2006

Blind adaptive equalization for QAM signals: New algorithms and FPGA implementation.

Kevin Banovic
University of Windsor

Blind Adaptive Equalization for QAM Signals: New Algorithms and FPGA Implementation

by

Kevin Banović

A Thesis
Submitted to the Faculty of Graduate Studies and Research through the Department of Electrical and Computer Engineering in Partial Fulfillment of the Requirements for the Degree of Master of Applied Science at the University of Windsor

Windsor, Ontario, Canada
2006

Your file Votre référence
ISBN: 978-0-494-17114-1
Our file Notre référence
ISBN: 978-0-494-17114-1

Abstract

Adaptive equalizers remove signal distortion attributed to intersymbol interference in band-limited channels. The tap coefficients of adaptive equalizers are time-varying and can be adapted using several methods. When these do not include the transmission of a training sequence, it is referred to as blind equalization.

The radius-adjusted approach is a method to achieve blind equalizer tap adaptation based on the equalizer output radius for quadrature amplitude modulation (QAM) signals. Static circular contours are defined around an estimated symbol in a QAM constellation, which create regions that correspond to fixed step sizes and weighting factors. The equalizer tap adjustment consists of a linearly weighted sum of adaptation criteria that is scaled by a variable step size. This approach is the basis of two new algorithms: the radius-adjusted modified multimodulus algorithm (RMMA) and the radius-adjusted multimodulus decision-directed algorithm (RMDA). An extension of the radius-adjusted approach is the selective update method, which is a computationally-efficient method for equalization. The selective update method employs a "stop-and-go" strategy based on the equalizer output radius to selectively update the equalizer tap coefficients, thereby, reducing the number of computations in steady-state operation. The transient performance of blind equalization algorithms that are modified by the selective update method experience similar transient performances, while their steady-state performance is dependent upon a static bound parameter. Simulation studies for RMMA and RMDA are completed for empirically-derived microwave radio and cable channels, while simulations for algorithms modified by the selective update method are completed for microwave radio and Ricean fading channels.

A custom implementation of a fractionally-spaced blind adaptive equalizer intellectual property (IP) core for QAM demodulators is presented, which is targeted for cable modems. The IP core

can be configured to operate in four error signal modes, which include RMMA and RMDA, and for square QAM signals up to 256-QAM. Implementation is completed for the Altera Stratix II EP2S130F780C4 FPGA and gate-level simulation is successful at a symbol frequency of 8 MBaud, which is comparable to recent QAM equalizer designs for cable modems.

To my family for their unending support.

Acknowledgments

There are several people who deserve my sincere thanks for their generous contributions to this project. First and foremost, I would like to express my sincere gratitude and appreciation to Dr. Esam Abdel-Raheem and Dr. Mohammed A. S. Khalid, my supervisors, for their invaluable guidance and support over the years. Their generosity, wisdom, and professionalism, will not be forgotten. Many thanks also go to my committee members, Dr. Kemel E. Tepe and Dr. Ahmed Tawfik; they have provided me with invaluable guidance and encouragement throughout the last couple of years. I would like to thank Dr. Moeness Amin and Dr. Khaled Mayyas, who lent their experience during discussions on adaptive filtering and Dr. Roberto Muscedere for his advice on hardware implementation. At this time, I would like to thank Raymond Lee and Harb Abdulhamid for reviewing the first draft of this thesis.

Next, I want to thank the colleagues of mine who made Essex Hall a thoroughly entertaining, if not always a productive, place to work. I have had the opportunity to work and develop friendships with some excellent people. These are, in order of appearance: Bijan Ansari, Songtao Huang, Jason Tong, Muqeeth Ali Syed, Raymond, Harb, Amir Ali Yazdanshenas, Ashkan Hosseinzadeh, and Ali Bidabadi. I will not soon forget the twenty-four hour simulation marathons, soccer matches, "game nights", or all the trips to 7-11 and Tim Hortons. We must have personally employed several of the staff members at the latter two with the amount of coffee we purchased. Then there was Covington, the Kentucky city on the Ohio river directly across from Cincinnati, where the 2005 Midwest Symposium on Circuits and Systems was hosted. An interesting city where Raymond and I went to White Castle, but not before seeing a show of gymnastics by our waitress at the Waffle House. The North American International Auto Show was another event that brought the group together. Not only did we witness automotive engineering at its finest, it provided a unique

opportunity for us to harness our skills as engineers. We successfully formulated a covert search operation to locate Harb, who go lost amid the masses of car crazy Americans. Finally, I would like to thank my parents Josip and Monika Banovic, and my brother Chris, for all their understanding and sacrifices over these last couple of years. Your support and encouragement has made my choice to undertake this degree a simple one.

My studies were funded by the National Sciences and Engineering Research Council (NSERC) and the University of Windsor, while the Canadian Microelectronics Corporation (CMC) has provided computer and FPGA workstations, access to leading VLSI software, and technical support.

Contents

List of Figures

List of Tables

List of Abbreviations

The notation used in this thesis is as follows. In general, bold face upper case letters designate matrices, bold face lower case letters designate vectors, and scalers are designated by italics. A scaler element of a vector is denoted $v_i \in \mathbf{v}$, which is read as "the i^{th} element of vector $\mathbf{v}$," with indexing of vectors starting at $i = 0$ and ranging to the length of the vector minus one. All vectors are assumed to be column vectors. Time is denoted in parenthesis for matrices, vectors, and scalers; for example, $v_i(n)$ is interpreted as the "i^{th}" element of vector $\mathbf{v}(n)$ at time instant "n". Some commonly used operators, symbols and abbreviations are listed below.

Abbreviation	Definition
$E\{\cdot\}$	Expectation operator.
$\text{csgn}(\cdot)$	Complex sign operator.
$\text{sgn}(\cdot)$	Real valued sign operator.
$\text{diag}\{\cdot\}$	Transforms a $N \times 1$ vector into a $N \times N$ diagonal matrix.
$\|\mathbf{x}\|_2$	Two norm of the vector '$\mathbf{x}$': $\sqrt{\sum_i \|x_i\|^2}$.
$\nabla_{\mathbf{w}}$	Gradient with respect to $\mathbf{w}$.
$(\cdot)^*$	Complex conjugation.
$(\cdot)^T$	Transposition.
$(\cdot)^H$	Transposition and conjugation.
$(\cdot)_R$	Real component of a complex number.
$(\cdot)_I$	Imaginary component of a complex number.
$\hat{x}$	The estimated value of 'x'.
$\Re\{\cdot\}$	Extraction of real-valued component.
$\Im\{\cdot\}$	Extraction of imaginary-valued component.
$\mathbf{I}$	Identity matrix.
$\mathbf{0}$	Zero matrix.
$\mathcal{F}$	FFT operator.
$\mathcal{F}^{-1}$	IFFT operator.
K	Rice factor.
L	Channel length.
N	Equalizer length.

T	Symbol period.
μ	Step size.
$\mathbf{c}$	Channel tap coefficient vector.
$\mathbf{w}$	Equalizer tap coefficient vector.
$\mathbf{x}$	Regressor vector of equalizer input samples.
e	Error signal.
s_m	Transmitted symbol: $s_m = \{a_m + jb_m\}_0^{M-1}$ for M-QAM constellations.
v	Additive white Gaussian noise.
y	Equalizer output.
$\xi_{\min}$	Minimum mean-squared error value.
ξ_{ss}	Steady-state mean-squared error value.
ADC	Analog-to-digital converter.
AGC	Automatic gain control.
ASIC	Application-specific integrated circuit.
ATSC	Advanced Television Systems Committee.
BCMA	Blended constant modulus algorithm.
BGA	Benveniste-Goursat algorithm.
CAD	Computer aided design.
CADAMA	CMA-assisted decision adjusted modulus algorithm.
CAP	Carrier-less amplitude/phase modulation.
CCA	Concurrent algorithm.
CMA	Constant modulus algorithm.
DAMA	Decision adjusted modulus algorithm.
DD	Decision-directed algorithm.
DDC	Digital down converter.
DFE	Decision feedback equalizer.
DOCSIS	Data over cable service interface specification.
DSP	Digital signal processing.
DSE	Dithered signed-error.
DVB	Digital Video Broadcasting.
EDA	Electronic design automation.
EMSE	Excess mean-squared error.
ETSI	European Telecommunications Standards Institute.
FBE	Feedback equalizer.
FFE	Feed-forward equalizer.
FIR	Finite impulse response.
FPGA	Field-programmable gate array.
FPID	Field-programmable interconnect device.
GSA	Generalized Sato algorithm.
HCGP	Hybrid complete-graph partial-crossbar.
HDTV	High definition television.
HW/SW	Hardware/software.
IC	Integrated circuit.
I/O	Input/output.
i.i.d.	Independent and identically distributed.
IPC	Interprocess communication.
ISI	Intersymbol interference.
ISS	Instruction-set simulator.
ITU	International Telecommunications Union.
IEEE	Institute of Electrical and Electronics Engineers.

Abbreviation	Definition
LUT	Look-up-table.
MAMA	MMA-assisted modulus algorithm.
MCMA	Modified constant modulus algorithm.
MMA	Multimodulus algorithm.
MMSE	Minimum mean-squared error.
MSE	Mean-squared error.
PAM	Pulse amplitude modulation.
PBE	Processor based emulator.
PCB	Printed circuit board.
PDSP	Programmable digital signal processor.
POT	Power-of-two (error).
PU	Partial update.
PXB	Partial-crossbar.
QAM	Quadrature amplitude modulation.
QPSK	Quadrature phase shift keying.
RMDA	Radius-adjusted multimodulus decision-directed algorithm.
RMMA	Radius-adjusted modified multimodulus algorithm.
RTL	Register transfer level.
SAG	stop-and-go algorithm.
SDD	Soft decision-directed algorithm.
SE	Signed-error.
SER	Symbol error rate.
SNR	Signal-to-noise ratio.
SoC	System-on-chip.
SU	Selective update.
UD	Update decimated.
TTC	Time-to-convergence or convergence time.
VDSL	Very high bit rate digital subscriber line.
VHDL	Very high speed integrated circuit (VHSIC) hardware description language.
VSB	Vestigal sideband modulation.

Chapter 1

Introduction

1.1 Blind Adaptive Equalization

Adaptive equalizers compensate for signal distortion attributed to intersymbol interference (ISI) which is caused by multipath within time-dispersive channels. They are typically employed in high-speed communication systems, which do not use differential modulation schemes or frequency division multiplexing. As illustrated by Fig. 1.1, there are several methods (or modes) in which equalizers can achieve adaptation of their tap coefficients. These include the transmission of a known training sequence, known symbol statistics, and decision-directed adaptation. When the methods applied to achieve channel equalization do not include the transmission of a training sequence, it is referred to as *blind equalization.* There exist many excellent tutorials on the subject of both trained [55][50] and blind equalization [68][80].

The basic data communications process can be explained with the simplified baseband equalizer block diagram of Fig. 1.1. A k-bit binary sequence is mapped to a symbol $s(n)$ which is pulse-shape filtered and modulated onto a band-limited communication channel. The received symbol $x(n)$ is corrupted by intersymbol interference (ISI) and Gaussian white noise. The equalizer removes the distortion caused by the channel by estimating the channel inverse. The equalizer output $y(n)$ is sent to a decision device, which results in the received symbol estimate $\hat{s}(n)$. The error computation, which determines the error signal $e(n)$ used to adjust the equalizer tap coefficients, depends on the equalizer mode of operation and the corresponding equalization algorithm applied. For a non-blind

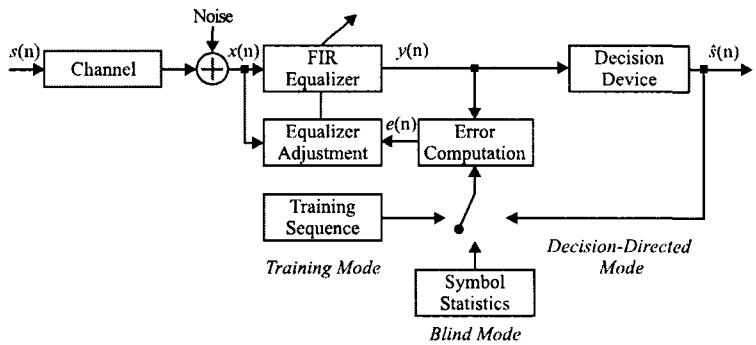


Figure 1.1: Simplified baseband model of a digital communication system.

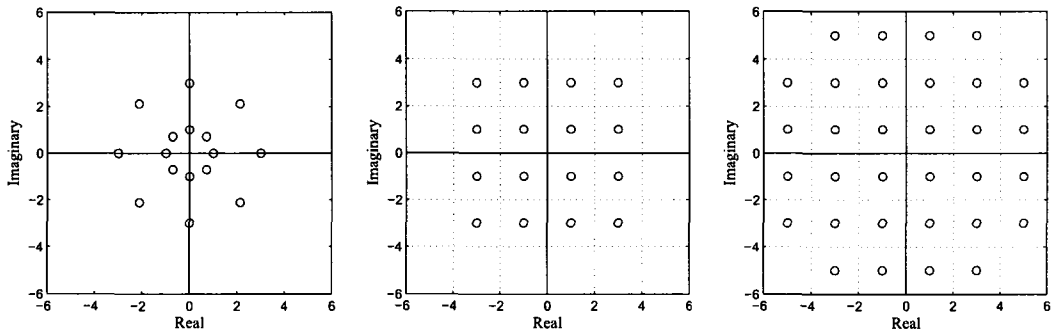
adaptive equalizer, such as those based on the least mean squares (LMS) algorithm, the equalizer will initially operate in the training mode. In this mode $s(n)$ is a training sequence known by the receiver and $e(n)$ will be calculated using the difference between $y(n)$ and $s(n)$. After convergence, the equalizer will be switched to the decision-directed (DD) mode, where $e(n)$ will be computed based on the difference between $y(n)$ and the estimated symbol $\hat{s}(n)$. For blind adaptive equalizers, such as those based on the constant modulus algorithm (CMA), the equalizer will initially operate in the blind mode, where $e(n)$ is a non-linear function of $y(n)$. After convergence, as for non-blind equalizers, the blind equalizer can be switched to the DD mode. Thus, the blind mode of an equalizer that switches to the DD mode after convergence is, in essence, the *training mode* of that equalizer.

1.2 Quadrature Amplitude Modulation

Quadrature amplitude modulation (QAM) is a passband digital transmission method that impresses two separate k-bit symbols onto the quadrature carriers $\cos(2\pi f_c t)$ and $\sin(2\pi f_c t)$, respectively, where f_c is the carrier frequency. The general M-ary QAM signal constellation is represented by the finite symbol set $\{s_m = a_m + jb_m\}_{m=1}^{M}$. Alternatively, a QAM symbol can be represented in polar notation as $A_m e^{j\theta_m}$, where $A_m = \sqrt{a_m^2 + b_m^2}$ and $\theta_m = \tan^{-1}(b_m/a_m)$. The modulated passband signal $s(t)$ is defined as

$$s(t) = \Re\left\{A_m(t)e^{j(2\pi f_c t + \theta_m(t))}\right\} \tag{1.1}$$

where $\Re\{\cdot\}$ represents the real part of a complex number. The modulated signal $s(t)$ is referred to as the narrow-band bandpass signal since $f_c \gg B$, where B is the bandwidth. The narrow-band



(a) Star 16-QAM. (b) Square 16-QAM. (c) Cross 32-QAM.

Figure 1.2: Various QAM signal constellations.

bandpass signal can be translated to an equivalent baseband signal by expanding (1.1) as follows

$$\begin{aligned} s(t) &= A_m(t)\cos\left(\theta_m(t)\right)\cos\left(2\pi f_c t\right) - A_m(t)\sin\left(\theta_m(t)\right)\cos\left(2\pi f_c t\right) \\ &= u_I\cos\left(2\pi f_c t\right) - u_Q\sin\left(2\pi f_c t\right) \end{aligned} \tag{1.2}$$

where $u_I = A_m(t)\cos\left(\theta_m(t)\right)$ and $u_Q = A_m(t)\sin\left(\theta_m(t)\right)$ are the inphase and quadrature components of $s(t)$, respectively. The complex bandpass envelope is given by

$$u(t) = u_I(t) + ju_Q(t) = A_m(t)e^{j\theta_m} \tag{1.3}$$

which when substituted into (1.1) allows $s(t)$ to be rewritten as

$$s(t) = \Re\left\{u(t)e^{j(2\pi f_c t)}\right\}. \tag{1.4}$$

This implies that the knowledge of $u(t)$ and f_c uniquely describes the modulated signal $s(t)$, where $u(t)$ contains all the useful information.

As indicated by Fig. 1.2, there are several different types of QAM signal constellations that can be used for QAM transmissions. When an even number of bits is to be encoded, the square type of QAM constellation is optimal for Gaussian channels [54][37], while cross-QAM constellations are typically used to encode an odd number of bits. Since odd-bit constellations have a higher encoding and decoding complexity, even-bit square constellations are significantly more common than odd-bit cross-QAM constellations [37]. Square QAM constellations, mainly 16-QAM, 64-QAM and 256-QAM will be considered throughout the remainder of this thesis.

Recently, QAM-based communication standards were adopted for satellite, cable, and VDSL applications. Blind equalization is recommended for both the Pan-European satellite-based Digital Video Broadcast (DVB-S) [1] and cable-based (DVB-C) [3] standards. Broadband standards

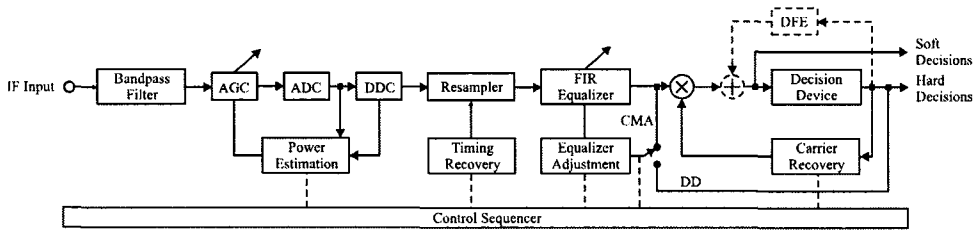


Figure 1.3: The block diagram of a generic blind demodulator for QAM signals.

for VDSL include provisions for both single- and multiple-carrier modulation [52]. The latter uses CAP/QAM modulation and requires the receiver to startup blindly (i.e. blind equalization), where CAP is carrier-less amplitude phase modulation. Although the Advance Television Systems Committee (ATSC) adopted 8-VSB over 32-QAM for terrestial HDTV broadcast [2], blind decision-feedback equalization (DFE) was chosen over trained equalization. In field tests conducted by HDTV manufacturers, the blind DFE achieved a lower error rate and faster data acquisition than its trained counterpart in time-varying terrestrial channels [31].

1.3 Demodulation of QAM Signals

In Fig. 1.3, the block diagram of a blind data demodulator for QAM signals is illustrated. The demodulation process begins by bandpass filtering the intermediate frequency (IP) input signal and adjusting the signal strength with the automatic gain control (AGC). The analog signal is transformed into digital words using an analog-to-digital converter (ADC). The resolution of this converter determines the overall quality of the demodulated signal and is typically between $8 - 14$ bits. A digital down converter (DDC) transforms the IF signal down to baseband and generates the inphase and quadratures components from the real input signal. All digital timing recovery is used to acquire the symbol timing and "time slice" the input signal to obtain the proper pulse amplitude and phase measurements. A blind FIR equalizer compensates for the channel distortion caused by ISI which is attributed to multipath within time-dispersive channels. An optional feedback equalizer (FBE) can be utilized to add past "assumed correct" decisions to the current equalizer output. This scenario is referred to as decision feedback equalization (DFE), whereby the FIR equalizer in Fig. 1.3 is referred to as the feed forward equalizer (FFE) and reduces the precursor ISI, while the FBE reduces the postcursor ISI. The equalizer can be adapted in two modes of operation, a blind mode and a decision-directed mode. At the onset of equalization and prior to convergence, the

equalizer will operate in the blind mode using a blind algorithm such as CMA. Once the distortion is minimized, the equalizer is switched to the decision-directed mode, thereby reducing the mean-squared error (MSE). The phase of the equalizer output is adjusted based on the correct carrier phase obtained by the carrier recovery circuit. Lastly, the phase adjusted equalizer output will be sliced into digital words, thereby recovering the original sent symbol.

1.4 Thesis Objectives

The work presented in this thesis conforms to the following objectives:

1. Investigate new and existing algorithms, including computationally-efficient methods, that achieve blind channel equalization for QAM signals.
2. Develop a custom blind adaptive equalizer intellectual property (IP) core, which targets QAM data demodulators for cable modems.
3. Implement the blind equalizer IP core for an Altera Stratix II FPGA.

The main challenges are to develop new algorithms for blind adaptive equalization with enhanced performance, while considering efficient implementation. Hybrid equalization algorithms can improve the transient and steady-state performance, however, they tend to be expensive in terms of arithmetic computations. While the physical synthesis process for FPGAs is simpler than that for a custom ASIC implementation, FPGAs are less flexible and there is limited logic and routing resources. This limits the size and symbol frequency of the implementation.

1.5 Thesis Organization

This thesis is organized as follows: Chapter 2 covers design methodologies for FPGA prototyping of DSP systems which include register-transfer level (RTL) design, hardware-software co-design and FPGA-based hardware emulation. Chapter 3 details adaptive equalization fundamentals beginning with minimum mean-squared error (MMSE) equalization and the non-blind least mean-squares algorithm. The chapter continues with a discussion on blind and hybrid blind equalization algorithms and closes with computationally-efficient methods that apply to both blind and non-blind adaptive equalizers. Chapter 4 introduces the radius-adjusted approach for QAM signals, two new algorithms: radius-adjusted modified-MMA algorithm (RMMA) and radius-adjusted MMA-DD algorithm (RMDA), and extends the radius-adjusted approach to computationally-efficient methods.

Chapter 5 presents the results of simulation studies for new and existing blind equalization algorithms and computationally-efficient methods. Chapter 6 details the design and implementation of a custom blind adaptive equalizer that is implemented on an Altera Stratix FPGA. Lastly, Chapter 7 provides concluding remarks and details future work.

Chapter 2

DSP System Design with FPGAs

2.1 Introduction

DSP algorithms have traditionally been implemented using application-specific integrated circuits (ASICs) or programmable digital signal processors (PDSPs). However, with the introduction of large capacity FPGAs, there has been a shift towards reconfigurable computing for DSP [78][73][23]. The fine-grained parallelism of FPGAs coupled with the inherent data parallelism found in many DSP functions, have made reconfigurable computing a viable alternative that offers a compromise between the performance of fixed-functionality hardware and the flexibility of software-programmable devices. As opposed to PDSPs, FPGAs allow non-standard word-length sizes and semi- or full-parallel signal processing, which can reduce implementation area and improve throughput. Additionally, FPGA based emulation platforms can offer real-time prototyping of ASIC logic, which allows system verification and optimization in an environment which resembles the target system[1].

This chapter presents a survey of DSP design methodologies and computer-aided design (CAD) tools for FPGAs, including methodologies for standard register-transfer-level (RTL) design, system-level design, and hardware/software (HW/SW) co-design. The application of FPGA emulation systems as a platform for rapid prototyping is addressed and the future trends of FPGA-based DSP

[1]©2005 IEEE. Reprinted, with permission from K. Banovic, M. A. S. Khalid, and E. Abdel-Raheem, "FPGA-Based Rapid Prototyping of Digital Processing Systems", in Proc. of the 48th International Midwest Symposium on Circuits & Systems, Cincinnati, Ohio, August 2005, pp. 647-650.

systems are discussed.

2.2 Standard RTL Design

Traditionally, FPGA design has paralleled ASIC design and has used similar design flows and methodologies. The basic top-down FPGA design flow, which is illustrated in Fig. 2.1, generally requires two distinct sets of design tools; the first for algorithmic development and analysis and the second for hardware synthesis and implementation. This creates a gap between algorithmic design and hardware specification within the design flow, which prevents it from being a true top-down design flow.

The first step in any DSP design is algorithmic development and analysis, which is typically performed at a high level of abstraction in a floating-point environment such as C/C++ or Matlab. The resulting floating-point algorithmic model is converted into a fixed-point model, and simulations are performed to verify their equivalence. Hardware specifications are based on the fixed-point representation and are used to manually create RTL models and testbenches. RTL design refers to the methodology of modeling a sequential circuit as a set of registers and a set of transfer functions which describe the flow of data between the registers. The design is simulated at the RTL level to confirm functionality. However, timing and resource usage remain unknown until gate-level simulation. Logic synthesis is performed to create an optimized gate-level netlist which is based on design constraints such as timing, area, and power. Synthesis constraints directly affect the effort required for placement and routing. If the design is over-constrained, routing failure may occur since routing resources are fixed in FPGAs. Physical synthesis follows logic synthesis, which is typically carried out using FPGA vendor place and route tools. The incoming netlist goes through design rule checking and is partitioned into available logic resources. An optional step of floorplanning can be carried out to reduce routing delay and area by assigning portions of the design to specific regions of the FPGA. Timing-driven placement and routing is performed, completing the physical synthesis process. Routing delays are back annotated to the gate-level netlist for final simulation and timing analysis. In order to verify the design, equivalence checking is carried out after both logic synthesis and physical synthesis. The last step in the design flow is the generation of a bit file to program the FPGA.

Although not indicated by Fig. 2.1, the FPGA design flow is an iterative process that requires many intermediate steps. There are a number of electronic design automation (EDA) tools targeting FPGA design that are offered by both FPGA and third party vendors. Complete design environ-

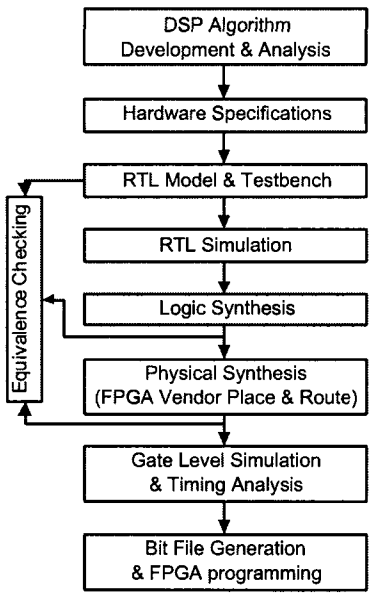


Figure 2.1: Standard RTL design flow.

ments are offered by Xilinx (ISE), Altera (Quartus II), and Mentor Graphics (FPGA Advantage), which integrate with a number of third party EDA tools. In an attempt to reduce development times, DSP design has gravitated towards the use of intellectual property (IP) cores and customizable designware for common DSP functions. These customizable cores can be integrated within the users own design, which can lead to reduced development times and more efficient designs.

2.3 System-Level Design

Most DSP designers work at the algorithmic level and use C/C++ or Matlab for verification and architectural exploration. While this reduces time for system verification it has created a gap between algorithmic development and hardware specification. In the standard RTL design flow, the process of translating the floating-point C/C++ or Matlab model to RTL is a manual error-prone process that prolongs the design cycle and makes it difficult to make changes at the algorithmic level. This has led to the development of several design tools that automate the conversion between high level languages and RTL to create a true top-down design methodology. This has reduced the design cycle, encouraged more algorithmic exploration and optimization and has allowed C/C++ or Matlab to remain the main source throughout the design flow.

Table 2.1: Third-party algorithmic synthesis CAD tools.

Product	Abstraction
AccelChip DSP	Matlab
Synplify DSP	Simulink
Mentor Graphics Catapult	C++
Forte Design Systems Cynthesizer	SystemC
CoWare SPW	Symbolic

Algorithmic synthesis solutions automate the generation of RTL models and testbenches from floating-point models and provide tools for auto-quantization, architectural definition, RTL model generation and area/timing optimization. Design solutions for DSP algorithmic synthesis are offered from FPGA vendors Xilinx (System Generator for DSP) and Altera (DSP Builder), third party vendors such as AccelChip (AccelChip DSP) and Synplicity (Synplify DSP), as well as academia [22]. A selected list of third party algorithmic synthesis tools is shown in Table 2.1. A re-occurring trend among vendors such as Xilinx, Altera, AccelChip and Synplicity, are integrated Matlab/Simulink based design solutions that contain libraries of parameterized fixed-point DSP building blocks. This enables designers to work in a familiar environment and to integrate their bit-true and cycle accurate[1] model within a full system, which allows for rapid system verification.

2.4 Hardware/Software Co-design

In the context of FPGA design, hardware/software (HW/SW) co-design refers to the methodology, tools, and practices that support the design of embedded systems and systems-on-programmable-chip (SoPC). An embedded system consists of one or more processors (which include PDSPs) and ASICs connected to a common bus, which provides an environment with both multiprogramming and multiprocessing capabilities [84]. The challenge of HW/SW co-design is to produce a near-optimal HW/SW design that meets system requirements within the design constraints. Traditionally, partitioning would occur early in the design flow, which resulted in the hardware and software being developed independently with minimal interaction between them due to the lack of a unified representation [45]. However, in modern HW/SW co-design, languages such as C/C++ are used to represent both hardware and software components, which has allowed partitioning to occur at a later stage. This has increased the interaction and feedback between design partitions and encourages architectural exploration.

The generic HW/SW co-design flow, which is illustrated in Fig. 2.2, begins with system spec-

[1]Refers to the number of clock cycles needed to perform an operation during RTL simulation.

Table 2.2: Third-party co-design CAD tools.

Product	Abstraction
Celoxica DK Design Suite	C
Impulse Accelerated Technologies CoDeveloper	C
Synfora PICO Express	C
Poseidon Design Systems Triton Builder	C

ification followed by high-level algorithmic development. After sufficient analysis, the design is partitioned into hardware and software components based on speed, complexity and flexibility requirements. Components that are better suited to software, such as complex algorithms, are assigned to software partitions, while components that can be accelerated in hardware, such as sensor applications, are assigned to hardware partitions. RTL and C/C++ models are generated for the hardware and software components, respectively, by manual or automatic methods. HW/SW co-simulation follows, which links an RTL simulator for hardware with an instruction-set simulator (ISS) for software. An interface between the simulators is necessary and typically consists of a bus wrapper and interprocess communication (IPC). The bus wrapper synchronizes the ISS and system simulation and translates the incoming data from the ISS into cycle-accurate bus transactions [13], while IPC primitives are used to communicate with the distinct processes run by the ISS on the host system. Based on co-simulation results, the designer determines whether to continue with the current architecture or explore different architectures by choosing a new HW/SW partition. Once an architecture has been chosen and the system has been verified, the synthesis of hardware components follows that of the standard RTL design from logic synthesis onward, while the software components go through compilation and debugging stages.

Platforms for embedded system design are offered by FPGA vendors Xilinx (EDK) and Altera (SoPC Builder) that are based on their MicroBlaze and Nios soft-core processors, respectively. They allow complex hardware and software systems to be developed manually through the use of parameterizable IP cores, custom hardware acceleration and extensive software debugging tools. There are a number of C-based co-design tools available from third party vendors, several of which are listed in Table 2.2, as well as from academia [7]. Co-design solutions from Celoxica, Impulse Accelerated Technologies and Synfora combine configurable IP and architectural exploration tools to create HW/SW partitions, generate RTL directly from C models, and create software interfaces for co-simulation. This allows the exploration of different HW/SW implementations, which enables trade-offs based on area/performance.

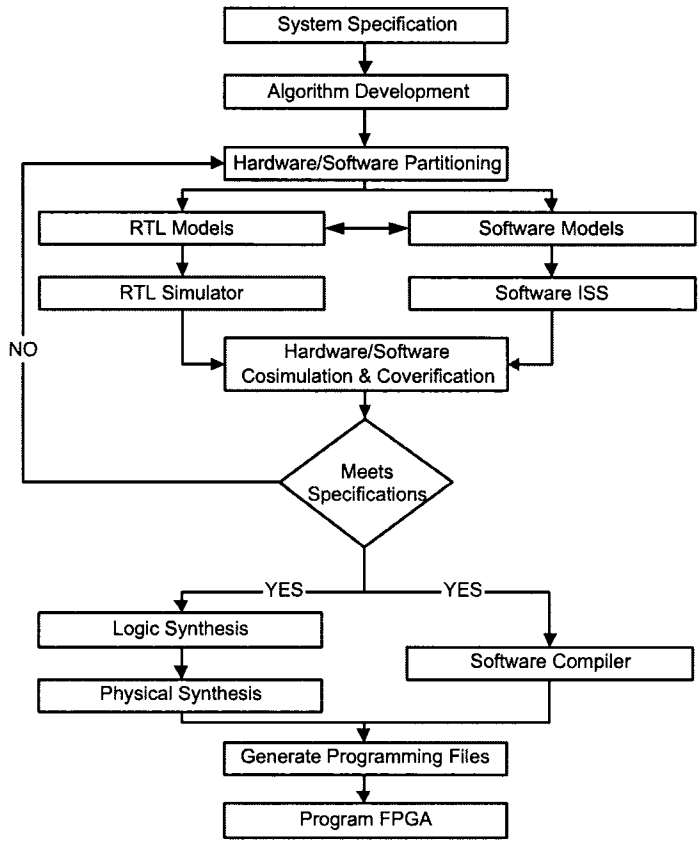


Figure 2.2: Hardware/software co-design flow.

2.5 Emulation and Prototyping of DSP Systems

Hardware emulation is as an alternate approach to system verification, which reduces simulation time while retaining high confidence in results. A hardware emulator is a reconfigurable computer that can be programmed to emulate a large digital design. The prototyping of ASIC circuits as well as the development and verification of new IP cores are applications that are well suited to emulation. FPGAs and multi-FPGA emulation systems, such as the Berkeley Emulation Engine (BEE) [26], are highly flexible platforms for the rapid prototyping of DSP systems.

Rapid prototyping is the accelerated development of a physical system for demonstration, evaluation, testing, or verification. Real-time prototyping allows the testing and optimization of more design parameters in shorter time and in an environment that closely resembles the target system [16]. There are three main classes of prototyping systems; concept-, architecture-, and implementation-oriented prototyping [70]. Concept-oriented prototyping explores the design requirements and specifications of the system, and generally consists of hardware-accelerated simulation or computation [48]. Architecture-oriented prototyping consists of the system-level design, testing, and verification, as well as subsystem specification. This level of prototyping supports HW/SW partitioning and co-simulation, allowing software co-development to occur at an earlier design stage [48]. Finally, implementation-oriented prototyping consists of module design and RTL or gate-level testing and verification. As illustrated in Fig. 2.3, the three classes of prototyping interact with each other and follow a natural progression to system implementation [70].

FPGA based emulation systems have been the dominant platform for ASIC prototyping for many years. With current FPGA capacities exceeding six million gates (Xilinx Virtex-4 families); it is possible to emulate large system-on-chip (SoC) designs on a single FPGA. Synopsys offers an ASIC prototyping solution called DC FPGA. DC FPGA eliminates the need for manual conversion between the ASIC and FPGA designs, which allows a single RTL design to be developed for both prototyping and implementation.

There have been several multi-FPGA emulation systems developed that have the logic capacity to emulate large ASIC designs; namely emulators from Mentor Graphics (VStationPRO) and BEE [26]. These systems consist of on upwards of several hundred FPGAs, which are distributed across multiple PCBs. Inter-FPGA routing is accomplished by several routing architectures that may utilize field programmable interconnects (FPID). These include the partial-crossbar (PXB) and the hybrid complete-graph partial-crossbar (HCGP) [46]. Multi-FPGA emulation systems have high place and route times, and partitioning is restricted by the FPGA pin count, which reduces the FPGA logic utilization. Routing delays reduce maximum frequency to a fraction of that of a single

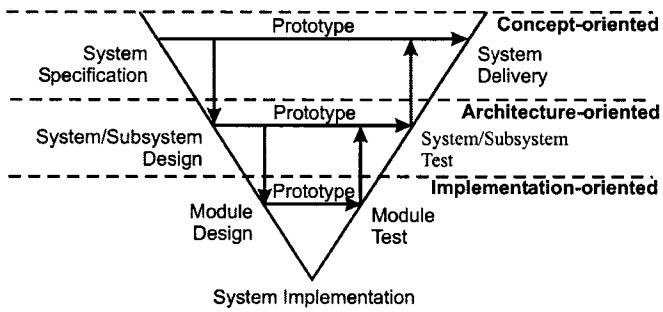


Figure 2.3: VP model for rapid prototyping.

FPGA. Recently, there has been a shift towards processor-based emulation (PBE) systems such as Palladium 2 from Cadence. These emulators consist of highly parallel hardware processors that are used to emulate ASIC logic.

2.6 Future Trends

The focus of the FPGA community has been divided into two main groups: signal processing and embedded system design. In the area of signal processing, FPGAs will continue to be a dominant platform for front end and sensor DSP applications [78]. In order to keep up with the demands of signal processing, the number of embedded multipliers and processors on the fabric of FPGA chips and prototyping boards will increase, which will enable FPGAs to better compete against PDSPs. As the performance gap between system-level and RTL design decreases, system-level design methodologies will become the convention. This shift has already begun with the plethora of system-level design tools available and the use of IP cores and customizable designware. In the area of embedded system design, less application-specific design tools and methodologies which are able to obtain efficient HW/SW partitions are essential for co-design methodologies to obtain widespread acceptance. A critical step is the development of sophisticated profiling tools that can efficiently partition designs into hardware and software based on execution performance. Lastly, FPGA-based emulation systems are expected to continue to rival PBE-based emulation systems as the dominant platform for the rapid prototyping of ASIC logic.

