FROM THEORY TO PRACTICE: EVALUATING SPARSENING FILTER DESIGNS ON A SOFTWARE-DEFINED RADIO PLATFORMS

by

Raquel Guerreiro Machado

A Dissertation

Submitted to the Faculty

of the

WORCESTER POLYTECHNIC INSTITUTE

in partial fulfillment of the requirements for the

Degree of Doctor of Philosophy

in

Electrical and Computer Engineering

November 2014

APPROVED:

Professor Andrew G. Klein, Engineering and Design Department,

Western Washington University (WSU), Primary Advisor

Professor Alexander M. Wyglinski, ECE Department, WPI

Professor George T. Heineman, Computer Science Department, WPI

FROM THEORY TO PRACTICE: EVALUATING SPARSENING FILTER DESIGNS ON A SOFTWARE-DEFINED RADIO PLATFORM

Raquel Guerreiro Machado, Ph.D.

Worcester Polytechnic Institute 2014

A comprehensive analysis of a novel detection scheme for SISO wireless transmission scenarios is presented in this dissertation. The scheme, which is based on Belief-Propagation (BP) detectors, is evaluated in both a computer simulation environment and a custom-built software-defined radio test-bed. In this dissertation, we address the design aspects of BP-based receivers, including several approaches to minimize the bit error rate of MAP detectors. We also present the development of an interface framework for a software defined radio platform that aims to implement complex communication transceivers capable of prototyping the hybrid structure with a pre-filter filter and BP detector.

Numerical simulations compared the proposed schemes with an existing approaches and showed significant performance gains without requiring great computational cost at the receiver. Furthermore, experiments using GNU Radio Companion and the FMCOMMS software defined radio hardware platform confirm the correct functionality of the proposed interface, and stress tests are conducted to assess the functionality of the interface and how it deteriorates across a range of operating conditions. Finally, we present several experiments using the FMCOMMS software defined radio platform that implement the proposed BP-based receiver scheme and discuss its capabilities and limitations.

ACKNOWLEDGEMENTS

First and foremost, all thanks go to God, lamp to my feet and light for my path.

I would like to thank the members of my PhD committee for all their guidance and support. Thank you Prof. Klein for believing in me and allowing me to come to WPI and have an amazing learning experience. You advised me during the first and most difficult years of my PhD studies; I will be forever grateful. Thank you Prof. Wyglinski for all the opportunities, the help and support during this last year; working with you was great. Thank you Prof. Heineman for all your great advice regarding the more software-related portion of my thesis. You all inspire me to be a better person and researcher.

I would also like to thank my colleagues at WPI for every discussion, every interesting conversation, every shared frustration, every pep-talk and every moment we spent together. Quingxiong, Min, Josh, Radu, Bengi, Zoe, Matt, Kalil, Aleksi, Di and Le: you will be always in my heart. Special thanks to Travis; thank you for your patience and help with GNU Radio. Very special thanks to Jen and Paulo for the amazing help toward the completion of my dissertation; I have no words to express my gratitude. And thanks to my friends in Boston for the great time, particularly Flavio and Christin.

Finally, I'm extremely grateful for my family and all their love and support during all these years. Thanks for cheering me up all the way from Brazil; I feel privileged to be your daughter and sister. And thanks to my love, my best friend and (now) husband, Andre, for being there with me in the United States and walking together the path of a PhD degree. You were essential to this success.

TABLE OF CONTENTS

Acknowledgements ii
Table of Contents iii
List of Abbreviations xii
List of Symbols xiv

1 Introduction **1**
1.1 Motivation 2
1.2 Problem Statement 6
1.3 Proposed Solutions 7
1.4 Contributions 8
1.5 Dissertation Organization 10

2 Inter-Symbol Interference Mitigation and Software Defined Radio Technology **12**
2.1 Wireless Transmission: Impairments and Mitigation 12
2.1.1 Optimal ISI Mitigation 13
2.1.2 Sub-Optimal ISI mitigation 17
2.1.3 Multicarrier Modulation 21
2.2 Software Defined Radio Technology 24
2.2.1 A Brief History 26
2.2.2 Anatomy of a Software-Defined Radio 31
2.2.3 Sampling 36
2.2.4 Current SDR Challenges 38
2.2.5 GNU Radio 40
2.3 Chapter Summary 42

3 Linear Sparsening Filter Design **43**
3.1 Background 43
3.1.1 BP detector in a Hybrid Structure 45
3.2 System Model 47
3.2.1 Receiver Structure 47
3.2.2 Sparsening Filter Design 48
3.3 Noise Coloration 53
3.4 BER and SSSNR 55
3.5 Numeric Simulations 60
3.6 Chapter Summary 67

4 Decision-Feedback Sparsening Filter Design **69**
4.1 Decision-Feedback vs Linear Equalizer 69
4.2 System Model 70
4.3 Decision Feedback Sparsening Filter 73
4.4 Numerical Results 77
4.5 Chapter Summary 80

5 Interface Architecture for Software Defined Radio Systems 81
5.1 Background . . . 81
5.1.1 FMCOMMS 1 RF Front End . . . 82
5.1.2 FMCOMMS 2 RF Front End . . . 83
5.1.3 ZedBoard and the SDR Platform . . . 88
5.1.4 Interface with GNU Radio . . . 92
5.2 Interface Development . . . 94
5.2.1 Source Block . . . 95
5.2.2 Sink Block . . . 99
5.2.3 Complete Interface Experimentation . . . 101
5.3 Chapter Summary . . . 106

6 Sparsening Filter Experimentation in SDR Platform 107
6.1 Experimentation in Baseband . . . 108
6.1.1 Experiment Set-Up . . . 108
6.1.2 Numerical Simulations . . . 111
6.2 Experimentation in RF Frequencies . . . 113
6.2.1 Proposed Experiment with Channel Emulators . . . 114
6.2.2 Experiment with Simulated Channel . . . 116
6.2.3 Numerical Simulations . . . 118
6.3 Chapter Summary . . . 120

7 Conclusions and Future Work 121
7.1 Concluding Remarks . . . 121
7.2 Future Work . . . 122
7.3 Peer Review Publications . . . 122

Bibliography 124

LIST OF FIGURES

1.1 The evolution of data requirements in cellular networks. Adapted from [1] 2

1.2 Intersymbol interference phenomenon forming the channel delay profile 5

2.1 High level digital transmission schematic with the equalizer filter. . 13

2.2 The $h = [1 \quad 0 \quad -1]$ channel represented as a factor graph. The graph is formed by bit nodes, which are the channel inputs ($x[\blacksquare]$) and the check nodes, which are the channel outputs ($y[\blacksquare]$). The connections between bit nodes and check nodes represent the dependencies between the received and transmitted symbols. 17

2.3 Decision-feedback equalizer structure with feedforward and feedback filters and decision device. 20

2.4 Block diagram representing a multicarrier transmission system. For the transmission, the information is first parallelized to be modulated and in the receiver, the information is also parallelized to be demodulated and detected. 22

2.5 Orthogonal subcarriers in a OFDM scheme. The subcarriers are separated by a Δf frequency spacing. 23

2.6 Processing technology vs. SDR technology. The timeline of the past decades shows how the evolution of different types of processors had a great impact in the development of SDR platforms. 27

2.7 The Universal Software Radio Peripheral 2 (USRP2). The software defined radio platform developed by Ettus Research. 29

2.8 The KUAR radio. Source: [2] 30
2.9 Block Diagram showing the Digital and Analog divide in a Software-Defined Radio Platform. The digital signal processing is performed in the digital domain in baseband, while the analog portion of the system performs the RF operations. Based on [3], Figure 5. 33
2.10 Block diagram of a typical RF Front-End. In the transmitter path, the the analog signal is modulated and transmitted in RF frequencies. In the receiver path, the analog high-frequency signal is converted to baseband before being processed by the ADC. The local oscillator (LO) drives both transmitter and receiver circuits. . . . 36
2.11 The GNU Radio Companion (GRC) tool. It allows design of communication systems using pre-made and custom blocks. 41

3.1 System Model 47
3.2 Contours of Sparse Shortening SNR (SSSNR) for a three-tap unit norm filter, parameterized by two angles in spherical coordinates. The $\times$ indicates the filter with the highest SSSNR. 56
3.3 Channel, filter, and effective channel tap magnitudes, using the SSSNR-optimal 3-tap filter. 56
3.4 Usage maps of the 6 taps of the effective channel $\mathbf{c}$. Axes are identical to Fig. 3.2. Dark areas indicate that the given tap is one of the μ largest taps in $\mathbf{c}$. 57
3.5 Contours of bit error rate (BER) for a three-tap unit norm filter, parameterized by two angles in spherical coordinates. The $\triangledown$ indicates the filters with the lowest BER. 58

3.6 Comparison of different optimal channel sparsening filters (CSFs), overlaid on contours of the surface (3.7) with $\beta = 1$. 58
3.7 Fig. 3.6, zoomed in on the lower right section. 59
3.8 Fig. 3.6, zoomed in on the middle section. 59
3.9 Bit error rate as a function of β for the weighted SSSNR scheme . 60
3.10 Bit error rate for different CSF design metrics 61
3.11 Bit error rate for same CSF design metrics shown in 3.10 plus the addition of the squared autocorrelation penalizing term. 63
3.12 CSF frequency response for combinatorial, combinatorial with addition of penalizing term, and Roy's approach. 65
3.13 Frequency response of original channel and effective channel frequency responses for combinatorial, combinatorial with addition of penalizing term, and Roy's approach. 66
3.14 Bit error rate curves for the Vehicular A channel 66

4.1 System Model 71
4.2 Impulse responses 76
4.3 Bit error rates for the Vehicular A channel 78
4.4 Bit error rates for 5-tap channel with equal power delay profile . . 79

5.1 The FMCOMMS1 module is a RF front-end (RFFE) board. Here we have the board and markers highlighting important components of the platform, such as the ADC, DAC, (de)modulators and amplifiers. Source: [4], used with permission. 83
5.2 The FMCOMMS2 is a high-speed analog module designed by Analog Devices. Source: [5], used with permission. 85

5.3 Signal paths in the FMMCOMMS2 software-defined radio platform. Inside the AD9361, both TX and RX sections are composed by two signal paths, one for each channel (I and Q). 87

5.4 Functional Schematics of the hardware of the SDR Platform. On left, we have the ZedBoard and on right we have the FMCOMMS1 RF Front-End. The figure illustrates the transmit and receive chain as well as the functional blocks that are in radio transmission using this platform. Source: [4], used with permission. 89

5.5 Schematic of the software-defined radio hardware. On left, we have the FPGA development board and on right we have the FMCOMMS2 RF Front-End. The figure illustrates the transmit and receive chain. Source: [5], used with permission. 90

5.6 High level illustration of the transmitting/receiving processes using the Ethernet connection in the ZedBoard. The sink/source blocks in GNU Radio communicate with the IIO drivers in the Zedboard. 93

5.7 High level illustration of the onboard transmitting/receiving processes. The sink/source blocks in GNU Radio communicate with the IIO drivers inside the same platform. 94

5.8 Experiment Set-Up. The points (a), (b) and (c) show the probing positions for the obtained measured signals in the Results subsection 95

5.9 GNU Radio Flow Graph 96

5.10 Plots obtained by probing the FMCOMMS1 board as indicated in Figure 5.8. The transmitted/received signal is shown at different stages of the radio transmission. 97

5.11 View of the oscilloscope application in GNU Radio. The GUI shows the signal sampled from the ADC and processed to be compatible with the GNU Radio environment. In this figure, we see a time domain plot of the signal with a measured frequency of 3.9447 MHz 99

5.12 View of the spectrum analyzer application in GNU Radio. The GUI shows the signal sampled from the ADC and processed to be compatible with the GNU Radio environment. In this picture we see a frequency domain plot of the signal with a measured frequency of 3.96753MHz MHz 100

5.13 GNU Radio environment for test of the sink block. 100

5.14 Test set-up for the sink block. The signal is generated in GNU Radio, passed through the RF front-end board and then analyzed using ADI IIO oscilloscope tool. 100

5.15 ADI IIO Oscilloscope measurement. 101

5.16 Transmitter and Receiver configurations in GNU Radio. The signal is generated and modulated in DBPSK symbols before being sent to the "Fmcomms sink". The signal goes through the transmitting and receiving paths in the hardware platform and the received I/Q are delivered back to GNU Radio through the "Fmcomms source" block. 103

5.17 Transmitted and received DBPSK symbols sent through the platform. The transmitted symbols are generated and modulated in GNU Radio and sent through the transmit chain. The symbols are then fed back to the receive chain and can be visualized using the ADI IIO oscilloscope tool in the ZedBoard and inside the GNU Radio environment. 104

6.1 Experiment using the SDR platform with simulated wireless channel 108

6.2 Schematics for the baseband experiment with the FMCOMMS1 board. The RF portion of the board is disconnected from the DAC and ADC outputs and the signal is redirected to go from the DAC output to the ADC input. 109

6.3 FMCOMMS1 board in baseband configuration. The output of the DAC is connected to the input of the ADC using SMB cables. . . . 109

6.4 The baseband signal received at the ADC. The BPSK symbols are scattered along the X axis making the detection operation unfeasible. 110

6.5 The baseband signal received at the ADC. The BPSK symbols are showed as expected and the constellation can be easily identified. . 111

6.6 Symbol error rates for multipath channel. 112

6.7 Symbol error rates for vehicular channel. 112

6.8 Experiment using the SDR platform with channel emulator. 114

6.9 PropSim channel emulator at the Center for Wireless Information Network (CWIN) Lab. 115

6.10 Experiment using the SDR platform with simulated channel. 117

6.11 Pre-processing required for experiment. At the top we have the pre-processing that occurs in the transmission. At the bottom we have the pre-processing in the reception. 118
6.12 The received BPSK symbols. In (a) the received constellation collected from the ADC's buffer. In (b) we have the constellation after the phase compesantion and in (c) after the square root raised cosine filter. 119
6.13 Symbol error rates for multipath channel. 119
6.14 Symbol error rates for vehicular channel. 120

LIST OF TABLES

2.1 Summary of optimal algorithms for ISI compensation. 17

2.2 Summary of sub-optimal algorithms for single carrier ISI compensation. 21

2.3 Summary of SDR platforms. 32

2.4 Sampling Capabilities of different SDR platforms. 38

3.1 Computational Complexity, Taps Selected, and SSSNR Achieved at 8 dB SNR 62

5.1 Listing of the FMCOMMS1 module hardware components present in both transmit and receive paths, with respective specifications. We show the part numbers for each component and the descriptions and specifications. Source: [4] 84

5.2 Listing of the FMCOMMS2 key features [5]. 86

5.3 Percentage of samples lost during the transmission of 10^7 samples . 105

LIST OF ABBREVIATIONS

Abbreviation	Meaning
ADC	analog to digital converter
AWGB	additive white Gaussian noise
BER	bit-error rate
BP	belief propagation
BPSK	binary phase-shift keying
CSF	channel sparsening filter
DAC	digital to analog converter
DBPSK	differential BPSK
DFE	decision-feedback equalization
DFSF	decision-feedback sparsening filter
DMA	digital memory access
DSP	digital signal processor
FPGA	field-programmable gate array
FB	feedback
FF	feedforward
FMCOMMS	RF front-end developed by Analog Devices
FIR	finite impulse response
IIO	industrial input/output
IIR	infinite impulse filter
ISI	intersymbol interference
LMMSE	linear minimum mean squared error
LMS	least mean square
LTE	linear transversal equalizer
MAP	maximum a posteriori
MLSE	maximum-likelihood sequential estimation
MMSE	minimum mean square error
MSE	mean square error
OFDM	orthogonal frequency-division multiplexing
RFFE	RF front end
RLS	recursive least-squares
SER	symbol error rates
SISO	soft-input soft-output
SDR	software defined radio
SSSNR	sparse shortening SNR
SNR	signal-to-noise ratio
TIR	target input response
USRP	Universal Software Radio Peripheral

LIST OF SYMBOLS

Symbol	Meaning
$\star$	convolution operation
$\mathbf{0}_{m\times n}$	$m \times n$ matrix of all 0's
$\mathbf{1}_{m\times n}$	$m \times n$ matrix of all 1's
$\boldsymbol{I}_n$	$n \times n$ identity matrix
$\lvert\cdot\rvert^2$	absolute value squared
$(\cdot)^\top$	matrix transpose
$(\cdot)^H$	matrix conjugate transpose
$[\boldsymbol{S}]_i$	ith column of matrix $\boldsymbol{S}$
$[\boldsymbol{S}]_{i,j}$	i,jth entry of matrix $\boldsymbol{S}$
$\lvert\lvert\boldsymbol{x}\rvert\rvert_p$	ℓ_p norm
$E[\cdot]$	Expectation

Chapter 1

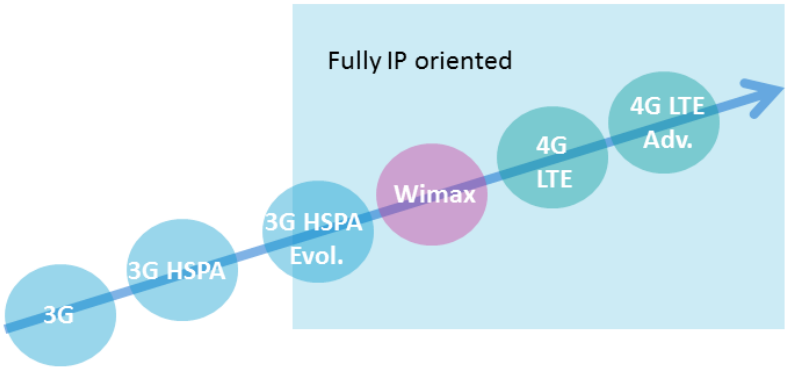
Introduction

The pursuit of instantaneous unlimited access to information has fueled research into designing communication systems capable of achieving high data rate transmissions, great mobility and good spectral efficiency. At present, the advancement of wireless networks is partly focused on increasing data rates in order to guarantee the required quality of service (QoS) for activities such as real-time video streaming and online gaming [6, 7]. The challenge of transmitting reliable and high-rate data over a wireless channel is significant, since a practical communication system has to compensate for phenomena that may prevent the correct detection of transmitted information [8, 9].

A significant amount of wireless communications research is focused on the development of optimal algorithms for mitigating phenomena that hinder wireless communications. However, many of these algorithms rely on theoretical assumptions concerning the properties of the transmitted data or implementations that do not necessarily translate to a real-world system. It is therefore of great importance to be able to combine theoretical development with practical experimentation. The ability to design and test novel theoretical schemes using practical hardware facilitates wireless communications research. Thus, the development of test platforms is important for the study of theoretical schemes applied to wireless communications systems.

1.1 Motivation

The evolution of mobile communications shows the increasing importance of data transmission in the requirements of the services provided to the users. Figure 1 illustrates achievable data rates provided by recent cellular technologies as well as the increasing data rates requirements. According to Qualcomm [10], in 2017 two-thirds of mobile traffic is going to be related to video content. This will require higher data rates in order to satisfy the quality of service requirements for such applications.



Market impact	2003	2008	2009	2010	2016
Peak rate	384 Kbps	7 Mbps	43 Mbps	~150 Mbps	~1Gbps
Typical user rate downlink	~200 Kbps	1-2 Mbps	2-10 Mbps	10-20 Mbps	~30-100 Mbps
Typical user Rate uplink	64 Kbps	64-884 Mbps	0.5-4.5 Mbps	5-10 Mbps	~ 10-60 Mbps

Figure 1.1: The evolution of data requirements in cellular networks. Adapted from [1]

As already mentioned, transmitting wireless information reliably is challeng-

ing due to the phenomena that interfere with the transmission and hinder correct reception. The phenomena that prevent the correct detection of transmitted information are refereed to as impairments. Three major types impairments are path loss, shadowing and inter-symbol interference (ISI) [8]. Path loss occurs due to the dissipation of a portion of the power of the signal when it is traveling through space, while shadowing is a form of power attenuation that occurs from objects that block the signal between the transmitter and the receiver. These two problems can be mitigated using power control [8] and intelligent cell-planning [11]. ISI is a phenomenon in which one symbol interferes with subsequent symbols, making the receiver unable to identify which symbol was transmitted. However, ISI is an intricate problem to solve and a large body of research has investigated different methods to overcome this impairment. Moreover, solving the ISI problem is of great importance to enable the higher data rates required by modern mobile communications.

The ISI phenomenon constitutes a significant challenge for wireless systems that aim at exchanging information at very high data rates. It is usually caused by multipath fading or the transmission of a signal through bandlimited channels. Multipath fading is caused by the multiple reflections that may occur in the signal path between the transmitter and receiver [8]. Bandlimited channels have a variable frequency profile which may distort the transmission of information and prevent the receiver of recognizing what was transmitted.

As shown in Figure 1.2, trees, buildings, and the ground can serve as reflectors. The reflected components superpose constructively and destructively at the receiver will yield a collection of copies of the signal arriving with different strengths

and at different time instants. As the objects between the transmitter and receiver may also change location, the profile of signal copies can change with time. Additionally, due to different propagation times, the differences in arrival time of the responses from the longest and shortest path (related to as delay spread [12]), may take over multiple symbol durations. When the delay spread of the wireless channel is longer than the duration of the symbol being transmitted, the signal suffers from inter-symbol interference. This impairment is considerable in systems with a very high data rate, where as the rate of transmission increases, the time duration of the symbol decreases and the ISI phenomenon becomes more severe. Consequently, the frequency response of a channel with ISI present within the signal passband varies significantly. In this situation, the received signal suffers from frequency selective fading, and the wireless channel can be modeled as a finite impulse response (FIR) filter in discrete time, in which each coefficient is modeled as a random variable [12].

To transmit over a bandlimited channel, it is necessary to shape the digital signal with an analog pulse using a bandwidth that is limited to that of the channel in order to not loose frequency components that are cutoff by the channel. This process is referred to as pulse shaping [13]. If the analog pulse width is larger than the symbol period, the adjacent symbols might overlap causing the ISI effect. To mitigate the ISI phenomenon on bandlimited channels, pulse shaping must be implemented with Nyquist pulses [12] that contain only values from the desired input symbol at the sampling instants, and no interference from other symbols.

The issue of compensating for ISI has been studied extensively over the past five decades, and a wide range of strategies are available for use by communication

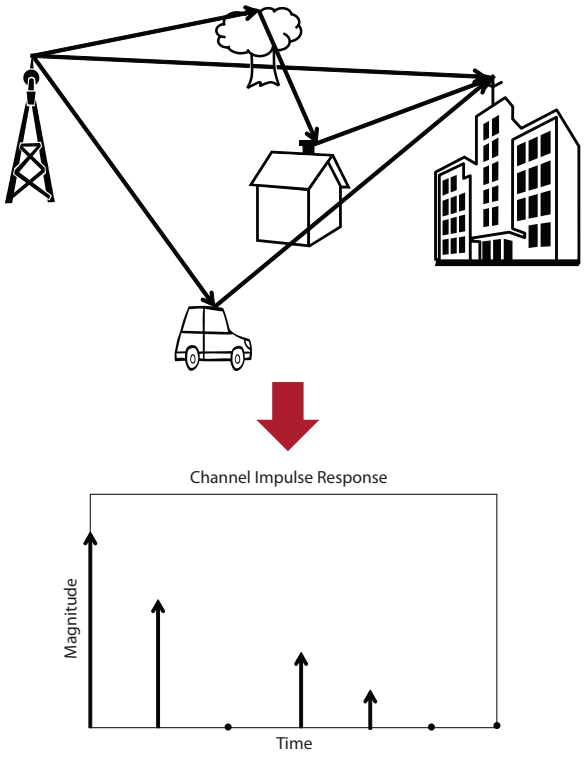


Figure 1.2: Intersymbol interference phenomenon forming the channel delay profile

system designers [14]. The seminal paper that discusses the signaling of transmissions over bandlimited channels was written by Nyquist [15]. Other papers discussed the mitigation of ISI, using pulse shaping [16] and joint transmitter and receiver designs for pulse amplitude modulation [17, 18]. For multipath channels, linear equalizers were proposed in [19, 20] and [21] to combat the resulting ISI, and a decision-feedback structure was proposed in [22]. Even though these techniques are not computationally complex, the achieved ISI compensation is sub-optimal and limited.

Finally, the maximum a posteriori (MAP) [23] and maximum-likelihood (ML) sequence estimator [24] can be employed to mitigate ISI with optimal performance. However, one problem with these optimal approaches lies in their complexity:

they are exponentially complex in the number of channel coefficients and thus cannot be used to combat ISI in channels with a long delay spread, *i.e.* a large number of channel coefficients. Examples of environments that produce channels with long delay spread include underwater acoustic communication and terrestrial communication over hilly terrain.

1.2 Problem Statement

In 2005, a MAP detector employing Belief-Propagation (BP) was proposed in [25] for ISI compensation in sparse channels, which are mainly characterized as having only a small fraction of nonzero coefficients. The proposed scheme is attractive because it permits near-optimal performance with complexity that depends only on the number of nonzero coefficients. A hybrid version of this detector was proposed in [26], and it uses a linear pre-filter in the receiver just before the BP-based MAP detector. In this thesis, the prefilter that precedes the BP detector is referred to as a sparsening filter. Furthermore, in [27] a low-complexity high-throughput architecture of this hybrid structure was presented using a digital signal processor (DSP) and a field-programmable gate array (FPGA).

The idea presented in [26] is very interesting, since by designing the prefilter such that the combined response of the sparse channel and prefilter has a reduced, limited number of nonzero coefficients, the complexity of the receiver can be controlled. However, in this work relatively little attention is paid to the interaction of the sparsening filter and the BP-based detector and only a simplistic method for designing the sparsening filter is provided. For example, the sparsening filter

is arbitrarily designed such that the channel coefficients, or taps, of the combined response of the sparse channel and sparsening filter coincide with the dominant taps in the original channel. Also, most analyzes performed for BP-based structures has been provided in light of numerical simulations and only a few hardware implementations have been attempted [27, 28].

Given the level of reconfigurability required by the prefilter design, coupled with the inherent complexity of the belief propagation algorithms, a software-defined radio (SDR) solution presents itself as a reasonable candidate for rapid testing and prototyping. Several software defined radio (SDR) architectures have been developed and are commercially available in order to enhance the prototyping phase of new wireless technologies as well as advance the current state-of-the-art in wireless and networking communications systems. While there exists several well-known SDR platforms that are commercially available [2, 29–34], many of these systems are designed primarily for conducting fundamental research and experimentation, and not for development of actual commercial products and prototypes. In this sense, the existence of SDR prototyping platforms and their accompanying software interfaces is the key for enabling continued advances in the wireless sector. In addition, it is of utmost importance better understand both the capabilities and limitations of SDR technology when used in real-life communications systems.

1.3 Proposed Solutions

In this thesis, we propose several novel design approaches for implementing belief propagation (BP)-based detectors. The main idea is to design a prefilter so that

the combined response of the sparse channel and prefilter has a reduced, limited number of nonzero coefficients, thus controlling the complexity of the receiver. In this work, the prefilter that precedes the BP detector has been termed a sparsening filter. We present two different techniques for the design of sparsening filters. The first one relies on a linear structure for the sparsening filter and uses a new metric also proposed in this work to design it. The second technique proposes a decision feedback structure as a sparsening filter.

Once these theoretical designs and numerical simulations have been explored, the next step is to conduct hardware experiments in a real-world environment. Thus, one of the contributions of this work includes the development of a custom-built interface framework for a software defined radio platform that aims to provide a design and testing environment for complex communication transceivers capable of prototyping receivers such as the hybrid structure with a sparsening filter and BP detector.

1.4 Contributions

The main contributions of this dissertation are as follows.

- Linear Sparsening Filter Design:

  We propose a filter design metric called the Sparse Shortening SNR, and showed that maximizing this quantity serves as a good proxy for minimizing BER. We also develop a greedy algorithm for tap selection, which provides near-optimal performance with reduced complexity. Finally, we take into

consideration the issue of noise coloration introduced by the sparsening filter to design the desired prefilter.

- Decision-Feedback Sparsening Filter Design:

  We propose a filter design metric based on classical DFE design, which improved in both performance and complexity when compared with the Linear Sparsening Filter Design. Once again the interaction of the sparsening filter and BP detector is considered in the filter design generating better error performance.

- Interface Architecture for Software Defined Radio Systems:

  We present an interface architecture that enables software connectivity and support for the FMCOMMS boards, a family of RF front-ends developed by Analog Devices (ADI), and GNU Radio, a software environment for the design systems that involve digital signal processing. We also provide experiments using GNU Radio Companion and the FMCOMMS hardware platform that attest the correct functionality of the proposed interface. Finally, we characterize the proposed interface in terms of supported sampling frequency and data throughput.

- Sparsening Filter experimentation in a SDR platform:

  We provide different experimentation environments and results for the implementation of the designed BP-based receptors in the FMCOMMS SDR platform.

1.5 Dissertation Organization

This dissertation is organized as follows.

Chapter 2 presents a literature survey and tutorial of several topics covered in this dissertation with the purpose of contextualize the subsequent Chapters and their focus of discussion. The chapters explains the different types of impairments that hinder wireless communications transmissions, with focus on inter-symbol interference (ISI). It also discuss several ISI mitigation schemes already existent in the literature. Finally, a tutorial on software defined radio is provided contextualizing the usage of the technology over the years and its importance on experiments design and prototyping.

Chapter 3 focuses on the design of linear *sparsening* prefilters for use with soft-input soft-output MAP detectors of the form considered in [25, 26]. While [25, 26] primarily focused on the case where the original channel is sparse, it is noted that even nonsparse channels can be sparsened with a simple linear, finite impulse response (FIR) filter. Consequently, this work can be applied in general situations, even where the original channel is not sparse. The issue of sparsening filter design is addressed with the goal of minimizing a metric designed to be a proxy for the detector bit error rate BER. The interaction of the sparsening filter and BP detector has been considered to develop a practically-implementable sparsening filter design method.

Chapter 4 focuses on the design of decision feedback *sparsening* filters for use with soft-input soft-output MAP detectors of the form considered in [25, 26]. In particular, the hybrid structure proposed by [26] is extended so that the linear

sparsening filter which performs partial equalization is replaced by a non-linear decision feedback sparsening filter.

Chapter 5 details the development of an interface framework between an radio platform and the GNU Radio SDR development environment in order to enable software support for the FMCOMMS family of RF front ends. This work expands the initial prototype interface framework started at ADI by enhancing the functionality of the work in order to facilitate seamless connection between a FMCOMMS SDR platform and GNU Radio. In addition to discussing the hardware platform and the proposed software interface framework, we provide experiments that attest the correct functionality of the SDR platform and that test the interface performance limits.

Finally, Chapter 6 includes the description of the experiments using the FMCOMMS SDR platforms on the implementation of the proposed linear and decision-feedback sparsening filters, presented in Chapters 3 and 4.

In Chapter 7, the research achievements of this work are summarized and topics for future work are presented.

Chapter 2

Inter-Symbol Interference Mitigation and Software Defined Radio Technology

In this chapter, we present a literature survey and tutorial of several topics covered in this dissertation. In the first section, we discuss the wireless transmissions impairments and the different techniques used in the literature for inter-symbol interference (ISI) compensation; these concepts are important and aid in the better understanding of Chapters 3 and 4 of this thesis. In addition, we provide a tutorial on software defined radio technology to complement the Chapters on software interface implementation and experimentation.

2.1 Wireless Transmission: Impairments and Mitigation

As already mentioned in Section 1.1, transmitting reliable and high-rate data over a wireless channel is challenging due to physical phenomena that make the reception and recognition of the transmitted information a difficult task. In this sense, one of the chief impairments faced by modern, high data-rate communication receivers is called inter-symbol interference (ISI) [11].

