector). A MMSE solution is an equalizer vector that minimizes Eq. 3.18. Such an equalizer vector \( f_{optm} \) is found to be

\[ f_{optm} = (C^HC + \frac{\sigma_n^2}{\sigma_d^2}I)^{-1}C^Hu_d \tag{3.19} \]

From the above formula, we can find that, in the absence of noise, the MMSE equalizer is \( C^{-1}u_d \), i.e. a delayed inverse of the channel. Under the presence of noise, the MMSE equalizer tries to make a compromise between a diminished ISI of the combined channel-equalizer system and an enhanced noise by the gain of the equalizer.

### 3.3 Fractionally Spaced Equalizer Design

When \( N \) is larger than 1 in Fig. 3.3, the received signal waveform is assumed to be sampled at a fractionally spaced interval \( T/N \) instead of \( T \). In this case, the received signal is over-sampled by \( N \) times the source signal sequence rate, so \( N \) is regarded as over-sampling factor.

Although the input sequence for a fractionally spaced equalizer (FSE) is \( T/N \) spaced, the FSE's output sequence has the same rate as the source sequence, so the FSE can be modeled as a parallel combination of \( N \) baud spaced equalizers and the output of FSE is the sum of all outputs of the baud spaced equalizers in this parallel structure. A general system model of FSE will be derived below. Based on our base-band equivalent system model in Fig. 3.3, the analog received signal \( x(t) \) can be expressed as:

23
\[ x(t) = s(t) \ast c(t) + n(t) \]
\[ = \sum_{i=0}^{\infty} s(i) \delta(t - iT) \ast c(t) + n(t) \]
\[ = \sum_{i=0}^{\infty} s(i)c(t - iT) + n(t) \quad (3.20) \]

The \( x(t) \) is then sampled at intervals \( T/N \) to its discrete time equivalent signal \( x(mT/N) \)

\[ x(mT/N) = \sum_{i=0}^{\infty} s(i)c(mT/N - iT) + n(mT/N) \]
\[ = \sum_{i=0}^{\infty} s(i)c((m - iN)T/N) + n(mT/N) \quad (3.21) \]

Here \( m \) is regarded as a time index that will be used to represent the samples in fractional space. As in the previous section, the index \( k \) is used to represent the samples in the baud space, where the sample rate is the same as the input symbol rate. In fractional space, the output of the equalizer is given by

\[ y(mT/N) = \sum_{i=0}^{NL-1} f(i)x((m - i)T/N) \quad (3.22) \]

Here, the \( f(i) \) is the equalizer response sequence or tap weight sequence with spacing of \( T/N \). The length of the FSE is assumed to be \( NL_f \). As we need only one output symbol corresponding to one source symbol, we down-sample the output of the FSE by a factor of \( N \) to obtain a baud rate sequence.

\[ y(k) = y(kT + N^{-1}T') \]
\[ = \sum_{i=0}^{NL_f-1} f(i)x((kN + N - 1 - i)T/N) \]
\[ = \sum_{i=0}^{NL_f-1} \sum_{q=0}^{N-1} f(iN + q)x((k - i)T + (N - q - 1)T/N) \quad (3.23) \]

Now, define \( f_q(i) = f(iN + q) \), \( x_q(i) = x(iT + (N - q - 1)T/N) \), then \( y(k) \) can be expressed as

\[ y(k) = \sum_{q=0}^{N-1} \sum_{i=0}^{N-1} f_q(i)x_q(k - i) \]
\[ = \sum_{q=0}^{N-1} f_q(k) \ast x_q(k) \quad (3.24) \]
From Eq. 3.21, we have

\[
x_q(k) = \sum_{i=0}^{\infty} s(i)c((kN + N - q - 1 - iN)\frac{T}{N}) + n((kN + N - q - 1)\frac{T}{N})
\]

\[
= \sum_{i=0}^{\infty} s(i)c((k - i)T + (N - q - 1)\frac{T}{N}) + n(kT + (N - q - 1)\frac{T}{N})
\]  \hspace{1cm} (3.25)

Letting \( c_q(i) = c(iT + (N - q - 1)\frac{T}{N}) \) and \( n_q(i) = n(iT + (N - q - 1)\frac{T}{N}) \), we can rewrite Eq. 3.25 as

\[
x_q(k) = \sum_{i=0}^{\infty} s(i)c_q(k - i) + n_q(m)
\]

\[
= s(k) \ast c_q(k) + n_q(k)
\]  \hspace{1cm} (3.26)

Now, with Eq. 3.26 and Eq. 3.24, the relation between the output of FSE equalizer and the source sequence can be expressed as

\[
y(k) = \sum_{i=0}^{N-1} f_q(i) \ast (c_q(i) \ast s(i) + n_q(i))
\]

\[
= \sum_{i=0}^{N-1} f_q(i) \ast c_q(i) \ast s(i) + \sum_{i=0}^{N-1} f_q(i) \ast n_q(i)
\]  \hspace{1cm} (3.27)

Obviously, the above formula leads to a system diagram as is shown in Fig. 3.5.
Since the channel in Fig. 3.5 can be seen as a system that has only one input sequence \( s(k) \) but multiple output sequence \( x_q(k) \), it is often modeled as a so-called Single Input Multiple Output (SIMO) system, and the combined channel-equalizer system is modeled as a multi-channel system. Each channel branch in the multi-channel system is usually referred to as a sub-channel and the equalizer connected with each sub-channel as a sub-equalizer. According to Eq. 3.27, the impulse responses of \( q^{th} \) sub-channel and sub-equalizer are \( c_q(k) \) and \( f_q(k) \), respectively. It should be noticed that, unlike baud-space equalizers, the sub-equalizers are no longer the inverse of their respective sub-channels. We will discuss about the design of FSE equalizer based on the multi-channel system model derived above.

### 3.3.1 Condition for perfect equalization: Constraint on the channel

Under the noiseless condition, the second term of Eq. 3.27 is zero, so the impulse response of the combined channel-equalizer system will be

\[
h(k) = \sum_{q=0}^{N-1} f_q(k) * c_q(k)
\]  

(3.28)

Taking the z-transform of both sides of Eq. 3.28, we can derive the transfer function of the combined channel-equalizer system

\[
H(z) = \sum_{q=0}^{N-1} F_q(z)C_q(z)
\]  

(3.29)

For a perfect zero-forcing equalizer solution, it must satisfy that

\[
H(z) = z^{-d}
\]

or

\[
F_0(z)C_0(z) + F_1(z)C_1(z) + ... + F_{N-1}(z)C_{N-1}(z) = z^{-d}
\]

(3.30)

Eq. 3.30 is known as a Bezout relationship [3]. The Bezout relationship directly leads to the perfect equalization requirement concerning sub-channel roots.
Specifically, for the existence of a (finite-length) zero-forcing equalizer, all the sub-channel polynomials $C_q(z)$ must not share a common root. For example, if the sub-channel polynomials have a common root, say $g(z) = 1 + pz^{-1}$, which can be factored out of all the sub-channel polynomials \{$C_0(z), C_1(z), ..., C_{N-1}(z)$\}, leaving \{\$C'_0(z), C'_1(z), ..., C'_{N-1}(z)$\}. Then Eq. 3.30 can be written as:

$$H(z) = g(z)H'(z) = z^{-d}$$

or

$$(1 + pz^{-1}) \sum_{q=0}^{N-1} C'_q(z) F_q(z) = z^{-d}$$  \hspace{1cm} (3.31)

However, there is no such a set of finite length polynomials \{\$F_q(z)$\} that can satisfy Eq. 3.31. Due to $g(z)H'(z) = z^{-d}$, $H'(z)$ must be an infinite length polynomial. Thus, a necessary condition for perfect equalization of fractionally spaced system is that there are no common roots among all the sub-channels.

**3.3.2 Condition for perfect equalization: Constraint on the Equalizer Length**

Like a baud spaced system, the FSE system can be expressed in a vector form. Let us define

$$c_q = [c_q(0), c_q(1), ..., c_q(L_c - 1)]^T$$  \hspace{1cm} (3.32)

as the coefficient vector that correspond to the impulse responses of the $q^{th}$ baud-spaced sub-channel of length $L_c$. Similarly, we define

$$f_q = [f_q(0), f_q(1), ..., f_q(L_f - 1)]^T$$  \hspace{1cm} (3.33)

as the coefficient vector of $q^{th}$ sub-equalizer which is a baud-spaced equalizer of length $L_f$. Now the baud-spaced impulse response vector of the overall combined channel-equalizer system can be formed by using $N$ baud-spaced convolution matrices:
\[ h = \begin{bmatrix} h(0) \\ h(1) \\ \vdots \\ h(L_c + L_f - 1) \end{bmatrix} = \begin{bmatrix} C_0 & C_1 & \cdots & C_{N-1} \end{bmatrix} \begin{bmatrix} f_0 \\ f_1 \\ \vdots \\ f_{N-1} \end{bmatrix}, \quad (3.34) \]

where

\[ C_q = \begin{bmatrix} c_q(0) & c_q(1) & \cdots & c_q(L_c - 1) \\ c_q(1) & c_q(0) & \cdots & \vdots \\ \vdots & \vdots & \ddots & \vdots \\ c_q(L_c - 1) & \cdots & c_q(0) & c_q(1) \end{bmatrix} \quad (3.35) \]

is the baud-spaced convolution matrix associated with sub-channel vector \( c_q \). The size of the matrices is \((L_c + L_f - 1) \times L_f\); \( L_c + L_f - 1 \) is the length of the sequence resulted from the convolution between the sub-channel and its respective sub-equalizer.