Chapter 3

Trained and Blind Adaptive Equalization

3.1 Introduction

This chapter discusses the underlying principles and methods applied for channel equalization in the context of single input single output (SISO) systems. The traditional receiver design for a linear modulation system consists of a filter matched to the actual channel cascaded with a T-spaced equalizer [55][54][38], where T is the symbol period. This senario is commonly referred to as baud-spaced equalization. T-spaced equalizers are sensitive to the sampling phase and, in theory, they require an infinite number of taps to achieve perfect equalization. An alternate equalizer realization is a T/M-spaced equalizer, where M is an integer number greater than one. This scenario is commonly referred to as fractionally-spaced equalization. The T/M-spaced equalizer, by virtue of its sampling rate, can synthesize the optimal combination of characteristics of a matched filter and T-spaced equalizer, within the constraints of its length and delay [38][55]. The performance of T/M-spaced equalizers is insensitive to the choice of sampler phase and, in theory, a fractionally-spaced equalizer can achieve perfect equalization with a finite number of taps [38][24]. For these reasons, equalizer design has gravitated to fractionally-spaced equalization typically with $M = 2$. Throughout the remainder of this thesis, $T/2$-spaced equalization will be considered exclusively.

This chapter begins with the derivation of the fractionally-spaced system model [44][38] and continues with an introduction to the mean-squared error (MSE) cost function and the minimum MSE (MMSE) equalizer which provides the theoretical optimal solution for MSE-based adaptive

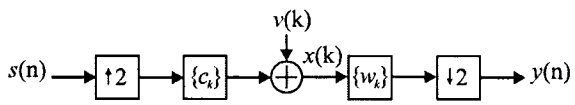


Figure 3.1: Multirate system model.

algorithms. The trained least-mean-squares (LMS) algorithm [83] is presented as a search method which simplifies the gradient calculation and the LMS equalizer tap coefficient update is derived for both real and complex signals [82]. A discussion of blind algorithms follows, which include the fundamental blind algorithms of Sato [59], Godard [33], and Yang *et al.* [88]. Hybrid methods for blind equalization are introduced and lastly, computationally-efficient methods that apply to both blind and non-blind equalizers are discussed.

3.2 Fractionally-Spaced System Model

In this section, a signal model is constructed for the $T/2$-spaced SISO baseband communication system, where T is the symbol period and $1/T$ is the baud rate. A multirate model of the system is illustrated in Fig. 3.1, where the index 'n' denotes T-spaced quantities while 'k' denotes $T/2$-spaced quantities. A T-spaced source symbol $s(n)$ is transmitted through a pulse-shaping filter and modulated onto a $T/2$-spaced propagation channel, whose impulse response is given by the finite series $\{c_k\}_{k=0}^{L-1}$, where L is the channel length. This corresponds to the $L \times 1$ channel impulse response vector of $\mathbf{c} = [c_0, c_1, \ldots, c_{L-1}]^T$ where $(\cdot)^T$ is the transpose operator and the channel is stationary[1]. The source symbol is a random variable that is independent and identically distributed (i.i.d.) with variance $\sigma_s^2 = E\{s(n)\}$ and is drawn from a finite alphabet, which is given by the finite set $\{s_m = a_m + jb_m\}_{m=1}^{M}$ for an M-QAM constellation, while $E\{\cdot\}$ is the expectation operator. The received $T/2$-spaced input signal $x(k)$ is corrupted by ISI and the additive white Gaussian noise signal $v(k)$. The baseband receiver consists of an N-tap $T/2$-spaced linear equalizer, whose tap coefficients are characterized by the finite series $\{w_k\}_{k=0}^{N-1}$, which corresponds to the $N \times 1$ vector $\mathbf{w}(n) = [w_0(n), w_1(n), \ldots, w_{N-1}(n)]^T$. The $T/2$-spaced convolution matrix is constructed from the

[1]The channel need not be stationary. A time-varying channel can be used as long as it does not vary faster than can be tracked by the equalization algorithm.

channel impulse response vector and is defined as

$$\mathbf{C}_{FS} = \begin{bmatrix} c_0 & & & & \\ c_1 & c_0 & & & \\ \vdots & c_1 & c_0 & & \\ c_{L-1} & \vdots & c_1 & \ddots & \\ & c_{L-1} & \vdots & \ddots & c_0 \\ & & c_{L-1} & & c_1 \\ & & & \ddots & \vdots \\ & & & & c_{L-1} \end{bmatrix} \tag{3.1}$$

where $\mathbf{C}_{FS}$ is an $(L+N-1) \times N$ matrix. The T-spaced convolution matrix is formed by the odd rows of (3.1) and is defined as

$$\mathbf{C} = \begin{bmatrix} c_1 & c_0 & & & & & \\ c_3 & c_2 & c_1 & c_0 & & & \\ \vdots & \vdots & c_3 & c_2 & \ddots & & \\ c_{L-1} & c_{L-2} & \vdots & \vdots & \ddots & c_1 & c_0 \\ & & c_{L-1} & c_{L-2} & & c_3 & c_2 \\ & & & & \ddots & \vdots & \vdots \\ & & & & & c_{L-1} & c_{L-2} \end{bmatrix} \tag{3.2}$$

where $\mathbf{C}$ is a $P \times N$ matrix, while $P = \lfloor (L+N-1)/2 \rfloor$. The regressor vector of equalizer input samples is comprised of the previous N received $T/2$-spaced samples and is defined as

$$\mathbf{x}(n) = \mathbf{C}^T \mathbf{s}(n) + \mathbf{v}(n) \tag{3.3}$$

where $\mathbf{s}(n) = [s(n), s(n-1), \ldots, s(n-P)]^T$ is the $P \times 1$ source symbol vector and the $N \times 1$ additive white Gaussian noise vector is $\mathbf{v}(n) = [v_0(n), v_1(n), \ldots, v_{L-1}(n)]^T$. The equalizer output is decimated by a factor of two and is defined as

$$\begin{aligned} y(n) &= \mathbf{x}^T(n)\mathbf{w}(n) \\ &= \mathbf{s}^T(n)\mathbf{C}\mathbf{w}(n) + \mathbf{v}^T(n)\mathbf{w}(n). \end{aligned} \tag{3.4}$$

Lastly, the $P \times 1$ vector of the noiseless T-spaced combined channel-equalizer impulse response is defined as

$$\mathbf{h}(n) = \mathbf{C}\mathbf{w}(n). \tag{3.5}$$

3.3 Minimum Mean-Squared-Error Equalization

Although the mean-squared error (MSE) criteria is, in general, not optimal in the sense of minimizing the symbol error rate (SER) [38], variants of its cost function are widely used for equalizer design [83][24]. This is in part due to the simplicity of the MSE cost function and its unimodal performace surface. The result of minimum mean-squared error (MMSE) equalization is an exact solution for the equalizer tap coefficients, providing the theoretical minimum for MSE-based equalization algorithms. The MSE criteria attempts to minimize the expected squared magnitude of the recovery error $e(n) = d(n) - y(n)$, where $d(n)$ is the desired signal. The desired signal is the transmitted symbol delayed by δ so that $d(n) = s(n - \delta)$. The MSE cost function is defined as

$$\begin{aligned} J^{\text{mse}} &= E\left\{e^2(n)\right\} \\ &= E\left\{d^2(n) - 2d(n)y(n) + y^2(n)\right\} \\ &= E\left\{d^2(n) - 2d(n)\mathbf{w}^T(n)\mathbf{x}(n) + \mathbf{w}^T(n)\mathbf{x}(n)\mathbf{x}^T(n)\mathbf{w}(n)\right\} \\ &= E\left\{d^2(n)\right\} - 2E\left\{d(n)\mathbf{w}^T(n)\mathbf{x}(n)\right\} + E\left\{\mathbf{w}^T(n)\mathbf{x}(n)\mathbf{x}^T(n)\mathbf{w}(n)\right\}. \end{aligned} \tag{3.6}$$

When the filter coefficients are fixed, the cost function in (3.6) is not time-varying and can be rewritten as

$$\begin{aligned} J^{\text{mse}} &= E\left\{d^2(n)\right\} - 2\mathbf{w}^T \underbrace{E\left\{d(n)\mathbf{x}(n)\right\}}_{\mathbf{p}} + \mathbf{w}^T \underbrace{E\left\{\mathbf{x}(n)\mathbf{x}^T\right\}}_{\mathbf{R}} \mathbf{w} \\ &= E\left\{d^2(n)\right\} - 2\mathbf{w}^T\mathbf{p} + \mathbf{w}^T\mathbf{R}\mathbf{w} \end{aligned} \tag{3.7}$$

where $\mathbf{p} = E\left\{d(n)\mathbf{x}(n)\right\}$ is the cross-correlation vector between the desired signal and the input signal and $\mathbf{R} = E\left\{\mathbf{x}(n)\mathbf{x}^T(n)\right\}$ is the input correlation matrix, which can be expanded as follows

$$\mathbf{R} = \begin{bmatrix} E\left\{x_0^2(n)\right\} & E\left\{x_0(n)x_1(n)\right\} & \cdots & E\left\{x_0(n)x_{N-1}(n)\right\} \\ E\left\{x_1(n)x_0(n)\right\} & E\left\{x_1^2(n)\right\} & \cdots & E\left\{x_1(n)x_{N-1}(n)\right\} \\ \vdots & \vdots & \ddots & \vdots \\ E\left\{x_{N-1}(n)x_0(n)\right\} & E\left\{x_{N-1}(n)x_1(n)\right\} & \cdots & E\left\{x_{N-1}^2(n)\right\} \end{bmatrix} \tag{3.8}$$

where $\mathbf{x}(n) = [x_0(n), x_1(n), \ldots, x_{N-1}(n)]^T$ is the time-varying regressor vector of equalizer input samples defined in (3.3). The gradient of the MSE cost function with respect to the equalizer tap weights is defined as

$$\begin{aligned} \nabla_{\mathbf{w}} J^{\text{mse}} &= \frac{\partial J^{\text{mse}}}{\partial \mathbf{w}} = \begin{bmatrix} \frac{\partial J^{\text{mse}}}{\partial w_0} & \frac{\partial J^{\text{mse}}}{\partial w_1} & \cdots & \frac{\partial J^{\text{mse}}}{\partial w_{N-1}} \end{bmatrix} \\ &= -2\mathbf{p} + 2\mathbf{R}\mathbf{w}. \end{aligned} \tag{3.9}$$

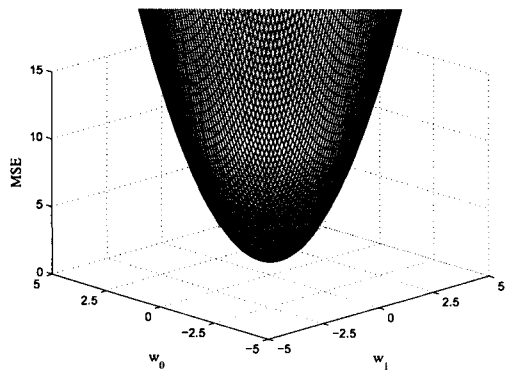


Figure 3.2: Unimodal MSE performance surface example.

The optimal equalizer taps $\mathbf{w}_o$ required to obtain the minimum mean-squared error (MMSE) can be determined by equating (3.9) to zero and solving for $\mathbf{w}_o$ as follows

$$0 = 2\mathbf{R}\mathbf{w}_o - 2\mathbf{p} \rightarrow \mathbf{w}_o = \mathbf{R}^{-1}\mathbf{p} \tag{3.10}$$

where the input correlation matrix $\mathbf{R}$ is assumed to be invertible. The MMSE, which is denoted $\xi_{\min}$, is obtained by substituting (3.10) for $\mathbf{w}$ in (3.7) as follows

$$\begin{aligned} \xi_{\min} &= E\left\{d^2(n)\right\} - 2\mathbf{w}_o^T\mathbf{p} + \mathbf{w}_o^T\mathbf{R}\mathbf{w}_o \\ &= E\left\{d^2(n)\right\} - 2\left[\mathbf{R}^{-1}\mathbf{p}\right]^T\mathbf{p} + \left[\mathbf{R}^{-1}\mathbf{p}\right]^T\mathbf{R}\left[\mathbf{R}^{-1}\mathbf{p}\right] \\ &= E\left\{d^2(n)\right\} - 2\mathbf{p}^T\mathbf{R}^{-1}\mathbf{p} + \mathbf{p}^T\mathbf{R}^{-1}\mathbf{p} \\ &= E\left\{d^2(n)\right\} - \mathbf{p}^T\mathbf{R}^{-1}\mathbf{p} \\ &= E\left\{d^2(n)\right\} - \mathbf{p}^T\mathbf{w}_o. \end{aligned} \tag{3.11}$$

In practical situations where adaptive equalizers are employed, an analytical description of the quadratic performance surface is not available. However, the location of points on that surface can be estimated by averaging the squared error over a period of time. Algorithms are applied to search the performance surface and locate optimal or near optimal solutions through iterative adjustments of the equalizer taps. One particular class of algorithms are based on the gradient search method employed by the method of steepest descent [83][24]. The method of steepest descent adjusts the equalizer tap weights in the direction of the negative gradient, which is not necessarily the direction of the minimum. The method of steepest descent is expressed by the following algorithm:

$$\mathbf{w}(n+1) = \mathbf{w}(n) + \mu\left(-\nabla_{\mathbf{w}}J^{\text{mse}}\right) \tag{3.12}$$

where μ is a constant step size. In order to converge, the step size is chosen to satisfy the following condition [83]:

$$0 < \mu < \frac{1}{\lambda_{\max}} \tag{3.13}$$

where $\lambda_{\max}$ is the maximum eigenvalue of the input correlation matrix $\mathbf{R}$. The minimal requirement for the method of steepest descent is a noisy estimate of the gradient, which hinders its application in real applications.

3.4 Trained Least-Mean-Squares Algorithm

The least-mean-squares algorithm (LMS) is a search method whose cost function simplifies the gradient calculation in (3.9) by replacing the expected values with instantaneous quantities. There are two modes in which the LMS operates: training and tracking. The training mode occurs only once during startup for point-to-point communications or periodically for broadcast applications and uses the difference between the equalizer output and the desired signal to adapt the equalizer tap coefficients. After initialization is complete, the LMS algorithm switches to the tracking mode, where the equalizer taps are adjusted based on the difference between the equalizer output and the estimated symbol (sliced output). The LMS algorithm will now be derived for both the real-valued [83][24] and complex-valued cases [82].

Real-Valued LMS

Recall from (3.9) that the gradient of J^{mse} is defined as

$$\nabla_{\mathbf{w}} J^{\text{mse}} = -2\mathbf{p} + 2\mathbf{R}\mathbf{w}.$$

An estimate of the gradient, $\nabla_{\mathbf{w}} \hat{J}^{\text{mse}}$, can be obtained by replacing $\mathbf{R}$ and $\mathbf{p}$ with their instantaneous estimates $\hat{\mathbf{R}} = \mathbf{x}(n)\mathbf{x}^T(n)$ and $\hat{\mathbf{p}} = d(n)\mathbf{x}(n)$, respectively. The gradient estimate $\nabla_{\mathbf{w}} \hat{J}^{\text{mse}}$ is equivalent to the LMS cost function which is defined as

$$\begin{aligned} \nabla_{\mathbf{w}} J^{\text{lms}} &= -2\hat{\mathbf{p}} + 2\hat{\mathbf{R}}\mathbf{w}(n) \\ &= -2\left(d(n)\mathbf{x}(n)\right) + 2\left(\mathbf{x}(n)\mathbf{x}^T(n)\right)\mathbf{w}(n) \\ &= -2\mathbf{x}(n)\underbrace{\left(d(n) - \mathbf{x}^T(n)\mathbf{w}(n)\right)}_{e(n)}. \end{aligned} \tag{3.14}$$

Substituting (3.14) into (3.12), the LMS equalizer tap adjustment algorithm for real-values signals is defined as

$$\mathbf{w}(n+1) = \mathbf{w}(n) + \mu\left(d(n) - y(n)\right)\mathbf{x}(n) \tag{3.15}$$

where μ is restricted to the limits defined in (3.13).

Complex-Valued LMS

The LMS cost function for the complex case is defined as

$$\begin{aligned} J^{\text{lms}} &= |e(n)|^2 = e(n)e^*(n) \\ &= \left(d(n) - \mathbf{w}^T(n)\mathbf{x}(n)\right)\left(d^*(n) - \mathbf{w}^H(n)\mathbf{x}^*(n)\right) \\ &= |d(n)|^2 - \mathbf{w}^H(n)\mathbf{x}^*(n)d(n) - \mathbf{w}^T(n)\mathbf{x}(n)d^*(k) + \mathbf{w}^T(n)\mathbf{x}(n)\mathbf{x}^H(n)\mathbf{w}^*(n) \\ &= |d(n)|^2 + \mathbf{w}_R^T(n)\left(\mathbf{x}^*(n)d(n) + \mathbf{x}(n)d^*(n)\right) - \mathbf{w}_I^T(n)\left(\mathbf{x}(n)d^*(n) - \mathbf{x}^*(n)d(n)\right) \\ &+ \mathbf{w}_R^T(n)\mathbf{x}(n)\mathbf{x}^H(n)\mathbf{w}_R(n) + \mathbf{w}_I^T(n)\mathbf{x}(n)\mathbf{x}^H(n)\mathbf{w}_I(n) \end{aligned} \tag{3.16}$$

where $(\cdot)^H$ is the complex conjugation and transposition operator, while $(\cdot)_R$ and $(\cdot)_I$ are the real and imaginary components of a complex number, respectively. Taking the gradient of (3.16) with respect to the real and imaginary components:

$$\nabla_{\mathbf{w}} J_R^{\text{lms}} = \frac{\partial J^{\text{lms}}}{\partial \mathbf{w}_R(n)} = 2\mathbf{x}(n)\mathbf{x}^H(n)\mathbf{w}_R(n)\} - \left(\mathbf{x}^*(n)d(n) + \mathbf{x}(n)d^*(n)\right) \tag{3.17}$$

$$\nabla_{\mathbf{w}} J_I^{\text{lms}} = \frac{\partial J^{\text{lms}}}{\partial \mathbf{w}_I(n)} = 2\mathbf{x}(n)\mathbf{x}^H(n)\mathbf{w}_I(n) - \jmath\left(\mathbf{x}(n)d^*(n) - \mathbf{x}^*(n)d(n)\right). \tag{3.18}$$

Using (3.17) and (3.18), the complex gradient of (3.16) is defined as

$$\begin{aligned} \nabla_{\mathbf{w}} J^{\text{lms}} &= \nabla_{\mathbf{w}} J_R^{\text{lms}} + \jmath\nabla_{\mathbf{w}} J_I^{\text{lms}} \\ &= 2\mathbf{x}(n)\mathbf{x}^H(n)\mathbf{w}_R(n) - \left(\mathbf{x}^*(n)d(n) + \mathbf{x}(n)d^*(n)\right) + \jmath 2\mathbf{x}(n)\mathbf{x}^H(n)\mathbf{w}_I(n) \\ &+ \left(\mathbf{x}(n)d^*(n) + \mathbf{x}^*(n)d(n)\right) \\ &= 2\mathbf{x}(n)\mathbf{x}^H(n)\underbrace{\left(\mathbf{w}_R(n) + \jmath\mathbf{w}_I(n)\right)}_{\mathbf{w}(n)} - 2\mathbf{x}^*(n)d(n) \\ &= 2\mathbf{x}(n)\mathbf{x}^H(n)\mathbf{w}(n) - 2d(n)\mathbf{x}^*(n) \\ &= -2\underbrace{\left(d(n) - \mathbf{w}^T(n)\mathbf{x}(n)\right)}_{e(n)}\mathbf{x}^*(n) \end{aligned} \tag{3.19}$$

The LMS equalizer tap adjustment algorithm is defined as

$$\begin{aligned} \mathbf{w}(n+1) &= \mathbf{w}(n) + \mu\left(-\nabla_{\mathbf{w}} J^{lms}\right) \\ &= \mathbf{w}(n) + \mu\left(d(n) - y(n)\right)\mathbf{x}^*(n) \end{aligned} \tag{3.20}$$

where once again μ is restricted to the limits defined in (3.13). This result, i.e. complex conjugation of the regressor $\mathbf{x}(n)$, applies to all algorithms that are of the stochastic gradient descent type

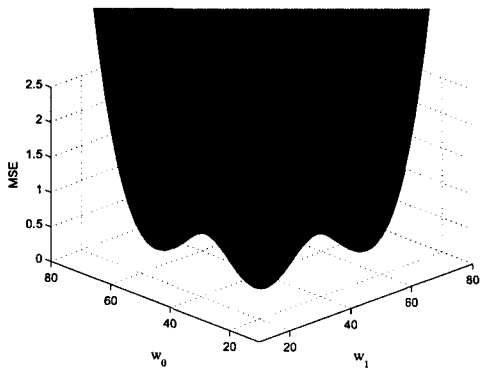


Figure 3.3: Multimodal constant modulus performance surface example.

regardless of the error signal definition. Therefore, the complex equalizer tap adjustment algorithm for the method of steepest descent type algorithms is defined as

$$\mathbf{w}(n+1) = \mathbf{w}(n) + \mu e(n)\mathbf{x}^*(n) \tag{3.21}$$

where $e(n)$ is the error signal of the particular algorithm.

3.5 Blind Equalization Algorithms

The first known work in the area of blind equalization was published by Sato in 1975 [59] for equalization of PAM signals. This initial work was generalized and patented for the 2D symbol case by Godard and Thirion [34]. In 1980, Godard introduced a new class non-convex cost functions that achieved equalization blindly and indepent of carrier-phase recovery [33]. In Godard's conjecture he stated:

> *"It should also be noted that the equalizer coefficients minimizing the dispersion functions closely approximate those which minimize the mean-squared error."*

This lead to the constant modulus algorithm (CMA), which was discovered independently by Godard [33] and Treichler *et al.* [77] and has since become the workhorse for blind equalization [54][76]. This is in part due to CMAs ability to converge prior to phase recovery. More recently, the multimodulus algorithm (MMA) was introduced by Yang *et al.* [88] which achieves low steady-state MSE and does not require phase recovery in steady-state operation. As illustrated by Fig. 3.3 for CMA, the performance surface of blind equalization algorithms is multimodal, which means that blind equalization algorithms may become trapped in local minima and converge to a false global minima.

This has the effect that different initializations may lead to convergence at different steady-state MSE levels. In practice baud-spaced blind equalizers are typically initialized with a single unitary center spike, while fractionally-spaced blind equalizers are initialized with a dual unitary center double spike [76][38].

3.5.1 Generalized Sato Algorithm

The cost function minimized by generalized Sato algorithm[1] (GSA) [59][34][68] is defined as

$$J^{\text{gsa}} = \frac{1}{2}E\left\{(y(n) - \gamma_S \text{csgn}(y(n)))^2\right\} \tag{3.22}$$

where $\text{csgn}(\cdot)$ is the complex sign operator and γ_S is a constant of the constellation. The complex sign operator is defined as

$$\text{csgn}(x) = \begin{cases} 1+j, & \{\Re\{x\} \geq 0, \quad \Im\{x\} \geq 0\} \\ 1-j, & \{\Re\{x\} \geq 0, \quad \Im\{x\} < 0\} \\ -1+j, & \{\Re\{x\} < 0, \quad \Im\{x\} \geq 0\} \\ -1-j, & \{\Re\{x\} < 0, \quad \Im\{x\} < 0\} \end{cases} \tag{3.23}$$

where $\Re\{\cdot\}$ and $\Im\{\cdot\}$ extract the real and imaginary components of a complex number, respectively. The constant γ_S is defined as

$$\gamma_S = \frac{E\left\{a^2(n) + b^2(n)\right\}}{E\left\{|a(n)| + |b(n)|\right\}} = \frac{E\left\{a^2(n)\right\}}{E\left\{|a(n)|\right\}}. \tag{3.24}$$

The equality to the right of (3.24) holds in the typical case where the symbols $a(n)$ and $b(n)$ have the same statistics. A stochastic gradient-descent equalizer adjustment algorithm that minimizes J^{gsa} is defined as

$$\begin{aligned} \mathbf{w}(n+1) &= \mathbf{w}(n) + \mu\left(-\nabla_{\mathbf{w}} J^{\text{gsa}}\right) \\ &= \mathbf{w}(n) + \mu \underbrace{\left(\gamma_S \text{csgn}(y(n)) - y(n)\right)}_{e^{\text{gsa}}(n)} \mathbf{x}^*(n) \\ &= \mathbf{w}(n) + \mu \underbrace{\left(\left(\gamma_S \text{sgn}(y_R(n)) - y_R(n)\right)\right.}_{e_R^{\text{gsa}}(n)} + \jmath \underbrace{\left.\left(\gamma_S \text{sgn}(y_I(n)) - y_I(n)\right)\right)}_{e_I^{\text{gsa}}(n)} \mathbf{x}^*(n) \end{aligned} \tag{3.25}$$

where $\text{sgn}(\cdot)$ is the real-valued sign operator and $e^{\text{gsa}}(n) = e_R^{\text{gsa}}(n) + \jmath e_I^{\text{gsa}}(n)$ is the GSA error signal.

[1] Also referred to as the reduced constellation algorithm (RCA) in literature.

3.5.2 Constant Modulus Algorithm

The constant modulus algorithm (CMA) [33][77][44] achieves channel equalization by penalizing the dispersion of the output modulus, $|y(n)|$, from the constant γ_C. The cost function minimized by CMA is defined as

$$J^{\text{cma}} = \frac{1}{pq} E\left\{(|y(n)|^p - \gamma_C^p)^q\right\} \tag{3.26}$$

where p and q are positive integers that are usually chosen to be either '1' or '2', resulting in four versions of the algorithm (denoted CMAp-q). The most typical case is CMA2-2 when $p = q = 2$, which will be considered exclusively throughout the remaining sections of this thesis. The dispersion constant γ_C^p is defined as

$$\gamma_C^p = \frac{E\left\{|s(n)|^{2p}\right\}}{E\left\{|s(n)|^p\right\}}. \tag{3.27}$$

A stochastic gradient-descent equalizer adjustment algorithm that minimizes J^{cma} is defined as

$$\begin{aligned}\mathbf{w}(n+1) &= \mathbf{w}(n) + \mu\left(-\nabla_{\mathbf{w}} J^{\text{cma}}\right) \\ &= \mathbf{w}(n) + \mu y(n)|y(n)|^{p-2}\left(\gamma_C^p - |y(n)|^p\right)^{q-1}\left(\text{sgn}(\gamma_C^p - |y(n)|^p)\right)^q \mathbf{x}^*(n).\end{aligned} \tag{3.28}$$

For the case where $p = q = 2$,

$$\begin{aligned}\mathbf{w}(n+1) &= \mathbf{w}(n) + \mu \underbrace{y(n)\left(\gamma_C^2 - |y(n)|^2\right)}_{e^{\text{cma}}(n)} \mathbf{x}^*(n) \\ &= \mathbf{w}(n) + \mu \underbrace{\left(y_R(n)\left(\gamma_C^2 - y_R^2(n) - y_I^2(n)\right)\right)}_{e_R^{\text{cma}}(n)} + \jmath \underbrace{y_I(n)\left(\gamma_C^2 - y_R^2(n) - y_I^2(n)\right)\big)}_{e_I^{\text{cma}}(n)} \mathbf{x}^*(n)\end{aligned} \tag{3.29}$$

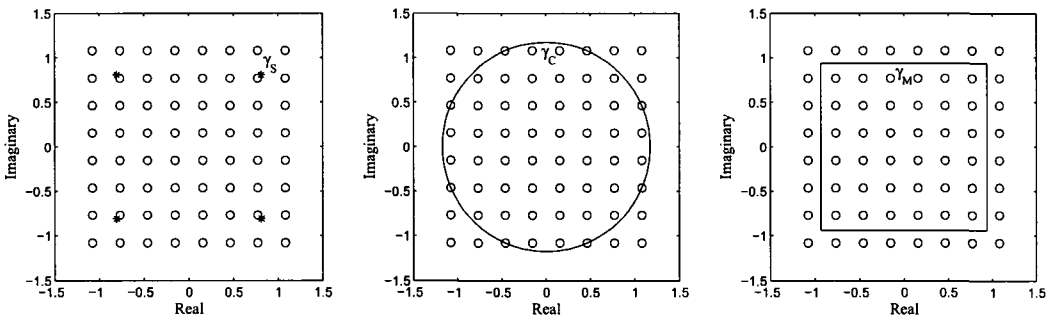
where $e^{\text{cma}}(n) = e_R^{\text{cma}}(n) + \jmath e_I^{\text{cma}}(n)$ is the CMA error signal. The CMA is an extention of Godard's algorithm [33], which has the same cost function as (3.26) except that 'q' is fixed at '2'.

3.5.3 Multimodulus Algorithm

The multimodulus algorithm (MMA) [87][88] separates the equalizer output into in-phase $y_R(n)$ and quadrature $y_I(n)$ components and penalizes the dispersion of each around separate straight contours. The cost function minimized by MMA is defined as

$$J^{\text{mma}} = \frac{1}{2p} E\left\{\left(y_R^p(n) - \gamma_M^p\right)^2\right\} + \jmath E\left\{\left(y_I^p(n) - \gamma_M^p\right)^2\right\} \tag{3.30}$$

where p is a positive integer. A value of $p = 2$ is typically chosen for implementation since it generally provides the best compromise between performance and implementation complexity [88]. This case



(a) GSA cost function. (b) CMA cost function. (c) MMA cost function.

Figure 3.4: Graphical interpretation of the GSA, CMA, and MMA cost functions for 64-QAM.

will be considered exclusively throughout the remaining sections of this thesis. The constant γ_M^p is defined as

$$\gamma_M^p = \frac{E\left\{a^{2p}(n) + b^{2p}(n)\right\}}{E\left\{|a(n)|^p + |b(n)|^p\right\}} = \frac{E\left\{a^{2p}(n)\right\}}{E\left\{|a(n)|^p\right\}}. \tag{3.31}$$

The equality to the right of (3.31) holds in the typical case where the symbols $a(n)$ and $b(n)$ have the same statistics. A stochastic gradient-descent equalizer adjustment algorithm that minimizes J^{mma} is defined as

$$\begin{aligned}\mathbf{w}(n+1) &= \mathbf{w}(n) + \mu\left(-\nabla_{\mathbf{w}} J^{\text{mma}}\right) \\ &= \mathbf{w}(n) + \mu\big(y_R(n)\left(\gamma_M^p - y_R^p(n)\right)|y_R(n)|^{p-2} + jy_I(n)\left(\gamma_M^p - y_I^p(n)\right)|y_R(n)|^{p-2}\big)\mathbf{x}^*(n).\end{aligned} \tag{3.32}$$

For the case where $p = 2$:

$$\mathbf{w}(n+1) = \mathbf{w}(n) + \mu\big(\underbrace{y_R(n)\left(\gamma_M^2 - y_R^2(n)\right)}_{e_R^{\text{mma}}(n)} + j\underbrace{y_I(n)\left(\gamma_M^2 - y_I^2(n)\right)}_{e_I^{\text{mma}}(n)}\big)\mathbf{x}^*(n) \tag{3.33}$$

where $e^{\text{mma}}(n) = e_R^{\text{mma}}(n) + je_I^{\text{mma}}(n)$ is the MMA error signal.

3.5.4 Decision-Directed Algorithm

The cost function that is minimized by the decision-directed (DD) algorithm [53], replaces symbol statistics with the instantaneous error across the slicer and is defined as

$$J^{\text{dd}} = \frac{1}{2}E\left\{\left(y(n) - \hat{s}(n)\right)^2\right\} \tag{3.34}$$

where $\hat{s}(n) = \hat{a}(n) + j \cdot \hat{b}(n)$ is the estimated QAM symbol. A stochastic gradient-descent equalizer adjustment algorithm that minimizes J^{dd} is defined as

$$\begin{aligned} \mathbf{w}(n+1) &= \mathbf{w}(n) + \mu \left(-\nabla_{\mathbf{w}} J^{\text{dd}}\right) \\ &= \mathbf{w}(n) + \mu \underbrace{(\hat{s}(n) - y(n))}_{e^{\text{dd}}(n)} \mathbf{x}^*(n) \\ &= \mathbf{w}(n) + \mu \Big(\underbrace{(\hat{a}(n) - y_R(n))}_{e_R^{\text{dd}}(n)} + \jmath \underbrace{\left(\hat{b}(n) - y_I(n)\right)}_{e_I^{\text{dd}}(n)} \Big) \mathbf{x}^*(n) \end{aligned} \tag{3.35}$$

where $e^{\text{dd}}(n) = e_R^{\text{dd}}(n) + \jmath e_I^{\text{dd}}(n)$ is the DD error signal. The DD algorithm requires the MSE to be lower than a specified threshold (refer to [17], [27]) and cannot be applied at the onset of equalization.

3.6 Hybrid Blind Equalization Algorithms

The so called "hybrid algorithms" of this section combine or augment existing cost functions to obtain enhanced performance. This may come in the form of increased stability, faster transient responses, lower steady-state MSE, improved transfer reliability to the DD mode or any combination of the previous. Hybrid algorithms tend to be more expensive in terms of the number of computations they require per iteration and consequently, they have higher implementation costs. Early contributions to hybrid methods include the work by Benveniste and Goursat [14], Picchi and Prati [53], and the dual-mode algorithms of Weerackody and Kassam [79].

3.6.1 Stop-and-Go Algorithm

The stop-and-go algorithm (SAG) [53] algorithm is based on a combination of the GSA and DD algorithms, which were discussed in sections 3.5.1 and 3.5.4, respectively. The adaptation of the tap coefficients for the SAG equalizer is stopped whenever the GSA and DD errors differ significantly. The SAG equalizer tap adjustment algorithm is defined as

$$\mathbf{w}(n+1) = \mathbf{w} + \mu \left(f_R(n) e_R^{\text{dd}}(n) + \jmath f_I(n) e_I^{\text{dd}}(n) \right) \mathbf{x}^*(n) \tag{3.36}$$

where $e_R^{\text{dd}}(n)$ and $e_I^{\text{dd}}(n)$ are the real and imaginary components of the DD error signal, respectively, which were defined in (3.35), while $f_R(n)$ and $f_I(n)$ are the real and imaginary "stop-and-go" flags,

respectively. The "stop-and-go" flags are computed as follows:

$$f_R(n) = \begin{cases} 1, & \text{sgn}\left(e_R^{\text{dd}}(n)\right) = \text{sgn}\left(e_R^{\text{gsa}}(n)\right) \\ 0, & \text{sgn}\left(e_R^{\text{dd}}(n)\right) \neq \text{sgn}\left(e_R^{\text{gsa}}(n)\right) \end{cases} \tag{3.37}$$

$$f_I(n) = \begin{cases} 1, & \text{sgn}\left(e_I^{\text{dd}}(n)\right) = \text{sgn}\left(e_I^{\text{gsa}}(n)\right) \\ 0, & \text{sgn}\left(e_I^{\text{dd}}(n)\right) \neq \text{sgn}\left(e_I^{\text{gsa}}(n)\right) \end{cases} \tag{3.38}$$

where $e_R^{\text{gsa}}(n)$ and $e_I^{\text{gsa}}(n)$ are the real and imaginary components of the GSA error signal, respectively, which were defined in (3.25).

A modified SAG algorithm is proposed in [66] that improves the convergence time by updating the equalizer coefficients in "stop" regions. The idea is that if the sign of the error is reversed in "stop" regions and the coefficients are updated in a controlled fashion, faster convergence can be achieved. This is accomplished by replacing the zeros in the flag terms defined in (3.37) and (3.38) with a ratio $\rho(n)$, where $-1 < \rho(n) < 1$. The value of $\rho(n)$ can be chosen to linearly increase from an initial value $-\beta$ to a final value $+\beta$ by a constant amount during each consecutive iteration.

3.6.2 Benveniste-Goursat Algorithm

The Benveniste-Goursat algorithm (BGA) [14] is a combination of the DD and GSA error signals. The BGA equalizer tap adjustment algorithm is defined as

$$\mathbf{w}(n+1) = \mathbf{w}(n) + \mu\left(k_1 e^{\text{dd}}(n) + k_2 \left|e^{\text{dd}}(n)\right| e^{\text{gsa}}(n)\right)\mathbf{x}^*(n) \tag{3.39}$$

where $e^{\text{dd}}(n)$ and $e^{\text{gsa}}(n)$ were defined in (3.35) and (3.25), respectively, while k_1 and k_2 are positive constants that are chosen in an ad hoc manor. When the equalizer is far from convergence, the $E\left\{\left|e^{\text{dd}}(n)\right|\right\}$ is large which allows the tap update in (3.39) to be dominated by GSA error. Conversely, near convergence, $E\left\{\left|e^{\text{dd}}(n)\right|\right\}$ tends toward zero. This allows the tap update to be dominated by DD error, which reduces the excess MSE due to the GSA error term. This presents a tradeoff between convergence and steady-state MSE, which is controlled by the selection of k_1 and k_2. In [68], selecting $k_1 = k_2 = 1$ provided satisfactory performance.