In order to be able to correctly detect the symbols that were transmitted, it is necessary to compensate for the ISI introduced by the channel. The role of equalization in a digital transmission is to mitigate the inter-symbol interference phenomenon by attempting to undo the scattering provided by the wireless channel

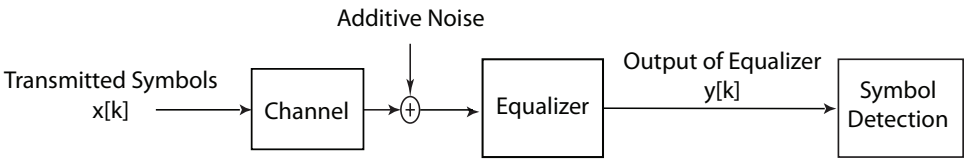


Figure 2.1: High level digital transmission schematic with the equalizer filter.

to the received symbols [35]. By doing so, the recovery of transmitted symbols becomes feasible and the receiver is able to detected and recognized the symbols correctly. Figure 2.1 shows a high-level digital transmission schematic with the equalizer filter.

The issue of compensating for ISI has been studied at length over the past five decades, and a wide range of strategies are available for use by communication system designers. In the following, we present several techniques for ISI compensation and discuss their properties.

2.1.1 Optimal ISI Mitigation

Given that the ISI channel can be modeled as a FIR filter, it can also be modeled as a finite-state machine, which can be represented by a trellis structure. In this case, the sequence of symbols can be represented by a path through the trellis and the problem of correctly deciding the symbol that was transmitted reduces to the finding the correct path in this trellis. Maximum a posteriori (MAP) or maximum-likelihood (ML) sequence estimators are algorithms that provide optimal detection performance and are discussed next.

The maximum likelihood sequence estimation (MLSE) algorithm finds the most

likely sequence corresponding to the received symbols [12]. In other words, the algorithm searches through the trellis that is equivalent to the ISI channel and finds the most probable path though the trellis. To calculate the likelihood of a determined path, the MLSE calculates the distance between the sequences; if the demodulator performs hard decisions, the Hamming distance metric is used, but if the demodulator performs soft decisions, the Euclidean distance metric is implemented.

The Viterbi algorithm [36] was developed by Andrew J. Viterbi and was originally designed to decode information coded using convolutional codes, but it was Forney [37] and Omura [38] that proposed the use of the Viterbi algorithm as the optimal maximum-likelihood sequence estimator for ISI mitigation. The Viterbi algorithm is a sequential trellis algorithm that reduces the number of sequences tested in the trellis, by eliminating sequences at each stage of the trellis. For example, if the ISI channel has a length of $L+1$ symbols and the information symbols are M-ary, the channel is described by an M^L-state trellis. We begin with L samples, and compute M^{L+1} metrics and M^{L+1} which are divided in M^L groups. In each group, one sequence is selected (the sequence with the largest probability) and $M-1$ are discarded, generating M^L surviving sequences and their metrics. As it can be noticed, the number of computations required by each stage of the algorithm grows exponentially with the length of the ISI channel L, limiting the usage of the Viterbi algorithm for small L.

The BCJR algorithm [39], named after its inventors (Bahl, Cocke, Jelinek, and Raviv), was also originally developed for convolutional codes. It uses a symbol-by-symbol maximum *a posteriori* (MAP) decoding algorithm to decode each input

symbol instead of searching for the most likely sequence as the MLSE estimators. In this sense, the BCJR finds the most likely individual bits or symbols in addition to values for the *a posteriori* probability $P(x|\boldsymbol{y})$, where x is the desired bit or symbol and $\boldsymbol{y}$ is the received sequence. This probability determines the level of certainty regarding the estimation of the bit or symbol x and are called soft outputs; for this reason, the BCJR is also called a *soft-input soft-output* (SISO) decoder. As the Viterbi decoder, the BCJR algorithm's complexity also increases exponentially with the length of the ISI channel [12].

Belief Propagation Detector

The belief propagation (BP) algorithm is also in the class of message passing algorithms, and is sometimes called the *sum-product algorithm* [25]. It can be used as the Viterbi and BCJR algorithm as symbol detector, compensating for ISI and allowing for correct reception of transmitted information. By representing the ISI channel as a factor graph, we can use the BP algorithm to implement MAP detection by estimating the sequence of symbols that maximizes the joint a posteriori probability mass function. In Figure 2.2, we have an example in which the channel $h = [1 \quad 0 \quad -1]$ is represented as a factor graph. The graph consists of bit nodes, which represent the channel inputs ($x[\blacksquare]$), and check nodes that represent the channel outputs ($y[\blacksquare]$). In this example, the output symbols can be written as:

$$\begin{aligned} y[0] &= -1 * x[0] = -x[0] \\ y[1] &= -1 * x[1] + 0 * x[0] = -x[1] \\ y[2] &= 1 * x[2] + 0 * x[1] + -1 * x[0] = x[2] - x[0] \end{aligned} \tag{2.1}$$

$$y[3] = 1 * x[3] + 0 * x[2] + -1 * x[1] = x[3] - x[1]$$

$$y[4] = 1 * x[4] + 0 * x[3] + -1 * x[2] = x[4] - x[2].$$

Note that in the graph, the connections between the channel input symbols and the channel output symbols represent the dependencies between the received and transmitted symbols (the negative signs are not represented in the factor graph). For example, check node $y[0]$ is only connected to bit node $x[0]$, since $y[0] = -x[0]$, check node $y[1]$ is connected to bit node $x[1]$, since $y[1] = -x[1]$, check node $y[2]$ is connected to bit nodes $x[2]$ and $x[0]$, since $y[2] = x[2] - x[0]$ and so fourth.

The BP algorithm proceeds iteratively, exchanging information between check nodes and bit nodes [26]. As already mentioned, the algorithm is in the class of message passing algorithm because the probabilist information about the received symbols is passed from check nodes to bit nodes. In this process, log likelihood ratios of the transmitted bits are computed and become more reliable with each iteration. After a sufficient number of iterations, the log likelihood ratios can be used to make bit decisions. In the case of the BP detector, the order in which these message are passed occur according the flooding schedule [25], where nodes pass the information to their neighbors subsequently.

If the BP algorithm proceeds over N total iterations, the total complexity requires on the order of $N(\mu + 1)M^{\mu+1}$ summations, where M is the size of the source alphabet and μ is the number of significant effective channel taps used in the detection. As such, the complexity of the BP is exponential in μ, and so the system designer can specify the total complexity by appropriate choice of μ. This interesting property is the main reason for this thesis to focus specifically on using the BP algorithm as symbol detector for ISI compensation. Table 2.2 provides a

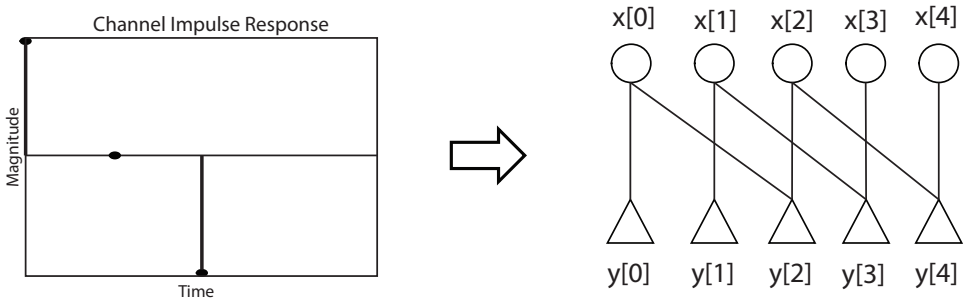


Figure 2.2: The $h = [1 \quad 0 \quad -1]$ channel represented as a factor graph. The graph is formed by bit nodes, which are the channel inputs ($x[\blacksquare]$) and the check nodes, which are the channel outputs ($y[\blacksquare]$). The connections between bit nodes and check nodes represent the dependencies between the received and transmitted symbols.

summary of the presented optimal approaches for ISI compensation.

Table 2.1: Summary of optimal algorithms for ISI compensation.

Algorithm	Principle	Complexity
Viterbi	MLSE	Exponential on the delay spread
BCJR	MAP	Exponential on the delay spread
BP	MAP	Exponential on # of non-zero taps

2.1.2 Sub-Optimal ISI mitigation

As the MLSE and MAP detectors have computational complexities grows exponentially with the length of the channel time dispersion, their usage with most real-world channels is very limited. Even for the BP detector, which has the advantage of the complexity growing only on the number of non-zero channel coefficients, the computational complexity can be prohibitively costly. For these reasons,

suboptimum channel equalization approaches were made necessary to mitigate ISI.

Linear Equalizers, are filters designed with a linear structure. In terms of structure, the linear transversal equalizer (LTE) is formed by tapped delay lines and it can be implemented as a finite impulse response (FIR) filter or as an infinite response (IIR) filter [40], which are the most used structures for linear equalizers. Other possible linear structures are the linear transversal equalizer [41] and the lattice equalizer [42]; these structures have good numerical stability, but also have too complicated structures. In terms of the filter's coefficients, there are different ways of choosing their values, such as the peak distortion criterion and the mean square error criterion [12].

The peak distortion criterion [19, 20] seeks the minimization of the worst-case inter-symbol interference effect on the received symbols. This goal is achieved when the equalizer's transfer function is the inverse of the channel's model; the ISI mitigation happens because the filter completely eliminates the effect of the channel on the symbols. This filter is called zero-forcing filter. In other words, if we have a channel with transfer function $H(z)$, the corresponding zero-forcing filter is $\frac{1}{H(z)}$, which eliminates the channel's filtering effects leaving only a multiplying constant gain. Although simple to calculate, the zero-forcing presents 2 important issues. The first one is that the inverse of a channel modeled as a FIR filter is an IIR filter, which is not practical to build. The second is that it can potentially enhance the additive noise; if the channel contains a null in its frequency response, the zero-forcing equalizer produces a spike to compensate for the null and will end up amplifying the noise which is also filtered.

For the mean square error (MSE) criterion [21], the filter is designed to minimize

the mean square value of the error signal, which is the signal at the output of the filter minus the transmitted signal. The calculation of the filter coefficients then becomes an optimization problem in which the cost function is [12]:

$$J = E[(y[k] - x[k])^2], \tag{2.2}$$

where $y[k]$ is the filter output and $x[k]$ are the transmitted symbols. After solving this optimization problem, the minimum of the cost function has the following transfer function:

$$\frac{1}{H(z) + N_0}, \tag{2.3}$$

which is very similar to the zero-forcing expression, with exception of the noise spectral density N_0. In this sense, the equalizer coefficients are adjusted to minimize both the MSE due to the ISI and the noise power at the filter's output. However, the minimum mean square error equalizer (MMSE) still has the same problem as the zero-forcing regarding the infinite length required for optimal cancellation, which can not be implemented in practice but only approximated by FIR filters.

Other than linear equalization, there are several non-linear structures that are also used in the design of equalizers. The decision-feedback equalizer [22] is one of such filters. This type of equalizer is usually formed by two components: the feedforward and the feedback filters as showed in Figure 2.3. The received symbols are inputed to the feedforward filter and its output is summed to the output of the feedback filter in order for the decision device to provide hard decisions on the symbols. These decisions are delayed and then inputed to the feedback filter; the feedback filter removes part of the ISI on the current symbol using the decisions on previous numbers.

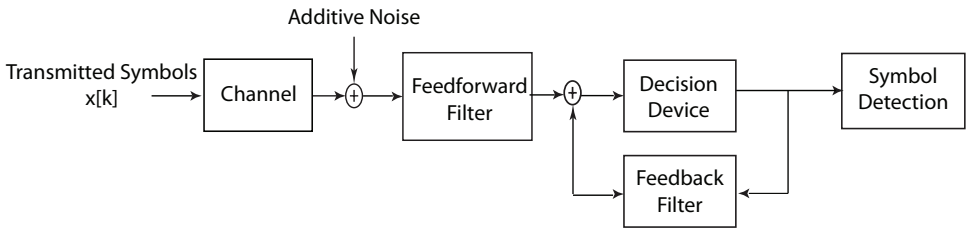


Figure 2.3: Decision-feedback equalizer structure with feedforward and feedback filters and decision device.

In comparison, DFE are able of performing ISI compensation with reduced noise enhancement and may thus provide significantly lower symbol error rates (SER) than a linear equalizer. On the other hand, due to the nonlinear feedback structure of DFE's, symbol errors induced by high noise may cause instability due to the feedback of wrong decisions. This phenomenon called error propagation, although of small probability, may also lead to poor error performance.

The equalizers mentioned so far all have a static structure, which are preset to mitigate channels that are invariant in time. Another category of equalizers are the adaptive equalizers, in which the filter coefficients are updated while the data is being processed [43]. In other words, the coefficients are automatically adapted using a pre-determined optimization metric and known transmitted symbols (for training), to be adjusted to the channel and even adapted to it if the channel is time-varying. There are several well-known algorithms for adaptive equalization, such as the least-mean-square (LMS) [44], the adaptive decision-feedback equalizer [45], the recursive least-squares (RLS) [46] and the Kalman filter [47]. In addition to traditional adaptive filtering, another technique of adaptive equalization was also extensively studied in the literature, the blind equalization. In these algorithms, the transmitted signal is equalized using only signal statistics, and no training

symbols. Among many blind equalization algorithms, the most well known is the constant-modulus algorithm (CMA) [48].

Table 2.2: Summary of sub-optimal algorithms for single carrier ISI compensation.

Equalizer	Principal Characteristics
Linear	Simple, has problems with noise enhancement
DFE	Better performance than linear, error propagation problems
LMS, RLS and Kalman	Linear adaptive filtering
CMA	Blind equalization

2.1.3 Multicarrier Modulation

All the methods presented so far are associated with a single carrier modulation scheme. This means that the information is transmitted using a single slot of frequency; all the information is contained in a specific frequency range. However, with single carriers, higher data rate requirements end-up resulting in problems with ISI. Higher data rates require smaller symbol periods, which in turn result in ISI if the symbol period is smaller than the channel's delay spread (see Section 1.1). Another way to deal with the ISI problem is to use a multicarrier modulation data transmission scheme [49].

The idea behind multicarrier modulation is to subdivide the available channel in many subchannels so the frequency response in each one of these subchannels is approximately constant [12]. This facilitates the equalization process and provides a solution to the ISI problem for high data rated systems. The multiple carriers

allow the symbol period to be larger than the channel's delay spread, still maintaining higher effective rates distributed over non-overlapping frequency bands. Figure 2.4 shows a multicarrier communication system. In the transmitter, the symbols are parallelized and then modulated; after the multicarrier modulation, the information is converted to a serial sequence to be transmitted over-the air. In the receiver, the received symbols are parallelized to pass through demodulation and detection.

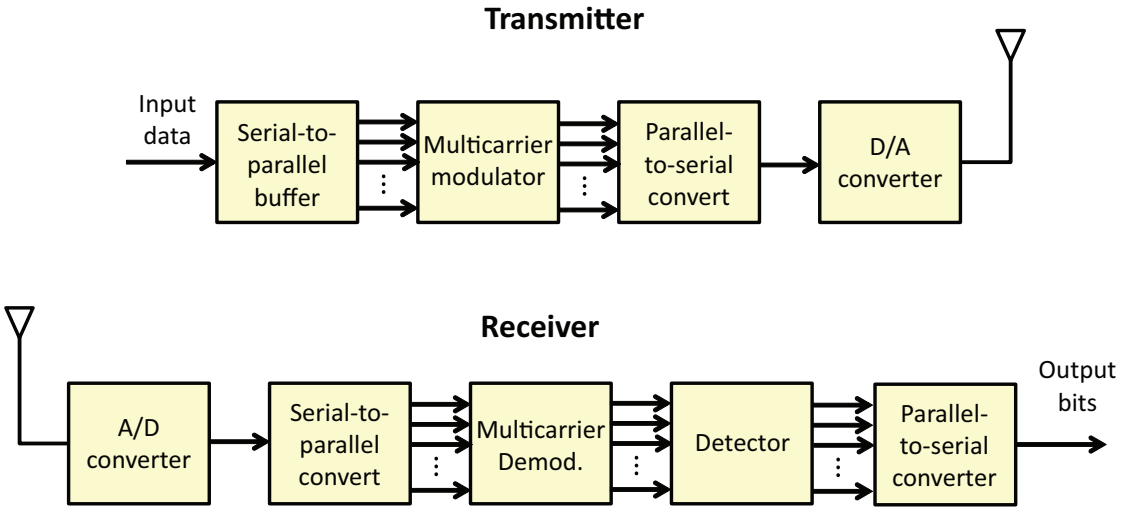


Figure 2.4: Block diagram representing a multicarrier transmission system. For the transmission, the information is first parallelized to be modulated and in the receiver, the information is also parallelized to be demodulated and detected.

The orthogonal frequency-division multiplexed (OFDM) scheme [50, 51] is a special type of multicarrier modulation in which the subcarriers are orthogonal to each other. In other words, the information is transmitted using multiple subcarriers in different frequencies; the frequency spacing of the carriers is chosen so the modulated carriers become orthogonal and therefore do not interfere with each other. Figure 2.5 shows the orthogonal subcarriers. Most modern communications

systems use the OFDM scheme for digital transmission: the wireless LAN (WLAN) radio interfaces, digital radio systems, the terrestrial digital TV systems DVB-T and ISDB-T, ultra-wideband (UWB) and others. Moreover, the OFDMA [52], a OFDM-based multiple access technology was also used in several cellular networks, specially in the downlink operation.

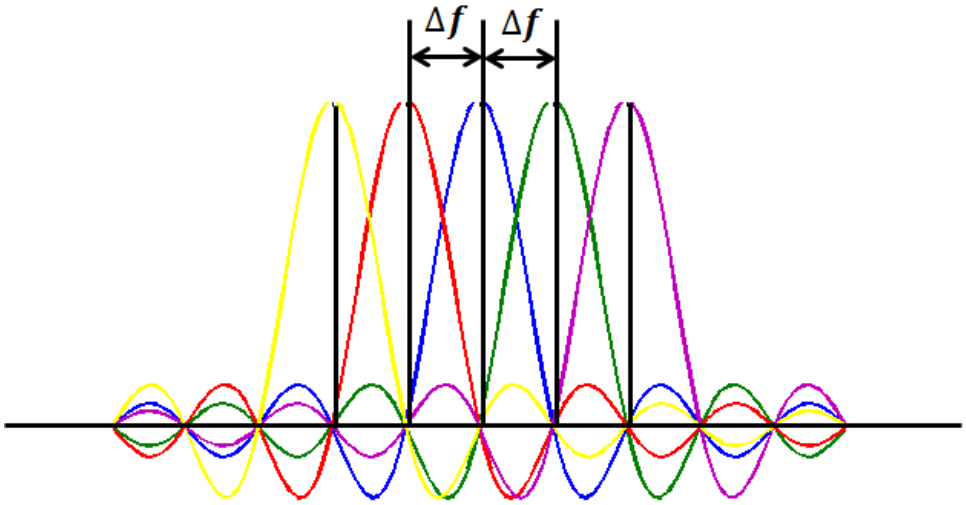


Figure 2.5: Orthogonal subcarriers in a OFDM scheme. The subcarriers are separated by a Δf frequency spacing.

One of the greatest disadvantages of the OFDM scheme is the fact that it suffers from a high peak-to-average-power-ratio (PAPR), which requires linear circuitry in the transmitter. The problem with this type of circuit is that it is not power efficient and it makes the power manager in small devices, such as mobile handsets, very difficult. For this reason, the uplink operation of cellular communication is implemented in a single carrier scheme; more specifically, the single-carrier frequency division multiplex access (SC-FDMA) scheme [53]. It was adopted in the uplink operation in the 3GPP Long Term Evolution (LTE) and 4G. In this work, we focus on single-carrier schemes to combat the ISI impairment.

2.2 Software Defined Radio Technology

Advances and innovation within the wireless sector have always been closely coupled to corresponding improvements in digital technology, including computing devices. Until the 1950s [54–56], wireless systems were exclusively operating in the analog domain, where various communications functions such as modulation and filtering were performed using analog circuits and components. As a result, the process of designing a robust communication system was time-consuming and costly since analog circuit designers were needed to devise systems that met specifications and were difficult to mass produce at a scale that could enable wide-spread penetration within a large consumer market. With the rapid evolution of digital technology, especially analog-to-digital and digital-to-analog converters (ADCs and DACs), it now became possible to perform these same baseband communication functions partially or entirely within the digital domain, greatly reducing cost, enabling mass production of these transceivers, and providing a greater a flexibility and system functionality. Consequently, when communication systems transcended the analog/digital divide, this became a defining moment of the Information Age and the enabler of ubiquitous wireless data access that todays society has grown accustomed to over the years.

The first wireless devices that employed digital technology were based on non-programmable, static designs realized using application-specific integrated circuits (ASICs). These implementations enabled wireless devices, such as cellular telephones and wireless local area networking modems, to eventually be produced on a large, commercial scale at a cost that would make these systems reasonably affordable in a consumer market. As various computing technologies began to

mature, such as digital signal processors (DSPs), they also began being incorporated into the baseband digital implementation of these wireless systems. However, these computing devices were programmed with a static set of operations to be performed by the wireless system, such as filtering, data compression, modulation, and other baseband operations.

Lately, the widely spread use of wireless mobile devices has presented great potential challenges in the area of wireless services provision as different standards can be used on the same device depending on the circumstances of use. Each of these radio standards require their own specific access terminal and base station infrastructure creating the need for installing and maintaining a plethora of specific equipments. The use configurable radio technologies the rises as a possibility, requiring an evolution from static programmed operations. This new concept permits providing an infrastructure from which service providers can evolve to meet the needs of the users without heavy re-investment in infrastructure.

As mentioned, the evolving wireless networks requirements in reconfigurable technology led in the past few years to the diffusion of the so-called Software Defined Radio (SDR) architectures. To conduct research in this area, it is necessary to possess sufficient knowledge in both hardware platforms and software environments. The background needed includes the different SDR platforms as well as some information on software development environments.

2.2.1 A Brief History

The term software radio was introduced by Joseph Mitola in 1992 [57]. However, a SDR prototype had already been presented in 1988 by Hoeher and Lang [58]. The establishment of SDR as a technology came with the first publicly funded SDR development initiative, called SpeakEasy I/II by the U.S. military [59]. The first generation of the SpeakEasy system initially used a Texas Instruments TMS320C40 processor (40 MHz), while the SpeakEasy II platform was the first SDR platform to involve field programmable gate array (FPGA). Later, the U.S. Navy developed the digital modulator radio (DMR), a platform with many waveforms and modes that could be remotely controlled with an Ethernet interface.

As Figure 2.6 shows the timeline of the evolution of both processing technology and SDR technology, it is possible to notice the necessity of developing components with higher computational power and flexibility to enable better SDR platforms. It was only after the year 2000, with powerful FPGAs and DSPs, that most of the existing platforms were developed. More recently, the ARM Cortex A9 opened the possibility of accessible on-board processing, discarding the necessity of a host computer for system development.

In the late 1990s SDR started to spread from the military domain to the commercial sector, with cellular networks being considered the natural area of application [60]. Several Companies such as Vanu [61], Airspan [62] and Etherstack [63] started to develop SDR products for cellular base stations. In 2005, Vanu released in 2005 the first SDR product approved by the software radio regulation: the Anywave^{TM} GSM base station. The BTS (base transceiver station), the BSC

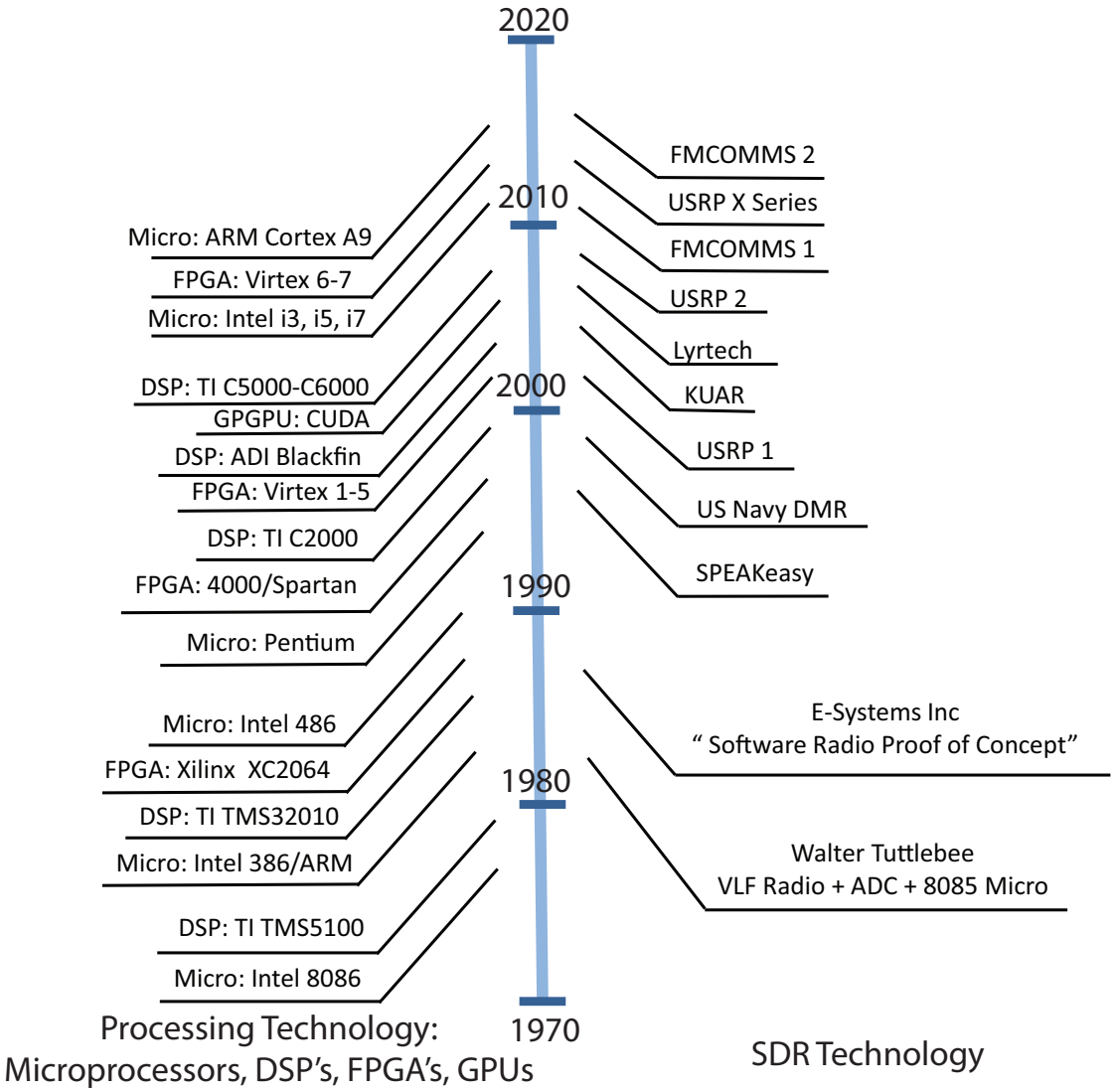


Figure 2.6: Processing technology vs. SDR technology. The timeline of the past decades shows how the evolution of different types of processors had a great impact in the development of SDR platforms.

(base station controller), and TRAU (transcoder and rate adaptation unit) modules of the BSS (base station subsystem) were implemented in software in the Anywave base station. Although this successful implementation brought substantial attention to SDR technology and it was thought at the time that SDR base stations would be key for 3G networks, the reality regarding the commercial usage

of SDR technology was still somewhat distant.

One of the most commonly used SDR hardware platform is the Universal Software Radio Peripheral (USRP) [29]. Developed by Ettus Research LLC, the USRP is a device that turns general purpose computers into flexible SDR platforms. The core of the USRP is a motherboard with four high-speed ADCs and DACs and a FPGA. The ADCs/DACs are connected to the radio Front-Ends (called daughterboards), while the FPGA is connected to a general purpose computer. In the Universal Software Radio Peripheral - Version 1 (USRP1) this connection is performed by a USB port, while the USRP2 (showed in Figure 2.7) includes a Gigabit ethernet interface. The main principle behind the USRP is that the digital radio tasks are divided between the internal FPGA and the external host CPU. The high speed general purpose processing, like down and up conversion, decimation, and interpolation are performed in the FPGA, while waveform-specific processing, such as modulation and demodulation, are performed at the host CPU. The USRP platform can be used with both GNU radio and MATLAB software development environments. More recently, Ettus Research released the new X Series, a platform that contains more powerful daughterboard slots, 6 GHz with up to 120 MHz of baseband bandwidth, and a large user-programmable Kintex-7 FPGA.

Another SDR hardware platform is the Kansas University Agile Radio (KUAR) [2], showed in Figure 2.8. The KUAR platform was designed to be a low-cost experimental platform targeted at the frequency range 5.25 to 5.85 GHz and a tunable bandwidth of 30MHz. The platform contains a Xilinx Virtex-II Pro FPGA board and a PCI Express 1.4 GHz Pentium-M microprocessor. With these features, almost all processes can be implemented in the platform, instead of the host com-

Figure 2.7: The Universal Software Radio Peripheral 2 (USRP2). The software defined radio platform developed by Ettus Research.

puter, which minimizes the host-interface requirements. In addition, the KUAR utilizes a modified form of the GNU Radio software framework to complete the hardware platform.

With respect to compact SDR platforms, the Maynooth Adaptable Radio System (MARS) [31] was designed to be connected to a personal computer which handles all of the signal processing algorithms. Another objective was to deliver a performance equivalent to a base station and the wireless communication standards in the frequency from 1700 to 2450 MHz. The software framework selected for initial development was the IRiS framework (Implementing Radio in Software).

Some other SDR platforms include:

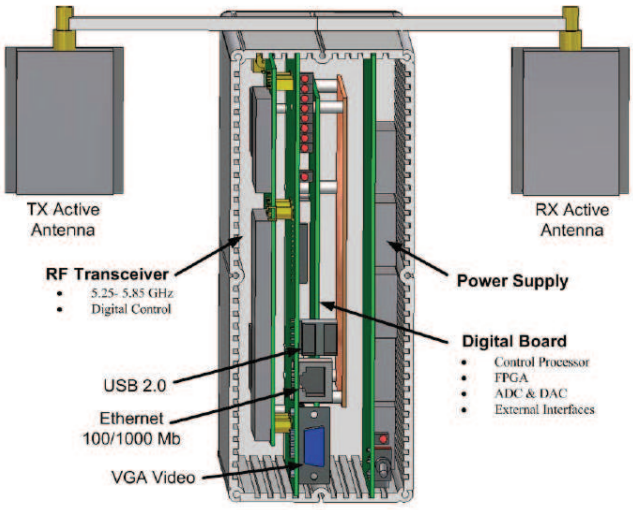


Figure 2.8: The KUAR radio. Source: [2]

- Berkeley BEE2 [64]: has five Xilinx Virtex-II Pro FPGAs on a custom-built emulation board.

- Japanese National Institute of Information and Communications Technology (NICT) SDR Platform [65]: contains two embedded processors, four Xilinx Virtex2 FPGA, and RF modules that could support 1.9 to 2.4 and 5.0 to 5.3 GHz.

- Rice University Wireless Open Access Research Platform (WARP) [32]: radios include a Xilinx Virtex-II Pro FPGA board as well as a MAX2829 transceiver.

While these SDR platforms are mainly used for research and experimentation, the final goal of developing SDR systems capable of implementing modern communication protocols remains far for reality. The development of new hardware platforms and accompanying software interfaces is key to advances in the area and to better understand the capabilities and limitations of the SDR technology when

used in real-life communications systems.