If we define \( C = [C_0 \ C_1 \ \cdots \ C_{N-1}] \) as given in Eq. 3.35 and \( f = [f_0 \ f_1 \ \cdots \ f_{N-1}]^T \), Eq. 3.34 can be written in a compact form:

\[ h = Cf \quad (3.36) \]

\( C \) is a \((L_c + L_f - 1) \times NL_f\) matrix and \( f \) is a \( NL_f \) column vector.

For a zero-forcing solution, the equalizer must make the combined channel-equalizer system satisfy \( H(z) = z^{-d} \) (\( d \) is a non-negative integer), which also means the impulse response vector of the overall combined channel-equalizer must be

\[ h = u_d = [0, \cdots, 0, 1, 0, \cdots, 0]^T \quad (3.37) \]

The \( u_d \) is a unit vector that has only one non-zero element in its \( d^{th} \) position. Since the length of \( h \) is \( L_c + L_f - 1 \), \( d \) must not be larger than \( L_c + L_f - 1 \).

Now we can find a solution for zero-forcing equalizer by solving the below equation:

\[ Cf = u_d \quad (3.38) \]
The above system equation is solvable if and only if $u_d$ lies in the column space of $C$. Therefore, the necessary and sufficient condition for zero-forcing solution of equalizer is that $u_d$ must lie in the column space of $C$. However, it has been shown that for complete identification of the channel, or in other words, for the existence of the perfect equalizer, the matrix $C$ must be full row-rank [10] [11]. This condition is sometimes referred to as strong perfect equalization. When the matrix $C$ is not full row-rank, the sub-channels of the multi-channel model share at least one common root. In this situation, as shown in the previous section, the existence of a perfect FIR equalizer is impossible. Therefore, for the existence of a zero-forcing FSE, the following conditions must be satisfied:

1. The matrix $C$ must have all linearly independent rows; and
2. $d$ must be chosen such that $u_d$ lies in the column space of $C$.

To satisfy the first condition, the number of columns of $C$ must be greater than or equal to the number of rows. As $C$ has a dimension of $(L_c + L_f - 1) \times NL_f$, the length of each sub-equalizer must satisfy

$$L_c + L_f - 1 \leq NL_f$$

\[ OR \]

$$L_f \geq (L_c - 1)/(N - 1)$$

If a baud spaced equalizer is used, rather than a FSE, to equalize the effect of a finite length FIR channel, it is clear from Eq. 3.39 that the length of the equalizer will be infinite, because $N = 1$. Consequently, it is not possible to find a baud-spaced FIR equalizer that can completely equalize the effect of an FIR channel. This conclusion is in accordance with the one we have derived for baud-spaced systems in the previous section. A similar expression as Eq. 3.39 was also derived by Ye Li and Zhi Ding [13].

We can also seek for a MMSE solution for FSE. Since the vector expression of FSE system (refer to Eq. 3.36) has the same form as the baud-spaced system expect for the different definitions of $C$ and $f$, the solution for FSE has the same form as the
baud-spaced equalizer (refer to Eq. 3.19) but with a different structure of $f$, which is a combination of sub-equalizer coefficient vector $f_q$.

3.4 Adaptive Algorithm for Equalization

3.4.1 Adaptive equalization concept

In the previous sections of this chapter, we have derived the general mathematical forms of both MMSE solution (Eq. 3.19) and zero-forcing solution (Eq. 3.38) for channel equalization. One method to obtain an optimal equalizer is to directly solving the Eq. 3.19 and Eq. 3.38 via inversion or pseudo-inversion matrix operation. This method requires the channel convolution matrix $C$ to be estimated first and its performance totally depends on the estimation accuracy of the channel. However, accurate channel estimation is not always available, and in some cases real-time channel estimation is impossible because almost all existing channel estimation techniques are very computationally consuming. Another option is to use certain kind of adaptive algorithm to obtain the equalizer iteratively without doing channel estimation and matrix operation. The core of such a equalizer is a adaptive filter whose coefficients are adapted according to a certain predefined criterion or a cost function. Fig. 3.6 is the block diagram of an equalizer that uses adaptive algorithm. The adaptive filter often uses a FIR structure as shown in Fig. 3.7.

Adaptive equalization algorithms are more widely used in practice, because they can work with unknown and time-varying channels and their implementation complexity is not very high.

The performance of an adaptive algorithm for equalization is mainly determined by three factors: convergence speed, residual error and computational complexity. The convergence speed of an adaptive algorithm is measured by the number of iterations required for the algorithm to reach an optimal condition as expected. A fast convergence algorithm can save computation and adapt itself rapidly to the changes in a channel. The residual error provides a quantitative measure to indicate the
Figure 3.6 An adaptive equalizer

Figure 3.7 An adaptive FIR filter
difference between the adaptively achieved result and the perfect result required by a certain criterion, for instance, the minimal mean square error (MMSE) criterion. Computational complexity determines the amount of hardware and/or software resources needed for implementation. Computational complexity also affects the data throughput of an equalizer, the lower is the computational complexity, the higher is the symbol rate.

3.4.2 LMS algorithm

One very popular adaptive equalization algorithm is the Least Mean Square (LMS) algorithm. The LMS algorithm attempts to minimize a Mean Squared Error (MSE) cost function between the equalizer output and a reference sequence:

\[
J_{\text{mse}} = \frac{1}{2} E(y(k) - y'(k))^2,
\]

where \(y(k)\) is the output of the equalizer and \(y'(k)\) is the reference sequence. For an equalizer with training, the reference sequence is the known training sequence. The transmitter sends the training sequence during start-up phase of transmission or periodically.

LMS is basically a stochastic gradient algorithm, which has a general iterative adaptation formula as below

\[
W(k+1) = W(k) - \mu(k) \nabla_W J_{\text{mse}}
\]

Here, \(W(k) = [w_0(k), w_1(k), ..., w_{L-1}(k)]^T\) is defined as the tap-weight vector of the L-tap-length equalizer at kth iteration step (represented by the index \(k\)), which is the equalizer coefficient or impulse response vector \(f\) as defined in the previous sections of this chapter. \(\mu(k)\) is a small positive real number that acts as the step-size of every update step. \(\nabla_W J_{\text{mse}}\) is the gradient of the MSE cost function regarding the equalizer tap weight vector. Now, we define \(X(k) = [x(k), x(k-1), ..., x(k-L+1)]^T\) as the input data to the equalizer. According the tapped-delay-line structure of the equalizer, the output of the equalizer can be expressed as the inner product of the
input vector and the tap-weight vector

\[ y(k) = W(k)X^T(k) \]  

(3.42)

From Eq. 3.40 and Eq. 3.42, it is easy to obtain the gradient of the MSE cost function with respect to the equalizer tap weight vector

\[ \nabla_f J_{\text{mse}} = E\{(y(k) - y'(k))X^*(k)\}, \]  

(3.43)

where * denotes complex conjugate.

The adaptive algorithm of LMS is obtained by replacing the true gradient with its instantaneous approximation, i.e., ignoring the expectation operator in Eq. 3.43

\[ W(k + 1) = W(k) - \mu(k)(y(k) - y'(k))X^*(k) \]  

(3.44)

In the above equation, the step-size \( \mu(k) \) determines the speed of convergence: the bigger the faster, but the larger is the adaptation error after convergence. However for the sake of stability, \( \mu(k) \) has to be smaller than \( 2/\|X(k)\|^2 \)[1].

LMS algorithm has such advantages as low implementation complexity, very robust and numerically stable, so it is very widely used in practice. Though LMS algorithm needs reference signals or training signals, it provides a good fundamental for understanding and designing adaptive algorithms for blind equalizations, many blind equalizers use an adaptive algorithm which is very similar to LMS.

The convergence rate of LMS algorithm is slow. There is a faster converging algorithm known as the recursive least squares (RLS) algorithm, which is based on a least square approach. The RLS algorithm can significantly improve the convergence of an adaptive equalizer, but it has higher tendency to instability and higher computation complexity than LMS algorithm. Refer to [1] for an introduction to the RLS algorithm.
4. Blind Equalization Algorithm: CMA

In this chapter, we introduce a blind adaptive equalization algorithm called the Constant Modulus Algorithm (CMA). Low in computational complexity and easy to implement, CMA may be the most widely used blind equalization technique in practice [3].

The basic structure of CMA is a linear adaptive filter whose taps are adjusted by a stochastic gradient descent procedure according to the so-called constant modulus (CM) criterion, which defines a cost function that penalizes deviation in the modulus (or magnitude) of the equalized signal away from a constant value.

CMA is usually regarded as a member of the family of Bussgang algorithms. In the trained LMS equalizer introduced in the last chapter, we need a reference signal \( y'(k) \) and calculate the error between \( y'(k) \) and the equalizer output \( y(k) \):

\[
e(k) = y(k) - y'(k),
\]

but, in a blind equalizer without training sequence, we cannot obtain the \( e(k) \) from \( y(k) \) and \( y'(k) \). To circumvent this obstacle, we may define other forms of \( e(k) \). One choice is to use the symbol estimate generated by the decision device to replace the training sequence in the calculation of \( e(k) \), but, it is unlikely to produce a reliable error signal \( e(k) \) at the start-up stage of the system, when the eye-diagram of the signal is still closed and the decision error rate is still very high. Another choice is to apply a suitable non-memory non-linear function of the equalizer output \( g(y(k)) \) to form the error signal, then the blind equalizers, just like the trained LMS equalizers, can use the stochastic gradient algorithm to update its tap-weight:

\[
W(k + 1) = W(k) + \mu(k)g(y(k))X^*(k),
\]

(4.1)

Where \( X^*(k) \) is the conjugated input vector of the equalizer. Those equalization algorithms that have a tap-weight update formula in the form of Eq. 4.1 are often
referred to as Bussgang algorithms.