3.6.3 Modified-CMA

The modified-CMA (MCMA) [39][41] augments the CMA cost function with constellation matched error (CME) term. This term provides MCMA with knowledge of the constellation, which improves the convergence time and steady-state MSE. The MCMA equalizer tap adjustment algorithm is

defined as

$$\mathbf{w}(n+1) = \mathbf{w}(n) + \mu\big(\underbrace{y(n)\left(\gamma_C^2 - |y(n)|^2\right)}_{e^{\text{cma}}(n)} + \beta\eta(n)\big) \tag{3.40}$$

where $\eta(n)$ is the CME error component of MCMA and β is a weighting factor that trades off the amplitude and constellation matched errors. The CME error signal is defined as

$$\eta(n) = -\frac{d}{dx}g(x)\Big|_{x=y_R(n)} - j\frac{d}{dx}g(x)\Big|_{x=y_I(n)} \tag{3.41}$$

where $g(x)$ is the CME function. This function is to be constructed from the equalizer output such that it satisfies a number of criteria, which include uniformity, symmetry, and zero/maximum penalties at the zero/maximum deviations from symbol points. Candidates for this function are proposed in [10] and [40], the latter being an even sinusoidal function. The following CME function for QAM constellations is zero at symbol points [41]:

$$g(x) = 1 - \sin^{2q}\left(\frac{x\pi}{d}\right) \tag{3.42}$$

where q is a positive integer and d is the distance between constellation points. This function reaches its maximum value of one at the center point between two consecutive symbols and is zero at symbol points. As the selection of q increases, the slope of $g(x)$ near symbol points becomes sharper.

The MCMA error can be modified to weigh the CMA and CME terms based on a data-dependent weighting factor. This version of MCMA, known as data-dependent MCMA, continuously trades off the CMA and CME terms in the MCMA equalizer tap adjustment over time depending on the adaptation phase and channel conditions. This enables data-dependent MCMA to achieve a faster convergence time than standard MCMA. However, the multiplications per each weight update are approximately five to six times that of fixed MCMA [41].

3.6.4 CMA-MMA Algorithm

The CMA-MMA algorithm [88] jointly uses the two cost functions of CMA and MMA for the real and imaginary signal components, respectively. This introduces asymmetry into the tap updating algorithm of the equalizer. The CMA-MMA equalizer tap adjustment algorithm is defined as

$$\mathbf{w}(n+1) = \mathbf{w}(n) + \mu\big(\underbrace{y_R(n)\left(\gamma_C^2 - |y(n)|^2\right)}_{e_R^{\text{cma}}(n)} + \jmath\underbrace{y_I(n)\left(\gamma_M^2 - y_I^2(n)\right)}_{e_I^{\text{mma}}(n)}\big) \tag{3.43}$$

where $e_R^{\text{cma}}(n)$ is the real component of the CMA error signal defined in (3.29), while $e_I^{\text{mma}}(n)$ is the imaginary component of the MMA error signal defined in (3.33). The CMA error component of CMA-MMA increases the convergence reliability, while the MMA error component of CMA-MMA rotates the constellation in steady-state operation.

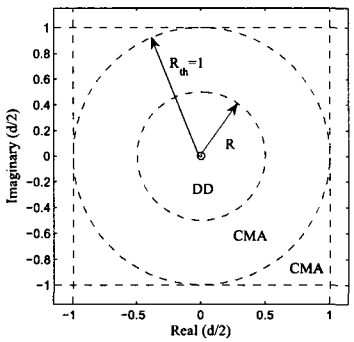


Figure 3.5: BCMA decision regions.

3.6.5 Blended-CMA

In traditional CMA-DD equalizer schemes, CMA is applied at the onset of equalization to reduce the amplitude distortion. After convergence, the equalizer is switched to the DD mode to reduce the steady-state MSE. The difficulty that arises with this scheme is determining at what point convergence occurs and whether the steady-state MSE after convergence is low enough that the equalizer can be switched to the DD mode. The blended-CMA (BCMA) [49] address this by introducing an adaptive radius to select between CMA and DD equalizer tap updates, providing a reliable method to switch between the respective modes. The choice of adaptation with CMA or DD error is derived from thresholding the DD error term. Large values of $|\hat{s}(n) - y(n)|$ correspond to regions that are far away from symbol points, while small values of $|\hat{s}(n) - y(n)|$ correspond to regions near symbol points.

The BCMA process is illustrated in Fig. 3.5. The radius, R, is initially zero and the equalizer is adapted with CMA. After processing k symbols, the number of symbols that are within the initial threshold distance from their estimates are assumed to an integer *sum*. Once *sum* is greater than the initial threshold, the radius begins to adapt, allowing the equalizer to be updated with DD error when $0 < |\hat{s}(n) - y(n)| < R$. A set of thresholds is stored in the vector $\mathbf{T}$, where each element T_i corresponds to an element in the radius vector $\mathbf{R}$.

3.6.6 Concurrent Algorithm

An alternative method to the CMA to DD mode transfer problem discussed in section 3.6.5 is provided by the concurrent algorithm (CCA) [17], which uses CMA and DD equalizers in parallel to avoid transfer failure. A flag $f(n)$ is added to the DD equalizer cost function which determines

whether it adapts. Carrier phase is jointly estimated along with equalization. The equalizer output for CCA is the sum of the individual CMA and DD equalizer outputs and is defined as

$$y(n) = \mathbf{w}_{\text{cma}}^T(n)\mathbf{x}(n) + \mathbf{w}_{\text{dd}}^T(n)\mathbf{x}(n) \tag{3.44}$$

where $\mathbf{w}_{\text{cma}}(n)$ and $\mathbf{w}_{\text{dd}}(n)$ are the equalizer tap coefficient vectors of the CMA and DD equalizers, respectively. The equalizer tap adjustment algorithms for the CMA and DD equalizers which constitute the CCA equalizer are defined as

$$\mathbf{w}_{\text{cma}}(n+1) = \mathbf{w}_{\text{cma}}(n) + \mu y(n)\left(\gamma_C^2 - |y(n)|^2\right)\mathbf{x}^*(n) \tag{3.45}$$

$$\mathbf{w}_{\text{dd}}(n+1) = \mathbf{w}_{\text{dd}}(n) + \mu f(n)\left(\hat{s}(n) - y(n)\right)\mathbf{x}^*(n) \tag{3.46}$$

where the CMA equalizer update is equivalent to that defined in (3.29), while the DD equalizer update differs from (3.35) by the inclusion of the $f(n)$ update flag. This flag is determined using symbol estimates and is defined as

$$f(n) = \begin{cases} 1 & \hat{s}_n = \hat{s}_n^{\wp} \\ 0 & \hat{s}_n \neq \hat{s}_n^{\wp} \end{cases} \tag{3.47}$$

where $\hat{s}_n^{\wp}$ is the estimated symbol of the perturbed output $\tilde{y}(n)$. The perturbed output is defined as

$$\begin{aligned} \tilde{y}(n) &= \left(\mathbf{w}_{\text{cma}}(n) + \Delta\mathbf{w}_{\text{cma}}(n)\right)^T \mathbf{x}(n) + \mathbf{w}_{\text{dd}}^T(n)\mathbf{x}(n) \\ &= \mathbf{w}_{\text{cma}}^T(n+1)\mathbf{x}(n) + \mathbf{w}_{\text{dd}}^T(n)\mathbf{x}(n) \end{aligned} \tag{3.48}$$

where $\Delta\mathbf{w}_{\text{cma}}(n)$ is the perturbation $\wp$ that is defined as $\Delta\mathbf{w}_{\text{cma}}(n) = \mathbf{w}_{\text{cma}}(n+1) - \mathbf{w}_{\text{cma}}(n)$. This requires the CMA equalizer to be updated prior to the DD equalizer.

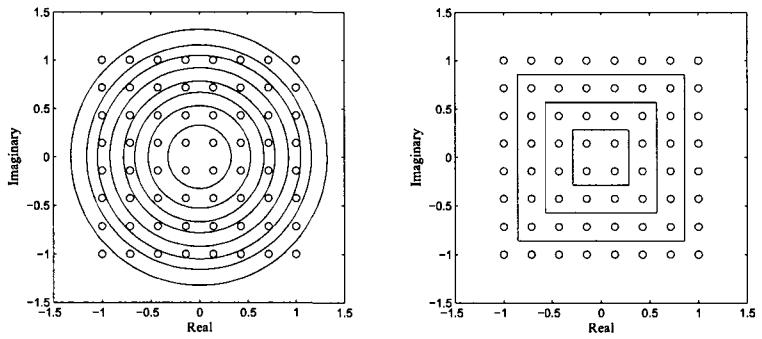
A modified concurrent algorithm is proposed in [18] and [19] that replaces the DD equalizer with a soft decision directed (SDD) equalizer. The SDD equalizer maximizes the *a posteriori* probability density function (p.d.f.) of the equalizer output. The equalizer tap adjustment algorithm for the SDD is defined as

$$\mathbf{w}_{\text{sdd}}(n+1) = \mathbf{w}_{\text{sdd}}(n) + \mu\nabla_{\mathbf{w}} J^{\text{lmap}} \tag{3.49}$$

where the SDD cost function is $J^{\text{lmap}} = \rho\log(\hat{p}(\mathbf{w}, y(n)))$ and ρ is a constant. The local *a posteriori* p.d.f. is defined as

$$\hat{p}\left(\mathbf{w}, y(n)\right) = \sum_{p=2i-1}^{2i}\sum_{q=2l-1}^{2l} \frac{1}{8\pi\rho}\exp\left(-\frac{|y(n) - s_{pq}|^2}{2\rho}\right) \tag{3.50}$$

where $s_{i,j}$ forms a cluster of four constellation points.



(a) Circular boundaries for CADAMA. (b) Square boundaries for MAMA.

Figure 3.6: Region boundaries for dual-model algorithms for 64-QAM.

3.6.7 CMA-Assisted Decision Adjusted Modulus Algorithm

The CMA-assisted decision adjusted modulus algorithm (CADAMA) [6] is a dual-mode algorithm that employs CMA for the first mode and the decision adjusted modulus algorithm (DAMA) [57][62] for the second mode. The latter mode divides the QAM constellation into several circular boundaries as illustrated in Fig. 3.6(a). The DAMA is a phase blind algorithm that is based on a cost function that goes to zero at each ring of a QAM constellation. The equalizer tap adjustment algorithm for DAMA is defined as [57]:

$$\mathbf{w}(n+1) = \mathbf{w}(n) + \mu \min_{i} \left(y(n) \left(\gamma_{C,i}^2 - |y(n)|^2 \right) \right) \tag{3.51}$$

where $\{\gamma_{C,i}^2\}_{i=1}^{p}$ are the signal point radii of the QAM constellation, while p is a positive integer. CADAMA begins operation in the CMA mode and switches to the DAMA mode after the amplitude distortion has been sufficiently reduced. The criterion for switching modes is based on comparing the histogram of radius decisions over an observation interval [6].

A similar dual-mode algorithm, the MMA-assisted modulus algorithm (MAMA), is proposed for MMA in [63]. The MAMA employs CMA for the first mode and MMA with multiple radii for the second mode. The latter mode divides the QAM constellation into several squares boundaries as illustrated in Fig. 3.6(b). Signal point radii are superimposed over the symbol points between successive boundaries, which reduces the error for MMA when $y(n) \in \{s_m\}_{m=1}^{M}$ for an M-QAM constellation. Within the region created by two consecutive square radii, there will be no error for corner symbol points, no error for the real component of symbol points in vertical columns, and no error for the imaginary components of symbol points in horizontal rows. The equalizer tap update

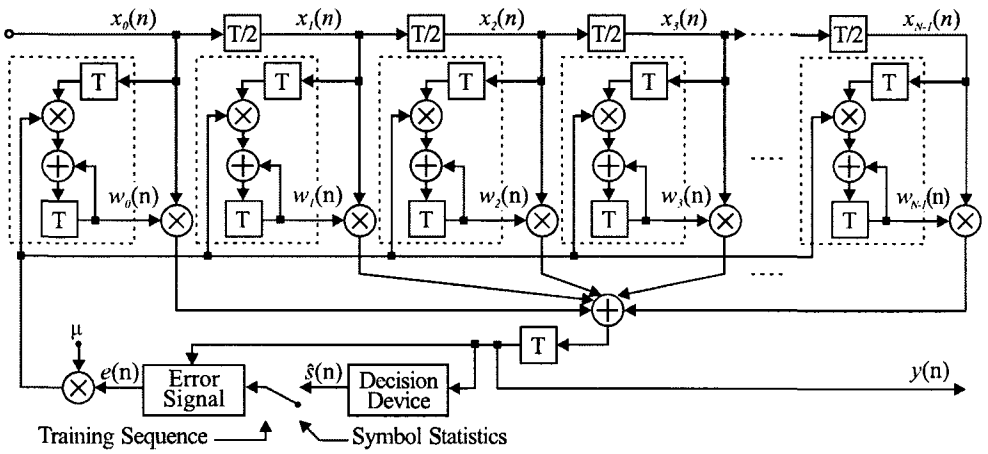


Figure 3.7: Adaptive linear equalizer structure (tap update portions indicated by dashed-boxes).

algorithm for the second mode of MAMA is defined as

$$\mathbf{w}(n+1) = \mathbf{w}(n) + \mu \left(\min_i \left(y_R(n) \left(\gamma_{M,i}^2 - y_R^2(n) \right) \right) + \jmath \min_i \left(y_I(n) \left(\gamma_{M,i}^2 - y_I^2(n) \right) \right) \right) \tag{3.52}$$

where $\{\gamma_{M,i}^2\}_{i=1}^{p}$ are the signal point radii of the QAM constellation, while p is a positive integer. MAMA begins operation in the CMA mode and switches to MMA with mulple radii after the amplitude distortion has been sufficiently equalized.

3.7 Computationally-Efficient Methods for Equalization

This section discusses computationally-efficient methods that apply to both blind and non-blind adaptive equalizers. As illustrated in Fig 3.7, adaptive equalization can be generalized into two operations: convolving the received symbol sequence with the filter tap coefficients and updating the filter tap coefficients. One method to improve computational efficiency is to simplify or reduce the number of multiplications needed to realize the filter. Signed-error [42][15] and power-of-two error [5][28] are methods which simplify multiplications in the equalizer tap adjustment portion to shift and add operations. The partial update method [36][35] reduces the number of multiplications by updating only a subset of the total taps during an iteration, while frequency-domain block algorithms [12][67] update the equalizer taps once every k iterations.

3.7.1 Signed-Error Method

Algorithms that employ the signed-error method only retain the sign of the error signal [42][15][61]. The general equalizer tap adjustment algorithm for signed-error algorithms is defined as

$$\mathbf{w}(n+1) = \mathbf{w}(n) + \mu \text{csgn}(e(n))\mathbf{x}^*(n) \tag{3.53}$$

where csgn$(\cdot)$ was defined in (3.23) and $e(n)$ is the error signal of the respective algorithm.

As opposed to signed-error versions of GSA (SE-GSA) and MMA (SE-MMA), signed-error CMA (SE-CMA) [15] does not retain the convergence characteristics of CMA and is prone to divergence. This drawback can be overcome by the inclusion of a controlled noise or dither signal, which improves the overall robustness. The equalizer tap adjustment algorithm for dithered SE-CMA (DSE-CMA) [61] is defined as

$$\mathbf{w}(n+1) = \mathbf{w}(n) + \mu\alpha_d \text{csgn}\left(e^{\text{cma}}(n) + \alpha_d d(n)\right)\mathbf{x}^*(n) \tag{3.54}$$

where α_d is a positive constant and $d(n)$ is an i.i.d. dithering process uniformly distributed over (-1,1]. Selecting $\alpha_d > 2(\gamma_c^2/3)^{3/2}$ is necessary to ensure that the expected update for DSE-CMA is identical to that for CMA within a certain bound [61].

The computational requirements for the signed-error equalizer tap update method are specified in Table 3.1. When coupled with a power-of-two step size, the multiplications are reduced to shift and add operations.

3.7.2 Power-of-Two Error Method

In the power-of-two error method [5][28], the error signal of the respective algorithm is quantized using a nonlinear power-of-two quantizer. The general equalizer tap adjustment algorithm for power-of-two error algorithms is defined as

$$\mathbf{w}(n+1) = \mathbf{w}(n) + \mu Q_2\{e(n)\}\mathbf{x}^*(n) \tag{3.55}$$

where $e(n)$ is the error signal of the respective algorithm and $Q_2\{\cdot\}$ is a nonlinear power-of-two quantizing operator, which can be defined as [28]:

$$Q_2\{x\} = \begin{cases} \text{csgn}(x), & |x| \geq 1 \\ 2^{\lfloor \log_2|x| \rfloor}\text{sgn}(x), & 2^{-WL+2} \leq |x| < 1 \\ \tau\text{csgn}(x), & |x| < 2^{-WL+2} \end{cases} \tag{3.56}$$

where WL is the data word length including the sign bit and τ is usually set to 0 or 2^{-WL+1}.

The computational requirements for the power-of-two error equalizer tap update method are specified in Table 3.1. When coupled with a power-of-two step size, the multiplications are reduced to shift and add operations.

3.7.3 Partial Coefficient Update Method

The computational complexity of an adaptive filter is proportional to the number of taps required. When considering channel equalization, the number of taps can range from several tens to several hundreds. Partially updating the tap coefficients enables efficient use of processor capacity [4][36][35][81] and less power consumption. The general equalizer tap adjustment algorithm for partial coefficient update algorithms is defined as

$$\mathbf{w}(n+1) = \mathbf{w}(n) + \mu e(n)\mathbf{A}_{\mathbf{I}_P(n)}\mathbf{x}^*(n) \tag{3.57}$$

where $e(n)$ is the respective error signal and $\mathbf{A}_{\mathbf{I}_P(n)}$ is a diagonal matrix having P elements equal to one in the positions indicated by $\mathbf{I}_P(n)$ and zeros elsewhere, where $\mathbf{I}_P(n)$ is the $N \times 1$ update constraint vector. The update constraint vector is determined through information evaluation, which can be accomplished using a number of methods. We will consider the fixed and time-varying set-membership criteria discussed in [81]. The first method is when P tap coefficients are updated during each iteration, where P is a fixed value between $0 < P < N$. The equalizer input vector $\mathbf{x}(n)$ is sorted and the index positions that correspond to the largest P input samples are set to one in $\mathbf{I}_P(n)$, while all other positions are set to zero. There is no restriction on the selection of P as long as the stability or convergence is not compromised.

An alternate method is to let P vary with time, such that $P_{\min} \leq P(n) \leq P_{\max}$. Initially, $P(n) = P_{\min}$ and is incremented by 1 until $P(n) = P_{\max}$ or the regressor power meets the following condition:

$$\|\mathbf{A}_{\mathbf{I}_P(n)}\mathbf{x}(n)\|_2^2 \geq \alpha_p \|\mathbf{x}(k)\|_2^2 \tag{3.58}$$

where $\|\mathbf{x}\|_2 = \sqrt{\sum_i |x_i|^2}$ is the two norm and α_p is a fixed constant that ranges from $0 < \alpha_p < 1$.

The computational requirements for the fixed and variable partial coefficient update methods are specified in Table 3.1. These figures do not include the overhead processing for the fixed and time-varying cases. One possible implementation of the equalizer input power calculation is to immediately square the input sample and apply the result to a separate tapped delay line. This would cost one additional multiplication and an accumulation. However, it would double the amount of storage elements.

Table 3.1: Arithmetic operations for the equalizer tap adjustment of a single FIR filter ($\mu = 2^{-n}$).

Method	Multiplications	Additions	Barrel Shifts	FFTs / IFFTs	Update Interval
None	N	N	1	0	1
Signed Error	0	N	N	0	1
Power-of-two Error	0	N	$N+1$	0	1
Block (Frequency Domain)	$2N$	$2N$	$2N$	3	N
Partial Update (Fixed)	P	P	1	0	1
Partial Update (Variable)	$P(n)$	$P(n)$	1	0	1

3.7.4 Block Method

Algorithms that employ the block method use a block of equalizer input samples and instantaneous error samples to update the equalizer tap coefficients once every B input samples, where B corresponds to the block length. This produces a more accurate estimation of the gradient, which allows a larger step size to be applied [20]. Block algorithms can be implemented in both the time [20] or frequency-domains [12][67][69]. While the time-domain realization does not reduce the number of computations, significant reductions can be obtained with their frequency-domain realization. This is a consequence of performing time-domain convolution in the frequency-domain and the efficiency of the fast Fourier transform (FFT) and the inverse fast Fourier transform (IFFT) algorithms.

Time-Domain Implementation

The general time-domain equalizer tap adjustment for block algorithms is defined as

$$\mathbf{w}(nB+B) = \mathbf{w}(nB) + \mu \sum_{i=0}^{B-1} e(nB+i) \cdot \mathbf{x}^*(nB+i) \tag{3.59}$$

where $e(n)$ is the error signal of the respective algorithm.

The recursion in (3.59) represents a single equalizer tap adjustment from time nB to $nB+B$ based on the B partial update accumulations. The selection of the block size B is a design parameter that determines the convergence characteristics of the algorithm.

Frequency-Domain Implementation

Block algorithms can be realized in the frequency-domain using the overlap-save or overlap-add sectioning methods. The block size is set to $B = N$ since this is the most efficient value for the FFT algorithms [21], which corresponds to $2N$ frequency-domain equalizer taps. The general frequency-domain equalizer tap adjustment algorithm for block algorithms is defined as [67][69]:

$$\mathbf{W}(nN+N) = \mathbf{W}(nN) + \mathcal{F}\mathbf{g}\mathcal{F}^{-1}\mu\mathbf{X}^H(nN)\mathbf{E}(nN) \tag{3.60}$$

where here uppercase letters denote frequency-domain quantities, $\mathbf{g}$ is the $N \times N$ gradient constraint matrix, and $\mathcal{F}$ and $\mathcal{F}^{-1}$ are the FFT and the IFFT, respectively. The input signal matrix $\mathbf{X}(nN)$ is comprised of two blocks of N input samples and is defined as

$$\mathbf{X}(nN) = \text{diag}\left\{\mathcal{F}\left[x(nN-N),\ldots,x(nN-1),x(nN),\ldots,x(nN+N-1)\right]^T\right\} \tag{3.61}$$

where the diag $\{\cdot\}$ operator transforms the $2N \times 1$ vector into a $2N \times 2N$ diagonal matrix. The frequency-domain equalizer output is $\mathbf{Y}(nN) = \mathbf{X}(nN)\mathbf{W}(nN)$, while the time-domain equalizer output is defined as

$$y(nN) = \mathbf{k}\mathcal{F}^{-1}\mathbf{X}(nN)\mathbf{W}(nN) \tag{3.62}$$

where $\mathbf{k}$ is the $N \times 2N$ constraint matrix that ensures the output result is a linear convolution. The frequency-domain error signal vector is defined as

$$\mathbf{E}(nN) = \mathcal{F}[\mathbf{0}^T, \mathbf{e}^T(nN)]^T \tag{3.63}$$

where $\mathbf{e}(nN) = [e(nN-N),\ldots,e(nN-1)]^T$ is the time-domain error signal vector and $\mathbf{0}$ is the $N \times 1$ zero vector. The gradient and convolution constraint matrixes, $\mathbf{g}$ and $\mathbf{k}$, respectively, are defined as

$$\mathbf{g} = \begin{bmatrix} \mathbf{I} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} \end{bmatrix} \qquad \mathbf{k} = \begin{bmatrix} \mathbf{0} & \mathbf{I} \end{bmatrix} \tag{3.64}$$

where here $\mathbf{0}$ is an $N \times N$ zero matrix and $\mathbf{I}$ is an $N \times N$ identity matrix.

The computational requirements for the block equalizer tap update method are specified in Table 3.1. These figures do not include computations needed for the equalizer output or time-domain error signal calculations, which require an additional 2 FFTs/IFFTs.

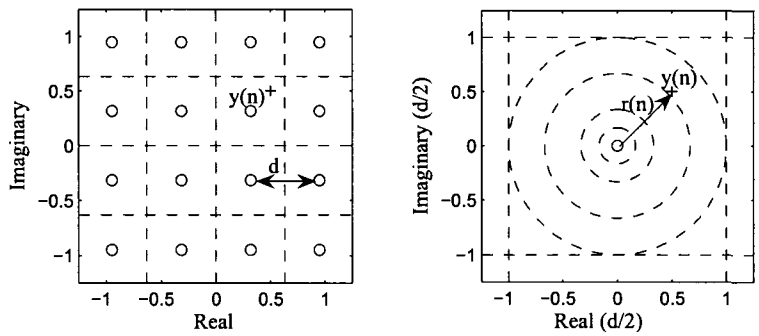
Chapter 4

The Radius-Adjusted Approach for Blind Equalization of QAM Signals

4.1 Introduction

This chapter introduces a novel radius-adjusted approach for blind equalization of QAM signals. The radius-adjusted approach is hybrid method to achieve equalizer tap adaptation based on the equalizer output radius. This approach superimposes static circular contours over an estimated symbol point, which create regions that correspond to a fixed step size and weighting factor. The equalizer tap adjustment consists of a linearly weighted sum of adaptation criteria that is scaled by a variable step size. Two new radius-adjusted equalization algorithms are proposed and analyzed: radius-adjusted modified multimodulus algorithm (RMMA) and radius-adjusted multimodulus decision-directed algorithm (RMDA). The radius-adjust approach is extendend to a computationally-efficient method termed the "selective update method", which uses the equalizer output radius to selectively adapt the equalizer tap coefficients.

This chapter begins with an introduction to the radius-adjusted approach and continues with the derivation of RMMA and RMDA. A method to tune the radius-adjusted algorithms is developed based on radius statistics, while their steady-state performance is analyzed based on a fundamental energy-preserving relation, which was first exploited by Sayed and Rupp in [60]. Lastly, the selective update method is introduced as a general computationally-efficient method for adaptive equalization,



(a) Square regions for 16-QAM. (b) Centered around symbol estimate.

Figure 4.1: Sample decision regions for symbol estimates in the radius-adjusted approach.

while a modified version is proposed to reduce the hardware requirements for implementation.

4.2 Radius-Adjusted Concept

The radius-adjusted approach is a method to achieve equalizer tap adaptation based on the equalizer output radius for QAM signals. Static circular contours are defined around an estimated symbol point in a QAM constellation, which create regions that can be mapped to adaptation phases. A region corresponds to a fixed step size, $\mu(n)$, and weighting factor, $\lambda(n)$, that can be used to create a time-varying equalizer tap update based on the equalizer output radius, $r(n)$, which is defined as

$$r(n) = \left|\hat{s}(n) - y(n)\right| \tag{4.1}$$

where $r(n)$ is the distance between the equalizer output, $y(n)$, and its corresponding symbol estimate, $\hat{s}(n)$. The equalizer tap update consists of a linearly weighted sum of adaptation criteria that is scaled by a variable step size.

The general concept of the radius-adjusted approach is illustrated for 16-QAM in Fig. 4.1. The equalizer output of Fig. 4.1(a) corresponds to the $r(n)$ of Fig. 4.1(b), where sample circular decision regions are superimposed over the original square decision region. The outer regions of Fig. 4.1(b) generally correspond to adaptation phases with high MSE, while the inner regions correspond to adaption phases with low MSE. The accuracy of the equalizer tap adjustment can be improved by grouping regions into adaptation phases and adjusting the parameters $\mu(n)$ and $\lambda(n)$ based on the characteristics of that phase. The convergence time can be reduced by applying large $\mu(n)$ in the outer regions, while the misadjustment can be reduced by applying small $\mu(n)$ in the inner regions.

Stability and transfer reliability can be improved by using $\lambda(n)$ to trade off error terms within the error signal. Algorithms that require an initial equalization period can modify their error signal by including an error term of an algorithm suitable for the initial equalization period. In the modified error signal, $\lambda(n)$ would select the error term for initial equalization only if the equalizer output lies in the outer regions. This method, which can be applied to dual-mode algorithms, allows for automatic transfer between error modes.

4.3 Radius-Adjusted Blind Equalization Algorithms

This section introduces the radius-adjusted blind equalization algorithms RMMA and RMDA, presents a tuning method for the algorithms based on radius statistics, and analyzes their steady-state performance.

4.3.1 Radius-Adjusted Modified-Multimodulus Algorithm

The radius-adjusted modified-multimodulus algorithm (RMMA) augments the MMA cost function in (3.30) with a CME term and linearly weighs the respective terms based on the radius-adjusted approach. The objective is to decrease the convergence time, while obtaining low steady-state MSE and misadjustment. The cost function that is minimized by RMMA is defined as

$$J^{\text{rmma}} = \frac{1}{2p}\lambda(n)E\left\{\left(y_R^p(n) - \gamma_M^p\right)^2 + \jmath\left(y_I^p(n) - \gamma_M^p\right)^2\right\} + (1-\lambda(n))\,E\left\{\beta g\left(y(n)\right)\right\} \tag{4.2}$$

where p is a positive integer, β is a weighting factor that trades off the amplitude and constellation-matched errors and $g\left(y(n)\right)$ is the CME function. A gradient-descent equalizer adjustment algorithm that minimizes J^{rmma} is defined as

$$\begin{aligned}\mathbf{w}(n+1) &= \mathbf{w}(n) + \mu(n)\left(-\nabla_{\mathbf{w}} J^{\text{rmma}}\right) \\ &= \mathbf{w}(n) + \mu(n)\Big(\overbrace{\left(\lambda(n)y_R(n)\left|y_R(n)\right|^{p-2}\left(\gamma_M^p - y_R^p(n)\right) + (1-\lambda(n))\beta\eta_R(n)\right)}^{e_R^{\text{rmma}}(n)} \\ &\quad + \jmath\underbrace{\left(\lambda(n)y_I(n)\left|y_I(n)\right|^{p-2}\left(\gamma_M^p - y_I^p(n)\right) + (1-\lambda(n))\,\beta\eta_I(n)\right)}_{e_I^{\text{rmma}}(n)}\Big)\mathbf{x}^*(n)\end{aligned} \tag{4.3}$$

where $\eta(n) = \eta_R(n) + \jmath\eta_I(n)$ is the CME error signal and $e^{\text{rmma}}(n) = e_R^{\text{rmma}}(n) + \jmath e_I^{\text{rmma}}(n)$ is the RMMA error signal, which is reduced to MMA error when $\lambda(n) = 1$ and to CME error when $\lambda(n) = 0$. The CME error signal is obtained by taking the negative gradient of the CME function

with respect to the equalizer tap coefficients as follows

$$\eta(n) = \underbrace{-\frac{d}{dx}g(x)\Big|_{x=y_R(n)}}_{\eta_R(n)} \underbrace{-\jmath\frac{d}{dx}g(x)\Big|_{x=y_I(n)}}_{\eta_I(n)}. \tag{4.4}$$

The following class of functions, which was defined for MCMA in (3.42), take a zero value at symbol points:

$$g(x) = 1 - \sin^{2q}\left(\frac{x\pi}{d}\right)$$

where q is a positive integer and d is the distance between symbol points. This function reaches its maximum value of one at the center point between two adjacent symbols and is zero at symbol points. As the selection of q increases, the slope of $g(x)$ near symbol points becomes sharper. The selection of the weighting factor β for RMMA is chosen similar to that in MCMA. However, with respect to the MMA error as follows

$$\beta\left|\eta(n)\right|_{\max} < \max_{a(n),b(n)\in s(n)} \left|a(n)(\gamma^2 - |a(n)|^2) + \jmath b(n)(\gamma^2 - |b(n)|^2)\right|. \tag{4.5}$$

The typical values of p and q for implementation are '2' and '1', respectively, since they provides the best compromise between performance and implementation complexity. This case will be considered exclusively throughout the remaining sections of this thesis. With this consideration, the CME function $g(x)$ can be rewritten as

$$g(x) = \frac{1}{2}\left(1 + \cos\left(2\pi\frac{x}{d}\right)\right) \tag{4.6}$$

which leads to

$$\eta(n) = \frac{\pi}{d}\left(\sin\left(2\pi\frac{y_R(n)}{d}\right) + \jmath\sin\left(2\pi\frac{y_I(n)}{d}\right)\right). \tag{4.7}$$

The RMMA error signal can then be rewritten as

$$\begin{aligned} e^{\text{rmma}}(n) = &\left(\lambda(n)y_R(n)\left(\gamma_M^2 - y_R^2(n)\right) + (1-\lambda(n))\,\beta\left(\frac{\pi}{d}\right)\sin\left(2\pi\frac{y_R(n)}{d}\right)\right) \\ &+ \jmath\left(\lambda(n)y_I(n)\left(\gamma_M^2 - y_I^2(n)\right) + (1-\lambda(n))\,\beta\left(\frac{\pi}{d}\right)\sin\left(2\pi\frac{y_I(n)}{d}\right)\right). \end{aligned} \tag{4.8}$$

During the initial stages of adaptation, MMA updates with a large step size will be applied to the equalizer taps most of the time, quickly decreasing the MSE. This allows the CME error term to be included at an earlier stage blended with MMA error. The CME error is able to rapidly decrease the convergence time and MSE since it contains knowledge of the constellation. Once the MSE has been reduced to low levels, CME updates with a small step size will be applied to the equalizer taps most of the time, reducing the steady-state MSE and misadjustment. The use of

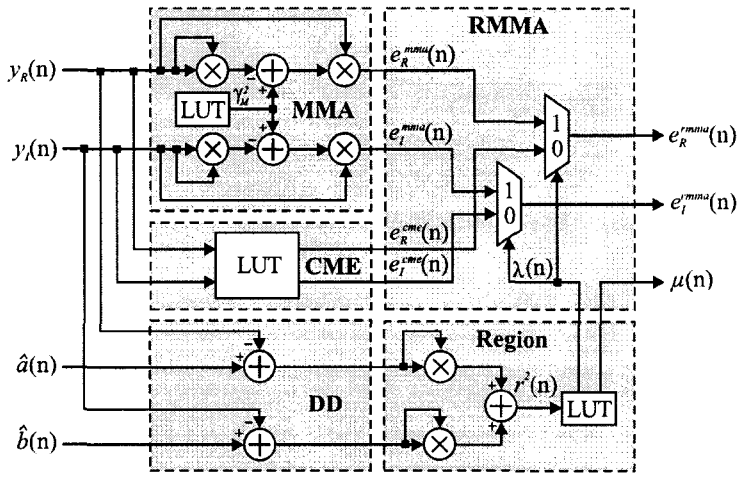


Figure 4.2: RMMA error signal realization when $\lambda(n) \in \{0, 1\}$.

fixed decision regions in RMMA serves a similar purpose as the weighting factor in data-dependent MCMA [41]. This would suggest that the radius-adjusted approach could replace the weighting factor in data-dependent MCMA to achieve similar results with less complexity.

When considering practical implementations of RMMA, as mentioned earlier, the parameters p and q are set to '2' and '1', respectively, while the CME term $\beta\eta(n)$ is implemented with LUTs. The total number of arithmetic computations required to implement the RMMA error signal in (4.8) are $8M + 6A$, where 'M' and 'A' correspond to the number of real multiplications and additions, respectively. A more efficient implementation is to quantize the weighting factor to $\lambda(n) \in \{0, 1\}$. The effect of this quantization is the RMMA error signal acting as a multiplexer, selecting either MMA or CME error during an iteration. This senario reduces the number of arithmetic operations to that of MMA ($4M + 2A$) and CME (LUT), respectively. Thus far the overhead required to determine the appropriate region has not been considered. This overhead requires $k - 1$ comparisons for k regions and $2M + 3A + 1SQRT$ for the calculation of $r(n)$, while '$SQRT$' is the square root function. However, if the operand precision is sufficient, $r^2(n)$ can be used to calculate the region eliminating the need for another LUT or a hardware implementation of the $SQRT$ function. Therefore, considering these simplifications, the total number of arithmetic calculations required to implement the RMMA error signal is $6M + 5A$.

4.3.2 Radius-Adjusted Multimodulus Decision-Directed Algorithm

The radius-adjusted multimodulus decision-directed Algorithm(RMDA) combines the MMA and DD cost functions that were defined in (3.30) and (3.34), respectively, and linearly weighs the respective terms based on the radius-adjusted approach. The objective is to obtain reliable and automatic transfer to the DD mode, while decreasing the convergence time and obtaining low steady-state MSE and misadjustment. The cost function that is minimized by RMDA is defined as

$$J^{\text{rmda}} = \frac{1}{2p}\lambda(n)E\left\{(y_R^p(n)-\gamma_M^p)^2 + \jmath\,(y_I^p(n)-\gamma_M^p)^2\right\} + \frac{1}{2}\left(1-\lambda(n)\right)E\left\{(\hat{s}(n)-y(n))^2\right\} \quad (4.9)$$

where p is a positive constant. A gradient-descent equalizer update algorithm that minimizes J^{rmda} is defined as

$$\begin{aligned}\mathbf{w}(n+1) &= \mathbf{w}(n) + \mu(n)\left(-\nabla_{\mathbf{w}} J^{\text{rmda}}\right) \qquad (4.10)\\ &= \mathbf{w}(n) + \mu(n)\Big(\overbrace{\left(\lambda(n)y_R(n)\,|y_R(n)|^{p-2}\left(\gamma_M^p - y_R^p(n)\right) + \left(1-\lambda(n)\right)\left(\hat{a}(n)-y_R(n)\right)\right)}^{e_R^{\text{rmda}}(n)}\\ &\quad + \jmath\underbrace{\left(\lambda(n)y_I(n)\,|y_I(n)|^{p-2}\left(\gamma_M^p - y_I^p(n)\right) + \left(1-\lambda(n)\right)\left(\hat{b}(n)-y_I(n)\right)\right)}_{e_I^{\text{rmda}}(n)}\Big)\mathbf{x}^*(n)\end{aligned}$$

where $e^{\text{rmda}}(n) = e_R^{\text{rmda}}(n) + \jmath e_I^{\text{rmda}}(n)$ is the RMMA error signal, which is reduced to MMA error when $\lambda(n) = 1$ and to DD error when $\lambda(n) = 0$.