Lately, a few companies have also been developing different SDR solutions for use in academia and industry. For example, Nutaq [33] developed two main SDR products, the ZeptoSDR and the PicoSDR, both of which support RF frequencies between 300 MHz and 3.8GHz and bandwidth of 1.5 to 28 MHz. The ZeptoSDR uses a Xilinx Zynq-7 and an embedded ARM Cortex-A9, and the PicoSDR uses a Xilinx Virtex-6 and an embedded Quad-Core i7. Epiq Solutions [34] also developed two compact SDR products, the Sidekiq and the Matchstiq, as well as another platform called Maveriq.

Analog Devices Inc. (ADI) has been in the SDR market since the 1990s, and serves numerous customers in this area. Consequently, in order to obtain a better understanding of the needs of both current and prospective customers, one approach is to learn more about SDR as an application by exploring the actual application/protocol using the product, to better understand how to optimize the RF front-end and SDR components for a specific application, and to provide the customers with a real working system before they employ it themselves. For these reasons, ADI developed the FMCOMMS1 [4] and FMCOMMS2 [5] RF Front-Ends, which are going to be the focus of this thesis.

2.2.2 Anatomy of a Software-Defined Radio

In traditional radios, all radio functionalities are performed by specialized components that execute specific functions, such as modulators/demodulators and coding/decoding. In this case, all signal processing is performed within these spe-

Table 2.3: Summary of SDR platforms.

SDR Platform	Processing Device	Sampling and Bandwidth
USRP2	Xilinx Spartan 3A-DSP 3400	100 MS/s ADC, 400 MS/s DAC
USRP X Series	Xilinx Kintex-7	120 MHz of bandwidth
Kuar	Xilinx Virtex-II Pro	bandwidth of 30MHz
MARS	Personal Computer	1700 to 2450 MHz
BEE2	5 Xilinx Virtex-II Pro	-
NICT	Xilinx Virtex2	1.9 to 2.4 and 5.0 to 5.3 GHz
ZeptoSDR	Zedboard	1.5 to 28 MHz
PicoSDR	Virtex-6	3.8 GHz, 28 MHz BW
Maveriq	Spartan 6 LX150T	DAC and ADC 50 MHz
FMCOMMS1	ZedBoard	250 MSPS ADC and 1 GSPS DAC
FMCOMMS2	Zedboard	640 MSPS ADC and 320 MSPS

cialized hardware. The software-defined radio technology replaces some of the traditional radio components with components implemented in software.

A software-defined radio transceiver is divided into two main parts: (i) an analog Front-End, which performs the narrowband frequency downconversion followed by an Analogue-to-Digital Conversion (ADC), and (ii) the digital signal processing components, which are responsible for the remaining signal processing flow [66]. Thus, operations such as (de)modulation, filtering, and channel (de)coding are performed in the digital domain. Figure 2.9 shows the typical data flow in a software-defined radio system. In this case, nearly the entire baseband signal processing on both the transmission and receiving ends is performed in the software

domain.

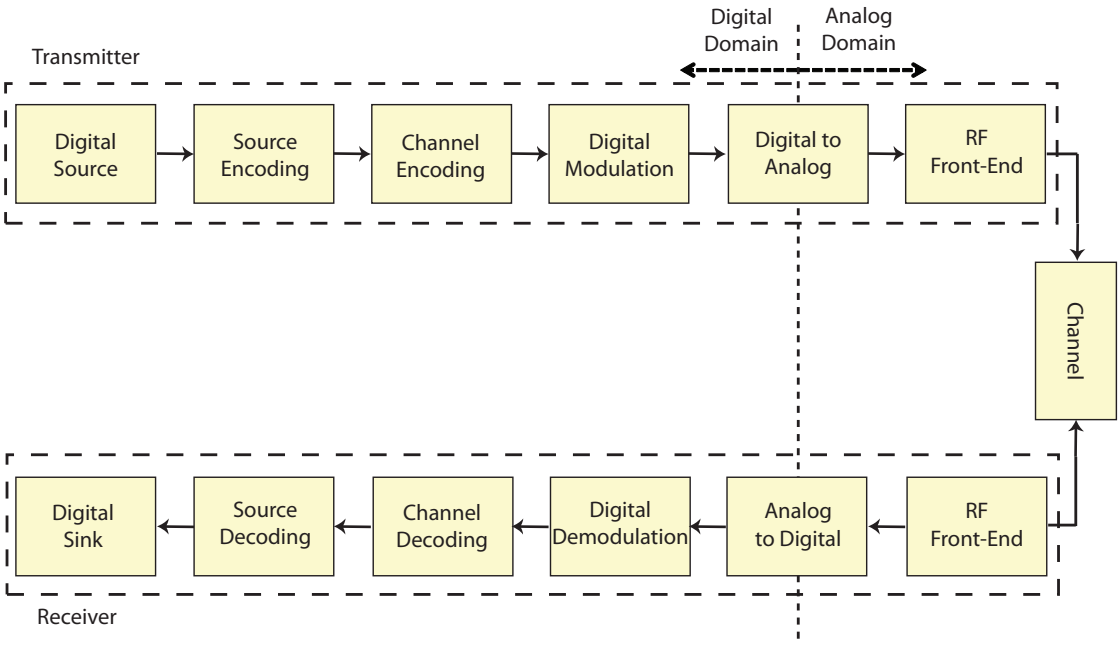


Figure 2.9: Block Diagram showing the Digital and Analog divide in a Software-Defined Radio Platform. The digital signal processing is performed in the digital domain in baseband, while the analog portion of the system performs the RF operations. Based on [3], Figure 5.

Ideally, if the Analog-to-Digital/Digital-to-Analog conversion can be pushed further into the RF block, the programmability could be extended to the RF front end and an ideal software radio could be implemented [60]. The advantage of having components implemented in software is flexibility, as different frequency bands, air interface protocols, and functionalities could be upgraded through a software download instead of having to completely replace the hardware.

Thus, the ultimate goal for software-defined radio is to move the AD/DA conversion as close as possible to the antenna so that all signal processing can be done digitally. However, some technical limitations make it currently infeasible to

perform the AD/DA conversion at the antenna.

Digital Domain

Figure 2.9 shows the functional blocks that can be implemented in the digital domain of a communication system. These blocks include modulation and demodulation blocks, which perform mapping between bits and electromagnetic waveform characteristics; coding/decoding blocks, which help mitigate impairments in the wireless channel; source encoding and decoding blocks, which remove redundant information from the binary data; and channel encoding and decoding blocks, which introduce redundant information to protect transmissions from potential errors.

In an SDR platform, all of these components are implemented in software and can run in different processing venues including field programmable gate arrays (FPGAs), graphics processing units (GPUs), digital signal processors (DSP), general purpose processors (GPP), or a combination thereof. While FPGAs are computationally powerful, they are power inefficient and inflexible, and it is difficult to implement new modules in them. Similarly, GPUs are very computationally powerful but are difficult to use and implement new modules into. DSPs are processors that perform specialized mathematical computations. While users can implement new modules into them with relative ease and they are relatively power efficient, they are not well suited for computationally intensive processes and can quickly lose speed. Finally, GPPs are a popular solution for SDR implementations and prototypes due to their high level of flexibility with respect to reconfigurability. However, since GPPs are not specialized for mathematical computations, they can be very power inefficient.

Returning to the problem of where the AD/DA conversion should happen, there are a number of challenges presented by the transition from hardware radio to software radio. First, transition from hardware to software processing results in a substantial increase in computation, which results in increased power consumption and reduced battery life. This large power consumption is one of the key reasons why software-defined radios have not been deployed in end-user devices but are instead used in base stations and access points, which can take advantage of external power resources. Second, the question of where the AD/DA conversion can be performed determines what radio functions can be done in software, and hence how reconfigurable a radio can be.

Analog Domain

Figure 2.10 shows a typical RF Front-End responsible for processing the analog portion of the digital transmission [67, 68]. In the transmission signal path, the digital samples are converted into analog signal by the DAC to be input to the RF Front-End; the analog signal is later mixed with high frequency carriers and modulated to a determined RF frequency and transmitted over the air. In the receiving signal path, the RF signal is captured by the antenna and brought back to base band to be processed by ADC. The RF mixing and modulation is driven by the local oscillator (LO), which generates the RF signal, which is mixed with the incoming signal. Another very important component used in radio transmission is the Low-noise amplifier (LNA), which is usually located close to the antenna and is used to amplify weak signals without significantly increasing noise level.

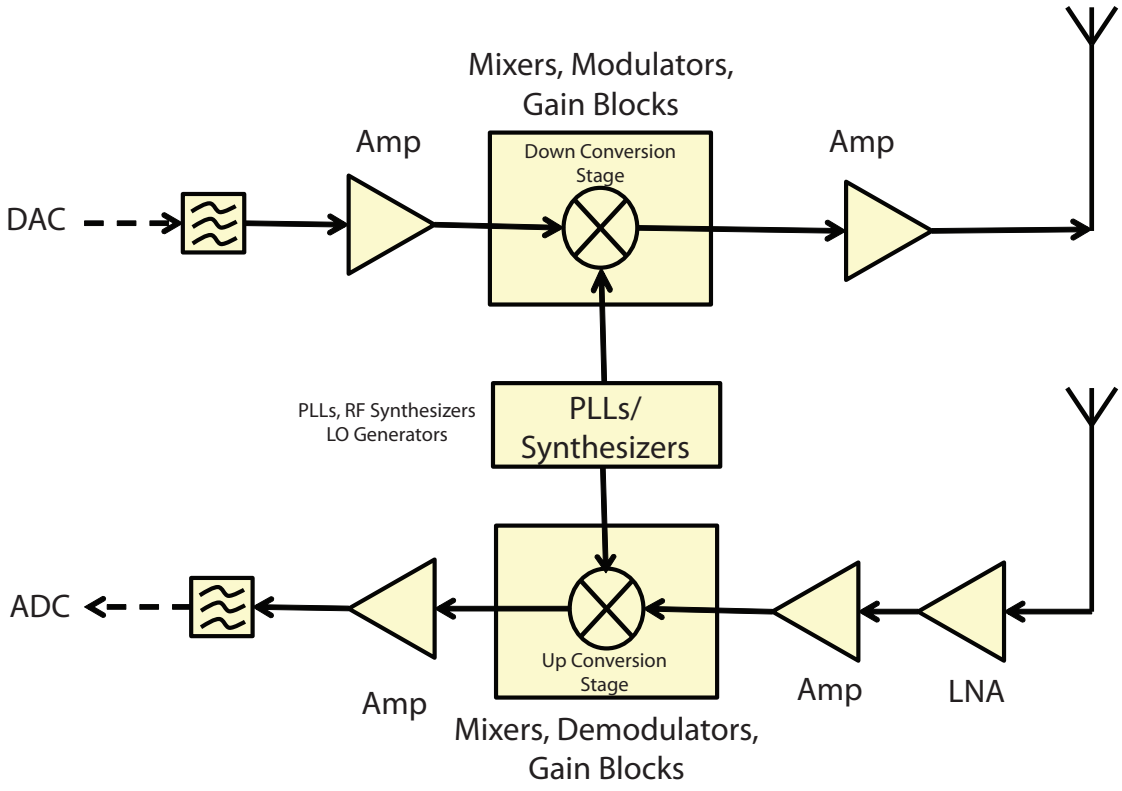


Figure 2.10: Block diagram of a typical RF Front-End. In the transmitter path, the the analog signal is modulated and transmitted in RF frequencies. In the receiver path, the analog high-frequency signal is converted to baseband before being processed by the ADC. The local oscillator (LO) drives both transmitter and receiver circuits.

2.2.3 Sampling

Digital transmissions are all about sampling. A continuous-time signal can be converted to a discrete-time signal using sampling, and a discrete-time signal can also be converted to a continuous-time signal using reconstruction [69, 70]. To sample a signal, instantaneous measurements are taken every T_s seconds; in this sense, T_s is the sampling period, and $f_s = 1/T_s$ is the sampling frequency. To reconstruct the original signal from the sampled signal, it is necessary to apply a

low-pass filter on the sampled signal. However, by the Nyquist theorem, for the reconstruction to be successful f_s needs to be higher than 2 times the analog signal's bandwidth, which is called the Nyquist frequency. The components responsible for sampling and reconstructing the signal are the Digital-to-Analog converter (DAC) and the Analog-to-digital converter (ADC).

As already mentioned, moving the analog-to-digital and digital-to-analog conversions closer to the antennas is the ultimate goal of software-defined radio technology. In order to do so, the major challenge lies in the DAC and ADC's sampling capabilities. To digitize an RF signal it is necessary to sample it at least at the Nyquist frequency, and the higher the data rate of the signal, the higher the resolution required to capture the information. For example, an 802.11n Wi-Fi channel is 40 MHz wide [71], which means the ADC has to digitize 80 MHz of signal bandwidth, resulting in a sample rate of at least 160 million samples per second (Msps).

For these reasons, the development of SDR platforms is closely related to the development of more powerful ADCs and DACs. In Table 2.4, we provide a list of SDR platforms and their corresponding sampling capabilities. It is possible to note that the advent of ADCs and DACs with higher maximum sampling values contributed to the rapid development of several SDR platforms in sequence. From the SPEAKeasy platform in the 1990s to the FMCOMMS 2 and USRP-X Series, the possible usable bandwidth increased 3000 times. This allows the implementation of modern wireless transmission standards in SDR platforms and pushes the edge of the software defined radio technology applications.

Table 2.4: Sampling Capabilities of different SDR platforms.

SDR Platform	Sampling Capabilities
SPEAKeasy	200 Kb/s
USRP 1	64 MS/s dual ADC and 128 MS/s dual DAC
KUAR	105 MSPS ADC and 100 MSPS DAC
USRP 2	100 MS/s dual ADC and 400 MS/s dual DAC
FMCOMMS 1	250 MSPS ADC and 1 GSPS DAC
FMCOMMS 2	640 MSPS ADC and 320 MSPS
USRP-X Series	640 MSPS ADC and 320 MSPS

2.2.4 Current SDR Challenges

As already mentioned, the sampling capabilities of ADs and DAs converters are a key ingredient for the implementation of SDR systems and prototypes. The ability to digitize high frequencies is fundamental for bridging the analog domain with the digital domain, and to leverage the flexibility made available by the digital hardware and software. On the other hand, the software and digital logic implementation imposes a computation burden on the platform and therefore an increase in power consumption. This trade-off leads to the usage of different SDR solutions for different applications thus forcing the designer has to decide which trade-off is more important: flexibility or energy efficiency.

A second consideration concerns interface to the RF portion of the digital transceiver [72]. It is challenging to design antennas over a wide range of frequencies since the antennas propagate signals differently for different frequencies

transmitted signals. In addition, the electronic circuits that connect the antennas to the rest of the circuit, which are called baluns, are also optimized for different antennas and should be matched for optimal power performance. This complicates the radio design, and prevents the implementation of systems with very difficult frequency ranges on the same SDR platform. Usually it is necessary to chose between baluns that have an excellent linear response on a narrow frequency band or baluns that have reasonable response on a broad frequency range.

Another issue related to sampling rates is the timing and synchronization required within the radio [72]. It is important that the rates of the processing devices (Microprocessors, FPGAs, DSPs) which are running the digital signal processing blocks are synchronized with the clock of the hardware components in the analog domain of the digital transmission. The hardware and software clocks should be equivalent and translatable to avoid confusion and mismatch in the sampling rates used. However, this is a major concern to software environment designers, but should be thus transparent to the users.

Other than technical issues, a major challenge related to SDR implementations focuses around the software environments to be used for the system design. Two commonly used software tools for the design and prototyping of SDR implementations are MATLAB [73] and GNU Radio [74]. MATLAB is a versatile software tool with substantial user support and many different modules and processing blocks already available for usage. On the other hand, MATLAB is proprietary and requires an initial investment in the designing process. On the other hand, GNU Radio is a free software toolkit that provides a flexible environment for the design of communication systems. Even though there are not as many blocks available

as is MATLAB, it is possible to change the pre-existing and develop new ones in C++ or Python. This increased flexibility can also be problematic since the learning curve to utilize GNU Radio and its capabilities is steep, as it requires considerable programming experience.

2.2.5 GNU Radio

GNU Radio [74] is a free software toolkit licensed under the GPL for implementing software-defined radios. Initially, it was mainly used by amateur radio enthusiasts, but it later gained significant interest from wireless researchers, and today it has a large community of users and contributors. It supports Linux natively, and packages are pre-compiled for the major Linux distributions.

GNU Radio provides means for performing the digital signal processing portion of a communication system design. Several software algorithms include filters, channel codes, synchronization elements, equalizers, demodulators, decoders, and many other elements. It is possible to use these components as building blocks of a communication system; GNU radio provides not only provides these blocks but also a method of connecting them together. Communications systems can be implemented by using the already available blocks or by developing new components for the software platform.

In addition to be able to design applications using scripts, GNU Radio also provides a graphical tool called GNU Radio Companion (GRC). The GRC is similar to MATLAB's Simulink and allows designers to develop systems using pre-made blocks as well as designing their own if necessary. It facilitates having a system

view of project and also diminishes the barrier of entrance for GNU radio users. Figure 2.11 shows a communications system being designed using the GRC tool.

The data management is performed by the software environment and is transparent to the user. In this sense, both the input data type for receivers and output data type for transmitters are complex baseband samples. In general, GNU Radio applications are primarily written using the Python programming language. The blocks, on the other hand, can be also written in Python but are mainly implemented in C++ due to performance reasons.

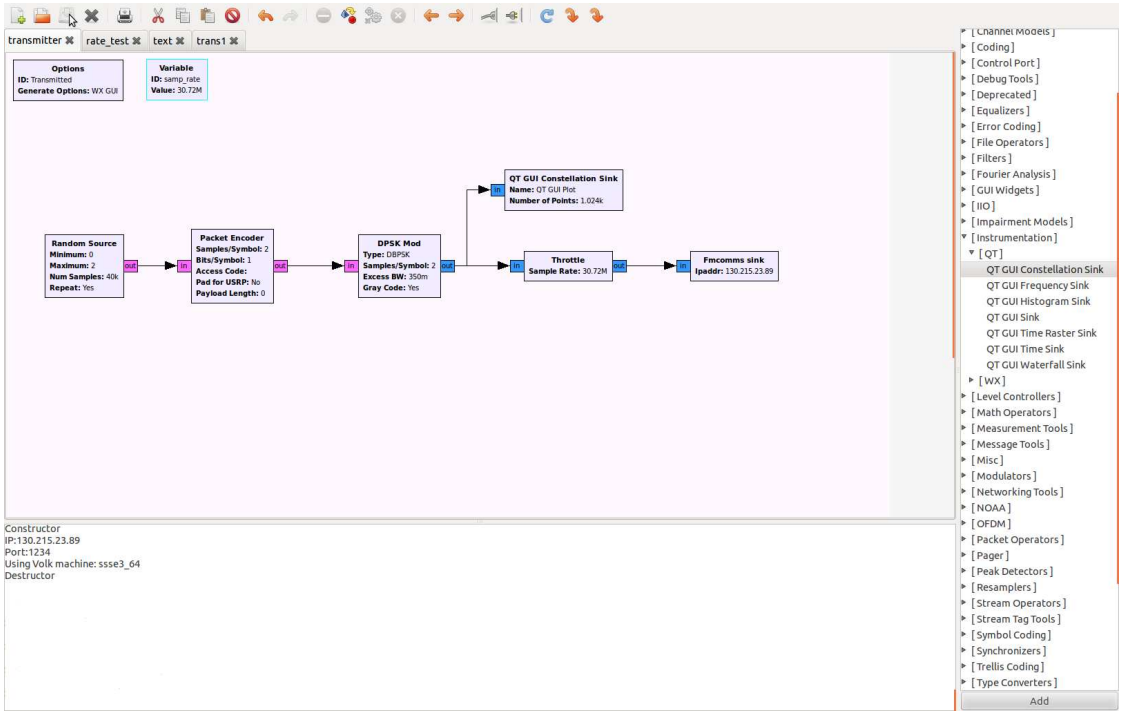


Figure 2.11: The GNU Radio Companion (GRC) tool. It allows design of communication systems using pre-made and custom blocks.

2.3 Chapter Summary

In this chapter, we discussed the impairments that hinder wireless communications, focusing on inter-symbol interference. We explained optimal and suboptimal techniques for ISI compensation and pointed to the usage of the BP algorithm as detector with feasible computational complexity. We also discussed the evolution of SDR technology in the past few decades some of the current challenges that prevent the spread usage of software-defined radio in commercial applications. As the processing technologies and the sampling components become more powerful, higher data rates can be achieved and the implementation of modern communications standards for different applications become feasible.

Chapter 3

Linear Sparsening Filter Design

A large body of research exists around the idea of channel shortening, where a prefilter is designed to reduce the effective channel impulse response to some smaller number of contiguous taps. This idea was originally conceived to reduce the complexity of Viterbi-based maximum likelihood equalizers. In this chapter, we consider a generalization of channel shortening which we term "channel sparsening". In this case, a prefilter is designed to reduce the effective channel to a small number of nonzero taps which do not need to be contiguous. When used in combination with belief-propagation-based maximum *a posteriori* (MAP) detectors, an analogous complexity reduction can be realized.

In this chapter we address the design aspects of sparsening filters, including several approaches to minimize the bit error rate of MAP detectors. We devote attention to the interaction of the sparsening filter and detector, and demonstrate the performance gains through simulation.

3.1 Background

We focus on the design of *sparsening* prefilters for use with soft-input soft-output MAP detectors of the form considered in [25, 26]. While [25, 26] primarily focused on the case where the original channel is sparse, we note that even non-sparse channels can be sparsened with a simple linear, FIR filter. Consequently, our work

can be applied in general situations, even where the original channel is not sparse. We address the issue of sparsening filter design with the goal of minimizing the detector BER. We consider the interaction of the sparsening filter and BP detector, and develop a practically-implementable sparsening filter design method.

We note that channel sparsening filters (CSF) can be seen as a generalization of so-called *channel shortening filters* proposed in [75–79]. Given an FIR channel $\boldsymbol{h}$ of length L_h, the channel shortening problem roughly amounts to designing a filter $\boldsymbol{w}$ so that the energy in the combined response $\boldsymbol{h} \star \boldsymbol{w}$ is concentrated in $\mu < L_h$ *contiguous* taps. Channel sparsening is nearly the same, though the μ taps which contain the majority of the energy are not constrained to be contiguous. Furthermore, while much of the recent interest in channel shortening has been for application to multicarrier systems, the original idea of channel shortening [75] was proposed for a reduced-complexity hybrid prefilter/ML detector which bears some resemblance to the one considered here. More recent works such as [80] have considered channel shortening in conjunction with iterative MAP detectors. Again, however, these works impose a constraint that the taps in the combined channel/filter response must be contiguous. One very recent work [81] has considered use of matching pursuit to find a sparse, non-contiguous target impulse response (TIR), and it is shown to yield a lower mean squared error (MSE) compared to the conventional contiguous approach.

3.1.1 BP detector in a Hybrid Structure

After discussing the principles of the BP detector in Chapter 2, we provide more details about the hybrid structure containing the sparsening filter and the BP detector. To compute the likelihood ratios, the BP detector needs to know the effective channel impulse response. Given a finite-length filter $\boldsymbol{h}$, it is not possible in general to find a finite-length filter $\boldsymbol{w}$ such that the combined response is exactly equal to some prescribed FIR response $\mathbf{c}$ since the problem is overdetermined[1]. In other words, in designing the CSF filter $\boldsymbol{w}$, we must accept that it is not possible to perfectly sparsen the channel so that the effective channel $\mathbf{c}$ consists of only μ nonzero taps. Consequently, the remaining $L_c-\mu$ taps of $\mathbf{c}$ will not be exactly equal to zero in general unless the CSF is chosen to have infinite length. Nevertheless, to keep computational complexity at the level prescribed by the choice of μ, we only use the largest μ taps of $\mathbf{c}$ in the computation of the likelihood ratios used inside the BP detector. As such the residual ISI contribution from the smallest $L_c - \mu$ taps of $\mathbf{c}$ in (4.3) will be treated as noise by the BP detector. A sufficiently large choice of CSF length L_w, however, can ensure arbitrarily small additional ISI.

Since the BP detector is typically implemented in the log domain, the majority of its complexity is due to the many summation operations which must be performed [25]. If the BP algorithm proceeds over N total iterations, the total complexity requires on the order of $N(\mu+1)M^{\mu+1}$ summations, where M is the size of the source alphabet and μ is the number of significant effective channel taps used in the detection. As already mentioned in Chapter 2, the complexity of the

[1]Note that a SIMO system employing either multiple receive antennas or fractional sampling *can* perfectly sparsen the channel under certain conditions on subchannel roots [82].

BP is exponential in μ, and so the system designer can specify the total complexity by appropriate choice of μ.

We note that the BP detector performance only truly coincides with the MAP detector when two conditions are met: 1) there are no cycles in the factor graph corresponding to the channel, and 2) the additive noise is white and Gaussian. In general, the first of these conditions is never satisfied. In practice cycles have been shown to be of little concern since they are a low probability event (in the case of potentially detrimental length 4 cycles) [83], or the cycles themselves do not pose a noticeable performance penalty [25]. The second condition on the noise, however, is more serious for this hybrid structure. Since the additive white Gaussian noise (AWGN) gets colored by the CSF, the noise at the input of the BP detector is no longer white[2]. We will address this issue in the sequel.

We emphasize that the CSF does not change the operation of the BP detector. As the CSF changes the effective channel taps, however, and passes the μ largest effective channel taps to the BP detector, the CSF obviously affects the behavior and performance of the combined filter/detector structure. Since the BP detector itself is unaltered from [25], it can accommodate a system employing channel codes such as LDPC encoding considered in [25], or can readily be extended to the MIMO case with, for example, space-time coding as in [26, 80]. Since our focus is on the design of the CSF, we consider an uncoded system.

[2]While it is possible that the noise observed at the receiver front-end is not white to begin with, we make the standard AWGN assumption throughout.

3.2 System Model

3.2.1 Receiver Structure

We consider the system model shown in Fig. 3.1. A sequence of symbols $x[n]$ drawn from an M-ary alphabet is transmitted through an intersymbol interference channel denoted $\boldsymbol{h}$ – which may or may not be sparse – and AWGN $n[k]$ with variance σ_n^2 is added. At the receiver, we employ a detector which consists of the cascade of a CSF which we denote $\boldsymbol{w}$, followed by a belief-propagation-based detector [25]. As mentioned, the BP detector is exponentially complex in the

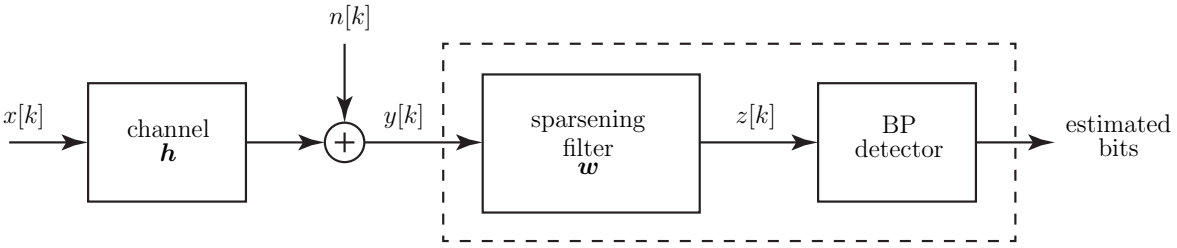


Figure 3.1: System Model

number of nonzero channel taps. Consequently, the purpose of the CSF is to reduce the number of nonzero coefficients in the effective channel to some specified quantity μ so that use of the BP detector becomes practically feasible.

Let $\mathbf{c} = \boldsymbol{h} \star \boldsymbol{w}$ be the effective channel (or combined response) where the $\star$ operator denotes convolution. We assume that the channel, CSF, and effective channel are modeled as FIR filters of lengths L_h, L_w, and $L_c = L_h + L_w - 1$,

respectively. Thus, the received data is given by

$$y[k] = \sum_{l=0}^{L_h-1} h[l]x[k-l] + n[k]. \tag{3.1}$$

After filtering by the CSF, the data is

$$\begin{aligned} z[k] &= \sum_{l=0}^{L_w-1} w[l]y[k-l] \\ &= \sum_{l=0}^{L_c-1} c[l]x[k-l] + v[k]. \end{aligned} \tag{3.2}$$

where the CSF output noise, which is colored, is given by $v[k] = n[k] \star w[k]$. Finally, the output of the CSF is passed to the soft-input soft-output BP detector which outputs likelihood values that can be used to make decisions as to what was transmitted.

In this work, we focus our attention on the optimal design of the sparsening filter. As such, we make the simplifying assumption that the channel $\boldsymbol{h}$ is known perfectly to the receiver. It is rather straightforward, however, to extend our proposed design method to adaptive implementations which can be employed in situations where the channel is unknown and/or slowly time-varying.

3.2.2 Sparsening Filter Design

In the design of the CSF $\boldsymbol{w}$, the goal is for the number of significant (nonzero) taps of $\mathbf{c}$ to be μ or less, regardless of where they lie in $\mathbf{c}$ or whether they are contiguous or not. We note that $\mu \in \{1, 2, \ldots, L_h\}$ is a parameter chosen by the system designer. If $\mu = 1$, then the detector coincides with traditional linear equalization since the goal of the CSF design is to make the effective channel be

a single spike. At the other extreme, the choice $\mu = L_h$ corresponds to "pure" BP detection as in [25] since the CSF need not do any sparsening and can be a simple unity gain filter. Larger choices of μ will result in an exponentially more complicated BP detector, but will also result in better BER performance.

Ideally, we would like to choose $\boldsymbol{w}$ to minimize the system BER. As no closed form expression for the BER exists, direct minimization of BER is intractable. Consequently, we consider choosing $\boldsymbol{w}$ to maximize a proxy for the BER which we term the SSSNR. In the sequel, we will assess the validity of this metric by measuring the proximity of the SSSNR-optimal filter to the BER-optimal filter for some low-dimensional examples. Accordingly, we define the SSSNR as the ratio of the useful signal power coming out of the μ large taps of $\mathbf{c}$ over the total power of the received signal plus noise,

$$\begin{aligned} J_S(\boldsymbol{w}) &= \frac{\sigma_x^2 \left\|\text{large taps of } \mathbf{c}\right\|^2}{\sigma_x^2 \left\|\text{large taps of } \mathbf{c}\right\|^2 + \sigma_x^2 \left\|\text{small taps of } \mathbf{c}\right\|^2 + \sigma_v^2} \\ &= \frac{\sigma_x^2 \sum_{k\in\mathcal{S}} |c_k|^2}{\sigma_x^2 \sum_k |c_k|^2 + \mathrm{E}\left\{|v[k]|^2\right\}} \end{aligned} \tag{3.3}$$

where $\mathcal{S}$ is the set of the locations of the desired largest μ taps in $\mathbf{c}$. The numerator of (3.3) is the signal power scaled by the power of the μ significant taps in $\mathbf{c}$, and the denominator contains the total received signal power. Ideally, the energy in the significant taps, $\sum_{k\in\mathcal{S}} |c_k|^2$, will make up almost all of the energy in the channel, $\sum_k |c_k|^2$, since we want all other taps to be as close to zero as possible. Ignoring noise for a moment, $0 \le J_S \le 1$, and the only way to force $J_S \to 1$ it to make all but μ taps in $\mathbf{c}$ go to zero. Adding in the noise term to the denominator ensures that the residual self-interference is weighted comparably to the noise, so that the excess taps are not made small at the expense of noise amplification.

The SSSNR in (3.3) is analogous to Melsa's Shortening SNR (SSNR) [76], with a few distinctions: the set of desired taps $\mathcal{S}$ is not contiguous, the denominator includes the noise power, and the denominator includes both the desired and undesired taps (rather than just the latter). The last distinction is for numerical reasons, and it can be shown that keeping or omitting the desired taps in the denominator leads to the same solution in this type of problem [84, III.B].