### 4.1 Constant Modulus (CM) Criterion

CMA is stemmed from Godard's work on the self-recovering equalization and carrier tracking for complex source signals such as QAM. In the reception of QAM signals, the receiver must not only recover the magnitude of source symbols, but also the frequency and phase of carriers. Godard has suggested a blind channel equalization scheme that handles the carrier recovery and the symbol magnitude recovery separately[12]. In his equalization scheme, Godard defines a cost function as below and tries to minimize this cost function, $J_{\text{godard}}$, via the stochastic gradient technique:

$$J_{\text{godard}} = \frac{1}{2p} E\{(|y|^p - \gamma)^2\}, \quad (4.2)$$

where $\gamma = E\{|s(k)|^p\}/E\{|s(k)|^p\}$ is the dispersion constant of source symbol sequence $s(k)$ and $p$ is a integer. When $p = 2$, Eq. 4.2 is referred to as CM criterion or CM cost, which has a form as below

$$J_{\text{cm}} = \frac{1}{4} E\{|y(k)|^2 - \gamma_{\text{cm}}^2\}, \quad (4.3)$$

where $\gamma_{\text{cm}} = E\{|s(k)|^4\}/E\{|s(k)|^2\}$. Taking partial differentiation of the instantaneous value of $J_{\text{cm}}$(without the Expectation operator) over the equalizer's tap weight vector $W$, we obtain the error function that can be used in Eq. 4.1:

$$g(y(k)) = y(k)(|y(k)|^2 - \gamma_{\text{cm}})X^*(k) \quad (4.4)$$

Eq. 4.3 shows that the CM criterion penalizes deviation of signal modulus away from a constant value, which is determined by the statistics of the source signals. The signals with a constant modulus can be represented by a circle in a constellation as shown in Fig. 4.1. For constant modulus source signals such as QPSK signals, it is
4.2 Convergence Properties of CMA

4.2.1 Minima of CM Cost Function

Since the CM cost function defined by Eq. 4.3 is not a quadratic (2nd order) equation, the surface of the CM cost function is not a simple convex surface as that of MSE. The surface is multi-modal. Therefore, unlike in the case of MMSE, the CMA may yield multiple results (including undesirable results) depending on different source signals and initializations.
Let us examine the relation between the CM cost and the channel-equalizer combined system response. The minima of the CM cost function are the channel-equalizer combined system responses that satisfy the below equations:

\[
\frac{dJ_{cm}}{dh(i)} = 0, \quad \forall i
\]  

(4.5)

where \( h(i), \ i = 0, 1, \ldots, L_h - 1 \), is the channel-equalizer combined system response sequence as defined in Chapter 3 and \( L_h \) is the length of the equalizer. Also note \( y(k) = \sum_{j=0}^{L_h-1} h(j)s(k - j) \). Here we assume the channel is a constant channel or a slow fading channel, so the channel keeps unchanging during the period of equalizer adaptation. In the absence of noise, Eq. 4.5 can be expanded into the following equations:

\[
E\{s^*(k - i)\left( \sum_{j=0}^{L_h-1} h(j)s(k - j) \right)^2 \} - E\{|s(k)|^4\} = 0, \quad \forall i,
\]  

(4.6)

Suppose that the source sequence \( s(k) \) is an i.i.d random process and the source symbol constellation is symmetrical, which is often the case in practical communications systems, then we have

\[
E\{s(i)s(j)^*\} = E\{|s(k)|^2\}, \quad i = j;
\]

\[
= 0, \quad i \neq j.
\]  

(4.7)

With the above property of the source sequence, we can rewrite Eq. 4.6 as below [12]

\[
h(i)\left( \frac{E\{|s(k)|^4\}}{2E\{|s(k)|^2\}^2}(|h(i)|^2 - 1) + \sum_{j \neq i} |h(j)|^2 \right) = 0, \quad \forall i
\]  

(4.8)

The above set of equations has an infinite number of solutions. We can classify them into a series of sets \( S_0, S_1, \ldots, S_M, \ldots \) according to the number of non-zero com-
ponent in $h(i)$, which is denoted by subscript $M$. Obviously, $S_1$ is the solution set that has a zero ISI because it has only one non-zero element.

### 4.2.2 Global convergence of CMA

It has been proven that for a double infinite length equalizer, among $S_0, S_1, ..., S_M, ...$, $S_1$ is the only global minima set of the CM cost while all others are unstable stationary points [12]. That means a doubly infinite length CMA equalizer can always converge to a zero ISI solution. In practice, CMA is implemented by using finite length equalizers, which cannot always reach a global minimum with zero ISI except when some specific conditions satisfied.

A finite length CMA baud spaced equalizer (TSE) in general cannot achieve the perfect equalization as discussed in Chapter 3 and may converge to local minima. However, it is proved that a CMA-TSE can converge very closely to a global minimum under certain conditions [13]. To introduce these conditions, we need first define below two terms.

**Unique Global Minimum Cones:**

$$G_i = \{ h = [\ldots, h(-1), h(0), h(1), \ldots]^T \in l^1(C) : |h(i)| > |h(j)|, \forall j \neq i \} \quad (4.9)$$

**Attainable Set:**

$$S = \{ h : h(k) = \sum_{i=-N}^{N} c(k-i) f(i), \forall f = [f(-N), \ldots, f(0), \ldots f(N)]^T \in l^1(C) \} \quad (4.10)$$

Unique Global Minimum Cones define the regions where all the global minima must be in, and Attainable Set defines the set that contains all the possible overall systems attained via finite length equalizers.

A finite length CMA equalizer for QAM signals can converge to a global minimum if the following conditions are satisfied at initialization [14]:

1. The initial impulse response of the overall system $h^{in}$ should be inside a global minimum cone, i.e. $h^{in} \in G_i$. 

38
(2) The initial equalizer output statistics should satisfy:

\[
\frac{E\{|y(k)|^4\}}{E^2\{|y(k)|^2\}} - 2 > 0.5
\]  

(4.11)

(3) If \( e_i \) is the global minimum corresponding to \( G_i \), it should be inside or very close to \( S \cap G_i \).

For a CMA TSE, condition (1) can be met by initializing the equalizer as

\[
f^{\text{in}} = [f(-N), ..., f(-1), f(0), f(1), ..., f(N)]^T
\]

\[
= [0, ..., 0, 1, 0, ..., 0]^T
\]  

(4.12)

With this initial value of \( f \), the initial overall system impulse response is exactly the response of the channel \( c(k) \). Since \( c(k) \) usually has a unique peak, \( h^{\text{in}} \) will most likely be inside a global cone. For a channel without any zero on the unit circle, Condition (3) can be satisfied if the equalizer is sufficiently long. Condition (2) cannot be guaranteed without knowledge of the actual channel. This problem is countered by updating the CMA equalizer in the following way [13]:

(a) *Center tap initialization*;

(b) *Tap centering*: Center the center of gravity of the tap weight vector at each iteration;

(c) *Length extension*: If \( f \) has heavy tails after an adequate number of iterations and tap centering, the equalizer length should be extended on the side of the heavy tail (longer equalizer).

For a CMA fractionally spaced equalizer (FSE), a set of conditions for global convergence is established in [14]. These conditions are given below.

(1) No common root among all the sub-channels.

(2) Length of each sub-equalizer must satisfy:

\[ L_c - 1 \leq (N - 1)L_f, \]
where $L_c$ is the length of sub-channel, $L_f$ the length of each sub-equalizer and $N$ is the number of sub-channels.

(3) The normalized source kurtosis satisfies

$$k_s = \frac{E\{|s(k)|^4\}}{(\sigma_s^2)^2} < k_g,$$

where, $\sigma_s^2$ is the variance of the source, $k_g = 3$ for a real source or $k_g = 2$ for a complex source.

It should be noted that condition (1) and condition (2) are the perfect equalization requirements for any linear FSE (not limited to CMA) as introduced in Chapter 3.

For CMA-FSE, there is no center tap initialization needed [14]. One simple initialization strategy is just to set

$$f_{in}^i = [1, 0, ..., 0];$$

$$f_{ln}^i = [0, 0, ..., 0], \forall i \neq 0,$$

(4.13)

where $f_{in}^i$ is the initial tap-weight vector of the $i^{th}$ sub-equalizer.

4.3 Comparison between CMA-TSE and CMA-FSE

The Constant Modulus Algorithm was first designed for T-spaced channel equalizers, which usually have a linear FIR structure. One weakness of a CMA-FSE is the undesirable local convergence. Due to the predetermined linear FIR structure, another weakness is the severe noise enhancement for a channel with zeros near the unit circle. Moreover, when a channel has zeros that are very close to the unit circle, a very long tap-line structure is required to remove sufficient ISI if a TSE is used. Longer equalizers typically require more measured data for adjustment and hence longer time for convergence. Therefore, it is ineffective and inefficient to apply CMA-TSE to the linear channels that have zeros on or near the unit circle.

CMA is often implemented in the form of FSE. Besides those benefits such as timing phase insensitivity and less noise enhancement, CMA-FSE can also overcome
the two major weaknesses of a CMA-TSE as described above. It is shown that a CMA-FSE can converge to a global minimum under some weak conditions as we discussed in the last section. A CMA-FSE can also be applied to the linear channels that have zeros on or near the unit circle. For some channels with deep nulls in spectrum, the CMA-FSE need not have a very long tap-line length and hence it will converge faster. Of course, a FSE may need more bandwidth or other resources because FSE is usually archived by over-sampling signals or using multiple sensors in the receivers. However, the costs of additional resources involved are usually affordable when the improved performances is desired.