The typical value of p for implementation is '2' since it provides the best compromise between performance and implementation complexity. This case will be considered exclusively throughout the remaining sections of this thesis. With this consideration, the RMDA error signal can then be rewritten as

$$\begin{aligned}e^{\text{rmda}}(n) &= \left(\lambda(n)y_R(n)\left(\gamma_M^2 - y_R^2(n)\right) + \left(1-\lambda(n)\right)\left(\hat{a}(n)-y_R(n)\right)\right)\\ &\quad + \jmath\left(\lambda(n)y_I(n)\left(\gamma_M^2 - y_I^2(n)\right) + \left(1-\lambda(n)\right)\left(\hat{b}(n)-y_I(n)\right)\right). \qquad (4.11)\end{aligned}$$

Similar to RMMA, the equalizer taps of RMDA will initially be updated using MMA with a large step size most of the time, quickly reducing the MSE. This allows the DD error term to be included at an earlier stage blended with MMA error. Once the MSE has been reduced to low levels, DD updates with a small step size will be used to obtain low steady-state MSE and misadjustment. The RMDA achieves automatic transfer between MMA and DD modes and can be applied at the onset of equalization without an initial equalization period.

The total number of arithmetic computations required to implement the RMDA error signal defined in (4.11) are $8M + 8A$. A more efficient realization is to quantize the weighting factor to

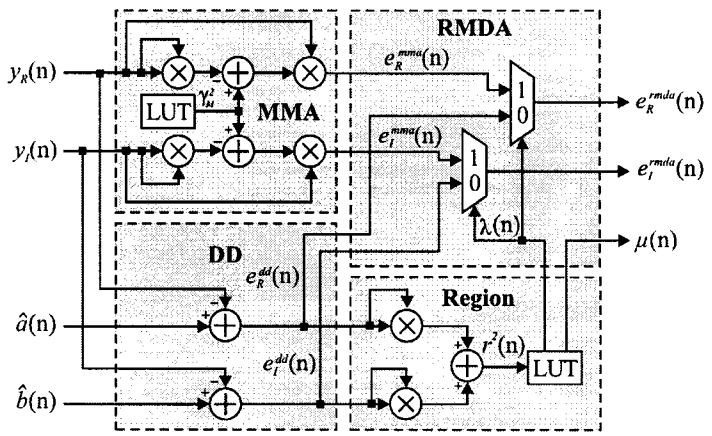


Figure 4.3: RMDA error signal realization when $\lambda(n) \in \{0, 1\}$.

$\lambda(n) \in \{0, 1\}$. The effect of this quantization is the RMDA error signal acting as a multiplexer, selecting either MMA or DD error during a given iteration. This senario reduces the number of arithmetic operations to that of MMA ($4M + 2A$) and DD ($2A$) for a total of $4M + 4A$. Thus far the overhead required to determine the appropriate region has not been considered. This overhead requires an additional $k - 1$ comparisons for k regions and $2M + 1A + 1SQRT$[1] for the calculation of $r(n)$. However, if the operand precision is sufficient, $r^2(n)$ can be used to calculate the region eliminating the need for another LUT or a hardware implementation of the $SQRT$ function. Therefore, considering these simplifications, the total number of arithmetic calculations required to implement the RMDA error signal is $6M + 5A$. This number is equivalent to that for RMMA. However, RMMA requires an additional LUT for the CME error term.

4.3.3 Selection of Region Parameters

A simulation study was performed to relate the radius to different MSE intervals. This was accomplished to justify the mapping of adaptation regions to circular regions around an estimated symbol point and to set guidelines for parameter tuning. The MMA with a small step size was applied to SPIB microwave channels #1, 2, 5, 6, 9, 10, 12, and 13 (refer to [11]), and the mean and standard deviation (S.D.) of $r(n)$ was calculated for 40 realizations each of 16-QAM, 64-QAM, 256-QAM per channel. The results, which have been averaged over all channels and normalized with respect to $d/2$, are listed in Table 4.1. It can be seen that the outer regions of Fig. 4.1(b) can in fact be mapped

[1]This value differs from that for RMMA since the DD error component of RMDA is incorporated into the $r(n)$ calculation, which saves two real additions.

Table 4.1: Statistical properties of $r(n)$ as a function of MSE range.

	16-QAM		64-QAM		256-QAM	
MSE Range (dB)	Mean	S.D.	Mean	S.D.	Mean	S.D.
(-5, -7.5)	1.0359	0.4834	1.2702	0.9058	1.7738	1.7437
(-7.5, -10)	0.9028	0.3606	1.0336	0.6252	1.2869	1.1270
(-10, -12.5)	0.7584	0.3102	0.9091	0.4453	1.0343	0.7474
(-12.5, -15)	0.5909	0.2598	0.8551	0.3434	0.9029	0.5218
(-15, -17.5)	0.4431	0.2036	0.7819	0.3054	0.8353	0.3980
(-17.5, -20)	0.3307	0.1564	0.6502	0.2864	0.8110	0.3333
(-20, -22.5)	0.2458	0.1206	0.5025	0.2428	0.7764	0.3037
(-22.5, -25)	0.1824	0.0921	0.3743	0.1894	0.6799	0.2925

to adaptation phases with high MSE, while the inner regions can in fact be mapped to phases with low MSE. This justifies the assumptions stated earlier in section 4.2. At high values of MSE, these statistics vary by channel. However, simulation results have indicated that as the MSE approaches the minimum required for transfer to the DD algorithm, which is denoted ξ_{DD}, the statistics are independent of channel, SNR, and algorithm. This result can be explained as follows. When the MSE is above ξ_{DD}, $y(n)$ is inaccurate and therefore $\hat{s}(n)$ is unreliable, which leads to high mean and standard deviation of $r(n)$. However, when the MSE is below ξ_{DD}, the accuracy of $y(n)$ is improved and the reliability of $\hat{s}(n)$ is within that required for convergence of the DD algorithm. In this mode of operation $r^2(n)$ approximates the MSE, which is equivalent to the expected value of the squared error across the slicer. This essentially makes $r(n)$ a function of MSE, which is dependent upon the constellation.

The mean and standard range of $r(n)$ are plotted in Fig. 4.4 for 16-QAM, 64-QAM, and 256-QAM using the statistics listed in Table 4.1. These figures can be used to obtain relative performance measures by extracting data from several intersecting lines. The minimum, maximum, and mean values of $r(n)$ are denoted $|r(n)|_{\min}$, $|r(n)|_{\max}$, and $|r(n)|_{\text{ave}}$, respectively, where $|r(n)|_{\min} \leq r(n) \leq |r(n)|_{\max}$ is the standard range. The horizontal line A_1A_2 represents ξ_{DD}, which is approximately -11.19dB, -17.40dB, and -23.5dB for 16-QAM, 64-QAM and 256-QAM, respectively [17][8]. The vertical line B_1B_2 is formed by the intersection of line A_1A_2 with $|r(n)|_{\min}$, while C_1C_2 is formed by the intersection of line A_1A_2 with $|r(n)|_{\text{ave}}$. The outermost boundary for inclusion of constellation specific errors corresponds to point C, which is the intersection of lines A_1A_2 and C_1C_2. This hard limit is approximately 2/3 for 16-QAM, 64-QAM, and 256-QAM. If constellation specific errors are included in regions above point C, the convergence time will increase and divergence may occur since $|r(n)|_{\text{ave}}$ will be in regions whose MSE is greater than ξ_{DD}. The maximum step size, which

Table 4.2: Parameter selection guidelines for RMMA and RMDA.

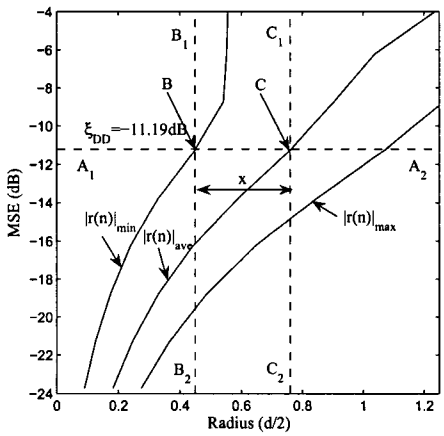
Region	Limits	$\mu(n)$	$\lambda(n)$
1	$r(n) \geq 1$	$\alpha_{\max}\mu_M$	1
2	$\frac{2}{3} \leq r(n) < 1$	$\alpha_{\max}\mu_M/2$	1
3	$\frac{1}{3} \leq r(n) < \frac{2}{3}$	μ_M	$0 \leq \lambda(n) \leq 1$
4	$\frac{1}{6} \leq r(n) < \frac{1}{3}$	μ_M	0
5	$r(n) < \frac{1}{6}$	$\mu_M/\alpha_{\max}$	0

corresponds to the $\mu(n)$ in the outermost decision region, can be determined as follows

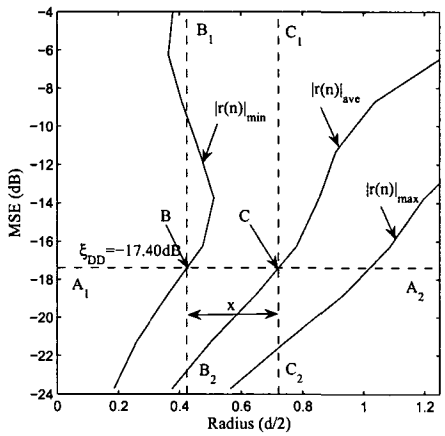
$$\mu_{\max} = \alpha_{\max}\mu_M = \Big(1 + \frac{100}{|\xi_{\text{DD}}|}\Big)\mu_M \tag{4.12}$$

where μ_M is the step size for MMA. The expression for $\alpha_{\max}$ was obtained by relating ξ_{DD} to the rate at which $|r(n)|_{\text{ave}}$ decreases with respect to MSE, which can be seen to approximate $f(x) = q/x$ where q is a positive constellation-specific constant. A model of $1+\kappa/\xi_{\text{DD}}$ was applied to $\alpha_{\max}$, which decreases $\mu_{\max}$ for increasing constellations orders. The constant κ was determined by solving $\alpha_{\max}$ for the $\mu_{\max}$ determined experimentally for 16-QAM that provided desirable transient responses. Subsequently, it was verified that similar transient responses were obtained with $\alpha_{\max}$ defined in (4.12) when applied to larger constellations.

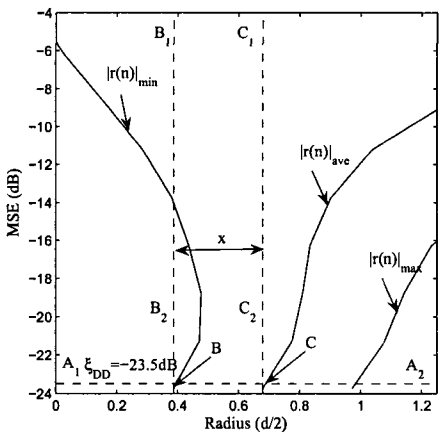
Guidelines for parameter selection are given in Table 4.2, which are based on five regions to allow greater flexibility and smooth transitions between errors and step sizes. Experimental results have indicated that defining three to five regions is sufficient and the performance enhancement of anything above five regions is marginal. Regions $1-2$ represent adaptation phases above ξ_{DD}, while regions $3-5$ represent phases that are at or below ξ_{DD}. Consequently, $\lambda(n)$ is fixed at '1' in regions $1-2$, while $\lambda(n)$ is fixed at '0' in regions $4-5$. However, $\lambda(n)$ is a fixed intermediate value in the range $0 \leq \lambda(n) \leq 1$ in region 3, whose limits corresponds to 'x' in Fig. 4.4. This is to allow smooth transition between the error terms within the respective error signals. However, as mentioned in sections 4.3.1 and 4.3.2, $\lambda(n)$ can also be quantized to $\lambda(n) \in \{0, 1\}$ in this region to produce a more efficient implementation. Simulation results have shown that $\mu(n)$ controls the initial rate of convergence as well as the convergence time and misadjustment. The selection of $\mu(n)$ in regions $1-2$ controls the initial rate of convergence, while $\mu(n)$ in regions $4-5$ controls the misadjustment and convergence time. In general, $\mu(n)$ can vary by region as a factor of $\alpha_{\max}$ and μ_M as indicated by Table 4.2. This reduces the selection of region parameters to two quantities; $\lambda(n)$ for region 3 and $\alpha_{\max}$. Simulation results have shown that this method of varying $\mu(n)$ works well for most channels. However, for some channels, increased performance may be obtained by manually tuning $\mu(n)$ in regions 2 and 4.



(a) Statistical properties for 16-QAM.



(b) Statistical properties for 64-QAM.



(c) Statistical properties for 256-QAM.

Figure 4.4: Statistical properties of $r(n)$ plotted as a function of MSE range.

4.3.4 Steady-State Performance

In this section the steady-state performance of RMMA and RMDA is analyzed based on a fundamental energy-preserving relation, which was first exploited in [60], [58] and [51] during studies on the robustness and l_2-stability of adaptive filters. This relation will be derived using an approach similar as [51] and will then be applied to analyze the steady-state performance of RMMA and RMDA.

Consider the following general stochastic gradient descent algorithm:

$$\mathbf{w}(n) = \mathbf{w}(n-1) + \mu e_i(n)\mathbf{x}^*(n) \tag{4.13}$$

where $e_i(n)$ denotes an instantaneous error signal and $\mathbf{x}(n)$ is a nonzero regressor vector of equalizer input samples. Subtracting the optimal weight vector $\mathbf{w}_o$ from both sides of (4.13), the following weight vector is obtained:

$$\Delta\mathbf{w}(n) = \Delta\mathbf{w}(n-1) - \mu e_i(n)\mathbf{x}^*(n) \tag{4.14}$$

where $\Delta\mathbf{w}(n) = \mathbf{w}_o - \mathbf{w}(n)$. Defining the *a priori* estimation error as $e_a(n) = \mathbf{x}^T(n)\Delta\mathbf{w}(n)$ and the *a posteriori* estimation error as $e_p(n) = \mathbf{x}^T(n)\Delta\mathbf{w}(n-1)$, and then multiplying by $\mathbf{x}^T(n)$, the weight error vector of (4.14) can be rewritten as

$$e_p(n) = e_a(n) - \mu\|\mathbf{x}(n)\|_2^2 e_i(n). \tag{4.15}$$

Solving for $e_i(n)$ leads to

$$e_i(n) = \frac{e_a(n) - e_p(n)}{\mu\|\mathbf{x}(n)\|_2^2}. \tag{4.16}$$

Substituting (4.16) into (4.14) and rearranging leads to the following equality:

$$\Delta\mathbf{w}(n) + \frac{\mathbf{x}^*(n)}{\|\mathbf{x}(n)\|_2^2}e_a(n) = \Delta\mathbf{w}(n-1) + \frac{\mathbf{x}^*(n)}{\|\mathbf{x}(n)\|_2^2}e_p(n). \tag{4.17}$$

The fundamental energy relation is obtained by replacing the terms in (4.17) with their respective energies as follows

$$\|\Delta\mathbf{w}(n)\|_2^2 + \bar{\mu}(n)|e_a(n)|^2 = \|\Delta\mathbf{w}(n-1)\|_2^2 + \bar{\mu}(n)|e_p(n)|^2 \tag{4.18}$$

where $\bar{\mu}(n) = 1/\|\mathbf{x}(n)\|_2^2$. The relation in (4.18) is derived without any approximations and establishes that the mapping from $\left\{\Delta\mathbf{w}(n-1), \sqrt{\bar{\mu}(n)}e_p(n)\right\}$ to the variables $\left\{\Delta\mathbf{w}(n), \sqrt{\bar{\mu}(n)}e_a(n)\right\}$ is energy preserving.

The steady-state MSE of blind stochastic gradient descent algorithms is given by

$$\xi_{ss} = \lim_{n\to\infty} E\left\{\left|y(n) - s(n-\delta)e^{j\theta}\right|^2\right\} = \lim_{n\to\infty} E\left\{\left|e_a(n)\right|^2\right\} \tag{4.19}$$

where $e^{j\theta}$ is an arbitrary carrier phase offset that is unavoidable for equalizers adapted with blind algorithms. The right most side of (4.19) is obtained by noting that $\mathbf{x}^T(n)\Delta\mathbf{w}(n) = s(n-\delta)e^{j\theta} - y(n)$ and $e_a(n) = \mathbf{x}^T(n)\Delta\mathbf{w}(n)$. Taking the expected value of (4.18) and noting that $E\left\{\|\Delta\mathbf{w}\|_2^2\right\} = E\left\{\|\Delta\mathbf{w}(n-1)\|_2^2\right\}$ in steady-state operation, leads to the following equality:

$$E\left\{\bar{\mu}(n)|e_a(n)|^2\right\} = E\left\{\bar{\mu}(n)|e_p(n)|^2\right\}. \tag{4.20}$$

Substituting (4.15) for $e_p(n)$, the equality in (4.20) is expanded to

$$E\left\{\bar{\mu}(n)\left|e_a(n)\right|^2\right\} = E\left\{\bar{\mu}(n)\left|e_a(n)\right|^2\right\} - \underbrace{\mu E\left\{e_a^*(n)e_i(n) + e_a(n)e_i^*(n)\right\}}_{T1} + \underbrace{\mu^2 E\left\{\|\mathbf{x}(n)\|_2^2\left|e_i(n)\right|^2\right\}}_{T2} \tag{4.21}$$

which implies that the terms $T1$ and $T2$ are equivalent. Therefore, an approximate expression for $E\left\{|e_a(n)|^2\right\}$ can be obtained by evaluating $T1 = T2$. The analysis that follows for RMMA and RMDA is based on the following two assumptions in steady-state operation as per [51]:

1. The transmitted signal $s(n)$ and the estimation error $e_a(n)$ are independent in steady-state so that $E\left\{s(n)e_a(n)\right\} = 0$ since $s(n)$ is assumed zero mean. Furthermore, this implies that $E\left\{s_R(n)e_{a,R}(n)\right\} = 0$, $E\left\{s_I(n)e_{a,R}(n)\right\} = 0$, $E\left\{s_R(n)e_{a,I}(n)\right\} = 0$, and $E\left\{s_I(n)e_{a,I}(n)\right\} = 0$, since it is assumed that $s_R(n)$ and $s_I(n)$ have the same statistics, while $s(n)$ is rewritten as $s(n) = s_R(n) + js_I(n)$ to simplify the notation where $s_R(n) = a(n)$ and $s_I(n) = b(n)$, respectively.

2. The scaled regressor energy $\mu^2\|\mathbf{x}(n)\|_2^2$ is independent of $y(n)$ in steady-state.

The first assumption requires that the estimation error $e_a(n)$ of the equalizer is insensitive to the actual transmitted symbols. This is a common assumption for steady-state analysis [29]. The second assumption requires that the scaled regressor energy of the input vector to be independent of the equalizer output. This assumption becomes realistic for equalizers with long filter lengths and sufficiently small step sizes [51] as $\mu^2\|\mathbf{x}(n)\|_2^2$ will tend to a constant and can, therefore, be assumed to be independent of $y(n)$.

Steady-State Analysis of RMMA

The steady-state analysis of RMMA to follow considers the RMMA error signal which was defined in (4.8). Prior to the evaluation of (4.21) for RMMA, it is necessary to compute the Taylor series

expansion of $g(x)$ around the point $s(n)$. The general expression for the Taylor series expansion around point 'a' is given by

$$f(x) = \sum_{i=0}^{\infty} \frac{d^n}{dx^n}\{f(a)\}\frac{(x-a)^n}{n!} \tag{4.22}$$
$$= f(a) + \frac{d}{dx}\{f(a)\}\frac{(x-a)}{1!} + \frac{d^2}{dx^2}\{f(a)\}\frac{(x-a)^2}{2!} + \cdots + \frac{d^n}{dx^n}\{f(a)\}\frac{(x-a)^n}{n!}.$$

Noting that $\cos(2\pi x/d) = -1$ and $\sin(2\pi x/d) = 0$ for all $x \in \{s_m\}_{m=1}^{M}$ for M-QAM constellations, the value of $g(x)$ and its first four derivatives are given as follows

$$\begin{aligned}
g(x) &= \frac{1}{2}\left(1 + \cos\left(2\pi\frac{x}{d}\right)\right) = 0 \\
\frac{d}{dx}g(x) &= -\frac{\pi}{d}\sin\left(2\pi\frac{x}{d}\right) = 0 \\
\frac{d^2}{dx^2}g(x) &= -2\left(\frac{\pi}{d}\right)^2\cos\left(2\pi\frac{x}{d}\right) = -2\left(\frac{\pi}{d}\right)^2 \\
\frac{d^3}{dx^3}g(x) &= 2^2\left(\frac{\pi}{d}\right)^3\sin\left(2\pi\frac{x}{d}\right) = 0 \\
\frac{d^4}{dx^4}g(x) &= 2^3\left(\frac{\pi}{d}\right)^4\cos\left(2\pi\frac{x}{d}\right) = 8\left(\frac{\pi}{d}\right)^4.
\end{aligned}$$

If the higher order terms are ignored and substituting '$x \rightarrow y(n)$', the Taylor series expansion of $g(y(n))$ around $s(n)$ can be approximated as

$$g(y(n)) \approx -\left(\frac{\pi}{d}\right)^2 (y(n) - s(n))^2. \tag{4.23}$$

Based on the Taylor series expansion of $g(y(n))$, the CME term $\eta(n)$ can be rewritten as

$$\eta(n) \approx 2\left(\frac{\pi}{d}\right)^2 ((y_R(n) - s_R(n)) + \jmath(y_I(n) - s_I(n))) = 2\left(\frac{\pi}{d}\right)^2 (y(n) - s(n)). \tag{4.24}$$

Substituting (4.24) into (4.8), the RMMA error signal can be rewritten as

$$\begin{aligned}
e^{\text{rmma}}(n) = {} & \lambda(n)y_R(n)\left(\gamma_M^2 - y_R^2(n)\right) + \alpha(n)\left(y_R(n) - s_R(n)\right) \\
& + \jmath\left(\lambda(n)y_I(n)\left(\gamma_M^2 - y_I^2(n)\right) + \alpha(n)\left(y_I(n) - s_I(n)\right)\right)
\end{aligned} \tag{4.25}$$

where $\alpha(n) = 2(1 - \lambda(n))\beta\left(\frac{\pi}{d}\right)^2$. Furthermore, by replacing $y_R(n)$ and $y_I(n)$ with $s_R(n) + e_{a,R}(n)$ and $s_I(n) + e_{a,I}(n)$, respectively, the RMMA error signal of (4.25) can expanded to

$$\begin{aligned}
e^{\text{rmma}}(n) = {} & \lambda(n)\left(s_R(n) + e_{a,R}(n)\right)\left(\gamma_M^2 - s_R^2(n) - 2s_R(n)e_{a,R}(n) - e_{a,R}^2(n)\right) + \alpha(n)e_{a,R}(n) \\
& + \jmath\left(\lambda(n)\left(s_I(n) + e_{a,I}(n)\right)\left(\gamma_M^2 - s_I^2(n) - 2s_I(n)e_{a,I}(n) - e_{a,I}^2(n)\right) + \alpha(n)e_{a,I}(n)\right).
\end{aligned} \tag{4.26}$$

By substituting $e_i(n)$ with $e^{\text{rmma}}(n)$, the terms $T1$ and $T2$ in (4.21) can be computed as follows

$$
\begin{aligned}
T1 &= \mu E\left\{e_a^* e_i + e_a e_i^*\right\} \qquad (4.27)\\
&= 2\mu E\left\{\left(\lambda\gamma_M^2+\alpha\right)|e_a|^2\right\} + 2\mu\gamma_M^2 E\left\{\lambda\left(e_a^* s + e_a s^*\right)\right\}\\
&\quad - 2\mu E\left\{\lambda\left(s_R^3 e_{a,R} + s_I^3 e_{a,I} + e_{a,R}^4 + e_{a,I}^4\right)\right\} - 6\mu E\left\{\lambda\left(s_R^2 e_{a,R}^2 + s_R e_{a,R}^3 + s_I^2 e_{a,I}^2 + s_I e_{a,I}^3\right)\right\}\\
T2 &= \mu^2 E\left\{\|\mathbf{x}\|_2^2 |e_i|^2\right\} \qquad (4.28)\\
&= \mu^2 E\left\{\|\mathbf{x}\|_2^2\alpha^2\right\}\cdot E\left\{|e_a|^2\right\} + 2\mu^2\gamma_M^2 E\left\{\|\mathbf{x}\|_2^2\alpha\lambda\right\}\cdot E\left\{|e_a|^2\right\} - 2\mu^2\gamma_M^2 E\left\{\lambda^2\|\mathbf{x}\|_2^2\left(s_R^4+s_I^4\right)\right\}\\
&\quad + \mu^2 E\left\{\lambda^2\|\mathbf{x}\|_2^2\left(s_R^6+s_I^6\right)\right\} + \mu^2\gamma_M^4 E\left\{\lambda^2\|\mathbf{x}\|_2^2|s|^2\right\} + \mu^2\gamma_M^4 E\left\{\lambda^2\|\mathbf{x}\|_2^2\right\}\cdot E\left\{|e_a|^2\right\}\\
&\quad - 6\mu^2 E\left\{\|\mathbf{x}\|_2^2\lambda\alpha\left(s_R e_{a,R}^3 + s_R^2 e_{a,R}^2 + s_I^2 e_{a,I}^2 + s_I e_{a,I}^3\right)\right\} + 2\mu^2\gamma_M^2 E\left\{\|\mathbf{x}\|_2^2\lambda\alpha\left(s^* e_a + e_a^* s\right)\right\}\\
&\quad - 2\mu^2 E\left\{\|\mathbf{x}\|_2^2\lambda\alpha\left(s_I^3 e_{a,I} + s_R^3 e_R + e_{a,R}^4 + e_{a,I}^4\right)\right\} - 2\mu^2\gamma_M^2 E\left\{\lambda^2\|\mathbf{x}\|_2^2\left(e_{a,I}^4 + e_{a,R}^4\right)\right\}\\
&\quad + 6\mu^2 E\left\{\lambda^2\|\mathbf{x}\|_2^2\left(s_I^5 e_{a,I} + s_I e_{a,I}^5 + s_R^5 e_{a,R} + s_R e_{a,R}^5\right)\right\} + \mu^2 E\left\{\lambda^2\|\mathbf{x}\|_2^2\left(e_{a,R}^6 + e_{a,I}^6\right)\right\}\\
&\quad + 15\mu^2 E\left\{\lambda^2\|\mathbf{x}\|_2^2\left(s_R^4 e_{a,R}^2 + s_R^2 e_{a,R}^4 + s_I^4 e_{a,I}^2 + s_I^2 e_{a,I}^4\right)\right\} + 20\mu^2 E\left\{\lambda^2\|\mathbf{x}\|_2^2\left(s_R^3 e_{a,R}^3 + s_I^3 e_{a,I}^3\right)\right\}\\
&\quad - 12\mu^2\gamma_M^2 E\left\{\lambda^2\|\mathbf{x}\|_2^2\left(s_R^2 e_{a,R}^2 + s_I^2 e_{a,I}^2\right)\right\} + 2\mu^2\gamma_M^4 E\left\{\lambda^2\|\mathbf{x}\|_2^2\left(s^* e_a + e_a^* s\right)\right\}\\
&\quad - 8\mu^2\gamma_M^2 E\left\{\lambda^2\|\mathbf{x}\|_2^2\left(s_R^3 e_{a,R} + s_I e_{a,I}^3 + s_R e_{a,R}^3 + s_I^3 e_{a,I}\right)\right\}
\end{aligned}
$$

where the time references have been omitted to improve the readability. Applying the assumptions stated earlier and neglecting the high order terms of e_a, $T1$ and $T2$ can be approximated as follows

$$T1 \approx E\left\{\left(\lambda\gamma_M^2+\alpha\right)\right\} E\left\{|e_a|^2\right\} \qquad (4.29)$$

$$
\begin{aligned}
T2 &\approx \mu^2 E\left\{\|\mathbf{x}\|_2^2\alpha^2\right\} E\left\{|e_a|^2\right\} + 2\mu^2\gamma_M^2 E\left\{\|\mathbf{x}\|_2^2\alpha\lambda\right\} E\left\{|e_a|^2\right\} - 2\mu^2\gamma_M^2 E\left\{\lambda^2\|\mathbf{x}\|_2^2\left(s_R^4+s_I^4\right)\right\}\\
&\quad + \mu^2 E\left\{\lambda^2\|\mathbf{x}\|_2^2\left(s_R^6+s_I^6\right)\right\} + \mu^2\gamma_M^4 E\left\{\lambda^2\|\mathbf{x}\|_2^2|s|^2\right\} + \mu^2\gamma_M^4 E\left\{\lambda^2\|\mathbf{x}\|_2^2\right\} E\left\{|e_a|^2\right\}. \qquad (4.30)
\end{aligned}
$$

The steady-state MSE for RMMA, denoted $\xi_{\text{ss}}^{\text{rmma}}$, is obtained by equating $T1$ and $T2$, which results in the following expression for the steady-state MSE for RMMA:

$$\xi_{\text{ss}}^{\text{rmma}} = E\left\{|e_a|^2\right\} \approx \frac{E\left\{\lambda^2\right\} E\left\{\left(s_R^6 + s_I^6 - 2\gamma_M^2\left(s_R^4+s_I^4\right) + \gamma_M^4|s|^2\right)\right\}}{\frac{2E\left\{\left(\lambda\gamma_M^2+\alpha\right)\right\}}{\mu E\left\{\|\mathbf{x}\|_2^2\right\}} - E\left\{\left(\alpha^2 + 2\gamma_M^2\alpha\lambda + \lambda^2\gamma_M^4\right)\right\}}. \qquad (4.31)$$

Several important observations can be drawn from (4.31), which characterize the steady-state performance of RMMA. When the step size μ is sufficiently small, the first term in the denominator is much larger than the second term. If the second term is neglected, the steady-state MSE of RMMA is proportional to the step size μ and the variance of the regressor vector of equalizer input samples $E\left\{\|\mathbf{x}\|_2^2\right\}$, which is similar to both LMS and CMA [51]. In addition, the steady-state MSE of RMMA is proportional to $E\left\{\lambda^2\right\}$. This means that for nonconstant modulus signals, such as for

QAM signals, as $E\{\lambda^2\} \to 0$ (i.e. when the RMMA equalizer update is dominated by the CME error signal), the steady-state MSE of RMMA will tend towards zero in the absence of noise. While this is true for LMS, this is not true for CMA since the CMA equalizer update is non-zero for all $y(n) \in \{s_m\}_{m=1}^{M}$ for M-QAM constellations.

Steady-State Analysis of RMDA

The steady-state analysis of RMDA to follow considers the RMDA error signal which was defined in (4.11). By replacing $y_R(n)$ and $y_I(n)$ with $s_R(n)+e_{a,R}$ and $s_I(n)+e_{a,I}(n)$, respectively, the RMDA error signal of can expanded as

$$\begin{aligned} e^{\text{rmda}}(n) = \lambda(n)\left(s_R(n)+e_{a,R}(n)\right)\left(\gamma_M^2 - s_R^2(n) - 2s_R(n)e_{a,R}(n) - e_{a,R}^2(n)\right) - \rho(n)e_{a,R}(n) \\ + \jmath\left(\lambda(n)\left(s_I(n)+e_{a,I}(n)\right)\left(\gamma_M^2 - s_I^2(n) - 2s_I(n)e_{a,I}(n) - e_{a,I}^2(n)\right) - \rho(n)e_{a,I}(n)\right) \end{aligned} \tag{4.32}$$

where $\rho(n) = (1-\lambda(n))$. Substituting $e_i(n)$ with $e^{\text{rmma}}(n)$, the terms $T1$ and $T2$ in (4.21) can be computed as follows

$$\begin{aligned} T1 &= \mu E\{e_a^* e_i + e_a e_i^*\} \\ &= 2\mu E\left\{\left(\lambda\gamma_M^2 - \rho\right)|e_a|^2\right\} + 2\mu\gamma_M^2 E\{\lambda\left(e_a^* s + e_a s^*\right)\} \\ &\quad - 2\mu E\left\{\lambda\left(s_R^3 e_{a,R} + s_I^3 e_{a,I} + e_{a,R}^4 + e_{a,I}^4\right)\right\} - 6\mu E\left\{\lambda\left(s_R^2 e_{a,R}^2 + s_R e_{a,R}^3 + s_I^2 e_{a,I}^2 + s_I e_{a,I}^3\right)\right\} \end{aligned} \tag{4.33}$$

$$\begin{aligned} T2 &= \mu^2 E\left\{\|\mathbf{x}\|_2^2 |e_i|^2\right\} \\ &= \mu^2 E\left\{\|\mathbf{x}\|_2^2\rho^2\right\}\cdot E\left\{|e_a|^2\right\} - 2\mu^2\gamma_M^2 E\left\{\|\mathbf{x}\|_2^2\rho\lambda\right\}\cdot E\left\{|e_a|^2\right\} - 2\mu^2\gamma_M^2 E\left\{\lambda^2\|\mathbf{x}\|_2^2\left(s_R^4 + s_I^4\right)\right\} \\ &\quad + \mu^2 E\left\{\lambda^2\|\mathbf{x}\|_2^2\left(s_R^6 + s_I^6\right)\right\} + \mu^2\gamma_M^4 E\left\{\lambda^2\|\mathbf{x}\|_2^2|s|^2\right\} + \mu^2\gamma_M^4 E\left\{\lambda^2\|\mathbf{x}\|_2^2\right\}\cdot E\left\{|e_a|^2\right\} \\ &\quad + 6\mu^2 E\left\{\|\mathbf{x}\|_2^2\lambda\rho\left(s_R e_{a,R}^3 + s_R^2 e_{a,R}^2 + s_I^2 e_{a,I}^2 + s_I e_{a,I}^3\right)\right\} - 2\mu^2\gamma_M^2 E\left\{\|\mathbf{x}\|_2^2\lambda\rho\left(s^* e_a + e_a^* s\right)\right\} \\ &\quad + 2\mu^2 E\left\{\|\mathbf{x}\|_2^2\lambda\rho\left(s_I^3 e_{a,I} + s_R^3 e_R + e_{a,R}^4 + e_{a,I}^4\right)\right\} - 2\mu^2\gamma_M^2 E\left\{\lambda^2\|\mathbf{x}\|_2^2\left(e_{a,I}^4 + e_{a,R}^4\right)\right\} \\ &\quad + 6\mu^2 E\left\{\lambda^2\|\mathbf{x}\|_2^2\left(s_I^5 e_{a,I} + s_I e_{a,I}^5 + s_R^5 e_{a,R} + s_R e_{a,R}^5\right)\right\} + \mu^2 E\left\{\lambda^2\|\mathbf{x}\|_2^2\left(e_{a,R}^6 + e_{a,I}^6\right)\right\} \\ &\quad + 15\mu^2 E\left\{\lambda^2\|\mathbf{x}\|_2^2\left(s_R^4 e_{a,R}^2 + s_R^2 e_{a,R}^4 + s_I^4 e_{a,I}^2 + s_I^2 e_{a,I}^4\right)\right\} + 20\mu^2 E\left\{\lambda^2\|\mathbf{x}\|_2^2\left(s_R^3 e_{a,R}^3 + s_I^3 e_{a,I}^3\right)\right\} \\ &\quad - 12\mu^2\gamma_M^2 E\left\{\lambda^2\|\mathbf{x}\|_2^2\left(s_R^2 e_{a,R}^2 + s_I^2 e_{a,I}^2\right)\right\} + 2\mu^2\gamma_M^4 E\left\{\lambda^2\|\mathbf{x}\|_2^2\left(s^* e_a + e_a^* s\right)\right\} \\ &\quad - 8\mu^2\gamma_M^2 E\left\{\lambda^2\|\mathbf{x}\|_2^2\left(s_R^3 e_{a,R} + s_I e_{a,I}^3 + s_R e_{a,R}^3 + s_I^3 e_{a,I}\right)\right\} \end{aligned} \tag{4.34}$$

where the time references have been omitted to improve the readability. Applying the assumptions stated earlier and neglecting the high order terms of e_a, $T1$ and $T2$ can be approximated as follows

$$T1 \approx E\left\{\left(\lambda\gamma_M^2 - \rho\right)\right\} E\left\{|e_a|^2\right\} \tag{4.35}$$

$$\begin{aligned} T2 \approx{}& \mu^2 E\left\{\|\mathbf{x}\|_2^2\rho^2\right\} E\left\{|e_a|^2\right\} - 2\mu^2\gamma_M^2 E\left\{\|\mathbf{x}\|_2^2\rho\lambda\right\} E\left\{|e_a|^2\right\} - 2\mu^2\gamma_M^2 E\left\{\lambda^2\|\mathbf{x}\|_2^2\left(s_R^4 + s_I^4\right)\right\} \\ &+ \mu^2 E\left\{\lambda^2\|\mathbf{x}\|_2^2\left(s_R^6 + s_I^6\right)\right\} + \mu^2\gamma_M^4 E\left\{\lambda^2\|\mathbf{x}\|_2^2|s|^2\right\} + \mu^2\gamma_M^4 E\left\{\lambda^2\|\mathbf{x}\|_2^2\right\} E\left\{|e_a|^2\right\}. \end{aligned} \tag{4.36}$$

The steady-state MSE for RMDA, denoted $\xi_{\text{ss}}^{\text{rmda}}$, is obtained by equating $T1$ and $T2$, which results in the following expression for the steady-state MSE for RMDA:

$$\xi_{\text{ss}}^{\text{rmda}} = E\left\{|e_a|^2\right\} \approx \frac{E\left\{\lambda^2\right\} E\left\{\left(s_R^6 + s_I^6 - 2\gamma_M^2\left(s_R^4 + s_I^4\right) + \gamma_M^4|s|^2\right)\right\}}{\frac{2E\left\{\left(\lambda\gamma_M^2 - \rho\right)\right\}}{\mu E\left\{\|\mathbf{x}\|_2^2\right\}} - E\left\{\left(\rho^2 - 2\gamma_M^2\rho\lambda + \lambda^2\gamma_M^4\right)\right\}}. \tag{4.37}$$

Several important observations can be drawn from (4.37), which characterize the steady-state performance of RMDA. When the step size μ is sufficiently small, the first term in the denominator is much larger than the second term. If the second term is neglected, the steady-state MSE of RMDA is proportional to the step size μ and the variance of the regressor vector of equalizer input samples $E\left\{\|\mathbf{x}\|_2^2\right\}$, which is similar to both LMS and CMA [51]. In addition, the steady-state MSE of RMDA is proportional to $E\left\{\lambda^2\right\}$. This means that for nonconstant modulus signals, such as for QAM signals, as $E\left\{\lambda^2\right\} \to 0$ (i.e. when the RMDA equalizer update is dominated by the DD error signal), the steady-state MSE of RMDA will tend towards zero in the absence of noise. While this is true for LMS, this is not true for CMA since the CMA equalizer update is non-zero for all $y(n) \in \{s_m\}_{m=1}^M$ for M-QAM constellations.