Let $\boldsymbol{H}$ be the $L_c \times L_w$ tall convolution matrix of $\boldsymbol{h}$, i.e. a Tœplitz matrix with first column $[\boldsymbol{h}^T, \mathbf{0}_{1\times L_w-1}]^T$ and first row $[h(0), \mathbf{0}_{1\times L_w-1}]$. Then let $\boldsymbol{H}_{\mathcal{S}}$ be the $\mu \times L_w$ matrix obtained by extracting the μ rows of $\boldsymbol{H}$ corresponding to the desired nonzero tap locations, $k \in \mathcal{S}$. As an example, for $L_w = 3$, $L_h = 3$, $\mu = 2$, and $\mathcal{S} = \{1, 3\}$,

$$\boldsymbol{H} = \begin{bmatrix} h_0 & 0 & 0 \\ h_1 & h_0 & 0 \\ h_2 & h_1 & h_0 \\ 0 & h_2 & h_1 \\ 0 & 0 & h_2 \end{bmatrix}, \quad \boldsymbol{H}_{\mathcal{S}} = \begin{bmatrix} h_0 & 0 & 0 \\ h_2 & h_1 & h_0 \end{bmatrix}$$

With these definitions, the combined response is $\mathbf{c} = \boldsymbol{H} \cdot \boldsymbol{w}$, and the values of the desired nonzero taps are contained in the vector $\mathbf{c}_{\mathcal{S}} = \boldsymbol{H}_{\mathcal{S}} \cdot \boldsymbol{w}$. Then we have

$$\sum_{k\in\mathcal{S}} |c_k|^2 = \|\boldsymbol{H}_{\mathcal{S}} \cdot \boldsymbol{w}\|^2$$

$$\sum_{k} |c_k|^2 = \|\boldsymbol{H} \cdot \boldsymbol{w}\|^2,$$

and the SSNR from (3.3) becomes

$$J_S(\boldsymbol{w}) = \frac{\boldsymbol{w}^H \boldsymbol{B}_{\mathcal{S}} \boldsymbol{w}}{\boldsymbol{w}^H \boldsymbol{C} \boldsymbol{w}} \tag{3.4}$$
$$\boldsymbol{B}_{\mathcal{S}} = \sigma_x^2 \boldsymbol{H}_{\mathcal{S}}^H \boldsymbol{H}_{\mathcal{S}}$$
$$\boldsymbol{C} = \sigma_x^2 \boldsymbol{H}^H \boldsymbol{H} + \sigma_n^2 \boldsymbol{I}.$$

The SSSNR expression in (3.4) is a generalized Rayleigh quotient. The value of $\boldsymbol{w}$ that maximizes this quantity, i.e. the SSSNR-optimal CSF for a given set $\mathcal{S}$, is computed by finding the generalized eigenvector of the matrix pair $(\boldsymbol{B}_{\mathcal{S}}, \boldsymbol{C})$ corresponding to the largest generalized eigenvalue. An algorithm for this general problem is given in [85, Section 8.7.2].

The method used to compute the tap values in [26], which is based on [80], is mathematically similar to our approach, with two key differences. Most importantly, the set $\mathcal{S}$ is fixed in [26]. Second, [26] uses the concept of a target impulse response (TIR). The optimal CSF is written as a function of the TIR, and then the TIR is optimized. (This is implicit within [26, eqn. (25)].) The choice of CSF is "optimal" in the sense that it minimizes the MSE between the outputs of the CSF and TIR, and the TIR is optimal in the sense that it maximizes the signal-to-noise ratio (SNR) at the CSF output. Similar to the channel shortening literature where the minimum MSE and the maximum SSNR channel shorteners are equivalent [86, Section 5], the approach in [26] is mathematically equivalent to our approach with the exception of the fixed sparse coefficient locations. However, the minimum MSE approach is more convoluted to implement, as two filters must be designed rather than one.

The CSF that maximizes (3.4) is only optimal for a given choice of $\mathcal{S}$. As such,

the design of $\boldsymbol{w}$ involves two issues: picking the best locations for the μ nonzero taps in $\mathbf{c}$ (i.e. choosing the set $\mathcal{S}$), and picking the values of the filter coefficients so that (3.4) is maximized. The first issue is related to the problem of choosing the optimal delay in linear minimum mean-squared error equalization, which is known to be nontrivial since there is no known expression for the optimal delay [87]. In the classical equalization problem, it is feasible to conduct a brute-force search over the L_c possible delays. Here, however, the problem is considerably more challenging since there are $\begin{pmatrix} L_c \\ \mu \end{pmatrix} = \frac{L_c!}{(L_c-\mu)!\mu!}$ possible choices of $\mathcal{S}$. In this paper, we consider three methods of choosing the set $\mathcal{S}$.

- Use the indices of the μ largest magnitude taps of $\boldsymbol{h}$, as in [26]. This will be referred to as Roy's tap selection method.
- Try all of the possible combinations. This will be referred to as the combinatorial tap selection method, and it is optimal (though expensive).
- Try the heuristic approach outlined below, which will be referred to as the greedy tap selection method.

The greedy method is as follows. First, set $\overline{\mu} = 1$, and find the location $\mathcal{S}_1$ of a single tap that maximizes the SSSNR. This involves computing $\boldsymbol{w}$ for all L_c possible tap choices. Next, set $\overline{\mu} = 2$ and $\mathcal{S} = \{\mathcal{S}_1, \mathcal{S}_2\}$. Keep $\mathcal{S}_1$ from the prior step, and find the location $\mathcal{S}_2$ such that the best two-tap channel is produced. This involves computing $\boldsymbol{w}$ for each of $L_c - 1$ values. Continue adding one tap at a time until μ locations have been chosen.

Roy's method requires designing a single CSF, although the tap locations are

likely far from optimal (as will be demonstrated in 3.5). The combinatorial method requires designing $\frac{L_c!}{(L_c-\mu)!\mu!}$ filters. Finally, the greedy method requires designing $\frac{1}{2}\mu(2L_c-\mu+1)$ filters. It is far cheaper than the combinatorial method, although its performance approaches that of the combinatorial method, as will be demonstrated in 3.5. For example, with $L_c = 20$ and μ=2, the greedy method is 4.9 times cheaper; and with $L_c = 25$ and μ=5, the greedy method is 460 times cheaper than the combinatorial method.

3.3 Noise Coloration

As mentioned previously, even if the noise $n[k]$ is white, the CSF outputs colored noise. To see this, we let $\boldsymbol{W}$ be a wide Tœplitz convolution matrix corresponding to the filter $\boldsymbol{w}^T$, and compute the covariance matrix of the noise observed by the BP detector as

$$\begin{aligned}\mathrm{E}\left\{\mathbf{v}\mathbf{v}^H\right\} &= \mathrm{E}\left\{(\boldsymbol{W}\boldsymbol{n})(\boldsymbol{W}\boldsymbol{n})^H\right\} = \boldsymbol{W}\mathrm{E}\left\{\boldsymbol{n}\boldsymbol{n}^H\right\}\boldsymbol{W}^H \\ &= \sigma_n^2\boldsymbol{W}\boldsymbol{W}^H \neq \sigma_n^2\boldsymbol{I}.\end{aligned}$$

This is a problem because the BP algorithm assumes white noise. The coloration in the noise will harm the BP performance, potentially making it worse than a classical linear minimum mean squared error (LMMSE) equalizer followed by a simple slicer. Thus, to avoid this pitfall, we consider a penalty term based on the

squared autocorrelation of the output noise, or equivalently of the CSF,

$$\begin{aligned} J_A(\boldsymbol{w}) &= \frac{1}{\|\boldsymbol{w}\|^4 \sigma_n^4} \sum_{l=1}^{L_w-1} \left| \mathrm{E}\left\{ v_k v_{k-l}^* \right\} \right|^2 \\ &= \frac{1}{\|\boldsymbol{w}\|^4} \sum_{l=1}^{L_w-1} \left| \sum_{m=0}^{L_w-1} w_m w_{m-l}^* \right|^2 \\ &= \frac{1}{\|\boldsymbol{w}\|^4} \sum_{l=1}^{L_w-1} \sum_{m,n=0}^{L_w-1} w_m w_n^* w_{m-l}^* w_{n-l} \end{aligned} \tag{3.5}$$

It can be shown that J_A is equivalent to

$$J_A = \int_0^1 \left(\frac{|W(f)|^2}{\int_0^1 |W(f')|^2 \, df'} - 1 \right)^2 df + 1, \tag{3.6}$$

where $2\pi f = \omega$. Thus, J_A penalizes non-flatness of the spectrum of $\boldsymbol{w}$, since J_A drops to its minimum value of 1 as the spectrum $W(\omega)$ approaches any constant value $\forall f$.

In order to combine J_S, which should be maximized with J_A, which should be minimized, we invert the former and include a relative weighting term,

$$J = J_S^{-1} + \beta J_A. \tag{3.7}$$

The weight β can be chosen to try to force the minimum of J to be in the proximity of the BER cost surface, J_E. In the next section, we look at the surfaces J_S, J, and J_E in order to visualize the effect of β.

The value of β can be set several ways. The simplest is to try various values of β and get a sense of which values lead to good results for the class of parameter values of interest. For example, for the parameters in our simulations, $\beta \in [0.1, 0.5]$ seems to yield good results. Alternatively, β can be included in the optimization problem. One could search the objective function of (3.7) for a new value of $\boldsymbol{w}$ (but without changing β), then occasionally adjust β (but not $\boldsymbol{w}$) to improve the

BER, and repeat. If β is updated on a much slower time scale than $\boldsymbol{w}$, then the computationally-intensive BER does not have to be evaluated very often during the search.

For a given value of β, (3.7) can be minimized over $\boldsymbol{w}$ by any method of unconstrained non-linear optimization. We chose to use the simplex method of [88], since it was already available in Matlab, via the "fminsearch" function.

3.4 BER and SSSNR

To visualize the cost surfaces, consider the following example. The channel is $\boldsymbol{h} = [1, 0.5, 0.9, 0.3]$, the target number of taps is $\mu = 2$, the SNR is 8 dB, the CSF $\boldsymbol{w}$ has 3 taps so $L_c = 6$, and we use the unit norm constraint $\|\boldsymbol{w}\| = 1$. With this constraint, the CSF lies on a unit sphere, and can be represented in spherical coordinates: $w_0 \stackrel{\triangle}{=} w_x = \cos(\theta)\sin(\phi)$, $w_1 \stackrel{\triangle}{=} w_z = \cos(\phi)$, $w_2 \stackrel{\triangle}{=} w_y = \sin(\theta)\sin(\phi)$. A contour plot of the SSSNR (inverted) is shown in Fig. 3.2; note that this is an amalgamated surface (i.e. maximized across all choices of $\mathcal{S}$). There is symmetry of a sort due to the fact that $\boldsymbol{w}$ and $-\boldsymbol{w}$ have the same SSSNR.

The impulse response magnitudes of the channel, SSSNR-optimal CSF, and effective channel are shown in Fig. 3.3, and the $\mu = 2$ significant taps of $\mathbf{c}$ are filled in. While the the other taps are small, they are not exactly equal to zero, and will contribute some residual ISI that is left uncorrected by the BP detector. Note that the impulse responses as shown have been normalized to have unit infinity norm. In Fig. 3.4, the i^{th} subplot (counting across the first row then the second row) shows the regions in which that tap is used in the final design, with "dark"

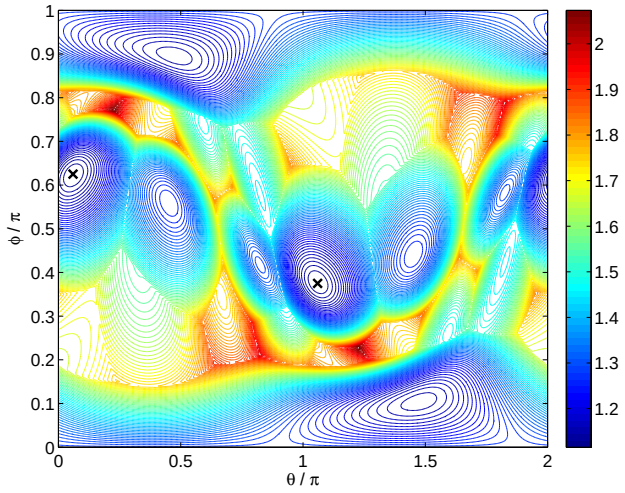


Figure 3.2: Contours of SSSNR for a three-tap unit norm filter, parameterized by two angles in spherical coordinates. The × indicates the filter with the highest SSSNR.

indicating that the tap is used. The axes are as in Fig. 3.2 and we see, for example, that the 6th tap is never selected to be one of the nonzero taps in the effective channel. Furthermore, the amalgamated cost surface is highly multimodal, with significant shape changes at the boundaries of each tap's usage region.

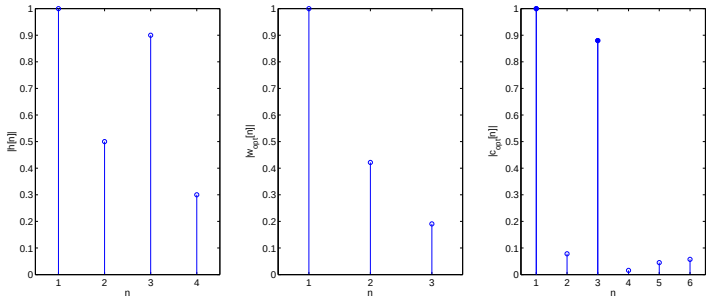


Figure 3.3: Channel, filter, and effective channel tap magnitudes, using the SSSNR-optimal 3-tap filter.

The corresponding amalgamated BER surface, shown in Fig. 3.5 is also highly

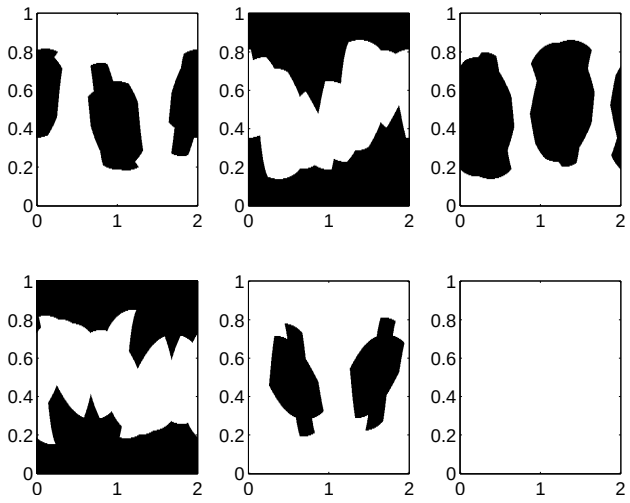


Figure 3.4: Usage maps of the 6 taps of the effective channel $\mathbf{c}$. Axes are identical to Fig. 3.2. Dark areas indicate that the given tap is one of the μ largest taps in $\mathbf{c}$.

multimodal. This plot was generated via Monte-Carlo simulation by exhaustively computing the BER at the output of the BP detector for each value of $\boldsymbol{w}$. Here, there are two minima with nearly equivalent BER of 0.0065 (or, four equivalent minima if one counts symmetric pairs due to the fact that $\boldsymbol{w}$ and $-\boldsymbol{w}$ also yield the same BER). By comparing Figs. 3.2 and 3.5, we see that the SSSNR and BER surfaces look quite different. We note, however, that there are minima in nearly identical locations. This provides some evidence that, at least for this low dimensional example, the SSSNR is a good proxy for BER.

Figs. 3.6–3.8 consider the added squared autocorrelation penalizing term, and show the amalgamated cost surface of (3.7) with $\beta = 1$, as well as various optimal CSFs. These include the two local optima of the BER cost surface, the global optimum of the SSSNR, the near-optimum of the SSSNR with heuristically chosen sparse tap locations, and the weighted SSSNR solution with $\beta = 1$. We note that

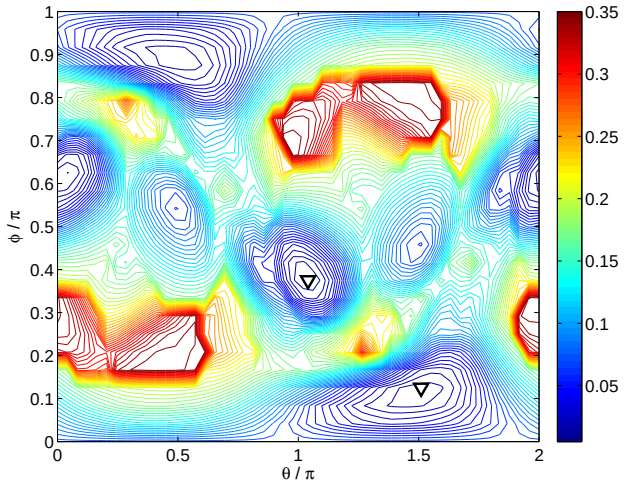


Figure 3.5: Contours of BER for a three-tap unit norm filter, parameterized by two angles in spherical coordinates. The ▽ indicates the filters with the lowest BER.

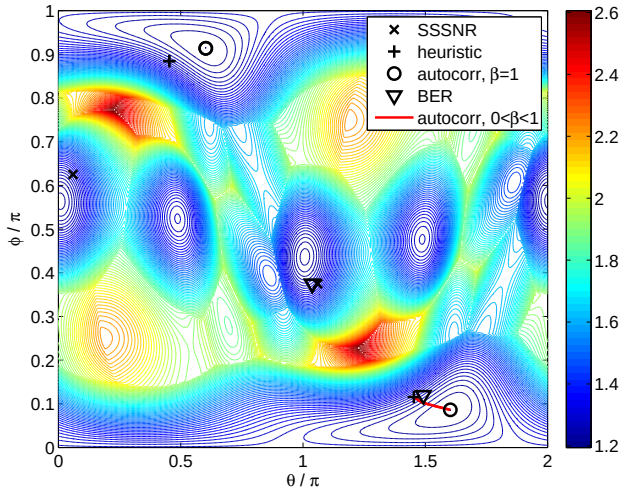


Figure 3.6: Comparison of different optimal CSFs, overlaid on contours of the surface (3.7) with $\beta = 1$.

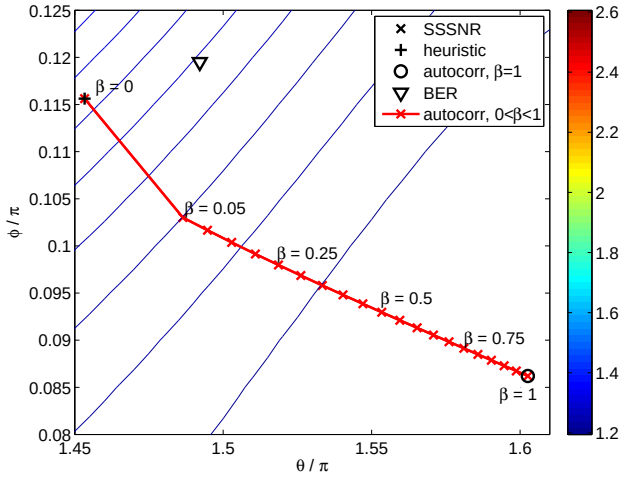


Figure 3.7: Fig. 3.6, zoomed in on the lower right section.

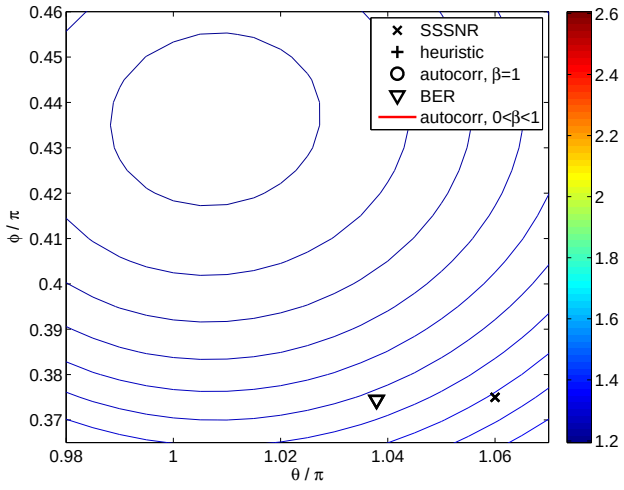


Figure 3.8: Fig. 3.6, zoomed in on the middle section.

in Fig. 3.6 there are two solutions with nearly identical BER; the heuristic solution is near one whereas the SSSNR solution is near the other, and the two pairs of solutions have different active taps. The goal is to determine which optimal solution is the best proxy for the minimum BER solution. Adding the squared autocorrelation penalizing term moves the SSSNR heuristic solution past the minimum BER solution, and it appears that β should be small, say in the range of 0 to 0.2. To further investigate the affect of β, Fig. 3.9 shows the BER performance of the weighted SSSNR scheme as a function of β at 8 dB SNR, and we see that the optimal value of β occurs at 0.1.

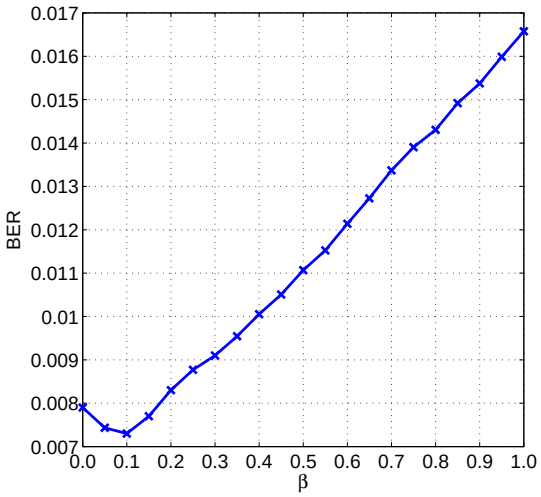


Figure 3.9: Bit error rate as a function of β for the weighted SSSNR scheme

3.5 Numeric Simulations

Having demonstrated that SSSNR is a good proxy for BER, we now compare the BER of the various CSF design approaches. We consider a longer channel than in the low-dimensional example of the previous sections, and we compare SSSNR and

computational time among the different design metrics and tap selection methods. Second, we evaluate the BER for two channels employing different sparsening filters in conjunction with BP.

In the first example, we consider the channel $\boldsymbol{h} = [0.0722, 0, 0, 0.7217, 0.6495, 0, 0, 0.2165, 0, 0.0722]$. We design the CSF to sparsen the channel to $\mu = 2$ taps, we let $L_w = 25$, we transmit uncoded BPSK symbols, and we use 10 iterations in the BP detector. The BER results are shown in Fig. 3.10 for the three tap selection

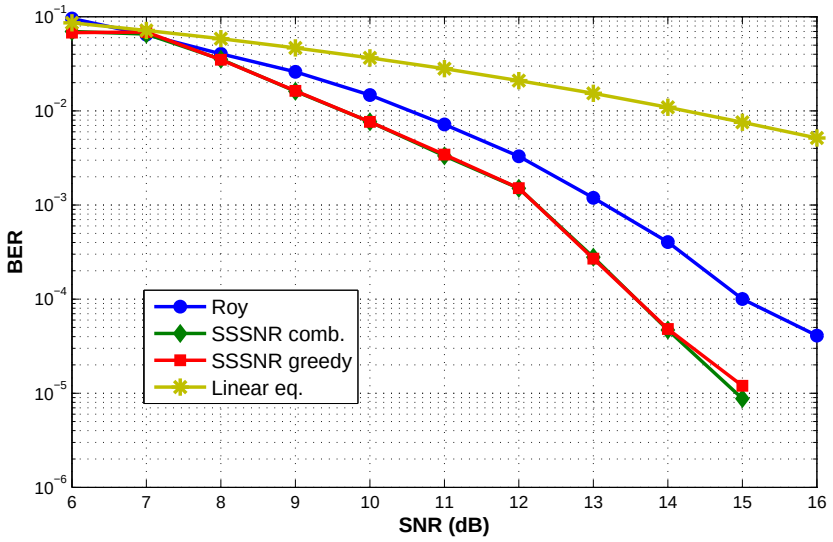


Figure 3.10: Bit error rate for different CSF design metrics

methods outlined at the end of Section 3.2.2, and for comparison we include the performance of a 25-tap classical linear MMSE equalizer with a memoryless slicer. The three methods employing a BP detector handily outperform the LMMSE equalizer, and the greedy tap selection approach is able to attain performance nearly equal to the optimal combinatorial method. In addition, we note that those methods which attempt to pick the set of nonzero taps $\mathcal{S}$ to maximize the SSSNR outperform Roy's method by approximately 1.5 dB for this example.

The complexity, taps selected, and the SSNR of each tap-selection method is displayed in Table 3.1. The combinatorial approach achieved the best SSSNR performance, but requires the design of 561 filters. On the other hand, the greedy approach achieved almost the same SSSNR performance with much fewer filter designs, being more efficient than the combinatorial method. Roy's tap selection method needs to design only one filter, but its SSSNR result is inferior to the other two.

Table 3.1: Computational Complexity, Taps Selected, and SSSNR Achieved at 8 dB SNR

tap-selection method	number of filters designed	taps selected	SSSNR
combinatorial	561	{17,18}	6.9344
greedy	67	{18,19}	6.9336
Roy	1	{4,5}	5.7961

In conducting simulations, we noticed that occasionally the hybrid CSF/BP structure yielded BER performance which was inferior to that of a simple linear equalizer with a memoryless slicer. Upon further investigation, it became clear that the performance degradation in such cases was due to noise coloring by the CSF, as addressed in Section 3.3. We now consider such an example, and show that the use of the modified cost function given in (3.7) results in flatter sparsening filters, and improves the BER performance. We now consider the the channel $\boldsymbol{h} = [-0.21, -0.36, 0.72, 0.5, 0.21]$, we again design the CSF to sparsen the channel to $\mu = 2$ taps, and we let $L_w = 25$. As before, we transmit uncoded BPSK symbols, and use 10 iterations in the BP detector. We also add the squared autocorrelation penalizing term to the combinatorial and greedy SSSNR CSF design metrics. To

choose the β value, we performed a grid search with 10 values between 0.1 and 1 at a 14dB SNR and the best value obtained was $\beta = 0.1$.

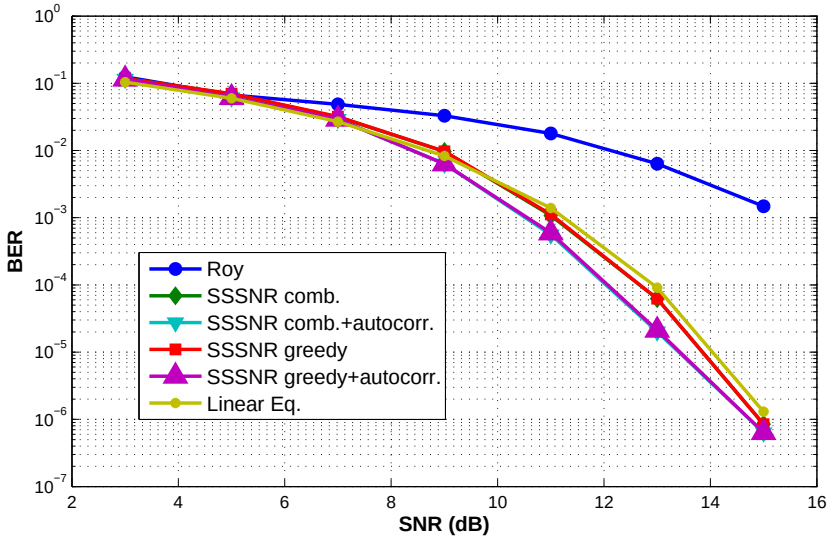


Figure 3.11: Bit error rate for same CSF design metrics shown in 3.10 plus the addition of the squared autocorrelation penalizing term.

From the simulation results in Fig. 3.11, it is apparent that approaches without the addition of the squared autocorrelation penalizing term perform worse than a simple linear equalizer, at least at low to moderate SNR. Overall, the addition of the squared autocorrelation penalizing term improves the BER performance for both the combinatorial and greedy approaches, by approximately 0.7 dB. Also, Roy's tap-selection scheme performs significantly worse than the classical linear MMSE equalizer for this example.

The motivation given for Roy's tap-selection approach [26] was that by choosing the locations of the desired nonzero taps (i.e. the set $\mathcal{S}$) to match the locations of strong arrivals of the incoming signal, there is good "spectral matching" between the channel and the TIR which results in reduced noise enhancement. However,

it is unclear that matching dominant tap *locations* between the channel and TIR (regardless of the tap values) results in spectra with a similar shape. In fact, in the example that intended to motivate the reduced noise enhancement of their approach [26, Fig. 12], the CSF apparently amplifies the noise by about 40 dB in the region of a deep channel null. Thus, while extensive simulations in [26] have demonstrated significant error-rate improvement when compared with competing approaches that employ decision-feedback equalizers, we note that further BER improvement can be made by wiser design of the CSF.

Examination of the frequency responses of the CSF and combined response for this second simulation scenario provides further evidence that does not support Roy's reduced noise enhancement claim. Fig. 3.12 depicts the frequency response of each sparsening filter chosen by the combinatorial approach, the combinatorial approach with addition of the penalizing term, and Roy's method. Roy's filter amplifies the input signal over the frequencies in the center of the band. Comparing the original channel $\boldsymbol{h}$ with the combined responses as shown in Fig. 3.13 provides another picture showing that Roy's chosen filter amplifies noise in the center of the band, which contributes to the degraded performance pictured in Fig. 3.11.

Conversely, by adding the penalizing term to the combinatorial SSSNR approach, the resultant sparsening filter becomes flatter, producing an effective channel similar to the original one. This, in fact, reduces the noise enhancement. Moreover, the flatter frequency response shown by the CSF will also reduce noise coloring thereby improving BP detector BER performance. Both reasons explain the error performance improvement and motivate the incorporation of the squared penalizing term to the CSF design. The optimal choice of β however, remains an

open issue and is likely to be channel-dependent.

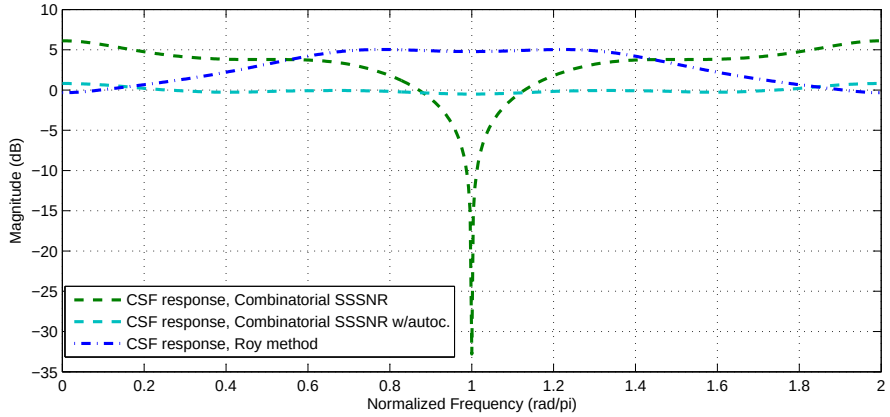


Figure 3.12: CSF frequency response for combinatorial, combinatorial with addition of penalizing term, and Roy's approach.

In addition, to provide further evidence of the proposed method's efficacy, we also considered the ITU Vehicular A channel [89] that has six paths arriving at $[0, 310, 710, 1090, 1730, 2510]$ ns and a power-delay profile of $[0, -1, -9, -10, -15, -20]$ dB. In our simulations we used a square-root raised cosine pulse and a symbol duration of $T = 80$ns, which generally resulted in a sparse equivalent discrete channel with average length of 21 taps. Also, we transmit uncoded BPSK symbols, use 10 iterations in the BP detector and let $\mu = 2$ non-zero taps, and $L_w = 32$. Again, to calculate the β value, we used a grid search at 20dB SNR and the valued obtained was $\beta = 0.2$.