4.4 Numerical examples of CMA equalizers

Here we present a numerical simulation on an FIR channel example taken from the SPIB database at Rice University. The channel data are extracted from field measurements of a digital microwave radio channel. The $T/2$ fractionally sampled impulse response of the channel is $(-0.0220-0.0157i, -0.0299-0.0310i, 0.0083-0.0204i, 0.0021-0.0034i, 0.1449+0.0394i, 0.6066+0.0653i, 0.8162-0.0155i, 0.3123-0.0870i, -0.1771-0.0169i, -0.0617+0.0535i, 0.1166+0.0225i, -0.0508+0.0095i, -0.2430+0.0474i, -0.1807+0.0513i, -0.0027+0.0277i, 0.0438+0.0099i)$. Fig. 4.2 shows the magnitude of the channel impulse response. A $T/2$-spaced CMA equalizer will be applied to this channel. The equalizer consists of two $T/2$ spaced sub-equalizers. To fulfill the condition of perfect equalization we discussed in Chapter 3, the tap length of each sub-equalizer in the FSE must satisfy Eq. 3.39. Since the length of the channel in Fig. 4.2 is 8 baud interval, the tap length of sub-equalizers must be equal or larger than 7. Here we choose 8 as the tap length of the two sub-equalizers. The source sequences for the simulation are i.i.d. random sequences of uniform distributions, and are modulated into 16QAM signals. In addition, a white noise sequence is added to the channel in the simulation. The SNR of the channel output signal (the ratio between the output signal power and the noise signal power) is 35dB. To show the performance of the equalizer in removing ISI, we define a quantitative measurement of ISI as below
In practice, the parameters of the channel are often unknown and hence the above formula is not widely used. Instead, the normalized mean square error (NMSE) metric is more often used to measure the performance of the equalizer. If the error due to ISI plus the error due to AWGN are Gaussian distributed, the lower the NMSE, the lower the bit error rate. Inter-symbol interference may also be measured by the eye diagram, which is constructed by overlaying plots of the waveform in successive unit time intervals [2].

Fig. 4.3 to Fig. 4.8 illustrate the numerical simulation result with CMA-FSE. We can see the equalizer converge in terms of both NMSE and ISI, and finally it opens
Figure 4.3 The constellation of the original received signal

the eye-diagram of the signal. We also carried out the simulation with a CMA-TSE equalizer (Fig. 4.10, Fig. 4.11), but it was unable to converge for the same channel used in the simulation with the CMA-FSE equalizer.
Figure 4.4 The eye diagram of the original signal
Figure 4.5 The convergence curve of the CMA-FSE in terms of ISI
Figure 4.6 The convergence curve of the CMA-FSE in terms of normalized MSE
Figure 4.7  The magnitude the combined channel and equalizer impulse response
Figure 4.8 The eye diagram of the equalized signal
Figure 4.9 The constellation of the equalized signal
Figure 4.10 The convergence curve of the CMA-TSE in terms of ISI
Figure 4.11 The convergence curve of the CMA-TSE in terms of normalized MSE
5. A Hybrid Study Between CMA and Decision-directed Algorithm

While the CMA has the advantages of robustness and implementation simplicity, it has the disadvantages of slow convergence and large residual error. The decision-directed or decision-feedback equalization algorithm converges faster and yields smaller residual errors if the eye-diagram of received signals is initially open. Unfortunately this algorithms is often unable to converge when the eye-diagram of received signals is initially closed and therefore it has not been widely used for blind equalizations. In this chapter, a new decision-directed equalization algorithm is proposed. The new algorithm can converge even when the signal eye-diagram is not open, and has smaller residual errors than the CMA. We also suggest a hybrid algorithm that combines this new decision-feedback based algorithm and the CMA. The hybrid algorithm is designed with an intention to take advantages of both of the two algorithms.

5.1 A Decision-directed Algorithm

Consider the cost function:

$$J = E[(|y(k)|^p - |s'(k)|^p)^2]$$

(5.1)

where \(s'(k)\) is the symbol signal determined by the output of the decision device that follows the equalizer and \(p = 1\) or \(2\). Unlike classical decision-directed equalizers, the new one only makes use of the magnitude information of the decision signal and hence is not sensitive to the phase errors of the received signals. If there is no magnitude error in the decision, which means \(|s'(k)| = |s(k - m)|\), where \(s(k - m)\) is the \(m\)-tap delayed version of the source signal \(s(k)\). It is obvious from Eq. 5.1 that when \(J \rightarrow 0\) we will have \(|y(k)| \rightarrow |s(k - m)|\) and consequently the channel and equalizer
combined response \[ h_0(k) \cdots h_{L-1}(k) \rightarrow [0, \cdots e^{io}, \cdots 0]. \] Of course, it is unlikely that \( |s'(k)| \) always equals to \( |s(k - m)| \) in practice, particularly when at the beginning the eye-diagram of \( y(k) \) is not yet open enough for the decision device to make correct decision on the original signals. In the latter part of this chapter, we will analyze the equalization algorithm with decision errors taken into account.

### 5.2 Stability and Convergence Analysis

We will analyze the stability and convergence characteristics of our decision-directed algorithm by using *aposteriori* analysis, which is a simple and effective method for analyzing the behaviors of adaptive filters. Below is a brief description of *aposteriori* analysis theory for adaptive filters, for a detailed introductions of the theory please refer to [16] and [17].

*Apriori* analysis is a technique for studying the stability of adaptive filters using the relationship between the *apriori* and *aposteriori* errors. A general tap-weight update formula for the linear adaptive filters is given below:

\[
W(k + 1) = W(k) + \mu(k)f(e(k))X(k),
\]

(5.2)

where

\[
e(k) = d(k) - W(k)X^T(k)
\]

(5.3)

is the *apriori* error signal at time \( k \), \( d(k) \) is the reference signal. The function \( f(.) \) is the derivative of a certain convex cost function \( \psi(e) \).

The *aposteriori* error signal is defined as

\[
e_p(k) = d(k) - W(k + 1)X^T(k),
\]

(5.4)

which is the error signal after the tap-weight has been updated from \( W(k) \) to \( W(k+1) \).

For our analysis, we define an unknown optimal tap-weight vector \( W_{opt} \) which has a relationship with reference signal \( d(k) \):

\[
d(k) = W_{opt}X^T(k) + \eta(k),
\]

(5.5)
where \( \eta(k) \) is an observation noise signal. We also assume that \( \| X(k) \|^2 \geq \epsilon \) for some \( \epsilon \geq 0 \). Under such assumptions, we have the following theorem [16]:

**Theorem 1:** Assume that an adaptive algorithm can be written in the form

\[
W(k + 1) = W(k) + (e(k) - Q(e(k))) \frac{X(k)}{\| X(k) \|^2},
\]

where \( Q(\cdot) \) is any function. Then, the *aposteriori* error is given by

\[
e_p(k) = Q(e(k)).
\]  

(5.7)

Moreover, if \( Q(\cdot) \) is contractive for all \( e(k) \) such that

\[
|e_p(k)| \leq \beta |e(k)|,
\]

(5.8)

for some \( 0 < \beta < 1 \), then the algorithm is \( l_2 - stable \).

**Proof:** refer to [16].

It is easy to show that \( e_p(k) \) is the *aposteriori* error for the tap-weight update formula (Eq. 5.6) if we express \( d(k) \) in the form of Eq. 5.5. By subtracting \( W_{opt} \) from both sides of Eq. 5.6, we obtain

\[
W'(k + 1) = W'(k) - (c(k) - Q(c(k))) \frac{X(k)}{\| X(k) \|^2},
\]

(5.9)

where \( W'(k) = W_{opt} - W(k) \) is the tap-weight error vector. Multiplying both sides of Eq. 5.9 by \( X'(k) \) and adding \( \eta(k) \) on both sides, we can obtain Eq. 5.7, where \( e_p(k) \) is defined by Eq. 5.4.

The theorem indicates that the robustness of any adaptive algorithm of the form in Eq. 5.2 can be determined by the relationship between the *apriori* error \( e(k) \) and the *aposteriori* error \( e_p(k) \).

Now we apply the *aposteriori* analysis to our decision-directed based algorithm.

From Eq. 5.1, by taking partial derivative of the term inside the expectation bracket over the equalizer tap weights, we get the update factor for the stochastic
gradient procedure:

\[ D_J(k) = -2|y(k)|^{p-2}(|s'(k)|^p - |y(k)|^p)y(k)X^*(k) \]  

(5.10)

For \( p = 1 \),

\[ D_J(k) = -2(|s'(k)| - |y(k)|)(y(k)/|y(k)|)X^*(k) \]  

(5.11)

and the update formula for the equalizer tap weights is

\[ W(k + 1) = W(k) + \mu(k)e(k)X^*(k), \]  

(5.12)

where

\[ e(k) = (|s'(k)| - |y(k)|)y(k)/|y(k)| \]  

(5.13)

The \( e(k) \) is the priori error signal at time \( k \). We also obtain the a posteriori error signal as

\[ e_p(k) = (|s'(k)| - |W^T(k + 1)X(k)|)y(k)/|y(k)| \]  

(5.14)

Here we introduced a random noise sequence caused by the errors in decision

\[ dR(k) = |s'(k)| - |s(k - m)| \]  

(5.15)

then the noiseless error signal can be defined as

\[ e'(k) = (|s(k - m)| - |y(k)|)y(k)/|y(k)| \]  

(5.16)

\[ = e(k) - \eta(k), \]  

(5.17)

where

\[ \eta(k) = dR(k)(y(k)/|y(k)|) \]  

(5.18)

Going through Eq. 5.12 to Eq. 5.18, the update formula can be rewritten as

\[ W(k + 1) = W(k) + (e(k) - Q(e(k)))X^*(k)/\|X(k)\|^2, \]  

(5.19)

where

\[ Q(e(k)) = (1 - \mu(k)\|X(k)\|^2)e(k). \]  

(5.20)

It can be proved that

\[ e_p = Q(e(k)) \]  