4.4 Extension to Computationally-Efficient Methods

This section extends the radius-adjusted approach discussed in 4.2 to a computationally-efficient method which has been termed the selective update method. This method employs a "stop-and-go" strategy based on the equalizer output radius to selectively update the equalizer tap coefficients.

4.4.1 Selective Update Method

The adaptation criteria employed by the selective update method is a "stop-and-go" strategy based on the equalizer output radius $r(n)$. The equalizer tap coefficients are only adjusted during iterations when $r(n) > r_S$, where r_S is a constant bound. The adaptation is controlled by the flag $\zeta(n)$, which results in the following equalizer tap adjustment algorithm for the selective update method:

$$\mathbf{w}(n+1) = \mathbf{w}(n) + \mu\zeta(n)e(n)\mathbf{x}^*(n) \tag{4.38}$$

where $e(n)$ is the error signal of the respective algorithm. The flag $\zeta(n)$ is defined as

$$\zeta(n) = \begin{cases} 1, & \text{if } r(n) > r_S \\ 0, & \text{otherwise} \end{cases} \tag{4.39}$$

where d is the distance between symbol points in the QAM constellation and $r_S = \alpha_S d/2$, while α_S is the selective update parameter. This parameter is to be chosen between the following limits:

$$2\underbrace{\mathbf{u}^T \max|\mu e(n)\mathbf{x}^*(n)|}_{\varphi} \leq \alpha_S \leq \frac{2}{3} \tag{4.40}$$

where φ is the maximum adjustment error that can occur when the equalizer output is a constellation point (i.e. when $y(n) \in \{s_m\}_{m=1}^{M}$ for an M-QAM constellation). The maximum adjustment error for both CMA and MMA equalizers occurs when $y(n) = (\sqrt{M}-1)d/\sqrt{2}$, where $M \geq 16$. When the equalizer is in steady-state operation, the expected equalizer tap update is as follows

$$E\{\mathbf{w}(n+1)\} = E\{\mathbf{w}(n)\} + \mu\underbrace{E\{e(n)\mathbf{x}^*(n)\}}_{\text{adjustment}=0}. \tag{4.41}$$

Although the adjustment portion of the equalizer tap update has a zero mean, it has a non-zero variance. When statistical mean algorithms are employed, such GSA, CMA and MMA, the adjustment error remains large in steady-state operation. This accentuates the "bottom of the bowl" scenario of classical gradient search methods, where the equalizer tap coefficients bounce around the optimal solution. These fluctuations increase the steady-state MSE thereby increasing the excess MSE (EMSE), which is the difference between the steady-state MSE and the MMSE, which was defined in (3.11). The upper limit of α_S in (4.40) corresponds to the minimum level of MSE required for transfer to the DD algorithm (i.e. ξ_{DD}) [8][17], while the lower limit corresponds to twice the maximum adjustment error. At the uppermost limit, the number of equalizer tap updates will be minimized at the expense of a high steady-state MSE. At the lowermost limit, the steady-state MSE will be equivalent or better than the original with slightly fewer updates. When selecting α_S, it is important to note that when the MSE level is below ξ_{DD}, the steady-state MSE for the original algorithm, denoted ξ_{ss}, can be approximated as the expected squared-error across the slicer:

$$\xi_{\text{ss}} \approx E\left\{|\hat{s}(n) - y(n)|^2\right\} = E\left\{r^2(n)\right\}. \tag{4.42}$$

The relationship between the steady-state MSE for the selective update method, denoted $\xi_{\text{ss}}^{\text{su}}$, and ξ_{ss} is as follows

$$\xi_{\text{ss}}^{\text{su}} \begin{cases} < \xi_{\text{ss}}, & \text{if } \xi_{\text{ss}} \leq r_S^2 \text{ and } r_S^2 \not\gg \xi_{\text{ss}} \\ > \xi_{\text{ss}}, & \text{if } \xi_{\text{ss}} \leq r_S^2 \text{ and } r_S^2 \gg \xi_{\text{ss}} \\ = \xi_{\text{ss}}, & \text{if } \xi_{\text{ss}} \gg r_S^2 \end{cases}. \tag{4.43}$$

Table 4.3: Arithmetic operations for the equalizer tap adjustment for a single FIR filter for the selective update methods ($\mu = 2^{-n}$).

Method	Multiplications	Additions	Barrel Shifts	Update Percentage
Selective Update	N	N	1	$(0,1)$
Modified Selective Update	0	N	$N+1$	$(0,1)$

This can be explained as follows: if $r_S^2 \geq \xi_{ss}$, then $r_S^2 \geq E\{r(n)^2\}$, which will cause a significant reduction in equalizer tap updates. This can effect ξ_{ss}^{su} constructively or destructively, depending on the selection of r_S (i.e. α_S). If $r_S^2 \gg \xi_{ss}$, the steady-state MSE will approach r_S^2 since the equalizer tap coefficients will only be updated once ξ_{ss}^{su} is degraded. However, as $r_S^2 \rightarrow \xi_{ss}$, the ξ_{ss}^{su} will decrease to the point where $\xi_{ss}^{su} < \xi_{ss}$. At first this may seem counterintuitive, however, recall the concept of adjustment error introduced by the equalizer tap update. Halting adaptation when the equalizer output is near symbol points can mitigate the unnecessary equalizer tap updates that cause the EMSE to increase. If $r_S^2 < \xi_{ss}$, then $r_S^2 < E\{r^2(n)\}$, which will cause frequent equalizer tap adjustments and performance will be near that of the original.

At the onset of equalization the equalizer output will be a random i.i.d. value, which will result in the following equalizer update probability:

$$\Pr[\text{update}] = \frac{d^2 - (\alpha_S d/2)^2 \pi}{d^2} = 1 - \frac{\alpha_S^2 \pi}{4}. \tag{4.44}$$

This probability decreases as the equalizer adapts and reaches a minimum when the equalizer is in steady-state operation. During the initial stages of adaptation, $E\{r(n)\} \gg r_S$ causing the equalizer taps to be updated frequently. This allows the respective algorithm to maintain its transient characteristics. As the $E\{r(n)^2\} \rightarrow \xi_{ss} \vee E\{r(n)\} \rightarrow r_S$, the equalizer is in steady-state operation and the number of equalizer tap updates will be at a minimum. If the channel should experience sudden changes, the MSE will increase and the process will repeat.

The computational requirements for the selective update method are specified in Table 4.3. There are no reductions in the hardware resources needed for implementation. However, there is a reduction in the update frequency which enables efficient use of processor capacity and reduces power consumption.

4.4.2 Modified Selective Update Method

While the original selective update method enables efficient use of processor capacity and reduces power consumption, there is no reduction in the hardware resources needed for implementation. This could be overcome by the application of several of the efficient methods discussed in section

3.7. The method considered here is the power-of-two error signal method, which was discussed in section 3.7.2. The equalizer tap adjustment algorithm for the modified selective update method is defined as

$$\mathbf{w}(n+1) = \mathbf{w}(n) + \mu\zeta(n)Q_2\left\{e(n)\right\}\mathbf{x}^*(n) \tag{4.45}$$

where $Q_2\left\{\cdot\right\}$ is the power-of-two quantizer defined in (3.56), $e(n)$ is the error signal of the respective algorithm, and $\zeta(n)$ is the selective update flag defined in (4.39).

The computational requirements for the modified selective update method are specified in Table 4.3. If a power-of-two step size is applied, the modification of the error signal eliminates the multiplications for the equalizer tap update. These multiplications are replaced by shift and add operations, which significantly reduces the implementation area. The expected consequence of this modification is a slight increase in the convergence time.

Chapter 5

Simulations

5.1 Introduction

This chapter presents simulation studies of both new and existing adaptive algorithms that are applied for blind equalization. The chapter begins with a discussion on the selection of the equalizer filter length and continues with evaluation metrics that are applied to quantify the simulation results. These include quantitative measures of the ISI, MSE, and misadjustment. Channel models are described for microwave radio, cable, and fading wireless channels. Empirically derived models are used for microwave radio and cable channels, while Ricean channel models are used for fading mobile radio channels. Simulation results are presented for the blind equalization algorithms: GSA, CMA, and MMA, which is followed by results for the radius-adjusted blind equalization algorithms: RMMA and RMDA. Lastly, comparative studies are presented which compare the new radius-adjusted algorithms to related hybrid algorithms and the selective update method to related computationally-efficient methods.

5.2 Equalizer Length Selection

Although, there is no general solution, the selection of the equalizer length depends on the type of channel and the sample rate relative to the transmitted bandwidth [75]. In practice, there are two common approaches to select the equalizer length:

1. Build a prototype of the equalizer and test it against a variety of actual channels.

2. Apply rules of thumb that seem intuitively reasonable.

The equalizer prototype of the first method can be built using software models and tested using the impulse response of the targeted channels. The equalizer filter length is initially set to a minimum value and the equalizer performance is measured for increasing values of the filter length. Once the required steady-state MSE has been achieved or when there is no longer a gain in performance, the last value which resulted in a performance gain is selected for the filter length. This experimental approach has been the main method applied for choosing the equalizer filter length in this thesis. The rules of thumb methods that will be discussed for determining the equalizer length are the the two ray truncated channel inverse method of [75] and the multiple of the channel delay spread method. A third method for $T/2$-spaced equalizers was suggested by Tong *et al.* in [74]. This method stated that in the absence of channel noise and under a specific set of conditions, there is an exact solution for the equalizer length which is equivalent to the length of the channel.

The first rule of thumb considers a propagation channel that can be adequately modeled with two rays: a main signal and a scaled version that is delayed by τ seconds. The two ray channel model is defined as

$$\mathbf{c}(t) = \delta(t) + \alpha\delta(t-\tau) \tag{5.1}$$

where $\delta(t)$ is the Kronecker delta function which is '1' for $t = 0$ and '0' for $t \neq 0$. The desired equalizer is the channel inverse $\mathbf{c}^{-1}(t)$, which is given by

$$\mathbf{w}(t) = \delta(t) - \alpha\delta(t-\tau) + \alpha^2\delta(t-2\tau) - \alpha^3\delta(t-3\tau) + \ldots + (-1)^n\alpha^n\delta(t-n\tau) \tag{5.2}$$

where $\mathbf{w}(t)$ has an infinite number of terms as $n \rightarrow \infty$. For a filter with a finite number of taps, the unequalized terms of (5.2) contribute to the additive white Gaussian noise seen at the input of the equalizer. The energy of these unequalized terms is bounded by the first unequalized term as $|\alpha|^{Q+1}$, where Q is the index of the final equalized term. Given the input signal constellation and the magnitude of α, the minimum value of Q can be obtained, which leads to the minimum equalizer time length of $\tau_{\mathbf{w}} = Q_{\min}\tau$. The number of taps which correspond to $\tau_{\mathbf{w}}$ for a $T/2$-spaced equalizer is defined as

$$L = 2f_k Q_{\min}\tau \tag{5.3}$$

where f_k is the input signal sampling rate. In [75], the values of $Q_{\min}$ are given for various QPSK and QAM signals for different values of α.

The second rule of thumb determines the equalizer length as a multiple of the delay spread of the channel. Typically, the length of a $T/2$-spaced equalizer is chosen to be $3\sigma_\tau$ [75][76], where σ_τ is the RMS delay spread of the channel. According to [56], σ_τ is defined as

$$\sigma_\tau = \sqrt{\bar{\tau^2} + (\bar{\tau})^2} \tag{5.4}$$

where $\bar{\tau}$ and $\bar{\tau^2}$ are the first and second moments of the power delay profile (PDP), respectively. The PDP is the set of delayed input signals that corresponds to a single sent symbol, which begins at time $\tau = 0$ when the first version of the input signal is received. The first moment of the PDP is known as the mean excess delay and is defined as

$$\bar{\tau} = \frac{\sum_k P(\tau_k)\,\tau_k}{\sum_k P(\tau_k)} \tag{5.5}$$

where $P(\tau_k)$ is the input signal power at time τ_k. The second moment of the PDP is defined as

$$\bar{\tau^2} = \frac{\sum_k P(\tau_k)\,\tau_k^2}{\sum_k P(\tau_k)}. \tag{5.6}$$

Therefore, using (5.4), the equalizer length for a $T/2$-spaced equalizer is defined as

$$L = 3 \cdot 2 f_k \sigma_\tau \tag{5.7}$$

where f_k is the sampling frequency of the input signal.

5.3 Evaluation Metrics

The evaluation metrics that are defined in this section are extensively used to quantify the performance of adaptive filters and more specifically, adaptive equalizers.

The first two performance metrics that will be introduced quantify the maximum distortion (MD) and ISI. These measures indicated the proximity of the equalizer to the desired solution, where values near zero indicate near optimal solutions while values much greater than zero indicate poor solutions. The MD measure is defined as

$$\mathrm{MD}(n) = \frac{\sum |\mathbf{h}(n)| - \max |\mathbf{h}(n)|}{\max |\mathbf{h}(n)|} \tag{5.8}$$

where $\mathbf{h}(n)$ is the combined T-spaced channel-equalizer impulse response which was defined in (3.5). This metric is also known as the "peak distortion" [32]. The ISI measure is defined as

$$\mathrm{ISI}(n) = \frac{\sum |\mathbf{h}(n)|^2 - \max |\mathbf{h}(n)|^2}{\max |\mathbf{h}(n)|^2}. \tag{5.9}$$

where the ISI simply squares each of the terms in (5.8). The MSE is the most common method applied to quantify the performance of an adaptive filter. The MSE is defined as

$$\xi(n) = \frac{1}{N_R}\sum_{N_R} \left|y(n) - s(n-\delta)e^{j\theta}\right|^2 \approx \frac{1}{N_R}\sum_{N_R} |y(n) - \hat{s}(n)|^2 \tag{5.10}$$

where N_R is the total number of realizations, $s(n-\delta)$ is the transmitted symbol delayed by δ, and $e^{j\theta}$ is an arbitrary phase offset. After convergence, the MSE is equivalent to the squared error across the slicer as indicated at the right of (5.10). For accurate results[1], the number of realizations should be $N_R \geq 100$. However, using a large number of realizations to calculate the MSE is not always practical. An estimate of the MSE can be obtained by averaging the instantaneous squared error across the slicer of a single realization. The estimated MSE, denoted $\hat{\xi}$, is defined as

$$\hat{\xi}(n) = \frac{1}{N_B}\sum_{i=n}^{n+N_B} \left|y(i) - s(i-\delta)e^{j\theta}\right|^2 \approx \frac{1}{N_B}\sum_{i=n}^{n+N_B} |y(i) - \hat{s}(i)|^2 \tag{5.11}$$

where N_B is the size of the data block. Once again, after convergence, the estimated MSE is equivalent to the squared error across the slicer as indicated at the right of (5.11). As the block size is increased, the MSE curve becomes smoother while its accuracy decreases. The block size chosen is typically in the range $100 \leq N_B \leq 250$. This method is only appropriate as an initial performance estimate of an algorithm and is not suitable for analysis of the algorithms convergence, transient, and steady-state behaviors. A related measure is misadjustment, which is the ratio between the steady-state MSE, ξ_{ss}, and the MMSE, ξ_{min}. The misadjustment is defined as

$$M = \frac{\xi_{excess}}{\xi_{min}} = \frac{|\xi_{ss} - \xi_{min}|}{\xi_{min}} \tag{5.12}$$

where ξ_{excess} is the EMSE and ξ_{min} was defined in (3.11). Finally, the symbol error rate (SER) is the average number of symbol errors after convergence. The SER is defined as

$$\text{SER} = \frac{1}{N_S}\sum_{n} (s(n-\delta) \neq \hat{s}(n)) \tag{5.13}$$

where N_S is the number of samples while $\hat{s}(n)$ is the symbol estimate of the phase corrected equalizer output.

There are several qualitative methods that can be used to determine an equalizers performance. These include visual inspection of the equalizer output constellation after convergence, the equalizer output modulus, and eye-diagrams [32][54]. Both quantitative and qualitative methods will be considered to determine the equalizer performance.

[1]Quoted in discussions with Dr. Khaled Mayyas, who is an active researcher in the field of adaptive filtering.

Table 5.1: Microwave radio channel models used for simulations.

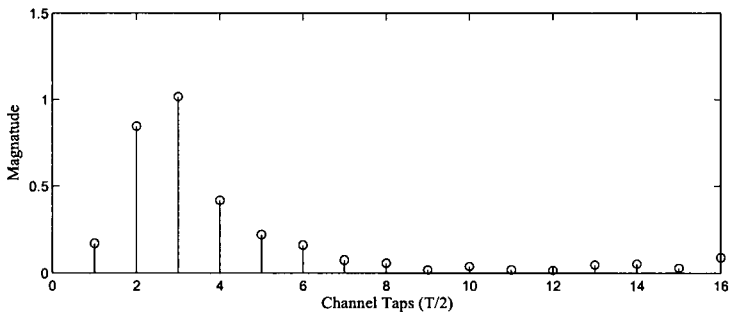
Channel	SPIB Database Designator	Taps	Frequency	MD
MRC-01	1	300	30 Mbaud	0.9418
MRC-02	2	230	22.5 Mbaud	0.7751
MRC-03	4	300	30 Mbaud	1.3698
MRC-04	5	206	22.5 Mbaud	1.3061
MRC-05	6	300	30 Mbaud	0.4114
MRC-06	8	300	30 Mbaud	0.5224
MRC-07	9	300	30 Mbaud	0.6090
MRC-08	10	300	30 Mbaud	0.9357
MRC-09	11	300	30 Mbaud	1.8595
MRC-10	12	227	22.5 Mbaud	0.9463
MRC-11	13	200	22.5 Mbaud	0.8077

5.4 Channel Models

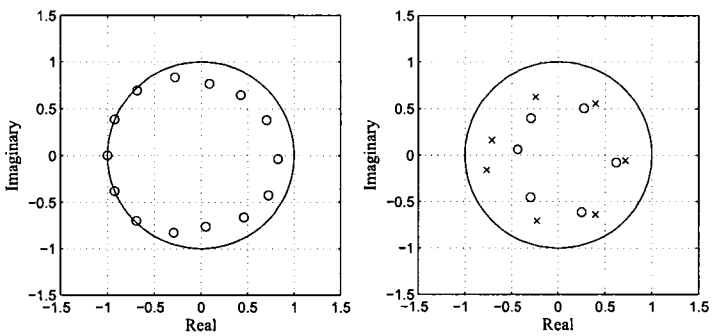
The channel models described in this section are empirically-derived $T/2$-spaced impulse responses of microwave radio and cable channels and Ricean fading channel model. A simulation model for both multitap Rayleigh and Ricean fading channels is developed based on the "sum of sinusoids" method of Jakes' simulator.

5.4.1 Microwave Radio Channel Models

The microwave radio channels considered are those models which reside at the signal processing information base (SPIB) at Rice University (refer to [11]). These models are the complex-valued baseband impulse response of empirically measured $T/2$-spaced microwave radio signals courtesy of Applied Signal Technologies, Sunnyvale, CA. They were extracted from the processing of field measurements of long QSPK and QAM sequences and are over 200 samples in length. In Table 5.1, the microwave channels used for simulations are listed along with their database designator, number of taps, baud frequency, and MD measure. The MD has been calculated using the odd samples of the $T/2$-spaced impulse response. The channel dynamics for a decimated version of MRC-02 are illustrated in Fig. 5.1. This version was derived by linear decimation of the FFT of the full length $T/2$-spaced impulse response of MRC-02. These channels are representative of digital terrestrial channels that are used for HDTV transmission.



(a) Magnitude of channel impulse response.



(b) $T/2$-spaced roots. (c) Subchannel roots.

Figure 5.1: Channel MRC-02 Dynamics.

Table 5.2: Cable channel models used for simulations.

Channel	SPIB Database Designator	Mean Distortion
CC-01	1	1.9927
CC-02	2	1.4370

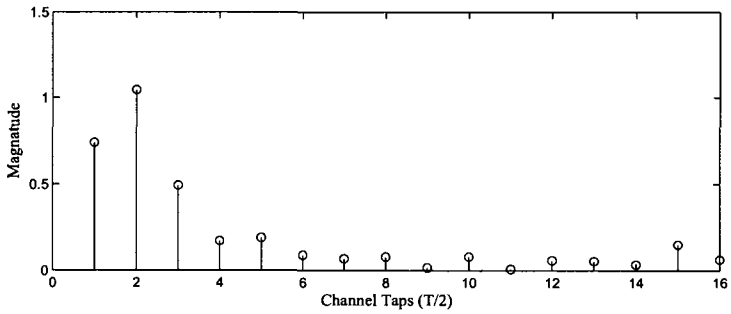
5.4.2 Cable Channel Models

The cable channels considered are the $T/2$-spaced cable channel models that reside at the signal processing information base (SPIB) at Rice university (refer to [11]). The baseband cable channel impulse responses are modelled by 128 complex taps that were extracted from received cable channel data streams. In Table 5.2, the cable channels used for simulations are listed along with their database designator and MD measure. The MD has been calculated using the odd samples of the $T/2$-spaced impulse response. The channel dynamics for a decimated version of CC-02 are illustrated in Fig. 5.2. This version was derived by linear decimation of the FFT of the full length $T/2$-spaced impulse response of CC-02. These channels are representative of the coaxial channels used for DVB-C and DOCSIS cable modems. These standards apply to both CATV and cable internet.

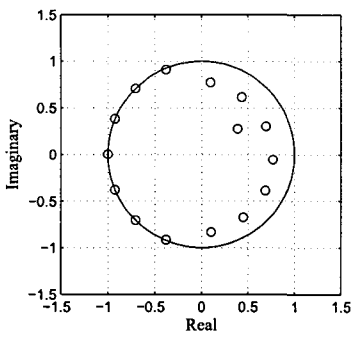
5.4.3 Fading Channel Models

The fading models for mobile radio considered are the multitap Rayleigh and Ricean models. Mobile radio channels experience fading due to multipath, which is the superposition of reflected and refracted versions of the transmitted signal at the receiver. When there is no line-of-sight (LoS) or specular component at the receiver, the channel envelope can be said to follow a Rayleigh fading distribution. This situation arises in urban areas where many obstacles exist that can prevent a direct path between the transmitter and receiver. Rayleigh fading channels can be modeled as the sum-of-sinusoids, such as Jakes' simulator [43]. The normalized low-pass fading process of Jakes' simulator is given by [43][89]:

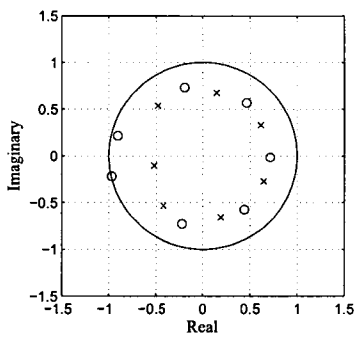
$$g(k) = \frac{2}{\sqrt{n}} \sum_{i=0}^{m} (a_i \cos(w_i k) + j b_i \cos(w_i k)) \tag{5.14}$$



(a) Magnitude of channel impulse response.



(b) $T/2$-spaced roots.



(c) Subchannel roots.

Figure 5.2: Channel CC-02 Dynamics.

where $g(k)$ is the fading channel, $n = 4m + 2$, and

$$a_i = \begin{cases} \sqrt{2}\cos(\beta_0), & i = 0 \\ 2\cos(\beta_i), & i = 1, 2, \ldots, m \end{cases} \tag{5.15}$$

$$b_i = \begin{cases} \sqrt{2}\sin(\beta_0), & i = 0 \\ 2\sin(\beta_i), & i = 1, 2, \ldots, m \end{cases} \tag{5.16}$$

$$\beta_i = \begin{cases} \pi/4, & i = 0 \\ \pi i/m, & i = 1, 2, \ldots, m \end{cases} \tag{5.17}$$

$$w_i = \begin{cases} w_d, & i = 0 \\ w_d \cos(2\pi i/n), & i = 1, 2, \ldots, m \end{cases} \tag{5.18}$$

where $w_d = 2\pi f_c v c^{-1}$ is the Doppler frequency, f_c is the carrier frequency, v is the velocity in m/s, and $c = 3 \times 10^8$ is the speed of light.

In situations where there is a LoS or specular component at the receiver, the channel envelope can be said to follow a Ricean fading distribution. This situation arises in microcellular and mobile satellite applications. The Ricean fading channel can be modeled as a Rayleigh fading component and stationary specular component scaled by a weighting factor called the rice factor. The rice factor K is the ratio of the specular power to scattered power, and is defined as [71]:

$$K = \frac{\alpha_g^2}{2\sigma_g^2} \tag{5.19}$$

where α_g is the complex stationary specular component and σ_g^2 is the variance of the Rayleigh distribution. The Ricean channel model in terms of K is defined as [71]:

$$h(k) = \frac{K}{K+1}\alpha_g + \frac{1}{K+1}g(k) \tag{5.20}$$

As illustrated in Fig. 5.3, eq. (5.20) can be modified for a multitap channel as follows:

$$\mathbf{h}(k) = \frac{1}{K+1}\left[K\alpha_g + g_0(k), g_1(k), \ldots, g_{L-1}(k)\right] \tag{5.21}$$

where there is a single specular component and L scattering components, where L is the length of the channel.

5.5 Simulation Studies

This section presents simulations studies and comparisons for blind equalization algorithms. The simulation environment consists of a $T/2$-spaced channel in cascade with a $T/2$-spaced equalizer,

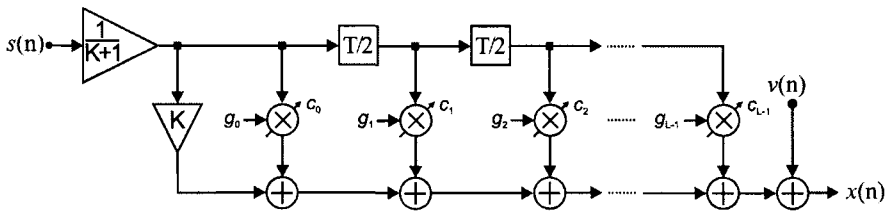


Figure 5.3: Fading channel model used for simulations (Rayleigh: $K = 0$, Ricean: $K \neq 0$).

where both the channel and equalizer are modelled as complex FIR filters. The source symbol sequence is randomly generated using an i.i.d. process and is drawn from a normalized QAM constellation. The received equalizer input samples are generated by convoluting the source sequence and channel impulse response and adding Gaussian noise, while the equalizer is updated at T-spaced intervals. Where appropriate, the output performance plots indicate the transmitted signal, SNR, channel model, filter type, and the number of realizations in the upper left or right corner.

5.5.1 Blind Equalization Algorithms

The simulations in this section are for the three fundamental blind equalization algorithms: GSA, CMA, and MMA, which were discussed in section 3.5. A constant step size of $\mu = 2^{-10}$ was applied to each algorithm and simulations were performed in channels CC-01 and MRC-07. Simulation results are illustrated in Fig. 5.4 for 16-QAM, 64-QAM, and 256-QAM. The rate of convergence and the convergence time for CMA and MMA are nearly identical. However, MMA consistently obtains a lower steady-state MSE than CMA. This is expected as QAM signals have a non-constant modulus and CMA penalizes the dispersion with respect to the squared output modulus. The rate of convergence and the convergence time of GSA is slower than that of CMA and MMA. The steady-state MSE of GSA is consistently lower than that of CMA, while it is consistently higher than that of MMA. The equalizer output signal constellations after convergence for GSA, CMA, and MMA, are illustrated in Fig. 5.5. These plots are commonly used to visually determine the equalizer's performance. As the equalizer output becomes "tighter" at symbol points, the lower the MSE. Consequently, the SER decreases since the reliability of the equalizer output increases. The equalizer output signal constellations for 64-QAM are similar for all algorithms and the proximity of the equalizer output to symbol points indicates that there will be low to moderate SER. For 256-QAM, once again the equalizer output signal constellations are similar for all algorithms. However, the proximity of the equalizer output to symbol points is more random, which indicates that the

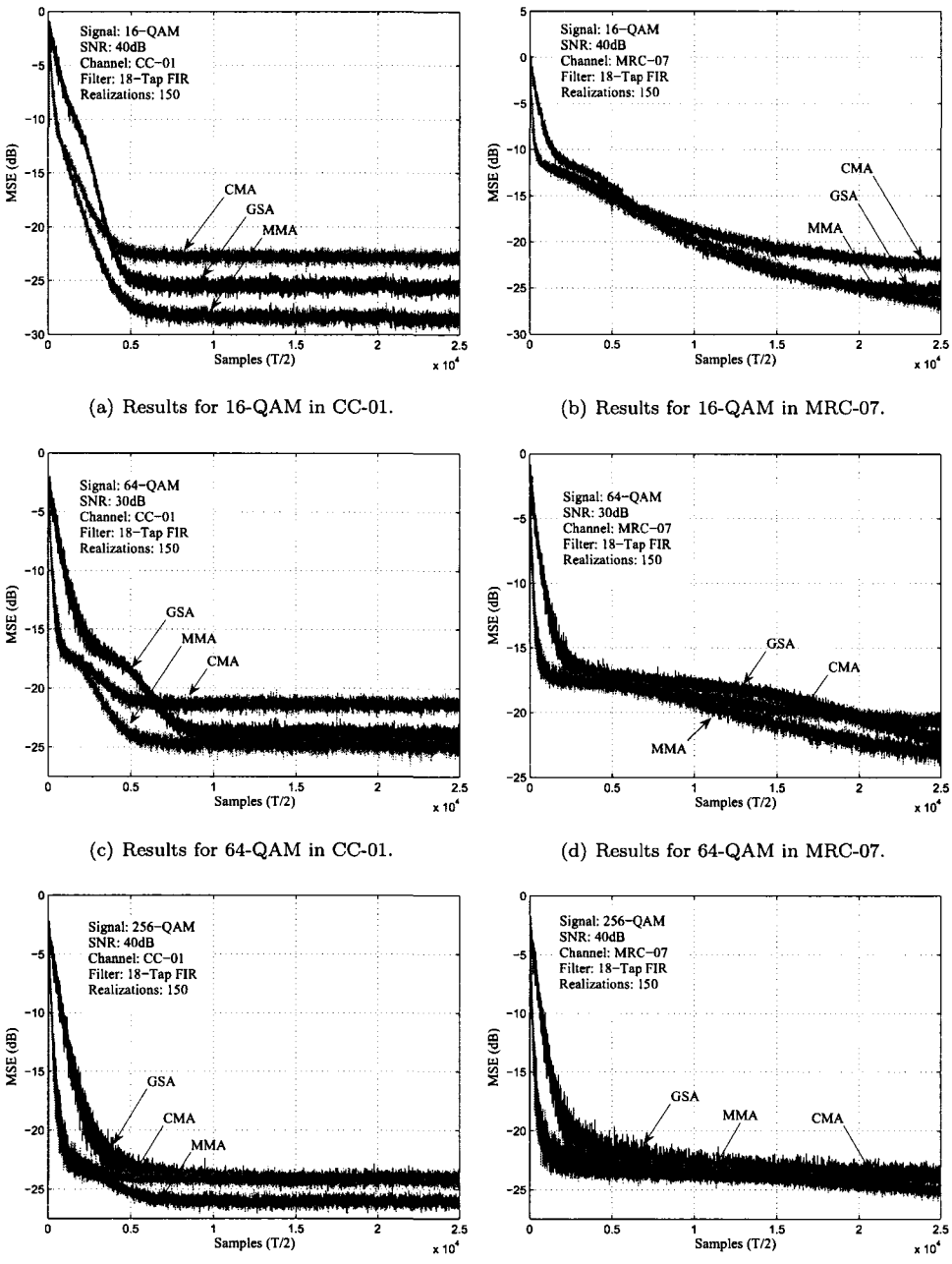
SER for 256-QAM will be much higher than that for 64-QAM. When analyzing the MSE curves in Fig. 5.4, the steady-state MSE does not vary that much for 64-QAM and 256-QAM in channel MRC-07. However, what does vary is the distance between symbol points, which is why the MSE level required for transfer to the DD algorithm is lower for higher-order signal constellations.

5.5.2 Radius-Adjusted Blind Equalization Algorithms

The simulations in this section are for RMMA and RMDA, which were discussed in section 4.3.1 and 4.3.2, respectively. Five decision regions were defined for RMMA and RMDA based on Table 4.2. The values of the weighting factor were set to $\lambda(n) = \{1, 1, 0, 0, 0\}$, while the values of the step size for both algorithms are set to $\mu(n) = \{2^{-7}, 2^{-8}, 2^{-9}, 2^{-9}, 2^{-10}\}$, $\mu(n) = \{2^{-8}, 2^{-9}, 2^{-10}, 2^{-10}, 2^{-11}\}$ and $\mu(n) = \{2^{-9}, 2^{-10}, 2^{-11}, 2^{-11}, 2^{-12}\}$ for 16-QAM, 64-QAM, and 256-QAM, respectively. The MSE curves for CC-01 and CC-02 are illustrated in Fig. 5.6 for RMMA and RMDA. The MSE curves for 16-QAM indicate that the transient and steady-state performance of RMMA and RMDA, for this signal, are similar. Both algorithms are able to achieve a fast convergence time and low steady-state MSE. For 64-QAM and 256-QAM, the initial rate of convergence is equivalent, however, the rate of convergence for RMDA drops off as the DD error term becomes the primary error signal. The steady-state MSE for 16-QAM and 64-QAM are similar for both algorithms, while the steady-state MSE for 256-QAM is lower for RMMA. The equalizer output signal constellations after convergence for RMMA and RMDA, are illustrated in Fig. 5.7. As expected, equalizer output signal constellations for 16-QAM and 64-QAM are similar for both algorithms and the proximity of the equalizer output to symbol points indicates that there will be a low SER. For 256-QAM, the output signal constellation of RMMA is much tighter than that for RMDA. The output constellation for RMMA indicates that there will be a low to moderate SER, while the output constellation for RMDA indicates that there will be a moderate SER.

5.5.3 Simulated Comparisons

This section presents comparative studies and quantitative results for RMMA, RMDA, and the selective update method. The RMMA and RMDA are compared with MCMA and CMA+SDD, while the selective update method is compared with the signed-error (SE), power-of-two (POT) error, and partial coefficient update (PU) methods. Simulations were carried out with microwave radio and Ricean fading channels. Quantitative results were averaged over a number of microwave radio channels for convergence time (TTC), steady-state MSE, misadjustment (M), and update percentage. The convergence time is calculated as the number of $T/2$-spaced samples needed to



(a) Results for 16-QAM in CC-01.

(b) Results for 16-QAM in MRC-07.

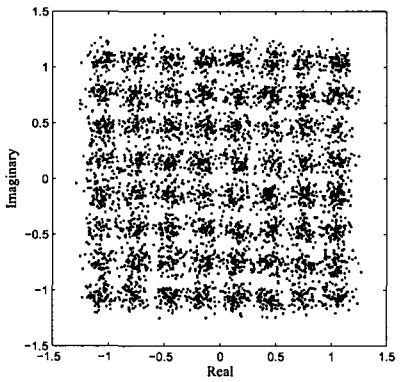
(c) Results for 64-QAM in CC-01.

(d) Results for 64-QAM in MRC-07.

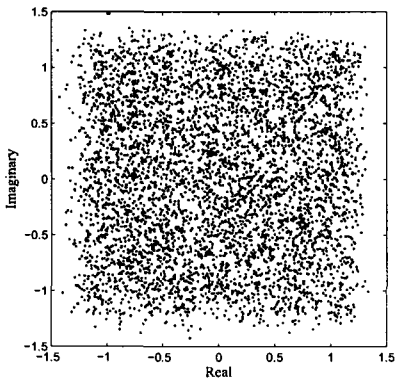
(e) Results for 256-QAM in CC-01.