Fig. 4.3 shows the following tap-selection methods in comparison with the linear equalizer: Roy's, combinatorial, greedy and greedy with squared autocorrelation penalizing term. Once again the BP employing methods outperformed the linear equalizer and the method with squared autocorrelation had the best performance. Moreover, at a 10^{-5} SNR the greedy method with squared autocorrelation outper-

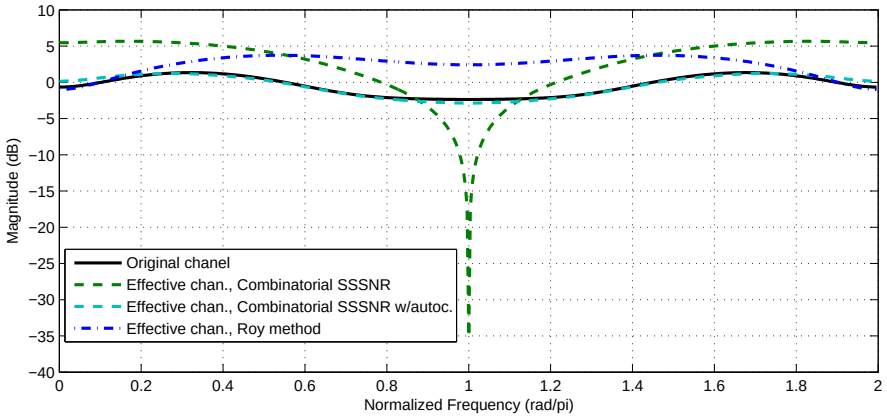


Figure 3.13: Frequency response of original channel and effective channel frequency responses for combinatorial, combinatorial with addition of penalizing term, and Roy's approach.

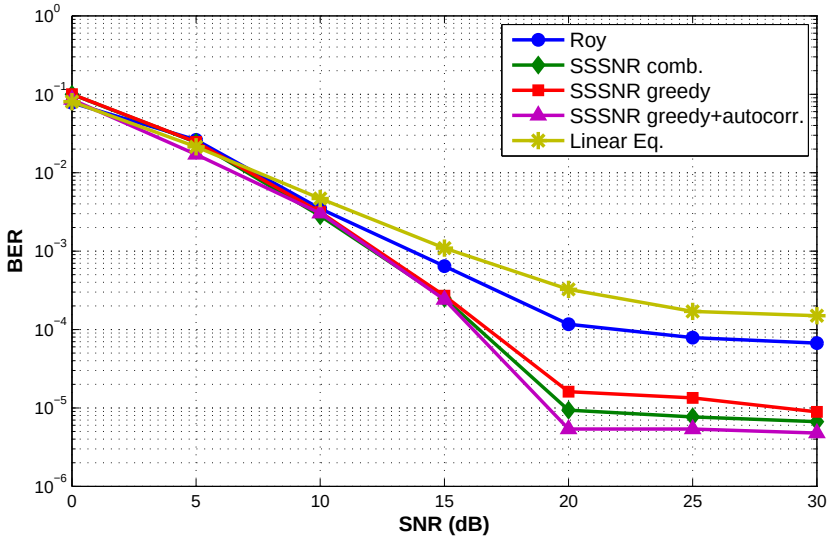


Figure 3.14: Bit error rate curves for the Vehicular A channel

forms the combinatorial method by approximately 2dB.

Finally, we again emphasize that this hybrid detector is quite flexible since its complexity can be adjusted by the system designer. While the complexity scales exponentially with μ, implementations are quite feasible on modern hardware in a wide range of applications [90]. While linear and decision feedback equalizer complexity scales only linearly with the channel length L_h, and are therefore attractive for applications where hardware simplicity is at a premium, the performance advantage offered by the hybrid BP detector (reported here and in [26]) may well be worth the additional complexity. Finally, when compared with traditional Viterbi and BCJR detectors which scale exponentially with L_h, the hybrid BP detector appears to have a considerable advantage in terms of complexity [25].

3.6 Chapter Summary

In this work we have considered the design of sparsening filters as a way to reduce the complexity of iterative soft-input soft-output MAP detectors. By designing the sparsening filter so that the combined response of the (possibly non-sparse) channel and filter has a sparse impulse response, i.e. a response with only a handful of significant taps, the use of a BP-based MAP detector becomes feasible for detecting the bits. We proposed a filter design metric called the Sparse Shortening SNR, and showed that maximizing this quantity serves as a good proxy for minimizing BER. We developed a greedy algorithm for tap selection, and showed that this approach yields near-optimal performance with a significant reduction in complexity when compared to the optimal, combinatorial tap selection approach. In addition, we

treated the issue of noise coloration introduced by the sparsening filter, and showed that the addition of a noise penalty term to the Sparse Shortening SNR results in solutions with a flatter frequency response, thereby limiting the amount of noise coloration. Numerical simulations compared our scheme with an existing approach due to Roy, and showed that significant performance gains can be had by intelligently choosing the tap locations.

Chapter 4

Decision-Feedback Sparsening Filter Design

In the previous chapter, the usage of linear filters in different techniques of sparsening filter design was investigated. In this chapter, a decision feedback filter is designed to reduce the effective channel to a small number of nonzero taps which do not need to be contiguous. When used in combination with belief propagation-based maximum *a posteriori* detectors, a better error performance can be achieved in comparison with the linear design methods. We address the design aspects of decision feedback sparsening filters, devote attention to the interaction of the sparsening filter and detector, and demonstrate the performance gains through simulation.

4.1 Decision-Feedback vs Linear Equalizer

Since its introduction in [22], the Decision-Feedback Equalizer (DFE) receiver structure has received considerable attention from many researchers due to its improved performance over the linear equalizer and reduced implementation complexity as compared to the nonlinear maximum-likelihood receiver.

Noise enhancement is known issue that plagues linear equalizer. This occurs because the main objective in linear equalization is to invert the effects of the wireless channel to the received signal. When the original channel has a deep valley

in its frequency response, the correspondent equalizer will greatly amplify the noise in the frequency band where the original valley lies. The noise enhancement effect has a considerable impact in the linear equalizer error performance.

In comparison, DFEs are able of performing ISI compensation with reduced noise enhancement and may thus provide significantly lower symbol error rates (SER) than a linear equalizer. On the other hand, due to the nonlinear feedback structure of DFE's, symbol errors induced by high noise may cause instability due to the feedback of wrong decisions. This phenomenon called error propagation, although of small probability, may also lead to poor error performance.

4.2 System Model

We consider the system model shown in Fig. 4.1. A sequence of symbols $x[k]$ drawn from an M-ary alphabet is transmitted through an ISI channel denoted $\boldsymbol{h}$ – which may or may not be sparse – and additive white Gaussian noise $n[k]$ with variance σ_n^2 is added. At the receiver, we employ a detector which consists of the cascade of the proposed decision-feedback sparsening filter (DFSF) followed by a BP-based detector [25]. The DFSF consists of a feedforward filter $\boldsymbol{w}$, a memoryless decision device (or slicer), a feedback (FB) filter $\boldsymbol{g}$, as well as a modified FB filter $\tilde{\boldsymbol{g}}$ whose output is added to the output of the feedforward filter to serve as input to the BP detector.

As mentioned, the BP detector is exponentially complex in the number of nonzero channel taps. Consequently, as well as in Chapter 2, the purpose of the DFSF is to reduce the number of nonzero coefficients in the effective channel to

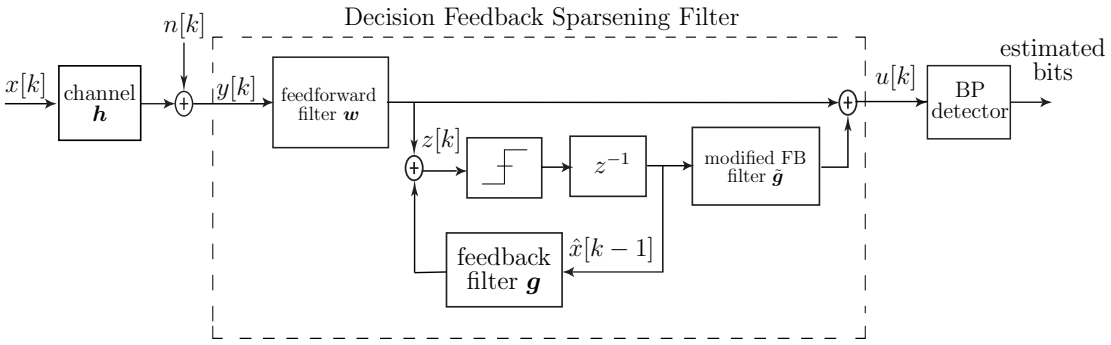


Figure 4.1: System Model

some specified quantity μ so that the use of the BP detector becomes practically feasible. The choice of μ poses as a complexity constraint on the number of ISI taps compensated by the BP detector. Smaller values of μ put more of the burden in ISI compensation on the DFSF, whereas larger values put more burden on the BP detector. We note that the DFSF includes a regular DFE comprised of the feedback loop with filters $\boldsymbol{w}$ and $\boldsymbol{g}$ that ideally suppress all ISI, enabling the slicer to make tentative decisions. These tentative decisions are filtered by $\tilde{\boldsymbol{g}}$ and added to the information corrupted by the ISI channel and the feedforward filter to result in the controlled ISI to be mitigated by the BP detector. While we delay discussion of the design of $\boldsymbol{w}$, $\boldsymbol{g}$, and $\tilde{\boldsymbol{g}}$ until section 4.3, we note that a structure similar to the DFSF was considered in the classic channel-shortening literature [91], though in that case the locations of the residual ISI taps were constrained to be contiguous. By setting the first $\mu - 1$ taps of $\tilde{\boldsymbol{g}}$ to zero, the channel shortener in [91] leaves μ contiguous uncompensated taps in the effective channel which are compensated by a Viterbi detector.

We assume that the channel, the feedforward filter, and the feedback filters are modeled as FIR filters of lengths L_h, L_w, L_g, and $L_{\tilde{g}} = L_g$, respectively. Thus, the

received data is given by

$$y[k] = \sum_{l=0}^{L_h-1} h[l]x[k-l] + n[k]. \tag{4.1}$$

After filtering the signal by the feedforward filter and adding the contribution of the feedback filter, the signal input to the slicer is given by

$$z[k] = \sum_{l=0}^{L_w-1} w[l]y[k-l] + \sum_{l=0}^{L_g-1} g[l]\hat{x}[k-l-1]$$

where $\hat{x}[k]$ are the unreliable tentative decisions output from the slicer. Similarly, the output of the DFSF can be written as:

$$u[k] = \sum_{l=0}^{L_w-1} w[l]y[k-l] + \sum_{l=0}^{L_g-1} \tilde{g}[l]\hat{x}[k-l-1] \tag{4.2}$$

which gets passed to the soft-input soft-output BP detector that outputs likelihood values that can be used to make decisions as to what was transmitted.

Under the optimistic assumption that the slicer makes correct symbol decisions, the output of the slicer is equal to a delayed version of the transmitted symbols, or $\hat{x}[k] = x[k-\Delta]$ where Δ is the symbol delay. In this case, the output of the DFSF can be written as

$$u[k] \;\; = \;\; \sum_{l=0}^{L_c-1} c[l]x[k-l] + v[k] \tag{4.3}$$

where $c[k] \triangleq (h[k] \star w[k]) + \tilde{g}[k-\Delta-1]$ is the combined response of the channel and DFSF having length $L_c = L_h + L_w - 1$, and the DFSF output noise, which is colored, is given by $v[k] = n[k] \star w[k]$. We use the $\star$ operator to denote convolution.

In this chapter, we focus our attention on the design of the decision-feedback sparsening filter. As such, we make the simplifying assumption that the channel $\boldsymbol{h}$ is known perfectly to the receiver. It is rather straightforward, however, to extend

our proposed design method to adaptive implementations which can be employed in situations where the channel is unknown and/or slowly time-varying.

4.3 Decision Feedback Sparsening Filter

In the design of the DFSF filters $\boldsymbol{w}$, $\boldsymbol{g}$, and $\tilde{\boldsymbol{g}}$, the goal is for the number of significant (nonzero) taps of the effective channel $\mathbf{c}$ to be μ or less, regardless of where they lie in $\mathbf{c}$ or whether they are contiguous or not. We note that $\mu \in \{1, 2, \ldots, L_h\}$ is a parameter chosen by the system designer. If $\mu = 1$, then the detector coincides with a traditional DFE since the goal of the DFSF design is to make the effective channel be a single spike. At the other extreme, the choice $\mu = L_h$ corresponds to "pure" BP detection as in [25] since the DFSF need not do any sparsening and can be a simple unity gain filter $\boldsymbol{w}$ and a zeroed $\tilde{\boldsymbol{g}}$. Large choices of μ will result in an exponentially more complicated BP detector, but will also result in better BER performance.

We now address the design of the filter coefficients $\boldsymbol{w}$, $\boldsymbol{g}$, and $\tilde{\boldsymbol{g}}$. As the DFSF contains an embedded DFE and requires tentative decisions from the memoryless slicer, we choose $\boldsymbol{w}$ and $\boldsymbol{g}$ so that the input to the slicer is (nearly) ISI-free. Toward that end, we adopt the popular minimum mean-squared error (MMSE) criterion to design the feedforward and feedback filters which cancel all of the ISI in a suboptimal manner. To shift some of the burden in ISI compensation from the DFE to the more sophisticated BP-detector, the DFSF has another signal path leading to the BP detector where a filter $\tilde{\boldsymbol{g}}$ is designed so that μ taps of controlled residual ISI are left in the effective channel. As we will discuss, we can derive $\tilde{\boldsymbol{g}}$

from the MMSE choice of $\boldsymbol{g}$ so that the combined response of the channel and DFSF results in an effective channel with μ taps of residual ISI.

First we define

$$\boldsymbol{x}[k] = [x[k],\ x[k-1],\ \ldots,\ x[k-L_w-L_h+2]]^\top$$

and

$$\boldsymbol{\Sigma}_\Delta \triangleq \left[\mathbf{0}_{L_g\times(\Delta+1)} \quad \boldsymbol{I}_{L_g} \quad \mathbf{0}_{L_g\times(L_w+L_h-\Delta-L_g-2)}\right].$$

Also, under the assumption that the output of the slicer consists of correct decisions delayed by an amount Δ, we have $\hat{x}[k] = x[k-\Delta]$ and can write the vector of symbol decisions with length L_g at time $k-1$ as

$$\hat{\boldsymbol{x}}[k-1] = \boldsymbol{\Sigma}_\Delta \boldsymbol{x}[k].$$

Let $\boldsymbol{H}$ be the $L_w \times L_c$ wide convolution matrix of $\boldsymbol{h}$, i.e. a Tœplitz matrix with first row $[\boldsymbol{h}^\top, \mathbf{0}_{1\times L_w-1}]$ and first column $[h[0], \mathbf{0}_{1\times L_w-1}]^\top$. As an example, for $L_w = 3$, $L_h = 3$

$$\boldsymbol{H} = \begin{bmatrix} h[0] & h[1] & h[2] & 0 & 0 \\ 0 & h[0] & h[1] & h[2] & 0 \\ 0 & 0 & h[0] & h[1] & h[2] \end{bmatrix}.$$

Then, the slicer input can be rewritten as

$$z[k] = \boldsymbol{w}^\top(\boldsymbol{H}\boldsymbol{x}[k] + \boldsymbol{n}[k]) + \boldsymbol{g}^\top \hat{\boldsymbol{x}}[k-1].$$

And, under the assumption of correct slicer decisions, we can write the DFSF output as

$$u[k] = \mathbf{c}^\top \boldsymbol{x}[k] + v[k],$$

as in (4.3), where $\mathbf{c} = \boldsymbol{H}^\top \boldsymbol{w} + \boldsymbol{\Sigma}_\Delta^\top \tilde{\boldsymbol{g}}$ and $v[k] = \boldsymbol{w}^\top \boldsymbol{n}[k]$. To find the MMSE DFE,

we minimize the cost function

$$\begin{aligned} J_{mse}(\boldsymbol{w},\boldsymbol{g}) &= E\left[|z[k]-x[k-\Delta]|^2\right] \\ &= \boldsymbol{w}^\top(\boldsymbol{H}\boldsymbol{H}^\top+\sigma_n^2\boldsymbol{I}_{L_w})\boldsymbol{w}+2\boldsymbol{w}^\top\boldsymbol{H}\boldsymbol{\Sigma}_\Delta^\top\boldsymbol{g} \\ &\quad +\boldsymbol{g}^\top\boldsymbol{g}-2\boldsymbol{w}^\top\boldsymbol{H}\boldsymbol{e}_\Delta+1 \end{aligned}$$

over $\boldsymbol{w}$ and $\boldsymbol{g}$, where

$$\boldsymbol{e}_\Delta = \left[\underbrace{0,\ \ldots,\ 0}_{\Delta},\ 1,\ \underbrace{0,\ \ldots,\ 0}_{L_w+L_h-\Delta-2}\right]^\top$$

and $x[k-\Delta]=\boldsymbol{e}_\Delta^\top\boldsymbol{x}[k]$. Note that we assume that the source has unit power, and that the data is uncorrelated with the noise. This results in the classic finite-length MMSE DFE design equations:

$$\begin{aligned} \boldsymbol{w}_{opt} &= [\boldsymbol{H}(\boldsymbol{I}_{L_c}-\boldsymbol{\Sigma}_\Delta^\top\boldsymbol{\Sigma}_\Delta)\boldsymbol{H}^\top+\sigma_n^2\boldsymbol{I}_{L_w}]^{-1}\boldsymbol{H}\boldsymbol{e}_\Delta \\ \boldsymbol{g}_{opt} &= -\boldsymbol{\Sigma}_\Delta\boldsymbol{H}^\top\boldsymbol{w} \end{aligned}$$

We now turn to the design of $\tilde{\boldsymbol{g}}$, which is chosen so that μ taps of controlled ISI are left uncanceled by the DFSF. In [91] the μ non-zero taps in the effective channel were required to be contiguous, and consequently $\tilde{\boldsymbol{g}}$ was chosen to be equal to $\boldsymbol{g}$ but with the first $\mu-1$ taps set to zero. Here, since we are employing a BP detector, we have the flexibility of choosing the μ nonzero taps to appear anywhere in a non-contiguous fashion, giving us $\begin{pmatrix} L_c \\ \mu \end{pmatrix} = \frac{L_c!}{(L_c-\mu)!\mu!}$ possible choices. As such, the design of the DFSF involves two issues: picking the best locations for the μ nonzero taps in $\mathbf{c}$, and picking the values of the filter coefficients for best performance. As discussed in [92], large taps in the feedback portion of DFEs have a tendency to enhance the effects of error propagation. As such, we propose to zero the $\mu-1$

largest taps in $\boldsymbol{g}$ to create the filter $\tilde{\boldsymbol{g}}$ which has equal length. Thus, we leave the large taps in the effective impulse response $\mathbf{c}$ to be compensated by the more effective BP detector.

To illustrate the design of $\boldsymbol{w}$, $\boldsymbol{g}$ and $\tilde{\boldsymbol{g}}$, Fig. 4.2 provides an example showing the calculated impulse responses of the filters for the channel $\boldsymbol{h} = [0.5, 0.2, 1, 0.3]$ at 5 dB SNR, with $\Delta = 5$ and $\mu = 2$.

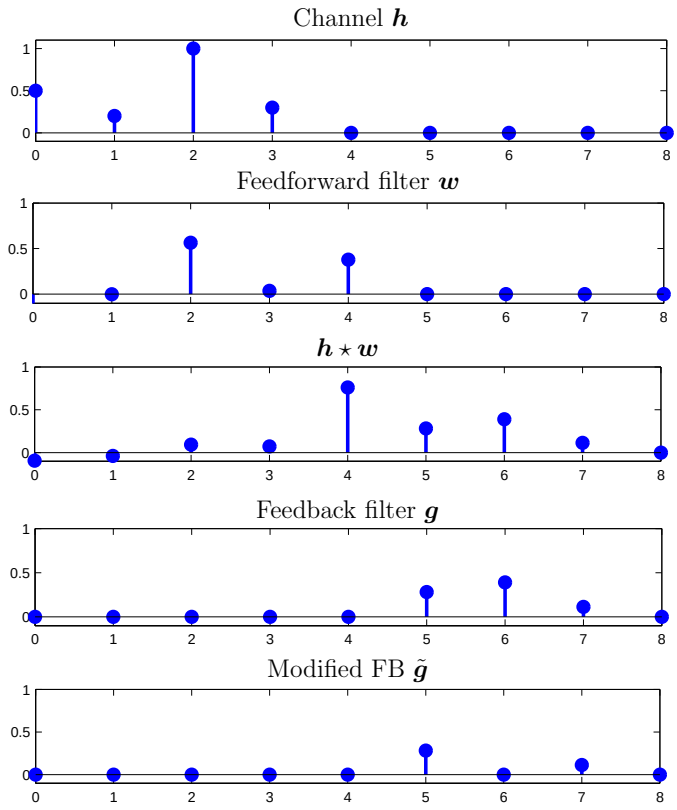


Figure 4.2: Impulse responses

4.4 Numerical Results

In order to provide evidence of the efficacy of the decision feedback sparsening filter design, we simulate two fading environments and compare their performance to Roy's sparsening filter design [26], a classical linear MMSE equalizer with a memoryless slicer and a classical DFE. In addition, to assess the performance degradation due to noise coloring, we include in the simulation a modified DFSF that minimizes the cost function given in (3.7), by adding the squared autocorrelation term that penalizes non-flatness of the spectrum of $\boldsymbol{w}$. To analyze the impact of the noise coloration in the DFSF scheme, we also compare it to the greedy Sparse Shortening SNR (SSSNR) [93], discussed in Chapter 2, that also uses the squared autocorrelation penalizing term. We choose $\beta = 0.2$ for both schemes.

We first consider the ITU Vehicular A channel [89] that has six paths arriving at $[0, 310, 710, 1090, 1730, 2510]$ ns and a power-delay profile of $[0, -1, -9, -10, -15, -20]$ dB. In our simulations we used a square-root raised cosine pulse and a symbol duration of $T = 80$ns, which generally resulted in a sparse equivalent discrete channel with average length of 21 taps. Also, we transmit uncoded BPSK symbols and use 10 iterations in the BP detector. We design the DFSF to sparsen the channel to $\mu = 3$ taps, we let $L_w = 32$, $L_g = 40$ and the delay to be equal to $\Delta = 18$. For this simulation, the classical DFE also has $L_w = 32$ and $L_g = 40$ and the linear equalizer has length equal to 32.

The BER results are shown in Fig. 4.3. It is apparent that DFSF-based schemes employing a BP detector have the best performance among the simulated equalization methods. Notice that at a 10^{-4} BER the DFSF-based schemes outperform

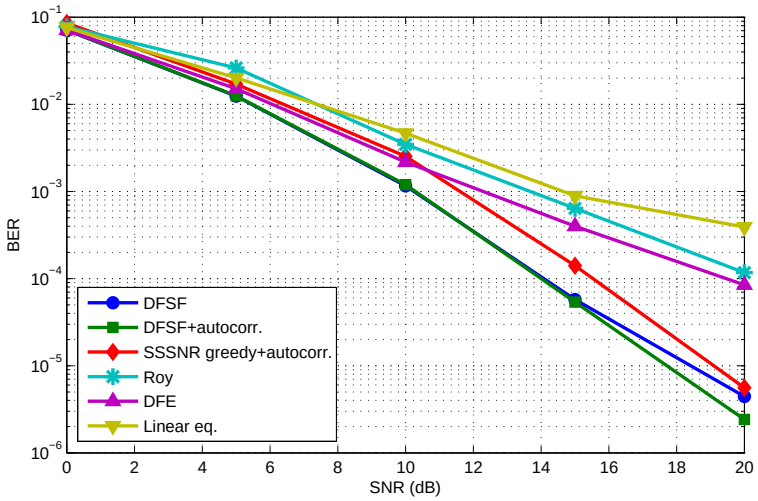


Figure 4.3: Bit error rates for the Vehicular A channel

the classical DFE by approximately 4 dB, indicating a significant improvement in relation to Roy's and the other methods that do not employ the BP detector. When comparing to the greedy SSSNR design technique with the squared autocorrelation term, the improvement is about 1 dB at 10^{-5} BER. However, the greedy SSSNR algorithm requires the calculation of 93 sparsening filters, while the DFSF design requires the calculation of only one filter. This suggests that the DFSF design improves BER performance when compared to existing equalization-detection methods while reducing receiver complexity.

We also note that the use of the squared autocorrelation term does not provide great improvement in the DFSF BER performance for the vehicular channel. At 10^{-5} BER, the difference is only 0.5 dB. When comparing with the SSSNR greedy approach in [93] for the same channel, we verify that the difference between SSSNR greedy and SSSNR greedy+autocorrelation is approximately 2 dB before the saturation. This contrast arises from the fact that in the feedforward filter in

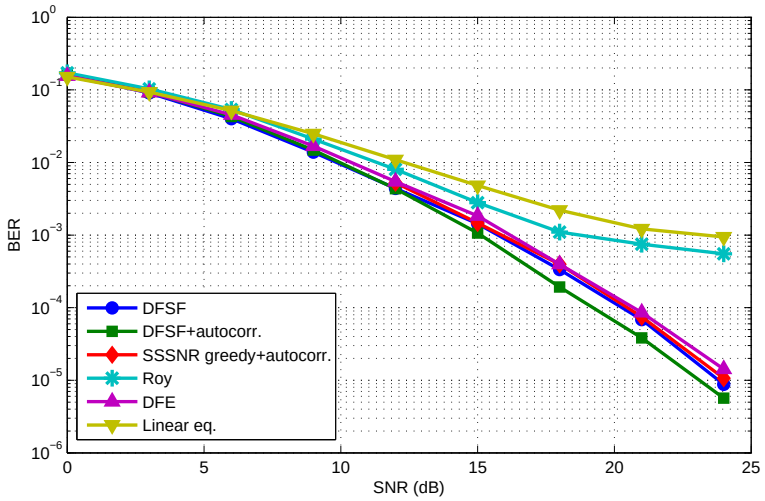


Figure 4.4: Bit error rates for 5-tap channel with equal power delay profile

the DFSF scheme already tends to be approximately flat for the vehicular channel and the addition of the penalizing term does not result in drastic noise-coloring improvement.

In addition, to show that the proposed scheme can also be employed successfully to mitigate non-sparse channels, we consider next a 5-tap fading channel with equal power delay profile of $[0.2, 0.2, 0.2, 0.2, 0.2]$. For this simulation environment, we transmit uncoded BPSK symbols, and we use 15 iterations in the BP detector. We design the DFSF to sparsen the channel to $\mu = 3$ non-zero taps, $L_w = 15$, $L_g = 5$ and the delay to be equal to $(\nu + L_w)/2$, where ν is the position of the largest tap in the channel. The length of the pre-filter for Roy's and SSSNR scheme is equal to 15 and both are also designed to use $\mu = 3$ non-zero taps. For this simulation, the classical DFE and the linear equalizer have $L_w = 15$ and the feedback filter for the DFE has length $L_g = 5$.

Fig. 4.4 shows the simulation results for the 5-tap fading channel with equal

power delay profile. Once again the DFSF method with squared autocorrelation has the best performance among all simulated schemes. We also note that the addition of the squared autocorrelation term increased the performance improvement in relation to the Vehicular A channel.

4.5 Chapter Summary

In this work we have considered the design of decision feedback sparsening filters as a way to reduce the complexity of iterative soft-input soft-output MAP detectors. By designing the sparsening filter so that the combined response of the (possibly non-sparse) channel and filter has a sparse impulse response, i.e. a response with only a handful of significant taps, the use of a BP-based MAP detector becomes feasible for detecting the bits. We proposed a filter design metric based on classical DFE design, but where we allow the largest $\mu-1$ taps in the feedback filter to be left as residual ISI which is compensated by the BP detector. Numerical simulations showed that the proposed scheme can be employed satisfactorily to both sparse and non-sparse channels without requiring great computational cost in the receiver.

Chapter 5

Interface Architecture for Software Defined Radio Systems

In this chapter we present a novel interface framework between an IIO-based radio platform and the GNU Radio SDR development environment in order to enable connectivity and software support for the FMCOMMS1 module. Specifically, we expanded the initial hardware interface framework developed at Analog Devices by enhancing the functionality of the libraries, facilitating seamless connectivity between the IIO-based radio platform and GNU Radio.

5.1 Background

The FMCOMMS1 high-speed analog module [4] was designed to demonstrate the latest generation of high-speed data converters. The FMCOMMS1 module displays sampling-level processing capabilities of 1GS/s and enables radio frequency (RF) applications across a wide frequency range. In addition, it is customizable across software, and without any hardware changes it is possible to use different configurations that can be applied in many different communications applications. Following the FMCOMMS1, ADI also developed the another RF Front-End: the AD-FMCOMMS2-EBZ [5]. The FMCOMMS2 is a high-speed analog module incorporating the AD9361, a high performance, integrated, RF agile transceiver. It is intended for use in applications, such as 3G and 4G base stations and soft-

ware defined radio systems. The device integrates a RF front-end portion with a mixed-signal baseband section and frequency synthesizers.

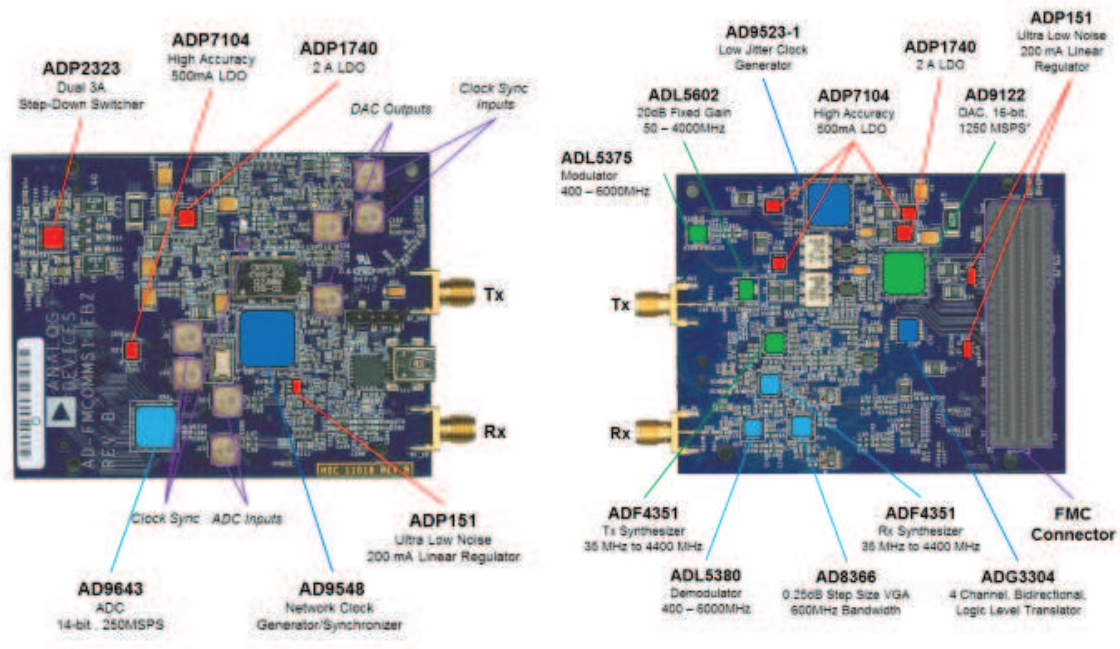
Although ADI's FMCOMMS modules are currently available to the community, there does not presently exist substantial software support for these products in terms of a software environment for communications system design and prototyping, e.g., GNU Radio [74], that can enable the community to use these platforms for over-the-air experimentation.