(5.21)
According to the results in [16] and [17], if \( \eta(k) \) is \( L_2 \)-bounded, \( \|X(k)\|^2 > 0 \) for all \( k \), and also if

\[ |e_p(k)| \leq \beta(k)|e(k)|, \quad 0 \leq \beta(k) < 1 \]  

then the algorithm in the form of Eq. 5.19 is \( L_2 \)-stable and satisfies

\[ \lim_{n \to \infty} \frac{1}{n} \sum_{k=0}^{n-1} \frac{|e(k)|^2}{\|X(k)\|^2} = \alpha \lim_{n \to \infty} \frac{1}{n} \sum_{k=0}^{n-1} \frac{\|\eta(k)\|^2}{\|X(k)\|^2} \]  

for some \( 0 < \alpha < \infty \). With Eq. 5.16 and Eq. 5.18, Eq. 5.23 is equivalent to

\[ \lim_{n \to \infty} \frac{1}{n} \sum_{k=0}^{n-1} \frac{\|s(k - m) - y(k)\|^2}{\|X(k)\|^2} = \alpha \lim_{n \to \infty} \frac{1}{n} \sum_{k=0}^{n-1} \frac{|dR(k)|^2}{\|X(k)\|^2}. \]  

In order to meet the requirement of Eq. 5.23, \( \mu(k) \) should be chosen as such

\[ 0 < \mu(k) < 1/\|X(k)\|^2 \]  

Because \( h(k) \) is \( L_2 \)-bounded for a symbol space of finite magnitude, The algorithm is \( L_2 \)-stable if \( \mu(k) \) satisfies Eq. 5.25.

In practice, there is always a AGC (Automatic Gain Control) circuit in the receiver, the average magnitude of \( x(k) \) is almost constant, so Eq. 5.24 can be simplified as

\[ \lim_{n \to \infty} \frac{1}{n} \sum_{k=0}^{n-1} \frac{(|s(k - m)| - |y(k)|)^2}{\|X(k)\|^2} \]

\[ = \alpha \lim_{n \to \infty} \frac{1}{n} \sum_{k=0}^{n-1} |dR(k)|^2 \]  

For \( p = 2 \), we can obtain the similar result but the \( e(k) \) needs to be changed to

\( (|s'(k)|^2 - |y(k)|^2)y(k) \).

However, so far we don't know yet if the algorithm really converges. If in Eq. 5.26 the time average is replaced by statistic average, and let \( \xi_r = E[|dR(k)|^2] \) and \( \xi_s = \frac{1}{\xi_r} \).
Figure 5.1 Auto-feedback loop of convergence

\[ E[(|s(k-m)|-|y(k)|)^2], \]

we find that \( \xi_e \to 0 \) as \( \xi_r \to 0 \), but \( \xi_e \) itself is determined by \( \xi_r \). It is obvious that, the smaller the \( \xi_e \) is, the smaller the decision error probability and the \( \xi_r \). If the initial value of \( \xi_e \) is larger than \( \alpha \xi_r \), according to Eq. 5.24, the adaptation of the equalizer will force \( \xi_e \) to move towards \( \alpha \xi_r \) and to become smaller than its original value, which will in turn decrease \( \xi_r \). This process actually has formed an auto-feedback loop as shown in Fig. 5.1 for \( \xi_e \) with a loop gain smaller than one, so \( \xi_e \) will gradually get closer and closer to zero. From above analysis it follows that the initial condition for the algorithm to converge is

\[ E[|dR(k)|^2] < (1/\alpha)E[(|s(k-m)|-|y(k)|)^2]. \]  (5.27)

A problem with the above formula is how to determine the value of the \( \alpha \), it is very difficult to find a method to calculate the precise value of the \( \alpha \). However it is possible to determine a gross range of value for the \( \alpha \). We notice, in a PAM or QAM symbol space, the magnitude of symbol is of finite value. Let \( A_{max} \) and \( A_{min} \) denote the maximal and the minimal symbol magnitudes respectively, so the the symbol magnitude is somewhere between \( A_{max} \) and \( A_{min} \), and also the magnitude of \( y(k) \) is controlled by the decision-directed algorithm within a range only a little wider than \( A_{max} \) and \( A_{min} \), most likely \( |y(k)| \) is located between \( [A_{min}-Q_{min}, A_{max}+Q_{min}] \), where
$Q_{\text{min}}$ is the minimal symbol magnitude step-size. Under the most extreme condition $E[(|s(k-m)| - |y(k)|)^2]$ is approximately equal to $(A_{\text{max}} - A_{\text{min}})^2$ and the correct decision rate is almost zero, in such a case, $E[|dR(k)|^2]$ should be larger than or at least equal $Q_{\text{min}}$, and of cause $E[|dR(k)|^2]$ may not be larger than $(A_{\text{max}} - A_{\text{min}})^2$.

With the above conditions and the formula of Eq. 5.26, we can find that the $\alpha$ should satisfy

$$1 \leq \alpha \leq (A_{\text{max}} - A_{\text{min}})^2 / Q_{\text{min}}^2.$$  \hspace{1cm} (5.28)

Take 16-QAM for example, we can obtain the maximum value of the $\alpha$ to be $\alpha_{\text{max}} \simeq 2$. The value of $E[|dR(k)|^2]$ depends on the distribution density of $e'(k)$ and what decision method being used. If $e'(k)$ can be regarded as a zero-mean Gaussian noise and a Least Euclid Distance Decision device is used, $E[|dR(k)|^2]$ can be estimated by the below formula:

$$E[|dR(k)|^2] \simeq 2Q_{\text{min}}(1 - G(Q_{\text{min}} / \sigma^2)), \hspace{1cm} (5.29)$$

where $\sigma^2 = E[|e'(k)|^2]$ and $G(\cdot)$ is the normalized Gaussian distribution function.

### 5.3 A Hybrid Algorithm

In the preceding section, we have discussed the effect of the error signal $(|s(k-m)| - |y(k)|)$ on the convergence of the new decision-directed algorithm. If the error signal is too large, the algorithm may converge very slowly or even cannot converge. In this case, it is more reasonable to use other techniques to make the error signal smaller before the decision-directed algorithm starts working. We have designed a hybrid algorithm that combines the CMA and the decision-directed algorithms. The hybrid algorithm defines an estimation of $E[(|s(k-m)| - |y(k)|)^2]$, that is

$$\xi_n = \sum_{i=0}^{n} r^{n-i} (|y(i)| - |s(i)|)^2, \hspace{0.5cm} 0 < r < 1 \hspace{1cm} (5.30)$$

As shown in Fig. 5.2 the hybrid algorithm begins with the CMA and switches to the decision-directed algorithm when the estimated NMSE is small enough and Eq. 5.27 is satisfied.
Figure 5.2 A hybrid equalization scheme
5.4 Numerical Simulation

Computer simulation results of the three algorithms are shown in Fig. 5.3 and Fig. 5.4 (results from MATLAB programs). All the three equalizers are of the $T/2$-spaced structure. The simulations use uniformly distributed random sequences for the source signals (16-QAM modulated) and the SNR of the channel is 35 dB. The original received signals are "eye-closed" (see Fig. 5.4.a) due to the ISI introduced from the channel distortion. In Fig. 5.3, it can be observed that our new decision-directed algorithm converges slower than the CMA within the first several hundred iterations, but when the Normalized Mean Square Error (NMSE) becomes small enough it converges much faster than the CMA and results in much smaller residual errors.
Figure 5.4 Signal constellations: (a) original received signals, (b), (c), and (d) equalized by the CMA, the decision-directed and the hybrid algorithms respectively.
6.  Hardware Implementation

6.1  Introduction to FPGA

6.1.1  FPGA architecture

The Field-Programmable Gate Arrays (FPGA) is an integrated circuit that consists of a large number of configurable logic element (CLE) or configurable logic blocks (CLB) and a matrix of wires and programmable switches that provides a very flexible connection mechanism between these blocks. Fig. 6.1 shows a general architecture of FPGA. A desired circuit design is implemented by specifying the logic function for each logic block and the interconnection between logic blocks [21].

The CLB (CLE) is the basic logic unit in FPGA. A complex logical design is built on a combination of the simple logic functions of many CLB (CLE)s. Generally, a CLB (CLE) contains combinational logic elements and sequential logic elements, both of them are programmable. A combinational logic element may be physically implemented as a small look-up table (LUT) memory or as a set of multiplexers and gates. The LUT, as a function generator that can quickly implement any logical function of the inputs, is more flexible and provide more inputs per cell than multiplexer cells, however, this is at the expense of propagation delay. The sequential logic elements in the CLB (CLE) are programmable registers that can be configured for D, T, JK, or SR operation. The register clock and clear control signals can be driven by global signals, general-purpose I/O pins, or any internal logic. Fig. 6.2 shows a example of the CLB (CLE) in FPGA.

The interconnect resources, i.e., wires and programmable switches, provide routing paths for connecting a CLB (CLE) to other CLB (CLE)s or to I/O cells. Fig. 6.3 depicts a FPGA with a two-dimensional array of logic blocks that can be interconnected
Figure 6.1 The architecture of FPGA

[21]
Figure 6.2  The structure of CLB

[21]
by interconnect wires. All internal connections are composed of metal segments with programmable switching points to implement the desired routing. An abundance of different routing resources is provided to achieve efficient automated routing.

![Image](image.png)

Figure 6.3 The interconnection between CLB (CLE)s

6.1.2 FPGA Programming

Fig. 6.4 shows a typical digital system design flow for FPGA implementation.

Design Entry

The digital design can be created with a schematic digital design tool or a hardware description language, i.e. Verilog or VHDL. It is recommended to use a hardware description language for design entry, because a hardware description language such as Verilog or VHDL is the industry standard that is accepted by almost all FPGA or
Figure 6.4 FPGA design flow
ASIC design tools.