(f) Results for 256-QAM in MRC-07.

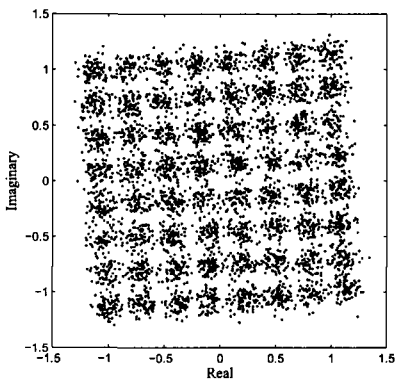
Figure 5.4: Simulation of the blind equalization algorithms: GSA, CMA, and MMA.



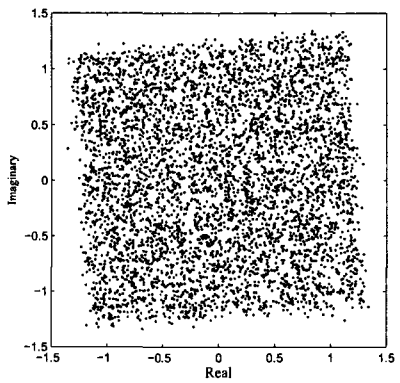
(a) GSA results for 64-QAM in MRC-07.



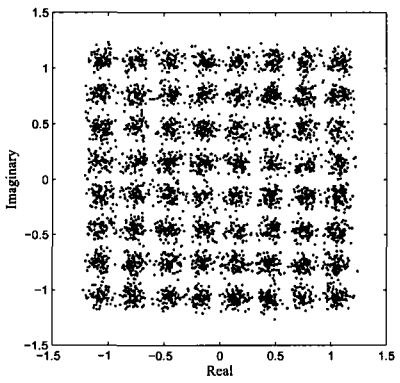
(b) GSA results for 256-QAM in MRC-07.



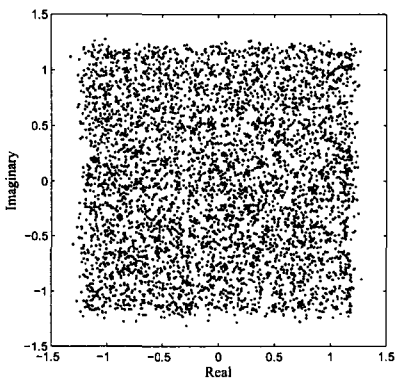
(c) CMA results for 64-QAM in MRC-07.



(d) CMA results for 256-QAM in MRC-07.

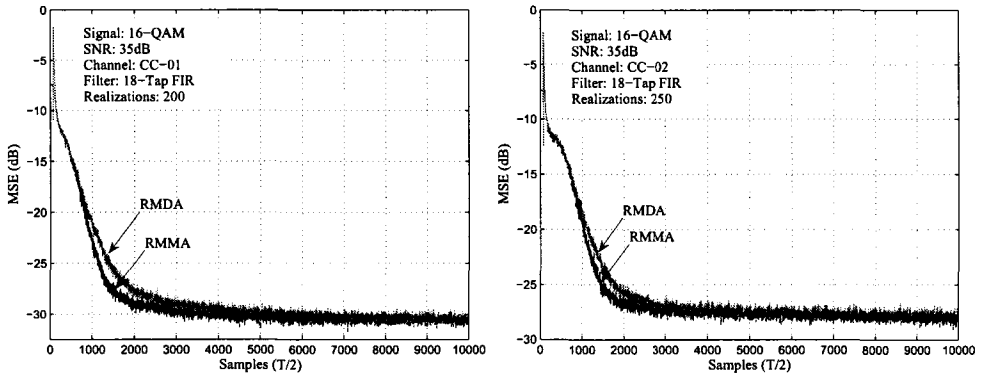


(e) MMA results for 64-QAM in MRC-07.



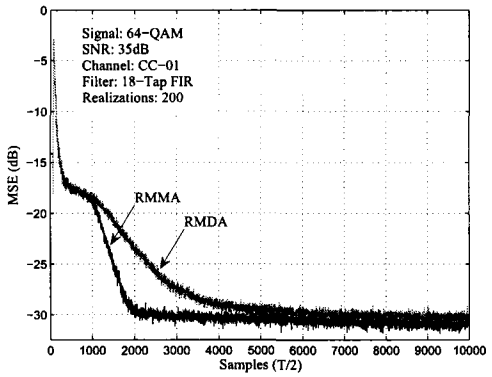
(f) MMA results for 256-QAM in MRC-07.

Figure 5.5: Output signal constellations for GSA, CMA, and MMA.

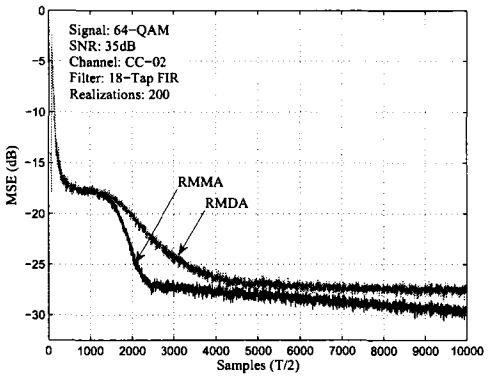


(a) Results for 16-QAM in CC-01.

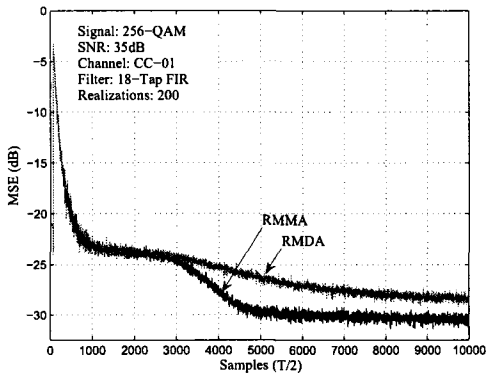
(b) Results for 16-QAM in CC-02.



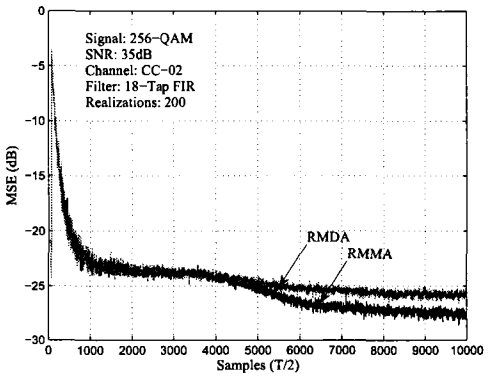
(c) Results for 64-QAM in CC-01.



(d) Results for 64-QAM in CC-02.

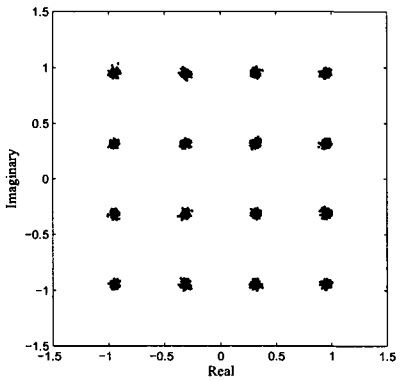


(e) Results for 256-QAM in CC-01.

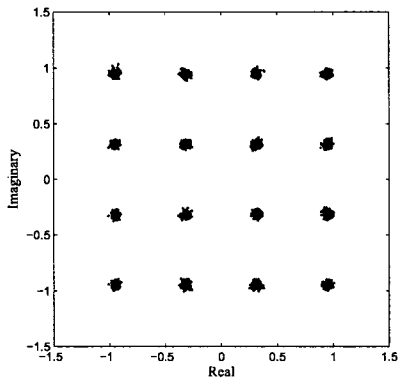


(f) Results for 256-QAM in CC-02.

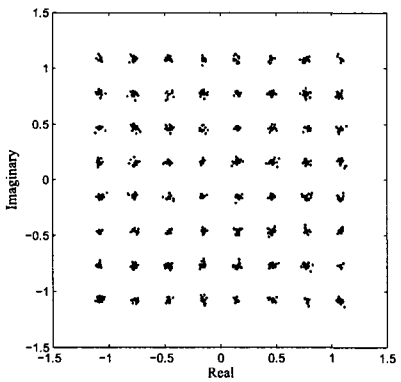
Figure 5.6: Results for RMMA and RMDA in cable channels.



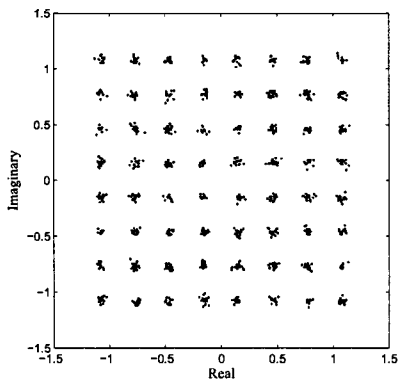
(a) RMMA results for 16-QAM in CC-01.



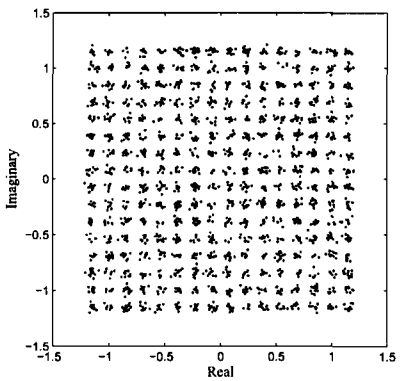
(b) RMDA results for 16-QAM in CC-01.



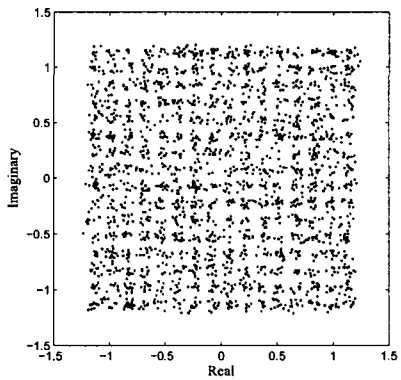
(c) RMMA results for 64-QAM in CC-01.



(d) RMDA results for 64-QAM in CC-01.



(e) RMMA results for 256-QAM in CC-01.



(f) RMDA results for 256-QAM in CC-01.

Figure 5.7: Output signal constellations for RMMA and RMDA.

Table 5.3: Simulation parameters for RMMA, RMDA, RMDA, and CMA+SDD.

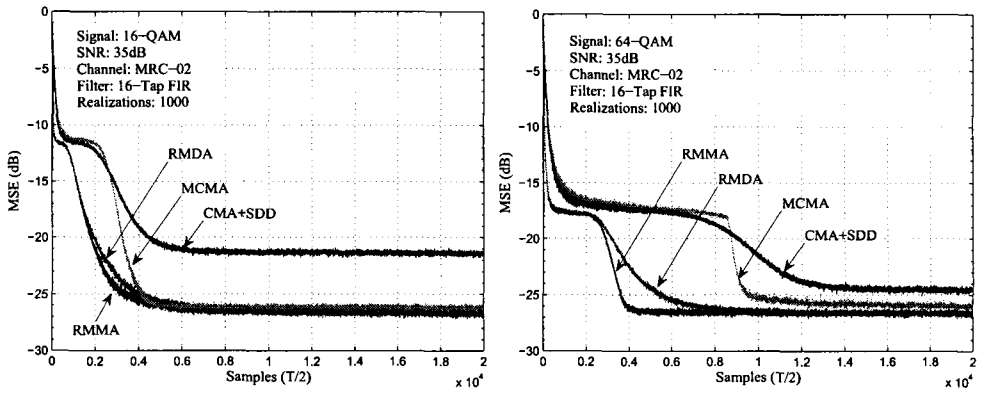
Method	p/ρ	β	μ_M/μ	$\alpha_{\max}$	λ_3
RMMA	1	$0.9\lvert\beta\rvert_{\max}$	$\{10^{-3}, 6\cdot10^{-4}\}$	$\{10, 6.5\}$	0.1
MCMA	1	$0.9\lvert\beta\rvert_{\max}$	$\{10^{-3}, 6\cdot10^{-4}\}$	–	–
RMDA	–	–	$\{10^{-3}, 6\cdot10^{-4}\}$	$\{10, 6.5\}$	0.1
CMA+SDD	0.6	–	$\{10^{-3}, 6\cdot10^{-4}\}$	–	–

Table 5.4: Quantitative results for RMMA, RMDA, MCMA, and RMMA.

	16-QAM			64-QAM		
Method	MSE (dB)	M	TTC (T/2)	MSE (dB)	M	TTC (T/2)
RMMA	-27.6890	0.0767	2644	-27.7202	0.0759	3025
RMDA	-27.2955	0.0899	4012	-27.3387	0.0891	4798
MCMA	-27.1381	0.0975	4823	-25.1393	0.1522	6580
CMA+SDD	-22.2613	0.2486	5905	-23.6271	0.2026	7300

reach 90% of the steady-state MSE. The minimum number of realizations were 300 for microwave radio channels and 150 for Ricean fading channels.

As mentioned in the introduction, the RMMA and RMDA are compared with MCMA and CMA+SDD, which are related hybrid blind equalization algorithms that were discussed in sections 3.6.3 and 3.6.6, respectfully. The simulations for these algorithms were conducted over microwave radio channels using the parameters listed in Table 5.3, where the regions for RMMA and RMDA are based on Table 4.2 using five decision regions. The MSE curves for MRC-02 are illustrated in Fig. 5.8 for 16-QAM and 64-QAM, respectively, which are representative of the typical results obtained. The MSE curves for this channel demonstrate that RMMA and RMDA achieve a faster convergence time and lower or equivalent steady-state MSE. The MSE curves of MCMA for 16-QAM and 64-QAM demonstrate MCMA's dependence on exact carrier recovery. MCMA is a CMA-based algorithm and converges with an arbitrary phase offset. The CME term in MCMA is phase sensitive and will initially contribute negatively until the carrier recovery circuit obtains the correct phase. However, the RMMA is a MMA-based algorithm and converges with a phase offset modulo 90^0. In addition, at the onset of equalization, the RMMA will update the equalizer taps predominantly with MMA error since $E\{\lambda(n)\} \approx 1$. Therefore, the CME term in RMMA will not contribute negatively, which improves the rate of convergence for RMMA. The carrier-recovery method applied to MCMA is the combined "four-corners" [17] and DD carrier tracking technique [53] that was proposed for MCMA in [41]. To prevent divergence and excess convergence time, the CME term in the MCMA error signal was included after a delay, which varied by channel and the constellation. For channel MRC-02, this delay corresponded to 1500 and 8500 samples for 16-QAM and 64-QAM, respectively. Quantitative results have been averaged over channels MRC-01→MRC-11 and are listed in Table



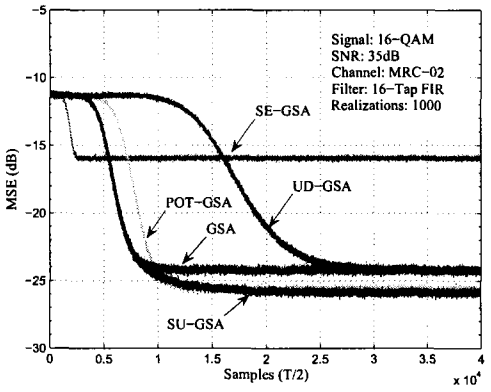
(a) Results for 16-QAM in MRC-02.

(b) Results for 64-QAM in MRC-02.

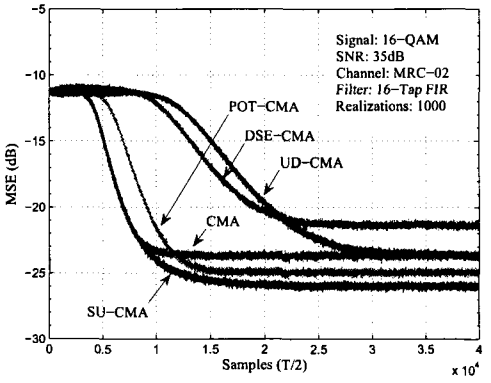
Figure 5.8: Comparison of RMMA and RMDA with MCMA and CMA+SDD.

5.4. These results have been published in [8]. The results indicate that, on average, both RMMA and RMDA achieve faster convergence time, lower steady-state MSE, and lower misadjustment than MCMA and CMA+SDD.

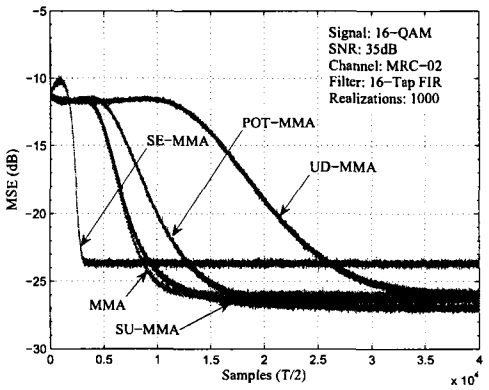
The selective update method is compared with the signed-error, power-of-two error, and partial coefficient update methods, which were discussed in sections 3.7.1, 3.7.2, and 3.7.3, respectfully. The simulations for these algorithms were conducted over microwave radio and Ricean fading channels. The selective update, signed-error, and power-of-two error methods were applied to GSA, CMA, and MMA for microwave radio simulations. The MSE curves for MRC-02 are illustrated in Fig. 5.8 for GSA-, CMA-, and MMA-based algorithms, respectively, which are representative of the typical results obtained. The modified algorithms are compared with the base algorithm and an update decimated (UD) version of the base algorithm, which is decimated by a factor of three. A constant step size of 2^{-10} was applied to all algorithms except for DSE-CMA, which required a step size of 2^{-11} in order to avoid divergence. The parameter α_S was chosen according to (4.40) using an ad hoc manor and ranged from $1/4 \rightarrow 1/6$, while the power-of-two error algorithms were quantized according to (3.56) using a word-length of 16-bits and $\tau = 0$. These curves indicate that algorithms modified by the selective update method maintain their transient characteristics, while they can achieve similar steady-state MSE. The convergence time of power-of-two error algorithms is slightly longer than the base algorithm, while the convergence time of update decimated algorithms, as expected, is three times that of the base algorithm. The signed-error algorithms, SE-GSA and SE-MMA, achieve the fastest convergence time, however, they also have the highest steady-state MSE. Quantitative results



(a) Results for GSA in MRC-02.



(b) Results for CMA in MRC-02.



(c) Results for MMA in MRC-02.

Figure 5.9: Comparison of computationally-efficient methods for GSA, CMA, and MMA.

averaged for channels MRC-01→MRC-08 are listed in Table 5.5. These results, with exception to those for GSA, have been published in [9]. It can be seen from Table 5.5 that the percentage of tap updates for the selective updated based algorithms prior to convergence is high, while the percentage after convergence is substantially reduced. The results indicate that the selective update algorithms, on average, achieve lower steady-state MSE and have a slightly longer convergence time than the base algorithms. In comparison with the update decimated algorithms, the selective update algorithms are able to maintain the transient performance and lower the misadjustment of the original algorithms, while reducing the number of tap updates in steady-state to under 15% on average.

Comparisons for the selective update, partial coefficient update, and power-of-two error methods

Table 5.5: Quantitative results for computationally-efficient methods for GSA, CMA, and MMA.

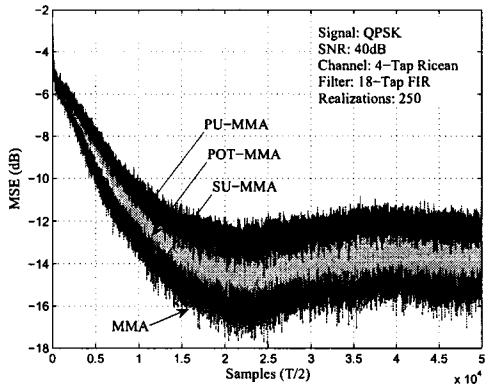
	Performance Measures			Tap Update Percentage	
Method	MSE	M	TTC (T/2)	Transient (%)	Steady-State (%)
GSA	-24.4299	0.1877	8524	100	100
SE-GSA	-15.9170	0.4686	3518	100	100
POT-GSA	-25.5981	0.1488	10596	100	100
UD-GSA	-24.0065	0.2013	24445	33.33	33.33
SU-GSA	-25.4434	0.1539	9448	83.77	33.27
CMA	-23.8834	0.2055	8587	100.00	100.00
DSE-CMA	-21.1884	0.2939	17892	100.00	100.00
POT-CMA	-25.3812	0.1573	12902	100.00	100.00
UD-CMA	-23.7006	0.2115	24525	33.33	33.33
SU-CMA	-26.1220	0.1324	10953	73.05	14.95
MMA	-26.4679	0.1218	10506	100.00	100.00
SE-MMA	-24.0006	0.2016	2530	100.00	100.00
POT-MMA	-27.0522	0.1039	17250	100.00	100.00
UD-MMA	-26.0838	0.1346	29880	33.33	33.33
SU-MMA	-27.5051	0.0888	11933	75.65	14.43

were also carried out for Ricean fading channels. The Ricean channel was modeled as Fig. 5.3, where the values of K varied for 2.5dB→7.5dB. The equalizer input signal was generated by convolving the random source symbol sequence with the Ricean channel model using the "overlap and add" sectioned convolution method with a block size of four. Two sets of simulations were carried out, the first according to DVB-S specifications for QPSK and the second according to the ATSC standard for HDTV except using 16-QAM instead of 8-VSB modulation. The carrier frequency for DVB-S is 11GHz and the symbol rate was set to 22.5MHz. For ATSC, the carrier frequency is 859.25MHz and the symbol rate was set to 10.76MHz. The respective simulations were carried out at low Doppler frequencies using a velocity of 10km/h and are illustrated in Fig. 5.10 for MMA-based algorithms. A step size of 2^{-9} was applied to all algorithms. The parameter α_S was chosen according to (4.40) using an ad hoc manor and was set to 1/10 and 1/6 for the DVB-S and ATSC standards, respectively. The variable partial coefficient update algorithm adjusted the largest $P(n)$ taps that corresponded to 75% of the regressor power, while the power-of-two error algorithm was quantized according to (3.56) using a word-length of 16-bits and $\tau = 0$. The MSE curves indicate that the transient response of SU-MMA is equivalent to that of MMA for K=2.5dB and K=5dB, while both algorithms obtain similar steady-state MSE. For K=7.5dB, SU-MMA obtains lower steady-state MSE for the ATSC standard, while MMA obtains lower steady-state MSE for the DVB-S standard. The POT-MMA experiences a delayed transient response with respect to MMA and obtains a higher steady-state MSE. Updating the $P(n)$ largest coefficients that correspond to 75% of the regressor

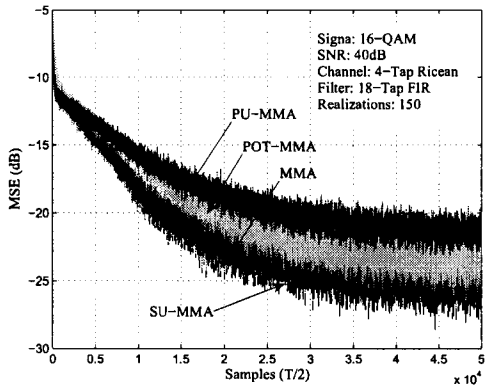
Table 5.6: Equalizer tap update percentage for SU-MMA in Ricean channel.

Standard	α_S	K=2.5dB	K=5dB	K=7.5dB
DVB-S	1/16	64.97%	57.59%	39.70%
ATSC	1/10	51.65%	46.59%	31.15%

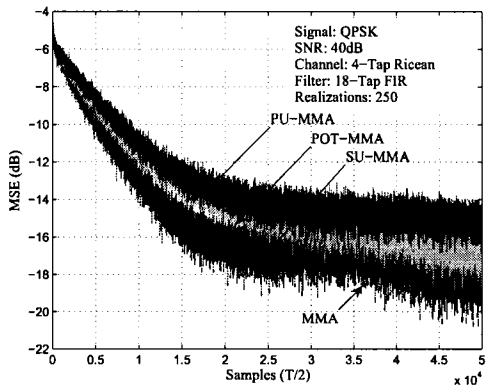
power, the PU-MMA has the slowest convergence rate and the highest steady-state MSE. In Table 5.6, the update percentage of PU-MMA is listed for the values of K. As expected, the update percentage of SU-MMA is highest for K=2.5dB at 64.97% and 51.65% for the DVB-S and ATSC standards, respectively, while it is lowest for K=7.5dB at 39.70% and 31.15% for the DVB-S and ATSC standards, respectively. This indicates that the selective update method can maintain the transient and steady-state performance of the base algorithm, while continuing to obtain substantial reductions in the number of tap updates.



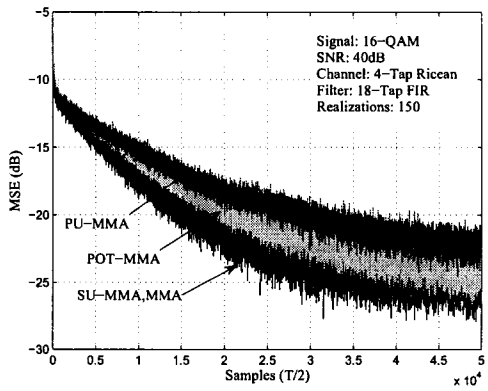
(a) Results for QPSK (K=2.5dB & $f_c = 11$GHz).



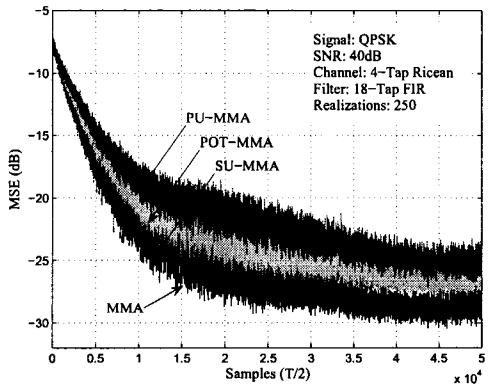
(b) Results for 16-QAM (K=2.5dB & $f_c = 859.25$MHz).



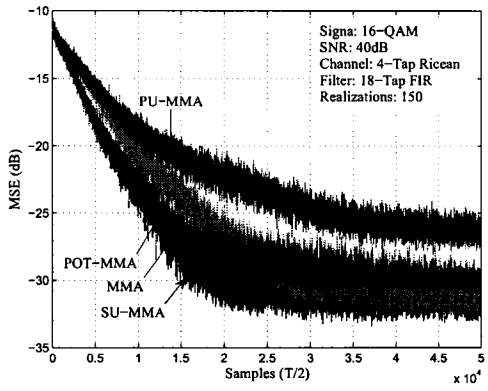
(c) Results for QPSK (K=5dB & $f_c = 11$GHz).



(d) Results for 16-QAM (K=5dB & $f_c = 859.25$MHz).



(e) Results for QPSK (K=7.5dB & $f_c = 11$GHz).



(f) Results for 16-QAM (K=7.5dB & $f_c = 859.25$MHz).

Figure 5.10: Comparison of computationally-efficient methods for MMA in Ricean fading channels.

Chapter 6

FPGA Implementation of a Blind Adaptive Equalizer

6.1 Introduction

This chapter discusses in detail the custom implementation of a complex 18-tap $T/2$-spaced blind adaptive equalizer intellectual property (IP) core for QAM signals. The IP core is implemented on the Altera Stratix II EP2S130F780C4 FPGA and targets cable demodulators. The design is configurable and can implement four different error signals including those of RMMA and RMDA, while it can equalize QPSK, 16-QAM, 64-QAM, and 256-QAM signals.

The chapter begins with the basic parameters and goals of the IP core implementation and continues with fixed-point simulation results. An overview of the IP core is presented that decomposes the blind equalizer into four main modules: complex tap, complex tap update, slicer, and error signal modules. The structure and operation of each of these modules is discussed in detail. The implementation results for the Stratix II EP2S130F780C4 FPGA are analyzed, which include those for RTL simulation, logic synthesis, physical synthesis and gate-level and timing simulation. Finally, the implementation is compared to recent QAM equalizer implementations for cable demodulators.

6.2 Complex Blind Equalizer Design and Objectives

The IP core is a linear complex 18-tap $T/2$-spaced blind equalizer, which is realized using the direct form complex equalizer structure. As illustrated in Fig. 6.1, the direct form complex equalizer structure consists of four real finite impulse response (FIR) filters. The IP core operates at twice the symbol frequency and uses two clocks: `clk` and `clk_baud`, where `clk` operates at twice the symbol frequency and `clk_baud` operates at the symbol frequency. The equalizer output and tap coefficients are updated once per symbol period on the rising edge of `clk_baud`. This factor is exploited by utilizing a custom FIR filter structure, where all the *odd taps* are evaluated during the first cycle of clk (i.e. when `clk_baud='1'`) and all the *even taps* are evaluated during the second cycle of `clk` (i.e. when `clk_baud='0'`). The same hardware is used to compute the odd and even taps, thereby, reducing the number of multipliers by a factor of two for the tap coefficient and update portions of the equalizer. The equalizer is configurable and can achieve tap coefficient adaptation by means of the CMA, MMA, RMMA, and RMDA error signals, while it can equalize QSPK, 16-QAM, 64-QAM, and 256-QAM signals. There are two pipelining stages: the first stage is after the equalizer output is generated, while the second stage follows the error signal calculation. This allows concurrent operation of the equalizer output, error signal, and tap coefficient adjustment. The primary goals of this implementation are:

1. To design a generic IP core of an adaptive equalizer for QAM signals and implement this core on an Altera Stratix II FPGA.

2. To achieve channel equalization without the aid of a training sequence for QPSK, 16-QAM, 64-QAM, and 256-QAM signals.

3. To implement multiple error signals for equalizer tap adaptation, such that the error signal can be configured to implement the CMA, MMA, RMMA, or RMDA error signals.

4. To target the IP core for QAM demodulators that are used in cable and microwave radio applications and to achieve data rates comparable to those of recent designs.

The IP core was taken from concept to implementation using the standard RTL design flow, which was discussed in section 2.2. Algorithmic development and analysis for CMA, MMA, RMMA and RMDA was performed in Matlab for cable and microwave radio channel models. These algorithms were converted to fixed-point using the fixed-point toolbox in Matlab, where the word-length (WL) was determined by extensive fixed-point simulations. The RTL model of the equalizer was created by modifying the direct form complex equalizer architecture and using the fixed-point models

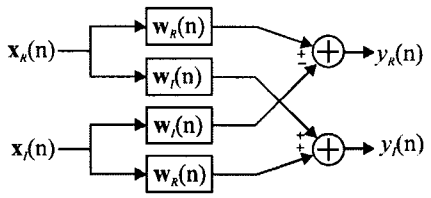


Figure 6.1: Direct form complex equalizer implementation

obtained for the target algorithms. The process of hardware synthesis and verification and consists of RTL simulation, logic synthesis, physical synthesis, and gate-level and timing simulation. Cadence NC-VHDL is utilized for both RTL and gate-level simulation, while logic synthesis is performed using Synopsys Design Compiler. The physical synthesis process and initial timing analysis is performed using Altera Quartus II. The sections to follow discuss these steps in detail and present implementation results.

6.3 Fixed-Point Simulations

The word-length and fractional word-length (FWL) for the fixed-point models of CMA, MMA, RMMA, and RMDA, were determined by fixed-point simulations that were completed using the fixed-point toolbox of Matlab. The number system chosen for implementation is the two's complement number system, which has a numeric range of $\left(-2^{\mathrm{IWL}-1}, 2^{\mathrm{IWL}-1} - 2^{-\mathrm{FWL}}\right)$ and a resolution of $2^{-\mathrm{FWL}}$, where the integer word-length is IWL = WL − FWL. Saturation is applied to handle overflow conditions, while truncation is applied instead of rounding. When selecting the initial WL and FWL values, it is important to consider the relationship between the equalizer input, tap coefficients, error signal, and output. If a large step size is applied to the error signal, a significant portion of the error signal will be truncated. Therefore, a small WL can be selected for the error signal without adversely effects. The decisive factor in the overall performance of the equalizer is the WL of the tap coefficients, which is typically chosen to be the largest WL. The tap coefficient WL is dependent upon the input WL and output WL, where the tap coefficient WL can be reduced based on the output WL. In general, the IWL affects the transient performance of the equalizer, while the FWL affects the steady-state performance.

The input and output WL and FWL of the IP core are set to 16-bit and 14-bit, respectively, since 16-bit is a standard WL size. The error signal WL and FWL is held constant during fixed-point simulations at 16-bit and 14-bit, respectively. Fixed-point simulations are conducted with

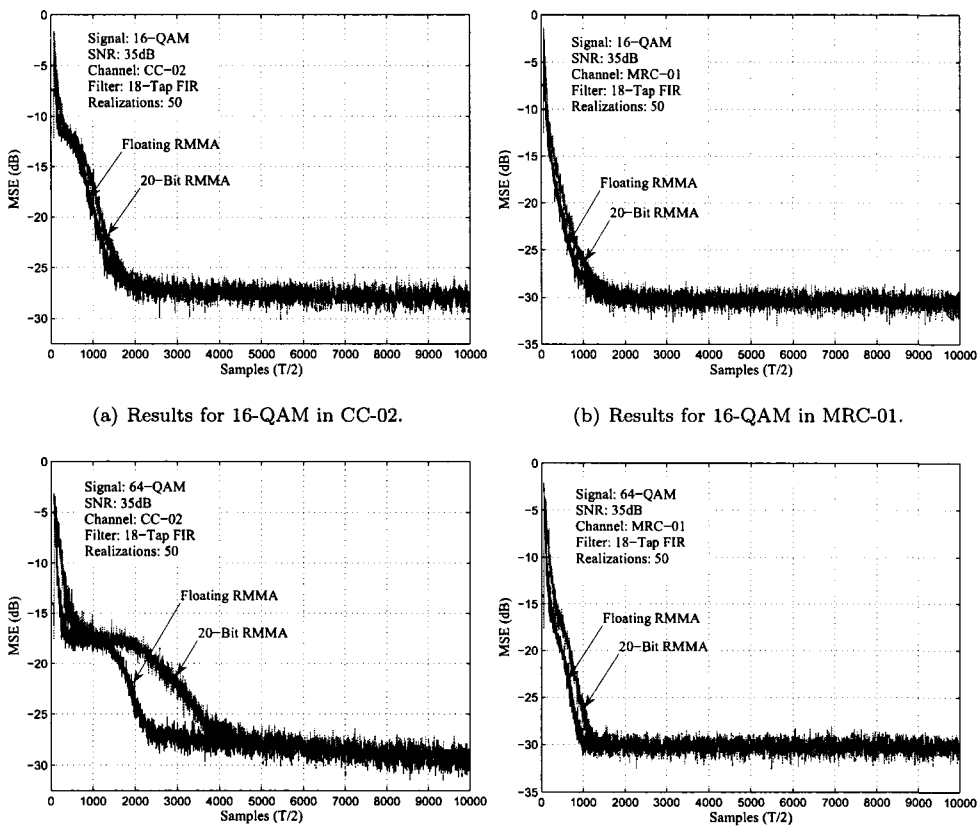
three different sets of WLs and FWLs for the tap coefficients, which are (18,16)-bit, (20,18)-bit and (22,20)-bit, respectively. There is a significant performance improvement between WLs of 18-bit and 20-bit for the tap coefficients. However, the performance difference between 20-bit and 22-bit WLs for the tap coefficients is marginal. Therefore, the tap coefficient WL and FWL for this implementation are 20-bit and 18-bit, respectively.

There are several additional considerations for the physical implementation of RMMA and RMDA, which affect their fixed-point simulations and WL selections. As mentioned in section 4.3 for both RMMA and RMDA, $r^2(n)$ can be used instead of $r(n)$ to determine the appropriate region if the precision is sufficient. This eliminates the need to compute the square root function. Initially, $r^2(n)$ was calculated using a WL and FWL of 16-bit and 14-bit, respectively. This, however, adversely affected the performance and subsequently, the WL and FWL were increased to 20-bit and 18-bit respectively, which provides acceptable performance. For the RMMA error signal, the CME component is implemented using a LUT. The size of this LUT is determined by the precision of the input, which is set to a WL and FWL of 10-bit and 8-bit, respectively. This results in a precision of $2^{-8} \approx 0.004$ which is sufficient for the CME error signal.

Fixed-point simulations are illustrated for RMMA and RMDA in Fig. 6.2 and Fig. 6.3, respectively. The fixed-point simulations for RMMA were conducted using channels CC-02 and MRC-01, while the fixed-point simulations for RMDA were conducted using channels CC-01 and MRC-04. In all simulations, the results of the fixed-point algorithm is compared with that of the floating point algorithm. In general, the transient performance of the fixed-point algorithm is very similar to that of the floating-point algorithm. The only exception was RMMA in channel CC-02 for 64-QAM illustrated in Fig. 6.2(c), which has a longer convergence time. The steady-state performance of the fixed-point algorithm is nearly identical to that of the floating-point algorithm for all simulations.