5.1.1 FMCOMMS 1 RF Front End

The FMCOMMS1 board (Figure 5.1) consists of two main functional partitions, namely, the transmit path and receive path [4]. In Table 5.1, the main components of the board in both paths, as well as their specifications, are presented.

In the transmit direction, the system converts complex in-phase (I) and quadrature (Q) signals into a corresponding RF signal. In the first stage, the digital-analog converter (DAC) interpolates the complex samples and translates the signal to baseband signal. From the DAC output, the signal is sent to a quadrature modulator and again translated to the specified RF output frequency. The analog filter still passes an image rejection filter and an amplifier for +20dB gain. Finally, the RF output power can be controlled by adjusting the baseband data.

In the receive direction, the RF signal is demodulated by the direct-conversion quadrature and brought back to baseband frequency. The resulting I and Q baseband signals are filtered and amplified to obtain 4.5 dB to 20.25 dB of gain. Before



(a) Top view

(b) Bottom view

Figure 5.1: The FMCOMMS1 module is a RF front-end (RFFE) board. Here we have the board and markers highlighting important components of the platform, such as the ADC, DAC, (de)modulators and amplifiers. Source: [4], used with permission.

the ADC, there is an anti-alias filter, which removes harmonics and other out-of-band signals. Finally, the samples are converted to the digital domain by the ADC and the complex samples are processed by the DMA interface.

5.1.2 FMCOMMS 2 RF Front End

Since the AD9361 chip operates in the 70 MHz to 6 GHz range, it covers most of the frequency bands used for most radio standards. As mentioned previously, by changing the sample rate, digital filters, and decimation and interpolation factors

Table 5.1: Listing of the FMCOMMS1 module hardware components present in both transmit and receive paths, with respective specifications. We show the part numbers for each component and the descriptions and specifications. Source: [4]

Components	Specifications
AD9122	Dual, 16-Bit, 1200 MSPS, Digital-to-Analog Converter
ADL5375	400 MHz to 6 GHz Broadband Quadrature Modulator
ADF4351	Wideband Synthesizer with Integrated VCO (35MHz to 4400MHz)
ADL5602	50 MHz to 4.0 GHz RF/IF Gain (20dB) Block
ADL5380	400 to 6000 MHz Quadrature Demodulator, 500MHz bandwidth
AD8366	DC to 600 MHz, Dual-Digital Variable Gain (4.5dB to 20.5dB) Amplifiers
AD9643	14-Bit, 250 MSPS, Dual Analog-to-Digital Converter (ADC)
ADF4351	Wideband Synthesizer with Integrated VCO (35MHz to 4400MHz)

inside the AD9361, the system is capable of supporting channel bandwidths up to 56 MHz. The data path consists of a LNA (Low Noise Amplifier), a demodulator, a LPF (Low-Pass Filter), an ADC (Analog-to-Digital Converter) and digital filters in the receiver portion and digital filters, a DAC (Digital-to-Analog Converter) and

modulators in the transmitter portion. The key features of receive and transmit paths are shown in Table 5.2.

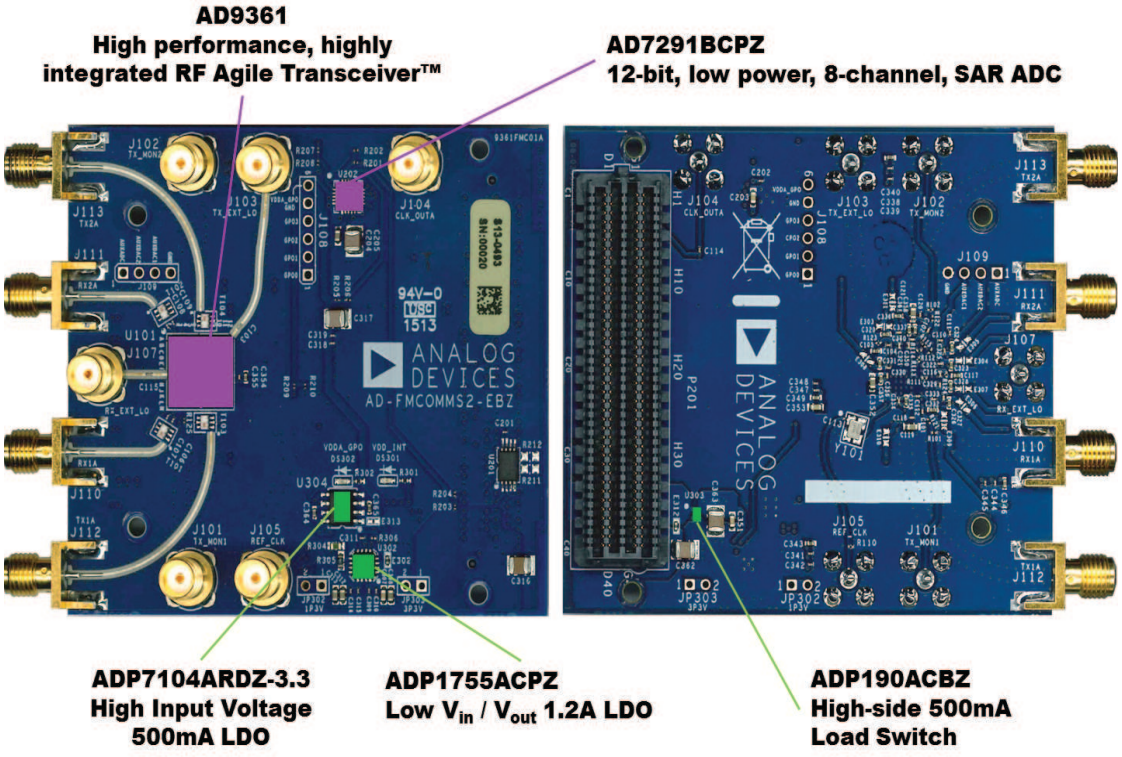


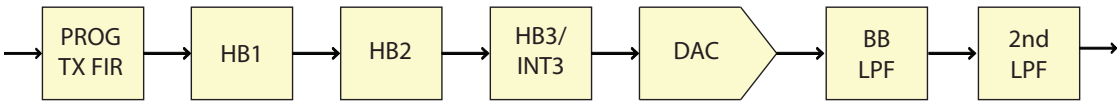
Figure 5.2: The FMCOMMS2 is a high-speed analog module designed by Analog Devices. Source: [5], used with permission.

The AD9361 transmit signal path receives 12-bit 2's complement data in I-Q format from the AD9361 digital interface, and each channel passes this data through four digital interpolating filters to a 12-bit DAC. In order to obtain different data rates, each of the four interpolating filters can be bypassed. Fig. 5.3a shows a block diagram for the AD9361 TX signal path. The TX FIR filter is a programmable poly-phase FIR filter. Its number of taps is configurable for a minimum of 16 and a maximum of 128 taps, and its gains can be set to values between -6dB and 0dB. The filters HB1 and HB2 are fixed-coefficient half-band interpo-

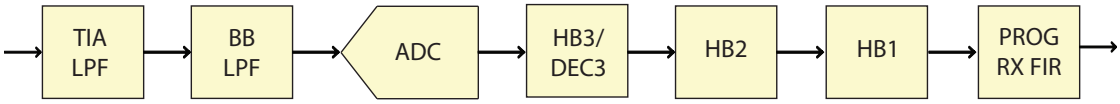
lating filters, while HB3/INT3 provides two different choices of fixed-coefficient interpolating filters; it can interpolate by a factor of 2 or 3. After the DAC, there are two analog filters: the BB LPF, a third-order Butterworth low-pass filter, and the secondary LPF, a single-pole low-pass filter. Both analog filters possess a programmable 3dB corner frequency and are responsible for reducing spurious outputs

Table 5.2: Listing of the FMCOMMS2 key features [5].

Receive Path	Specifications
NF	2.5dB @1GHz
ADC	Continuous time sigma-delta, 640MSPS
Digital Filters	128 complex taps, decimation between 2 and 48
Gain	1dB step size, 80dB analog range, 30dB digital range (post ADC scaling)
Transmit Path	**Specifications**
DAC	320MSPS
Digital Filters	128 complex taps, decimation between 2 and 48
Gain	0.25dB step size, 86dB range
Clocking & Power	**Specifications**
Clock	40MHz crystal
ADP1755	Low dropout, linear regulator, 1.2A, 1.6 to 3.6V
ADP7104	High accuracy, 500mA LDO
ADP190	High side power switch, 1.1 V to 3.6V input range



(a) Transmitter (TX) signal path. The input data for this block diagram is in 12-bit 2s complement in I/Q format and its output goes to the RF mixer.



(b) Receiver (RX) signal path. The input data for this block diagram is an IF signal and it outputs downconverted I and Q signals to the baseband section.

Figure 5.3: Signal paths in the FMMCOMMS2 software-defined radio platform. Inside the AD9361, both TX and RX sections are composed by two signal paths, one for each channel (I and Q).

and providing general low pass filtering prior to up-conversion. Note that both the I and the Q paths are schematically identical to each other.

The AD9361 RX signal path passes downconverted signals (I and Q) to the baseband receiver section. The baseband RX signal path is composed of two programmable analog low-pass filters, a 12-bit ADC, and four stages of digital decimating filters. Each of the four decimating filters can be bypassed. Figure 5.3b shows a block diagram for the AD9361 RX signal path. The RX LPF is a single-pole low-pass filter and the BB LPF is a third-order Butterworth low-pass filter. As in the TX signal path, they are responsible for reducing spurious signal levels and for providing low pass filtering prior to upconversion. The digital filters HB3/DEC3, HB2, HB1 and RX FIR are equivalent to the digital filters present in the TX signal path. Note that both the I and Q paths are schematically identical to each other.

As already mentioned, the four blocks leading up to the DAC in Fig. 5.3a form the digital filtering section of the transmit path, while the four blocks following the ADC in Fig. 5.3b comprise the digital filtering for the receive path. These programmable filters provide the bandwidth limiting required prior to conversion from digital to analog in the transmitter section and bandwidth limiting and out of band noise and spurious signal reduction after digitalization in the receiver section. They also provide interpolation/decimation to generate the correct data rates. In each filter, interpolation/decimation is performed first, followed by the filter transfer function.

5.1.3 ZedBoard and the SDR Platform

As already mentioned, the FMCOMMS modules are analog front-end hardware platforms [66] that are responsible for dealing with the RF portion of a wireless transmission. In addition to the front-end, a digital communication system also requires a radio back-end, which is responsible for the remaining signal processing operations in a receiving or transmitting chain [94]. It is in the back-end that operations such as (de)modulation, filtering, and channel (de)coding are performed, already in the digital domain. The main idea of a SDR system is to implement the digital radio back-end in software in order to provide some degree of reconfigurability.

We can observe in Figures 5.4 and 5.5 that the FMCOMMS board is attached to the fabric portion of the SDR platform. This fabric can be represented by various FPGA development boards, where the software is loaded in order to perform the

digital signal processing methods to implement the communications algorithms. Originally, the FPGA boards combined with FMCOMMS boards can use different microprocessors: ML605, KC705, ZC702, ZC706 and the ZED (Zync). For each one of these devices, ADI provides Linux support by developing drivers for the different parts on the FMCOMMS board. In this work, the software portion of the SDR platform is implemented through the usage of Xilinx's ZedBoard [95]. The Zedboard is a development kit based on the Zync-7000 SoC and can be combined with the FMCOMMS boards to form a complete SDR platform

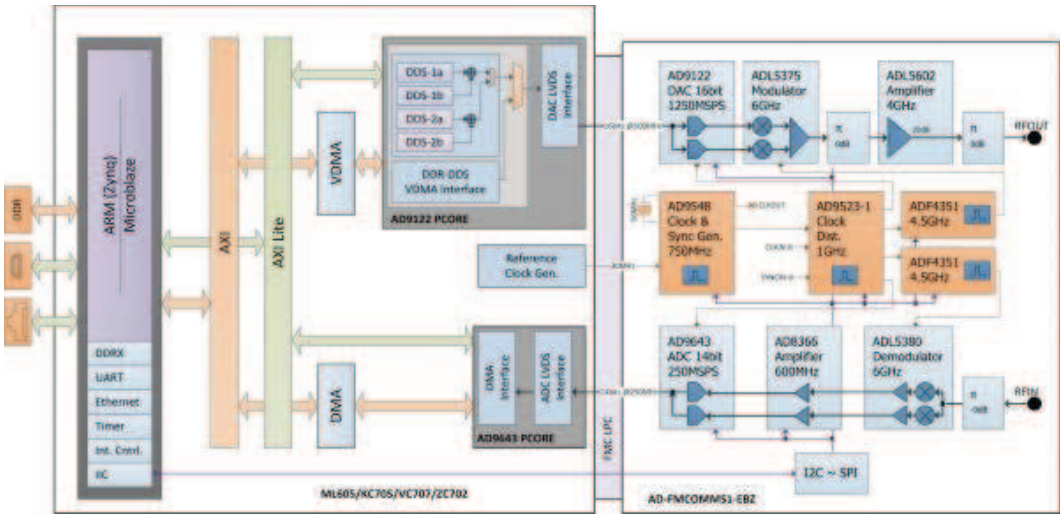


Figure 5.4: Functional Schematics of the hardware of the SDR Platform. On left, we have the ZedBoard and on right we have the FMCOMMS1 RF Front-End. The figure illustrates the transmit and receive chain as well as the functional blocks that are in radio transmission using this platform. Source: [4], used with permission.

The connection between the two boards is performed through a FMC (LPC) connector, which delivers and receives the complex samples to the DAC and to the ADC at a signaling speed supported of up to 10GB/s. Note that the complex samples in this platform can be generated from an internal DDS or an external memory. The internal DDS is formed by four independent signal generators. These

four signal generators are combined to create two tones (the I and Q signals) that are delivered to the DAC. Additionally, the Zedboard includes a Gigabit ethernet interface that allows remote access to the onboard system. It also provides a 4GB SD Card that can be used to boot a Linux environment. With this feature in mind, Analog Devices provides Linux images built for the FMCOMMS modules that complete and enable the development environment.

The industrial I/O subsystem [96] in the Linux kernel provides a unified framework for drivers for many different types of converters, sensors, RF devices, etc

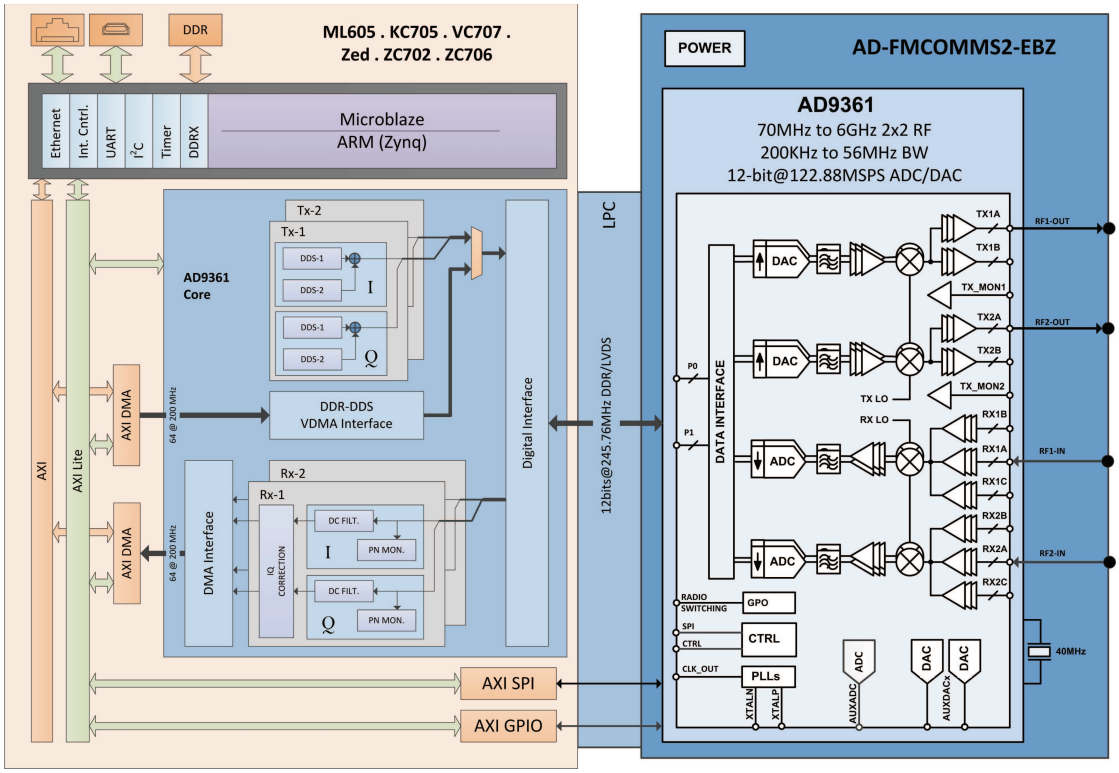


Figure 5.5: Schematic of the software-defined radio hardware. On left, we have the FPGA development board and on right we have the FMCOMMS2 RF Front-End. The figure illustrates the transmit and receive chain. Source: [5], used with permission.

using a number of different physical interfaces (i2c, spi, parallel, high speed serial, etc). It does this by provide a common user space API for these types of devices. Originally targeted at analog to digital (ADCs) and digital to analog converters (DACs), it has expanded to accelerometers, digitally controlled amplifiers (DVGAs), capacitance to digital converters, direct digital synthesis, frequency synthesizers/phase-locked loop, gyroscopes, impedance converters and network analyzers, and inertial measurement units, which are natively supported by Linux.

Consequently, the forced abstraction between hardware devices, and userspace algorithms, ensures that hardware can be swapped out, and algorithm development stays exactly the same. Changing hardware is no longer an task of pouring through semiconductor vendor datasheets to write a new driver, but a simple matter of recompiling the Linux kernel. The IIO subsystem was accepted into the mainline Linux kernel (in `./drivers/iio/`) as of April 2012, and over the last years it has been going through numerous improvements, and iterations, similar to the others pieces of the Linux kernel.

The IIO Command Server [97] is a library that was developed to allow to communicate with the components in the board through the network, and runs on the embedded target and translates a set of simple human readable commands into more complex sysfs and device node interactions. This allows network clients (Matlab, GNURadio, Labview, etc) access to the data available from the real hardware, and makes it a low cost, low overhead networked based data acquisition platform. The IIO Command Server is also referenced as IIO lib 1.0. Its later version, IIO lib 2.0, was released in 2014 and it unifies the network and local accesses.

5.1.4 Interface with GNU Radio

Once the libraries and drivers that directly interface with the FMCOMMS components are installed on the ZedBoard, communications with the RF-front end can be performed by the user. To assist the user, a software development environment such as GNU Radio can be used to implement communications systems that utilize the FMCOMMS1 module for real wireless transmission. However, in order to access and communicate with the components on the SDR platform, it is necessary to access the IIO-lib from inside GNU Radio.

As already mentioned, GNU Radio works by using flow graphs. This means that each digital signal processing (DSP) component is represented by a block and systems can be built by connecting the blocks according to a determined signal flow. For the GNU Radio user, it is important that the communication with the hardware portion of the radio transmission to be transparent since the user is mainly interested in the digital portion of the signal processing. Consequently, the objective is to deliver or collect the complex samples to be processed or generated by the DSP blocks in a flow graph from within a GNU Radio application.

To address this problem, we developed both sink and source blocks in GNU Radio in order to collect and deliver the I/Q samples to other blocks in a flow graph. When necessary, it is also possible to control certain parameters of the components in FMCOMMS1 module, such as transmission power, center frequency and bandwidth. The sink block functions as an information sink: it absorbs the samples from the output of flow graph and manages the transmitting process in the hardware. On the other hand, the source block manages the receiving process

in the RF-Front End and delivers the complex samples to the other blocks in the flow graph. In a lower level, the blocks communicate directly to the DAC and ADC buffers. Using the drivers and libraries (IIO lib) made available by ADI, it is possible to write and read directly to and from the buffers of the hardware components, taking into consideration the used data types (offset binary in the FMCOMMS1 case).

In this work, we present two ways of accessing and driving the components in the RF front-end: from a GNU Radio environment running inside the Zync, or from a remote host running GNU Radio, communicating through the Ethernet connection. Figures 5.7 and 5.6 illustrate these two different options.

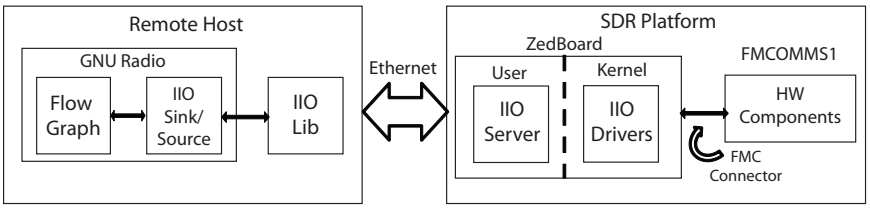


Figure 5.6: High level illustration of the transmitting/receiving processes using the Ethernet connection in the ZedBoard. The sink/source blocks in GNU Radio communicate with the IIO drivers in the Zedboard.

Figure 5.7 shows the signal flow of the transmitting/receiving processes when the application implemented in GNU Radio is running inside the SDR Platform. The flow graph of the application delivers or collects complex samples to the sink block or from the source block, respectively. The sink/sources blocks communicate with the FMCOMMS1 hardware components using the IIO Lib and the IIO Drivers.

Figure 5.6 shows the signal flow of the transmitting/receiving processes when

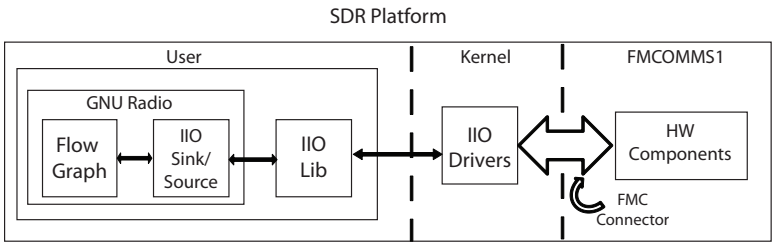


Figure 5.7: High level illustration of the onboard transmitting/receiving processes. The sink/source blocks in GNU Radio communicate with the IIO drivers inside the same platform.

the application implemented in GNU Radio is running in a remote host computer. In this case, the interactions between the GNU Radio blocks and the IIO Lib remains identical to the former case, but now they happen on a remote host computer instead of the ZedBoard. To communicate with the SDR platform a network interface is used to access the IIO-Server running on the Zynq processor. The IIO-Server program is then used to communicate with the IIO Drivers and later to the FMCOMMS1 board. In this setup, the bottleneck becomes the ethernet connection, which limits greatly the speed of processing when compared to the previous option.

5.2 Interface Development

To validate the interface between GNU Radio and the FMCOMMS RF front-ends we provide a suite experiments to test the correct functioning of the developed GNU Radio blocks. The objective is to demonstrate the exchange of information between a GNU Radio application and the SDR platform as well as the potential use of the new hardware on the implementation of advanced and practical communications

systems with this development environment.

5.2.1 Source Block

To test the interface with GNU Radio, we start with the source block by guaranteeing the correct reception of a known signal. We thus use the DDS component from the ZedBoard, a loop connecting the transmitting and the receiving chains and simple python application in GNU Radio. A sinusoid is generated from the DDS component in the development kit and is passed through the transmit chain. The signal is fed back to the receiving chain and then is sampled from the ADC. Figure 5.8 shows the experiment's signal path.

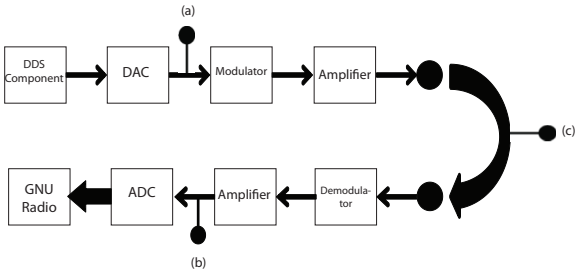


Figure 5.8: Experiment Set-Up. The points (a), (b) and (c) show the probing positions for the obtained measured signals in the Results subsection

After sampling the ADC, the samples are fed into the GNU Radio environment and processed to be displayed using two different python applications. The corresponding setup can be visualized by the flow graph represented in Figure 5.9. With this figure, it is possible to understand the data path in the user space. The ADC's buffer is sampled and the samples are processed inside the IIO Source to generate the I/Q information which is required as input for GNU Radio. The processed samples then feed python blocks created as GUI elements [98] used to visualize the

received signal. We provide visualizations in both time and frequency domains.

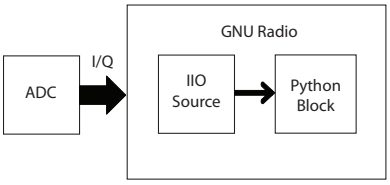
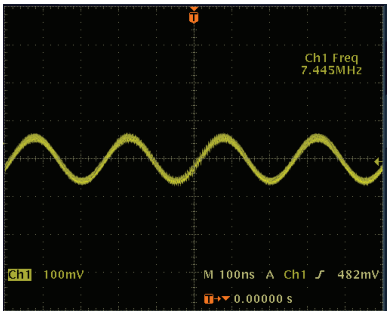


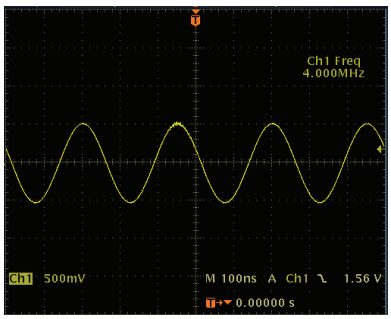
Figure 5.9: GNU Radio Flow Graph

As can be observed, in Figure 5.1, the FMCOMMS1 board presents various probing points that can be used to measure the flowing signal and help on debugging any kind of system implemented using this platform. With the aid of this probe points, it is possible to analyze the hardware's signal path and to confirm the expected behavior of the SDR Platform as well as to compare it with the results obtained in the software platform. Figure 5.10 presents plots of the signal measured in the points indicated in Figure 5.8.

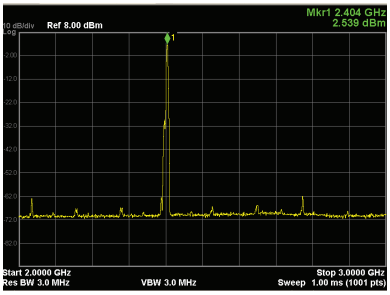
In Figure 5.10, it is possible to visualize the transmitted/received signal at three different points: right after the DAC component, when the signal has been converted to the analog domain, at the receiving/transmitting RF antennas and right before the ADC component, which will convert the analog signal back to the digital domain. The first two pictures show the transmitted and received sinusoids in the time domain. We note that the frequencies for both sinusoids match the frequency designated in the DDS (4MHz), but the signals differ in amplitude and DC gain from each other. While the signal measured in (a) presents an amplitude of approximately 180 mVpp and a DC gain of 482 mV, the signal in (b) presents 1 Vpp of amplitude and 1.56 mV of DC gain. This discrepancy can be explained by the gains and amplifiers stages between the two measurements and the different voltage levels required by each analog component. In this regard, the



(a) Signal measured in a oscilloscope right after the DAC component. The measure point is indicated as (a) in Fig. 5.8.



(b) Signal measured in a oscilloscope right before the ADC component. The measure point is indicated as (b) in Fig. 5.8.



(c) Frequency Response of the RF signal obtained with a spectrum analyzer. The measure point is indicated as (c) in Fig. 5.8

Figure 5.10: Plots obtained by probing the FMCOMMS1 board as indicated in Figure 5.8. The transmitted/received signal is shown at different stages of the radio transmission.

most important feature of the signal, which is the frequency, is conserved.

The last plot shows the RF signal frequency spectrum. As the RF frequency is in the GHz range, the time domain information looses relevance and a spectrum analyzer becomes the ideal measurement tool. It can be observed that the signal spikes at approximately the expected frequency: 2.404 MHz. Some other

harmonics can be observed around it, but at the maximum level of -30dBm. In summary, the analog information guarantees the preservation of the characteristics of the digitally generated signal and provide a strong base of comparison with the desired results in the software environment.

Figures 5.11 and 5.12 are images of the GUIs created in python to aid the analysis of the received sinusoid. The GUI provides basic information usually common in real-life oscilloscopes and spectrum analyzers. In Figure 5.11, it is possible to visualize the received sinusoid in the time domain. In this case, the signal's amplitude is being represented in counts, in relation to the quantization levels of the ADC component. The time scale is represented in μs. With the two datatips, it is possible to calculate the signal's period, which is 0.25286μs. The frequency is calculated to be $\approx$3.9447 MHz, matching the digitally generated sinusoid's frequency and the measured analog signal in hardware.

Finally, Figure 5.12 shows the frequency response of the received signal. The datatip shows the frequency of the signal at $\approx$3.96753MHz as expected, and the noise floor at approximately -10dB. The noise floor indicates precision problems in the FFT calculation and requires further investigation in terms of consistent visualization. Being able to obtain frequency information is essential to any communications system design and marks the first step toward more complex system implementations.

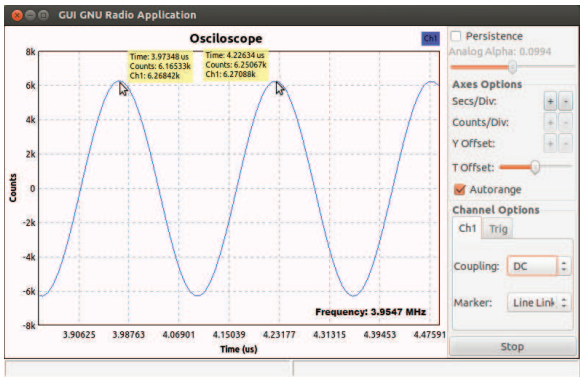


Figure 5.11: View of the oscilloscope application in GNU Radio. The GUI shows the signal sampled from the ADC and processed to be compatible with the GNU Radio environment. In this figure, we see a time domain plot of the signal with a measured frequency of 3.9447 MHz

5.2.2 Sink Block

The next step to validate the interface was to test the functionality of the sink block. We use the GNU Radio environment to generate a sinusoid in the digital domain and to transmit over the SDR platform. The complex samples are delivered to the sink block and written to the buffer of the DAC component in the RF board; the signal passes through the transmit chain, is fed back to the receive chain and later sampled by the ADC. Fig. 5.13 shows the test environment in GNU Radio and Fig. 5.14 shows the complete test set-up.

After the signal is sampled by the ADC, the complex samples are collected by the DMA interface and can be processed by applications running in Linux in the ARM processor. The ADI IIO Oscilloscope is an example application, which enables plotting of the data captured from the ADC in three different modes (time domain, frequency domain and constellation). We use the IIO oscilloscope to verify

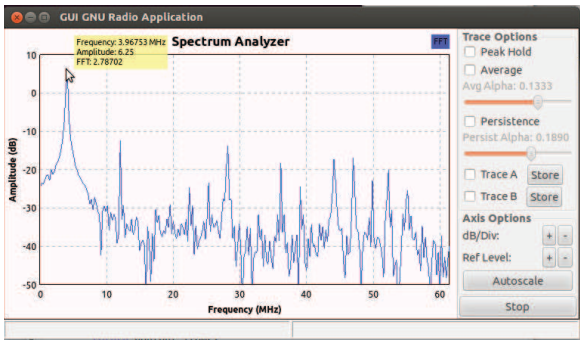


Figure 5.12: View of the spectrum analyzer application in GNU Radio. The GUI shows the signal sampled from the ADC and processed to be compatible with the GNU Radio environment. In this picture we see a frequency domain plot of the signal with a measured frequency of 3.96753MHz MHz

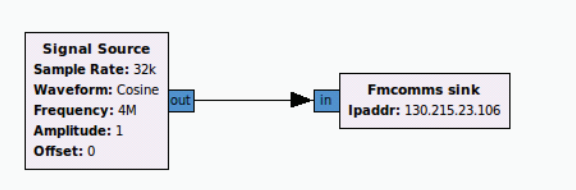


Figure 5.13: GNU Radio environment for test of the sink block.