RTL Simulation

Designs created by hand or via design entry tools are usually at the Register Transfer Level (RTL). These designs are FPGA implementation independent. Prior to synthesis, you must verify the RTL design to confirm that its functionality is as intended. It is at this point that you create a testbench for the design. Use this testbench throughout the FPGA flow to verify the functionality of the design at each level (RTL, Functional Gate, Timing Gate).

Synthesis

Synthesis uses user-specified synthesis constraints (timing, power and area etc.) to implement and optimize the RTL design into primitive blocks (flip-flops, logic gates). The result is a device independent netlist ready for place and route that represents the functionality of the design without any timing information. It is in the Place and Route (PR) step that you convert the blocks in this netlist into their FPGA device specific elements. A standard output format for synthesis tools is EDIF (Electronic Design Interchange Format).

Functional Simulation

The net-list exported by the synthesis tools is at functional level, containing no timing information. By performing a functional simulation, one can test the logical operation of the design without the need for timing information. In a functional simulation, the output logic levels change at the same time as the input vectors and no propagation delays are used during simulation.

Place and Route

The synthesis tool exports a net-list (EDIF) that represents the functionality of the design. Place and Route tools take this net-list and implement it in the technology that their FPGA devices are based upon. The PR tool then exports a device configuration file for programming the targeted FPGA device.
Timing simulation

Timing simulation can be used to verify that the design meet the timing requirements. Timing simulation uses a fully compiled net-list that includes estimated or actual timing information.

FPGA Configuration

Configuration is a process in which the circuit design is downloaded into the FPGA. The configuration data of the design is stored in the device configuration file that exported from the PR tool. One can download configuration data into a device through the communications cable provided by the FPGA vendor.

6.2 Arithmetic Circuit Element

6.2.1 Two's Complement Number

In most cases it is necessary to represent digital signals and the results of arithmetic operations on them in the form of signed numbers. The most common way to represent signed numbers in the binary number system is the two's-complement representation.

In the two's-complement representation, the most significant bit (MSB) of a number is the sign bit. If that bit is 0, then the number is positive; if it is 1, then the number is negative. To change a positive number into the corresponding negative number in the two's complement system, we perform two steps: complement the number (change all 1s to 0 and all 0s to 1). Add 1 to the complemented number. Let’s illustrate the process using simple 8-bit integers. The 8-bit binary representation of the number 3 is 00000011. The two's-complement representation of the number \(-3\) is found as follows: Complement the positive number: 11111100 Add 1 to the complemented number: 11111100 + 1 = 11111101 Exactly the same process is used to convert negative numbers back to positive numbers. To convert the number \(-3\) (11111101) back to a positive 3, perform the following steps: Complement the
negative number: 00000010 Add 1 to the complemented number: 00000010 + 1 = 00000011

The special advantage of two’s-complement arithmetic is that positive and negative numbers may be added together according to the rules of ordinary addition regardless of the sign, and the resulting answer will be correct, inducing the proper sign. Consequently, an arithmetic circuit may add any two integers together without checking their signs. This process simplifies the design of arithmetic circuits. The following examples illustrate this point:

1. Add $3 + 4$ in two’s-complement arithmetic.

   $00000011 + 00000100 = 00000111 (+7)$  \hspace{1cm} (6.1)

2. Add $(-3) + (-4)$ in two’s-complement arithmetic.

   $11111101 + 11111100 = 111111001 (-7)$  \hspace{1cm} (6.2)

In a case like this, we ignore the extra 9th bit resulting from the sum, and the answer is 11111001. The two’s complement of 11111001 is 00000111 or 7, so the result of the addition was -7!

3. Add $3 + (-4)$ in two’s complement arithmetic.

   $00000011 + 11111100 = 11111111 (-1)$ \hspace{1cm} (6.3)

The answer is 11111111. The two’s complement of 11111111 is 00000001 or 1, so the result of the addition was -1. With two’s-complement numbers, binary addition comes up with the correct answer regardless of whether the numbers being added are both positive, both negative, or mixed.

Many hardware implementations of signal processing algorithms use fixed point numbers. The precision of fixed point number can be described by a precision string ‘x.x’, say ‘2.14’, which means the word-length of the number is 16 bits, 2 bits for the integer part and 14 bits for the fractional part. For a two’s complement number, there must be at least one bit in the integer part to indicate the sign of the number.
6.2.2 Digital Adder and Subtractor

The addition of two \( n \)-bit binary numbers: \( A = (a_{n-1}\ldots a_0) \) and \( B = (b_{n-1}\ldots b_0) \) can be accomplished by \( n \) full adders cascaded as shown in Fig. 6.5. The full adder, which is the basic building block for most arithmetic circuits, produces the sum and carry-out bits from two one-bit binary numbers and a carry-in bit from another full adder. A logic symbol and a truth table of a full adder are shown in Fig. 6.6. The circuit of Fig. 6.6 is called the ripple-carry adder since carries produced since the carries produced by lower-order stages must propagate or ripple through the higher-order stages before the addition operation is completed.

![Figure 6.5 The architecture of ripple carry adder](image)

Ripple-carry adders are simple in both operation and structure but are slow. In the worst case (\( A = 1\ldots 11 \) and \( B = 0\ldots 01 \)) a carry produced in the least significant full adder must propagate through all the more significant ones.

We can use carry-lookahead circuits to speed up the addition. A carry-lookahead adder calculates all the carry bits in parallel by using a separate logic circuit (called carry-lookahead circuit), instead of full adders, so there is no propagation of a carry bit from one full adder to the next as in a ripple-carry adder. Fig. 6.7 is the block diagram of a 4-bit ripple-carry adder. Since the combinational logic of the carry
Figure 6.6  The symbol and the truth table of full adder

<table>
<thead>
<tr>
<th>a</th>
<th>b</th>
<th>c_i</th>
<th>000</th>
<th>001</th>
<th>010</th>
<th>011</th>
<th>100</th>
<th>101</th>
<th>110</th>
<th>111</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

Figure 6.7  The architecture of carry lookahead adder
lookahead circuit will become very complex as the bit number grows, it is very costly
to implement a ripple-carry Adder longer than four bits. It is more reasonable to
built a long multi-bit adder by a series of cascaded 4-bit ripple-carry adders.

As to subtraction in a two's complement number system, it can easily be performed
by adding the minuend to the negative of the subtrahend, i.e. \((A - B)\) can be obtained
by computing \(A + \overline{B} + 1\). Therefore, a two's complement subtractor can be built from
a adder by placing inverters on the B inputs of this adder and setting the first carry-in
bit \((c_0)\) to 1.

6.2.3 Digital Multiplier

Similar to decimal multiplication, binary multiplication involves the addition of
shifted versions of the multiplicand based on the value and position of each of the
multiplier bits. Actually binary multiplication is even easier, because the value of
each digit of a binary number can only be 0 or 1. Thus, depending on the value of
the multiplier bit, the partial products can only be a copy of the multiplicand, or 0.
In digital logic, this is simply an AND function.

A straightforward way to realize a binary multiplier is just to use an array of full
adders that sums up all the partial products of multiplicand and multiplier numbers.
Fig. 6.8 illustrates a 4-bit unsigned multiplier of such kind. The basic building block,
is a full adder circuit that sums a locally computed partial product, an input passed
into the block from above \((\text{Sum In})\), and a carry passed from a block at its right
side. It generates a carry-out \((\text{Cout})\) and a new sum \((\text{Sum Out})\). Fig. 6.8 shows the
interconnection of 16 of these blocks to implement the full multiplier function. The
bits of \(A\) are distributed along block diagonals and the bits of \(B\) are passed along
rows. This implementation uses 16 AND gates and adders in total.

The critical propagation path length of the above array multiplier is 10. Assuming
\(t_d\) is the worst case delay per cell, the total worst case delay of the array is \(10t_d\). For
a \(n \times n\) bit array, the critical propagation path length is \(3n - 2\), the worst case delay
can be shown to be \((3n - 2)t_d\) [20].
Figure 6.8  A 4-bit unsigned array multiplier
Figure 6.9  A carry save multiplier
The array of Fig. 6.8 could be modified to work faster by applying the carry-save principle. Instead of propagating the carry length-wise, we could feed it forward diagonally as in the carry-save adder circuit. This results in a better circuit as shown in Fig. 6.9. Now the worst-case delay is only $2n \cdot t_d$, which is better than $(3n - 2)t_d$ (especially when $n$ is large).