6.4 Complex Blind Equalizer Implementation

The IP core implementation is a complex 18-tap $T/2$-spaced blind adaptive equalizer for QAM signals. A block diagram of the IP core is illustrated in Fig. 6.4. The input signals are `x_real(15:0)`, `x_imag(15:0)`, `const_sel(1:0)`, `e_sel(1:0)`, `clk`, `clk_baud`, and `rs`, while the output signals are `y_real(15:0)`, `y_imag(15:0)`, `a_est(15:0)` and `b_est(15:0)`. The signals `x_real` and `x_imag` are the real and imaginary equalizer input samples, respectively, while `const_sel` and `e_sel` specify the type of input signal and error signal, respectively. The output signals `y_real` and `y_imag` are the real and imaginary samples of the equalizer output, while `a_est` and `b_est` are the real and



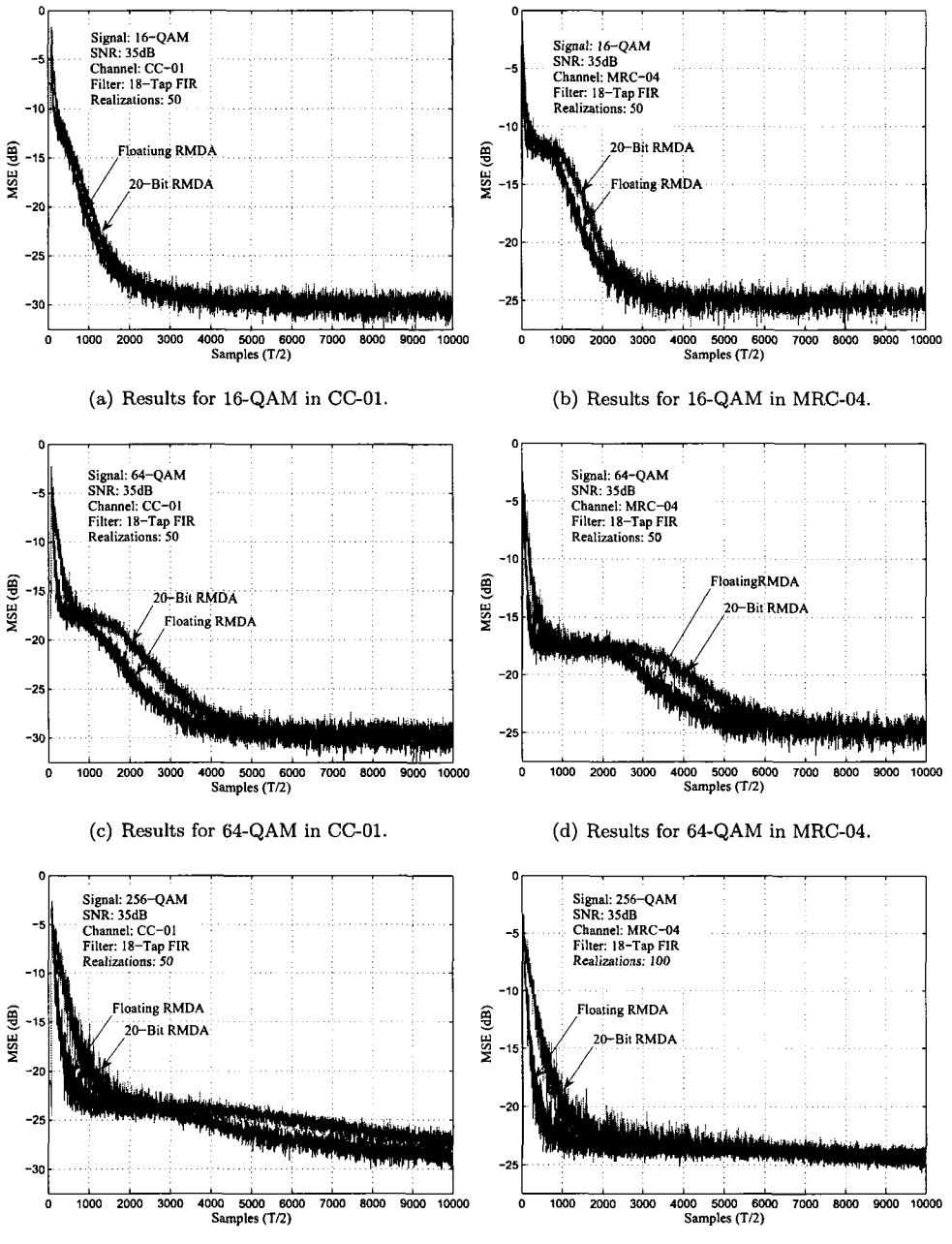
(a) Results for 16-QAM in CC-02.

(b) Results for 16-QAM in MRC-01.

(c) Results for 64-QAM in CC-02.

(d) Results for 64-QAM in MRC-01.

Figure 6.2: Fixed-point simulations for RMMA.



(a) Results for 16-QAM in CC-01.

(b) Results for 16-QAM in MRC-04.

(c) Results for 64-QAM in CC-01.

(d) Results for 64-QAM in MRC-04.

(e) Results for 256-QAM in CC-01.

(f) Results for 256-QAM in MRC-04.

Figure 6.3: Fixed-point simulations for RMDA.

imaginary symbol estimates, respectively. The symbol estimates are solely for equalizer testing purposes and would not be implemented in the final design of the IP core since the estimates are not for phase corrected signals. The IP core is sectioned into four main modules: complex tap, complex tap update, slicer, and error signal. There are nine complex tap and complex tap update modules instantiated, which correspond to 18 complex tap coefficients, while one each of the the error signal and slicer modules are instantiated. The equalizer input signals are stored in two tapped delay lines (TDL), one each for the real and imaginary samples. The TDL consist of 20 16-bit registers each, where the final four 16-bit registers are the result of the delayed adaptation due to pipelining. T-spaced registers are inserted after the equalizer output and error signal calculation, which allows the equalizer output, error signal, and tap adaptation calculations to occur simultaneously. There are four nine-operand 20-bit accumulators (ACC), which realize the outputs of the four real filters. These accumulators are reset to zero on the positive edge of `clk_baud`. An internal slicer is utilized for the error signal calculation since symbol estimates are required to implement the DD error signal and to determine the region for RMMA and RMDA. These estimates are for the equalizer output prior to carrier-recovery. While this does not affect the error signal calculation, carrier-recovery would be required if the estimates were used to determine the decided symbol. For this reason, the symbol estimate outputs are only for equalizer testing purposes. The IP core can be adjusted to use an external slicer by adding input ports for the decided symbols and extending the TDLs to account for the increased adaptation delay. The initialization process blocks the equalizer output and delays adaptation until there are enough input samples in the TDL. The output of the equalizer is that from a 2×1 MUX, which is connected to the delayed equalizer output and a ground signal and whose output is determined by the initialization process. After a `rs='1'` signal, the MUX output is the ground signal until 22nd cycle of `clk`.

The IP core is described in hardware using VHDL and the IEEE std_logic_1164 and numeric_std libraries. The hierarchy of VHDL design files for the IP core is illustrated in Fig. 6.5, where `blind_equalizer.vhd` is the top module. This top module of the IP core instantiates the complex tap, complex tap update, slicer and error signal modules, which are described in `complex_tap.vhd`, `complex_tap_update.vhd`, `slicer.vhd`, and `error_signal.vhd`, respectively. The error signal module instantiates the CMA, MMA and CME error modules, which are described in `cma_error.vhd`, `mma_error.vhd`, and `cme_error.vhd`, respectively. The basic arithmetic and logic operations that are utilized by all modules are described by `sign_logic_arith.vhd`.

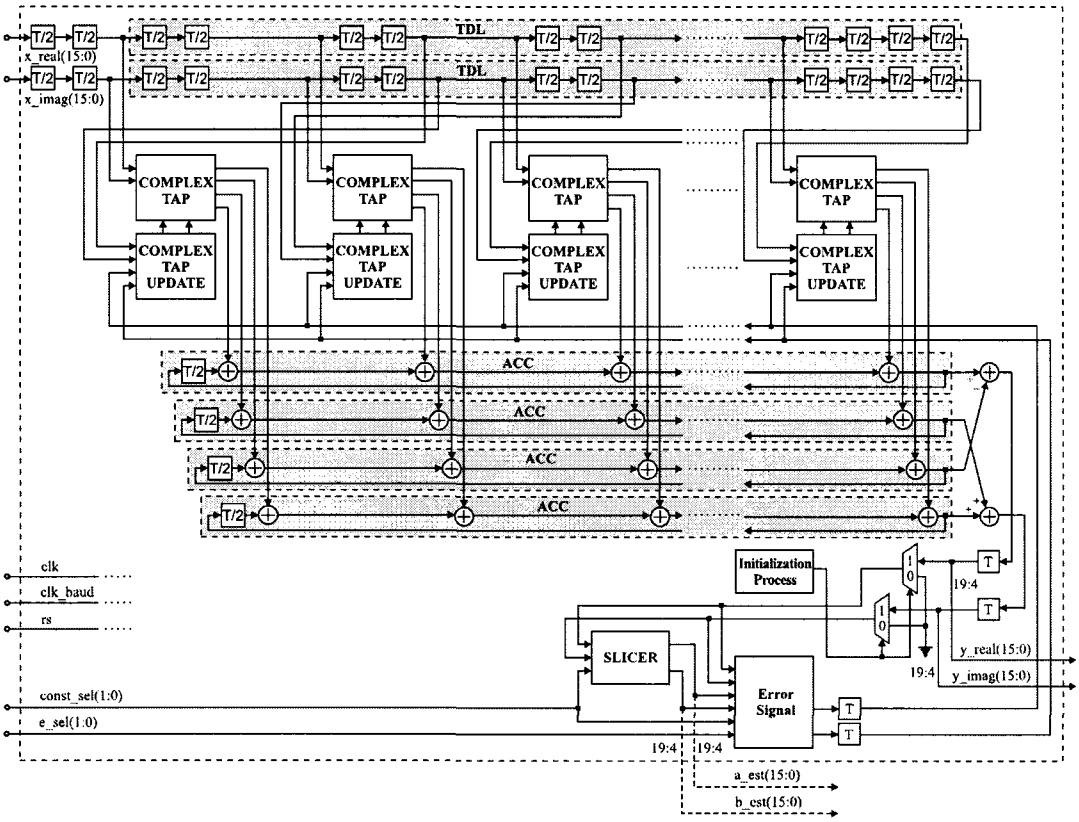


Figure 6.4: Blind equalizer block diagram.

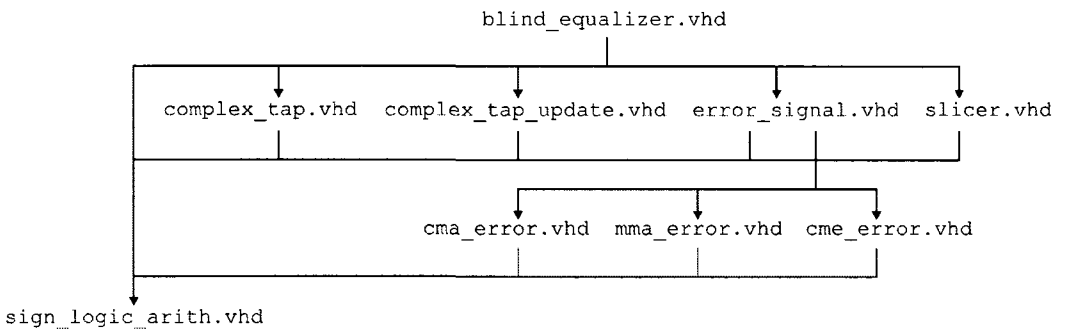


Figure 6.5: Hierarchy of VHDL files for the blind equalizer IP core.

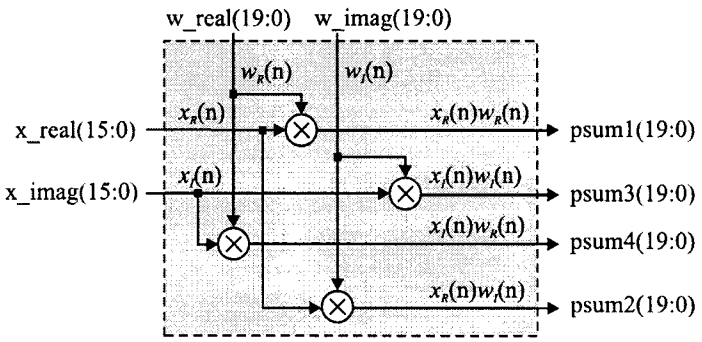


Figure 6.6: Complex tap implementation.

6.4.1 Complex Tap Implementation

The complex tap module performs a single complex multiplication between a complex equalizer input sample and an equalizer tap coefficient. As illustrated in Fig, 6.6, the input signals of the complex tap update module are `x_real(15:0)`, `x_imag(15:0)`, `w_real(19:0)` and `w_imag(19:0)`, while the output signals are `psum1(19:0)`, `psum2(19:0)`, `psum3(19:0)` and `psum4(19:0)`. The input signals `w_real` and `w_imag` are the real and imaginary complex tap coefficients, respectively. The equalizer input samples are sign-extended to 20-bit and are multiplied with the equalizer tap coefficient samples to generate the four partial products of a complex multiplication. These partial products, truncated to 20-bit, are the output of the complex tap module which correspond to the partial products of the four real filters of Fig. 6.1. The output of the complex tap module is sent to four accumulators which are used to generate the equalizer output. The complex tap module does not in itself implement an equalizer tap, only a complex multiplication. It is the complex tap update module in the section to follow that stores and adjusts an equalizer tap coefficient.

6.4.2 Complex Tap Update Implementation

The complex tap update module stores the current equalizer tap coefficient and calculates the next value of the tap coefficient. As illustrated in Fig. 6.7, the input signals of the complex tap update module are `x_real(15:0)`, `x_imag(15:0)`, `mue_real(15:0)`, `mue_imag(15:0)`, `clk`, `clk_baud`, and `rs`, while the output signals are `w_real(19:0)` and `w_imag(19:0)`. The input signals `mue_real` and `mue_imag` are the real and imaginary components of the error signal scaled by the step size, respectively. The complex tap update module implements two consecutive complex equalizer tap coefficients and their respective updates. The update term for the odd tap coefficient is evaluated

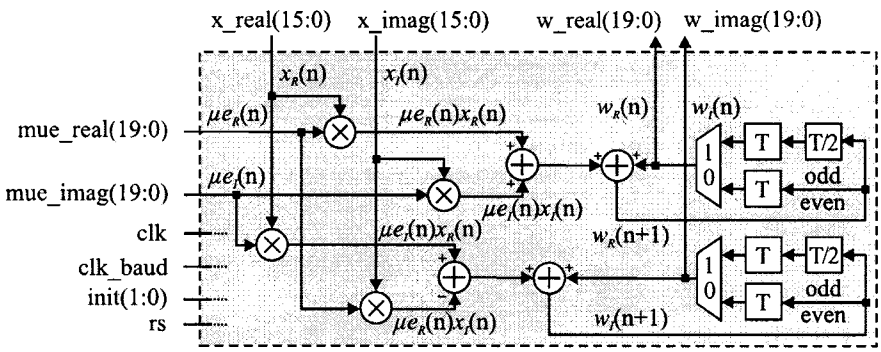


Figure 6.7: Complex tap update implementation.

when `clk_baud='1'` and the even tap coefficient is evaluated when `clk_baud='0'`, while both odd and even taps are updated on the positive edge of `clk_baud`. This is achieved by using multiplexors to select the tap coefficient to be adjusted, where the multiplexor output is controlled by `clk_baud`. This structure allows the evaluation of two tap coefficients using the same arithmetic components, thereby, reducing the number of arithmetic components in half for the evaluation and update portions of the equalizer.

At the onset of equalization, a reset signal `rs='1'` is applied and the equalizer tap coefficients are synchronously reset based on the `init(1:0)` input vector. The imaginary component of the tap coefficients are initialized to zero, while all bits of the real component of the tap coefficients are initialized to zero except for the integer bit. The bit `init(0)` corresponds to the initialization value of the odd tap coefficient, while the bit `init(1)` corresponds to the initialization value of the even tap coefficient. Therefore, the tap coefficients are initialized to `'0'` or `'1'`. Recall from (3.21), that the general complex equalizer tap adjustment consists of four partial products as follows

$$\begin{aligned}\mathbf{w}(n+1) &= \mathbf{w}(n) + \mu e(n)x^*(n) \\ &= \mathbf{w}(n) + \mu e_R(n)\mathbf{x}_R(n) + \mu e_I(n)\mathbf{x}_I(n) + j\left(\mu e_I(n)\mathbf{x}_R(n) - \mu e_R(n)\mathbf{x}_I(n)\right).\end{aligned}$$

The equalizer input samples are sign-extended to 20-bit and are multiplied with the tap coefficient samples to generate the four partial products which are truncated to 20-bit. When `clk_baud='1'`, the real component of the odd tap coefficient is added to the partial products $\mu e_R(n)x_R(n)$ and $\mu e_I(n)x_I(n)$, while the imaginary component of the odd tap coefficient is added to the partial products $\mu e_I(n)x_R(n)$ and $-\mu e_R(n)x_I(n)$. These results are stored with intermediate $T/2$-spaced registers. Similar to the odd tap adjustment, when `clk_baud='0'`, the real component of the even tap coefficient is added to the partial products $\mu e_R(n)x_R(n)$ and $\mu e_I(n)x_I(n)$, while the imaginary com-

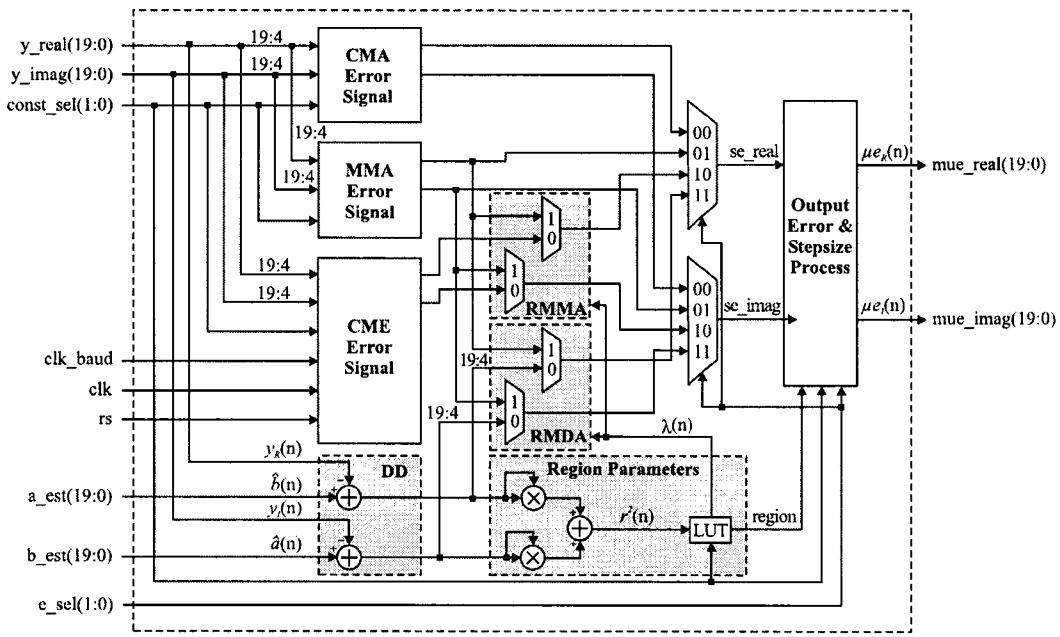


Figure 6.8: Error signal implementation.

ponent of the even tap coefficient is added to the partial products $\mu e_I(n)x_R(n)$ and $-\mu e_R(n)x_I(n)$. At the rising edge of `clk_baud`, the next values of the odd and even tap coefficients are sampled and become the current value of the equalizer tap coefficients.

6.4.3 Error Signal Implementation

The error signal is composed of CMA error, MMA error, CME error, and DD error modules, which combine to form the CMA, MMA, RMMA and RMDA error signals. The error signal is controlled by the signals `e_sel(1:0)` and `const_sel(1:0)`, which together are used to configure the IP core in one of the 13 different modes of operation listed in Table 6.1. As illustrated in Fig. 6.8, the input signals of the error signal module are `y_real(19:0)`, `y_imag(19:0)`, `a_est(19:0)`, `b_est(19:0)`, `e_sel(1:0)`, `const_sel(1:0)`, `clk`, `clk_baud`, and `rs`, while the output signals are `mue_real(19:0)` and `mue_imag(19:0)`. The precision of the error signal is 16-bit, while the precision of the region parameters is 20-bit. The output of the error signal module is the error signal scaled by the step size process, which is dependent on the values of `const_sel`, `e_sel`, and the region parameters for RMMA and RMDA.

Table 6.1: Error Signal Operation Modes and Stepsizes

Mode	const_sel	e_sel	Signal	Stepsizes	Error Signal
1	"00"	"00"	QPSK	2^{-6}	CMA
2	"00"	"01"	QPSK	2^{-6}	MMA
3	"01"	"00"	16-QAM	2^{-9}	CMA
4	"01"	"01"	16-QAM	2^{-9}	MMA
5	"01"	"10"	16-QAM	$\{2^{-7}, 2^{-8}, 2^{-9}, 2^{-9}, 2^{-11}\}$	RMMA
6	"01"	"11"	16-QAM	$\{2^{-7}, 2^{-8}, 2^{-9}, 2^{-9}, 2^{-11}\}$	RMDA
7	"10"	"00"	64-QAM	2^{-10}	CMA
8	"10"	"01"	64-QAM	2^{-10}	MMA
9	"10"	"10"	64-QAM	$\{2^{-8}, 2^{-9}, 2^{-10}, 2^{-10}, 2^{-11}\}$	RMMA
10	"10"	"11"	64-QAM	$\{2^{-8}, 2^{-9}, 2^{-10}, 2^{-10}, 2^{-11}\}$	RMDA
11	"11"	"00"	256-QAM	2^{-11}	CMA
12	"11"	"01"	256-QAM	2^{-11}	MMA
13	"11"	"11"	256-QAM	$\{2^{-9}, 2^{-10}, 2^{-11}, 2^{-11}, 2^{-12}\}$	RMDA

Table 6.2: Dispersion constants for CMA and MMA for LUT implementation.

const_sel	Signal	Constant γ_C^2	Constant γ_M^2
"00"	QPSK	1.0000	0.5000
"01"	16-QAM	1.3199	0.8199
"10"	64-QAM	1.3810	0.8810
"11"	256-QAM	1.3953	0.8953

CMA, MMA, and CME Error Signal Implementations

The CMA error signal module is illustrated in Fig. 6.9(a), where the input signals are `y_real(15:0)` and `y_imag(15:0)`, while the output signals are `e_real(15:0)` and `e_imag(15:0)`. The CMA error signal, which was defined in (3.29), is given by

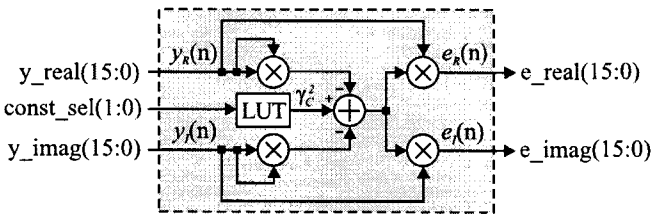
$$e^{\text{cma}}(n) = \underbrace{y_R(n)\left(\gamma_C^2 - y_R^2(n) - y_I^2(n)\right)}_{e_R^{\text{cma}}(n)} + \jmath \underbrace{y_I(n)\left(\gamma_C^2 - y_R^2(n) - y_I^2(n)\right)}_{e_I^{\text{cma}}(n)}$$

and can be implemented with $4M + 1A$, where 'M' and 'A' correspond to the number of real multiplications and additions, respectively. The dispersion constant γ_C^2 is implemented using a LUT, whose values are listed in Table 6.2 for QPSK, 16-QAM, 64-QAM, and 256-QAM signals.

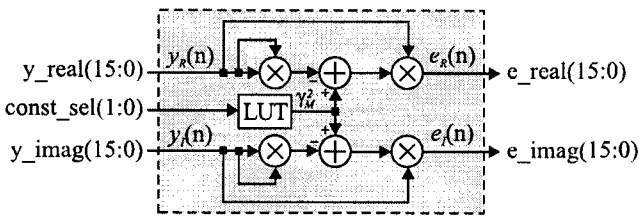
The MMA error signal module is illustrated in Fig. 6.9(b), where the input signals are `y_real(15:0)` and `y_imag(15:0)`, while the output signals are `e_real(15:0)` and `e_imag(15:0)`. The MMA error signal, which was defined in (3.33), is given by

$$e^{\text{mma}}(n) = \underbrace{y_R(n)\left(\gamma_M^2 - y_R^2(n)\right)}_{e_R^{\text{mma}}(n)} + \jmath \underbrace{y_I(n)\left(\gamma_M^2 - y_I^2(n)\right)}_{e_I^{\text{mma}}(n)} \tag{6.1}$$

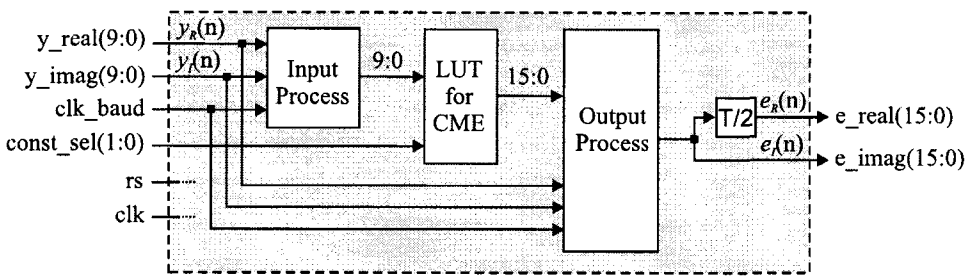
and can be implemented with $4M + 2A$. The dispersion constant γ_M^2 is implemented using a LUT, whose values are listed in Table 6.2 for QPSK, 16-QAM, 64-QAM, and 256-QAM signals.



(a) CMA error signal implementation.



(b) MMA error signal implementation.



(c) CME error signal implementation.

Figure 6.9: CMA, MMA, and CME error signal implementations.

The CME error signal is illustrated in Fig. 6.9(c), where the input signals are `y_real(9:0)`, `y_imag(9:0)`, `const_sel`, `clk_baud`, `clk`, and `rs`, while the output signals are `e_real(15:0)` and `e_imag(15:0)`. The CME error signal selected for this implementation was defined in (4.7) and is given by

$$\eta(n) = \frac{\pi}{d}\left[\sin\left(2\pi\frac{y_R(n)}{d}\right) + \jmath\sin\left(2\pi\frac{y_I(n)}{d}\right)\right].$$

This CME error signal $\eta(n)$ is scaled by the weighting factor β to form the constellation specific components of the RMMA error signal, where the values of β are 0.0389 and 0.0771 for 16-QAM and 64-QAM, respectively. The term $\beta\eta(n)$ is implemented as a LUT for 16-QAM and 64-QAM signals. By noting that $\sin(-x) = -\sin(x)$, the LUTs are implemented for only for positive values of the input. Separate input and output processes complement the real and/or imaginary input and corresponding output signal(s) when `y_real(9)='1'` and/or `y_imag(9)='1'`, respectively. Furthermore, since the real and imaginary components of the CME error signal are equivalent, the same LUT is used to calculate both components. The LUTs for 16-QAM and 64-QAM each have 513 different input combinations (513th input is `"1000000000"` or -2), which correspond to 513 different 16-bit outputs. When `clk_baud='1'`, the internal signal `eta_temp` is assigned `y_real`, while `eta_temp` is assigned `y_real` when `clk_baud='0'`. The intermediate result, for `e_real`, is stored in a $T/2$-spaced flip-flop, while the result for `e_imag` evaluated during the interval when `clk_baud='0'` such that the output is ready to be sampled on the rising edge of `clk_baud='1'`.

RMMA and RMDA Error Signal Implementation

The RMMA error signal, which was defined in (4.8), is a combination of the MMA and DD error signals and is given by

$$\begin{aligned} e^{\text{rmma}}(n) &= \lambda(n)\left(y_R(n)\left(\gamma_M^2 - y_R^2(n)\right)\right) + \jmath y_I(n)\left(\gamma_M^2 - y_I^2(n)\right)\right) + (1-\lambda(n))\,\beta\eta(n) \\ &= \underbrace{\left(\lambda(n)e_R^{\text{mma}}(n) + (1-\lambda(n))\beta\eta_R(n)\right)}_{e_R^{\text{rmma}}(n)} + \jmath\underbrace{\left(\lambda(n)e_I^{\text{mma}}(n) + (1-\lambda(n))\beta\eta_I(n)\right)}_{e_I^{\text{rmma}}(n)}. \end{aligned}$$

For this implementation, five decision regions were used according to Table 4.2 and the weighting factor was quantized to $\lambda(n) = \{0, 1\}$. This results in the efficient RMMA error signal implementation illustrated in Fig. 4.2, where the RMMA error signal is the muxed output of the MMA and CME error signals with $\lambda(n)$ as the selector. The RMMA error signal is implemented for 16-QAM and 64-QAM signals.

The RMDA error signal, which was defined in (4.11), is a combination of the MMA and DD

Table 6.3: Region boundary values for RMMA and RMDA.

const_sel	Signal	Region Boundary (Squared)
"01"	16-QAM	0.09999942
"01"	16-QAM	0.04444408
"01"	16-QAM	0.01440620
"01"	16-QAM	0.00277709
"10"	64-QAM	0.02380943
"10"	64-QAM	0.01058197
"10"	64-QAM	0.00264549
"10"	64-QAM	0.00066089
"11"	256-QAM	0.00588226
"11"	256-QAM	0.00261402
"11"	256-QAM	0.00065326
"11"	256-QAM	0.00016307

error signals and is given by

$$e^{\text{rmda}}(n) = \lambda(n)\left(y_R(n)\left(\gamma_M^2 - y_R^2(n)\right)\right) + jy_I(n)\left(\gamma_M^2 - y_I^2(n)\right)\right) + (1-\lambda(n))\left(\hat{s}(n) - y(n)\right)$$
$$= \underbrace{\left(\lambda(n)e_R^{\text{mma}}(n) + (1-\lambda(n))e_R^{\text{dd}}(n)\right)}_{e_R^{\text{rmda}}(n)} + j\underbrace{\left(\lambda(n)e_I^{\text{mma}}(n) + (1-\lambda(n))e_I^{\text{dd}}(n)\right)}_{e_I^{\text{rmda}}(n)}$$

For this implementation, five decision regions were used according to Table 4.2 and the weighting factor was quantized to $\lambda(n) = \{0, 1\}$. This results in the efficient RMDA error signal implementation illustrated in Fig. 4.3, where the RMMA error signal is the muxed output of the MMA and DD error signals with $\lambda(n)$ as the selector. The RMDA error signal is implemented for 16-QAM, 64-QAM, and 256-QAM signals.

In order to avoid implementing the square root function, the region for the RMMA and RMDA error signals is calculated using $r^2(n)$ instead of $r(n)$. Consequently, a 20-bit WL is required for this calculation, which increases the resolution to 2^{-18} from 2^{-16}. The calculation of $r^2(n)$ for the region requires $2M + 3A$. However, only $2M + 1A$ is required since the RMDA error signal incorporates the DD error signal, which is used in the calculation of the region. The four static boundaries that correspond to $r^2(n)$ are listed in Table 6.3 for 16-QAM, 64-QAM, and 256-QAM.

6.4.4 Complex QAM Slicer Implementation

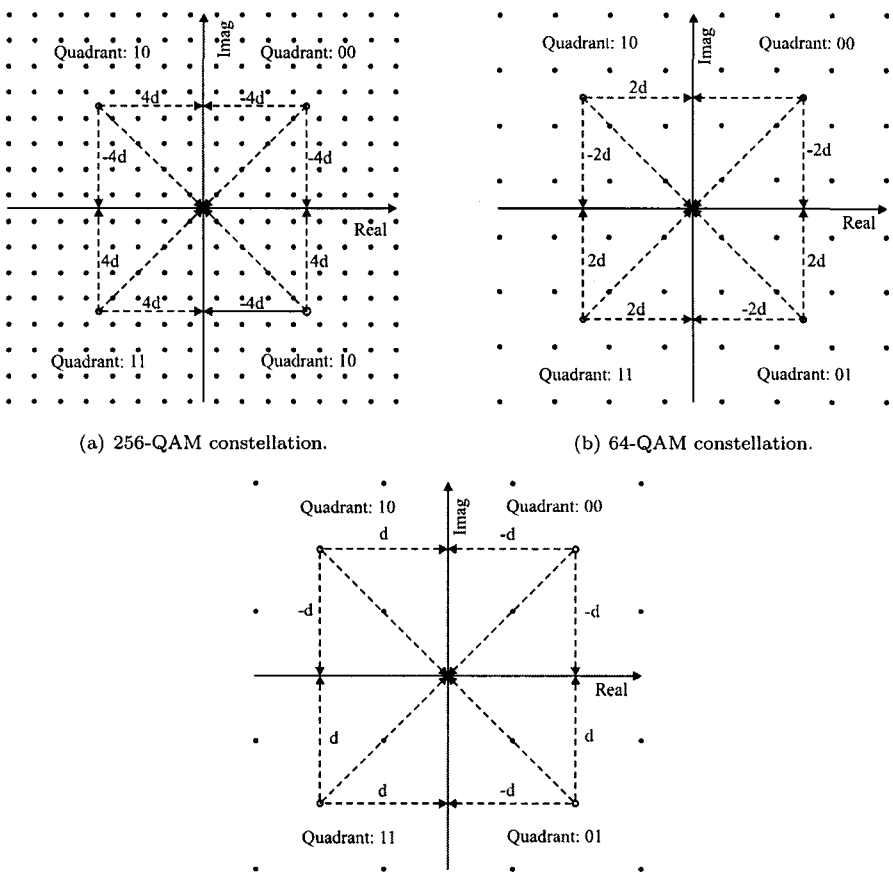
The complex QAM slicer module produces an estimate of the transmitted symbol from the equalizer output. The input signals to the QAM slicer module are `y_real(19:0)`, `y_imag(19:0)`, and `const_sel(1:0)`, while the output signals are `a_est(19:0)` and `b_est(19:0)`. The QAM slicer module operates in four modes: QPSK ("00"), 16-QAM ("01"), 64-QAM ("10"), and 256-QAM ("11"),

which are based on the value of the control signal `const_sel`. The symbol estimate is determined by successively dividing the QAM constellation into four quadrants based on the sign bit of the `y_real` and `y_imag` signals and then biasing the center of the selected quadrant to the center of the signal constellation, i.e. to (0,0). With this method, a large signal constellation is broken down into smaller constellations until the constellation is QPSK. The process for a 256-QAM constellation is as follows: the quadrant is determined based on the sign bit of the equalizer output signals and the 256-QAM constellation is reduced to a 64-QAM constellation by biasing the center of the selected quadrant to the point (0,0) as illustrated in Fig. 6.10(a). The new quadrant is determined and the 64-QAM constellation is reduced to a 16-QAM constellation by biasing the center of the selected quadrant to the point (0,0) as illustrated in Fig. 6.10(b). The new quadrant is determined and the 16-QAM constellation is reduced to a QPSK constellation by biasing the center of the selected quadrant to the point (0,0) as illustrated in Fig. 6.10(c). The estimated symbol can then be obtained by determining the final quadrant. This method requires $6A$, $4A$, and $2A$ for 256-QAM, 64-QAM, and 16-QAM, respectively, where 'A' is a real addition.

6.5 Implementation Results

The functionality of the IP core at the RTL level was verified using Cadence NC-VHDL. The verification strategy consisted of successive testing and assembling of lower modules until all the modules in the hierarchy were tested. A module was tested by creating a set of fixed-point inputs and expected outputs using the fixed-point toolbox in Matlab. The data was saved to a stimulus file with an extension of "`.dat`" and the stimulus vectors were applied by the VHDL testbench using the IEEE `std_logic_textio` and STD `textio` libraries for file I/O. The stimulus data consisted of several thousand to tens of thousands of data vectors randomly generated by fixed-point Matlab programs. The output of the RTL simulation was written to a file in Matlab format in order to produce a graphical output. The final RTL results for the IP core are illustrated in Fig. 6.11, Fig. 6.12, Fig. 6.13, and Fig. 6.14, for CMA, MMA, RMMA, and RMDA, respectively. These results are for a single realization and illustrate the instantaneous squared error across the slicer and the output signal constellation after convergence. These results are similar to the floating-point Matlab results of sections 5.5.1 and 5.5.2.

After verification, the IP core was synthesized using Synopsys Design Compiler (DC). The target library for synthesis was the "`stratixii.db`" library for Altera Stratix II FPGAs, while the synthetic libraries targeted were the Altera "`stratixii.sldb`" and "`dw_foundation.sldb`" libraries.

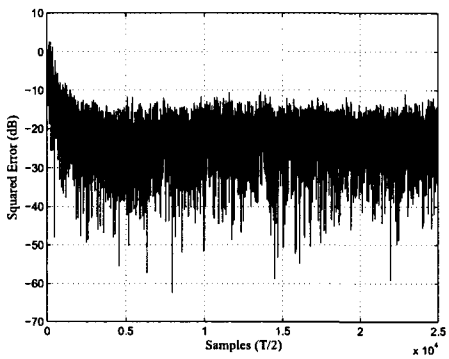


(a) 256-QAM constellation.