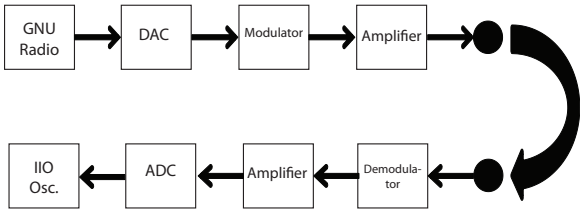


Figure 5.14: Test set-up for the sink block. The signal is generated in GNU Radio, passed through the RF front-end board and then analyzed using ADI IIO oscilloscope tool.

the received signal and to compare it with the signal that was generated by the application in the GNU Radio environment. Fig. 5.15 shows the signal in the IIO Oscilloscope.

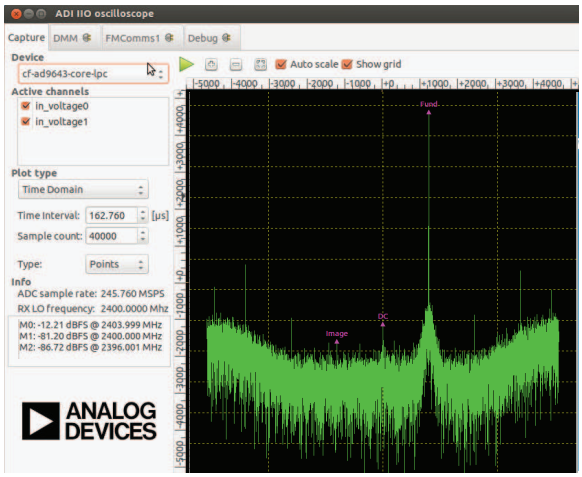


Figure 5.15: ADI IIO Oscilloscope measurement.

As it can be noticed, the signal measured by the IIO Oscilloscope in the frequency domain matches the generated sinusoid of $4MHz$. It is possible to see that the fundamental component lies at approximately $2403.999MHz$, where the LO frequency is of $2400MHz$. As expected, the sink block is capable of correctly transmitting a sinusoid generated inside the GNU Radio environment.

5.2.3 Complete Interface Experimentation

To provide a complete test for developed the interface with GNU Radio, we again use a loop connecting the transmitting and the receiving chains and a flow graph in GNU Radio. A signal is generated using digital signal processing blocks in GNU Radio companion and its I and Q samples are written to the DAC buffer and passed through the transmit chain. The signal is then fed back to the receiving chain and sampled from the ADC. After sampling the ADC, the samples are fed into the GNU Radio environment to be processed and analyzed. The ADC's buffer is sampled and the samples are processed inside the IIO Source to generate the

I/Q information which is required as input for GNU Radio.

In this sense, we provide an experiment where DBPSK symbols are transmitted using the developed interface. Fig. 5.16 shows the digital blocks used in GNU Radio to generate, transmit, receive and the analyze the digital symbols. The transmitter and receiver flow-graphs appear together in this picture for illustration purposes only; during the experiment each portion is executed by a different instance of the GNU Radio Companion tool. The "Random Source" block generates random bits which are encapsulated in blocks of 2 by the "Packet Encoder" block. The information is then modulated and pulse-shaped using the "DPSK Mod" block. For this experiment, we used 2 samples per symbol and a root-raised cosine filter with an excess bandwidth of 0.35. The "Throttle" block used in both the transmitter and receives serves as rate-limiting for resource management in GNU Radio. The I and Q samples are then delivered to the "Fmcomms sink" block, which interfaces with the DAC component in the RF front-end. In the receiver portion, the samples are collected from the ADC into the GNU Radio environment using the "Fmcomms Source" block.

Fig. 5.17 shows the DBPSK symbols in different at different stages during the transmission. The transmitted symbols can be visualized in the "QT GUI Constellation Sink" block in the transmitter portion of GNU Radio flow graph. It is possible to see the DBPSK symbols spread over the In-phase axis given the pulse-shaping operation. The received symbols can be visualized in two manners: using the ADI IIO Oscilloscope tool running in the Linux from the ZedBoard or the other "QT GUI Constellation Sink" block in the receiver portion of the GNU Radio flow-graph. It can be noticed that the received symbols present a phase

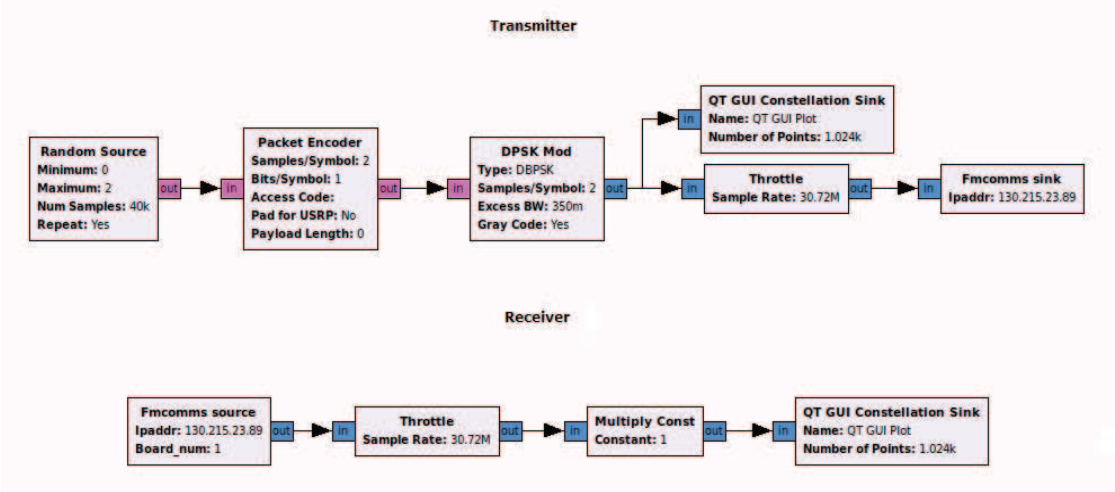


Figure 5.16: Transmitter and Receiver configurations in GNU Radio. The signal is generated and modulated in DBPSK symbols before being sent to the "Fmcomms sink". The signal goes through the transmitting and receiving paths in the hardware platform and the received I/Q are delivered back to GNU Radio through the "Fmcomms source" block.

rotation in comparison to the transmitted symbols. This rotation is introduced by the RF cable used to connect the RF in and RF out of the FMCOMMS1 board. But more importantly, it is possible to compare the received constellations outside and inside the GNU Radio environment and guarantee perfect match, which shows the correct functioning of the developed interface for GNU Radio.

In addition to check the correct transmission and reception of digital symbols, it is important to stress the developed interface to verify the limits in which it starts to deteriorate. In this sense, we tested different sampling frequencies allowed by the hardware components (ADC and DAC) and verified the percentage of lost samples. The test consisted of writing I and Q samples directly to the DAC buffer, feeding them back to the receiver chain, without passing through the RF portion

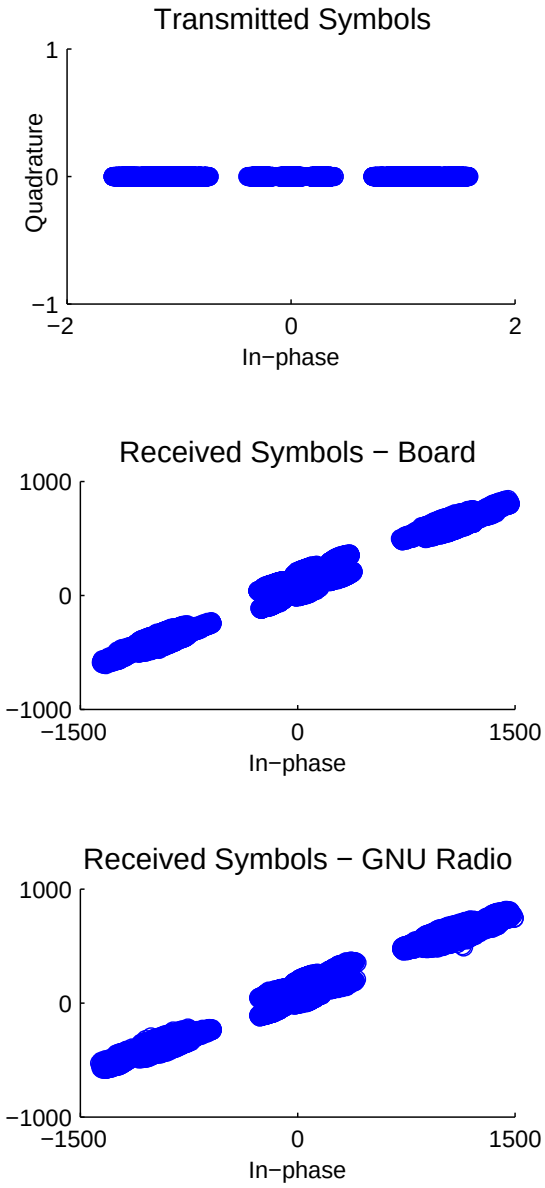


Figure 5.17: Transmitted and received DBPSK symbols sent through the platform. The transmitted symbols are generated and modulated in GNU Radio and sent through the transmit chain. The symbols are then fed back to the receive chain and can be visualized using the ADI IIO oscilloscope tool in the ZedBoard and inside the GNU Radio environment.

Table 5.3: Percentage of samples lost during the transmission of 10^7 samples

Frequencies(MHz)	Lost samples(%)
10.31	0.07
30.72	0.16
61.44	0.28
81.30	0.57
98.22	1.34
122.88	2.62
163.54	2.73
245.76	14.29

if the FMCOMMS board, and collecting them in the GNU Radio environment. As such, we seek to provide an upper bound for the sampling frequencies that can be supported by the use this software interface with the FMCOMMS SDR platform. Table 5.3 shows the tested sampling frequencies and the respective percentages of samples lost during the GNU Radio processing. For each frequency tested, 10^7 samples were analyzed.

It is easy to note that the interface starts to deteriorate when the sampling frequency of 245.76 is reached. At this sampling frequency, the ADC buffer fills up quicker than the GNU radio block is capable of deal with. For future works, the implementation of a buffer in the GNU Radio blocks should help with accommodating higher sampling frequency values. However, in this cases, the trade-of between throughput and delay needs also to be analyzed.

5.3 Chapter Summary

In this chapter, we presented an interface architecture that enables software connectivity and support for the FMCOMMS1 board and the GNU Radio development environment. We also provided experiments using GNU Radio Companion and the FMCOMMS hardware platform that attest the correct functionality of the proposed interface. Finally, we provide a stress test for the developed interface determining that its functioning deteriorates when reaching the sampling frequency of 245.76 MHz.

Chapter 6

Sparsening Filter Experimentation in SDR Platform

In order to demonstrate the capability of implementing more complex communications algorithms and schemes in software and being able of experimenting using the FMCOMMS SDR platform, different experiments were designed and performed using the platform. In this paper, we present the SDR experimentation of the BP-based receptor discussed in Chapter 3 and Chapter 4. As already mentioned, the pre-filter show-cased in these experiments was first presented in literature in [99] in its linear version and a decision-feedback structure was later proposed in [100].

We then provide different experimentation set-ups and results for the experimentation of the BP-based receptors with the greedy SSSNR algorithm and the DFSF in the FMCOMMS SDR platform. In this case, the experiments are not executed in real-time given the complexity of the BP detector and the number of interactions necessary to guarantee reasonable performance; the symbols are collected at base-band and later fed to the detection algorithm.

In addition, to provide error-performance results for the designed detector, such as BER curves, it is imperative to perform the experiments in controlled environments for repeatability and accuracy. In this work we show two sets of experiments executed with the platform running BP-based receptors: experiments in base-band and experiments in RF frequencies.

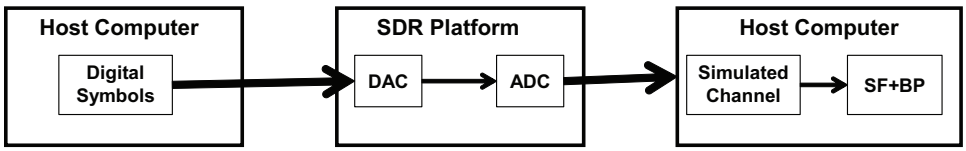


Figure 6.1: Experiment using the SDR platform with simulated wireless channel

6.1 Experimentation in Baseband

In Fig. 6.1, we show a block diagram that illustrates the experimentation environment. In a host computer, BPSK symbols are generated and loaded to the SDR platform. The I and Q samples then pass through the transmit chain and are fed-back to the receive chain without passing the RF portion of the digital transmissions; the samples are transmitted and received in baseband in a digital loop-back enabled by software. The received samples are then collected and are used as input to a wireless channel model also simulated in software in another host computer before being delivered for processing and detection to the hybrid structure formed by the sparsening filters and the BP detector.

6.1.1 Experiment Set-Up

In order to perform baseband experimentation with the FMCOMMS1 board, it was necessary to change the board configuration and re-solder jumps to disconnect the RF portion of the board as illustrated in Figure 6.2. The FMCOMMS1 board in baseband configuration is shown in Figure 6.3, where the output of the DAC is connected to the input of the ADC using SMB cables. The next step is to verify the signal received in baseband to determine if it is possible to proceed with the software simulation portion of the experiment. However, as Figure 6.4 shows, the

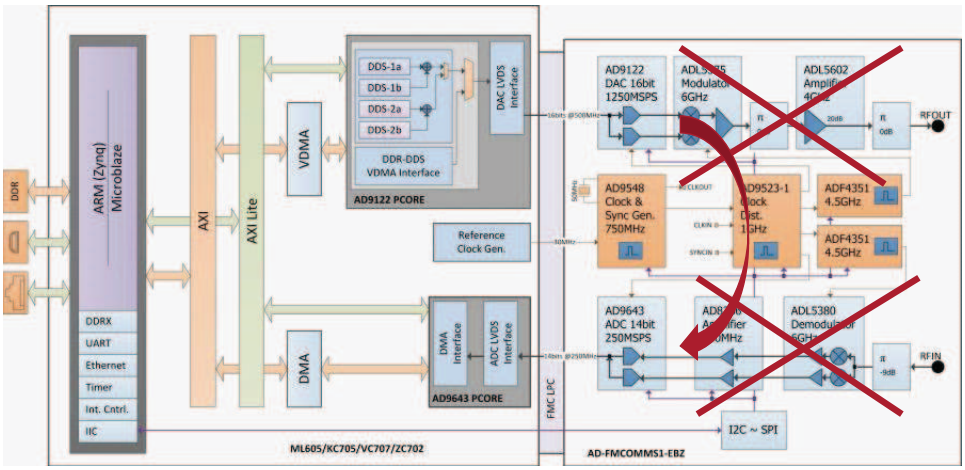


Figure 6.2: Schematics for the baseband experiment with the FMCOMMS1 board. The RF portion of the board is disconnected from the DAC and ADC outputs and the signal is redirected to go from the DAC output to the ADC input.

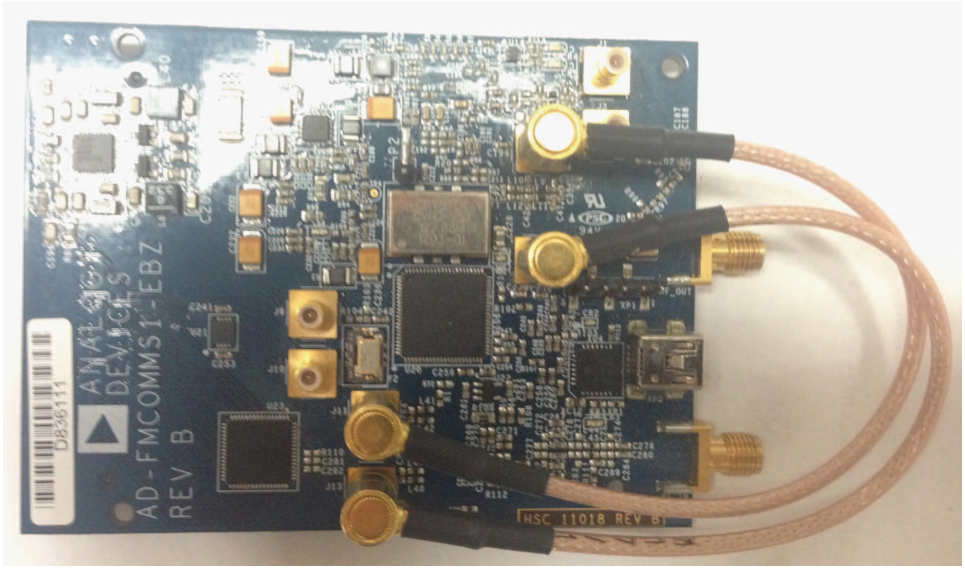


Figure 6.3: FMCOMMS1 board in baseband configuration. The output of the DAC is connected to the input of the ADC using SMB cables.

symbols received in baseband get scattered due to an antenna effect provided by the connecting cables. In this sense, the detection operation to be performed in software is hindered and a reasonable error performance becomes infeasible.

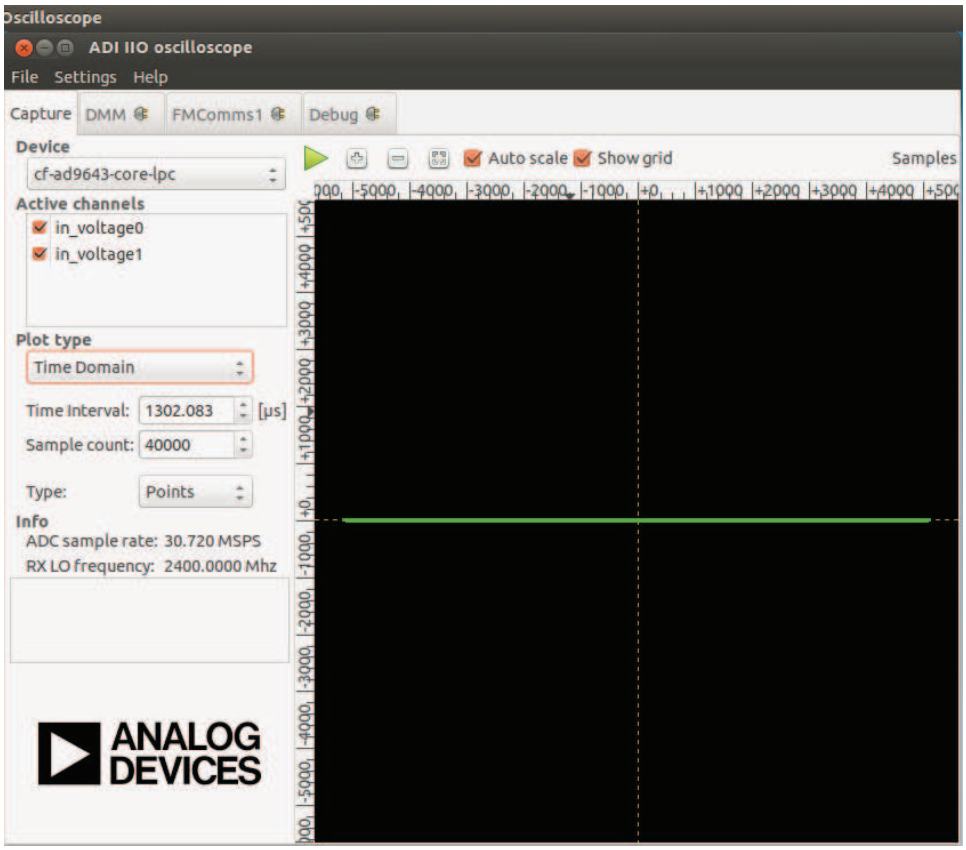


Figure 6.4: The baseband signal received at the ADC. The BPSK symbols are scattered along the X axis making the detection operation unfeasible.

To perform baseband experimentation with the FMCOMMS2 board, the change in the configuration can be done by software. As the entire front end is implemented in a single ship, it is possible to change the value of the configuration registers to create a digital loopback and obtain the baseband transmission-reception. The digital back can be implemented by using the FMComms2 Advanced plug-in, in the "BIST" table, selecting the "TX-¿RX loopback option. Figure 6.5 shows the received BPSK signal for the FMCOMMS2 board, looking as a BPSK constellation, as expected. In this manner, we can assure that the baseband experimentation will not hinder the performance of the detection portion to be executed in hardware.

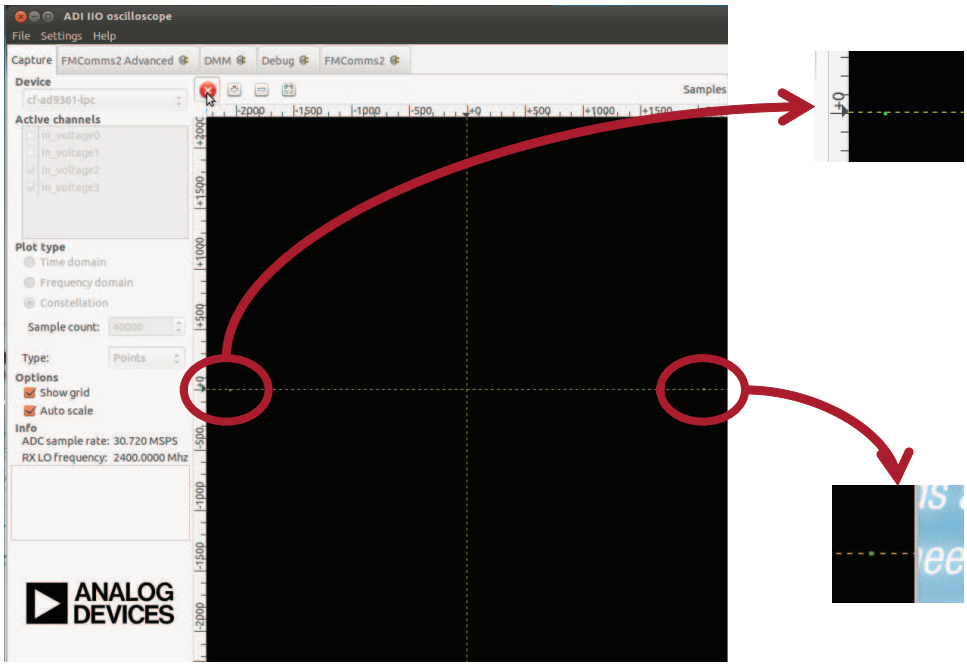


Figure 6.5: The baseband signal received at the ADC. The BPSK symbols are showed as expected and the constellation can be easily identified.

For these reasons, we decided to use the FMCOMMS2 board for the rest of the experiments.

6.1.2 Numerical Simulations

In the first example, we consider the channel h = [0.0722, 0, 0, 0.7217, 0.6495, 0,0, 0.2165, 0, 0.0722]. We design the sparsening filters to leave $\mu = 2$ taps in the effective channel perceived by the BP detector, we let length of the linear sparsening filter (greedy SSSNR) and the feedforward portion of the DFSF to be equal to 15, $L_w = 15$. Also, we let the feedback portion of the DFSF to be equal to 5, $L_g = 5$, and the delay to be equal to $\Delta = 10$. We transmit uncoded BPSK symbols, and we use 10 iterations in the BP detector. The BER results

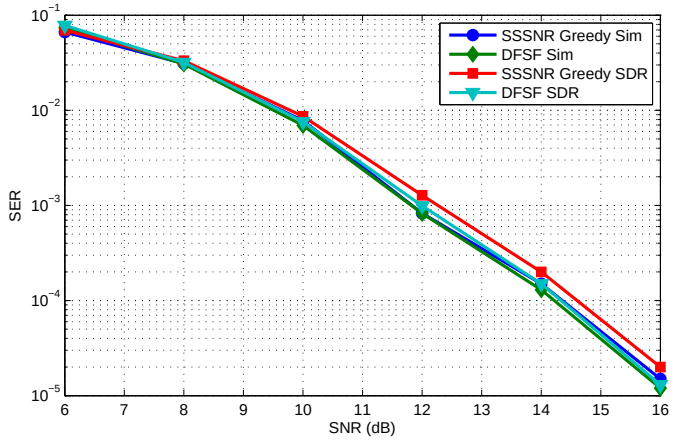


Figure 6.6: Symbol error rates for multipath channel.

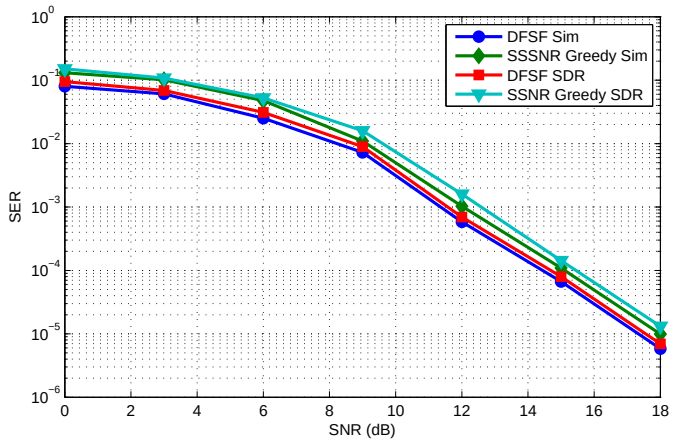


Figure 6.7: Symbol error rates for vehicular channel.

are shown in Fig. 6.6 and we compare the obtained result with the same scenario completely simulated in software. We note that the BER performance of the SDR implementations is very close to the numerical simulations, showing only a minor degradation, as expected, due to non-linearities in the hardware components and sampling errors.

In addition, we also considered the ITU Vehicular A channel [89] that has six paths arriving at [0, 310, 710, 1090, 1730, 2510] ns and a power-delay profile of

[0,.1,.9,.10,.15,.20] dB. In our simulations we used a square-root raised cosine pulse and a symbol duration of T = 80ns, which generally resulted in a sparse equivalent discrete channel with average length of 21 taps. Also, we transmit uncoded BPSK symbols and use 10 iterations in the BP detector. We design the DFSF to sparsen the channel to $\mu = 2$ taps, we let $L_w = 32$, $L_g = 40$ and the delay to be equal to $\Delta = 18$. It is possible to note that in this scenario the SDR implementations also have error performance similar to the ideal simulated case, which shows the correct implementation of the proposed scheme.

6.2 Experimentation in RF Frequencies

The baseband experiment is important to guarantee a sanity check for the integration of the software defined radio platform and the software environment that is running the digital signal processing portion of the communication system to be implemented. For a experiment more similar to a real-world situation, it is necessary to transmit and receive the information in RF frequencies, which is used in all wireless systems.

As already mentioned, for obtaining error performance analysis, such as BER curves, the experiments should be conducted using controlled environments for repeatability and accuracy. In addition, for the BP detector function correctly, the channel must be known by the algorithm, in order to build the factor graph used to calculate the *a posteriori* probabilities. One option is to include a channel estimator to the system, but the BP detector performance would be limited by the channel estimation performance and it would no be possible to analyze separately

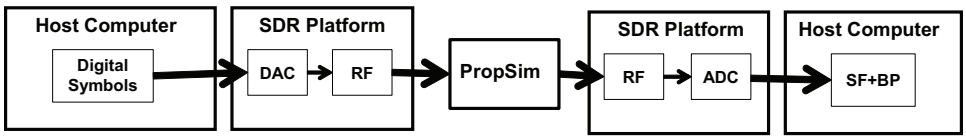


Figure 6.8: Experiment using the SDR platform with channel emulator.

the sparsening filter + BP detector structure performance as desired. One second option would be to use a channel emulator. Radio channel emulators or simulators are equipments designed to provide a test environment for air interface of wireless communications systems [101].

6.2.1 Proposed Experiment with Channel Emulators

In Fig. 6.8, we show the proposed experiment set-up. In a host computer, BPSK symbols are generated and loaded to the SDR platform. The samples are converted to an analog signal by the DAC and is later modulated to a determined RF frequency by the RF portion of the SDR platform. The signal is then fed to the channel emulator, which simulates the environment of a wireless channel in RF domain. The output of the channel emulator is then fed-back to the receive chain of the SDR platform, passing through the demodulation and sampling processes. After being sampled by the ADC, the signal is stored and finally used as input to the SF+BP structure that is executed in another host computer.

The PropSim [102] equipment is a radio channel emulator that enables recreating the wireless channel propagation effects in a controlled laboratory environment. It performs a realistic and accurate emulation of typical radio channel propagation effects, such as multipath and fading and also supports multiple channels for

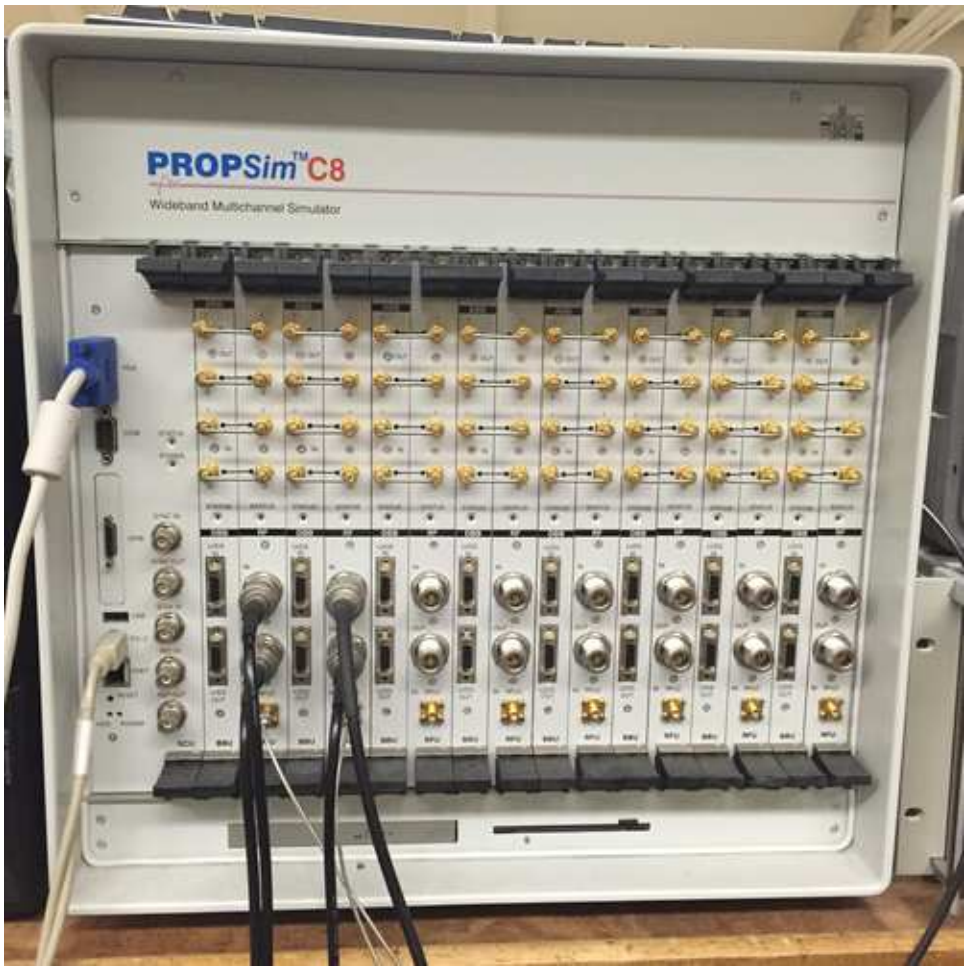


Figure 6.9: PropSim channel emulator at the Center for Wireless Information Network (CWIN) Lab.

MIMO experimentations. The physical radio channel characteristics can all be emulated independently on Propsim, allowing the repeatability necessary for generation of BER curves and other error performance metrics. Figure 6.9 shows the equipment.