![Figure 6.10 A signed multiplier](image)

For signed multiplication, let's consider two numbers in two's complement form:

$A = (a_{n-1}...a_1a_0)$ and $B = (b_{n-1}...b_1b_0)$. They are equivalent to

\[
A = -2^{n-1}a_0 + \sum_{i=0}^{n-2} a_i 2^i,
\]

\[
B = -2^{n-1}b_{n-1} + \sum_{i=0}^{n-2} b_i 2^i. \tag{6.4}
\]

The product of $A$ and $B$ is given by

\[
P_i = \sum_{j=0}^{i} (a_j \cdot b_{i-j})
\]
\[ P = 2^{2n-2}a_{n-1}b_{n-1} + \sum_{i=0}^{n-2} \sum_{j=0}^{n-2} a_i b_j 2^{i+j} - 2^{n-1}a_{n-1} \sum_{i=0}^{n-2} b_i 2^i \]

\[-2^{n-1}b_{n-1} \sum_{i=0}^{n-2} a_i 2^i \]

\[= 2^{2n-2}a_{n-1}b_{n-1} + \sum_{i=0}^{n-2} \sum_{j=0}^{n-2} a_i b_j 2^{i+j} + 2^{n-1}a_{n-1}[-2^{n-1} + 1 + \sum_{i=0}^{n-2} b_i 2^i] \]

\[+2^{n-1}b_{n-1}[-2^{n-1} + 1 + \sum_{i=0}^{n-2} a_i 2^i]. \]

\[= 2^{2n-2}a_{n-1}b_{n-1} - 2^{2n-2}(a_{n-1} + b_{n-1}) + 2^{n-1} \sum_{i=0}^{n-2} a_{n-1} b_i 2^i \]

\[+2^{n-1} \sum_{i=0}^{n-2} b_{n-1} a_i 2^i + 2^{n-1}(a_{n-1} + b_{n-1} - 1) \]

(6.5)

(6.6)

(6.7)

To avoid the substraction in the above equation, we convert the 2nd term into two's complement numbers

\[-2^{2n-2}(a_{n-1} + b_{n-1}) \Rightarrow 2^{2n-2}[2 + a_{n-1} + 1] + (2 + b_{n-1} + 1)] \]

\[\Rightarrow 2^{2n} + 2^{2n-1} + 2^{2n-2}(a_{n-1} + b_{n-1}). \]

(6.8)

The term \(2^{2n}\) can be omitted because the product only needs \(2n\) bits. Then, the product in two's complement form can be written as

\[ P = 2^{2n-2}(a_{n-1}b_{n-1} + a_{n-1} + b_{n-1}) + 2^{n-1} \sum_{i=0}^{n-2} a_{n-1} b_i 2^i + 2^{n-1} \sum_{i=0}^{n-2} b_{n-1} a_i 2^i \]

\[+2^{n-1}(a_{n-1} + b_{n-1}) \]

(6.9)

The above formula leads to an array multiplier very similar to the unsigned multiplier that is built with a full-adder array. Fig. 6.10 shows a 4-bit two's complement multiplier based on Eq. 6.9.

### 6.3 Hardware Design of Equalizer

#### 6.3.1 Overall Architecture

We focus on channel equalizer in this study, but we need to consider an overall communication receiver structure in the hardware implementation, because in the
real world a channel equalizer never works by itself alone but as an integral part of a communication receiver. Fig. 6.11 illustrates how a channel equalizer fits into a QAM system.

As we mentioned earlier, a channel equalizer usually works at baseband, so we have not only implemented a channel equalizer in hardware but also a QAM demodulator, which converts a received signal to the baseband signal that can be processed by the channel equalizer.

Figure 6.11 A QAM transmission system

Fig. 6.12 is the overall hardware architecture of the combined system of demodulator and equalizer. The QAM demodulator has a frequency mixer (multiplier) to remove the carrier frequency of the received signal, down-converts the high frequency bandpass signal to the baseband signal. There is also a carrier recovery loop that corrects the frequency shift and the phase shift in the received signal. This carrier
recovery loop is put at the output of the channel equalizer instead of at the output of the mixer. We make such an arrangement because our carrier recovery is based on decision-directed algorithm that makes use of the decision signal, the long tapped-delay-line of the equalizer introduces a delay to the carrier recovery loop if it is put at the input of the equalizer. Such a delay, if long enough, will degrade the tracking ability of the carrier recovery loop.

![Diagram](image)

**Figure 6.12** Hardware system architecture including demodulator and equalizer

### 6.3.2 Channel Equalizer

Since the channel equalizer implemented is a $T/2$-spaced equalizer, the input signal is sampled at a rate of $2/T$ samples per second, but the output of the equalizer is sampled at the symbol rate $1/T$ and also the tap weights of the equalizer are updated at a rate of $1/T$ too. That means a two-branch structure as shown in Fig. 6.13 can

78
be used to efficiently and economically implement the design. In this two-branch structure, there are two parallel tapped-delay-lines, each has the same number of taps and works at the symbol rate. The output of the entire equalizer is the sum of the results of these two tapped-delay lines.

![Diagram of the equalizer hardware structure](image)

**Figure 6.13** The hardware structure of the equalizer

In the equalizer hardware structure as shown in Fig. 6.13, there are four basic components: data de-multiplexer, taps, gradient calculator and adder trees.

**The Equalizer Tap**

The heart of FIR structured equalizer is the tap. The tap holds the input data for a period of one clock cycle, multiples the data by the tap-weight and outputs the result (tap-value) to the adder tree, forwards the data to the next tap in the tapped-delay-line when next clock cycle arrives. Here the clock rate is the same as the symbol rate because the data comes into each branch of the equalizer at the symbol rate $1/T$. 

79
In a QAM system, the symbol data is a combination of two orthogonal signals I and Q, so the input data, the tap-weight and the output all can be regarded as complex numbers.

Since the tap-weight in each tap needs to be updated in every clock cycle, there must be a tap-weight calculator that produces a new tap-weight every clock cycle. To implement a separate tap-weight calculator for each tap, four multipliers are needed because the symbol data and the tap weight are complex numbers (one complex number multiplying operation needs four real number multiplying operations). To conserve resources the tap-weight calculator is integrated into the tap to share the multipliers with the other operations of the tap.

Figure 6.14 The tap structure

Fig. 6.14 indicates the two main functional parts in the tap: the two D-registers that act as a delay element for the input data and the arithmetic unit that carries out
all the arithmetic calculations in the tap. Not shown in the figure is a global clock signal, which drives all the registers in the tap.

Figure 6.15 The logical structure of the arithmetic unit

The logical structure of the arithmetic unit is shown in Fig. 6.15. There are two tasks for the arithmetic unit, calculating the tap output and updating the tap-weight. A complex multiplier, which consists of four real multipliers and two adders, is shared by these two tasks. The two tasks cannot be done simultaneously, there is a control signal ‘Sel’ to switch the arithmetic unit between the tasks. ‘Sel’ controls four multiplexors to select what data should be feed to the complex multiplier. Actually, the clock has been divided into two phases as shown in Fig. 6.16. During the phase 1, ‘Sel’ being high and turning the multiplexers to the data \((x_i(k), x_q(k))\) and the tap-weight, the arithmetic unit calculates the tap-value. During the phase 2, ‘Sel’ being low and turning the multiplexers to the data \((x_i(k-1), -x_q(k-1))\) and the gradient data \((e_i(k-1), e_q(k-1))\), the arithmetic unit updates the tap-weight. Tap-weight
values are stored in two D-registers.

Figure 6.16 The clock waveform

The Adder Tree

Adder tree adds up all the tap values to create the output data of the equalizer. Fig. 6.17 shows a tapped-delay-line structure with adder tree. The depth of the adder tree and the total number of adders in the adder tree are determined by the number of taps in the equalizer. In our design, there are 8 taps in each branch of tapped-delay-line and totally 16 taps in the equalizer, each tap output has 12 bits, we need an adder tree that has a depth of \( \log_2 16 = 4 \), and \( 2 \times (4 + 2 + 1) = 14 \) of 12-bit adders in total.

The Gradient Calculator

The error calculator generates the gradient data that taps need to update their tap-weights. The hardware structure of the gradient calculator as shown in Fig. 6.18 can be straightforwardly obtained from the numerical expression of the gradient for CMA. The gradient data is scaled by the step-size \( u \) before it is feed to each tap, so there is no step-size multiplication in the tap. To save hardware resources, the value of the step-size is chosen to be a number in the form of \( 1/2^m \), so the scaling operation can be achieved by simply using a shifter instead of a multiplier.
Figure 6.17 The structure of adder tree
The Data De-Multiplexor

The data de-multiplexer distributes the input data to each branch of the equalizer. For every two consequent input data, the data de-multiplexer feeds the first one to the 1st branch, then feeds the next data to the 2nd branch. The data de-multiplexer is comprised of two arrays of registers (see Fig. 6.19), one of them acts as a sampler, the other one acts as a buffer. There are two clocks to trigger the registers in the de-multiplexer, one is exactly the symbol clock ('clk' in Fig. 6.19), the other clock ('clk1') is the inverse of the symbol clock, there is a 180° phase difference between each of them. The rate of the input data is double that of the symbol rate. That means, in the register array that acts as the sampler, the two registers triggered by 'clk' and the two registers triggered by 'clk1' take turns to sample the input data (Fig. 6.16), producing two different data sequences feed to the buffer, both at the symbol rate.
Figure 6.19 The logical structure of the de-multiplexor
6.3.3 Demodulator and Carrier Recovery Loop

To test the performance of the channel equalizer, we designed a simple demodulator and a carrier recovery loop for experimental purpose. They are not the focus of this research.

![Diagram of the demodulator](image)

**Figure 6.20** The structure of the demodulator

The demodulator is built with the mixer and the low-pass filters as shown in Fig. 6.20. The function of the mixer is multiplying the incoming signal by the locally generated sinusoid signal to shift the spectrum of the signal. A straightforward implementation of mixer uses two multipliers, one each for the sine and cosine signals. In our design, the bandpass signal is sampled at a rate exactly 4 times of the carrier frequency, so the samples of the local sinusoid and cosine signals will take values of $\sin(\pi/2 * k)$ and $\cos(\pi/2 * k)$, respectively. Thus, the sinusoid and cosine signals can be replaced by the 0,1,0,-1,... sequences, which simplifies the design of the multipliers. The high frequency harmonics produced by the mixer will be filtered out by the low-pass filters.
Figure 6.21 The structure of the half-band filter
The low-pass filters are achieved by half-band filters. Half-band filter is a class of low-pass filter that has a cutoff frequency at 1/2 of the Nyquist frequency or 1/4 of the sampling frequency. In our case, the carrier frequency is exactly 1/4 of the sampling frequency, the cutoff frequency is the same as the carrier frequency, so the half-band filters filter out the signals above the carrier frequency and keep the baseband signals. Note that the balanced mixer structure of Fig. 6.20 results in output at baseband and at 2fc. There should be no output at fc. This allows the use of a LPF with cutoff frequency at fc. Half-band filter has the property that almost half of the filter coefficients are zeros, so the number of multipliers needed for a half-band filter is only half of that needed for an ordinary FIR filter of the same order. Fig. 6.21 shows the hardware structure of the half-band filter we used in this demodulator.