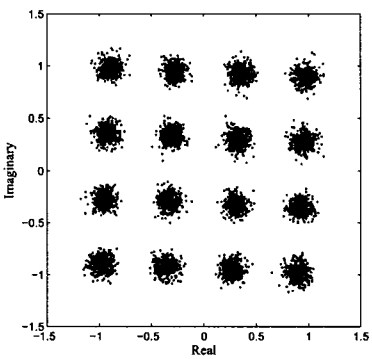
(b) 64-QAM constellation.

(c) 16-QAM constellation.

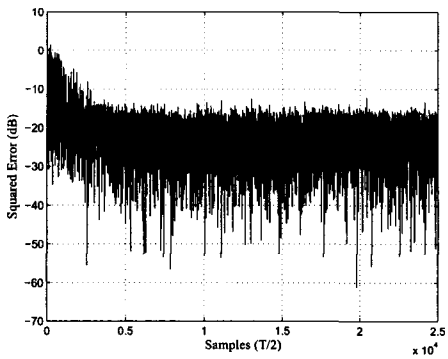
Figure 6.10: Equalizer output bias based on output signal quadrant.



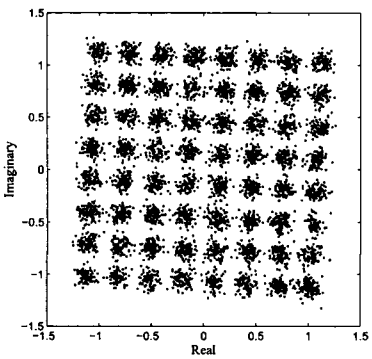
(a) Squared error for 16-QAM.



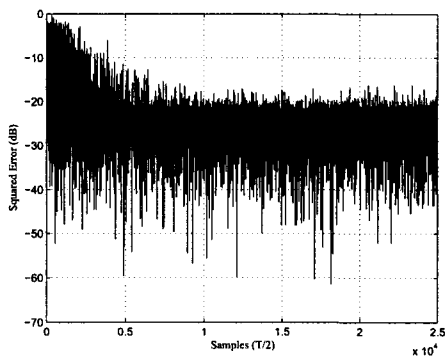
(b) Output constellation for 16-QAM.



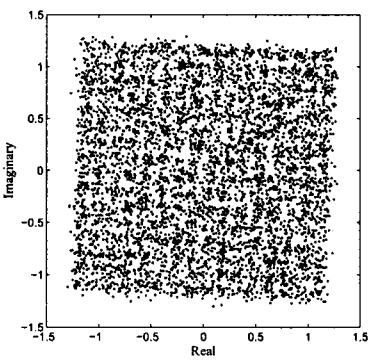
(c) Squared error for 64-QAM.



(d) Output constellation for 64-QAM.

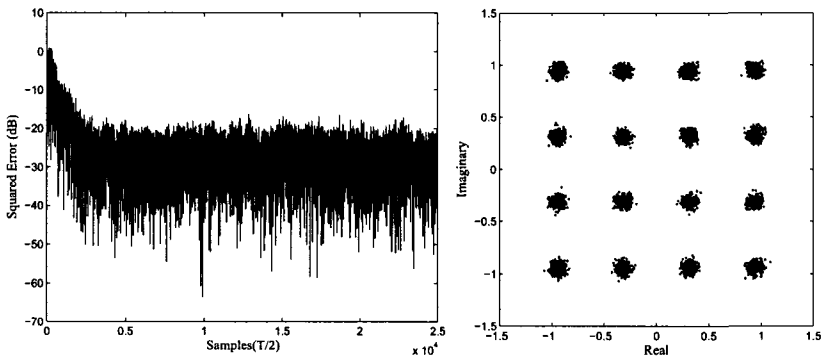


(e) Squared error for 256-QAM.

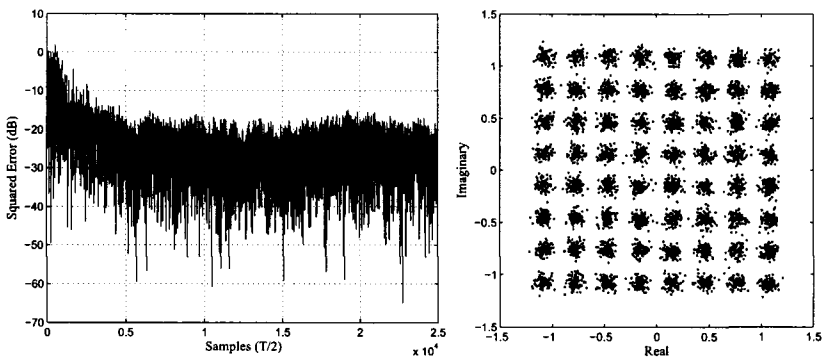


(f) Output constellation for 256-QAM.

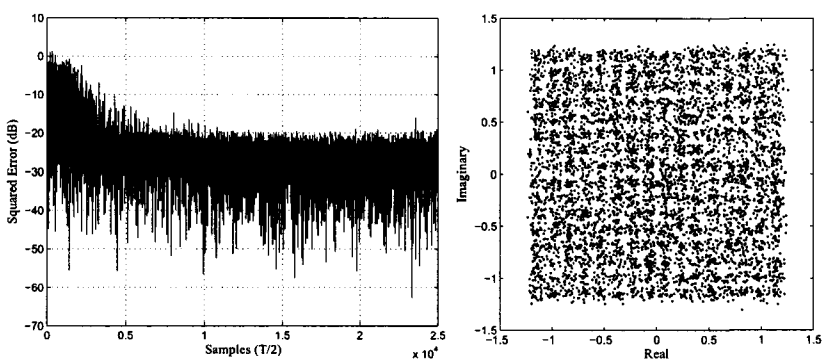
Figure 6.11: RTL simulation results for CMA in CC-01 using NC-VHDL.



(a) Squared error for 16-QAM. (b) Output constellation for 16-QAM.

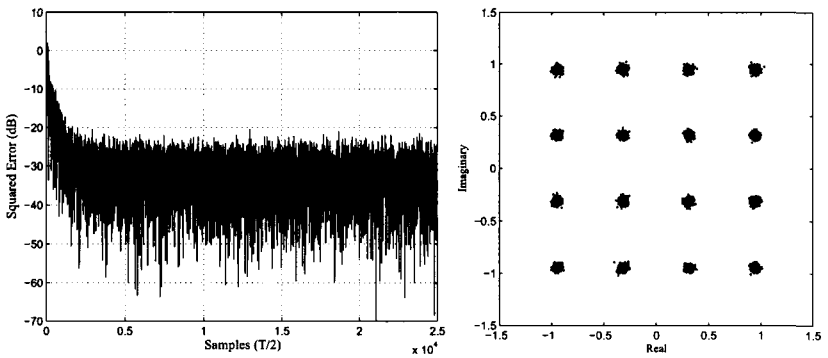


(c) Squared error for 64-QAM. (d) Output constellation for 64-QAM.



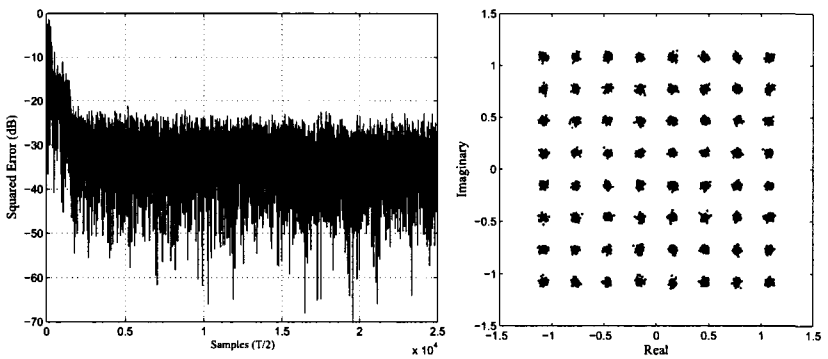
(e) Squared error for 256-QAM. (f) Output constellation for 256-QAM.

Figure 6.12: RTL simulation results for MMA in CC-01 using NC-VHDL.



(a) Squared error for 16-QAM.

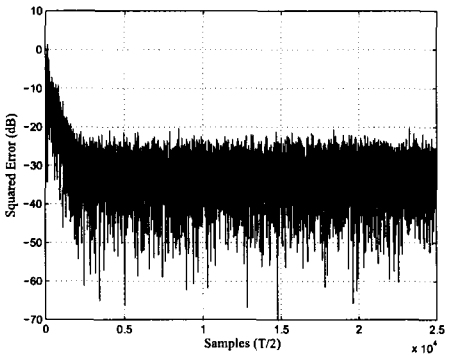
(b) Output constellation for 16-QAM.



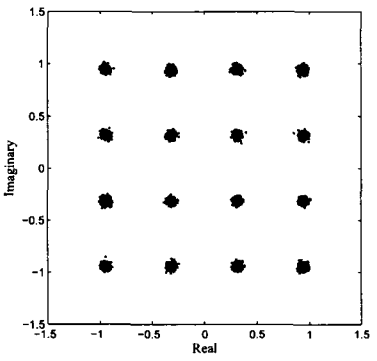
(c) Squared error for 64-QAM.

(d) Output constellation for 64-QAM.

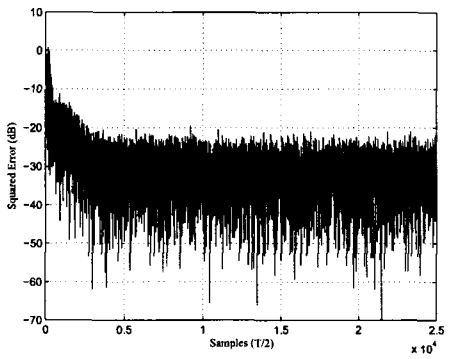
Figure 6.13: RTL simulation results for RMMA in CC-01 using NC-VHDL.



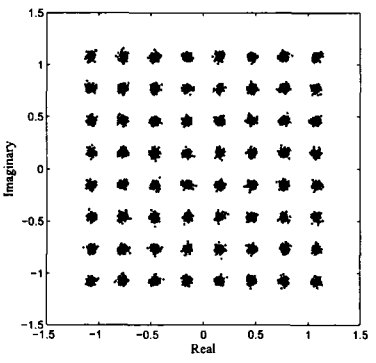
(a) Squared error for 16-QAM.



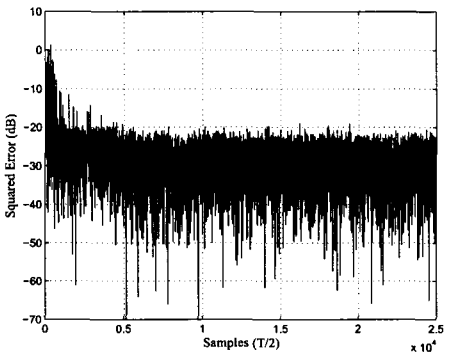
(b) Output constellation for 16-QAM.



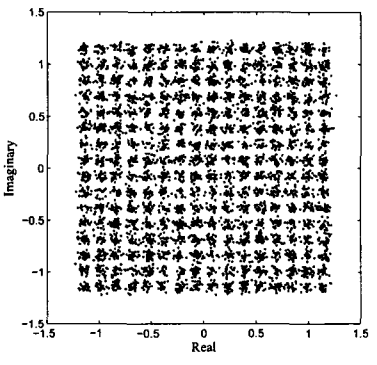
(c) Squared error for 64-QAM.



(d) Output constellation for 64-QAM.



(e) Squared error for 256-QAM.



(f) Output constellation for 256-QAM.

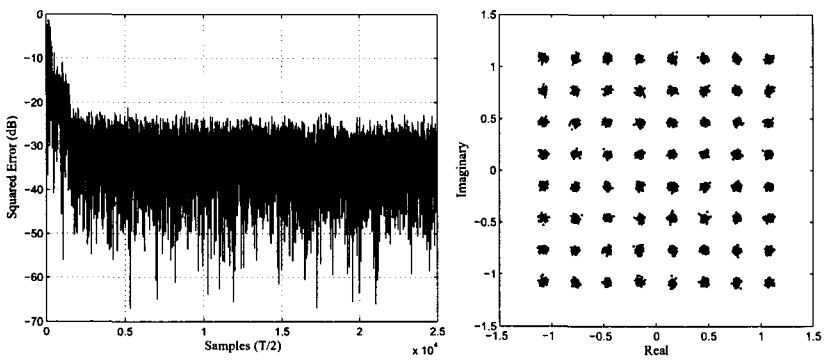
Figure 6.14: RTL simulation results for RMDA in CC-01 using NC-VHDL.

Table 6.4: Implementation characteristics and performance.

FPGA	Altera Stratix II
Target Device	EP2S130F780C4
Total ALUTs	53,862 (Utilization: 50%)
Total Registers	1969
Total I/O Pins	103 (Utilization: 19%)
Maximum Clock Frequency	16.11 MHz
Maximum Symbol Frequency	8.055 MBaud
Maximum Bit Rate	64.44 Mbits
Dynamic Power Consumption	2.47 Watts

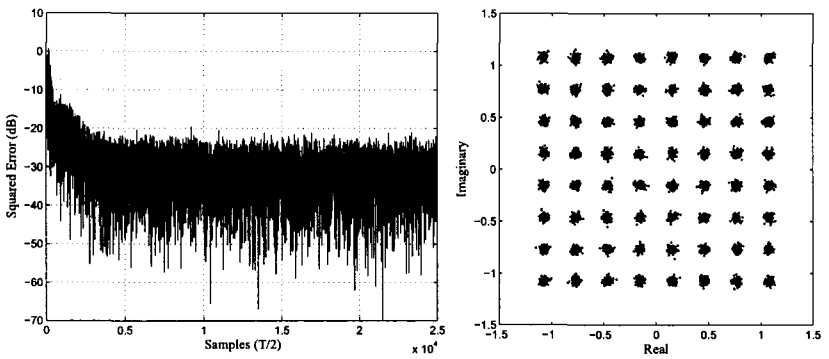
The IP core was synthesized using most of the default settings and the design hierarchy was exported to the "`.edif`" file format to be compatible with the Altera Quartus II FPGA tools used for physical synthesis. The synthesized gate-level netlist from DC was analyzed by the Quartus II software and physical synthesis was conducted for the Stratix II EP2S130F780C4 FPGA, which has a capacity of 106,032 adaptive look-up tables (ALUTs). This is equivalent to 132,540 logic elements (LEs), which is the four-input LUT-based architecture used in Altera Stratix FPGAs. The physical synthesis results are listed in Table 6.4. The IP core utilized 53,862 ALUTs or approximately 50% of the total ALUTs, while 1969 registers (single bit D flip-flops) were instantiated. Timing analysis was completed with the "Timing Analyzer" module of Quartus II. The IP core can operate at a maximum clock frequency of 16.11 MHz and a maximum symbol frequency of 8.055 MBaud. This corresponds to a maximum bit rate of 64.44 Mbits/s. Additionally, the power dissipation of the IP core was estimated using the "PowerPlay Power Analyzer" module of Quartus II. The dynamic power consumptation was estimated to be 2.47 Watts.

The functionality and timing of the final design was verified at the gate-level using the output gate-level netlist ("`.vho`") and standard delay format files ("`.sdo`") produced by Quartus II for NC-VHDL. The "`.sdo`" file is a standard delay format file that contains information for back-annotation of the delay at the gate level. In addition to the output files from Quartus II, the Altera libraries "`lpm`", "`altera_mf`", and "`stratixii`" (atoms) were compiled for simulation. The original VHDL testbench was modified for the gate-level netlist and the frequencies of `clk` and `clk_baud` were set to 16MHz and 8MHz, respectively. The gate-level simulation results for RMMA and RMDA are illustrated in Fig. 6.15 for 64-QAM. These results are nearly identical to their respective RTL results, which serve to validate the correct operation of the design at a symbol frequency of 8MBaud.



(a) RMMA squared error.

(b) RMMA output constellation.



(c) RMDA squared error.

(d) RMDA output constellation.

Figure 6.15: Gate-level simulation results for RMMA and RMDA using NC-VHDL.

Table 6.5: Recent QAM equalizer implementations for cable modems.

	Technology	QAM Modes	Symbol Rate	Algorithms
Yamanaka [86]	0.5μm	16, 64, 256	8.25MBaud	DD
Shin [65]	0.35μm	64, 256	8M Baud	MMA, DD
Kurakake [47]	0.35μm	1024	5.274MBaud	MCMA (MMA)
Fukuoka [30]	n/a	4, 16, 32, 64	5M	SAG-DD
Shen [64]	0.25μm	4→256	10MBaud	CMA, DD
Tan [72]	0.5μm	4, 16, 32, 64, 128, 256, 1024	7MBaud	S-LMS
D'Luna [25]	0.35μm	4, 16, 32, 64, 128, 256	7MBaud	S-LMS
Wu [85]	0.6μm	64	5MBaud	S-DLMS
This work	FPGA	4, 16, 64, 256	8.055M	CMA, MMA, RMMA, RMDA

6.6 Comparisons

In this section, the IP core implementation on the Stratix II EP2S130F780C4 FPGA is compared with the equalizer component of recent QAM demodulator implementions for cable modems. A number of recent ASIC QAM demodulator designs have been implemented that incorporate blind [86][65][47][64] or trained adaptive equalizers [72][25][85]. The symbol frequency ranges from 5 MBaud to 10 MBaud, while the signal constellation ranges from 4-QAM (or QPSK) to 1024-QAM. The IP core implementation operates at a maximum symbol frequency of 8.055Mbaud, which is the third highest symbol frequency and within 0.195MHz of the second highest symbol frequency. This implementation is rated for square QAM constellations up till 256-QAM, which is the largest signal constellation mode for implementations that operate at or above 8MBaud. Additionally, this implementation can be configured in for error modes. These results are significant since this is the only FPGA implementation and it is able to achieve performance that is comparable to recent ASIC implementations.

Chapter 7

Conclusions and Future Work

This thesis has introduced a new radius-adjusted approach for blind equalization of QAM signals. This approach adjusts the equalizer tap coefficients using a linearly weighted sum of adaptation criteria scaled by a variable step size, where the weighting factor and step size were dependent on the equalizer output radius. The approach is the basis for two new radius-adjusted blind equalization algorithms, RMMA and RMDA. A method to tune the new algorithms was developed based on statistics of the equalizer output radius. The transient and steady-state performance of RMMA and RMDA were analyzed and analytic expressions for their steady-state analysis were derived. A series of comparative studies were performed which compared the performance of RMMA and RMDA with related hybrid blind equalization algorithms. Simulations were conducted over static microwave and cable channels. The results have indicated that, on average, the radius-adjusted algorithms achieve faster convergence time and lower steady-state MSE.

The radius-adjusted approach was extended to a computationally-efficient method termed the selective update method. This method employs a stop-and-go strategy based on the equalizer output radius to selectively update the equalizer tap coefficients. The convergence and steady-state behavior of algorithms modified by the selective update method was analyzed based on the statistics of the equalizer output radius and the concept of adjustment error. A modified selective update method was also proposed, which reduces the hardware requirements for implementation in addition to the number of computations. A series of comparative studies were performed that compared the performance of algorithms modified by the selective update method with related computationally-

efficient methods. Simulation results have indicated that the transient characteristics of algorithms modified by the selective update method are retained, even in Ricean fading channels. The steady-state MSE is related to a static paramater, which depending on the selection, can result in a lower steady-state MSE.

A complex 18-Tap $T/2$-spaced blind equalizer IP core was designed for QAM signals and implemented on an Altera Stratix II FPGA. The IP core can be configured in four error signal modes and can equalize square QAM signals up to 256-QAM. The IP core implementation targeted the Altera Stratix II EP2S130F780C4 FPGA, where a maximum symbol frequency of 8.055 MBaud was obtained. Gate-level simulation and timing was successfully performed at a frequency of 8 MBaud. The results are comparable to recent QAM equalizer designs for cable modems.

There are several ways to expand the work presented in this thesis. The radius-adjusted approach can be extended to other types of modulation such as VSB. This would be ideal since VSB modulation is used in the ATSC standard for terrestrial HDTV transmission, which recommends blind DFE. The IP core can also be targeted for a custom ASIC implementation and tested in an analog environment.

References

[1] Digital video broadcasting (DVB); framing structure, channel coding and modulation for 11/12 GHz satellite services. ATSC standard revision A, European Telecommunications Standards Institute, 1998.

[2] Transmission measurement and compliance for digital television. ATSC standard revision A, Advanced Television Systems Committee, 2000.

[3] Digital video broadcasting (DVB); framing structure, channel coding and modulation for cable systems. ATSC standard revision A, European Telecommunications Standards Institute, 2004.

[4] T. Aboulnsar and K. Mayyas. Complexity reduction of the NLMS algorithm via selective coefficient update. *IEEE Transactions on Signal Processing*, 47:1421–1424, 1999.

[5] D. J. Allred, H. Yoo, V. Krishnan, W. Huang, and D. V. Anderson. A novel high performance distributed arithmetic adaptive filter implementation on an fpga. In *Proc. ICASSP'04*, pages 161–164, Monreal, Quebec, May 2004.

[6] R. A. Axford, L. B. Milstein, and J. R. Zeidler. A dual-mode algorithm for blind equalization of QAM signals: CADAMA. In *Proc. Asilomar Conference*, pages 172–176, October 1995.

[7] F. Balarin, Y. Wantanabe, H. Hsieh, L. Lavagno, C. Passerone, and A. Sangiovanni-Vincentelli. Metropolis: an integrated electronic system design environment. *IEEE Computer Magazine*, 36:45–52, April 2003.

[8] K. Banovic, E. Abdel-Raheem, and M. A. S. Khalid. A novel radius-adjusted approach for blind adaptive equalization. *IEEE Signal Processing Letters*, 13(1):37–40, January 2006.

[9] K. Banovic, R. Lee, E. Abdel-Raheem, and M. A. S. Khalid. Computationally-efficient methods for blind adaptive equalization. In *Proc. Int. Midwest Symposium on Circuits & Systems*, pages 341–344, Cincinnati, Ohio, August 2005.

[10] S. Barbarossa and A. Scaglione. Blind equalization using cost functions matched to the signal constellation. In *Proc. Asilomar Conf. Signal Syst. Compt.*, November 1997.

[11] Signal Processing Information Base. http://www.spib.rice.edu, 2005.

[12] J. Benesty and P. Duhamel. Fast constant modulus adaptive algorithm. *IEE Proceedings*, 138:379 – 387, 1991.

[13] L. Benini, D. Bertozzi, D. Bruni, N. Drago, F. Fummi, and M. Poncino. SystemC cosimulation and emulation of multiprocessor SoC designs. *IEEE Computer Magazine*, 36:53–59, April 2003.

[14] A. Benveniste and M. Goursat. Blind equalizers. *IEEE Trans. on Communications*, COM-32:871–883, August 1984.

[15] D. R. Brown, P. Schniter, and C. R. Johnson. Computationally efficient blind equalization. In *Proc. Allerton Conf. on Communication, Control and Computing*, pages 54–63, Monticello, IL, October 1997.

[16] S. Cardillo, M. Chiodo, P. Giusto, A. Jurecska, L. Lavagno, and A. Sangiovanni-Vincentelli. Rapid-prototyping of embedded systems via reprogrammable devices. In *Proc. 7th Int'l Workshop on Rapid System Prototyping*, pages 133–138, June 1996.

[17] F. C. C. De Castro, M. C. F. De Castro, and D. S. Arantes. Concurrent blind deconvolution for channel equalization. In *Proc. International Conference on Communications*, pages 366–371, June 2001.

[18] S. Chen. Low complexity concurrent constant modulus algorithm and soft decision directed scheme for blind equalisation. *IEE Proc.-Vis Image Signal Processing*, 150(5):312–320, July 2003.

[19] S. Chen and E. S. Chng. Concurrent constant modulus algorithm and soft decision directed scheme for fractionally-spaced blind equalization. In *Proc. International Conference on Communications*, pages 2342–2346, June 2004.

[20] H. Choi, G.-T. Ryu, D.-S. Kim, K.-T. Lim, T.-W. Kwon, H.-J. Lim, and H.-D. Bae. Variable block lms adaptive filter with gaussian input. In *Proc. 10th DSP Workshop/2nd Signal Processing Education Workshop*, pages 217–220, Pine Mountain, Georgia, October 2002.

[21] G. A. Clark, S. K. Mitra, and S. R. Parker. Block implementation of adaptive digital filters. *IEEE Transaction on Circuits & Systems*, CAS-28:584–592, 1981.

[22] G. Constantinides. Perturbation analysis for word-length optimization. In *Proc. Field-Programmable Custom Computing Machines*, pages 81–90, Napa, CA, April 2003.

[23] C. Dick. Re-discovering signal processing: a configurable logic based approach. In *Proc. Of Asilomar*, pages 1370–1374, Pacific Grove, CA, November 2003.

[24] P. S. R. Diniz. *Adaptive Filtering.* Kluwar Academic Publishers, Norwell, Massachusetts, 2002.

[25] L. J. D'Luna *et. al.* A single-chip universal cable set-top box/modem transceiver. *IEEE Journal of Solid-State Circuits*, 34(11):1647–1660, November 1999.

[26] Berkeley Emulation Engine (BEE). http://bwrc.eecs.berkeley.edu/research/BEE, 2004.

[27] T. J. Endres, S. D. Halford C. R. Johnson, and G. B. Giannakis. Simulated comparisons of blind equalization algorithms for cold start-up applications. *International Journal of Adaptive Control and Signal Processing*, 12(3):283–301, May 1998.

[28] E. Eweda. Convergence analysis and design of adaptive filter with finite power-of-two quantized error. *IEEE Trans. on Circuits and Systems II*, 39(2):113–115, February 1992.

[29] I. Fijakow, C. F. Manlove, and C. R. Johnson. Adaptive fractionally spaced blind CMA equalization: excess MSE. *IEEE Trans. on Signal Processing*, 46:227–231, January 1998.

[30] T. Fukuoka, Y. Nakai, D. Hayashi, T. Hayashi, S. Soga, K. Fukuda, and Y. Nakakura. An area effective 1-chip QAM LSI for digital CATV. *IEEE Trans. on Consumer Electronics*, 43(3):649–654, August 1997.

[31] M. Ghosh. Blind decision feedback equalization for terrestrial television receivers. *Proceedings of the IEEE*, 86:2070–2081, 1998.

[32] R. D. Gitlin, J. F. Hayes, and S. B. Weinstein. *Data Communications Principles.* Plenum Press, New York, 1992.

[33] D. N. Godard. Self-recovering equalization and carrier tracking in two-dimensional data communication systems. *IEEE Transaction on Communications*, 28(11):1867–1875, November 1980.

[34] D. N. Godard and P. E. Thirion. U.s. pat. 4 227 152: Method and device for training an adaptive equalizer by means of an unknown data signal in a QAM transmission system, October 1980.

[35] M. Godavarti and A. O. Hero. Partial update LMS algorithms. *IEEE Transaction on Signal Processing*, 53:2382– 2399, 2005.

[36] S. Gollamudi, S Kapoor, S. Nagaraj, and Y.-F. Huang. Set-membership adaptive equalization and an updator-shared implementation for multiple channel communications systems. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 46:2372 – 2385, 1998.

[37] L. Hanzo, W. Webb, and T. Keller. *Single- and Multi-carrier Quadrature Amplitude Modulation.* John Wiley & Sons, West Sussex, England, 2000.

[38] S. Haykin. *Unsupervised Adaptive Filtering.* Wiley, New York, 1999.

[39] L. He and M. Amin. A dual-mode technique for improved blind equalization for QAM signals. *IEEE Signal Processing Letters*, 10(2):29–31, February 2003.

[40] L. He, M. Amin, and C. Reed. Adaptive equalization for indoor dynamic wireless communication channels. In *Proc. SPIE*, April 2001.

[41] L. He, M. G. Amin, Jr. C. Reed, and R. C. Malkemes. A hybrid adaptive blind equalization algorithm for QAM signals in wireless communications. *IEEE Transactions on Signal Processing*, 52(7):2058–2069, July 2004.

[42] G. Im, C. Park, and H. Won. A blind equalization with the sign algorithm for broadband access. *IEEE Communication Letters*, 5(2), Feb. 2001.

[43] W. C. Jakes. *Microwave mobile communications.* IEEE Press, Piscataway, New Jersey, 1994.

[44] C. R. Johnson, P. Schniter, T. J. Endres, J. D. Behm, D. R. Brown, and R. A. Casas. Blind equalization using the constant modulus criterion: a review. *Proceedings of the IEEE*, 86:1927–1950, 1998.

[45] A. Kalavade and E. Lee. A hardware/software codesign methodology for DSP application. *IEEE Design & Test of Computers*, 10:16–28, September 1993.

[46] M. A. S. Khalid and J. Rose. A hybrid complete-graph partial-crossbar routing architecture for multi-FPGA systems. In *Proc. ACM Int'l Symposium on FPGAs (FPGA98)*, pages 45–54, February 1998.

[47] T. Kurakake, N. Nakamura, and K. Oyamada. A blind 1024-QAM demodulator for cable television. In *Proc. Int. Zurich Seminar on Communications*, pages 136–138, February 2004.

[48] K. Kuusilinna, C. Chang, M. Ammer, B. Richards, and R. Brodersen. Designing BEE: a hardware emulation engine for signal processing in low-power wireless applications. *EURASIP Journal on Applied Signal Processing, Special Issue on Rapid Prototyping of DSP Systems*, 2003.

[49] L. R. Litwin, M. D. Zoltowski, T. J. Endres, and S. N. Hulyalkar. Blended CMA: smooth, adaptive transfer from CMA to DD-LMS. In *Proc. IEEE Wireless Communications and Networking Conference*, pages 797–800, September 1999.

[50] J. Liu and X. Lin. Equalization in high-speed communication systems. *IEEE Circuits and Systems Magazine*, pages 4–17, 2004.

[51] J. Mai and A. H. Sayed. A feedback approach to the steady-state performance of fractionally spaced blind adaptive equalizers. *IEEE Trans. on Signal Processing*, 48(1):80–91, January 2000.

[52] V. Oksman. VDSL draft specification. ANSI standards contribution T1E1.4/98 045R1, 1998.

[53] G. Picchi and G. Prati. Blind equalization and carrier recovery using a "stop and go" decision directed algorithm. *IEEE Transaction on Communications*, 35(9):877–887, September 1987.

[54] J. G. Proakis. *Digital Communications 4th Edition.* McGraw Hill, New York, NY, 2001.

[55] S. U. H. Qureshi. Adaptive equalization. *Proceedings of the IEEE*, 73(9):1349–1387, September 1985.

[56] T. S. Rappaport. *Wireless Communications 2nd Edition.* Prentice Hall PTR, Upper Saddle River, New Jersey, 2002.

[57] M. J. Ready and R. P. Gooch. Blind equalization based on radius directed adaptation. In *Proc. IEEE International Conference on Acoustics, Speech, and Signal Processing*, pages 1699–1702, April 1990.

[58] M. Rupp and A. H. Sayed. A time-domain feedback analysis of filtered-error adaptive gradient algorithms. *IEEE Trans. on Signal Processing*, 44:1428–1439, June 1996.

[59] Y. Sato. A method of self-recovering equalization for multilevel amplitude-modulation systems. *IEEE Trans. on Communications*, 23:679–682, June 1975.

[60] A. H. Sayed and M. Rupp. A time-domain feedback analysis of adaptive gradient algorithms via the small gain theorem. In *Proc. SPIE*, pages 458–469, July 1995.

[61] P. Schniter and C. R. Johnson. Dithered signed-error CMA: robust, computationally efficient blind adaptive equalization. *IEEE Trans. on Signal Processing*, 47(6):1592–1603, June 1999.

[62] W. A. Sethares, G. A. Rey, and C. R. Johnson. Approaches to blind equalization with multiple modulus. In *Proc. ICASSP*, pages 972–975, 1989.

[63] M. Shah and M. H. Kahaei. A new dual-mode approach to blind equalization of QAM signals. In *Proc. International Symposium on Computers and Communications*, pages 277–281, June 2003.

[64] B. Shen, J. Tian, Z. Li, J. Su, and Q. Zhang. A high performance QAM receiver for digital TV with integrated A/D and FEC decoder. In *Proc. ASP-DAC*, pages 1276–1279, January 2005.

[65] D. Shin, K. H. Park, and M. H. Sunwoo. A 64/256 QAM receiver chip for high-speed communications. In *Proc. IEEE International ASIC/SOC Conference*, pages 214–218, September 2000.

[66] J. Shin, J.-S. Lee, E.-T. Kim, C.-S. Won, and J.-K Kim. An improved stop-and-go algorithm for blind equalization. *IEICE Trans. Fundamentals*, E79-A(6):784–789, June 1996.

[67] J. J. Shynk. Frequency-domain and multirate adaptive filtering. *IEEE Signal Processing Magazine*, pages 14 – 37, 1992.

[68] J. J. Shynk, R. P. Gooch, G. Krishnamurthy, and C. K. Chan. A comparative performance study of several blind equalization algorithms. *SPIE*, 1565:102–117, 1991.

[69] J. J. Shynk, R. P. Gooch, D. P. Witmer, C. K. Chjan, and M. J. Ready. Adaptive equalization using multirate filtering techniques. In *Proc. Asilomar Conference*, pages 756–762, Pacific Grove, CA, 1991.

[70] B. Spitzer, M. Kuhl, and K. Muller-Glaser. A methodology for architecture-oriented rapid prototyping. In *Proc. 12th Int'l Workshop on Rapid System Prototyping*, pages 200–205, June 2001.

[71] G. L. Stuber. *Principles of mobile communication 2nd edition.* Kluwer Academic Publishers, Norwell, Massachusetts, 2001.

[72] L. K. Tan, J. S. Putnam, F. Lu, L. J. D'Luna, D. W. Mueller, K. R. Kindsfater, K. B. Cameron, R. B. Joshi, R. A. Hawley, and H. Samueli. A 70-Mb/s variable-rate 1024-QAM cable receiver IC with integrated 10-b ADC and FEC decoder. *IEEE Journal of Solid-State Circuits*, 33(12):2205–2217, December 1998.

[73] R. Tessier and W. Burleson. Reconfigurable computing for digital signal processing: a survey. *Journal of VLSI Signal Processing*, 28:7–27, 2001.

[74] L. Tong, G. Xu, and T. Kailath. Blind identification and equalization based on second order statistics: a time domain approach. *IEEE Trans. on Information Theory*, 40:340–349, March 1994.

[75] J. Treichler, I. Fijalkow, and C. R. Johnson. Fractionally spaced equalizers. *IEEE Signal Processing Magazine*, pages 65–81, May 1996.

[76] J. Treichler, M. Larimore, and J. Harp. Practical blind demodulators for high-order QAM constellations. *Proceedings of the IEEE*, 86(10):1907–1926, October 1998.

[77] J. R. Treichler and B. G. Agee. A new approach to multipath correction of constant modulus signals. *IEEE Trans. on Acoust., Speech, Signal Processing*, ASSP-31(2):459–472, April 1983.

[78] J. Villasenor and B. Hutchings. The flexibility of configurable computing. *IEEE Signal Processing Magazine*, pages 67–84, September 1998.

[79] V. Weerackody and S. A. Kassam. Dual-mode type algorithms for blind equalization. *IEEE Trans. on Communications*, 42(1):22–28, January 1994.

[80] J.-J. Werner, J. Yang, D. Harman, and G. A. Dumont. Blind equalization for broadband access. *IEEE Communications Magazine*, pages 87–93, 1999.

[81] S. Werner, M. L. R. de Campos, and P. S. R. Diniz. Partial-update NLMS algorithms with data-selective updating. *IEEE Transactions on Signal Processing*, 52:938–949, 2004.

[82] B. Widrow, J. McCool, and M. Ball. The complex LMS algorithm. *Proceedings of the IEEE*, 63(4):719–720, April 1975.

[83] B. Widrow and S. D. Sterns. *Adaptive Signal Processing.* Prentice Hall, New York, 1985.

[84] W. Wolf. A decade of hardware/software codesign. *IEEE Computer Magazine*, 36:38–43, April 2003.

[85] C. F. Wu, M. T. Shiue, C. C. Huang, and S. J. Jou. QAM/VSB dual mode equalizer design and implementation. In *Proc. AP-ASIC*, pages 323–326, 1999.

[86] K. Yamanaka, S. Takeuchi, S. Murakami, M. Koyama, J. Ido, T. Fujiwara, S. Hirano, K. Okada, and T. Sumi. A multilevel QAM demodulator VLSI with wideband carrier recovery and dual equalizing mode. *IEEE Journal of Solid-State Circuits*, 32(7):1101–1107, July 1997.

[87] J. Yang, J.-J. Werner, and G. A. Dumont. The multimodulus blind equalization algorithm. In *Proc. International Conference on Digital Signal Processing*, pages 127–130, July 1997.

[88] J. Yang, J.-J. Werner, and G. A. Dumont. The multimodulus blind equalization and its generalized algorithms. *IEEE Journal on selected areas in communication*, 20(5):997–1015, June 2002.

[89] Y. R. Zheng and C. Xiao. Simulation models with correct statistical properties for rayleigh fading channels. *IEEE Transactions on Communications*, 51:920–928, 2003.

VITA AUCTORIS

Kevin Banović was born in Windsor, Ontario, Canada, on September 19, 1980. He received his B.A.Sc. degree in electrical engineering in 2003 from the University of Windsor. He is currently a candidate in the electrical and computer engineering M.A.Sc. program at the University of Windsor. His research interests include adaptive signal processing, field-programmable logic, computer arithmetic and high performance VLSI circuit design.