We initially proposed the use of the PropSim channel emulator for the experiment in RF frequencies because the Center for Wireless Information Network Lab (`http://www.cwins.wpi.edu/`) at WPI possessed the equipment. However, at the

moment this dissertation was being produced, the equipment was not functional, needing specialized technicians to repair it. Thus, an alternative route was made necessary for the experimentation in RF Frequencies, the experimentation with a simulated channel.

6.2.2 Experiment with Simulated Channel

In Figure 6.10 the set-up for the experiment with Simulated Channel is shown. BPSK symbols are generated in a host computer and loaded to the DAC buffer at the transmit chain. The signal is then transformed to the analog domain, modulated to a determined RF frequency and amplified to be transmitted over-the-air using the RF out antenna. In the following, the signal is fed back to the receive chain using a SMA cable to pass through amplification, be brought back to baseband and transformed in digital samples by the ADC. The samples are then collected from the ADC buffer and delivered to the host computer. In the computer the samples are processed and filtered by a channel model simulated in software. After the channel, the samples are delivered to the sparsening filter + BP detector to be detected. With the RF transmission followed by a channel simulated in software, we are able to identify and analyze impairments and nonlinearities inherent to wireless transmission maintaining the controlled environment that a channel modeled in software provides.

As we are now transmitting over RF frequencies, it is necessary to implement pulse shaping to transmit in a finite bandwidth, so a square root raised cosine is added in both the transmitter and receiver. Also, some phase is usually added

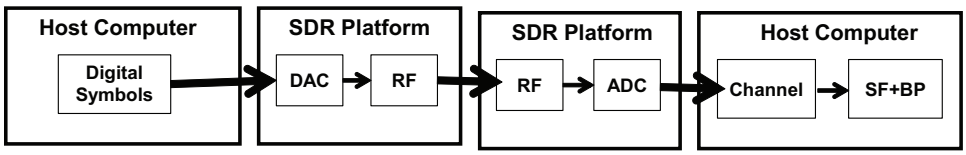


Figure 6.10: Experiment using the SDR platform with simulated channel.

when transmitted over the SMA cable and the constellation is rotated. Thus, a phase compensation is also made necessary in the receiver pre-processing. Figure 6.11 shows the pre-processing required in both transmission and reception.

In this experiment, we use in the transmitter a square root raised cosine filter with a length of 16 symbols, a rolloff factor of 0.25 and 4 output samples per symbol; generating a bandwidth of approximately 20 MHz. For the FMCOMMS2 parameters, we chose a RF frequency of 2.4 GHz, a sampling rate of 61.44 MSPS and a RF bandwidth of 25 MHz. At the receiver we used a square root raised cosine filter with 16 symbols of length, 0.25 of rolloff factor, 2 input samples per symbol and decimation factor of 2. Figure 6.12 shows the received constellation. In (a) we show the symbols collected from the ADC buffer; note the rotation in the constellation. Part (b) shows the received symbols after the phase compensation and Part (c) shows the symbols after the square root raised cosine filter. Note that even though it is possible to recognize the BPSK constellation, the RF transmission introduced noise in the process resulting the smearing of the BPSK symbols, specially in the real axis. The symbols collected after the receiver filter are delivered to the simulated channel + detection portion of the experiment.

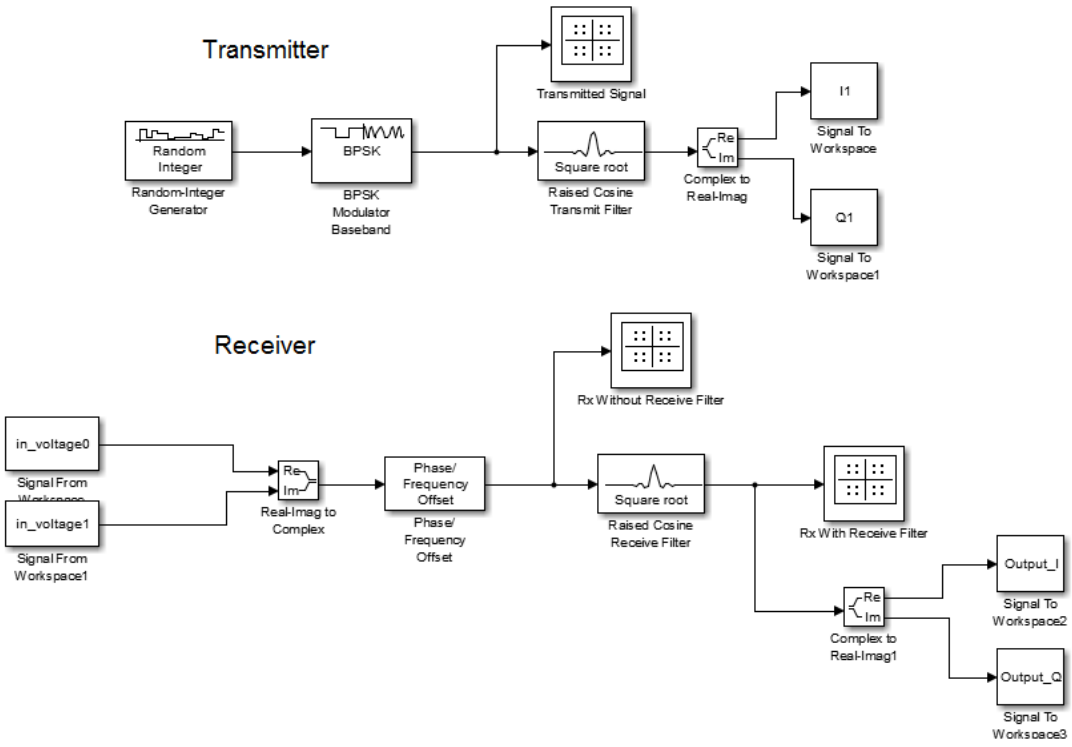


Figure 6.11: Pre-processing required for experiment. At the top we have the pre-processing that occurs in the transmission. At the bottom we have the pre-processing in the reception.

6.2.3 Numerical Simulations

For the error performance evaluation in the experiments with RF frequencies, we once more first consider the channel h = [0.0722, 0, 0, 0.7217, 0.6495, 0,0, 0.2165, 0, 0.0722]. As the previous experiment, $\mu = 2$, $L_w = 15$, $L_g = 5$, $\Delta = 10$ and 10 iterations in the BP detector. We also considered the same model for the ITU Vehicular A channel with $\mu = 2$ taps, $L_w = 32$, $L_g = 40$ and $\Delta = 18$. The BER results are shown in Figures 6.13 and 6.14. For comparison, BER curves of the ideal completely simulated scenario are also included.

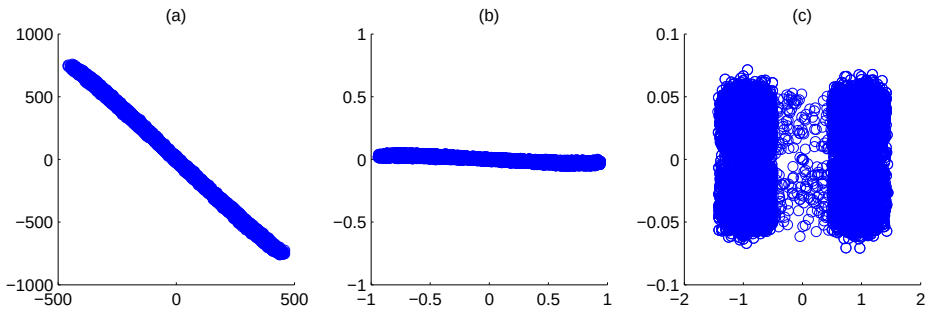


Figure 6.12: The received BPSK symbols. In (a) the received constellation collected from the ADC's buffer. In (b) we have the constellation after the phase compesantion and in (c) after the square root raised cosine filter.

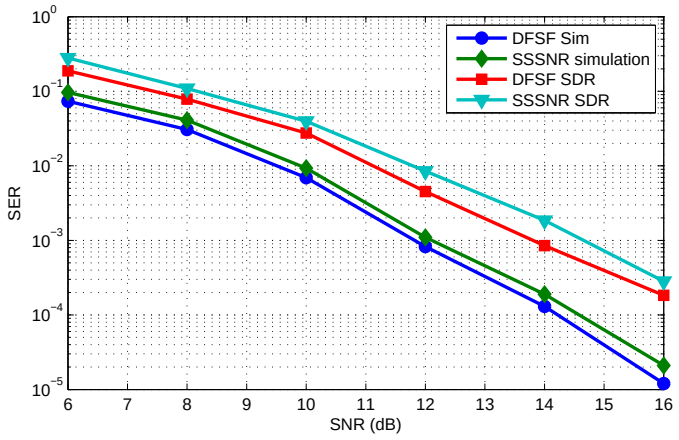


Figure 6.13: Symbol error rates for multipath channel.

As expected, the additional noise process added by the RF transmission (antenna effects, non-linearities in the RF modulation process and transmission over the SMA cables) had a significant impact on the performance of the sparsening filter + BP detector structure. At higher SNR's, the performance loss was of 2dB for the h = [0.0722, 0, 0, 0.7217, 0.6495, 0,0, 0.2165, 0, 0.0722] and 3 dB for the ITU Vehicular A channel. However, although the proposed hybrid structure showed a degraded error performance in the experiment with RF frequencies, it still showed a monotonically decreasing error behavior in terms of SNR, showing encouraging

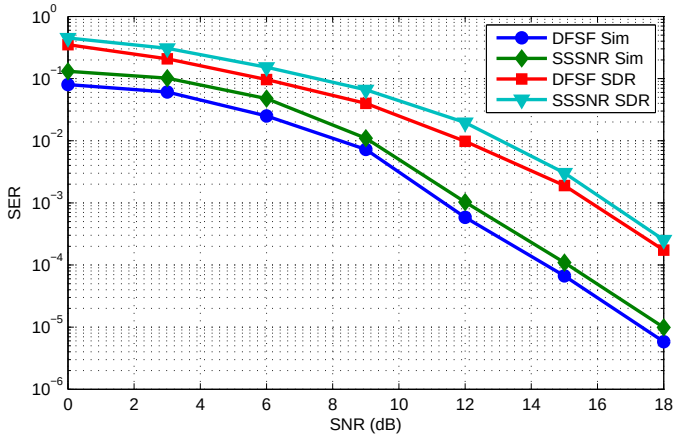


Figure 6.14: Symbol error rates for vehicular channel.

potential for real-world applications.

6.3 Chapter Summary

In this chapter we presented experiments on a SDR platform of the sparsening filters discussed in Chapters 3 and 4. The experiments with the simulated channel show that the ideal and SDR implementations error performances are similar in both multipath and vehicular channels. In the RF frequencies experiments case, the RF transmission showed to have significant impact on the error performance of the proposed BP-based receivers.

Chapter 7

Conclusions and Future Work

7.1 Concluding Remarks

In this dissertation, several contributions have been made in the area of pre-filtering design for BP-based receivers and Software-Defined radio technology. The research achievements of this thesis are the following:

- A filter design metric called the Sparse Shortening SNR and linear sparsening filter design techniques that reduce the complexity of BP-based receivers. In addition, we incorporated a solution to the noise coloring issue in the design of the sparsening filters. The proposed schemes showed a better error performance than other schemes previously proposed.

- A filter design metric based on Decision-Feedback equalizers to reduce the complexity of BP-based receivers. The proposed scheme showed a satisfactory error performance with great computational gain.

- A novel interface architecture that allows connectivity between the FMCOMMs SDR platform and the GNU Radio environment. Experiments showed the correct functionality of the proposed platform and a throughput performance characterization was also provided.

- Experimentation of BP-based receivers in a Software-Defined radio environment.

7.2 Future Work

There also exists several topics resulted from this research that could be continued.

- In the design presented in 4.3, the tentative decisions that are fed back in the sparsening filters structure are executed by a simple slicer. A next step is to replace the tentative decisions in the feedback filter with more reliable decisions output by the BP detector.
- Both schemes proposed in 3.2.2 and 4.3 are static. An interesting next step is to consider fractionally-spaced or adaptive versions of the filter design.
- Analog Devices developed an updated version of the libraries used to connect to the FMCOMMs boards. Future work should include incorporating the new library to the GNU Radio blocks and get the blocks incorporated to the GNU Radio Tree.
- The utilization of channel emulators for experiments in RF frequencies would be valuable for obtaining more test results regarding the proposed sparsening filter + BP detector structure. A natural next step is using channel emulators such as PropSim for further analysis.

7.3 Peer Review Publications

The work presented in this thesis is based on the following articles:

R. G. Machado and A. M. Wyglinski., "Software-Defined Radio: Bridging the Analog-Digital Divide," *Proceedings of IEEE*, Submitted.

R. G. Machado and A. M. Wyglinski., "Experimentation of BP-based receivers in a SDR platform," *IET Electronics Letters*, to be submitted in November 2014.

R. Machado, A. Klein, and R. Martin., "Decision feedback sparsening filter design for belief-propagation detectors," in 46^{th} *Annual Conference on Information Sciences and Systems*, March, 2012.

R. Machado, A. Klein, and R. Martin, "Sparsening filter design for iterative soft-input soft-output detectors," *EURASIP Journal on Wireless Communications and Networking*, February 2012.

BIBLIOGRAPHY

[1] P. Panigrahi, "3g lte info," http://www.3glteinfo.com/tag/lte-data-rate/.

[2] G. Minden, J. Evans, L. Searl, D. DePardo, V. Petty, R. Rajbanshi, T. Newman, Q. Chen, F. Weidling, J. Guffey, D. Datla, B. Barker, M. Peck, B. Cordill, A. Wyglinski, and A. Agah, "Kuar: A flexible software-defined radio development platform," in *New Frontiers in Dynamic Spectrum Access Networks, 2007. DySPAN 2007. 2nd IEEE International Symposium on*, 2007, pp. 428–439.

[3] D. Pu and A. M. Wyglinski, *Digital Communication Systems Engineering with Software-defined Radio*. Artech House, 2013.

[4] Analog Devices INC, "AD-FMCOMMS1-EBZ User Guide," 2012, http://wiki.analog.com/resources/eval/user-guides/ad-fmcomms1-ebz.

[5] ——, "AD-FMCOMMS2-EBZ User Guide," 2013, https://wiki.analog.com/resources/eval/user-guides/ad-fmcomms2-ebz.

[6] S. Ohmori, Y. Yamao, and N. Nakajima, "The future generations of mobile communications based on broadband access technologies," *Communications Magazine, IEEE*, vol. 38, no. 12, pp. 134–142, 2000.

[7] S. Y. Hui and K.-H. Yeung, "Challenges in the migration to 4g mobile systems," *Communications Magazine, IEEE*, vol. 41, no. 12, pp. 54–59, 2003.

[8] A. Goldsmith, *Wireless Communications*. Cambridge University Press, 2005.

[9] T. Rappaport, *Wireless Communications - Principles and Practice*. Prentice Hall PTR, 2002.

[10] Qualcomm, "The 1000x data challenge," https://www.qualcomm.com/1000x.

[11] D. Tse and P. Viswanath, *Fundamentals of Wireless Communications*. Cambridge University Press, 2005.

[12] J. Proakis and M. Salehi, *Digital Communications*. McGraw-Hill Higher Education, 2008.

[13] S. Haykins, *Digital communication*. Willey India, 2010, vol. 11.

[14] R. Lucky, "A survey of the communication theory literature: 1968-1973," *IEEE Transactions on Information Theory*, vol. 19, no. 6, pp. 725–739, Nov 1973.

[15] H. Nyquist, "Certain topics in telegraph transmission theory," *Transactions of the American Institute of Electrical Engineers*, vol. 47, no. 2, pp. 617–644, 1928.

[16] I. Gerst and J. Diamond, "The elimination of intersymbol interference by input signal shaping," *Proceedings of the IRE*, vol. 49, no. 7, pp. 1195–1203, July 1961.

[17] D. Tufts, "Nyquist's problem - the joint optimization of transmitter and receiver in pulse amplitude modulation," *Proceedings of the IEEE*, vol. 53, no. 3, pp. 248–259, March 1965.

[18] T. Berger and D. Tufts, "Optimum pulse amplitude modulation–i: Transmitter-receiver design and bounds from information theory," *IEEE Transactions on Information Theory*, vol. 13, no. 2, pp. 196–208, April 1967.

[19] R. Lucky, "Automatic equalization for digital communication," *Bell System Technical Journal*, vol. 44, no. 4, pp. 547–588, 1965.

[20] ——, "Techniques for adaptive equalization of digital communication systems," *Bell System Technical Journal*, vol. 45, no. 2, pp. 255–286, 1966.

[21] B. Widrow, *Adaptive filters I: fundamentals*. Systems Theory Laboratory, Stanford Electronics Laboratories, Stanford University, 1966.

[22] M. Austin, "Decision-feedback equalization for digital communication over dispersive channels." MIT Research Laboratory of Electronics Technical Report 461, Tech. Rep., 1967.

[23] L. Bahl, J. Cocke, F. Jelinek, and J. Raviv, "Optimal decoding of linear codes for minimizing symbol error rate (corresp.)," *Information Theory, IEEE Transactions on*, vol. 20, no. 2, pp. 284–287, 1974.

[24] G. Forney, "Maximum-likelihood sequence estimation of digital sequences in the presence of intersymbol interference," *Information Theory, IEEE Transactions on*, vol. 18, no. 3, pp. 363–378, 1972.

[25] G. Colavolpe and G. Germi, "On the application of factor graphs and the sum-product algorithm to ISI channels," *IEEE Trans. Commun.*, vol. 53, no. 5, pp. 818–825, May 2005.

[26] S. Roy, T. M. Duman, and V. K. McDonald, "Error rate improvement in underwater mimo communications using sparse partial response equalization," *IEEE Journal of Oceanic Engineering*, vol. 34, no. 2, pp. 181–201, Apr. 2009.

[27] Y. Peng, X. Huang, A. G. Klein, and K. Zhang, "Design and implementation of a low-complexity symbol detector for sparse channels," *Very Large Scale Integration (VLSI) Systems, IEEE Transactions on*, vol. PP, no. 99, pp. 1–1, 2012.

[28] Y. Peng, K. Zhang, A. Klein, and X. Huang, "Design and implementation of a belief propagation detector for sparse channels," in *Application-Specific Systems, Architectures and Processors (ASAP), 2011 IEEE International Conference on*, 2011, pp. 259–262.

[29] Ettus Research LLC, "Usrp networked series," https://www.ettus.com/product/category/USRP-Networked-Series.

[30] Q. Shi, D. Taubenheim, S. Kyperountas, P. Gorday, and N. Correal, "Link maintenance protocol for cognitive radio system with ofdm phy," in *New Frontiers in Dynamic Spectrum Access Networks, 2007. DySPAN 2007. 2nd IEEE International Symposium on*, Dublin, Ireland, 2007.

[31] R. Farrell, M. Sanchez, and G. Corley, "Software-defined radio demonstrators: An example and future trends," *International Journal of Digital Multimedia Broadcasting*, 2009.

[32] Rice University WARP, "Warp:wireless open-access research platform," http://warp.rice.edu.

[33] Nutaq, "Nutaq," http://nutaq.com/en.

[34] Epiq, "Software Defined Radio Solutions for a Mobile World," http://epiqsolutions.com/.

[35] S. U. Qureshi, "Adaptive equalization," *Proceedings of the IEEE*, vol. 73, no. 9, pp. 1349–1387, 1985.

[36] A. J. Viterbi, "Error bounds for convolutional codes and an asymptotically optimum decoding algorithm," *Information Theory, IEEE Transactions on*, vol. 13, no. 2, pp. 260–269, 1967.

[37] G. D. Forney, "Maximum-likelihood sequence estimation of digital sequences in the presence of intersymbol interference," *Information Theory, IEEE Transactions on*, vol. 18, no. 3, pp. 363–378, 1972.

[38] J. K. Omura, "Optimal receiver design for convolutional codes and channels with memory via control theoretical concepts," *Information Sciences*, vol. 3, no. 3, pp. 243–266, 1971.

[39] L. Bahl, J. Cocke, F. Jelinek, and J. Raviv, "Optimal decoding of linear codes for minimizing symbol error rate (corresp.)," *IEEE Transactions on Information Theory*, vol. 20, no. 2, pp. 284–287, Mar 1974.

[40] B. Sklar, *Digital communications*. Prentice Hall NJ, 2001, vol. 2.

[41] J. Proakis, "Adaptive equalization for tdma digital mobile radio," *Vehicular Technology, IEEE Transactions on*, vol. 40, no. 2, pp. 333–341, May 1991.

[42] J. A. Bingham, *The theory and practice of modem design.* John Wiley & Sons, Inc., 1988.

[43] S. S. Haykin, *Adaptive filter theory.* Pearson Education India, 2008.

[44] B. Widrow, J. McCool, and M. Ball, "The complex lms algorithm," in *IEEE Proceedings*, vol. 63, 1975, p. 719.

[45] C. Belfiore and J. Park, J.H., "Decision feedback equalization," *Proceedings of the IEEE*, vol. 67, no. 8, pp. 1143–1156, Aug 1979.

[46] B. Widrow and S. D. Stearns, "Adaptive signal processing," *Englewood Cliffs, NJ, Prentice-Hall, Inc., 1985, 491 p.*, vol. 1, 1985.

[47] R. E. Kalman, "A new approach to linear filtering and prediction problems," *Journal of Fluids Engineering*, vol. 82, no. 1, pp. 35–45, 1960.

[48] J. Johnson, R., P. Schniter, T. Endres, J. Behm, D. Brown, and R. Casas, "Blind equalization using the constant modulus criterion: a review," *Proceedings of the IEEE*, vol. 86, no. 10, pp. 1927–1950, Oct 1998.

[49] J. Bingham, "Multicarrier modulation for data transmission: an idea whose time has come," *IEEE Communications Magazine*, vol. 28, no. 5, pp. 5–14, May 1990.

[50] R. W. Chang, "Synthesis of band-limited orthogonal signals for multichannel data transmission," *Bell System Technical Journal*, vol. 45, no. 10, pp. 1775–1796, 1966.

[51] T. Hwang, C. Yang, G. Wu, S. Li, and G. Y. Li, "Ofdm and its wireless applications: a survey," *IEEE Transactions on Vehicular Technology*, vol. 58, no. 4, pp. 1673–1694, 2009.

[52] H. Yin and S. Alamouti, "Ofdma: A broadband wireless access technology," in *Sarnoff Symposium, 2006 IEEE.* IEEE, 2006, pp. 1–4.

[53] H. G. Myung, J. Lim, and D. Goodman, "Single carrier fdma for uplink wireless transmission," *Vehicular Technology Magazine, IEEE*, vol. 1, no. 3, pp. 30–38, 2006.

[54] K. Tremellen and J. Cox, "The influence of wave-propagation on the planning of short-wave communication," *Electrical Engineers - Part IIIA: Radiocommunication, Journal of the Institution of*, vol. 94, no. 11, pp. 200–219, March 1947.

[55] F. Bond and H. Meyer, "The effect of fading on communication circuits subject to interference," *Proceedings of the IRE*, vol. 45, no. 5, pp. 636–642, May 1957.

[56] D. Bailey, "The effect of multipath distortion on the choice of operating frequencies for high-frequency communication circuits," *Antennas and Propagation, IRE Transactions on*, vol. 7, no. 4, pp. 397–404, October 1959.

[57] I. Mitola, J., "Software radios: Survey, critical evaluation and future directions," *Aerospace and Electronic Systems Magazine, IEEE*, vol. 8, no. 4, pp. 25–36, 1993.

[58] P. Hoeher and H. Lang, "Coded-8psk modem for fixed and mobile satellite services based on dsp," in *First International Workshop on Digital Signal Processing Techniques Applied to Space Communications*, 1988.

[59] R. Lackey and D. Upmal, "Speakeasy: the military software radio," *Communications Magazine, IEEE*, vol. 33, no. 5, pp. 56–61, 1995.

[60] A. M. Wyglinski, M. Nekovee, and T. Hou, *Cognitive radio communications and networks: principles and practice*. Academic Press, 2009.

[61] Vanu, "Vanu," http://www.vanu.com/.

[62] Airspan, "Airspan," http://www.airspan.com/.

[63] Etherstack, "Etherstack," http://www.etherstack.com/eu/.

[64] BEE2, "Berkeley Emulation Engine 2," http://bee2.eecs.berkeley.edu/.

[65] H. Harada, "Software defined radio prototype toward cognitive radio communication systems," in *New Frontiers in Dynamic Spectrum Access Networks, 2005. DySPAN 2005. 2005 First IEEE International Symposium on*, Nov 2005, pp. 539–547.

[66] P. Kensington, *RF and baseband Techniques for Software Defined Radio*. Artech House, 2005.

[67] H. Tsurumi and Y. Suzuki, "Broadband rf stage architecture for software-defined radio in handheld terminal applications," *IEEE Communications Magazine*, vol. 37, no. 2, pp. 90–95, Feb 1999.

[68] A. Abidi, "Evolution of a software-defined radio receiver's rf front-end," in *2006 IEEE Radio Frequency Integrated Circuits (RFIC) Symposium*, June 2006, pp. 4 pp.–.

[69] J. G. Proakis and D. G. Manolakis, *Introduction to digital signal processing*. Prentice Hall Professional Technical Reference, 1988.

[70] A. V. Oppenheim, R. W. Schafer, J. R. Buck *et al.*, *Discrete-time signal processing*. Prentice-hall Englewood Cliffs, 1989, vol. 2.

[71] IEEE, "Ieee standard for information technology– local and metropolitan area networks– specific requirements– part 11: Wireless lan medium access control (mac)and physical layer (phy) specifications amendment 5: Enhancements for higher throughput," *IEEE Std 802.11n-2009 (Amendment to IEEE Std 802.11-2007 as amended by IEEE Std 802.11k-2008, IEEE Std 802.11r-2008, IEEE Std 802.11y-2008, and IEEE Std 802.11w-2009)*, pp. 1–565, Oct 2009.

[72] Federal Communications Commission, "Issues of Software Defined Radio Implementations for Public Safety," 2014, http://transition.fcc.gov/.

[73] The Mathworks, "MATLAB - The Language of Technical Computing," 2014, http://www.mathworks.com/products/matlab/.

[74] GNU Radio, "Welcome to gnu radio!" http://gnuradio.org/.

[75] D. D. Falconer and F. R. Magee, "Adaptive channel memory truncation for maximum likelihood sequence estimation," *Bell Sys. Tech. Journal*, pp. 1541–1562, Nov. 1973.

[76] P. J. W. Melsa, R. C. Younce, and C. E. Rohrs, "Impulse response shortening for discrete multitone transceivers," *IEEE Trans. on Comm.*, vol. 44, pp. 1662–1672, Dec. 1996.

[77] M. Kallinger and A. Mertins, "Room impulse response shortening by channel shortening concepts," in *Proc. IEEE Asilomar Conf. on Signals, Systems and Comp.*, Pacific Grove, CA, Nov. 2005, pp. 898–902.

[78] G. Arslan, B. L. Evans, and S. Kiaei, "Equalization for discrete multitone receivers to maximize bit rate," *IEEE Trans. Signal Proc.*, vol. 49, no. 12, pp. 3123–3135, Dec. 2001.

[79] K. Vanbleu, G. Ysebaert, G. Cuypers, M. Moonen, and K. Van Acker, "Bitrate maximizing time-domain equalizer design for DMT-based systems," *IEEE Trans. on Comm.*, vol. 52, no. 6, pp. 871–876, Jun. 2004.

[80] G. Bauch and N. Al-Dhahir, "Reduced-complexity space-time turbo-equalization for frequency-selective MIMO channels," *IEEE Trans. Wireless Commun.*, vol. 1, no. 4, pp. 819–828, Oct. 2002.

[81] A. Gomaa and N. Al-Dhahir, "A new design framework for sparse FIR MIMO equalizers," *IEEE Trans. Commun.*, vol. 59, no. 8, pp. 2132–2140, Aug. 2011.

[82] L. Tong, G. Xu, and T. Kailath, "Blind Identification and Equalization Based on Second-Order Statistics: A Time Domain Approach," *IEEE Trans. Info. Theory*, vol. 40, no. 2, Mar. 1994.

[83] M. Kaynak, T. Duman, and E. Kurtas, "Belief propagation over frequency selective fading channels," in *Proc. IEEE Vehicular Technology Conf. (VTC'04)*, vol. 2, Sep. 2004, pp. 1367–1371.

[84] R. K. Martin, J. M. Walsh, and C. R. Johnson, Jr., "Low Complexity MIMO Blind, Adaptive Channel Shortening," *IEEE Trans. Signal Proc.*, vol. 53, no. 4, pp. 1324–1334, Apr. 2005.

[85] G. H. Golub and C. F. Van Loan, *Matrix Computations*. Baltimore, MD: The Johns Hopkins University Press, 1996.

[86] R. K. Martin, K. Vanbleu, M. Ding, G. Ysebaert, M. Milosevic, B. L. Evans, M. Moonen, and C. R. Johnson, Jr., "Unification and Evaluation of Equalization Structures and Design Algorithms for Discrete Multitone Modulation Systems," *IEEE Trans. Signal Proc.*, vol. 53, no. 10, pp. 3880–3894, Oct. 2005.

[87] L. Szczecinski, "Low-complexity search for optimal delay in linear FIR MMSE equalization," *IEEE Signal Process. Lett.*, vol. 12, no. 8, pp. 549–552, Aug. 2005.

[88] J. C. Lagarias, J. A. Reeds, M. H. Wright, and P. E. Wright, "Convergence properties of the nelder-mead simplex method in low dimensions," *SIAM Journal of Optimization*, vol. 9, pp. 112–147, 1998.

[89] "Guidelines for the evaluation of radio transmission technologies for IMT-2000," recommendation ITU-R M.1225, 1997.

[90] Y. Peng, K. Zhang, A. G. Klein, and X. Huang, "Design and implementation of a belief propagation detector for sparse channels," in *Proc. IEEE Intl. Conf. on Application-specific Systems, Architectures and Processors (ASAP'11)*, Sep. 2011.

[91] W. Lee and F. Hill, "A maximum-likelihood sequence estimator with decision-feedback equalization," *IEEE Trans. Commun.*, vol. 25, pp. 971–979, Sep. 1977.

[92] R. Kennedy, B. Anderson, and R. Bitmead, "Tight bounds on the error probabilities of decision feedback equalizers," *IEEE Trans. Commun.*, vol. 35, pp. 1022–1028, Oct. 1987.

[93] R. G. Machado, A. G. Klein, and R. K. Martin, "Sparsening filter design for iterative soft-input soft-output detectors," *EURASIP Journal on Wireless Communications and Networking*, 2012.

[94] J. Reed, *Software Radio: A Modern to Radio Engineering*. Prentice Hall PTR, 2002.

[95] Xilinx, "Welcome to zedboard.org and microzed.org," http://www.zedboard.org/.

[96] Analog Devices, "Linux industrial io subsystem," http://wiki.analog.com/software/linux/docs/iio/iio.

[97] ——, "Iio command server," http://wiki.analog.com/resources/tools-software/linux-software/iiocmdsrv.

[98] GNU Radio, "Gnu radio 3.7.1 documentation," http://gnuradio.org/doc/sphinx/wxgui/blks.html.

[99] R. Machado, A. Klein, and R. Martin, "Sparsening filter design for iterative soft-input soft-output detectors," *EURASIP Journal on Wireless Communications and Networking*, February 2012.

[100] R. Machado, A. Klein, and R. Martin., "Decision feedback sparsening filter design for belief-propagation detectors," in 46^{th} *Annual Conference on Information Sciences and Systems*, March, 2012.

[101] M. Pätzold, M. Patzold, and M. Paetzold, *Mobile fading channels*. John Wiley England, 2002.

[102] Anite, "Propsim channel emulator solutions," http://www.anite.com/propsim.