The carrier recovery module is based on the decision-directed method. As shown in Fig. 6.22, it consists of a decision-directed phase error detector, a loop filter, and a phase rotator. The phase error detector detects the phase error between the received signal and the decision signal. Then the phase error is fed through a low-pass loop filter to the phase rotator, which corrects the phase rotation caused by the carrier error.
Figure 6.22 The structure of the carrier recovery module
7. Real Time Simulation System Design

This chapter introduces the real time simulation system for testing the real-time performance of the channel equalizer hardware.

7.1 Test-bed

Ideally we hope to test the hardware in a real communication system, but building a test bed from a real communication system is very time-consuming and costly. A much more practical approach is to use a real time channel simulator that can be configured to simulate the channel characteristics of various real-world communication systems. Some professional channel simulators that provide real-time channel simulations are available on the market, but they are expensive. In our test-bed system, instead of using an expensive commercial channel simulator, we designed a simple channel simulator circuit ourself and implemented it in FPGA.

Fig. 7.1 is the block diagram of the test-bed. The signal source is the bandpass QAM signals that output from a RF signal generator. Here we use Agilent’s E4437B signal generator, which can generate QAM signals with different carrier frequencies and symbol rates. To test the performance of the channel equalizer for the bandpass QAM signals, we need to first convert the bandpass signal to the baseband signal that can be processed by the channel equalizer. For this purpose, a simple QAM demodulator is implemented to work as the front-end of the channel equalizer.

A logic analyzer collects the output data of the channel equalizer. The data can be exported from the logic analyzer in ASCII format for off-line analysis.
7.2 Channel Simulator

In the previous chapters, the analysis and simulation of the system is based on the baseband equivalent channel model while, in practice, most channels are bandpass channels. In our test-bed, the signal generated by the signal generator is a bandpass signal, and the channel simulated by the channel simulator is bandpass channel.

As we learn from the chapter 3, a bandpass channel can be described by its baseband equivalent channel model and center frequency. The center frequency is determined by the carrier frequency of the system. Theoretically, the characteristics of the baseband equivalent channel is independent of the carrier frequency. Once we have a baseband equivalent channel, we can obtain a bandpass channel at any center frequency.

Many baseband equivalent channels can be expressed in the form of finite lengthened impulse response sequence and hence can be simulated by using FIR filters. This theory applies to bandpass channels too, because a bandpass channel has a simple relationship with its baseband equivalent channel as such (please refer to chapter 3
The above formula leads to a simple digital implantation of bandpass channel:

\[ c_{pass}(t) = Re[c_{base}(t)e^{jw_c t}] \]  \hspace{1cm} (7.1)

A bandpass channel can also be obtained from its baseband equivalent channel in frequency domain:

\[ C_{pass}(z) = Re[\sum_{k=0}^{N-1} c_{base}(k)e^{jw_c z^{-k}}] \]  \hspace{1cm} (7.2)

\[ c_{pass}(k) = IFFT[C_{base}(w - w_c) + C_{base}^*(-w - w_c)], \]  \hspace{1cm} (7.3)

where \( C_{base}(w) \) is the frequency response of the baseband equivalent channel and the IFFT means Inverse Fast Fourier Transform.

---

**Figure 7.2** The channel simulator
In the test-bed, the channel simulator actually is a digital FIR filter implemented in a FPGA with an external A/D converter (see Fig. 7.2). The digital channel simulator has many advantages over the analog channel simulator. It is much easier to implement the tapped-delay line structure of the digital FIR filter in digital circuits than in analog circuits. In addition, the digital channel simulator is more accurate and flexible, compared with the analog simulator.

The FPGA device that we use for hardware implementation is Altera's Apex20K1000, a high density FPGA chip that contains 51,840 logic elements (LEs) or equivalently 1 million typical gates [19]. One Apex20k1000 chip is large enough for all the channel simulator, the QAM demodulator and the equalizer, so they are all implemented in the same FPGA chip.
8. Discussion on the Real Time Simulation Results

The impulse response of a bandpass channel for real time simulation is shown in Fig. 8.1. The length of the impulse response sequence is 32, so the channel simulator is implemented as a 32-order FIR filter. The source signal has a symbol rate of 6MS/s and is 16QAM modulated. The carrier frequency of the QAM signal is 6MHz, and so is the center frequency of the channel. The sampling frequency of the A/D converter is 24MHz. The channel simulator and the demodulator both work at 24MHz, but the equalizer re-samples the signal at 12MHz (two samples for one symbol), so the equalizer is a T/2-spaced equalizer. There are two branches of tapped-delay-line in the equalizer; each of them has 8 taps.

As the result of the real time simulation, the signal constellations of the equalizer output at different period are displayed in Fig. 8.2 to Fig. 8.5. We can see a clear signal constellation of the equalizer output between (15000/4)th and (20000/4)th symbols, so the equalizer obviously converges after (15000/4)th iterations.

A slight difference is noticeable between the carrier of the source signal and the local carrier of the demodulator by the rotation of I-Q axes, which results in phase errors in the signals at the receiver (see Fig. 8.2, Fig. 8.3 and Fig. 8.4), but finally the phase errors can be corrected by the carrier recovery circuit in the receiver (see Fig. 8.5). The carrier recovery circuit works simultaneously with the channel equalizer. This proves that the performance of CMA algorithm is very little affected by the carrier errors in the signals.

The equalizer we implemented is based on the hybrid algorithm introduced in Chapter 5, but in the real time simulation it failed to switch to the decision-directed...
Figure 8.1 The impulse response of the channel for real time simulation
mode. Our dual mode equalizer scheme is designed such that the decision directed mode should begin to work after the estimated mean squared error becomes under a threshold for a certain period of time. It has been observed from Fig. 8.6 that after 5000 iterations (equivalent to 5000 symbols period in time) from the initialization of the equalizer, the performance of the equalizer was severely degraded. This is due to the timing error in the demodulated signal. There is no timing recovery circuit in our simplified QAM demodulator to correct this timing error. A gate-level simulation of the equalizer design was carried out on software, which can imitate the work of the entire digital circuit of the equalizer. It has been found that, there is no performance degradation of the equalizer if there is no symbol timing error between the source and the receiver in the simulation on software, but the above mentioned phenomena has happened in the simulation on software if a timing error is added on to the system simulated. According to [18], a fractionally spaced equalizer is insensitive to the phase error of symbol clock. However the frequency error of symbol clock does affect a fractionally spaced equalizer. Referring to Fig. 3.5 in Chapter 3, the channel of a fractionally spaced system is made up of a series of sub-channels that are sampled at the symbol rate. Different symbol timing may result in different sampled channels. A phase error in symbol clock may result in a different channel, but, without symbol clock frequency error, the channel is still a constant channel and the equalizer still converges to a global minimum and stays in this global minimum. However, a frequency error in symbol clock changes the channel into a time varying channel. Therefore, even if the equalizer converges to a global minimum, it cannot stay in this global minimum. To re-converge to a global minimum, the equalizer needs to re-initialize itself. In most practical communication systems, equalizers begin to work after a correct timing recovery is achieved. Because of these reasons explained above, our dual mode equalizer did not switch to the decision directed mode in the current real-time test-bed.
Figure 8.2  The signal constellation of the equalizer output during first (5000/4) symbols
Figure 8.3  The signal constellation of the equalizer output during (5000/4)th (10000/4)th symbols
Figure 8.4 The signal constellation of the equalizer output during (10000/4)th (15000/4)th symbols
Figure 8.5  The signal constellation of the equalizer output during (15000/4)th (20000/4)th symbols
Figure 8.6 The signal constellation of the equalizer output after (20000/4)th symbols
9. Conclusion and Future Work

This thesis has investigated the blind channel equalization techniques for the high-speed digital communication systems, which often suffer from severe ISI problems. We have found a number of blind equalization algorithms in literature; some of them are based on the Higher Order Statistics (HOS) of the signals, and some just make use of the Second Order Statistics (SOS). We are interested in those algorithms that are low in computation complexity and easy to implement. The Constant Modulus Algorithm (CMA), which indirectly makes use of the 4-th order statistics, has such advantages and is most widely used in practice for blind channel equalization. The performance of the CMA algorithm has been evaluated through numerical simulations. The results of the simulations show that, the CMA can work for severely ISI corrupted FIR channels, but the algorithm itself converges very slow and has a large residual error. Attempting to overcome these shortcomings, we have designed a hybrid equalizer scheme that combines the CMA and a decision-directed technique. In the numerical simulations, the hybrid equalizer shows a better performance in both convergence speed and residual error. This thesis also includes a hardware design of blind equalizer for broadband communication applications. The hardware of equalizer has been implemented in an Altera FPGA chip. To test the real time performance of the equalizer hardware, we have built a test-bed that is comprised of an equalizer, a simplified QAM demodulator and a channel simulator. On this test-bed, real time simulations has been carried out, which shows a good performance of the CMA algorithm realized in the digital hardware.

As mentioned in Chapter 8, there is no timing recovery circuit in the test-bed. This imperfectness of the test-bed design limits the function of the test-bed; we are unable to test the entire hybrid algorithm in real time. For future work, we may need to
design a more sophisticated test-bed, which would enable us to study how equalizers work in a more practical situation. Such a demodulator should include carrier/timing recovery, AGC and other function blocks that a practical communication system may have.

In this research, we use 16QAM signal as source signals. Recently, more and more applications begin to use higher order QAM modulation systems such as 64QAM and 256QAM. It is of interest to know if the equalizer in this thesis, with or without possible modification, still works for higher-order QAM signals.

More difficult issues for further study may include the blind equalization algorithms for those unconventional channels such as time varying channels and IIR channels.
References


